

Universität Stuttgart

Fakultät Informatik

Advisor: Prof. Dr.-Ing. U. G. Baitinger
Co-Advisor: Prof. Dr.-Ing. M. Berroth
Date of Oral Examination: June 30th 2000

Contributions to Low Energy Consumption in Digital Circuits

A Thesis Submitted to the Faculty of Informatics
at the University of Stuttgart
for the Degree of a Doctor Rerum Naturalium

by
Markus Bühler

from
Offenburg

**Institut für Parallele und Verteilte
Höchstleistungsrechner**

Breitwiesenstraße 20-22

D-70565 Stuttgart

2000

Acknowledgments

I would like to express my gratitude towards the advisor of my thesis, Prof. Dr.-Ing. Utz G. Baitinger who gave me the opportunity to pursue this Ph.D. work at his institute. His way of giving large freedom and responsibilities to the members of his group and his engagement in international projects allowed me to gather valuable experiences and to get important feedback for this work.

Furthermore, I would like to thank the co-advisor of my thesis for his time, considerations and feedback, Prof. Dr.-Ing. Manfred Berroth from the "Institut für Elektrische und Optische Nachrichtentechnik" (INT) in Stuttgart.

Moreover, I would like to thank the people who were involved in the HIPERLOGIC project.: Dr. Selim Abou-Samra from the INPG at Grénoble, Dr. Volker Dudek from the IMS in Stuttgart and, particularly my colleague Bernd Stöhr for the valuable scientific discussions that we had, often late in the night, during the last three years, that we shared our office at the IPVR-ISE.

Special thanks go to members of our system administrator group, namely Andreas Schaupp, for their availability and willingness to solve the numerous CAD-problems that arose during the years, and to my other colleagues at ISE.

I am particularly grateful to my friend and former colleague Dr. Martin Gumm from the EPFL, for proof-reading this document and for his valuable comments.

I am also grateful to the numerous students who helped me to implement my ideas into computer programs during their master theses or a semester project.

Finally, I would like to thank my wife Susanne for her patience, her mental and practical support which allowed me to fully concentrate on my scientific work, and my parents for having me enabled to pursue my studies.

Contents

Acknowledgments	3
Contents	5
List of Figures	9
List of Tables	13
List of Listings	15
Abbreviations	17
Symbols	21
1 Introduction	23
1.1 State-of-the-Art	23
1.1.1 IC Technology	23
1.1.2 Low Power Design	26
1.2 Objectives and Content of this Thesis	26
2 Background	29
2.1 Mathematical Background	29
2.1.1 Set Theory	29
2.1.2 Graph Theory	32
2.1.3 Boolean Functions	36
2.1.4 Binary Decision Diagrams	38
2.1.5 Probability Theory	43
2.2 Physical Background	50
2.2.1 CMOS Technology	50
2.2.2 Power Dissipation in CMOS Circuits	53
2.3 Delay Models	59
2.3.1 Zero Delay Model	60
2.3.2 Unit Delay Model	60
2.3.3 Real Delay Model	61
3 Power Minimization at the Circuit Level	65
3.1 A 3D-CMOS T-Gate Technology	65
3.2 Circuit Techniques	66
3.2.1 Design Constraints in Hiperlogic	66
3.2.2 State-of-the-Art in Circuit Techniques	67
3.2.3 aCPL and DPL versus static CMOS	72

3.3	A Double Pass Logic Cell Library	73
3.3.1	Library Specification	73
3.3.2	Buffering	75
3.3.3	Demonstrator Circuits	76
3.4	Simulation Results	77
3.5	Summary	80
4	Zero Delay Power Estimation in Combinational Circuits	81
4.1	Preliminaries	81
4.2	Logic Simulation	82
4.2.1	Speeding up Simulation	83
4.2.2	Input Vector Determination	84
4.3	Probabilistic Methods	87
4.3.1	Properties	87
4.3.2	State-of-the-Art in Probabilistic Simulation	88
4.4	Sequential Circuits	92
4.4.1	High Level Simulation	94
4.4.2	State Transition Graph	94
4.4.3	State Line Probabilities	95
4.5	A New Set Based Simulation Method	98
4.5.1	Basic Method	98
4.5.2	Exact Optimization	99
4.5.3	Optimization Methods	101
4.5.4	Optimization Results	102
4.5.5	Simulation Results	107
4.6	Trading Speed versus Accuracy	111
4.6.1	Grouping	111
4.6.2	Determination of Correlation Groups	113
4.6.3	Vector Recomposition	114
4.6.4	Results	115
4.7	Summary	117
5	Real Delay Power Estimation	119
5.1	Limits of the Zero Delay Model	119
5.2	Real Delay Model	120
5.2.1	Signal Representation	120
5.2.2	The Simulation Algorithm	122
5.2.3	Results	128

5.3	Application to a Bitparallel Simulator	130
5.3.1	Transition and Hazard Counting	130
5.3.2	Results	131
5.4	Optimizations	131
5.4.1	Dynamic Package Sizing	132
5.4.2	Schedule Compression	133
5.5	Summary	136
6	Probabilistic Simulation	139
6.1	Analysis of the State-of-the-Art	139
6.2	Probabilistic - Logic Simulation	140
6.2.1	Correlation Layers	140
6.2.2	The Simulation Algorithm	141
6.2.3	Probability Propagation	144
6.2.4	Logic Operations	148
6.2.5	Results	149
6.3	Covering	155
6.3.1	Problems of BDD approaches	155
6.3.2	Local BDDs as Speed - Accuracy Trade-off	156
6.3.3	The Covering Algorithm	157
6.3.4	Results	159
6.4	Summary	164
7	Conclusion	167
8	Zusammenfassung	173
8.1	Minimierung der Verlustleistung auf der Schaltkreisebene	174
8.2	Abschätzung der Verlustleistung	175

A	Hiperlogic Transistor Parameters	185
B	The Benchmark Circuits	187
C	Detailed Results	189
C.1	Power Minimization on the Circuit Level	189
C.1.1	aCPL with $V_{thp}=0V$ and $V_{thp}=0.5V$	189
C.1.2	CPL and DPL versus CMOS	190
C.1.3	32-Bit CLAs	190
C.1.4	4-Bit CLAs with Varying Transistor Sizes	191
C.2	Zero Delay Simulation	191
C.2.1	Optimization Results	191
C.2.2	Simulation Results	193
C.2.3	Signal Grouping	195
C.3	Real Delay Power Estimation	198
C.3.1	ASSeT versus Synopsys	198
C.3.2	ASSeT, PAPSAS and Optimizations	199
C.4	Probabilistic Simulation	201
C.4.1	Correlation Layers	201
C.4.2	Local BDDs	204
D	Example for the Covering Algorithm	207
	References	213
	Curriculum Vitae	225

List of Figures

Figure 1-1:	Design Gap [SIA 97]	24
Figure 1-2:	The Y-chart [SIA 97]	25
Figure 2-1:	Partition and cover of the set X	32
Figure 2-2:	Graph illustration	33
Figure 2-3:	Bipartite graph	35
Figure 2-4:	OBDD and ROBDD for	39
Figure 2-5:	Shared ROBDD for and	40
Figure 2-6:	Negative Edges	42
Figure 2-7:	A stochastic digital system	47
Figure 2-8:	Two state Markoff chain	50
Figure 2-9:	Schematic and cross-section of a CMOS inverter	51
Figure 2-10:	Parasitic capacitances and channel length of a MOS transistor	51
Figure 2-11:	Inverter in SOI technology	52
Figure 2-12:	T-gate transistor	52
Figure 2-13:	3D-CMOS	53
Figure 2-14:	CMOS inverter	54
Figure 2-15:	CMOS inverter during rising inputs	55
Figure 2-16:	Leakage current	56
Figure 2-17:	pseudo-NMOS inverter	57
Figure 2-18:	CMOS buffer	58
Figure 2-19:	complete transition and glitch	59
Figure 2-20:	Circuit simulated with ZDM	60
Figure 2-21:	Circuit simulated with UDM	61
Figure 2-22:	Halfadder	61
Figure 2-23:	Slew rate	62
Figure 2-24:	Load and slew rate dependence of gate delays	63
Figure 2-25:	Buffer with transport and inertial delay model	63
Figure 2-26:	AND gate	64
Figure 3-1:	The Hiperlogic-Technology	66
Figure 3-2:	XOR in static CMOS	68
Figure 3-3:	XOR in dynamic DOMINO logic	69
Figure 3-4:	XOR in CPL	70
Figure 3-5:	XOR in alternative CPL	70
Figure 3-6:	Energy and delay for different V_{thp}	71
Figure 3-7:	XOR in DPL	71
Figure 3-8:	Comparison of CMOS, CPL, and DPL	72
Figure 3-9:	The DPL base cell	74
Figure 3-10:	XOR in Low Power DPL	74
Figure 3-11:	DPL D-Latch	75

Figure 3-12:	Nand2 in DPL with output and input buffers	75
Figure 3-13:	Transistor counts of the 32-bit CLAs	77
Figure 3-14:	Worst case delays of the 32-bit CLAs	77
Figure 3-15:	Energy consumption per operation of the 32-bit CLAs	78
Figure 3-16:	DPL versus static CMOS with varying driver sizes	79
Figure 4-1:	General combinational circuit	82
Figure 4-2:	Bitparallel AND	84
Figure 4-3:	XOR circuit	89
Figure 4-4:	ROBDD for	90
Figure 4-5:	Part of an XOR-BDD	91
Figure 4-6:	Synchronous sequential circuit	93
Figure 4-7:	Modeling correlations	93
Figure 4-8:	A state transition graph	95
Figure 4-9:	Feedback loop	96
Figure 4-10:	k-unrolled network	97
Figure 4-11:	A 2-expanded network	97
Figure 4-12:	Input modeling FSM	98
Figure 4-13:	Venn diagrams of two signals a and b	98
Figure 4-14:	Signal representation with blocks	99
Figure 4-15:	Arbitrary physical signals	100
Figure 4-16:	Rearranged Input Sequence	100
Figure 4-17:	Gray code arrangement	101
Figure 4-18:	Optimization for uncorrelated vectors	103
Figure 4-19:	Optimization for FSM-like vectors	104
Figure 4-20:	Optimization for spatially correlated vectors	104
Figure 4-21:	Optimization for low active vectors	105
Figure 4-22:	Average optimization times	106
Figure 4-23:	Simulation times for uncorrelated vectors	107
Figure 4-24:	Simulation times for FSM-like vectors	108
Figure 4-25:	Simulation times for correlated vectors	108
Figure 4-26:	Simulation times for low active signals	109
Figure 4-27:	Block optimization for sequential circuits	110
Figure 4-28:	Simulation times of sequential circuits	110
Figure 4-29:	Two-address instruction	111
Figure 4-30:	Correlation groups	112
Figure 4-31:	Partial vectors	112
Figure 4-32:	Separately optimized partial vectors	112
Figure 4-33:	Input patterns after recomposition	114
Figure 4-34:	Speedup with signal grouping	116
Figure 4-35:	Global error with signal grouping	116
Figure 4-36:	Local error with signal grouping	117
Figure 5-1:	Circuit simulated with ZDM and UDM	120
Figure 5-2:	RDM signal	121
Figure 5-3:	RDM signal in CO-time notation	121

Figure 5-4:	AND gate with input schedules	123
Figure 5-5:	The input waveforms of figure 5-4	123
Figure 5-6:	Speedup of set based over logic simulation	129
Figure 5-7:	Hazard rate	129
Figure 5-8:	Speedup with load dependence	130
Figure 5-9:	Speedup of bitparallel over bitwise simulation	131
Figure 5-10:	Speedup of bitparallel simulation with DPS over bitwise simulation	133
Figure 5-11:	Speedup of bitparallel simulation with DPS over bitparallel simulation without DPS	133
Figure 5-12:	Packages sizes for DPS	134
Figure 5-13:	Speedup over bitwise simulation	134
Figure 5-14:	Speedup of schedule compression and DPS over DPS only	135
Figure 5-15:	Global error	135
Figure 5-16:	Local error	136
Figure 6-1:	Two representations for	141
Figure 6-2:	BDD of an AND-gate	142
Figure 6-3:	Arbitrary BDD	144
Figure 6-4:	Example BDD	146
Figure 6-5:	Simulation times for random and optimized variable orders	150
Figure 6-6:	Number of BDD nodes for random and optimized variable orders	150
Figure 6-7:	Deviations between logic and probabilistic simulation	151
Figure 6-8:	CPU and memory requirements of Najm's approximation versus XOR-BDDs	152
Figure 6-9:	Global error of Najm's approximation	153
Figure 6-10:	Global deviations for probabilistic and probabilistic-logic simulation	153
Figure 6-11:	Local deviations for probabilistic and probabilistic-logic simulation	154
Figure 6-12:	Global and local deviation if correlations are neglected . . .	154
Figure 6-13:	Global and local deviation if correlations are taken into account	155
Figure 6-14:	Circuit with two partitions	156
Figure 6-15:	Circuit with 3 input blocks for gate a and c	157
Figure 6-16:	Circuit for which a local BDD is searched	157
Figure 6-17:	Increase of CPU and memory requirements with block size	161
Figure 6-18:	Influence of block size to estimation accuracy	162
Figure 6-19:	Global deviations for uncorrelated and correlated PIs with local BDDs	162
Figure 6-20:	Local deviations for uncorrelated and correlated PIs with global BDDs	163
Figure 6-21:	Global and local deviation from logic simulation	163
Figure 6-22:	Speedup with local BDDs over pure probabilistic-logic	

	simulation	164
Figure 7-1:	The complete power estimation system	171
Figure 8-1:	Die Hiperlogic-Technologie	174
Figure 8-2:	Mengenbasierte Signaldarstellung	176
Figure 8-3:	RDM-Signal mit Schedule	178
Figure 8-4:	BDD für ohne und mit Korrelationsschicht	180
Figure 8-5:	Das komplette System zur Ermittlung der Schaltaktivität . .	183
Figure D-1:	Circuit to find a local BDD for	207
Figure D-2:	Gates b and c have been added	208
Figure D-3:	Gate d has been added	209
Figure D-4:	Gates f and g have been added	210
Figure D-5:	Gates f and g have been added	210
Figure D-6:	Gate h has been added	211
Figure D-7:	Final block	212

List of Tables

Table 1-1:	Trends in microelectronics	23
Table 2-1:	Boolean functions of two arguments and their ITE form	41
Table 2-2:	Common delay and slew rate approximations	62
Table 4-1:	Rules for probability propagation for independent signals	87
Table 4-2:	Rules for probability computation for disjoint signals	88
Table 4-3:	Logic, set, and probabilistic operations	99
Table 4-4:	Parameters of the benchmark circuits	107
Table 4-5:	Parameters of the benchmark circuits	109
Table 5-1:	Array representation of signal waveforms	122
Table 5-2:	Preliminary schedule of y after two logic operations	125
Table 5-3:	Final schedule of y after two logic operations	126
Table 5-4:	Preliminary schedule of y after three logic operations	127
Table 5-5:	Final schedule of y after three logic operations	127
Table 5-6:	Final output schedule after processing of all input events	128
Table 6-1:	Path probabilities and possible input vectors of a 2-input circuit	142
Table 6-2:	Vector probability of a 2-input AND-gate	142
Table 6-3:	Pattern numbers and probabilities	146
Table 6-4:	Variable probabilities	146
Table 6-5:	Pattern numbers and probabilities for the single variable x_0	148
Table 6-6:	Properties of possible extensions for initial block $\{a\}$	158
Table 6-7:	Sort criteria for the cost list	159
Table 8-1:	Trends in der Mikroelektronik	173
Table 8-2:	Zusammenhang zwischen logischen Operationen und Mengenoperationen	177
Table B-1:	Statistics of the ISCAS-85 benchmark circuits	187
Table B-2:	Statistics of the ISCAS-89 benchmark circuits	188
Table B-3:	Statistics of the LGSYNTH benchmark circuits	188
Table C-1:	CPL with $V_{thp}=0V$ and $V_{thp}=0.5V$ for the pass transistors	189
Table C-2:	CPL and DPL versus CMOS	190
Table C-3:	Results from the 32-bit CLAs	190
Table C-4:	4-bit CLAs with varying transistor sizes	191
Table C-5:	Optimization times and results for uncorrelated PIs	191
Table C-6:	Optimization times and results for FSM-like PIs	192
Table C-7:	Optimization times and results for spatially correlated PIs	192
Table C-8:	Optimization times and results for low active PIs	192
Table C-9:	Simulation times for uncorrelated PIs	193
Table C-10:	Simulation times for FSM-like PIs	193
Table C-11:	Simulation times for spatially correlated PIs	194
Table C-12:	Simulation times for low active PIs	194
Table C-13:	Simulation times for patterns of sequential circuits	195

Table C-14:	Combinational circuits with different group sizes	195
Table C-15:	The logic parts of some FSMs with different group sizes	197
Table C-16:	Simulation times of ASSeT and Synopsys	198
Table C-17:	Hazard rates and runtimes of real delay simulations with random patterns	199
Table C-18:	Error rates of schedule compression for different d	200
Table C-19:	Hazard rates and runtimes of real delay simulations with RTL generated patterns	200
Table C-20:	Probabilistic compared to logic simulation	201
Table C-21:	Najm's approximation versus XOR-BDDs	203
Table C-22:	XOR-BDDs with and without correlation layers	204
Table C-23:	Resource requirements and accuracy for different block sizes	204
Table C-24:	Local BDDs for combinational circuits and FSMs	205
Table D-1:	Cost list of gates b and c	207
Table D-2:	Cost list of gates d, e and g	208
Table D-3:	Cost list after eliminating gate d from all entries	209
Table D-4:	Cost list of gates e, f and g	209
Table D-5:	Cost list after gates f and g have been removed	210
Table D-6:	Cost list of gates h, i and g	211
Table D-7:	Cost list after gate h has been removed from all entries	212

List of Listings

Listing 4-1:	The grouping algorithm	113
Listing 6-1:	The covering algorithm	160

Abbreviations

3D	three D imensional
aCPL	alternative C omplementary P ass Transistor L ogic
ASSeT	A ctivity S imulator based on S et T heory
BDD	B inary D ecision D iagram
CVSL	C ascode V oltage S witch L ogic
CLA	C arry- L ook-ahead A dder
CMOS	C omplementary M etal O xide S ilicon
CPL	C omplementary P ass transistor L ogic
CRF	C orrelations due to R econvergent F anout
CSP	C anonical S um of P roducts
DC	D on't C are
DCVSPG	D ifferential C ascode V oltage S witch with P ass G ate
DPL	D ouble P ass L ogic
EDA	E lectronic D esign A utomation
FDD	F unctional D ecision D iagram
fig.	F igure
FSM	F inite S tate M achine

Hiperlogic	H igh P erformance at L ow power L ogic
IC	I ntegrated C ircuit
Iff	I f and only i f
IMFSM	I nput M odeling F inite S tate M achine
IMS	I nstitute for M icroelectronics S tuttgart
IPVR	I nstitute of P arallel and D istributed H igh P erformance S ystems
ISE	D epartment of I ntegrated S ystems E ngineering
LAN	L ocal A rea N etwork
LUT	L ookup T able
OBDD	O rdered B inary D ecision D igram
OKFDD	O rdered K ronecker F unctional D ecision D igram
PCI	P eripheral C omponent I nterconnect
PI	P rimary I nput
PO	P rimary O utput
RDM	R eal D elay M odel
ROBDD	R educed O rdered B inary D ecision D igram
RTL	R egister T ransfer L evel
SF(X)	S ubfunction of node X
SFP(X)	S ubfunction P robability of node X

SI	S econdary I nput
SIA	S emiconductor I ndustry A ssociation
SOI	S ilicon- o n- I nsulator
SRPL	S wing R estored P ass-transistor L ogic
SSS	S trict S ense S tationary
STG	S tate T ransition G raph
t_d	d elay (t ime)
UDM	U nit D elay M odel
WSS	W ide S ense S tationary
ZDM	Z ero D elay M odel

Symbols

α	average toggle rate
σ	standard deviation
\in	element of
\emptyset	empty set
\notin	not element of
\subset	subset
\forall	for any
\exists	there exists
$\lfloor a \rfloor$	the greatest integer number $\leq a$
$\lceil a \rceil$	the smallest integer number $\geq a$
$*x$	a reference to variable x
\longrightarrow	0-edge in a BDD
\longrightarrow	1-edge in a BDD
\downarrow	ground
\top	supply voltage
B	a binary set B

div	$a \text{ div } b = \lfloor a/b \rfloor \cdot b$
f, f_{CLK}	operating frequency of a circuit
k	Boltzmann constant, $k = 1.380662 \times 10^{-23} \text{ J/K}$
mod	$a \text{ mod } b = a - \lfloor a/b \rfloor \cdot b$
GND	ground
R	the set of the real numbers
$R.x$	Element x of record R
V_{dd}	supply voltage
V_{th}	threshold voltage
\hat{x}	an estimate of the parameter x
<u>X</u>	matrix with m rows X_1, X_2, \dots, X_m
<u>X</u>^T	the transposed of matrix <u>X</u>
Z	the set of the integer numbers

1.1 State-of-the-Art

1.1.1 IC Technology

For more than 20 years now, progress in IC technology has been driven by the ever decreasing feature size and thus increasing complexity of circuits that can be integrated onto a single chip. This development was predicted by Gordon Moore of Intel Corporation. It has become known as *Moore's Law* [Moor 75]: “*The number of transistors per chip and thus design complexity doubles every 18 months*”.

Table 1-1 depicts the SIA-forecast for trends in microelectronics for the next 15 years [SIA 97].

Year	1997	2001	2006	2012
minimum feature size [nm]	250	150	100	50
maximum clock frequency [MHz]	750	1500	3500	10.000
transistors/chip	11M	40M	200M	1.4G
Supply voltage [V]	1.8-2.5	1.2-1.5	0.9-1.2	0.5-0.6
power dissipation (stationary devices) [W]	70	110	160	175
power dissipation (mobile devices) [W]	1.2	1.7	2.4	3.2

Table 1-1: Trends in microelectronics

With the advent of deep submicron technology, the major design limitations and therefore the optimization objectives during the design process have changed substantially. Until the early 1990s, most designs were area and speed limited. Consequently, the focus of the optimization efforts was put on these two points. But today, as more than 10 million transistors are available on a single chip, area is not a major concern anymore. It has been replaced by the question how to cope with the complexity of a multi-million transistor design. This fact has become popular as the “*Design Gap*”. It is not the silicon technology that represents the bottleneck in microelectronics, but the productivity of the designer and the tools that sup-

port the design process. While the technologically feasible complexity is growing exponentially over time, design productivity, measured in transistors per designer and day for instance, grows only little more than linearly (figure 1-1).

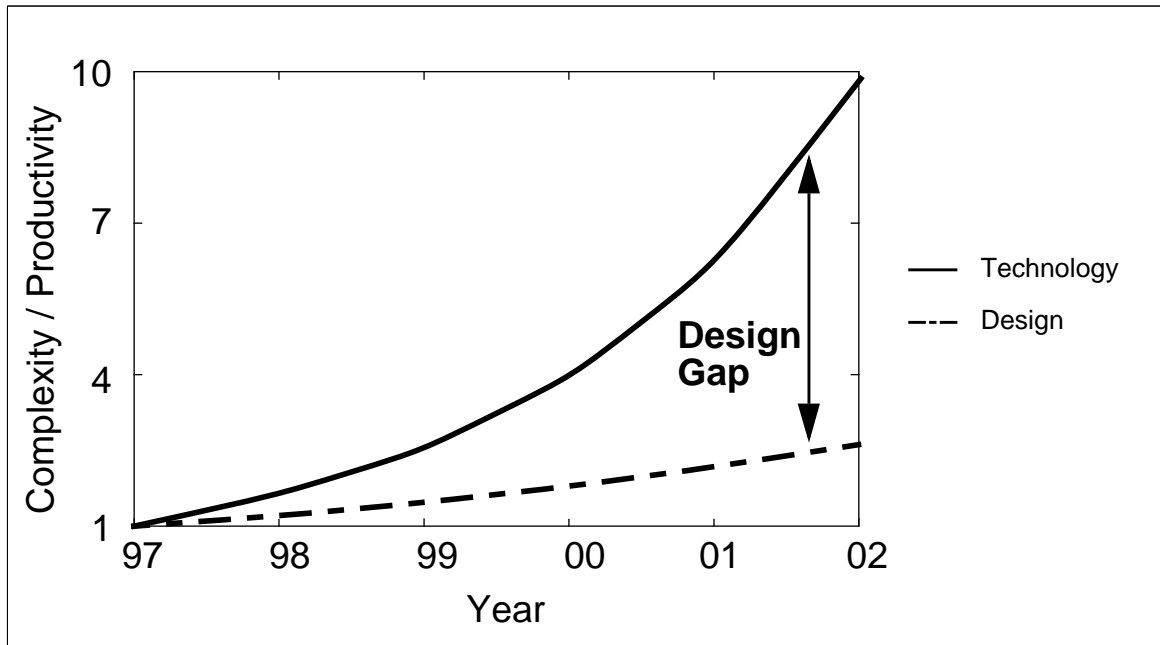


Figure 1-1: Design Gap [SIA 97]

A widely accepted means to deal with complexity is the *top down design* style. Starting from a highly abstract specification of the system to be developed, the design team approaches the final solution through step by step refinement of the original specification. The whole top down design process can be depicted with the Y-chart [Gajs 83]. It represents the 3-dimensional design space. Each axis depicts a specific view on the design: behavior, structure, and geometry. The concentric circles denote the different design levels. The level of abstraction decreases from the outer levels to the inner ones. The center where the three axis intersect, represents the design goal: the final layout data of the chip. The designer usually starts with a system specification on the behavioral axis. The intersections of the design levels and the axis denote design data, whereas the lines between the intersections represent transformations. There exist several possible ways from the system specification to the center of the Y-chart, which depend on the design style, e.g. full custom, standard cell, or FPGA based.

All the design steps can be considered either as synthesis steps which direct towards the center, as analysis steps which direct away from the center, or as transformations from one view to another [Retti 96]. Of course, in order to close the design gap, it is highly desirable to automate as many steps of the design cycle as possible. On RTL (Register Transfer Level) and below, most synthesis, transformation, and analysis tasks are already widely automated and commercial tools like logic synthesizers, VHDL simulators, or place and route tools are

available. However, for today's multimillion transistor designs, similar tools for higher levels are indispensable. Although much research has been done in this field and is still ongoing, only analysis tools like simulators have reached commercial maturity so far.

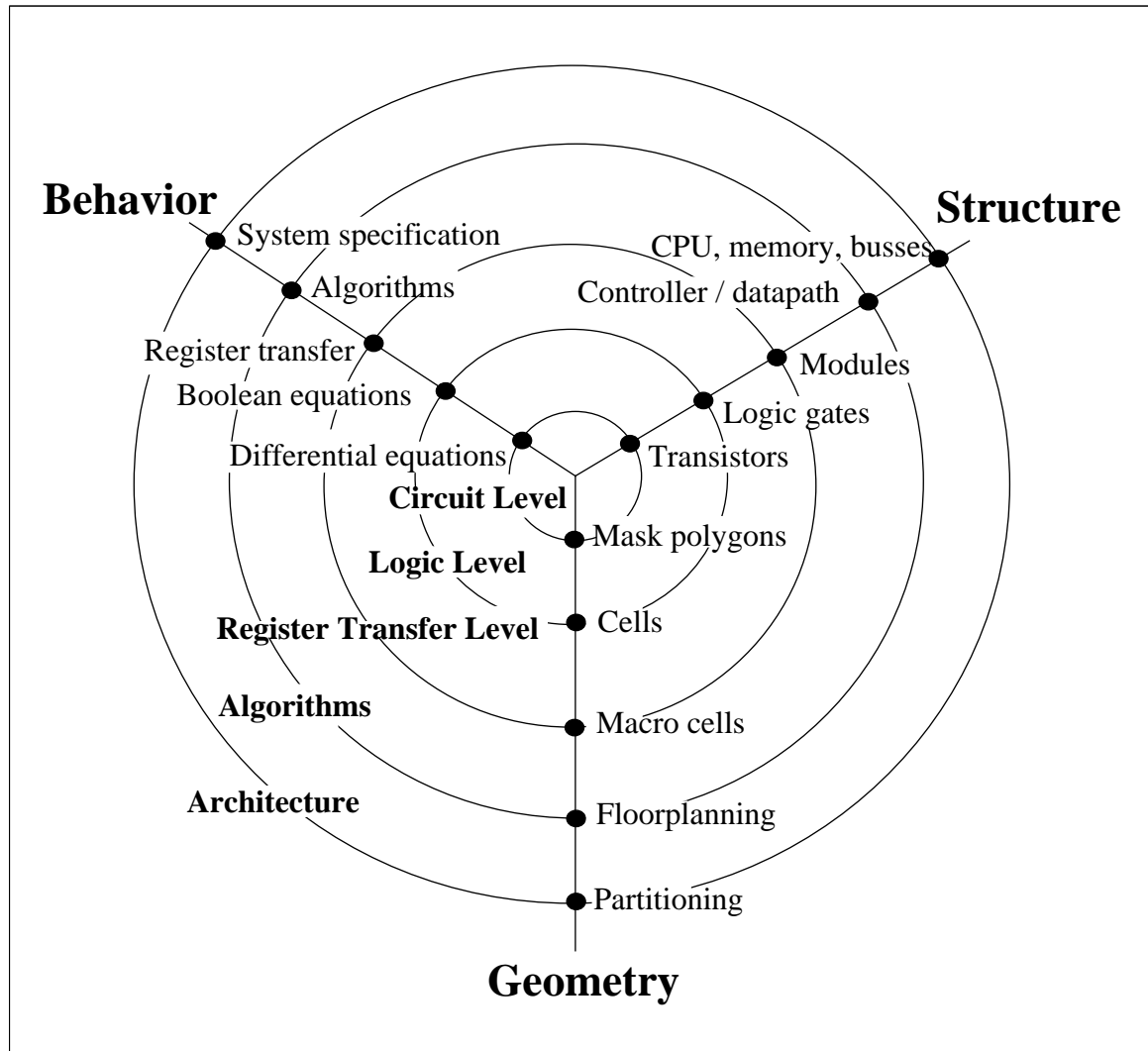


Figure 1-2: The Y-chart [SIA 97]

But growing complexity and decreasing feature sizes also influence other areas in IC technology [SIA 97]. Test becomes more and more costly, since the number of test vectors which are required for a certain test coverage grows exponentially with circuit size. The wiring delay all of a sudden predominates the gate delay since the transistors and the gate capacitances have become so small. Last but not least, the growing number of transistors per chip and increasing operation frequencies result in high power consumption. This limits the battery lives of mobile devices but it also influences stationary machines through increasing cooling and packaging costs. Thus, low power design has regained lots of interest as the third design paradigm, beside area and speed.

Consequently, new tools and methodologies for low power design are required [SIA 97]. Feasibility of systems on a chip crucially depends on new analysis and simulation algorithms. Parallelism must be exploited wherever possible. Novel design tools that support and exploit hierarchical design styles must become available in order to increase the designers' productivity.

1.1.2 Low Power Design

From all the problems that were outlined in the previous section, the focus is put on low power design in this section. The term *low power design* includes all efforts that are made to optimize circuits for lower power consumption. It can be divided into two subcategories:

- **Power Estimation:** techniques to estimate the power consumption of digital circuits. It is highly desirable to develop estimation techniques for as high as possible design levels. There are two properties that make power estimation harder than e.g. delay estimation. Power consumption does not only depend on static circuit properties like structure and supply voltage, but also on the input patterns that are applied to the circuit. Hence, only statistical statements can be made. Furthermore, on higher design levels, it is not possible to give an absolute estimate of the average power consumption in Watt. More abstract metrics like signal activity, are used instead in order to compare different designs that implement the same functionality.
- **Power Optimization and Low Power Synthesis.** Circuits optimized for high speed and low area consumption are in most cases not the optimum implementation from a power consumption point of view. Therefore, dedicated methods and synthesis tools for the design of low power circuits must be developed for all levels of the design hierarchy.

Today, commercial tools for power estimation and synthesis are limited to the transistor and gate level. The latter are still on a very rudimentary level, though. EPIC software¹ offers a quite advanced set of tools for power analysis on layout and transistor level [Huan 95]. SYNOPSIS has extended its gate level synthesis tool *design analyzer* [SyLi 96] by some basic power estimation and low power synthesis capabilities. There are hardly any commercial tools available for RT level and above.

1.2 Objectives and Content of this Thesis

In the previous section, today's most urgent problems in IC technology were demonstrated. In a single dissertation it is not possible to cover them all. Even the specific areas like analysis, test, and synthesis can be divided into a multitude of specific, unsolved aspects. Each of them represents a field of research for itself with its specific mathematical and physical terms, models, and methodologies.

1. EPIC software has just recently merged with Synopsys.

This thesis is therefore restricted to a subdomain of the low power design. The main focus is put on the development of novel power estimation techniques on gate level. But since low power design cannot be restricted to a single level, some very specific design aspects on circuit level will be touched too.

In the power estimation domain, two types of tools can be observed. They are either based on explicit simulation or on probabilistic methods. Both have their specific advantages and drawbacks. But for any tool, two mostly contradictory requirements exist: speed and accuracy. For both estimation methods novel approaches are proposed in this work. By combination with advanced methods from the existing literature significant improvement of the state of the art could be obtained. It will be shown in a later chapter that power estimation on gate level can be reduced to the estimation of signal activities.

In contrast to the top down design method mentioned in section 1.1.1, this thesis follows a bottom up strategy. This strategy has proven to be superior from a didactical point of view. The rest of this work is therefore organized as follows.

Chapter 2 reflects the mathematical and physical background that will be used in the later investigations. Graph theory, Boolean functions, and probability theory are introduced as well as CMOS technology. The origin of power dissipation and different time models are explained in detail because they are closely related to any power estimation method.

Chapter 3 deals with a new technology that is just being developed at the Institute for Microelectronics Stuttgart (IMS) [Hipe 95]. The question of suitable circuit techniques in terms of high performance and low power consumption is investigated under the specific constraints of this new technology.

Chapters 4 to 6 represent the core of this thesis. They introduce novel ideas in power estimation on logic level. Chapter 4 outlines the state-of-the-art in probabilistic and pattern simulation based power estimation before it presents a new, set theory based method. Since this method was first limited to the zero delay model, chapter 5 describes an approach that extends it to any more sophisticated delay model. This approach could also be applied to another, very efficient simulation method that was published in the literature and has also been restricted to the zero delay model so far. Furthermore, the effect of simplifications in the time model for the estimation result is investigated. Chapter 6 deals with the problem of improving accuracy as well as runtime for probabilistic simulation methods. Through a combination of traditional, BDD based probabilistic methods and pattern simulation, it was succeeded in accurately handling correlations between the primary input signals of a circuit. They must be neglected in most probabilistic simulators today. Significant speed up was traded for a slight loss in accuracy by a dynamical partition algorithm which is also presented in chapter 6. All statements in the chapters 3 to 6 are supported by extensive simulation results.

Chapter 7 concludes this work with a summary of the new methods which have been developed in this work and a final discussion of the results. It also gives some hints, where more extensive investigations are promising and could be of interest.

Finally, chapter 8 gives an extended summary of this thesis in German.

This chapter compactly presents the background which is necessary to understand the rest of this work. Due to the nature of the problems in design and analysis processes, the background is divided into two domains: mathematics and physics.

Mathematics provide the abstract means for data modeling and processing. The mathematical section starts with a short introduction into set and graph theory, as both are used for several purposes in this work. Another basic domain are the Boolean functions. They are formally defined and their specific properties and theorems are briefly revisited. Boolean functions can be efficiently represented and manipulated using Binary Decision Diagrams (BDD). As they will be applied for probabilistic signal activity estimation later in this work, their basic concepts as well as the most fundamental algorithms for BDD manipulation are outlined here. Finally, probability theory is explained with a special emphasis on binary variables. Terms like signal and switching probability or correlation and statistical dependence are defined.

The physical background is divided into two subsections. The first deals with CMOS technology as the basis for most digital circuit implementations today. Common bulk technology is explained as well as new trends like *Silicon-on-Insulator (SOI)*, *T-gate transistor*, and *3D-integration*. The second part takes a closer look on the origin of power dissipation in CMOS circuits. It will become obvious that power dissipation divides into three classes of different importance: dissipation due to leakage, short circuit and capacitive currents.

Time models belong semantically to the physical background. Because of their importance for the simulation and power estimation process, they were reserved for a separate, last section.

2.1 Mathematical Background

This section can only have a refreshing character. For deeper insight please consult the references given at the beginning of each subsection.

2.1.1 Set Theory

Set theory provides a formal means to handle sets, elements of sets, and relations between them. The definitions are in close accordance with [Lane 67], [Marc 66], and [Lipp 92].

The term *set* is ubiquitous in mathematics. However, it is usually left undefined. A set consists of mathematical objects which are called *elements*. The set is the totality of these elements [Marc 66]. If an element x belongs to a set S we say x is *element of* S or $x \in S$. If an element x does not belong to the set S we write: $x \notin S$.

Definition 2-1: Empty Set \emptyset

The *empty set* is the set without any element. It is denoted by \emptyset .

Definition 2-2: Subset

If every element x of a set X also belongs to a set Y , then X is a *subset* of Y and we write: $X \subseteq Y$.

If $X \subseteq Y$ and $Y \subseteq X$, then X and Y are *equal*: $X = Y$.

If $X \subseteq Y$ but $X \neq Y$ then X is a *proper subset* of Y and we write $X \subset Y$.

Definition 2-3: Union and Intersection

The *union* of two sets X and Y is the set which contains all elements that belong either to X , to Y , or to both:

$$X \cup Y = \{x | x \in X \vee x \in Y\}$$

The *intersection* of X and Y is the set of all elements common to X and Y :

$$X \cap Y = \{x | x \in X \wedge x \in Y\}$$

Definition 2-4: Complement

The *complement* \bar{X} of a set X is the set which contains all elements that do not belong to X :

$$\bar{X} = \{x | x \notin X\}$$

Definition 2-5: Ordered Pair

An *ordered pair* (x,y) denotes a set of two elements one of which is designated as the first element of the pair. Thus $(x_1, y_1) = (x_2, y_2)$ if and only if (*iff*) $x_1 = x_2$ and $y_1 = y_2$.

Definition 2-6: Cartesian Product

The *Cartesian product* of X and Y is the set of all ordered pairs (x,y) with $x \in X$ and $y \in Y$:

$$X \times Y = \{(x, y) | x \in X \wedge y \in Y\}$$

Definition 2-7: Binary Relation

For any two sets X and Y , a subset $R \subset X \times Y$ is called a *binary relation* on X to Y , or a *relation* between X and Y . Then $(x, y) \in R$ is usually written as xRy and reads: “ x bears relation R to y ”. $(x,y) \notin R$ is denoted as $x \bar{R} y$.

Definition 2-8: Reflexivity

A relation R is *reflexive* if xRx for all $x \in X$. If xRx for no $x \in X$, then R is called *antireflexive*.

Definition 2-9: Symmetry

If for all pairs (x,y) xRy implies yRx , then R is called *symmetric*. If there are some (x,y) with xRy and $y \not R x$, then R is *asymmetric* and if xRy always implies $y \not R x$, then R is *antisymmetric*.

Definition 2-10: Transitivity

If for all elements $x,y,z \in X$ xRy and yRz implies xRz then R is *transitive*.

Definition 2-11: Equivalence Relation

A binary relation R on set X is called an *equivalence relation* if R is reflexive, symmetric, and transitive. For an equivalence relation R we usually write $x \sim y$ instead of xRy .

Definition 2-12: Partial Order

A binary relation \leq is called a *partial order* if it is reflexive, transitive, and antisymmetric.

Definition 2-13: Power Set

Given a set X , the set of all subsets of X (including \emptyset and X itself) is called the *power set of X* and is denoted by $P(X)$.

Definition 2-14: Partition

Let X be a non-empty set. Any subset Π of $P(X)$ which does not contain \emptyset is called a *partition* if each element of X belongs to one and only one of the elements of Π .

Definition 2-15: Cover

Let X be a non-empty set. Any subset C of $P(X)$ which does not contain \emptyset is called a *cover* if each element of X belongs to at least one of the elements of C .

Definition 2-16: Block

The subsets of a cover or a partition are called *blocks*.

Note that the definitions for a partition and cover are quite similar, but whereas a partition requires that each element of X belongs to exactly one block of the partition, the requirements for a cover are weaker: all elements of X must be in at least one block of the cover. Thus, any partition is a cover as well, but not vice versa. Figure 2-1 shows an example of a partition and a cover. On the right hand side, the hatched regions denote the overlapping portions of the adjacent subsets b , c , and b , e respectively.

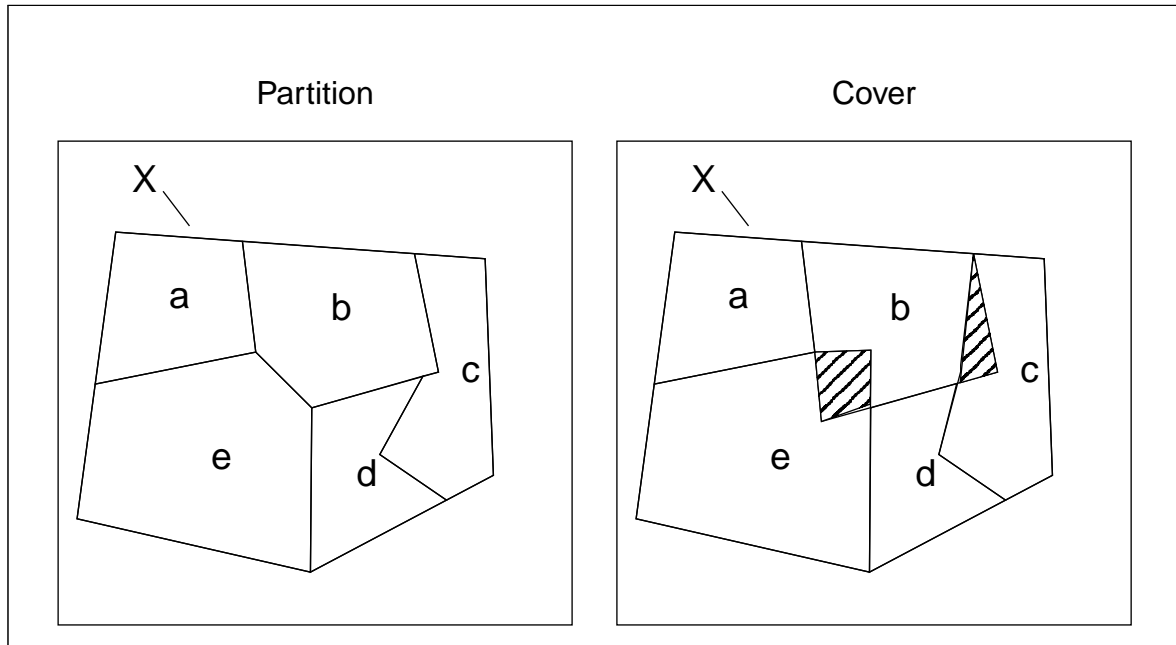


Figure 2-1: Partition and cover of the set X

2.1.2 Graph Theory

Graphs are one of the basic means for modeling and manipulation of data and relations between data in electronic design automation (EDA). They are commonly used e.g. for netlist or Boolean function representation, as well as for solving floorplanning and place and route problems. The definitions in this paragraph follow those in [Tutt 84], [DeMi 94], and [Boll 98].

Definition 2-17: Graph

A *graph* $G(V,E)$ is an ordered pair (V, E) . V is a set, its elements v_i are called *vertices*. E is a binary relation on V , its elements e_j are called *edges*. The edges are pairs of vertices, also called *end-points* of the edge.

An edge is *incident* to a vertex if this vertex is one of the end-points of the edge.

Definition 2-18: Directed and Undirected Graph

A graph is a *directed graph* if the edges are ordered pairs, otherwise it is *undirected*. Undirected edges are denoted by $\{v_i, v_j\}$, directed edges by (v_i, v_j) where v_i is the *head* and v_j is the *tail* of the edge.

In this work, only *finite graphs* shall be considered, where both sets, V and E , are finite sets. Other graphs are called *infinite*.

Graphs are commonly illustrated by diagrams. The vertices are depicted as dots or circles, which are labeled by the name of the vertex. An edge between two vertices is represented by a line which connects the two vertices (figure 2-2a). Directed edges are marked by an arrow at the end of the line that touches the second vertex of the edge (figure 2-2b). A graph is said to be *planar*, if it has a diagram on a plane surface such that no two edges cross. The two graphs in figure 2-2 are planar.

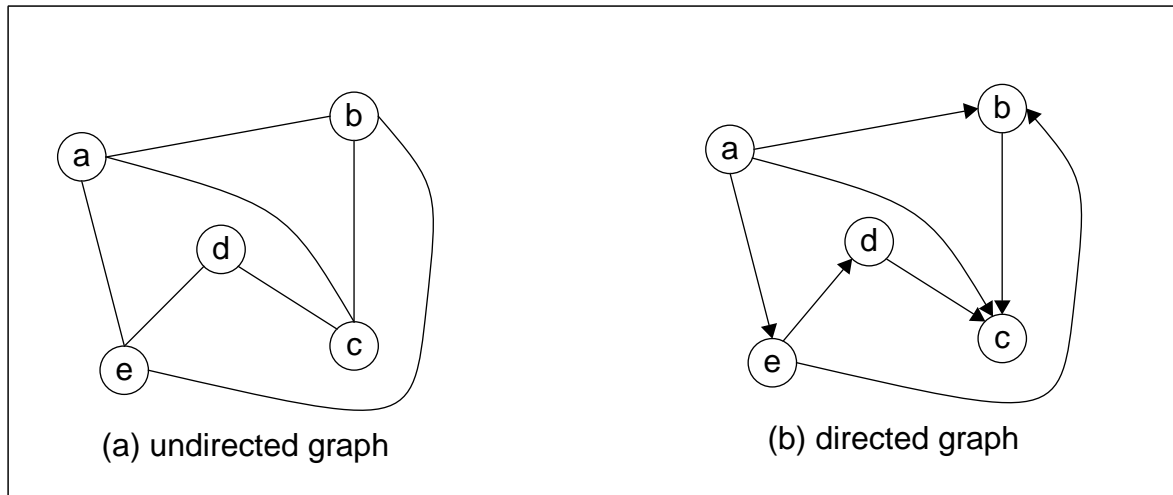


Figure 2-2: Graph illustration

The *degree* of a vertex is the number of edges incident to it.

Besides their representation by a vertex set and an edge set, a graph $G(V,E)$ can also be represented by an *incidence matrix* which is defined as follows.

Definition 2-19: Incidence Matrix

The *incidence matrix* $B(G)=(b_{ij})$ of a graph $G(V,E)$ is the $n \times m$ matrix defined by:

$$b_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the head of } e_j \\ -1 & \text{if } v_i \text{ is the tail of } e_j \\ 0 & \text{otherwise} \end{cases}$$

Thus, n corresponds to the number of vertices and m to the number of edges of $G(V,E)$. An incidence matrix can also be defined for undirected graphs. In that case, $b_{ij}=1$ if v_i is element of the edge e_j .

Both, directed and undirected graphs can be *weighted*. Weights can be attached to vertices and/or edges. In that case, a graph is *vertex* and/or *edge weighted*.

Undirected Graphs

Definition 2-20: Isomorphic

Two graphs are said to be *isomorphic* if there is a one to one correspondence between their vertex sets that preserves adjacency. Thus $G_1(V_1, E_1)$ is *isomorphic* to $G_2(V_2, E_2)$ if there is a bijection $\emptyset : V_1 \rightarrow V_2$ such that $\{x, y\} \in E_1$ iff $\{\emptyset(x), \emptyset(y)\} \in E_2$.

Definition 2-21: Adjacency, Loop, Simple Graph, Multi-Graph

Two vertices are *adjacent* to each other if there is an edge incident to both vertices. An edge with two identical end-points is called a *loop*. A graph without any loop is a *simple* or a *strict graph*. Otherwise it is a *multi-graph*.

A second important matrix that can be defined on graphs is the *adjacency matrix*.

Definition 2-22: Adjacency Matrix

Given graph $G(V, E)$, its *adjacency matrix* $A(G) = (a_{ij})$ is defined by:

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

The adjacency matrix for an undirected graph is always symmetric.

Note, definition 2-22 can be applied to directed graphs if the brackets $\{ \}$ are replaced by $()$.

Definition 2-23: Walk, Trail, Path, Cycle

A *walk* is an alternating sequence of vertices and edges. A *trail* is a walk with distinct edges while a *path* is a walk with distinct vertices. A *cycle* is a closed walk with distinct vertices.

If all vertex pairs of a graph are joined by a path, this graph is *connected*. An *acyclic graph* has no cycles and if it is also connected, it is called a *tree*. Vertices of trees are also called *nodes*. Some trees have a distinct node which is called *root*. Such a tree is called a *rooted tree*. Nodes that are adjacent to only one other node are *leaves*.

A graph is said to be *chordal* or *triangulated* if every cycle with more than three edges possesses a *chord*, i.e. an edge which joins two non-consecutive vertices in the cycle.

Definition 2-24: Subgraph

A graph $G_S(V_S, E_S)$ is called *subgraph* of $G(V, E)$ if $V_S \subseteq V$ and $E_S \subseteq E$. If $U \subseteq V$, the subgraph *induced or spanned by* U is the maximal subgraph of $G(V, E)$ whose edges have both end-points in U .

Definition 2-25: Bipartition, Bipartite Graph

A *bipartition* of a graph $G(U, E)$ is an ordered pair (V, W) of complementary subsets of U such that each edge of E has one end-point in V and the other in W . A *bipartite graph* is one that has a bipartition.

A bipartite graph can have neither a loop nor a chord. Figure 2-3 shows a bipartite graph. The two sets V and W are marked grey and white.

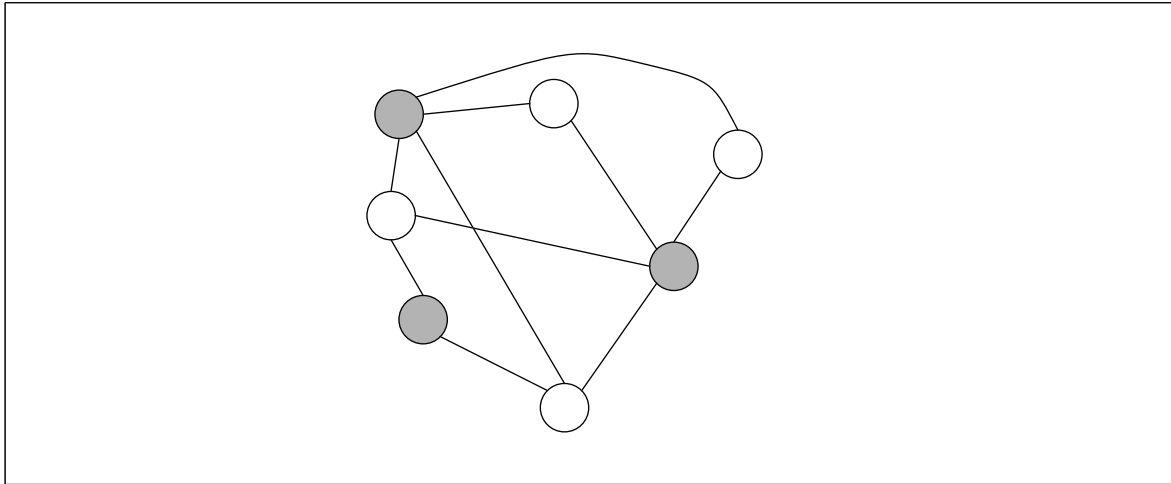


Figure 2-3: Bipartite graph

Directed Graphs

Most concepts of undirected graphs can be straightforwardly extended to directed graphs by replacing *edges* with *edges with the same direction*. But directed graphs allow for some specific definitions that deal with the direction of the edges.

The *indegree* and *outdegree* of a vertex are the numbers of edges where the vertex is the head and the tail, respectively.

Definition 2-26: Dag

A directed, acyclic graph is called a *dag*. Dags represent partially ordered sets.

The netlist of a combinational circuit is a dag. In a dag, vertex v is called the *successor* of vertex w if v is the head of a path whose tail is w . We also say, w is *reachable* from v . Under the same preliminaries, v is the *predecessor* of w . If there is an edge (v,w) then v and w are *direct predecessor* and *direct successor*, respectively.

Definition 2-27: Polar Dag

A dag is *polar* if it has two distinguished vertices with the following properties. One vertex, the *sink*, is reachable from all other vertices while from one vertex, the *source*, all other vertices are reachable.

The graph in figure 2-2b is a polar dag, vertex a is its source and vertex c its sink.

2.1.3 Boolean Functions

Any combinational or sequential digital circuit can be expressed as a set of boolean functions. Thus, the theory of Boolean functions provides the mathematical foundation for circuit analysis and manipulation. The terminology proposed in this section corresponds to the one which is used in [Bait 87] and [DeMi 94].

Definition 2-28: Boolean Algebra

A Boolean algebra $BA = [B, \nabla, \Delta]$ is defined by a set B , two operations ∇ and Δ , and Huntington's postulates:

1. Completeness: $\forall a, b \in B : a \nabla b \in B$ and $a \Delta b \in B$
2. Commutativity: $\forall a, b \in B : a \nabla b = b \nabla a$ and $a \Delta b = b \Delta a$
3. Distributivity: $\forall a, b, c \in B : (a \nabla b) \Delta c = (a \Delta c) \nabla (b \Delta c)$ and
 $(a \Delta b) \nabla c = (a \nabla c) \Delta (b \nabla c)$
4. One-element e : $\exists e \in B : \forall a \in B : a \nabla e = a$
 Null-element n : $\exists n \in B : \forall a \in B : a \Delta n = a$
5. Complement: $\forall a \in B : \exists \bar{a} \in B : a \nabla \bar{a} = n$ and $a \Delta \bar{a} = e$

Definition 2-29: Binary Boolean Algebra

In a binary Boolean algebra $BA_2 = [\{0, 1\}, \cdot, +]$ the set B has only two elements. \cdot is called *conjunction*, $+$ is called *disjunction*. 1 is the one-element e , 0 is the null-element n . $\bar{1}=0$ and $\bar{0}=1$. A binary Boolean algebra is also called *switching algebra*.

In a binary Boolean algebra, \cdot and $+$ are sometimes denoted as $\&$ and \vee , respectively. The expression $a \cdot b$ can be abbreviated by ab and in $(a \cdot b) + (c \cdot d)$ the brackets can be omitted.

The following properties can be directly derived for binary Boolean algebras from Huntington's postulates:

1. Associativity: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
2. Idempotence: $a + a = a$ and $a \cdot a = a$
3. Absorption: $a + (a \cdot b) = a$ and $a \cdot (a + b) = a$
4. DeMorgan's Law: $\overline{(a + b)} = \bar{a} \cdot \bar{b}$ and $\overline{(a \cdot b)} = \bar{a} + \bar{b}$
5. Involution: $\overline{(\bar{a})} = a$

Any instance of a variable, either negated or not, is called a *literal*. A product of literals is called a *cube* and a sum of cubes is a *cubeset*.

In this work only binary Boolean algebras are considered.

Definition 2-30: Boolean Function

A *completely specified Boolean function* is a mapping from the n -dimensional Boolean space to the m -dimensional Boolean space:

$$f: B^n \rightarrow B^m$$

An *incompletely specified, scalar Boolean function* is a mapping

$$f_i: S \rightarrow B \text{ with } S \subset B^n.$$

The points where f_i is not defined are called *don't care conditions (DC)*. An incompletely specified Boolean function can equivalently be defined as

$$f: B^n \rightarrow \{0, 1, *\}^m,$$

where the symbol $*$ denotes a DC condition. For each output, the subsets of the domain for which the function takes the values 0 , 1 , and $*$ are called the *off set*, *on set*, and *dc set*, respectively.

Any specific vector $\underline{b} = (b_0 b_1 \dots b_{n-1})$ is also called a *pattern*. $b_i = -$ is equivalent to $b_i = 0 \vee b_i = 1$. The symbol $-$ is called *input don't care* or briefly *don't care (DC)*.

Definition 2-31: Cofactor

The *cofactor* f_{x_i} of a Boolean function $f(x_1, x_2, \dots, x_i, \dots, x_n)$ with respect to variable x_i is $f_{x_i}(x_1, x_2, \dots, x_i, \dots, x_n) = f(x_1, x_2, \dots, 1, \dots, x_n)$. The *cofactor* $f_{\bar{x}_i}$ of f with respect to variable x_i is $f_{\bar{x}_i} = f(x_1, x_2, \dots, 0, \dots, x_n)$.

The following theorem allows to decompose any Boolean function into cofactors.

Theorem 2-1: Shannon's Expansion

Let $f: B^n \rightarrow B$. Then $\forall i=1\dots n$

$$f(x_1, \dots, x_i, \dots, x_n) = x_i \cdot f_{x_i} + \bar{x}_i \cdot f_{\bar{x}_i} = (x_i + f_{\bar{x}_i}) \cdot (x_i + f_{x_i}).$$

Shannon's expansion was first published in [Shan 38], a prove can also be found in [Hach 96]. Any function $f: B^n \rightarrow B$ can be represented as a sum of products with n literals by recursive expansion. These products are called *minterms*. Using the second form of Shannon's expansion, the same function can be expressed as a product of sums of n literals. These sums are called *maxterms*. The two representations are also called *minterm* and *maxterm canonical form*, respectively.

For power estimation, the following operand is essentially important.

Definition 2-32: Boolean Difference, Boolean Derivative

The *Boolean difference* or *Boolean derivative* of $f(x_1, \dots, x_i, \dots, x_n)$ with respect to

$$\text{variable } x_i \text{ is } \frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{\bar{x}_i}.$$

The symbol \oplus denotes the XOR operation which is defined as by

Definition 2-33: XOR \oplus

$$a \oplus b = a\bar{b} + \bar{a}b$$

The Boolean difference indicates whether the function f is sensitive to changes in the variable x_i .

Although the following two definitions originate from the coding theory [Boss 92] they are also applied for power estimation purposes.

Definition 2-34: Hamming Weight

Given a Boolean vector $\underline{b} = (b_0b_1\dots b_{n-1})$, its *hamming weight* $wt(\underline{b})$ is defined by

$$wt(\underline{b}) = \sum_{i=0}^{n-1} b_i.$$

Definition 2-35: Hamming Distance

Given two Boolean vectors $\underline{a} = (a_0a_1\dots a_{n-1})$ and $\underline{b} = (b_0b_1\dots b_{n-1})$ their *Hamming Distance* $dist(\underline{a}, \underline{b})$ is given by

$$wt(\underline{a}, \underline{b}) = \sum_{i=0}^{n-1} a_i \oplus b_i.$$

Hence, the Hamming Weight denotes the number of 1s in a Boolean vector while the Hamming Distance is the number of different literals between two Boolean vectors.

2.1.4 Binary Decision Diagrams

Binary Decision Diagrams (BDD) were first introduced by Lee [Lee 59] and Akers [Aker 78]. But only after the publication of Bryant [Brya 86] in 1986 they have started to gain widespread use in many areas of design automation like test, verification, and power estimation. This section gives a short overview over BDDs and some of their extensions. However, it cannot cover all the different flavors of BDDs and FDDs that were developed for different purposes during the last decade. The interested reader is referred to two excellent textbooks which provide an introduction to all kinds of decision diagrams: [Mina 96] and [Sasa 96].

Definition 2-36: Ordered Binary Decision Diagram

An *Ordered Binary Decision Diagram (OBDD)* is a rooted, directed graph $G(V, E)$.

Each non-leaf vertex v has an attribute $index(v) \in \{1, 2, \dots, n\}$ and two children $low(v)$ and $high(v) \in V$. $index(v)$ assigns one of the input variables $\{x_1, x_2, \dots, x_n\}$ to node v . A leaf vertex has an attribute $value(v) \in B$.

For any vertex pair $\{v, low(v)\}$ and $\{v, high(v)\}$ such that no vertex is a leaf, the following expressions must be true:

$$index(v) < index(low(v))$$

$$index(v) < index(high(v)).$$

The last two requirements in definition 2-36 introduce a variable order which guarantees acyclic graphs.

Definition 2-37: OBDD evaluation

An OBDD with root v denotes a function f^v such that:

1. If v is a leaf, then $f^v = value(v)$.
2. If v is not a leaf and $index(v) = i$, then $f^v = \bar{x}_i \cdot f^{low(v)} + x_i \cdot f^{high(v)}$.

According to theorem 2-1, $f^{low(v)}$ and $f^{high(v)}$ must be the two cofactors of f with respect to variable x_i . Hence, the decomposition of definition 2-37.2 represents a Shannon's expansion.

Two BDDs are *isomorphic* if there is a one-to-one mapping between the vertex sets that preserves adjacency, indexes, and leaf values. Two isomorphic OBDDs represent the same Boolean function.

It is one major strength of OBDDs that they offer a canonical representation of Boolean functions. However, canonicity requires the removal of any redundancy in the OBDD.

Definition 2-38: Reduced Ordered Binary Decision Diagram

An OBDD is said to be a *reduced OBDD (ROBDD)* if it contains neither any vertex v with $low(v) = high(v)$, nor any vertex pair $\{u,v\}$ such that the subgraphs rooted in u and in v are isomorphic.

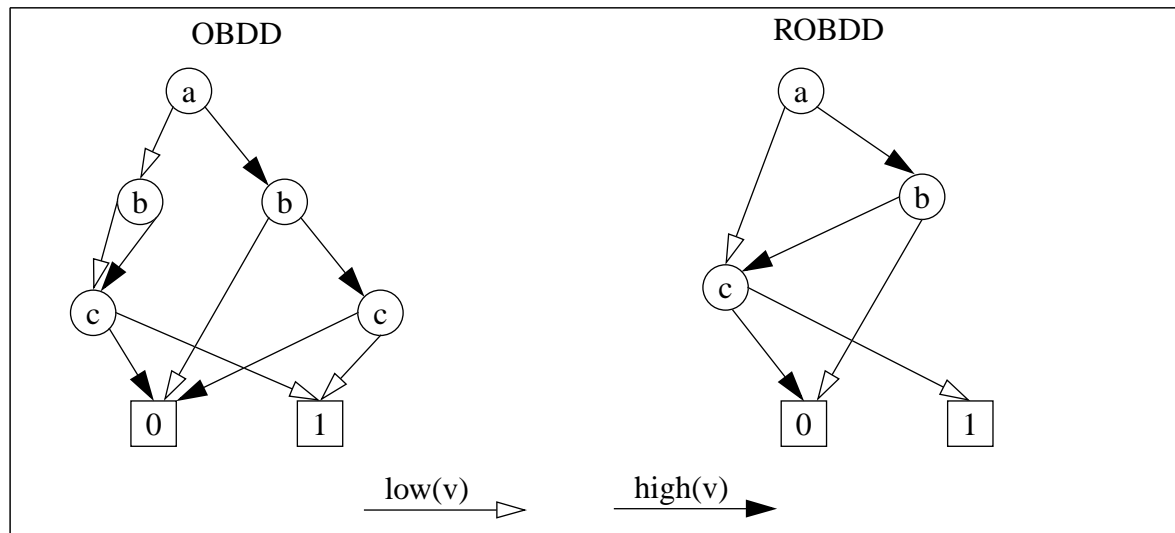


Figure 2-4: OBDD and ROBDD for $\bar{c} + ab$

ROBDDs can be derived either from OBDDs by a reduction algorithm [Brya 86] or they can be directly reduced during construction using the *unique-table*. It implements a hashtable that contains a key for each vertex of a ROBDD. The key is a triple, which is composed of the corresponding variable and the identifiers of the left and right children. Whenever a new vertex is to be added to the OBDD under construction, a lookup in the unique-table allows

to check whether a vertex already exists that implements the same functionality as required. Figure 2-4 shows an OBDD and its ROBDD. Unless stated otherwise, all BDDs shall be reduced and ordered in the sequel of this thesis.

Definition 2-39: Shared BDD, Multi-Rooted ROBDD

A set of BDDs can be joined into one graph with multiple root nodes and shared sub-graphs. Each root node represents the root of one BDD. Such a graph is called a *shared BDD* or a *multi-rooted BDD*.

Figure 2-5 shows a shared BDD. Beside efficient memory usage, shared BDDs allow very efficient implementation of some operations. E.g. equivalence checking requires only to compare the root nodes.

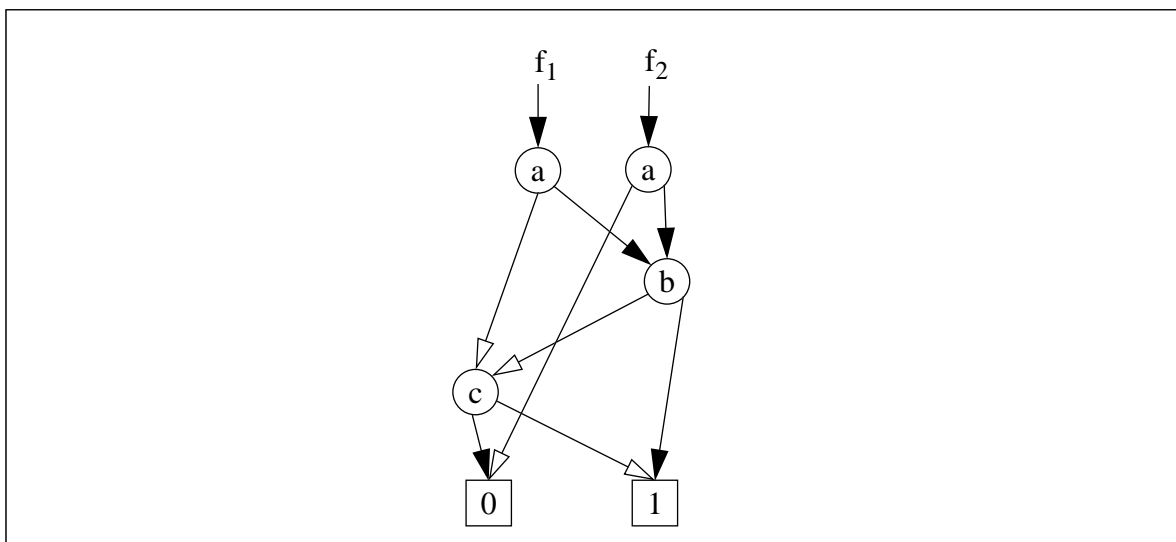


Figure 2-5: Shared ROBDD for $f_1 = \bar{c} + ab$ and $f_2 = a \cdot (b + \bar{c})$

Definition 2-40: ITE Operator

Given three scalar Boolean functions, f , g , and h , $ite(f, g, h) = f \cdot g + \bar{f} \cdot h$ which is equivalent to “*If f Then g Else h* ”.

The benefit of the ITE operator is based on two facts:

1. It can be used to construct and manipulate BDDs since all functions of two arguments can be implemented using the ITE operator (table 2-1).
2. It can be very efficiently implemented using look-up tables [DeMi 94].

BDDs are very sensitive to the variable order. For most practical Boolean functions, good variable orders exist such that the size of the BDD grows only linearly with the number of variables while for unfavorable orders, they can grow exponentially. Thus, finding good

Operator	ITE Form	Operator	ITE Form
0/1	0/1	\bar{f}	$\text{ite}(f, 0, 1)$
$f \cdot g$	$\text{ite}(f, g, 0)$	$\overline{f + g}$	$\text{ite}(f, 0, \bar{g})$
$f \oplus g$	$\text{ite}(f, \bar{g}, g)$	$\overline{f \oplus g}$	$\text{ite}(f, g, \bar{g})$
$f + g$	$\text{ite}(f, 1, g)$	$\overline{f \cdot g}$	$\text{ite}(f, \bar{g}, 1)$

Table 2-1: Boolean functions of two arguments and their ITE form

variable orders is crucial for efficient BDD operations. However, the optimal order can only be found for relatively small functions since the problem is NP-complete [Sasa 96]. For larger functions heuristics are applied. Good variable orders have two empirical properties:

1. *Local computability:* variables that are closely related to each other should be kept near to each other. For instance, given the function $f = x_1x_2 + x_3x_4 + x_5x_6$ the variable pairs x_1 and x_2 , x_3 and x_4 , x_5 and x_6 should be kept adjacent.
2. *Power to control the output:* the inputs that greatly affect the function should be located in higher positions. E.g. the selector input of a multiplexer should be in the highest position.

Unfortunately, both conditions are often contradictory and in shared BDDs, the different functions may have different optimal orders. Several heuristics, like the *dynamic weight assignment method* [Mina 90] or the *minimum width method* [Mina 96] and many others were proposed in the past. But the development of new heuristics is still subject to ongoing research. Even the best methods that are known today, perform good on many circuits but they fail on several others and/or have very long runtimes. Furthermore, Bryant has proven that there exist inherently complex circuits which are exponential in size for all possible variable orders [Brya 86]. Multipliers represent such a circuit type of practical importance.

Extensions to BDDs

Starting from ROBDDs, several extensions of BDDs were proposed during the last twelve years. Those that did influence this work will be presented in the sequel.

The first class of extensions are *attributed edges*, like *negative edges*, *input inverters*, and *variable shifters*. They are used to further compress the BDD.

Definition 2-41: Negative Edge

A *negative edge* has an attribute \bullet which indicates that the function of the subgraph, pointed to by the edge, should be complemented. In order to keep canonicity, two new constraints must be introduced for BDDs with negative edges:

- Omit the 0-terminal.
- Don't negate the 1-edge.

Negative edges provide the following advantages compared to ordinary BDDs:

- BDD size is reduced up to one half.
- Negation can be performed in constant time.
- Logic operations are accelerated by specific rules such as

$$f \cdot \bar{f} = 0, f + \bar{f} = 1, \text{ and } f \oplus \bar{f} = 1.$$

Figure 2-6a shows an example of a BDD without and with negative edges while the conversion rules for the 0-node and negated 1-edges are depicted in figure 2-6b.

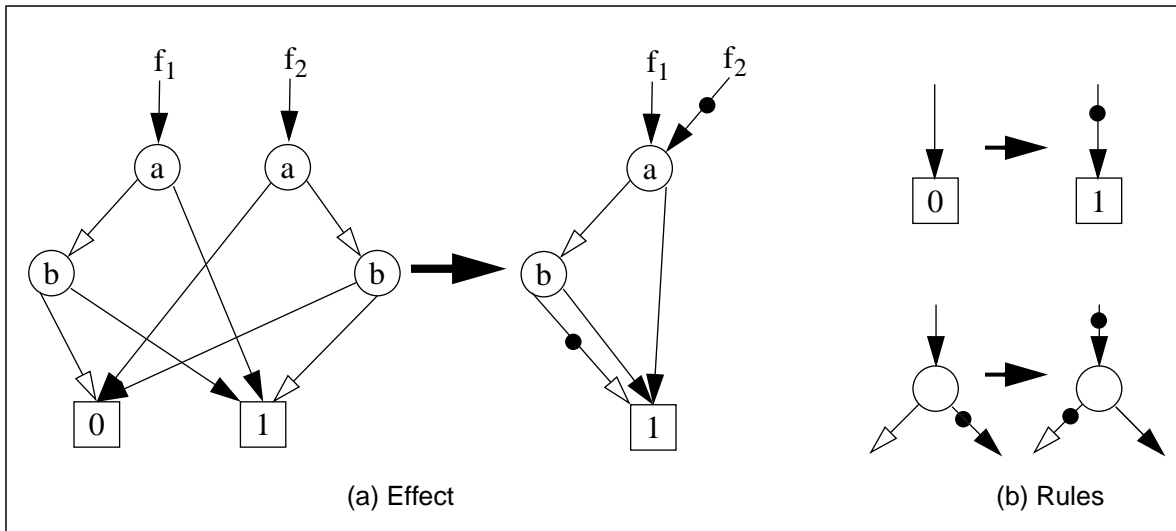


Figure 2-6: Negative Edges

Input inverters are similar to negative edges. They specify that the 0- and 1-edges of the next node should be exchanged. This can be considered as inverters for the according input variable. Just like negative edges, input inverters require special constraints in order to preserve canonicity.

Variable shifters can be useful if there exist two subgraphs that are isomorphic except for a difference in the indexes. The variable shifter indicates that a certain integer number should be added to the indexes. A detailed description can be found in [Mina 96].

Attributes are not only attached to edges but also to the BDD nodes, which changes their functionality.

Definition 2-42: Ordered Kronecker Functional Decision Diagram (OKFDD)

In an *Ordered Kronecker Functional Decision Diagram (OKFDD)* each variable has an attribute that defines the decomposition type of the nodes with this variable index. The decomposition type can be one of the following three types:

1. Shannon $f = x_i \cdot f_{x_i} + \bar{x}_i \cdot f_{\bar{x}_i}$ (S)

2. Positive Davio $f = f_{x_i}^- \oplus x_i \cdot (f_{x_i}^- \oplus f_{x_i})$ (pD)

$$3. \text{ Negative Davio} \quad f = f_{x_i} \oplus \bar{x}_i \cdot (f_{\bar{x}_i} \oplus f_{x_i}) \quad (\text{nD})$$

A vertex v with label x_i is then evaluated by

$$1. \text{ Shannon} \quad f_v = x_i \cdot f_{\text{high}(v)} + \bar{x}_i \cdot f_{\text{low}(v)}$$

$$2. \text{ Positive Davio} \quad f_v = f_{\text{low}(v)} \oplus x_i \cdot f_{\text{high}(v)}$$

$$3. \text{ Negative Davio} \quad f_v = f_{\text{low}(v)} \oplus \bar{x}_i \cdot f_{\text{high}(v)},$$

depending on the attribute of x_i .

Hence, an OBDD can be considered as an OKFDD, where all variables are of the Shannon type. Due to the additional degree of freedom, OKFDDs can result in substantially smaller representations than BDDs. They still don't solve the problem of multiplier representation, though. Like for BDDs, there exist special reduction rules and heuristics for OKFDDs. Detailed explanation would go beyond the scope of this introduction. The interested reader is referred to [Sasa 96] and [Drec 96].

2.1.5 Probability Theory

This section gives the most important definitions and results of the probability theory. Detailed introductions can be found in [Papo 65] and [Haus 77]. For topics that are related to binary signals [Najm 93] should be consulted.

Definition 2-43: Probability

Given is a set S of events $\{s_1, s_2, \dots, s_k\}$. The certain event C is the union of all possible events s_i :

$$C = \bigcup_{i=1}^k s_i$$

The *probability* of an event s_i is a number $P(s_i)$ assigned to this event. $P(s_i)$ has the following properties:

- $P(s_i) \geq 0$
- $P(C) = 1$
- $P(s_i \cup s_j) = P(s_i) + P(s_j)$ if s_i and s_j are mutually exclusive

Definition 2-44: Elementary Event, Probability Space

If all the elements of S are mutually exclusive, then s_1, s_2, \dots, s_k are called *elementary events* and S is a *probability space*. p_i denotes the probability of the elementary event s_i : $P(s_i) = p_i$

The elements of the probability space S are arbitrary, abstract events. Given for instance, a set A that consists of cars in the three colors red, blue, and yellow. Then $S = \{\text{red}, \text{blue}, \text{yellow}\}$ and p_{red} denotes the probability that a car, randomly chosen from A , is red.

Definition 2-45: Conditional Probability

Given two events a, b with nonzero probability, then the *conditional probability of a assuming b* is defined by

$$P(a|b) = \frac{P(a \cap b)}{P(b)}.$$

Bayes theorem describes the relationship between the conditional probabilities of two variables:

Theorem 2-2: Bayes Theorem

$$P(a|b) \cdot P(b) = P(b|a) \cdot P(a)$$

Definition 2-46: Statistical Independence

Two events a and b are called *statistically independent* if $P(a \cap b) = P(a) \cdot P(b)$.

Hence, $P(a|b) = P(a)$ if a and b are statistically independent.

Random Variables**Definition 2-47: Random Variable**

A real random variable $\mathbf{x}: S \rightarrow \mathbf{R}$ is a real function whose domain is the probability space S such that:

1. The set $\{\mathbf{x} \leq x\}$ is an event for any real number x .
2. $P(\{\mathbf{x} = +\infty\}) = P(\{\mathbf{x} = -\infty\}) = 0$.

The concept of *random variables* allows to transfer any abstract probability space S , into the domain of the real numbers where the following terms can be defined.

Definition 2-48: Distribution Function

The *distribution function* of the random variable \mathbf{x} is the function

$$F_{\mathbf{x}}(x) = P(\{\mathbf{x} \leq x\}).$$

It is defined for any number from $-\infty \leq x \leq +\infty$.

Definition 2-49: Probability Density

The function

$$f_{\mathbf{x}}(x) = \frac{dF_{\mathbf{x}}(x)}{dx}$$

is called the *probability density* or *density function* of the random variable \mathbf{x} .

Definition 2-50: Expected Value, Mean

The *expected value* or *mean* $E(\mathbf{x})$ of a random variable \mathbf{x} is defined as

$$E(\mathbf{x}) = \int_{-\infty}^{\infty} x f_{\mathbf{x}}(x) dx$$

$E(\mathbf{x})$ is sometimes denoted as $\mu_{\mathbf{x}}$.

The *mean* is a special case of the more general term *moment* which is defined next.

Definition 2-51: Moment, central moment.

The *n*th moment and the *n*th central moment of a random variable \mathbf{x} are defined by

$$m_n = E(\mathbf{x}^n) = \int_{-\infty}^{\infty} \mathbf{x}^n f_{\mathbf{x}}(x) dx \text{ and}$$

$$cm_n = E((\mathbf{x} - \mu_{\mathbf{x}})^n) = \int_{-\infty}^{\infty} (x - \mu_{\mathbf{x}})^n f_{\mathbf{x}}(x) dx ,$$

respectively.

Some moments got special names due to their practical importance. The first moment is the *mean* $\mu_{\mathbf{x}}$. The second central moment is called *variance*, its square root is the *standard deviation* $\sigma_{\mathbf{x}}$. There exists an interesting relation between the variance and the second moment:

$$E(\mathbf{x}^2) = \sigma_{\mathbf{x}}^2 + \mu_{\mathbf{x}}^2 \tag{ 2-1 }$$

Sometimes it is necessary to model the statistical relations between two or more random variables. Therefore most definitions of a single variable can be extended to a multi-variable form.

Multi-Variable Statistics

Definition 2-52: Joint Distribution, Joint Density

The *joint distribution* and the *joint density function* of two random variables \mathbf{x} and \mathbf{y} are defined by

$$F_{\mathbf{xy}}(x, y) = P(\{\mathbf{x} \leq x, \mathbf{y} \leq y\}) \text{ and } f_{\mathbf{xy}}(x, y) = \frac{\partial^2 F_{\mathbf{xy}}(x, y)}{\partial x \partial y} ,$$

respectively.

Definition 2-53: Joint Moments, Joint Central Moments

The *joint moments* and the *joint central moments* of two random variables \mathbf{x} and \mathbf{y} are defined by

$$m_{kl} = E(\mathbf{x}^k \mathbf{y}^l) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{x}^k \mathbf{y}^l f_{\mathbf{xy}}(x, y) dx dy \text{ and}$$

$$cm_{kl} = E((\mathbf{x} - \mu_{\mathbf{x}})^k (\mathbf{y} - \mu_{\mathbf{y}})^l) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_{\mathbf{x}})^k (y - \mu_{\mathbf{y}})^l f_{\mathbf{xy}}(x, y) dx dy ,$$

respectively.

cm_{11} is also called *covariance* σ_{xy}^2 . Equation 2-1 can be extended for two dimensional moments:

$$\sigma_{xy}^2 = E(\mathbf{xy}) - \mu_x \mu_y \quad (2-2)$$

Definition 2-54: Correlation Coefficient

The *correlation coefficient* r_{xy} of two random variables x and y is defined by

$$r_{xy} = \frac{\sigma_{xy}^2}{\sigma_x \sigma_y}$$

Definition 2-55: Uncorrelated, Orthogonal, and Independent Random Variables

- Two random variables are called *uncorrelated* if $E(\mathbf{xy}) = \mu_x \mu_y$.
- They are *orthogonal* if $E(\mathbf{xy}) = 0$.
- They are *independent* if $f_{xy}(x,y) = f_x(x)f_y(y)$.

If two random variables x , y are uncorrelated, their correlation coefficient r_{xy} and their covariance σ_{xy}^2 are zero. Depending on the sign of r_{xy} , we also say they are *positively* or *negatively* correlated. Independent variables are always uncorrelated but not vice versa.

Stochastic Processes

The concept of *stochastic processes* is fundamental to probabilistic treatment of signal processing systems.

Definition 2-56: Stochastic Process

Given an experiment E specified by its outcomes s . The outcomes form the probability space S , by certain subsets of S called *events* and by the probabilities of these events. If a time function $x(t,s)$ is assigned to every outcome s according to a certain rule, $x(t,s)$ is called a *stochastic process*. Usually the second parameter s can be omitted.

Figure 2-7 illustrates the definition of a stochastic process. The stochastic process consists of a function generator and one or many multiplexers. The function generator G generates an arbitrary number of time functions $x_i(t)$. Each multiplexer selects one of these functions, depending on the random events s_a or s_b . Thus, the signals $x(t,s_a)$ and $x(t,s_b)$ are samples of stochastic processes. If s_a and s_b are random events of the same space S , then $x(t,s_a)$ and $x(t,s_b)$ are two samples of one stochastic process $x(t,s)$. The NAND-gate represents a system where these two signals are applied to. It is important to note that the functions $x_i(t)$ are deterministic, it is the selection of specific functions that introduces the probabilistic character. Thus, $x(t_j)$ is a random variable for fixed t_j . This allows to directly transfer the definitions of distribution, density, moment, and all the related terms to stochastic processes.

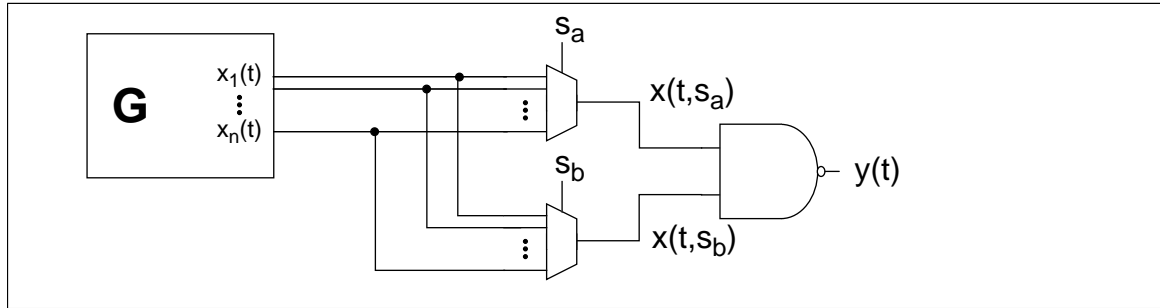


Figure 2-7: A stochastic digital system

Definition 2-57: Distribution, Density

Given a stochastic process $\mathbf{x}(t)$ one defines the following terms:

- *Density*: $F_{\mathbf{x}}(\mathbf{x}, t) = P(\{\mathbf{x}(t) \leq \mathbf{x}\})$
- *Distribution*: $f_{\mathbf{x}}(\mathbf{x}, t) = \frac{\partial F_{\mathbf{x}}(\mathbf{x}, t)}{\partial \mathbf{x}}$
- *Joint Density*: $F_{\mathbf{xx}}(x_1, t_1; x_2, t_2) = P(\{\mathbf{x}(t_1) \leq x_1, \mathbf{x}(t_2) \leq x_2\})$
- *Joint Distribution*: $f_{\mathbf{xx}}(x_1, t_1; x_2, t_2) = \frac{\partial^2 F_{\mathbf{xx}}(x_1, t_1; x_2, t_2)}{\partial x_1 \partial x_2}$

The density and joint density are also called *first* and *second order distributions*, respectively.

The moments are defined according to definitions 2-51 and 2-53 for specific time instances t_i , for instance:

Definition 2-58: Mean

The *mean* $\mu_{\mathbf{x}}(t)$ is defined by

$$\mu_{\mathbf{x}}(t) = E(\mathbf{x}(t)) = \int_{-\infty}^{\infty} \mathbf{x} f_{\mathbf{x}}(\mathbf{x}, t) d\mathbf{x}.$$

In general, all moments are functions of time.

Definition 2-59: Autocorrelation, Autocovariance

The *autocorrelation* $R(t_1, t_2)$ of a process $\mathbf{x}(t)$ is the joint moment of the random variables $\mathbf{x}(t_1)$ and $\mathbf{x}(t_2)$:

$$R(t_1, t_2) = E(\mathbf{x}(t_1), \mathbf{x}(t_2)) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f_{\mathbf{xx}}(x_1, t_1; x_2, t_2) dx_1 dx_2$$

The *autocovariance* is defined by

$$C(t_1, t_2) = E([\mathbf{x}(t_1) - \mu_{\mathbf{x}}(t_1)] \cdot [\mathbf{x}(t_2) - \mu_{\mathbf{x}}(t_2)])$$

Equations 2-1 and 2-2 can also be applied to the autocorrelation and the autocovariance.

If the autocorrelation of a process is not zero, the process is said to be *temporally correlated*. If the correlation of two processes does not disappear, they are *spatially* or *mutually correlated*.

Definition 2-60: Strict Sense Stationary

A stochastic process $\mathbf{x}(t)$ is *strict sense stationary (SSS)* if its statistics are not affected by a shift in the time origin. Two processes $\mathbf{x}(t)$ and $\mathbf{y}(t)$ are *jointly SSS* if their joint statistics are not affected by a shift in the time origin.

Definition 2-61: Wide Sense Stationary

A stochastic process $\mathbf{x}(t)$ is *wide sense stationary (WSS)* if its mean is constant: $\mu_{\mathbf{x}}(t) = \mu_{\mathbf{x}}$ and its autocorrelation does only depend on $\tau=t_1-t_2$: $R(t_1, t_2) = R(\tau)$.

Definition 2-62: Mean Ergodic

A stochastic process is called *mean ergodic* if

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \mathbf{x}(t) dt = \mu_{\mathbf{x}}$$

Note that mean ergodicity requires constant mean.

Mean ergodicity has very important practical consequences. Only for a mean ergodic process it is possible to determine an estimate of $\mu_{\mathbf{x}}$ by finite time observation of a single sample of $\mathbf{x}(t)$.

Logic Signals

For logic signals there are only two events $\{0,1\}$. Najm has proven in [Najm 93] that a logic signal is always a sample of a *SSS mean-ergodic 0-1 process*. Moreover, this process can be built from a single signal $x(t)$.

Definition 2-63: Companion Process

Given a logic signal $x(t)$ and a random variable τ , uniformly distributed over R . The *companion process of $x(t)$* is the 0-1 stochastic process given by:

$$\mathbf{x}(t) = x(t + \tau).$$

Using the companion process, it is possible to derive estimates for all statistical properties of the process $\mathbf{x}(t)$ from a single sample $x(t)$. In practice, $x(t)$ can be obtained from a simulation of the system under consideration.

Definition 2-64: Signal Probability

The *signal probability* p_x or $P(x)$ of a signal $x(t)$ is the probability for x to be I :

$$p_x = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t) dt.$$

The signal probability is also called *equilibrium probability*.

The signal probability is identical with the mean of the according companion process $x(t)$. In practice, the exact value of p_x is usually not known, but an estimate \hat{p}_x can be obtained by finite time observation of $x(t)$:

$$\hat{p}_x = \frac{1}{T} \int_0^T x(t) dt \tag{2-3}$$

In a clocked design, $x(t)$ changes its value only once per clock cycle. In that case the integral degenerates to a sum.

$$\hat{p}_x = \frac{1}{n} \cdot \sum_{i=1}^n x(i \cdot T) = \frac{1}{n} \cdot \sum_{i=1}^n x_i \tag{2-4}$$

Estimates for other moments can be obtained accordingly.

Definition 2-65: Transition Density

Let $n_x(T)$ be the number of transitions in the logic signal $x(t)$ in the interval $(-T/2, T/2]$. Then the *transition density* is defined as

$$D(x) = \lim_{T \rightarrow \infty} \frac{n_x(T)}{T}.$$

The transition density gives the average number of transitions per time unit.

Theorem 2-3

Given $x(t)$ and $y(t)$ as the companion processes of two logic signals $x(t)$ and $y(t)$. If $x(t)$ and $y(t)$ are uncorrelated then they are also statistically independent.

Prove

If $x(t)$ and $y(t)$ are uncorrelated then $E(xy) = \mu_x \mu_y = P(x)P(y)$. And according to definition 2-53:

$$E(xy) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy f_{xy}(x, y) dx dy = 1 \cdot 1 \cdot P(x \cap y) = P(x \cap y).$$

Thus $P(x \cap y) = P(x)P(y)$. □

Markoff Chains

Markoff chains are used for mathematically describing finite state machines (FSM) and switching operations in digital circuits.

Definition 2-66: Markoff Chain

Given an ordered set of random variables (x_1, x_2, \dots, x_n) where $x_i \in \{a_1, a_2, \dots, a_N\}$. If for all x_i we have

$$F(x_n | x_{n-1}, x_{n-2}, \dots, x_1) = F(x_n | x_{n-1})$$

that means, each random variable depends only on its direct predecessor, then such a set is called a *Markoff chain*.

As already mentioned before, Markoff chains are used to model FSMs. x_i represents the state of the machine in clock cycle i and a_1, a_2, \dots, a_N are the 2^m possible states of an FSM that consists of m flip flops. In an FSM, the next state depends only on the current state and the current input vector but not on previous states.

Figure 2-8 shows the automata graph of a two state FSM. This FSM can be interpreted as a Markoff chain with $a_1=a$ and $a_2=b$.

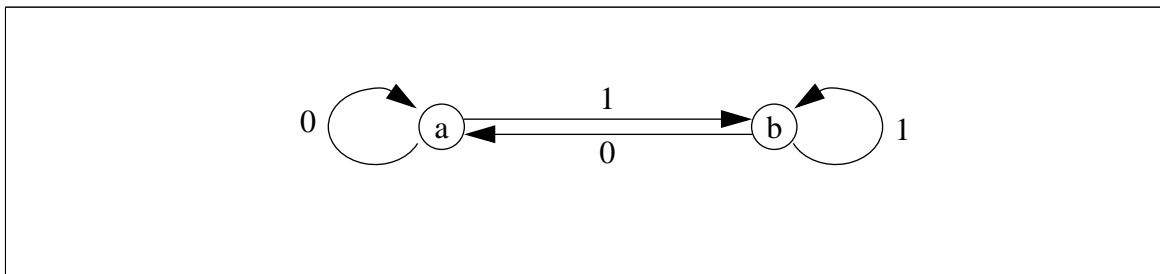


Figure 2-8: Two state Markoff chain

2.2 Physical Background

2.2.1 CMOS Technology

The MOS (metal oxide silicon) technology has become the dominant technology for the design of VLSI circuits for several reasons. MOS transistors enable much higher integration densities than their predecessors, the bipolar transistors, due to their simple structure. Furthermore, they work at much lower supply voltages and static currents are negligible which avoids expensive cooling devices. Compared to other materials like the 3-5 semiconductor GaAs, silicon is cheap and easy to manufacture. Figure 2-9 shows the schematics and a cross-section of a CMOS inverter.

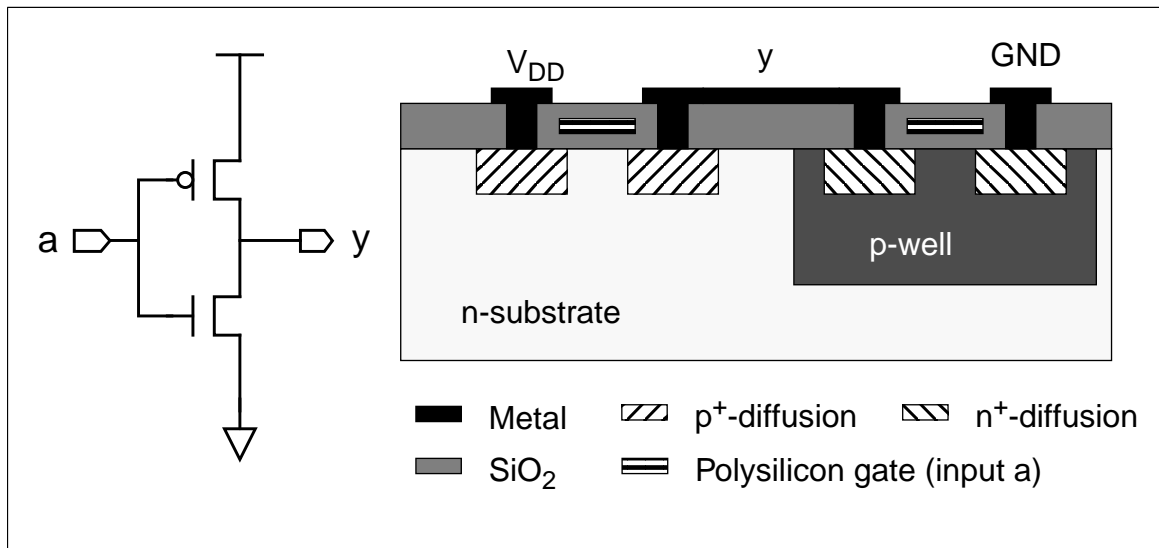


Figure 2-9: Schematic and cross-section of a CMOS inverter

Beside some technological parameters, the maximum operation frequency of a MOS transistor depends on the channel length L and the parasitic capacitances which are depicted in figure 2-10. The latter influence also the dynamic energy consumption of the transistor, as it will be shown later.

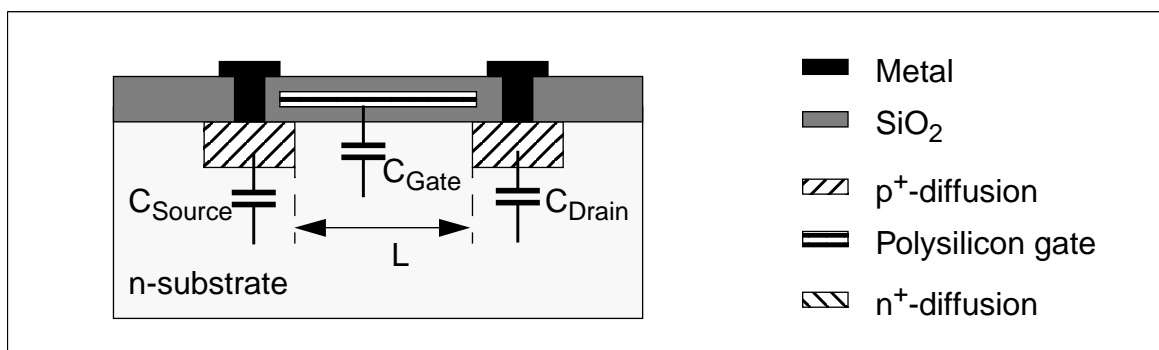


Figure 2-10: Parasitic capacitances and channel length of a MOS transistor

Silicon-on-Insulator

While the conventional MOS transistor is produced by diffusing the drains into bulk material, the SOI (Silicon-on-Insulator) transistor is built on top of a SiO_2 substrate. SOI reaches even higher integration densities since it doesn't require any area-consuming wells [Alle 97]. Furthermore, the parasitic drain and source capacitances are dramatically reduced because of the insulation of the transistors. This results into faster circuits with lower power consumption.

Problems of the SOI technology are related to production cost and the channel contact. Today, the cost for an SOI wafer is rather high compared to a bulk CMOS wafer. But this may change in the near future as production capacities for SOI will be increased. The chan-

nel contact poses a design problem since any contact requires additional chip area and consumes valuable routing space. If the active layer is thin enough, the channel can be left floating without severe electrical drawbacks [Coli 91]. However, circuit level simulation of floating channel transistors requires up to four times the CPU time as bulk contacted transistors [Maye 98] since the floating channel cannot be efficiently modeled with the existing transistor models.

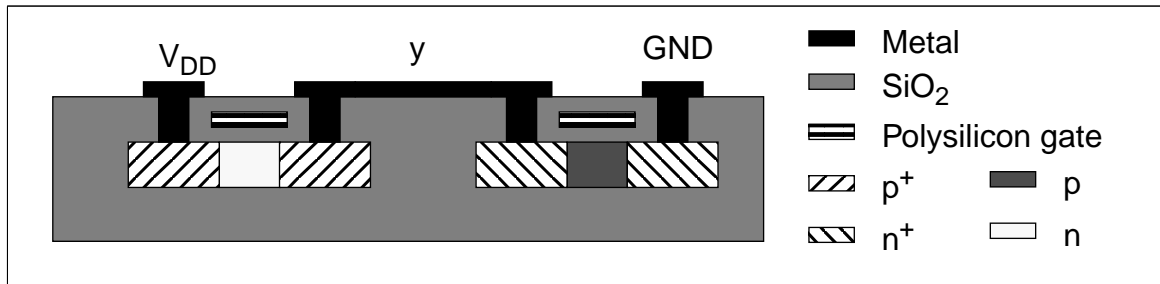


Figure 2-11: Inverter in SOI technology

T-Gate

Figure 2-12 depicts a T-gate transistor. The T-shaped gate is obtained by laterally growing a spacer on an oxide edge [Dude 96], [Dude 97]. The thickness of the spacer can be very accurately controlled by the time of the growing process. Thus, the effective channel length of the transistor becomes independent from the lithographic resolution. According to [Dude 97] an effective channel length as short as $0.1 \mu\text{m}$ can be reached with a $0.8 \mu\text{m}$ process.

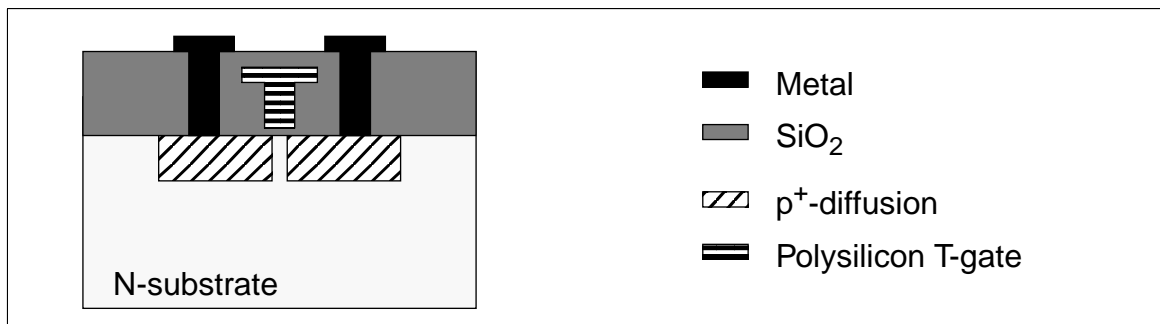


Figure 2-12: T-gate transistor

The reduced channel length results in a higher drive strength of the transistor which enables to design minimum width transistors. Since the area of the parasitic gate capacitor is determined by the channel length and the transistor width, the gate capacitance is dramatically reduced. Both effects, reduced channel length and minimum width transistors, result in increased performance and lower power consumption.

3D-CMOS

In 3D-CMOS an NMOS and a PMOS transistor are stacked above each other. The simplified structure of a 3D-CMOS transistor pair is depicted in figure 2-13 [Reti 96]. The junction of the n- and p-drain works in the tunnel breakthrough mode [Roos 93]. If the stacked

transistors are optimally utilized, the chip area can be reduced as much as 40% [Abou 98] which leads to higher performance and lower power consumption. On the other hand, the 3D-structure poses high demands on the routing tools because of the high contact density on the chip surface. Therefore, multiplexer based logic styles are to be preferred. They allow the stacked transistor pair to share their drain and gate contacts [Roos 93].

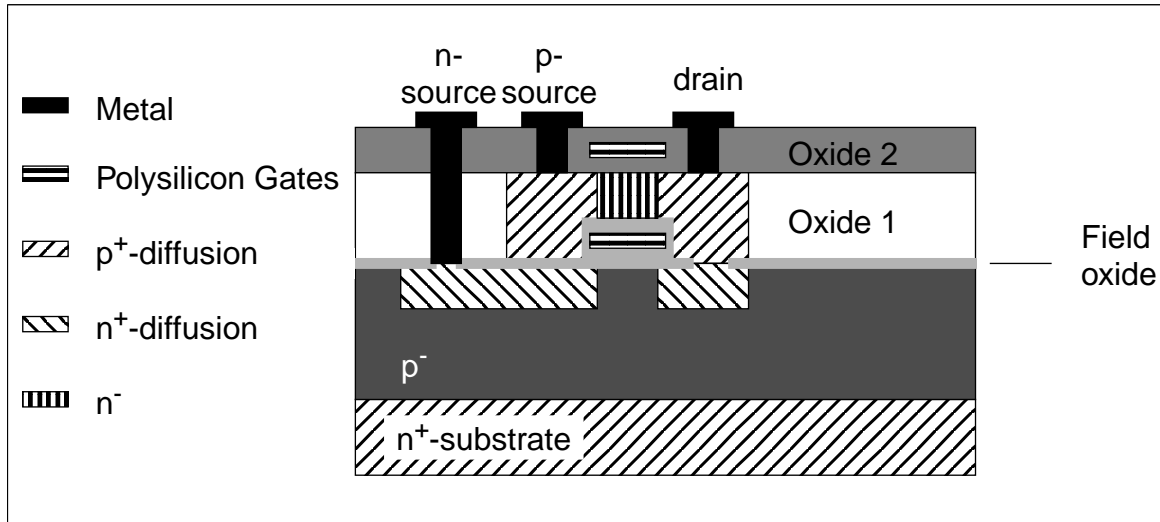


Figure 2-13: 3D-CMOS

2.2.2 Power Dissipation in CMOS Circuits

Power analysis and low power synthesis require a deep understanding of the physical sources of power dissipation in CMOS circuits. The physical phenomena must be described by mathematical models that allow for accurate and yet runtime efficient analysis and synthesis, even for large circuits. This problem can be addressed in two ways. The first one is a technological approach. It tries to extract simple equations from the highly nonlinear physical behavior of the electrical devices which form the circuit. The second approach is more empirical. It is preferred by some commercial tool providers due to its easier practical applicability [SyLi 96]. It models power consumption from an abstract view.

In the low power literature, the distinction between energy and power is not as strict as it should be. Of course, battery life and cooling effort directly depend on average power consumption. However, one can easily reduce the power consumption by slowing down the clock frequency, but the circuit performance decreases for the same amount. Hence, the figure of merit is rather given by the energy dissipation per operation or its equivalent, the *power-delay-product*. If we consider a design where the operation frequency f is fixed by the specification, then power and energy become interchangeable by the equation:

$$P = E \cdot f. \quad (2-5)$$

Technological View

From a technological point of view there are four sources of power dissipation in digital CMOS circuits. They are summarized in the following equation:

$$P = P_{sw} + P_{sc} + P_l + P_{stat} , \quad (2-6)$$

where P_{sw} denotes *switching power*, P_{sc} *short circuit power*, P_l *power caused by leakage current*, and P_{stat} *static power*. These four types of power consumption are discussed briefly in the sequel. For a more elaborate introduction the reader is referred to [Chan 95].

Figure 2-14 shows a static CMOS inverter. The load capacitance C_L concentrates all parasitic capacitances that are effected by the output of the inverter: the drain capacitances of T_1 and T_2 , wiring capacitances, and the gate capacitances of the transistors that are driven by the inverter. During the first time period, while the input is falling and the output is rising, C_L is being charged across T_1 by the supply voltage. If C_L is fully charged, it stores the energy

$$E = \frac{1}{2} C_L V_{dd}^2 . \quad (2-7)$$



Figure 2-14: CMOS inverter

It can be proven [Chan 95] that the same amount of energy is being dissipated in T_1 during the charging process. During the second cycle, while the input is rising and the output falling, C_L is being discharged and the energy that has been stored in C_L is being dissipated in T_2 . Thus, the total energy dissipated during a complete charge-discharge cycle equals $C_L V_{dd}^2$ and the average energy dissipated per output transition is given by equation 2-7. This type of energy is usually referred to as *switching energy* E_{sw} . The *switching power* P_{sw} is defined as

$$P_{sw} = \frac{1}{2} \alpha \cdot C_L \cdot V_{dd}^2 \cdot f_{CLK}, \quad (2-8)$$

where f_{CLK} denotes the clock frequency the circuit is operating at, V_{dd} the supply voltage, and α the average number of transitions per clock cycle, also called *toggle rate*.

Next, we will focus on the period τ while the input voltage is rising or falling (figure 2-15). During this time, neither T_1 nor T_2 are fully shut off. They can rather be considered as variable resistors which results in a short circuit current I_{sc} from V_{dd} to ground. Figure 2-15b illustrates the model of the inverter that is valid during τ .

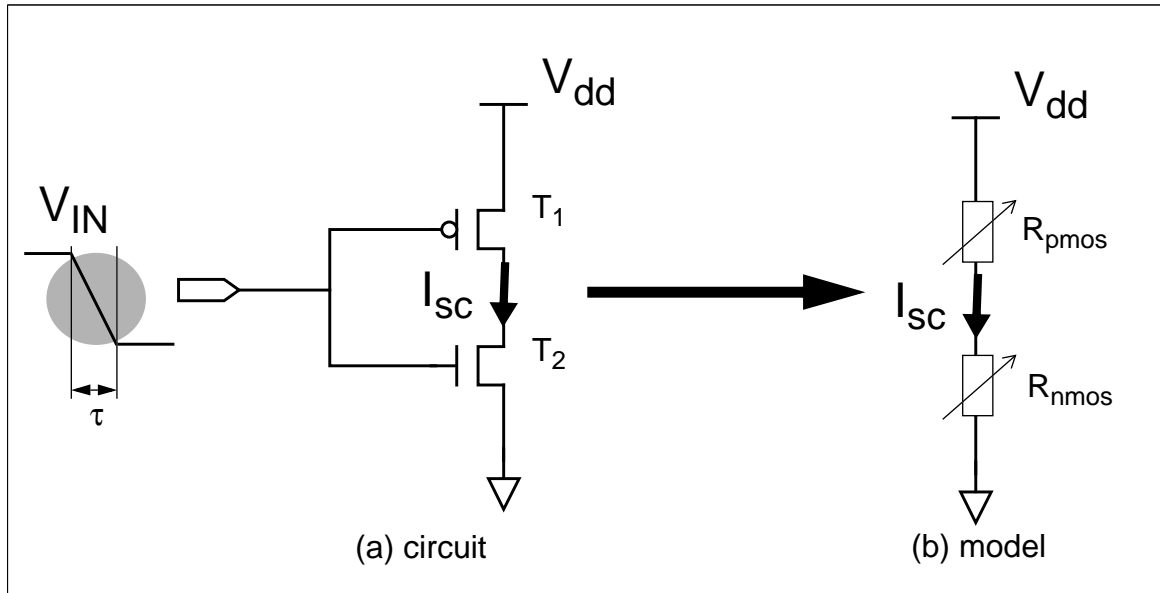


Figure 2-15: CMOS inverter during rising inputs

The short circuit energy e_{sc} of a single transition can be calculated by

$$e_{sc} = V_{dd} \cdot \int_0^T I_{sc}(t) dt \quad (2-9)$$

where $T=1/f_{clk}$ is a complete clock period. Clearly, I_{sc} is only non-zero during τ . The overall short circuit power P_{sc} is then given by

$$P_{sc} = \alpha \cdot e_{sc} \cdot f_{CLK} = \alpha \cdot V_{dd} \cdot \overline{I_{sc}}. \quad (2-10)$$

$\overline{I_{sc}}$ denotes the average short circuit current during a complete clock cycle. According to [Chan 95] it can be roughly estimated by

$$\overline{I_{sc}} = \frac{\beta}{12 \cdot V_{dd}} (V_{dd} - 2V_t)^3 \cdot \frac{\tau}{T}, \quad (2-11)$$

where β denotes the average gain factor of the PMOS-NMOS transistor pair.

Leakage current combines the non-ideal behavior of the transistors. It consists of the reverse diode current I_{rD} (figure 2-16a) and the sub-threshold current I_{sth} that flows between source and drain when the transistor is off (figure 2-16b).

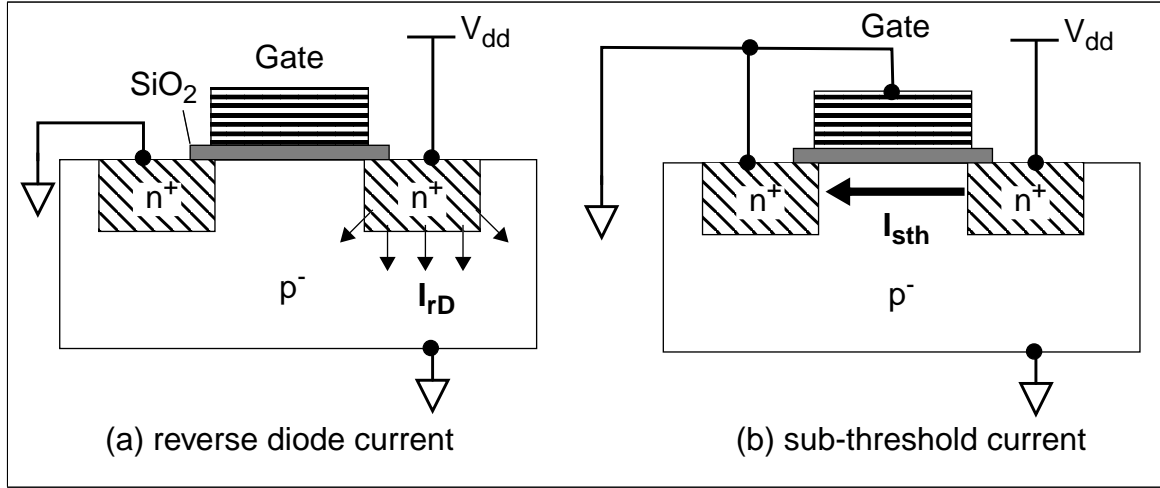


Figure 2-16: Leakage current

The reverse diode current is given by

$$I_{rD} = I_S \left(e^{-\frac{V}{V_T}} - 1 \right) \approx -I_S \text{ for } V = V_{dd} \quad (2-12)$$

where I_S is the reverse saturation current, V is the diode voltage. $V_T = kT/q$ is the thermal voltage, where k is the Boltzmann constant [Bron 87] and q is the charge of the charge carriers, usually $q = 1,609 \times 10^{-19} \text{ J}$. The current in the subthreshold region I_{sth} flows if the gate voltage V_{GS} is below the threshold voltage V_{th} ($V_{GS} < V_{th}$) and is given by

$$I_{sth} = I_{DS} = K e^{\frac{V_{GS} - V_{th}}{nV_T}} \left(1 - e^{-\frac{V_{DS}}{V_T}} \right) \approx K e^{\frac{V_{GS} - V_{th}}{nV_T}} \text{ for } V_{DS} \gg V_T \quad (2-13)$$

K and n are technology dependent, V_T is the thermal voltage, V_{th} is the threshold voltage, and V_{DS} and I_{DS} denote the drain source voltage and current, respectively.

Static power consumption is caused by static current paths in bipolar, pseudo-NMOS, or analog circuits. Figure 2-17 shows a pseudo-NMOS inverter. The PMOS transistor acts as a resistor. If the input is high, a static current from V_{dd} to GND causes the power dissipation P_{static} :

$$P_{static} = \frac{V_{dd}^2}{R_{pmos}}, \quad (2-14)$$

where R_{pmos} is the resistance of the pmos transistor. In a static CMOS design, static power dissipation does not exist since one of the two complementary transistors is always shut off during the steady state.

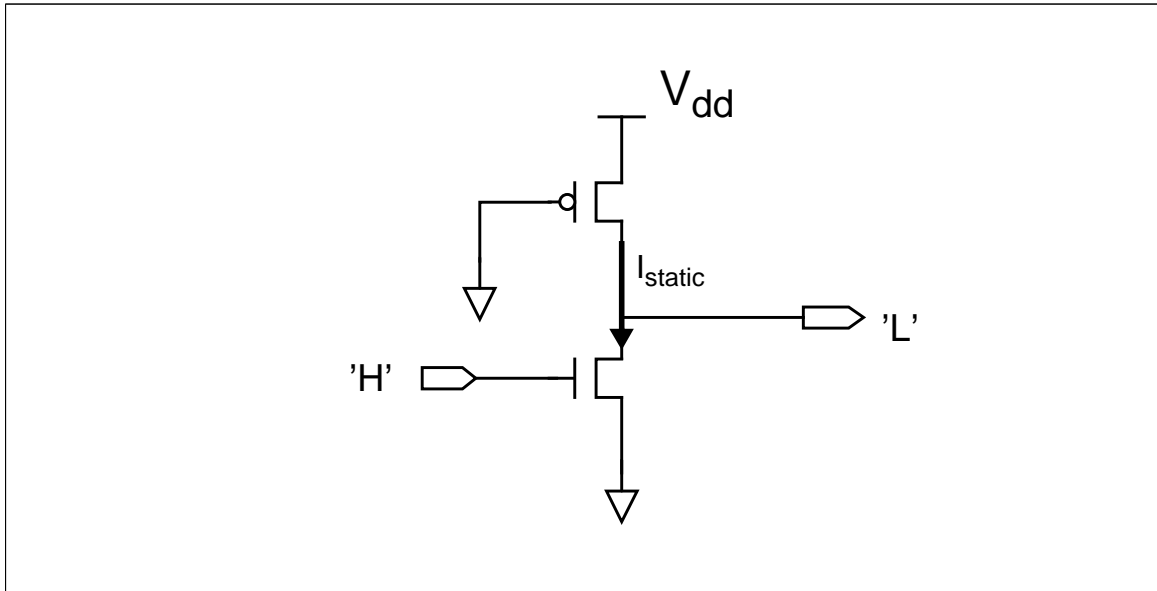


Figure 2-17: pseudo-NMOS inverter

The typical share of the different types of power dissipation to the overall dissipation varies widely among different publications in the literature. While most authors agree that power dissipation due to static and leakage current can be neglected, the figures for the typical share of dynamic and short circuit power vary widely. For a properly designed circuit, short circuit power accounts for below 10% according to [Chan 95] and for up to 20% according to [West 93]. If less care is taken for buffer sizing it maybe up to 50% according to [Turg 95] and [Nebe 97].

If leakage current is neglected, the overall power consumption of a circuit can be estimated as the sum of the switching and the short circuit power over all gates of the circuit:

$$P = \sum_{\text{all gates } i} P_{sw_i} + P_{sc_i} = V_{dd} \cdot f_{clk} \cdot \sum_{\text{all gates } i} \alpha_i \cdot \left(\frac{1}{2} C_{L_i} V_{dd} + \bar{I}_{sc_i} \right) \quad (2-15)$$

In many publications that focus switching activity estimation, even short circuit power is neglected for the sake of simplicity. This results in the well known formula 2-16.

$$P = \sum_{\text{all gates } i} P_{sw_i} = \frac{1}{2} \cdot V_{dd}^2 \cdot f_{clk} \cdot \sum_{\text{all gates } i} \alpha_i \cdot C_{L_i} \quad (2-16)$$

In the equations 2-15 and 2-16, two types of parameters can be observed. Static parameters like the load capacitance C_L or the short circuit current I_{sc} that can be derived from technological parameters or the circuit structure, and the dynamic parameter α_i . The latter does not

only depend on the circuit structure but also on the input patterns that are applied to the circuit. Thus, power estimation can be broken down into two subdomains: technological characterization of logic gates and switching activity estimation.

Abstract View

The abstract view is preferred by some commercial tool providers [SyLi 96] because of its simple implementation. It basically requires a straightforward modification of delay model implementations. The abstract view distinguishes only between three types of power dissipation: external power P_{ext} , internal power P_{int} , and static power P_{stat} .

P_{ext} combines the power dissipation of a gate that is caused by its load. All the non-static power that a gate consumes internally is called *internal power* P_{int} . Compared to the technological view, P_{int} corresponds to P_{sc} and P_{ext} to P_{sw} as long as one considers single stage cells like the inverter in figure 2-14. However, for cells with more than one stage, the switching power of internal nodes contribute also to P_{int} of the cell. In figure 2-18, C_{Lint} concentrates the input capacitances of stage two. It represents the load capacitance of the first stage but cannot be seen from outside the cell. Whenever the input switches, C_{Lint} is charged or discharged as well, thus contributing to the cell's internal power consumption.

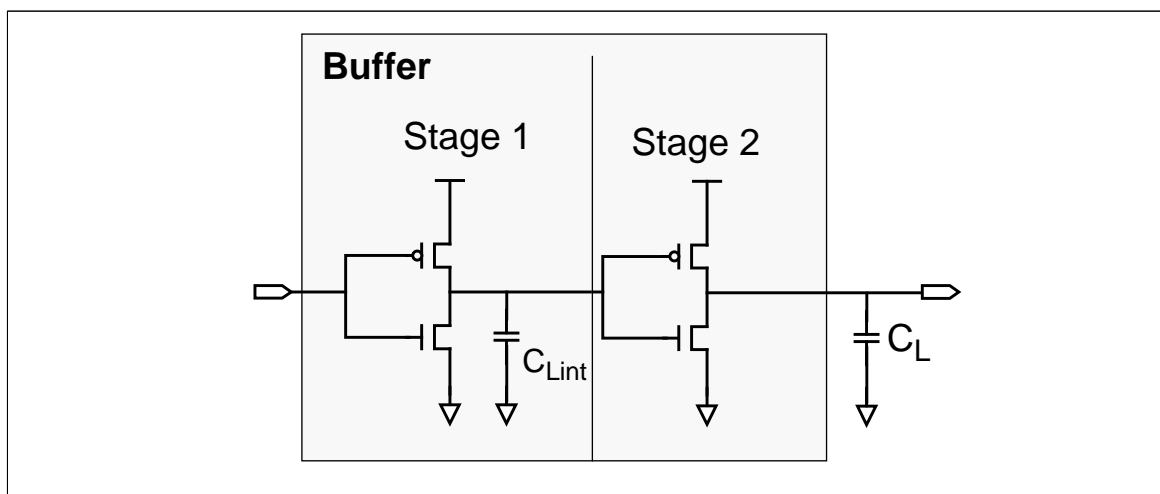


Figure 2-18: CMOS buffer

Glitches

Glitches are incomplete transitions of the output that are caused by short pulses at the input and the gate's inertia. Since the transition of the output from high to low or vice versa takes a finite amount of time, incomplete transitions may occur at the output as depicted in figure 2-19.

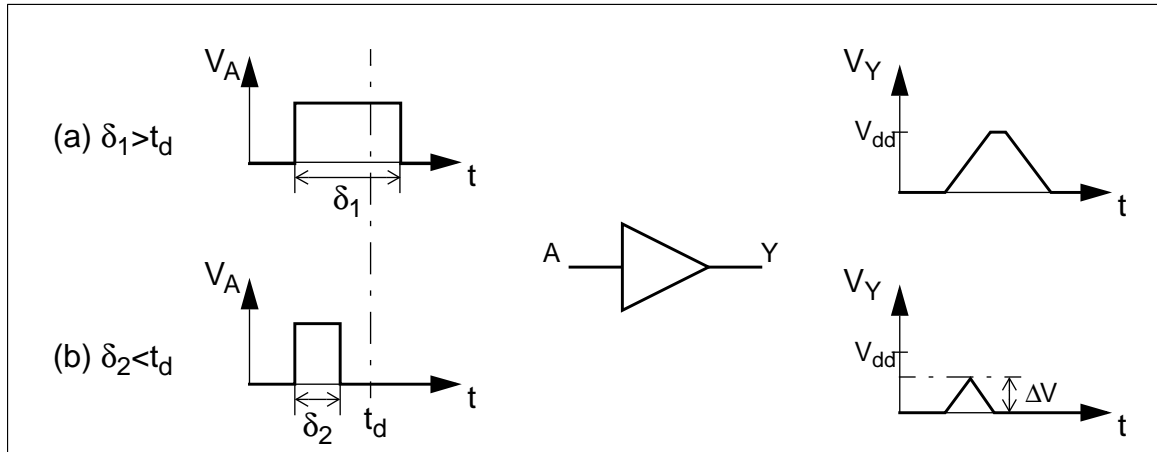


Figure 2-19: complete transition and glitch

In figure 2-19(a) the pulse width δ_I at the buffer input A is longer than the gate delay t_d . Hence, the output performs two complete transitions. But in figure 2-19(b) the pulse width δ_2 is shorter than t_d . Thus, the output Y rises only up to ΔV . The energy consumption of a glitch is given by [Rabe 98]

$$E_{\text{glitch}} = C_L \cdot V_{dd} \cdot |\Delta V|. \quad (2-17)$$

2.3 Delay Models

Accurate gate delay models play an important role during functional validation and verification of digital circuits. Even small inaccuracies, e.g. in the setup time of a flip flop, can turn malfunction into function which feigns correctness of an incorrect design. For power estimation, accuracy is not as crucial as for design validation since the figure of merit, the overall power dissipation, is a more integral value and thus less sensitive to small changes in the gate delays. Nevertheless, if the delay model is chosen to simple, the estimation result may become very inaccurate because some effects cannot be correctly modeled which are introduced by gate delays. Hence, understanding delay models, their effects, and limits is essential for accurate power estimation as well. A good overview over time models in connection with power estimation can be found in [Cong 96].

2.3.1 Zero Delay Model

The *zero delay model* (ZDM) is the most simple delay model. It neglects all gate delays. Hence, the finest granularity of time in the ZDM is one clock cycle. The ZDM tends to importantly underestimate power dissipation since it cannot cover the effects which are introduced by gate delays. Nevertheless, it maybe applied for a first rough estimation early in the design cycle when no gate delays are available [Schn 95] or during the prototyping of new simulation algorithms. Due to its simplicity, it allows to concentrate on the implementation of the new algorithm and it simplifies comparison to other approaches because the influences of implementation details and cell libraries are reduced. Figure 2-20 shows a circuit which was simulated with a ZDM. As soon as one of the inputs A or B takes a new value that effects the output Y , Y switches without any delay.

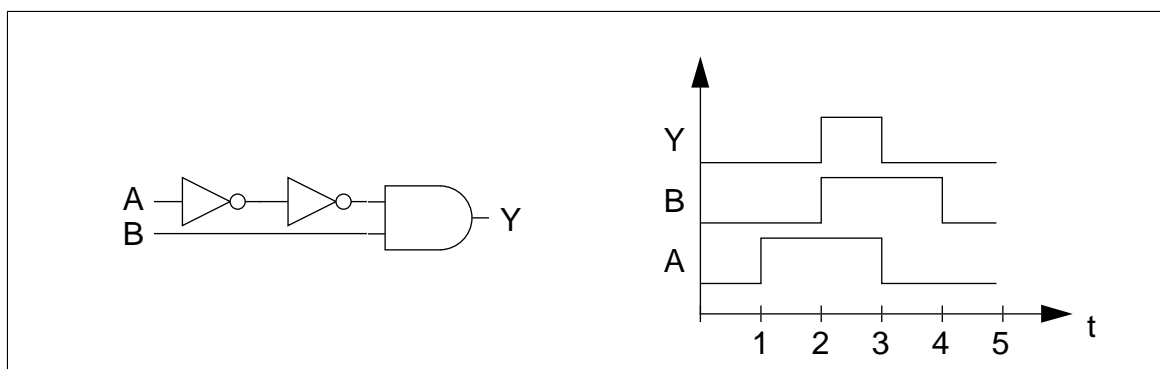


Figure 2-20: Circuit simulated with ZDM

2.3.2 Unit Delay Model

A first improvement of the ZDM is the unit delay model (UDM). In the UDM all gates are assumed to have an equal delay of one time unit. Although this is still a rather rough approximation of the real gate delays, it allows to model *hazards*. Hazards are unwanted signal transitions that are introduced by the gate delays. Figure 2-21 shows the same circuit as figure 2-20 with different stimuli. A_1 and A_2 denote the outputs of the two inverters. If the circuit was simulated with the ZDM, the output would not switch at all since one of the two inputs A or B would always be '0'. However, due to the UDM, both inputs of the AND-gate, A_2 and B , become '1' at time $t=3$. That causes the output Y to switch to '1' one time unit later.

Like the ZDM, the UDM can be used to obtain a first rough approximation of the power consumption before a design has been mapped to a technology and delay values become available. However, the UDM tends to overestimate power consumption because of the high hazard rate generated by this model.

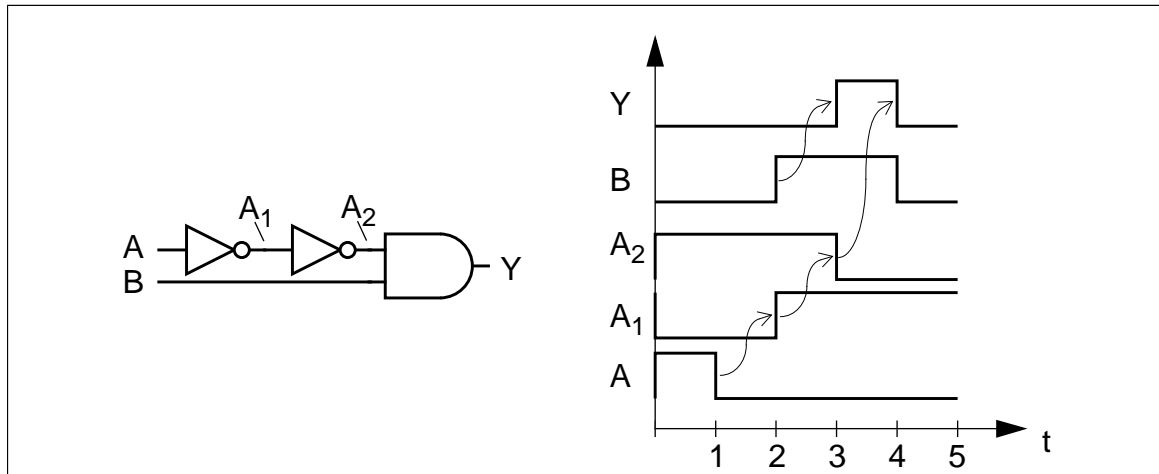


Figure 2-21: Circuit simulated with UDM

2.3.3 Real Delay Model

In the real delay model (RDM), a gate's delay is given by a real number. Depending on the model, this number may depend on the gate's load and other parameters. Thus, the RDM is the most flexible and accurate model. In some simulators, the delays are represented by integers instead of real numbers for efficiency reasons [SyLi 96]. If the time base is chosen properly, the accuracy of this type of simulators is comparable to those that use real numbers. Therefore, they will be also included into the class of real delay simulators. Beside hazards, RDM simulators are also capable to detect glitches which are filtered out in most implementations.

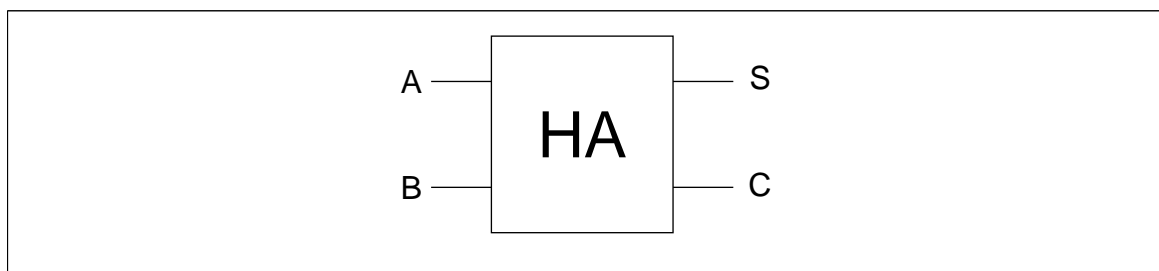


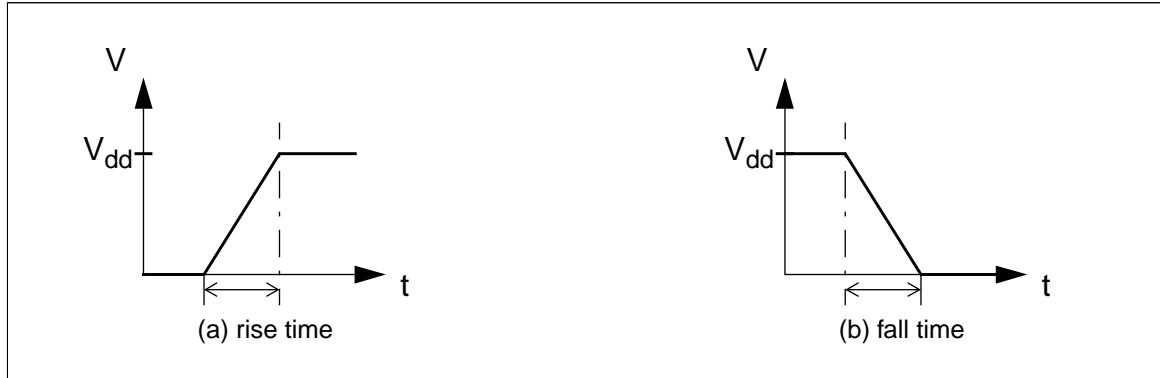
Figure 2-22: Halfadder

There exist several refinements of the RDM that are presented in the sequel. In the *cell oriented* model all cells of the same type have the same delay parameters, whereas in the *pin oriented* model, there exists a set of delay parameters from each input to each output, also called *timing arcs*. The halfadder of figure 2-22 has two inputs A and B and two outputs S and C . Thus, there are four possible timing arcs: A to S , A to C , B to S , and B to C . Furthermore, it is possible to define different delays for rising and falling transitions at the output.

In general, the gate delay is a function of the gate load C_L and of the *slew rate* of the input signal τ_{in} that causes the output to switch.

Definition 2-67: Slew Rate

The *slew rate* is the time it takes for the transition of a signal from an initial high/low voltage to a final low/high value. It is also called *rise* or *fall time* (figure 2-23).

**Figure 2-23:** Slew rate

Both, the gate delay t_d and the output slew rate τ_{out} , can be modeled with a two dimensional Taylor approximation:

$$t_d = \sum_{i=0}^n \sum_{j=0}^m k_{ij} \cdot \tau_{in}^i \cdot C_L^j \quad (2-18)$$

$$\tau_{out} = \sum_{i=0}^n \sum_{j=0}^m k'_{ij} \cdot \tau_{in}^i \cdot C_L^j \quad (2-19)$$

The constants k_{ij} and k'_{ij} are called *k-parameters*. The k-parameters are determined by analog simulations for various (τ_{in}, C_L) - pairs, followed by a parameter fit. Since analog simulation is computationally intensive and the number of simulation runs required for parameter determination increases with the order of the Taylor series, only the lower order versions of equations 2-18 and 2-19 are commonly used. The most common approximations are summarized in table 2-2.

n	m	Equation	Remarks
0	0	$t_d = k_{00}$	The delay is a constant of the gate. τ_{out} is not required.
0	1	$t_d = k_{00} + k_{01} C_L$	The delay increases linearly with the load capacitance C_L . τ_{out} is not required.
1	1	$t_d = k_{00} + k_{01} C_L + k_{10} \tau_{in}$ $\tau_{out} = k'_{00} + k'_{01} C_L + k'_{10} \tau_{in}$	k_{11} and k'_{11} are assumed to be 0.

Table 2-2: Common delay and slew rate approximations

Figure 2-24 illustrates the load dependence of t_d and τ_{out} .

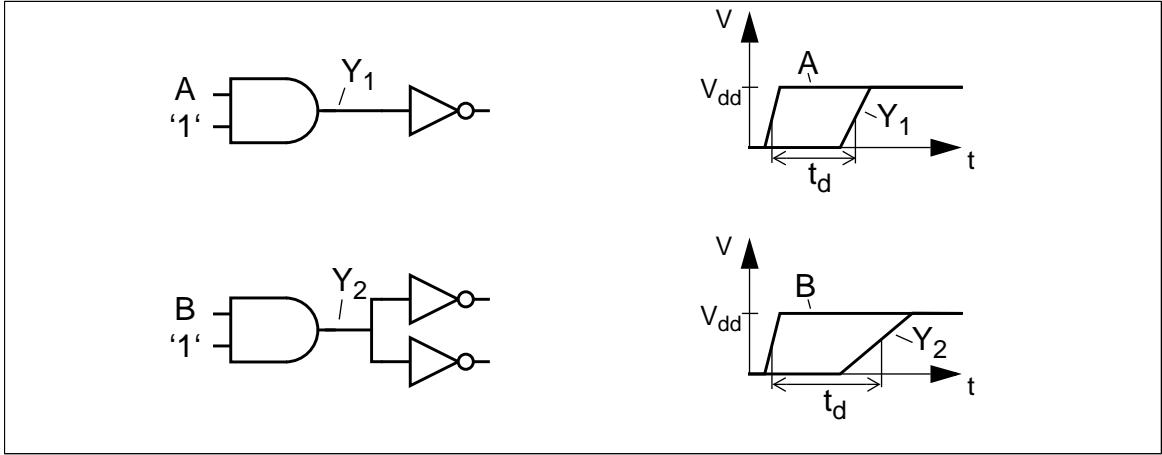


Figure 2-24: Load and slew rate dependence of gate delays

Instead of using higher order forms of equations 2-18 and 2-19, most commercial tools like Synopsys use a lookup table (LUT). This LUT stores t_d and τ_{out} for different (C_L, τ_{in}) pairs. Values in between the table entries are interpolated. Synopsys stores two LUTs per timing arc, one for rising and one for falling transitions. This method resembles a first order Taylor series with adjustable, but piecewise constant parameters. It is also called *piecewise linear model*. The accuracy of the LUT-based approach and the computational effort for LUT determination are comparable to the higher order Taylor approximations (equations 2-18 and 2-19) and finding their coefficients, respectively. The high memory requirement of the LUTs poses no major problem today.

Pulse Propagation Models

Most gate level simulators don't model glitches (equation 2-17). They use some simplifications instead. In the *transport delay model* even short pulses are treated like complete transitions. The inertia of the gates is neglected.

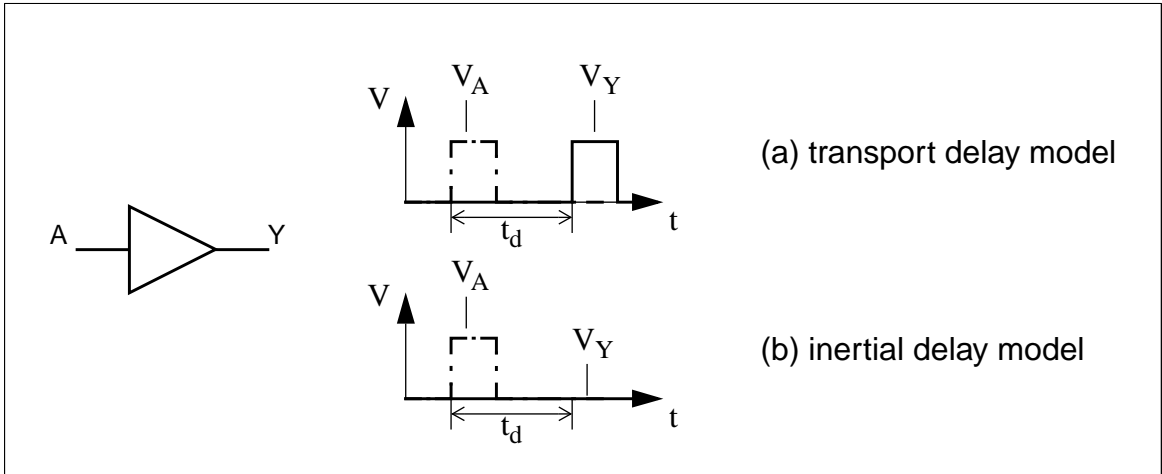


Figure 2-25: Buffer with transport and inertial delay model

In the *inertial delay model*, pulses that are shorter than the gate delay are filtered out. Figure 2-25 illustrates the difference between the two models.

While the transport delay model tends to overestimate the power consumption because of additional hazards, the inertial delay model is usually more accurate. Especially for modeling multistage gates, some tools allow a combination of transport and inertial delay model [Lehm 94]. Any pulse is delayed by the transport delay, but if it is shorter than the inertial delay, it is neglected. The idea behind this model becomes obvious when considering the AND-gate of figure 2-26. It consists of a NAND-gate with inertial delay t_{nand} , followed by

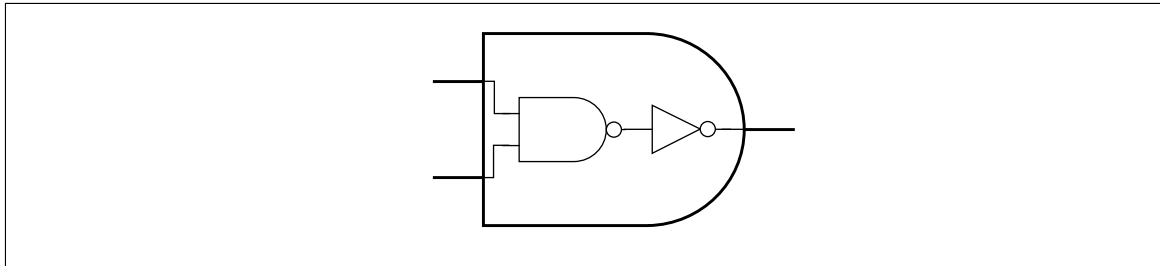


Figure 2-26: AND gate

an inverter with inertial delay t_{inv} . Since usually $t_{inv} < t_{nand}$ any pulse that is long enough to pass the NAND-gate ($\delta > t_{nand}$) will also pass the inverter. Hence, the overall delay of the AND-gate t_{and} is given by

$$t_{and} = t_{nand} + t_{inv}, \quad (2-20)$$

while the inertial delay t_{and}^i is given by

$$t_{and}^i = t_{nand}. \quad (2-21)$$

Power Minimization at the Circuit Level

Chapter

3

New technologies do not only provide new possibilities opened by higher integration density or higher performance. Sometimes they do also introduce new design constraints. This can be accepted as long as the advantages clearly dominate. Recently, a new technology has been proposed. It combines three technological steps that could only be realized separately before [Hipe 95]: T-gate transistors, SOI, and 3D-integration. It is intended to use this new technology in a gate array environment. This chapter investigates the design aspects of this novel approach. After demonstrating the design constraints imposed by the technology, several circuit techniques are investigated whether they can meet the constraints or not. It becomes obvious that only pass transistor styles are possible. Two experimental cell libraries were developed on schematic level in the frame of this thesis. They were used to implement several 32-bit carry-look-ahead adders (CLA) in order to compare delay and power consumption of the different circuit techniques.

The results that were obtained by simulating the demonstrators on transistor level, differ remarkably from those presented in previous publications. It is shown that the reason is most probably due to the specific constraints that are imposed by the gate array technology, which was used for the investigations here.

3.1 A 3D-CMOS T-Gate Technology

The technology which is depicted in figure 3-1 builds the foundation of the European project *Hiperlogic*¹ (*High Performance at Low Power Logic*). It is called *Hiperlogic technology* in the sequel. The goal of the project is to combine the advantages of three recent technological steps, which were already introduced in chapter 2.2.1: T-gate transistor, SOI, and 3D-structure. Through reduced area requirements and reduced parasitic capacitances which are combined with increased drive strength of the transistors, circuits with extremely low power consumption at high performance shall become possible.

1. Hiperlogic is supported by the European Commission under *HIPERLOGIC ESPRIT IV, Project No. 200023*

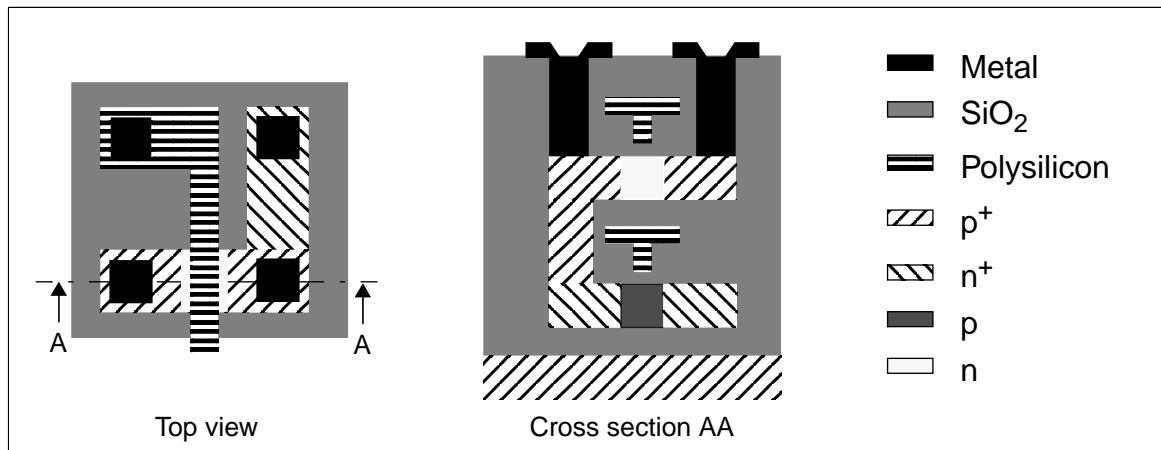


Figure 3-1: The Hiperlogic-Technology

3.2 Circuit Techniques

The *Institute for Microelectronics Stuttgart (IMS)* intends to use the Hiperlogic technology for low power applications in the frame of their sea-of-gates microarchitecture [Beun 88], [IMS 96]. This imposes specific constraints on the design style for circuits that are developed in the Hiperlogic technology. This section investigates which basic circuit techniques are best suited for the technological requirements.

3.2.1 Design Constraints in Hiperlogic

The Hiperlogic technology, the gate array design style and the low power requirements impose the following five constraints, which must be fulfilled by a suitable circuit technique.

Constraint 1: Equal Number of PMOS and NMOS Transistors

Only if both of the two stacked transistors in figure 3-1 are used, the advantages of the 3D-structure can be fully exploited.

Constraint 2: Minimum Width Transistors

The drive strength of the T-gate transistors is strong enough to design transistors with the minimum width that is allowed by the design rules of the technology [Dude 97].

Constraint 3: Contact Sharing

In gate arrays, the transistor contacts consume a considerable amount of area. This share will be even increased in the Hiperlogic technology for two reasons. If minimum width transistors are used, their area consumption is small compared to other technologies with wider

transistors, while the area requirements of the contacts remain constant. Furthermore, the contacts of the lower NMOS transistor require additional area at the surface. These problems can be reduced if the two stacked transistors share as many contacts as possible.

Constraint 4: Low Power Consumption

Circuit techniques with inherently low power consumption are to be preferred in order to meet the project goals. Especially the suitability for low V_{DD} operation plays an important role.

Constraint 5: Robustness

In a gate array technology, all transistors have the same fixed size. Individual transistor sizing is not possible. Therefore, functional robustness of the circuit technique to varying gate loads plays an important role.

3.2.2 State-of-the-Art in Circuit Techniques

Although static CMOS has been the quasi standard during more than a decade now, investigations on alternative circuit techniques have regained much attention in the low power design domain as indicated by the high number of recent publications, e. g. [KoBa 95], [Corr 96], and [Zimm 97]. Beside all its advantages, static CMOS has two major drawbacks. The logic function must be implemented twice: in the pull-down network and in the pull-up network [Farn 94]. Furthermore, only the slower PMOS transistors can be used for the pull-up network. Hence, they must be designed wider than the NMOS transistors and several PMOS transistors in series must be avoided. As a consequence, most alternative circuit techniques try to avoid or simplify the pull-up network. This section will give an overview over circuit techniques that are attractive for low power applications.

Static CMOS

Figure 3-2 shows an XOR in static CMOS. The advantages of static CMOS are well known. Unlike in the PMOS- or NMOS-technique which are the predecessors of static CMOS, static currents are negligible. Hence, the DC power consumption is quasi zero which results in a low overall power consumption. Static CMOS circuits are easy to design since they are functionally insensitive to varying gate loads. They become slower with increasing load but they do not fail completely. Furthermore, static CMOS is robust against noise. However, the major reason for its widespread use is the powerful CAD support that is available today for most common tasks in the design flow.

On the other hand, static CMOS has also some drawbacks which will be summarized here as well. The relatively high transistor count was already mentioned before. Some basic logic functions, like the XOR, result in complex implementations in static CMOS. Furthermore, the maximum number of inputs per gate is limited to four or less in most technologies. Cells with more inputs must be implemented through multilevel gates, which are more area and power consuming.

Static CMOS requires the same number of PMOS and NMOS transistors but the stacked

The statements about the power consumption of dynamic logic styles are quite controversial in the existing literature. While [Farn 94] points out the low power consumption caused by reduced transistor sizes and transistor count, other publications observe a much higher power consumption for dynamic styles than for static CMOS because of the unconditional pre-charge [KoBa 95], [Raba 96], [Zimm 97]. The latter consumes a significant amount of power even if the circuit inputs are stable.

Most dynamic logic styles like DOMINO CMOS avoid PMOS transistors. This violates constraint 1. In NORA [Farn 94], gates with NMOS network and PMOS pull-up, and gates with PMOS network and NMOS pull-down alternate each other. This results in almost equal numbers of NMOS and PMOS transistors. However, optimal use of the 3D-structure is hindered since, in general, the stacked transistors don't belong to the same cell, which violates constraint 3 and aggravates routing.

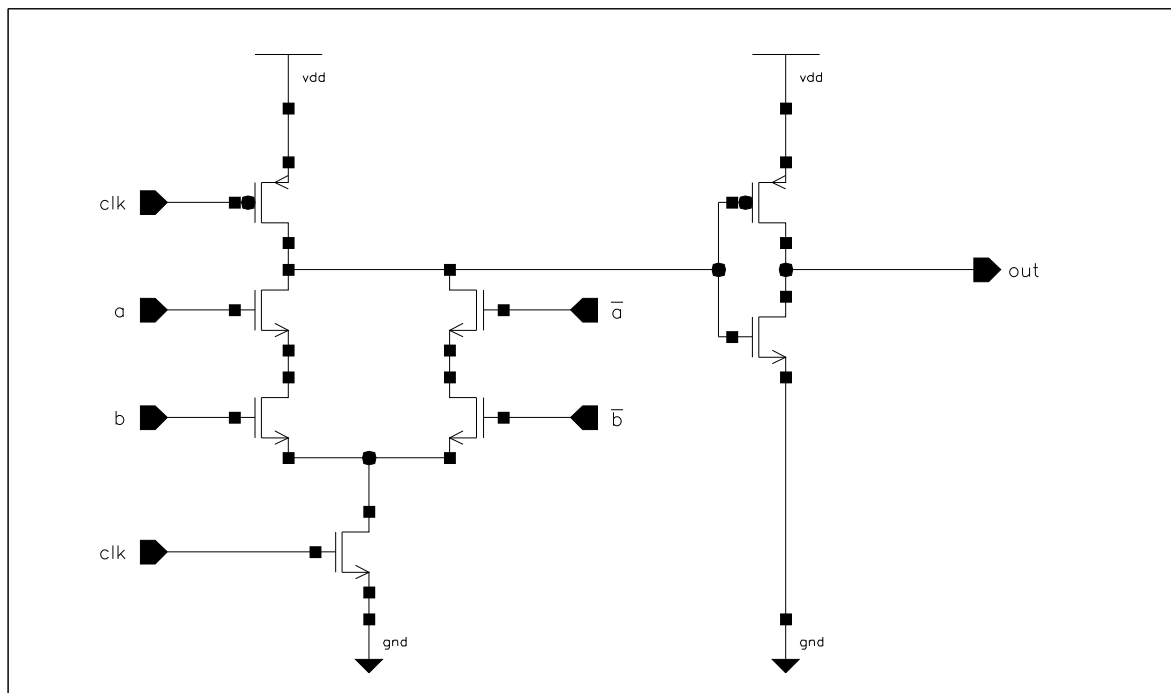


Figure 3-3: XOR in dynamic DOMINO logic

Mainly because of the last point and the noise sensitivity, but also because of the controversial situation concerning power consumption, dynamic styles were not investigated any further in this thesis.

Complementary Pass Transistor Logic

Complementary Pass Transistor Logic (CPL) is another approach to reduce the transistor count. Similar to dynamic logic styles, the logic is implemented through one transistor type only, usually the NMOS. The transistor count is even lower than in DOMINO CMOS. Therefore CPL results in the smallest implementations of all logic styles. For conventional supply voltages ($V_{DD} > 3V$) CPL has a low power consumption [KoBa 95]. However, as the supply

voltage decreases the effect of the voltage drop across the pass transistors becomes more and more significant. It decreases the gate performance and heavily increases power consumption because of leakage currents across the output inverter. Furthermore, the available noise margin is also decreased.

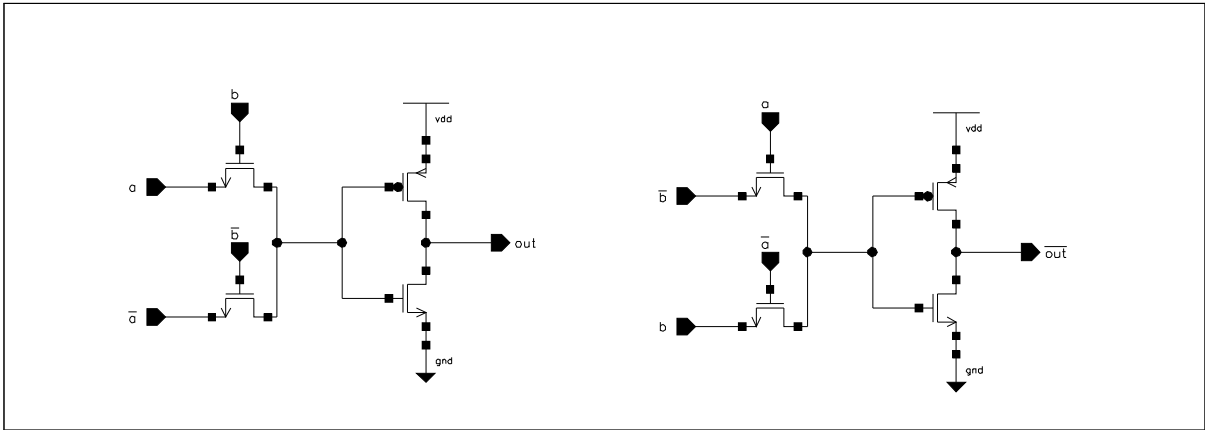


Figure 3-4: XOR in CPL

Like the dynamic logic styles, CPL prefers NMOS transistors. However, if one of the two pass transistors is replaced by a PMOS transistor it fulfills all the structural requirements of the 3D-structure. Figure 3-5 depicts an XOR gate in this alternative CPL (*aCPL*).

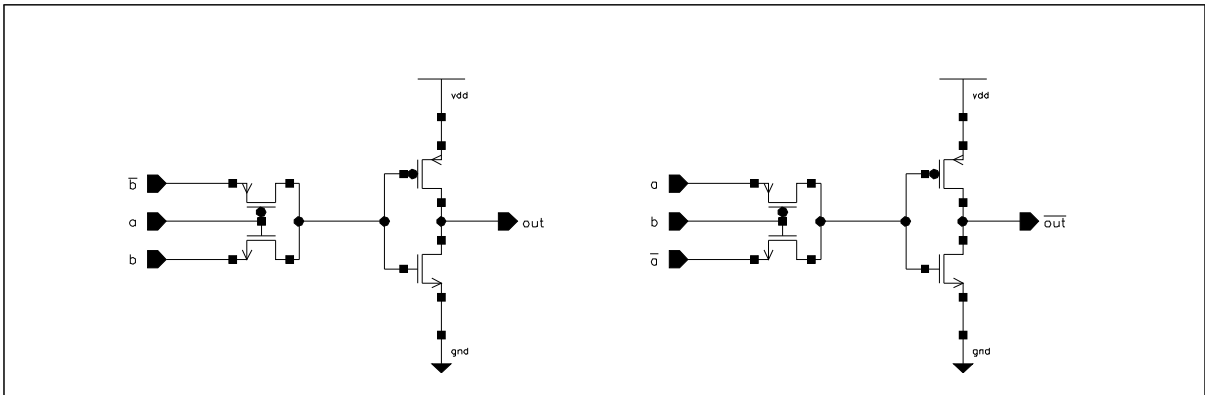


Figure 3-5: XOR in alternative CPL

The voltage drop across the pass transistors can be overcome if their threshold voltage V_{thp} is set to zero.

In order to estimate the effect of V_{thp} variations, a NAND gate with four inputs was simulated with $V_{thp}=0$ and $V_{thp}=0.5V$, Hiperlogic Spice transistor parameters, and $V_{DD}=2V$. The results are depicted in figure 3-6. Since the version with $V_{thp}=0$ outperforms the version with the conventional $V_{thp}=0.5V$ so clearly, the latter was excluded from any further consideration.

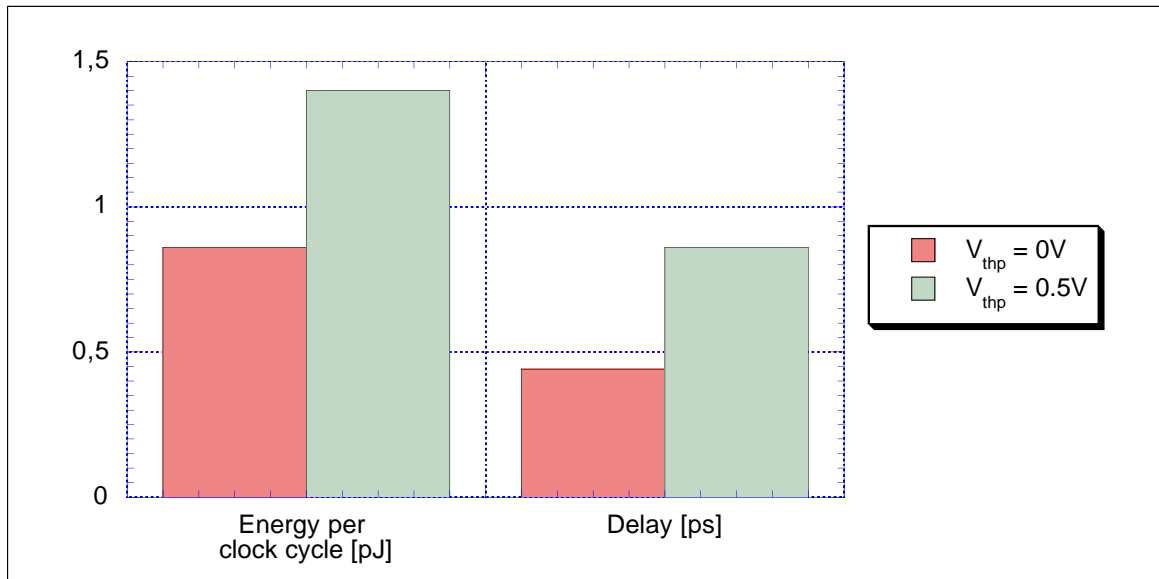


Figure 3-6: Energy and delay for different V_{thp}

Double Pass Logic

In [Suzu 93] and [Suzu 95] Suzuki et al. propose *Double Pass Logic (DPL)* as another pass transistor technique. DPL has attracted much attention in the recent literature [Pram 94], [KoBa 95], [Tcho 95], [Corr 96], and [Zimm 97]. Figure 3-7 shows an XOR in DPL. In DPL, the voltage drop across the path transistors is avoided by alternative signal paths across complementary transistors.

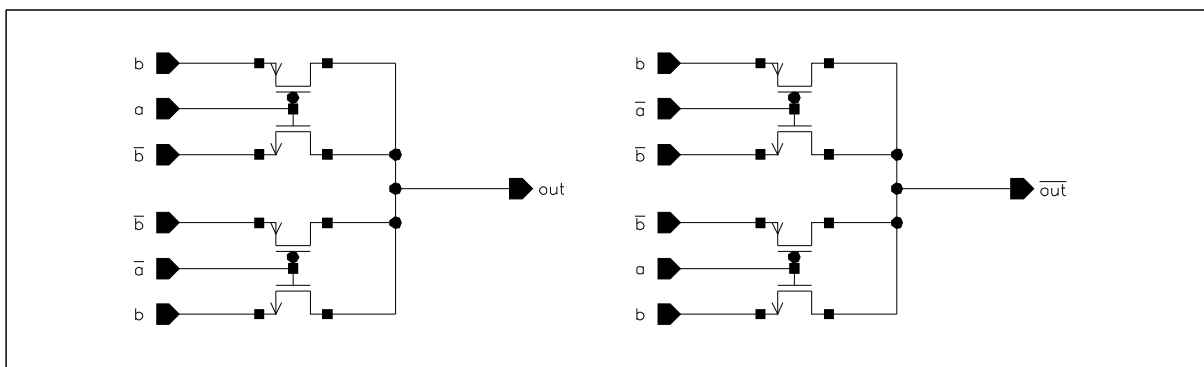


Figure 3-7: XOR in DPL

According to the literature, DPL results in very fast circuits since even complex logic functions like an XOR or a multiplexer can be implemented in a single logic stage. Furthermore, a good power-delay product is reported for DPL in [KoBa 95]. Structurally, DPL is very well suited for 3D-implementation because it requires equal numbers of PMOS and NMOS transistors, and the two stacked transistors can always share their gate and drain contacts.

These properties seemed attractive enough to further investigate DPL. But on the other hand there are also some drawbacks. Many basic logic functions require complementary signals

in their DPL implementation. This does not only aggravate routing but also importantly increase the transistor count. As illustrated in figure 3-7, DPL gates are not always connected to the supply voltage. As a consequence, the slew rates do significantly increase if several DPL stages are connected in series to each other without inverters to restore the slew rates. Furthermore, there is no high level CAD support available for DPL above the circuit level.

Other Techniques

There exist a few other techniques that were proposed for low power applications. Most pass transistor styles like SRPL, DCVSPG [Pram 94], or CVSL [West 93] had to be dropped since they are dominated by NMOS transistors and/or require careful transistor sizing. In general, dynamic techniques suffer from the high power consumption during the precharge phase. They were excluded as well.

3.2.3 aCPL and DPL versus static CMOS

As a result from the last section, only three circuit techniques can fulfill the requirements of the Hiperlogic technology: aCPL, DPL, and partly static CMOS. For a first comparison of the three techniques, three fairly simple gates were designed on circuit level in each technique and were characterized for power consumption and worst case delay. The three gates are: NAND, NOR, and XOR with four inputs each (*nand4*, *nor4*, and *xor4*, respectively). The *nand4* and *nor4* in static CMOS were implemented in one logic stage, the *xor4* in two stages. In aCPL and DPL all circuits require two logic stages. The DPL circuits were implemented without drive strength restoring inverters. Ideal voltage sources were used as input drivers. All circuits were simulated with a load $C_L=25fF$ which corresponds to the input capacitance of two inverters. The results that were normalized to static CMOS are summarized in figure 3-8.

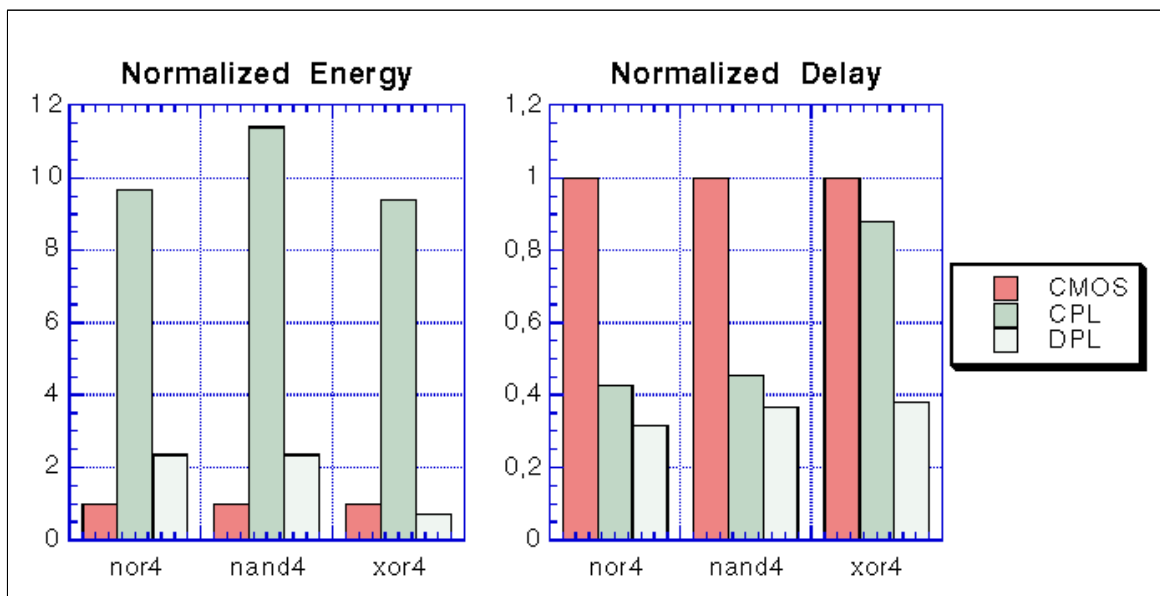


Figure 3-8: Comparison of CMOS, CPL, and DPL

Obviously, aCPL doesn't offer any advantage over DPL. It is slower and has a very high energy consumption. The latter is not surprising since the devices with $V_{thp}=0V$ have an extremely high static current. This technique was therefore also excluded from any further investigations.

However, the DPL experiments confirm the results from [Suzu 93] and [KoBa 95]. They indicate an interesting power-delay trade-off compared to static CMOS. Moreover, for complex gates like the *xor4*, DPL is faster and has a lower energy consumption at the same time. The DPL experiments are quite idealistic though. The inherent buffer problem of DPL has been avoided so far. All circuits were driven by ideal voltage sources. Thus, further more realistic experiments are indispensable.

3.3 A Double Pass Logic Cell Library

In order to compare the properties of DPL and static CMOS more thoroughly, two small libraries were designed on circuit level. Both libraries use the following basic parameters:

- The NMOS transistors have the minimum width that is allowed by the design rules of the Hiperlogic technology: $w_{nmos}=2.2 \mu m$ [IMS 97].
- In accordance to the $0.8 \mu m$ Gate Forest technology [IMS 96], where the Hiperlogic technology was derived from, the PMOS transistor was designed 50% wider than the NMOS in order to partly compensate the reduced mobility of the charge carriers. Thus $w_{pmos}=3.3 \mu m$.
- Both libraries were designed for $V_{DD}=2V$.

All cells were implemented in DPL and static CMOS. In the following, only the DPL library is presented in more detail. A detailed description of the static CMOS cells was omitted because they are state-of-the-art. It can be found in [Gers 97].

3.3.1 Library Specification

Figure 3-9 depicts the basic cell that was used to design all logic and sequential functions of the DPL library. It implements a 2:1 multiplexer. In fact, according to Shannon's expansion theorem (theorem 2-1), any logic function can be decomposed into 2:1 multiplexers. The base cell of figure 3-9 fulfills all the topological constraints postulated in section 3.2.1. Both gates as well as both drains of the stacked transistor pair are connected to each other. Thus, the number of contacts per transistor pair is reduced from six to four. Furthermore it guarantees equal numbers of PMOS and NMOS transistors.

Using this base cell, several basic logic and sequential cells were designed. Those are in particular:

- Inverter, buffer.
- NANDs and NORs with two, three, and four inputs.
- 2:1 multiplexer, 2-input XOR.

- D-latches and D-flip flops in five different variations: active low, active high, and with several variations of set and reset inputs.

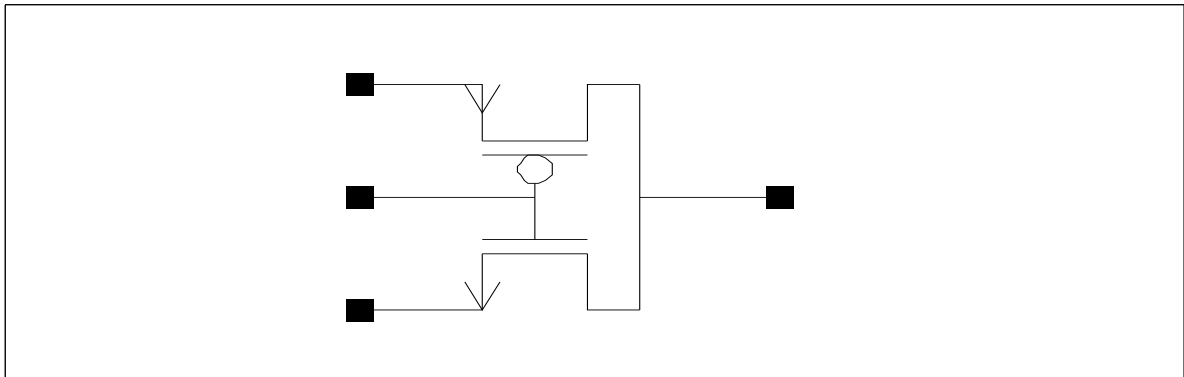


Figure 3-9: The DPL base cell

A detailed presentation of the schematics of all cells would exceed the frame of this work. It can be found in [Gers 97].

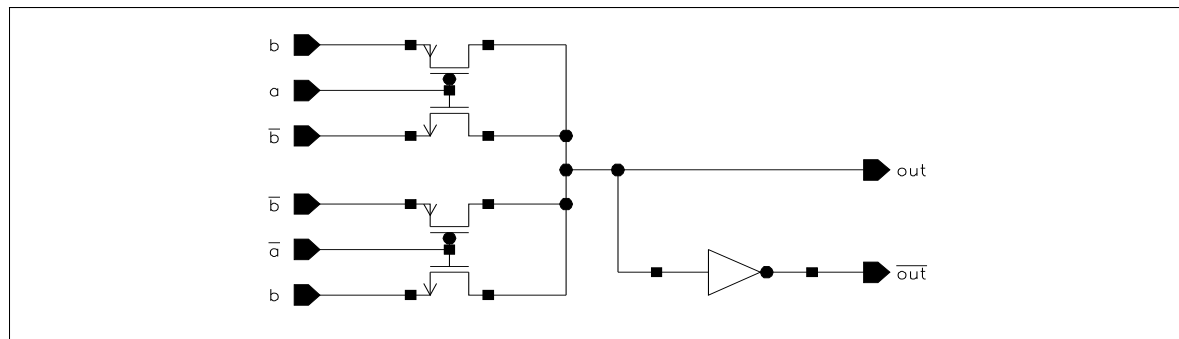


Figure 3-10: XOR in Low Power DPL

It has already been stated in section 3.2.2 that most DPL gates require complementary inputs. Therefore, all cells were designed such that they yield complementary outputs. In general there are two possibilities for logic cells to realize the complementary output signals. The parallel structure of figure 3-7 has been called *dual rail DPL* or *fast DPL* in [KoBa 95] since both signals pass only one logic stage. Another possibility is depicted in figure 3-10 where an inverter is used to realize the complementary output \overline{out} . \overline{out} passes two logic states but the circuit of figure 3-10 requires only six transistors instead of eight and consumes less power. Therefore, this option has been called *low power DPL* in [KoBa 95]. Any cell can be negated just by switching the output signals because each logic gate provides complementary output signals.

Sequential cells like the D-latch, which is depicted in figure 3-11, naturally yield the complementary outputs. Thus, there exists no distinction between *low power DPL* and *dual rail DPL* for the sequential cells.

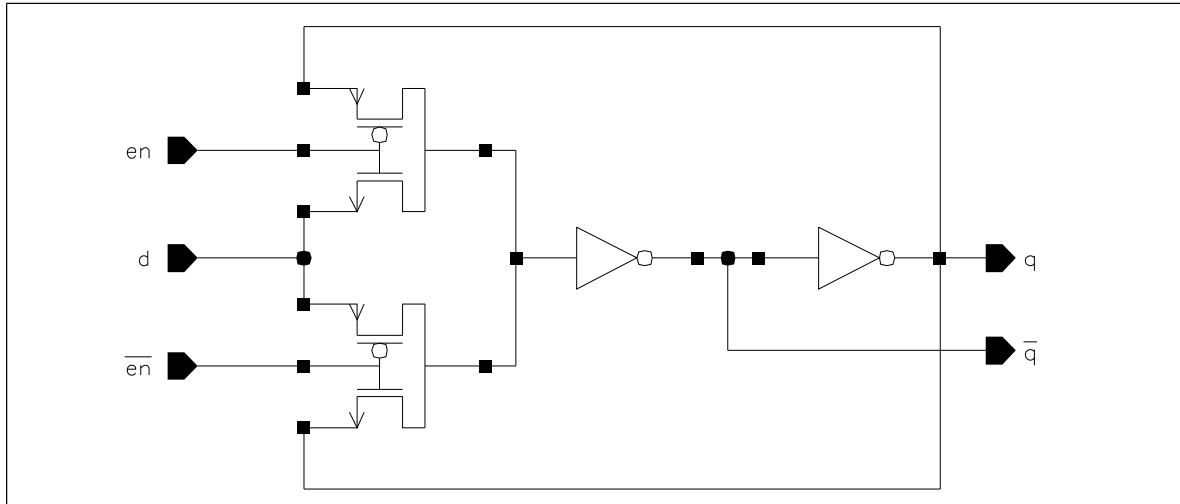


Figure 3-11: DPL D-Latch

Beside the output inverters of the low power versions, no additional, slew rate restoring inverters were inserted into the basic cells. This problem will be discussed in the next section.

3.3.2 Buffering

The basic logic gates in DPL are not always connected to V_{DD} and to GND (see figure 3-7). Instead they propagate directly the input signals to the outputs. While full voltage swing is guaranteed through complementary paths, the drive strength and the slew rates of the output signals decrease if several DPL gates are connected in series. Hence, slew rate restoring inverters need to be inserted. They will be called *buffers* in the sequel. The buffers can be put either at the outputs of a cell or at its inputs, as depicted in figure 3-12.

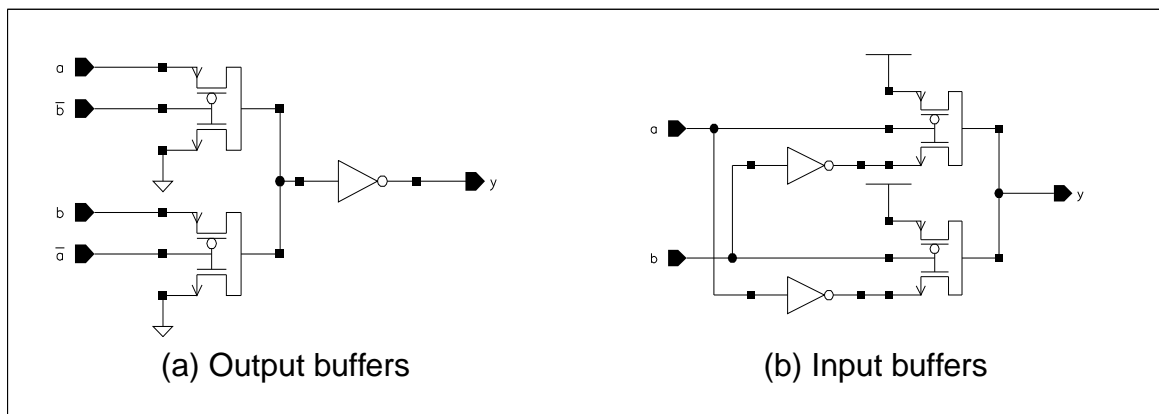


Figure 3-12: Nand2 in DPL with output and input buffers

In combination with the *low power DPL* approach (figure 3-10), output buffering results in the smallest DPL implementation if special care is taken for inserting the buffers. The extra output buffers can be omitted if an output is exclusively connected to transistor gates in the

following stage. Thus, many inverters can be avoided. E. g. if the gate in figure 3-12a is to be driven by another gate, only outputs which drive inputs a or b have to be buffered, but not those that drive \bar{a} or \bar{b} .

The circuit in figure 3-12b with buffered inputs has an interesting property. It doesn't require complementary input signals. Hence, additional output inverters or dual rail implementations can be avoided. Furthermore, the primary inputs are connected to transistor gates only. This simplifies the characterization of the cells for state-of-the-art logic synthesis and analysis tools. Such cells behave just like any static CMOS cell since the output load cannot be "seen" from the input pins and the input capacitances don't depend on the input pattern of the cell.

3.3.3 Demonstrator Circuits

Realistic comparison of static CMOS and the different DPL strategies is not possible on the basis of basic logic gates only. Too many idealistic assumptions must be made. Specific conditions and side effects which are symptomatic for each circuit technique can only be taken into account if larger circuits are considered. In accordance with [KoBa 95], [Corr 96], and [Zimm 97], 32-bit carry-look-ahead adders (CLA) [Patt 96] were chosen as demonstrator circuits.

Beside an implementation in static CMOS (*cadder*) the following DPL variations were designed:

1. *dadder*: This CLA is based on low power DPL and uses the inverter avoiding strategy that was described in section 3.3.2.
2. *dbadder*: It is also based on low power DPL. But in contrast to the *dadder*, all gate outputs are buffered by inverters.
3. *dsadder*: The *dsadder* uses the input buffers as described in section 3.3.2.
4. *ddadder*: The *ddadder* is similar to the *dadder* but uses the dual rail implementation of figure 3-7.
5. *ddbadder*: The *ddbadder* is the dual rail pendant to the *dbadder*.
6. *madder*: The *madder* combines DPL and static CMOS. The output inverters in DPL are useless from a logic point of view. They are only required for slew rate regeneration. In the *madder* (*mixed adder*) the inverters were replaced by static CMOS gates like NANDs or NORs wherever possible. Thus, they perform a logic operation while they restore the slew rates at the same time.

In order to preserve a fair comparison of the different circuit techniques and to account for the gate array approach, no further optimizations like buffer sizing or specific logic optimizations were performed. Figure 3-13 depicts the transistor counts of the different CLA implementations.

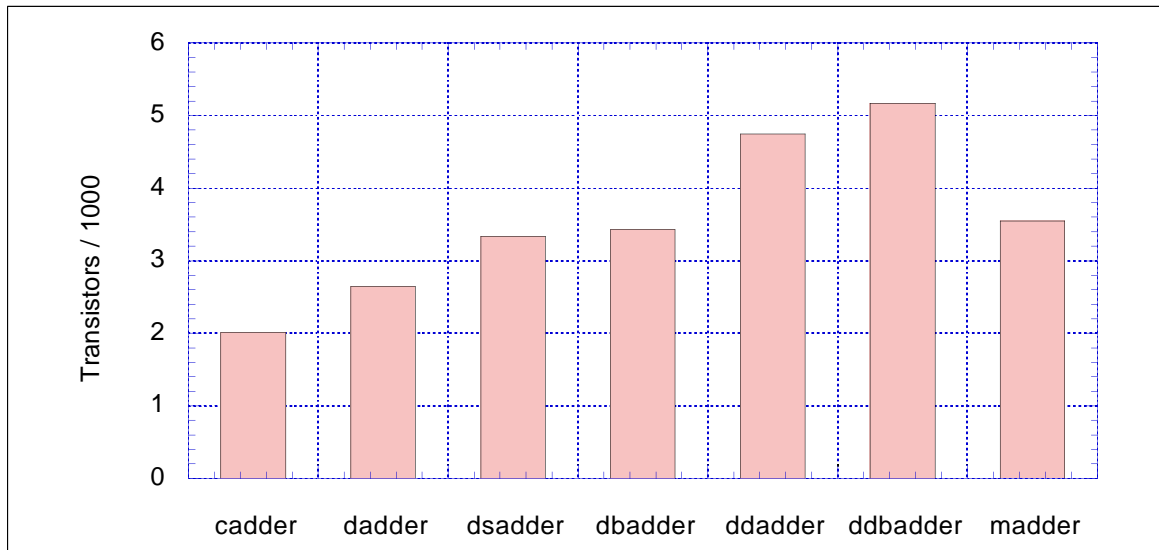


Figure 3-13: Transistor counts of the 32-bit CLAs

3.4 Simulation Results

The simulation results are based on the latest Hiperlogic transistor parameters that were available (appendix A) and the *Spectre* [Kund 95] analog simulator from *Cadence* [Cade 95]. Unfortunately, the transistor parameters model only the T-gate bulk effect. SOI and 3D-CMOS parameters were not available. Wiring capacitances were not taken into account since the simulations were performed on schematic level. A set of 59 test patterns was generated such that the critical path was stimulated for any CLA implementation. The power consumption values are based on the same set of test patterns. The simulation results in terms of the worst case delay and the power consumption are depicted in figures 3-14 and 3-15.

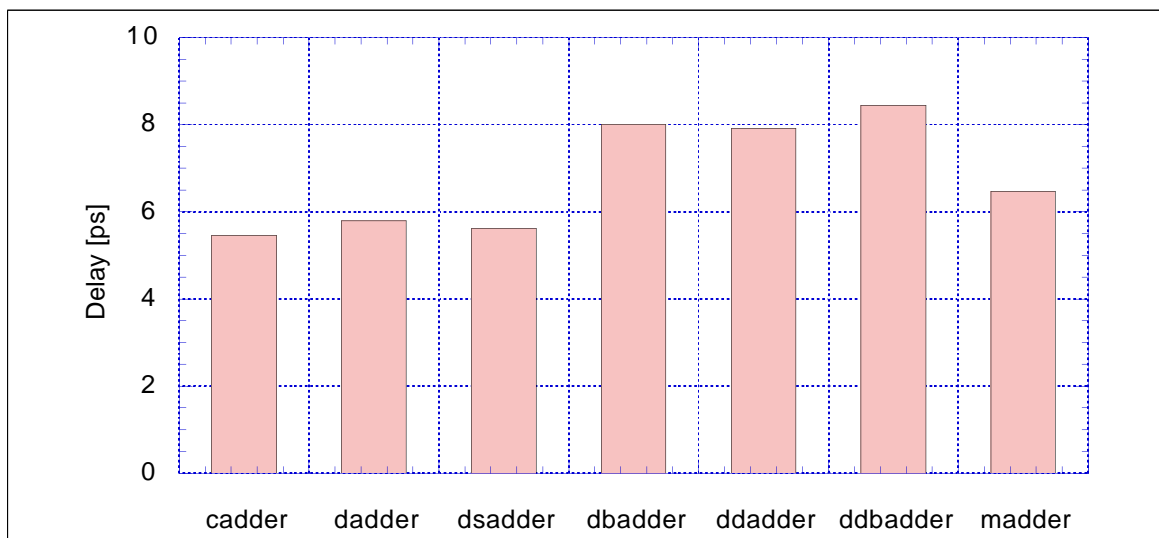


Figure 3-14: Worst case delays of the 32-bit CLAs

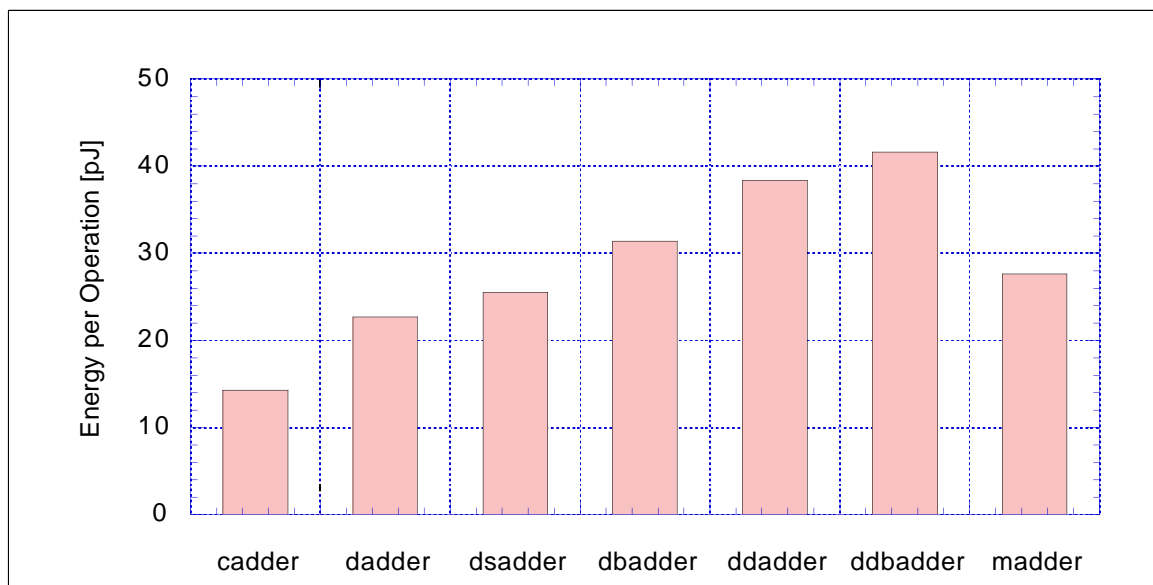


Figure 3-15: Energy consumption per operation of the 32-bit CLAs

Obviously, the static CMOS implementation outperforms all DPL versions in terms of worst case delay and, even more clearly, in terms of power consumption. It is a little surprising that the *dadder* is not only the DPL adder with the lowest power consumption, but also the fastest one. Obviously, the much lower cell loads in the *dadder* compensate the speed advantage that had been assumed at a first glance for the dual rail implementations. The mixed implementation (*madder*) also doesn't offer any advantage over static CMOS or the *dadder*.

Comparison to Other Publications

The results that are depicted in figure 3-14 and figure 3-15 differ remarkably from those that were presented in [KoBa 95] where DPL was superior to static CMOS in both, the worst case delay as well as the energy consumption. The major difference between Ko et al.'s investigations and the experiments that were obtained in this work is that Ko et al. used a full custom design style. They judiciously scaled all the transistors; they used wider ones in the critical path and smaller ones in noncritical parts of the design. Hence, they can take full advantage of a property which is specific to all pass transistor logic styles: if the drive strength of a DPL gate is to be increased, it is sufficient to enlarge the transistors in the output inverters and to keep the pass transistors small, whereas in static CMOS all transistors must be enlarged. Thus, DPL implementations can result in smaller, more energy efficient designs. In the Hiperlogic technology, this advantage cannot be exploited for two reasons. In a gate array technology, the prefabricated transistors prohibit individual transistor scaling. Further, the drive strength of the minimum size transistors is sufficient in most cases because of the T-gate. Hence, minimum width transistors can also be used in static CMOS, which diminishes one of the advantages of DPL.

In [Zimm 97] Zimmermann et al. used similar constraints like the ones that were applied in this work. But they allowed the transistors of their library cells to be scaled in predefined steps. However, individual scaling for each cell instantiation was not allowed. Their results are comparable to those presented here.

In order to “simulate“ Ko et al.’s driver sizing and so to estimate the potentials of DPL compared to static CMOS, the following experiment was carried out. 4-bit CLAs¹ in static CMOS and the most promising DPL version (*dadder*) were simulated with varying transistor widths. They reached from 1-5 times the original widths. In static CMOS all transistors were scaled, in DPL only those in the output inverters. The results are depicted in figure 3-16 where w_{CMOS}/w_{DPL} is the scaling factor mentioned before. Obviously, the delays remain almost constant, with a small decrease for DPL at the beginning. This justifies the assumption that minimum width transistors are sufficient in most cases. The energy consumption per input pattern on the other hand increases linearly for static CMOS and for DPL, where DPL has a higher offset but a lower gradient. The break even point, where both lines intersect, is approximately at $w_{CMOS}/w_{DPL} = 3.5$.

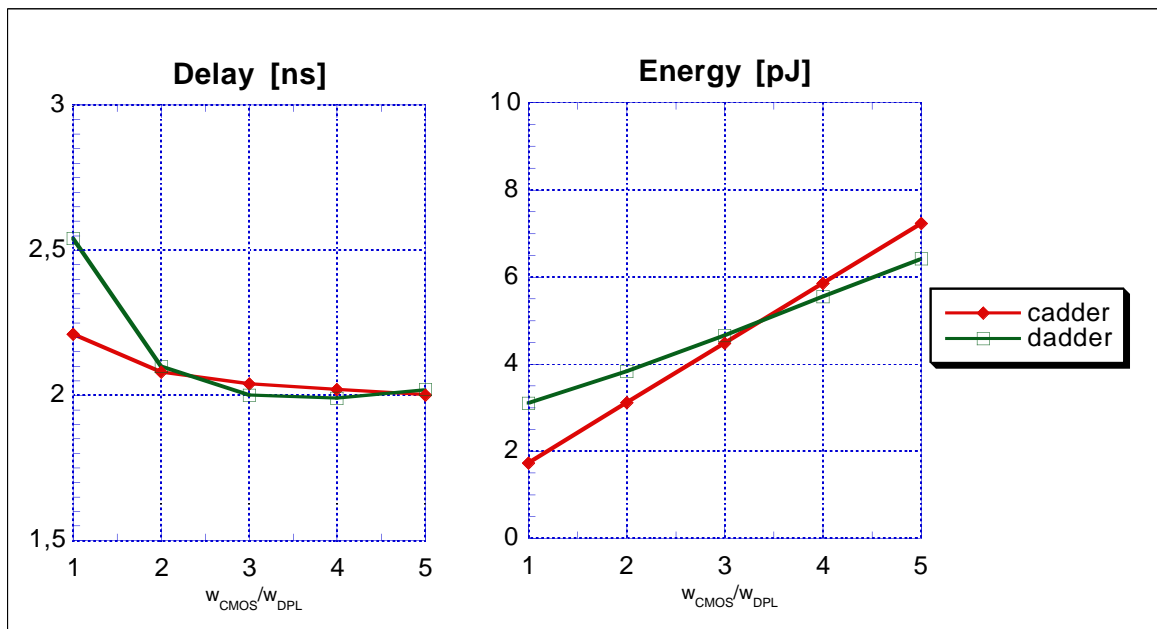


Figure 3-16: DPL versus static CMOS with varying driver sizes

While this experiment is only a rough approximation of a full custom approach with optimum driver sizing, it clearly shows that the choice of the best circuit technique in terms of power consumption and performance depends heavily on the design style and specific technological constraints, e.g. whether minimum width transistors are applicable or not.

1. For these experiments smaller circuits were chosen for practical reasons. The simulation times of a single 32-bit CLA came close to one day on the available hardware.

3.5 Summary

The Hiperlogic technology combines three recent technological steps: 3D-structure, T-gate transistors, and Silicon on Insulator (SOI). This chapter dealt with the question which circuit technique optimally supports this new technology in a gate array design environment, particularly for low power applications. From all the circuit techniques that were investigated, only DPL and a modified form of CPL (aCPL) could fulfill the structural constraints. Simulations with some basic circuits revealed very quickly that aCPL had to be rejected because of its high energy consumption. In order to more deeply investigate DPL, two cell libraries were designed, one in DPL and a similar one in static CMOS. The properties of both techniques were compared by 32-bit CLAs that were implemented on the basis of these libraries. Because of the need for complementary signals and the buffer problem which is inherent to DPL, six different DPL implementations were investigated. Analog simulations of the CLAs on circuit level have shown that none of the DPL CLAs can outperform the static CMOS implementation. While some DPL variations are comparable to static CMOS in terms of the worst case delay, even the best DPL implementation consumes 50% more energy than static CMOS. This left the question why other researchers came to different results. Experiments with some smaller circuits and varying transistor sizes show that DPL offers high optimization potentials through efficient driver sizing. However, this can only be exploited in full custom approaches.

If the Hiperlogic technology is to be used in a semi custom environment, static CMOS yields much better results. However, static CMOS requires that the two stacked transistors can be individually contacted which results in a large area overhead since the three contacts of the buried NMOS transistor must be made independently accessible at the chip's surface. The overhead can only be avoided if the gate array approach is given up in favor of a standard cell technology. In that case, very encouraging results were presented in [Abou 98].

Zero Delay Power Estimation in Combinational Circuits

Chapter

4

Capacitive and short circuit currents are the main sources for power dissipation in digital circuits. Capacitive current dominates in most circuits and accounts for 80-90% of the overall power consumption. But no matter whether short circuit current is taken into account or not, switching activity estimation is a crucial step in the overall power estimation procedure. Thus, the rest of this work deals with the question how to efficiently and yet accurately estimate switching activity. The focus is put on purely combinational circuits. The chapter starts with an overview over power estimation methods that can be found in the recent literature. Most of the algorithms which are presented in the following are limited to the zero delay model (ZDM). Despite its inaccuracy, the ZDM methods represent the core of any simulator with more sophisticated delay models.

Most of the power estimation methods belong to one of two categories: logic simulation or probabilistic simulation. While power estimation by logic simulation is a straightforward extension of common, commercially available logic simulators [SyPo 96], probabilistic methods estimate power dissipation in one single analysis run by propagating probabilities through the circuit. The advantages and drawbacks of both categories are discussed. Although all the methods that are presented in this chapter can only be applied on purely combinational circuits, a special section comments on the relation between switching activity estimation for combinational and sequential circuits. It is shown that power estimation for sequential circuits heavily relies on the methods for combinational circuits plus some additional algorithms for state probability estimation.

After this overview over the state of the art in signal activity estimation, a new, set based approach is presented. It can be classified between logic and probabilistic methods. The simulation results reveal that this set based approach performs particularly well on sequential, and small and medium sized combinational circuits. For larger circuits, an extension is proposed that enables an efficient speed-accuracy trade-off. This is demonstrated by experiments on several benchmark circuits.

4.1 Preliminaries

Figure 4-1 depicts a combinational circuit with input vector \underline{X} and output vector \underline{Y} . The Boolean function F transforms the input vector to the output vector. In a combinational circuit, F has neither feed back loops nor any memories like flip flops or latches. Thus, $\underline{Y}(t)$

depends only on the current input vector $\underline{X}(t)$. Therefore, the time parameter t can be omitted unless the focus is put on temporal relations.

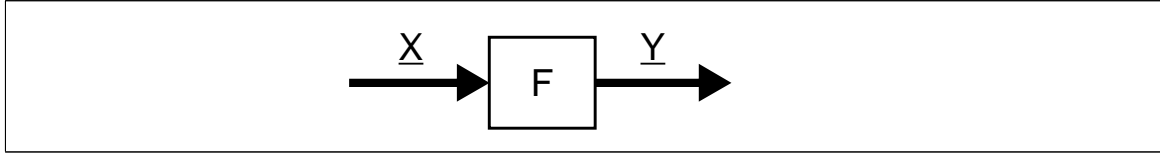


Figure 4-1: General combinational circuit

Under the zero delay assumption, any signal changes its at most once during a clock cycle. Under this assumption, we can write for any signal $x(t)$:

$$x(t) = x(i \cdot T) = x^i \quad (4-1)$$

where T is the clock period and i indicates the clock cycle.

Equations 2-15 and 2-16, the latter being repeated here for convenience, require to determine the switching activity α_i of the output of each gate i in the circuit. If the output wave-

$$P = \frac{1}{2} \cdot V_{dd}^2 \cdot f_{clk} \cdot \sum_{\text{all gates } i} \alpha_i \cdot C_{L_i} \quad (2-16)$$

forms y_i of all gates are available, the signal toggles can be counted by scanning y_i . However, sometimes it may be advantageous to define the switching signal tr_i by

$$tr_i = y_i(t) \oplus y_i(t-T) = y_i^j \oplus y_i^{j-1}. \quad (4-2)$$

α_i results then in

$$\alpha_i = P(tr_i). \quad (4-3)$$

For probabilistic switching activity estimation y_i and tr_i are expressed by Boolean equations while for logic simulation they are represented by signal waveforms. In the latter case, equation 4-3 yields only an estimate $\hat{\alpha}_i$ of α_i and since y_0 does not exist, tr_i^1 is defined by:

$$tr_i^1 = y_i^1 \oplus y_i^n, \quad (4-4)$$

where n denotes the last clock cycle.

Unless stated otherwise, the estimated values are considered as accurate for logic simulation based approaches and no special distinctions between α and $\hat{\alpha}$ will be made.

4.2 Logic Simulation

Power estimation by means of logic simulation has several advantages. Existing simulators need only to be extended by a capability to count the signal toggles, which represents only a minor modification. Although these simulators may not exploit optimization potentials that

exist if only toggle rates but no exact signal waveforms are required, all the advantages of a mature commercial tool come for free:

- good tool flow integration
- high sophisticated time models
- easy to use user interface.

Logic simulators inherently handle correlations and sequential circuits. Both require special treatment in other approaches as it is shown in a later section. But there are also some severe drawbacks coming along with logic simulation based power estimation. For logic simulation, a large set of input patterns is required, which must represent the typical operating conditions of the circuit. If “real” data is available, it is to be preferred. This maybe the case e.g. in audio or image processing systems. However, in most other applications, the patterns are generated through a high level simulation on system level. In [Dres 93] it was proposed to use the same patterns as for functional verification of the high level system. It is doubtful though, whether these patterns represent the typical operation mode of a circuit. The intention for generating such patterns is to stimulate each path exactly once, while in the typical operation mode the different paths are stimulated with significantly different probabilities. Thus, test patterns for power estimation must be generated separately.

Sometimes a high level model is either not available or high level pattern generation maybe impractical for other reasons, e.g. in a Monte Carlo approach. In these cases, the input stimuli must be generated in another way. A trivial idea would be to perform an exhaustive simulation, i.e. applying a set of input vectors corresponding to a transition from any possible input pattern to any other. This implies a total of 2^{2n} vectors for n input variables, which is impractical even for small circuits. Furthermore, such patterns usually don't represent the typical operation mode. The input stimuli must be rather generated with a random number generator. This is easily possible as long as spatially correlated PIs are not required. In [Rade 96] a tool is presented that generates random input patterns with user defined spatio-temporal correlations within limited groups of primary input variables. It is based on Markoff chains in order to model temporal and spatial dependencies. As these Markoff chains grow exponentially with group size, they become impractical for groups larger than seven variables with a first order temporal correlation. Moreover, the tool becomes hard to use even for smaller groups since the consistent definition of correlations is not trivial [Rade 97].

In logic simulation based power estimation approaches, the accuracy increases with the number of input patterns. But so does the runtime. Hence, most work in this field concentrates on improving the runtime behavior and on limiting the number of input patterns.

4.2.1 Speeding up Simulation

One way to reduce simulation time is to speed up simulation by parallelism. In the approach from [Dres 93] the authors use a parallel logic simulator that was extended by some power estimation features. The circuit is divided into partitions which are then simulated in parallel on the different processors of a multicomputer. The simulator can handle any delay

model, but because of a high communication overhead between the different processors, the speedup doesn't scale linearly with the number of processors involved. It meets an upper bound between four and eight with eight processors [Lanc 95]. If the number of CPUs is further increased, the additional communication overhead is higher than the additional compute power, thus simulation speed decreases.

The communication problem of the previous approach is avoided in a method that was presented by Schneider in [Schn 95]. Instead of simulating different circuit partitions in parallel, the bitparallel approach, which was first used in the test area [Waic 85], was adapted to power estimation purposes. The basic idea of bitparallel simulation is to exploit the full machine word length w during simulation by simulating w clock cycles at a time.

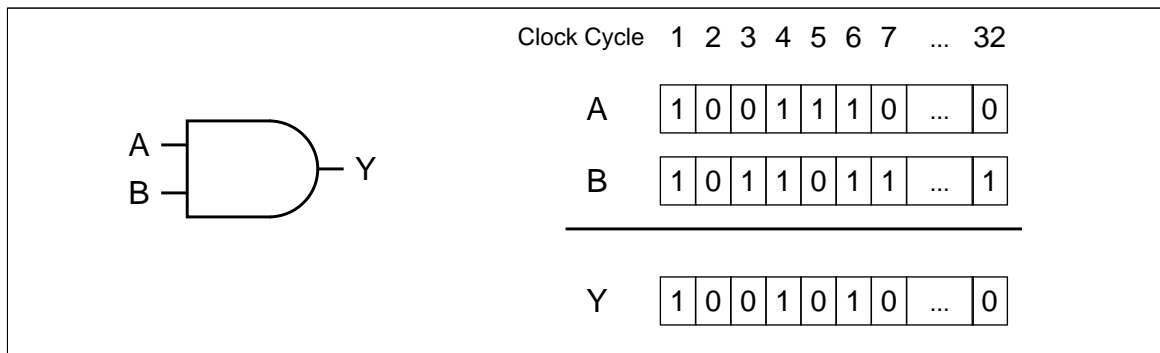


Figure 4-2: Bitparallel AND

E.g. consider the AND-gate of figure 4-2. It is to be simulated with 32 input patterns. Assume further that the simulator is running on a 32-bit machine. Thus $w=32$. The input patterns can be stored in two machine words, W_A and W_B , and can be processed with one single machine instruction. This results in a theoretical speedup of 32 over bitwise simulation. A look up table (LUT) based method is used for efficient determination of the number of transitions per word. Since the LUT grows exponentially with w , its size was limited to 8-bit words, resulting in $w/8$ LUT accesses per machine word and a certain performance penalty. The author reports an average speedup of 24 over bitwise simulation on a 64 bit machine. His algorithm is limited to the zero delay model and to purely combinational circuits. In sequential circuits the values of some signals depend on the previous clock cycle but for bitparallel simulation all values of w successive clock cycles must be known. [Goud 91] presents a method to extend the bitparallel approach to sequential circuits. The authors introduce a new signal value "unknown" which is iteratively replaced by '0' or '1'. However, this extension suffers from a performance penalty, that reduces the average speedup to 6.3. The figures that were presented in [Schn 95] and [Goud 91] are hard to compare since Gouders et al. don't focus on signal toggles and use a 32-bit machine.

4.2.2 Input Vector Determination

Assume an experiment where a circuit is simulated with n test patterns. The desired output of this experiment be the average energy consumption per clock cycle \bar{E}_1 . A second experiment with a different set of test patterns but the same statistical properties would yield a

slightly different result \bar{E}_2 . The average energy consumption per experiment can be estimated by the mean of the two experiments. But how reliable is this estimate?

According to the rules of statistics, the standard deviation of the estimated values, either absolute power, energy, or switching activity, decreases with the number of randomly generated input patterns that are applied to the circuit. Thus, for accuracy reasons, a high number of input patterns is desirable. On the other hand, simulation time increases linearly with the number of input patterns. Thus, a trade-off between accuracy and runtime is needed. A first approach in this direction was presented in [Burc 93]. The authors use a Monte Carlo method in order to stop the simulation procedure as soon as the required accuracy is reached.

Consider a circuit with m internal nodes. This circuit is simulated with n randomly generated patterns. It is assumed that the simulation yields the overall energy consumption e . The simulation process is called an *experiment* and the result of an experiment is also called a *sample*. The sample e can be considered as the sum of $n-1$ independent atomic experiments e_a . Each atomic experiment e_a is a representative of a random variable e_a with unknown distribution. Since e_a is a random variable, e is a representative of another random variable e . According to the central limit theorem [Bron 87], its distribution can be approximated by a normal distribution if n is chosen large enough (usually $n=100$ is sufficient). N experiments of size n result in N different samples e_i . The mean \bar{e} of these experiments is a normally distributed random variable itself. But we still don't know the variance of \bar{e} . Therefore an estimate of the standard deviation of e is built:

$$S = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^N (e_i - \bar{e})^2}. \quad (4-5)$$

Using S and \bar{e} we can build a new random variable X :

$$X = \frac{\bar{e} - \bar{E}}{S} \cdot \sqrt{N-1} \quad (4-6)$$

where \bar{E} is the unknown average energy consumption per experiment of size n . X can be considered as the normalized error of the current estimate. X is *Student-t* distributed and does not depend on the standard deviations of \bar{e} and S . For X we can say with $(1-\beta)$ confidence:

$$|X| < t(\beta, N). \quad (4-7)$$

From equation 4-7 we can derive a statement about the maximum relative error ε of the current estimate \bar{e} and the confidence $(1-\beta)$ of this statement:

$$\varepsilon = \left| \frac{\bar{e} - \bar{E}}{\bar{e}} \right| < \frac{S}{\bar{e} \cdot \sqrt{N-1}} \cdot t(\beta, N). \quad (4-8)$$

A table of the *t-function* $t(\beta, N)$ as well as its exact definition can be found in [Bron 87]. Equation 4-8 enables to check after each experiment whether ε meets a given bound ε_l with a given confidence $1-\beta$. The Monte Carlo approach can be applied to any circuit type, either

combinational or sequential, and to any pattern based power simulator where it yields good accuracy at reasonable runtime. However, the Monte Carlo approach still suffers from two drawbacks:

1. While spatial or temporal correlations cause no problem for the simulation process itself, generation of adequate test patterns maybe difficult [Rade 96].
2. the number of experiments N^* which are required to meet the accuracy requirements is still unpredicted in [Burc 93]. The authors only prove that N^* typically decreases or remains constant with circuit size m .

The problem of a-priori estimation of N^* was addressed in [Hill 95]. It is assumed that the average switching activity per clock cycle s of a specific circuit node is to be determined. s is a sample of the random variable s and is obtained by the simulation of N patterns. Thus we get

$$s = \frac{\text{tog}}{N} \quad (4-9)$$

where tog is the total number of toggles during the experiment. The mean of s is given by

$$E(s) = \frac{1}{N} \sum_{i=1}^t i \cdot p_i \quad (4-10)$$

where p_i is the probability that a single input pattern produces exactly i toggles at the node under consideration. Its variance σ^2 can be estimated by

$$\hat{\sigma}^2 = E(s^2) - E^2(s) = \frac{1}{N} \left(\sum_{i=1}^t i^2 \cdot p_i - \left(\sum_{i=1}^t i \cdot p_i \right)^2 \right). \quad (4-11)$$

For sufficiently large N , s is normally distributed and Hill et al. replace the unknown, real value of σ by $\hat{\sigma}$. The required limit N^* results then in

$$N^* \approx \left(\frac{\Phi^{-1}(\beta/2)}{\varepsilon'} \right)^2 \cdot \left(\sum_{i=1}^t i^2 \cdot p_i - \left(\sum_{i=1}^t i \cdot p_i \right)^2 \right) \quad (4-12)$$

for a given confidence β . Where $\Phi^{-1}(x)$ is the inverse of the probability integral $\Phi(x)$ [Bron 87] and ε' denotes the absolute error:

$$\varepsilon' = |s - E(s)|. \quad (4-13)$$

The question, how the probabilities p_i in equation 4-12 can be determined, is still open. According to the authors, they can be estimated with sufficient accuracy from a short simulation with $n=100$. This assumption was justified by simulation results, which were obtained by simulation of a variety of benchmark circuits.

Low activity nodes impose special problems for Monte Carlo approaches since they require a much higher number of input patterns to obtain the same relative accuracy as high activity nodes. On the other hand, the contribution of low activity nodes to the overall power con-

sumption is low, thus a higher relative error is acceptable. Accordingly, in [Xake 94] the authors distinguish between high and low activity nodes. For the former, they use the relative error, for the latter the absolute error as stopping criteria. Their results show that about 20% of all circuit nodes can be considered usually as low active, which results in a considerable runtime improvement without significant loss in overall accuracy.

4.3 Probabilistic Methods

In the case of logic simulation, a large amount of input patterns is propagated through a circuit in order to determine its average switching activity. However, for power estimation, the exact waveforms of the circuit nodes are not really needed. Equations 2-15 and 2-16 require only the switching activity α , which is a scalar value. Since α is a statistical property of the circuit nodes, it is an obvious solution to determine the statistical properties of the primary input signals and to propagate those properties through the circuit in one single simulation run. Depending on the approach chosen, the *statistical properties* are signal probabilities, transition densities, switching probabilities, and/or signal correlation coefficients.

4.3.1 Properties

The result of power estimation by probabilistic means is equivalent to logic simulation with an infinite number of input patterns. Under the assumption that the statistic properties of the primary inputs are accurately modeled, probabilistic simulation yields the exact result in one single simulation run. Hence, at a first glance probabilistic approaches seem to be faster and more accurate than pattern simulation based approaches. In some cases it may also be an advantage that probabilistic approaches don't require explicit test patterns.

Statistical independence plays an important role for probabilistic circuit simulation. Only if statistical independence of the circuit inputs is guaranteed, the following simple propagation rules hold:

Logic Operation	Signal Probability
$y = \bar{a}$	$p_y = 1 - p_a$
$y = a \cdot b$	$p_y = p_a \cdot p_b$
$y = a + b$	$p_y = p_a + p_b - p_a \cdot p_b$

Table 4-1: Rules for probability propagation for independent signals

For spatially correlated signals a and b , simple rules do only exist if a and b are disjoint. Those rules are summarized in table 4-2.

Obviously, the major problems of probabilistic approaches are imposed by signal correlations. From a methodical point of view, they can be classified into three classes: temporal

Logic Operation	Signal Probability
$y = a \cdot b$	$p_y = 0$
$y = a + b$	$p_y = p_a + p_b$

Table 4-2: Rules for probability computation for disjoint signals

correlations, spatial correlations caused by reconvergent fanout, and spatially correlated primary input signals. The latter are neglected by most approaches so far. For some methods, concurrent switching of primary inputs represents an additional source of estimation error.

4.3.2 State-of-the-Art in Probabilistic Simulation

Probabilistic treatment of digital circuits was first proposed by Parker and McCluskey in order to solve problems in the area of test pattern generation [Park 75]. The authors introduce the *signal probability* and present methods to propagate it through digital circuits, assuming the PIs are uncorrelated and their signal probabilities are given. For each node, they compute its Boolean function as a *canonical sum of products (CSP)*. It can be obtained e.g by exhaustive Shannon's expansion. Since all addends are disjoint in a CSP, the rules of table 4-2 are valid. While CSPs guarantee that correlations due to reconvergent fanouts are correctly modeled, they become very large for practical circuits. [Savi 84] presents an algorithm to reduce this complexity. Its validity has been mathematically proven in [Mark 87]. This approach allows to incrementally compute the probabilities by cutting reconvergent fanout lines in the circuit. But instead of absolute values, this algorithm gives only intervals for the probabilities.

Both algorithms estimate only static signal probabilities for test purposes. The first applications of probabilistic methods for power estimation can be found in [Burc 88] and [Najm 90]. Both publications deal with power estimation on circuit level. The circuits are partitioned into p- and n-blocks which are represented by graphs. The graphs are then reduced to single edges between V_{dd} or GND using probabilistic serial/parallel reduction. For each node, there are two graphs to be build: one for the static probabilities and one for the transition probabilities. The algorithm assumes uncorrelated PIs but can handle spatial correlations that are introduced by reconvergent fanout using the *supergate* approach that was first presented in [Seth 85]. The circuit is partitioned such that the reconvergent fanouts are hidden inside the partitions. These partitions are called a *supergate* and their inputs are supposed to be spatially uncorrelated. The algorithms were implemented in a simulator called *CREST (CuRent ESTimator)*. Since CREST can also model gate delays, it was used to compute current waveforms in digital circuits. Compared to SPICE runs, it yields speed-ups of two or three orders of magnitude at error rates below 20%, according to the authors.

The theoretical basis of probabilistic power estimation was first formulated in [Najm 93], where Najm introduces some fundamental terms like *the companion process*, *the equilibrium probability* or the *transition density* (see definitions 2-63, 2-64 and 2-65). They are

applied with some modifications and specializations in many later publications. Najm has proven that a logic signal is always a *strict sense stationary, mean ergodic 0-1 process*, such that the limits in definitions 2-64 and 2-65 always exist. For transition density estimation, Najm applies the Boolean difference operator (definition 2-32) in the following way. Given any internal node j of a circuit with primary inputs $x_1 \dots x_n$, the transition density $D(y_j)$ is estimated by

$$D(y_j) = \sum_{i=1}^n P\left(\frac{\partial y_j}{\partial x_i}\right) D(x_i) \quad (4-14)$$

where $P(\partial y_j / \partial x_i)$ denotes the probability that the Boolean difference $\partial y_j / \partial x_i$ results into '1'. Equation 4-14 is only an approximation. It does not correctly cover simultaneous switching inputs since it does not include any higher order terms with mixed derivatives (e.g. terms like $\delta^2 y_1 / \delta x_1 \delta x_2$). This assumption maybe reasonable for asynchronous designs or for some very specific, low active circuits. However, in a synchronous design all the primary inputs change simultaneously. This may lead to large estimation errors as shown in the following example. Consider the XOR-circuit of figure 4-3. Assume that $D(a)$ and $D(b)$ are given and they switch statistically independently.

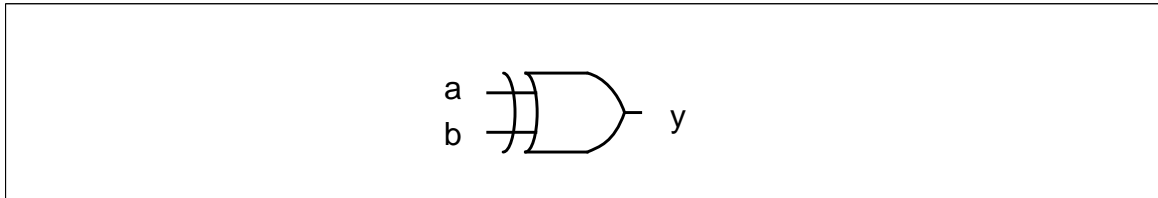


Figure 4-3: XOR circuit

The logic function of the XOR is denoted by

$$y = a\bar{b} + \bar{a}b. \quad (4-15)$$

Obviously, the output y switches if either a or b switches, but not if both inputs toggle simultaneously. Thus,

$$D(y) = D(a) + D(b) - 2D(a)D(b). \quad (4-16)$$

However, equation 4-14 results in

$$D(y) = D(a) + D(b). \quad (4-17)$$

Equation 4-14 tends to overestimate the circuit activity. It can be easily proven that the error increases with increasing switching densities $D(a)$, $D(b)$.

This approach was implemented into the simulator *DENSIM* [Najm 93]. The supergate approach was applied in order to handle correlations that are caused by reconvergent fanouts. The PIs are supposed to be spatially uncorrelated. The results are good in runtime and in accuracy, as long as the partitions (supergates) are chosen carefully. However, for some larger benchmark circuits from the ISCAS-85 benchmark set [Brgl 85] a lowest level

partitioning was chosen, where each logic gate is a supergate itself. In this case the error increases to values up to 40%. It will be shown in chapter 6 that even better partitioning won't improve this result as long as simultaneous switching is neglected.

Nevertheless, [Najm 93] presents a BDD based method to compute signal probabilities. Because of its importance for this work, it shall be described in more detail. Consider the BDD of figure 4-4. Assume that the primary input variables x_1 , x_2 , and x_3 are uncorrelated and their signal probabilities $P(x_1)$, $P(x_2)$, and $P(x_3)$ are given.

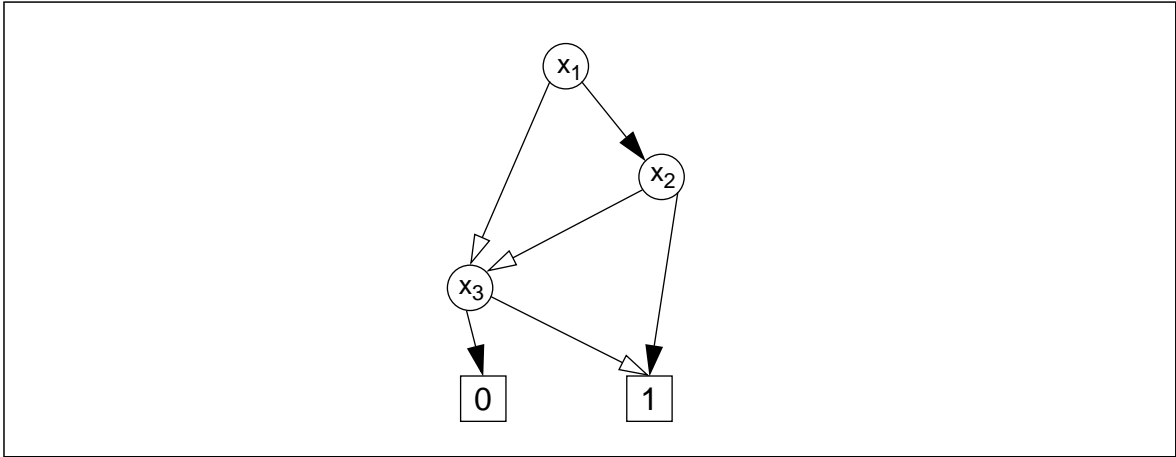


Figure 4-4: ROBDD for $y = \bar{x}_3 + x_1x_2$

For any Boolean function $y=f(x_1, x_2, \dots, x_n)$ the 1-probability of the function can be obtained in linear time in the size of the BDD. Each node represents a Shannon's expansion of the subfunction defined by the node. Thus,

$$y = x_1f_{x_1} + \bar{x}_1f_{\bar{x}_1}. \quad (4-18)$$

The two addends of equation 4-18 are disjoint since $x_1\bar{x}_1=0$, and we can apply the results of table 4-2 to compute $P(y)$:

$$P(y) = P(x_1f_{x_1}) + P(\bar{x}_1f_{\bar{x}_1}). \quad (4-19)$$

Since the cofactors of x_1 do not depend on x_1 and since all variables are independent, table 4-1 can be applied to evaluate the products:

$$P(y) = P(x_1)P(f_{x_1}) + P(\bar{x}_1)P(f_{\bar{x}_1}). \quad (4-20)$$

The probabilities of the cofactors can be computed accordingly. Thus, $P(y)$ can be determined recursively by a depth-first traversal of the BDD of y . The BDD must be ordered but not necessarily reduced. In this way, the probabilities of the Boolean differences in equation 4-14 can be easily evaluated for any internal circuit node if its BDDs are available. The BDDs can become very large, though. In the example of figure 4-4 we obtain:

$$P(y) = P(x_1)[P(x_2) + P(\bar{x}_2)P(\bar{x}_3)] + P(\bar{x}_1)P(\bar{x}_3). \quad (4-21)$$

Several researchers tried to improve Najm's approach. In [Kapo 94] two methods are presented. The first speeds up the simulation process by a smart reuse of the BDDs that were created during the evaluation of preceding gates. The second tries to improve the accuracy by a partitioning algorithm that packs correlated nodes into one supergate. However, there is no statement about general accuracy, only the percentage of low error nodes is mentioned.

[Schn 94] and [Schl 95] address the problem of concurrently switching inputs by a modification of equation 4-14. Instead of building the Boolean differences with respect to each input variable x_i , they use the switching function tr_y (equation 4-2). tr_y is represented by a BDD which is also called *XOR-BDD* because of the XOR-operation in equation 4-2. While traversing the XOR-BDD, they take into account the temporal correlations of the primary input variables x_i , that are given by their switching probability α_{x_i} . The variable order for the XOR-BDD is chosen such that x_i^j and x_i^{j-1} are adjacent (figure 4-5).

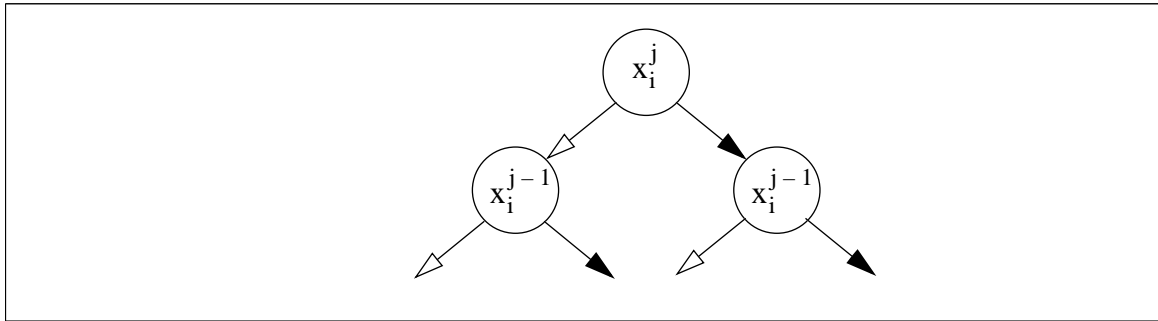


Figure 4-5: Part of an XOR-BDD

The four possible paths in figure 4-5 are evaluated using the following equations [Schl 95], [KapD 98]:

$$P_{00}(x_i) = 1 - P(x_i) - \frac{\alpha_{x_i}}{2} \quad (4-22)$$

$$P_{01}(x_i) = \frac{\alpha_{x_i}}{2} \quad (4-23)$$

$$P_{10}(x_i) = \frac{\alpha_{x_i}}{2} \quad (4-24)$$

$$P_{11}(x_i) = P(x_i) - \frac{\alpha_{x_i}}{2} \quad (4-25)$$

Where $P_{00}(x_i)$ denotes the path where $x_i^j = 0$ and $x_i^{j-1} = 0$, $P_{01}(x_i)$ is the path where $x_i^j = 0$ and $x_i^{j-1} = 1$, etc. Equations 4-22 - 4-25 can be easily verified using a two state Markoff chain where the two nodes represent the two possible values of x_i .

In [Marc 94] and [Marc 95] global BDDs and the problems that are related to circuit partitioning, are avoided by the definition of *static correlation coefficients* and *transition correlation coefficients*. They are propagated through the circuit in addition to the static

probabilities of the primary input variables. The correlation coefficients are derived from a four state Markoff chain that allows to model all the first order spatial and temporal correlations of two variables. Higher order correlations are neglected, but according to the results, they have no major influence on the accuracy.

The approach presented in [Chou 94] is basically an extension of [Najm 93]. The Taylor expansion of equation 4-14 was extended by higher order terms. Thus, the effects of simultaneous switching inputs can be taken into account, which yields high improvements in accuracy. However, like in [Najm 93], the runtime behavior heavily depends on a careful partitioning of the circuit.

Probabilistic methods seem to offer a much faster and more accurate solution to the problem of switching activity estimation than pattern simulation with a conventional logic simulator. However, correct correlation handling imposes a major problem. Thus, simple methods, which neglect all correlations are very fast but suffer from high estimation errors. BDD based methods are more accurate but slower. Depending on the approach, they can model spatial correlations caused by reconvergent fanout and temporal correlations more or less accurately. However, accurate handling of spatially correlated PIs is still a problem for most of today's probabilistic approaches. The practical importance of the latter becomes obvious in the next section.

4.4 Sequential Circuits

The circuit in figure 4-6 depicts a synchronous sequential circuit. It can be mathematically modeled by a *finite state machine (FSM)*. Analysis and design of asynchronous circuits are out of the scope of this thesis. The interested reader is referred to [Brzo 95] and [Lava 93] for general introductions to asynchronous circuits and to [Birt 95] for special aspects of asynchronous FSM design.

A synchronous FSM consists of two combinational circuits F and G , and a register bank that stores the current state \underline{Z}^i . F and G are functions of the primary input vector \underline{X}^i and the current state \underline{Z}^i . F is called the *output function* or *output logic*. G is the *next state function* or *next state logic*. The output of G is the *next state* \underline{Z}^{i+1} . At each active edge of the system clock clk , \underline{Z}^{i+1} is stored in the register bank and forwarded to the register outputs. So, it becomes the current state \underline{Z}^i .

The FSM in figure 4-6 is a *Mealy machine* because the output \underline{Y}^i depends on the current state \underline{Z}^i and the input vector \underline{X}^i :

$$F = F(\underline{X}^i, \underline{Z}^i). \quad (4-26)$$

If F is only a function of the current state \underline{Z}^i , the FSM is called a *Moore machine*:

$$F = F(\underline{Z}^i). \quad (4-27)$$

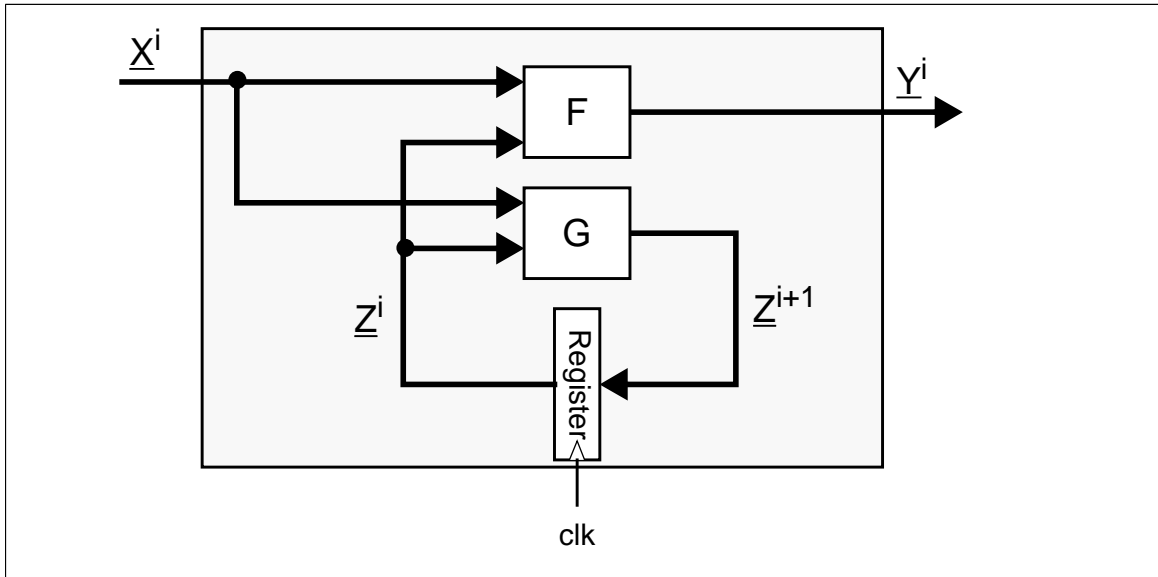


Figure 4-6: Synchronous sequential circuit

Summarizing, an FSM consists of two combinational networks F and G , and a feedback loop with a register bank that stores the current state Z^i . As soon as the waveforms or the statistical properties of the current state Z^i and/or the next state Z^{i+1} are known, the power consumption of the FSM can be reduced to a purely combinational problem.

However, the spatial correlations between the current state bits Z_i as well as the temporal correlations between Z^i and Z^{i+1} must be taken into account in order to obtain correct results. In [Gosh 92] Gosh et al. propose the configuration of figure 4-7 as a general model of this problem.

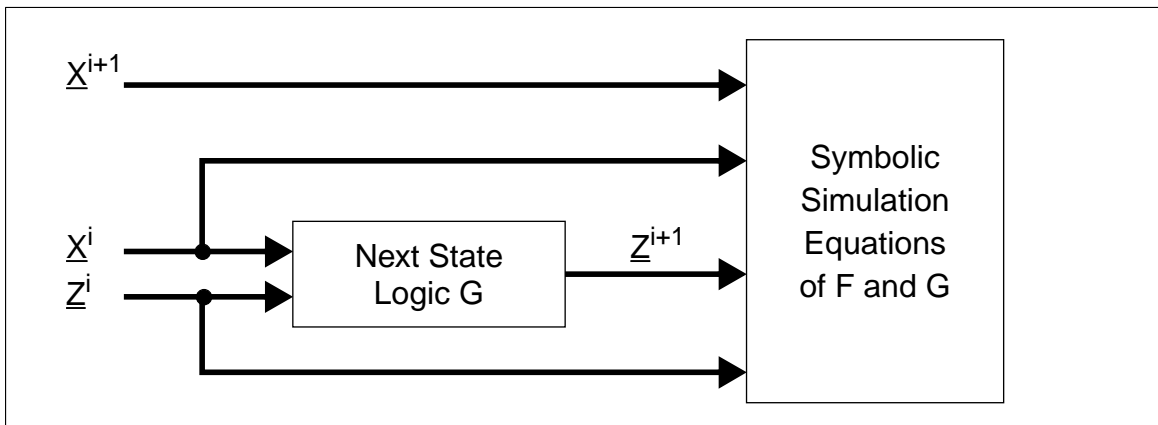


Figure 4-7: Modeling correlations

It implies that the statistical informations about the current input vector X^i , the next input vector X^{i+1} , and the current state Z^i are sufficient to solve the power estimation problem. The information about the next state can be computed using the next state logic G . With the symbolic simulation equations of F and G , the signal activity of the FSM can be computed then. Yet, the authors haven't specified in detail which statistical information they require at

last and how the *next state logic block* transforms its input variables to its output values. It becomes obvious though that the information about the next state \underline{Z}^{i+1} can be derived from the current input vector \underline{X}^i and the current state \underline{Z}^i .

The *symbolic simulation equations* represent any combinational power estimator, either pattern based or probabilistic. For pattern based methods the *next state logic* consists of the Boolean equations of G . For probabilistic approaches it computes the probabilities and correlations of the next state by applying one of the following three approaches:

1. High level simulation
2. State transition graph
3. State line probabilities.

The following sections focus on the details of these three approaches to implement the next state logic.

4.4.1 High Level Simulation

If the FSM is a part of a larger design, there exists usually a formal high level model of the system. If this model can be simulated, the desired waveforms or their statistics can be generated through high level simulation. This approach is straight forward and can be performed for a high number of patterns since high level simulators are very fast because of their high abstraction level. However, one has to be aware that the proper choice of input patterns suffers from the same problems as pattern simulation on logic level. Furthermore, sequential circuits require a much higher number of input patterns for the same level of accuracy than combinational circuits because of the history effect of the registers.

Examples for high level specification languages are *State Charts* [Hare 87], *behavioral VHDL* or *SDL* [Ells 97].

4.4.2 State Transition Graph

Figure 4-8 depicts the state transition graph (STG) of a Mealy machine with four states, one input, and one output variable. The nodes represent the states with their symbolic names and codes. The edges depict the state transitions. They are labeled with the transition conditions and the output values (*condition/output value*). The STG can be derived from a high level model or it can be extracted from the Boolean functions F and G . The exact state probabilities can be computed using the Chapman-Kolmogorov equations for discrete time Markoff Chains [Deva 94].

In an FSM with K states $s_1 \dots s_K$, we associate a variable $P(s_i)$ for each state s_i , corresponding to the steady-state probability of the machine being in state s_i for time $t \rightarrow \infty$. For each edge e in the STG we have $e.Curr$, $e.Next$, and $e.In$ which signify the head, the tail, and the input condition of the edge, respectively. For each state s_i we can write an equation

$$P(s_i) = \sum_{\forall e, e.Next = s_i} P(e.In) \cdot P(e.Curr). \quad (4-28)$$

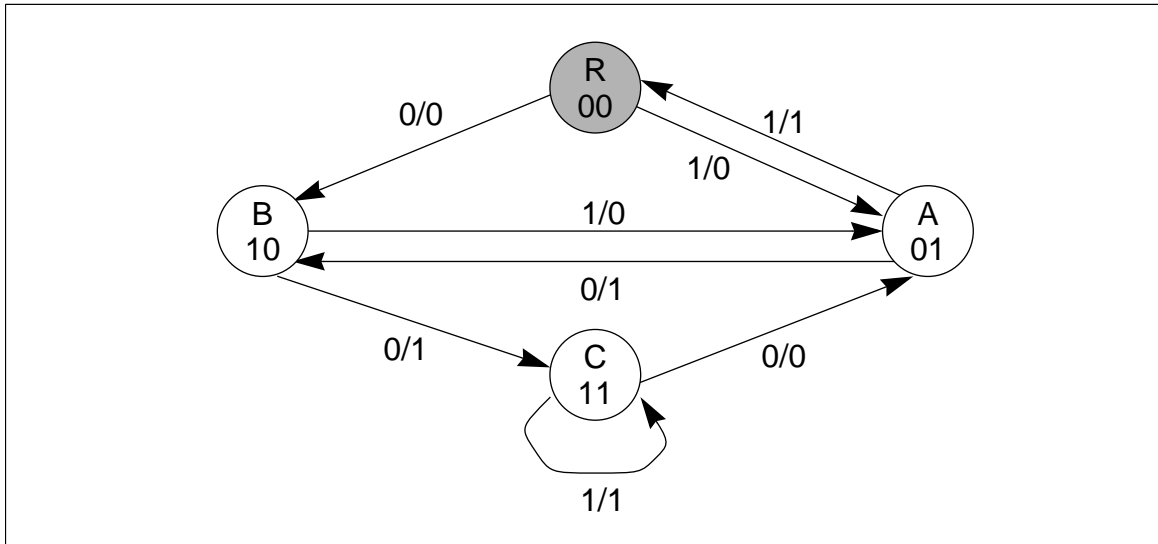


Figure 4-8: A state transition graph

If the input variables are assumed to be uncorrelated and their signal probabilities are given, $P(e.In)$ can be easily computed. Given K states, we obtain K equations out of which any single equation can be derived from the remaining $K-1$ equations. Hence, one of the K equations is redundant. It must be replaced by

$$\sum_{i=1}^K P(s_i) = 1. \quad (4-29)$$

This linear set of equations can be solved to obtain the $P(s_i)$ using the Gauss algorithm. This results in a problem of complexity K^3 . As soon as the $P(s_i)$ s are known, the state transition probabilities can be easily computed.

The major drawback of this approach is its average-case exponential complexity: for an FSM with n flip flops we have $K=2^n$ states, making this method impractical for large FSMs.

4.4.3 State Line Probabilities

As shown in the previous section, exact state probability computation is impractical for large FSMs. A first approximation that overcomes this problem is presented in [Gosh 92]. Instead of state probabilities, state line probabilities are computed. Spatial correlations between the state line probabilities are neglected. In order to further simplify the problem, Gosh et al. make the following additional assumptions:

1. The probability that the FSM is in any of its 2^n states is uniform. Thus, $P(z_i)$ is also uniform.
2. The probability of the primary inputs $P(x_i)$ as well as their switching probabilities α_{x_i} are given.
3. The primary input signals and their switching activities are spatially uncorrelated.

The next stage logic consists of Boolean equations which model the temporal correlations between the present and the next state lines. These correlations are expressed by the transition probabilities of the state lines α_{z_i} . Considering figure 4-6, the information about \underline{X}^i and \underline{Z}^i is represented by the primary input probabilities $P(x_i)$ and the present state line probabilities $P(z_i)$, while the switching probabilities α_{x_i} and α_{z_i} are interpreted as the information about \underline{X}^{i+1} and \underline{Z}^{i+1} .

Feedback Loop

The major drawback of the previous approach is its assumption for uniform state probabilities which is not given in most practical applications. [Deva 94], [Tsui 94], and [Tsui 95] present a means to solve this problem. The authors take advantage of the observation that the steady state probabilities of the current state lines $P(z_j^i)$ and the next state lines $P(z_j^{i+1})$ are identical. The same is true for the primary inputs \underline{X}^i . Hence, they formulate the following equations for each state line z_i :

$$P(z_i) = P(f_i(x_1, x_2, \dots, x_n, z_1, z_2, \dots, z_m)). \quad (4-30)$$

It results in a system of m nonlinear equations for the m state lines. This system is solved with the Picar-Peano or the Newton-Raphson method [Bron 87]. The transition probabilities can then be computed using the same method as [Gosh 92]. Equation 4-30 can be illustrated with a feedback loop in the next state logic (figure 4-9).

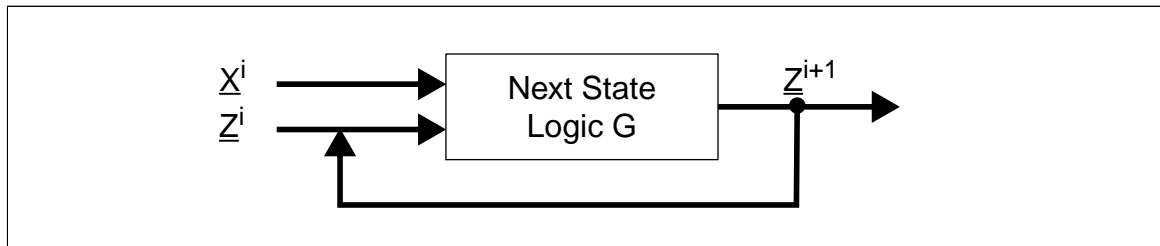
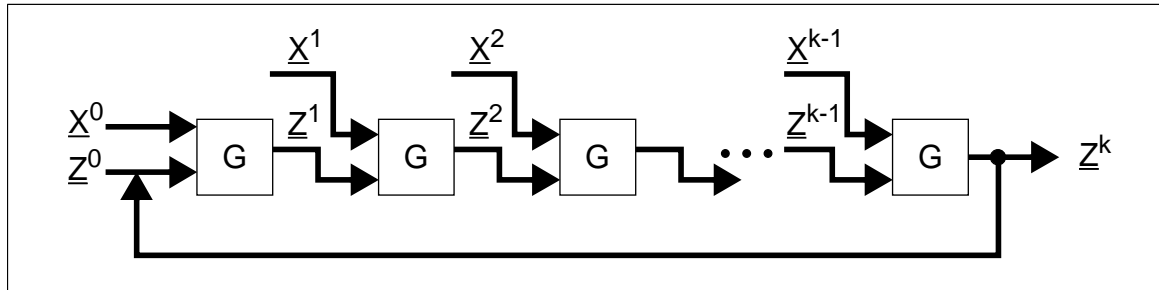


Figure 4-9: Feedback loop

While this method allows to account for non-uniform state line probabilities, it can still not cover spatial correlations between the state bits.

k-Unrolled Network

In [Deva 94] and [Tsui 95], Devadas et al. propose the method of *k-unrolled networks* as a generalization of the feedback method in order to model spatial correlations between state lines. Consider the configuration of figure 4-10. The next state logic has been unrolled k times. As before, a set of nonlinear equations corresponding to this k -unrolled network can be constructed. When computing the state line probabilities $P(z_i)$, these equations partially take into account correlations between the state lines z_i . The method of figure 4-9 corresponds to a k -unrolled network with $k=1$. The exact probabilities would be obtained for $k \rightarrow \infty$.


 Figure 4-10: k -unrolled network

m -Expanded Networks

While the k -unrolled network takes into account the correlations between the state lines during state line probability computation, the correlations can still not be expressed for the symbolic simulation equations. m -expanded networks allow to model the correlations between m -tuples of the present state lines [Tsui 95]. Consider for instance the 2-expanded network of figure 4-11. It models the correlations between the first two state bits by separately computing their joint signal probabilities. In the next state logic, a k -unrolled network can be used, thus combining the two approaches.

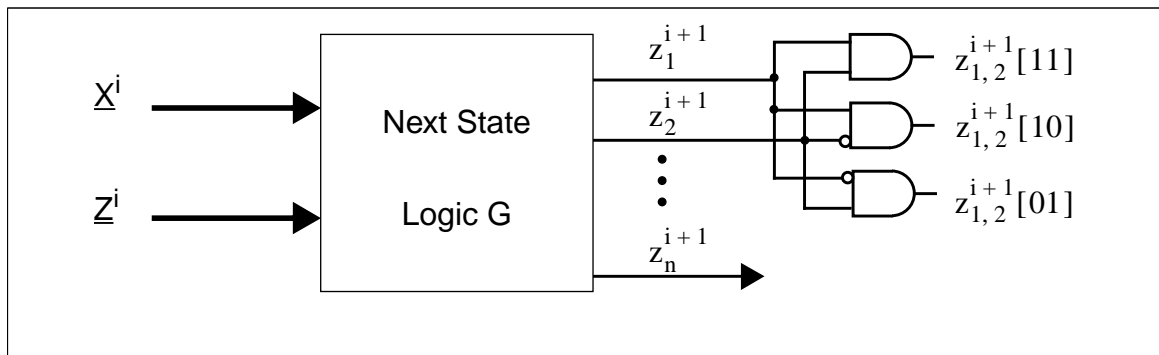


Figure 4-11: A 2-expanded network

Experiments on a wide variety of benchmarks indicate that the approximation methods presented before are accurate to within 5%. Depending on the values of k and m an average error below 1% becomes even possible at the cost of a higher computational effort. However, all these experiments assume spatially uncorrelated primary inputs \underline{X}^i .

Input Modeling FSMs

The problem of spatially correlated primary inputs \underline{X}^i can be addressed using *Input Modeling FSMs (IMFSM)* [Mont 95]. The IMFSM are virtually designed to generate the input waveforms \underline{X}^i with the desired statistics (steady state and transition probabilities, and spatial correlations) from a white noise input stream (figure 4-12). From the IMFSM the statistical parameters of the primary inputs \underline{X}^i of the FSM under test can then be extracted using one of the previous methods.

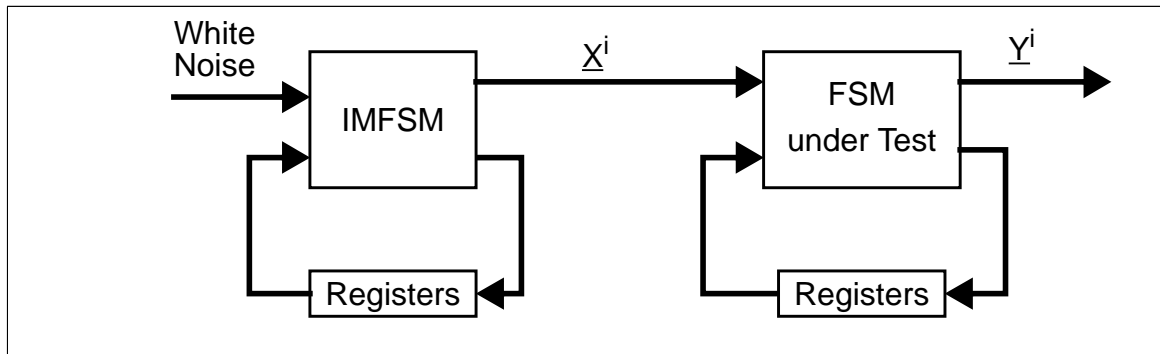


Figure 4-12: Input modeling FSM

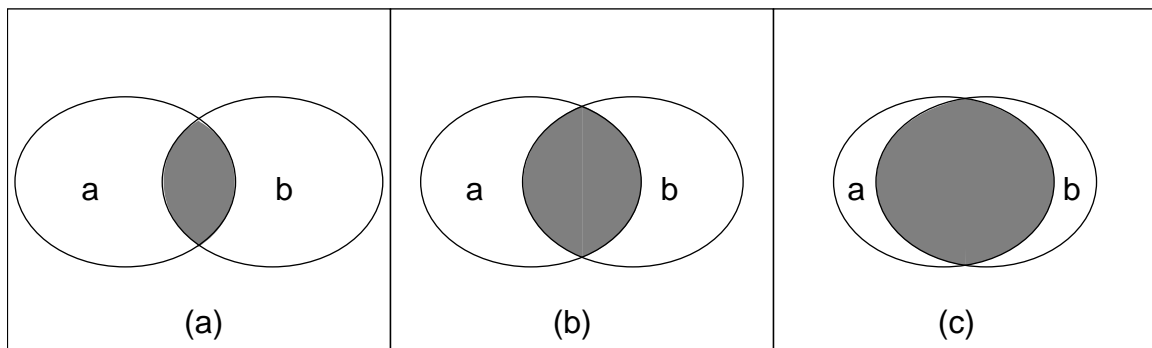
4.5 A New Set Based Simulation Method

In this section, a novel technique will be presented that enables accurate estimation of the node activity in combinational CMOS circuits. The approach is based on set theory. It can be classified between logic and probabilistic techniques. It is able to take into account temporal as well as spatial correlations. The results will show that the approach is particularly efficient for the simulation of circuits with highly correlated and/or low active primary inputs. Since both types of signals are characteristic for FSMs, set based simulation is best suited to be applied in connection with the methods that were presented in section 4.4.

4.5.1 Basic Method

Venn Diagrams

Venn diagrams provide a clear way to visualize relations between sets. Consider for instance figure 4-13. It depicts three Venn diagrams of two variables a , b , with different correlation coefficients. The shaded regions denote the intersection of the two signals corresponding to a logic AND. The variables are negatively correlated, uncorrelated, and highly correlated, in diagrams a , b , and c , respectively. Although the areas of a and b which correspond to p_a and p_b , are equal in all three diagrams, their intersections are significantly different because of the different correlation coefficients. Obviously, Venn diagrams allow to

Figure 4-13: Venn diagrams of two signals a and b

model correlations in a simple and accurate way. How this model can be applied to switching activity estimation in digital circuits is shown in the sequel.

From Logic Operations to Set Operations

Firstly the signal representation is changed. Instead of storing the signals bit by bit they are represented as *1-blocks* or simply *blocks*. A block is a group of consecutive '1's in the signal. Only the position (clock cycle) of the first and the last '1' of each block are stored (figure 4-14).

clk	1	2	3	4	5	6	7	8	9	...							
Bits	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	1	1
Blocks		2-3					7-11						14-17				

Figure 4-14: Signal representation with blocks

If the signals are to be propagated through a logic gate, the blocks are considered as sets. Thus, all logic operations are to be replaced by set operations. Table 4-3 depicts the relations between the basic logic, set, and probabilistic operations. Note that the latter assume uncorrelated signals. Any function of higher complexity can be decomposed into a sequence of these basic operations.

Logic Operation		Set Operation		Probabilistic Operation
Operation	Symbol	Symbol	Operation	
AND	$x \cdot y$	$x \cap y$	Intersection	$p_x \cdot p_y$
OR	$x + y$	$x \cup y$	Union	$p_x + p_y - p_x \cdot p_y$
NOT	\bar{y}	\bar{y}	Inversion	$1 - p_y$

Table 4-3: Logic, set, and probabilistic operations

While for probabilistic simulation the logic signal waveforms are compressed to one single value, the signal probability p , the compression rate is lower for the set representation because each block requires two values, and several blocks per signal are possible. The block sizes correspond to signal probabilities and the positions of the blocks of one variable in relation to the positions of the blocks of another variable express the correlations between variables.

4.5.2 Exact Optimization

The set operations are more efficient if the average number of blocks per signal is low since the costs of the set operations grow linearly with the number of blocks. This means, low active signals profit most from the set representation. However, in practical applications

there exist always some signals with high switching activity. Their average block size can be increased, and thus the average number of blocks decreased, through reordering of the input patterns. However, simple pattern rearrangement does not only effect the number of blocks per signal. It does also change the signal activity of the signals and importantly deteriorate the accuracy of the simulation result. In order to avoid this accuracy loss, the notion of the *physical signal* is introduced.

Definition 4-1: Physical Signal

The *physical signal* $\underline{x}(t)$ is the ordered pair of the signal $x(t)$ and a copy of $x(t)$ which is delayed by one clock cycle T :

$$\underline{x}(t) = (x(t), x(t - T)) = (x, x_T).$$

Figure 4-15 depicts two arbitrary physical signals a and b . Note that $a_T(1)$ and $b_T(1)$ are obtained by wrapping around the undelayed signals:

$$a_T(1) = a(9) \text{ and } b_T(1) = b(9). \quad (4-31)$$

time	1	2	3	4	5	6	7	8	9
a	0	0	1	0	1	0	0	1	0
a _T	0	0	0	1	0	1	0	0	1
b	1	1	1	0	0	0	1	1	0
b _T	0	1	1	1	0	0	0	1	1

Figure 4-15: Arbitrary physical signals

Each column represents now two consecutive input patterns and contains all static and dynamic information required to compute correct signal activities, regardless of the order of the columns. Therefore the *time* is omitted and the columns can be arbitrarily reordered in

a	0	0	0	1	1	1	0	0	0
a _T	0	0	0	0	0	0	1	1	1
b	1	1	1	1	1	0	0	0	0
b _T	0	0	1	1	1	0	1	1	0

Figure 4-16: Rearranged Input Sequence

order to reduce the block numbers. Figure 4-16 shows a possible optimization of the input sequence of figure 4-15 where the number of blocks was reduced from 10 to 5.

4.5.3 Optimization Methods

As mentioned already in the last section, the goal of the optimization is to create blocks as large as possible, in order to reduce the simulation costs. A brute force method, which corresponds to a full search over all possible rearrangements of the input patterns, is impractical even for a low number of input patterns N . The cost of the full search can be estimated by [Dall 98]

$$\text{cost} \sim \frac{1}{2} \cdot (N - 1)! . \tag{4-32}$$

Even for as few as $N=100$ patterns, equation 4-32 results in the unimaginable number of $8.5 \cdot 10^{160}$ possible rearrangements. On the other hand, the problem of finding the best rearrangement is NP-complete. Thus, the problem can only be solved approximately by suitable heuristics which will be described in the sequel.

Consider a circuit with n primary input signals (PI) and a set of N test patterns from which an optimal set representation shall be derived. If $N \gg 2^{2n}$ or if the input streams are highly correlated, identical input patterns are very likely¹. They are merged together in a first step, which is also called *duplicate-removal*. If $N \ll 2^{2n}$ but also for highly correlated signals, heuristic rearrangement can further improve the set representation.

The methods for duplicate-removal were mostly presented in [Dall 98]. They are based on a combination of hash and lookup tables (LUT), depending on the number of PIs. The computational effort is quite low with runtimes² below 0.5s. As details of these approaches are not part of this thesis, the interested reader is referred to the above-mentioned reference.

Heuristic Rearrangement

For the heuristic rearrangements it is assumed that duplicates were already removed, N^* shall be the number of different input patterns. If $N^* \approx 2^{2n}$, Gray code like arrangements yield the best results.

a	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
a _T	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
b	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
b _T	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Figure 4-17: Gray code arrangement

In figure 4-17 the columns represent a four bit Gray code. In order to account for duplicates, the columns, which correspond to input patterns, are weighted by the number of occur-

1. There are 2^{2n} possible input patterns for n PIs, since we always consider physical signals.
2. The experiments were carried out on a Sun Ultra Sparc 1, 170 MHz, with 192 MB main memory, running Solaris 2.5. 10,000, 15 bit wide vectors were searched for duplicates.

rences of the specific patterns. If $N^* = 2^{2n}$ it can be very easily proven that the Gray code arrangement is the best possible solution.

Proof

Assume that a Gray code arrangement G of arbitrary width with g blocks and no duplicates is given. In each column of G there exists exactly one block that begins or ends in this column since exactly one bit changes its value from one column to the next. Let us assume now, a second arrangement K with $k < g$ blocks was found. $k < g$ implies that there is at least one column with no change to the next column. However, in this case the two columns are identical which contradicts our original assumption that all duplicates have been removed beforehand. Thus, $k \geq g$ and K cannot be a better arrangement than G . \square

[Dall 98] presents a method to generate a Gray code arrangement in linear time in N^* . However, in most practical applications $N^* \ll 2^{2n}$ holds. Then the Gray code arrangement is only suboptimal and other, more efficient heuristics must be found. These heuristics are based on the *hamming distance* of binary vectors [Boss 92] (see chapter 2.1.3). The input patterns are arranged such that two succeeding vectors have the lowest possible hamming distance. From this point of view, the Gray code arrangement corresponds to the special case where the hamming distance of two succeeding patterns is always 1. In the sequel, three heuristics based on the hamming distance will be presented.

1. *Complete Search (CS)*. Two sets A and B are created where A is unordered and B is ordered. Initially, A contains all the input vectors, B is empty. A first vector v_1 is arbitrarily removed from A and attached to B . Then A is searched for the vector v_2 with the lowest hamming distance to v_1 . It is removed from A and attached to B as the successor of v_1 . This process is repeated for the best successor of v_2 etc. until set A is empty. B contains then the optimized arrangement. The cost of this approach is $O(N^{*2})$ because for any vector v_i all the remaining vectors in B must be investigated in the worst case.
2. *Pool Search (PS)*. In this case, a third set P (*Pool*) is introduced. Initially P is filled with p vectors from set A . Then PS works just like CS , but the vectors that are attached to set B are chosen from P and P is always refilled with an arbitrary vector from A . The cost of PS is reduced to $p \cdot O(N^*)$ since the search space is reduced.
3. *Gray code + Pool search (GPS)*. The results of the pool search can be improved without a significant increase in complexity if PS is preceded by a *Gray code arrangement*.

For all the search steps, the same hash and LUT methods can be applied as for duplicate-removal [Dall 98].

4.5.4 Optimization Results

The algorithms described so far have been implemented into a computer program called *ASSeT (Activity Simulator Based on Set Theory)*. In this section, the optimization algorithms are investigated before the experimental results will be presented that were obtained

with *ASSeT*. Five different optimization methods were evaluated for their runtime and efficiency under different statistical assumptions for the input streams:

1. No optimization at all (*NoOpt*)
2. Duplicate-removal only (*NoDup*)
3. *NoDup* + *PS* with pool size $p=16$ (*PS*)
4. *NoDup* + *GPS* with pool size $p=16$ (*GPS*)
5. *NoDup* + *CS* (*CS*)

Unless stated otherwise, 20 000, uniformly distributed test patterns were used per experiment. Their static as well as their switching probabilities were chosen to 0.5. An optimization is said to be successful if the number of blocks could be reduced by more than 50%. Otherwise, the original arrangement is to be preferred since it doesn't require the delayed signals $x(t-T)$, which reduces the number of blocks by half. In the following figures, *PIs* denotes the number of primary input signals, *blocks* are the number of blocks, and *time* is the optimization or simulation time.

Uncorrelated Patterns

For the first experiments the input patterns are uncorrelated white noise. According to the results depicted in figure 4-18, *NoDup* is only successful for $PIs < 8$, while *CS* and *GPS* succeeded up to $PIs \approx 16$.

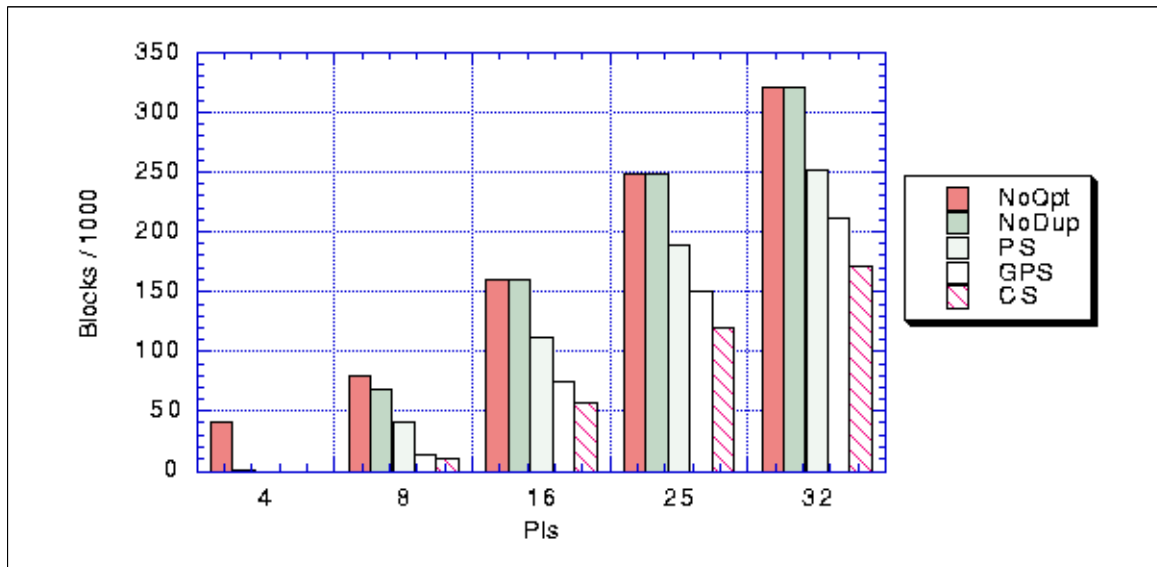


Figure 4-18: Optimization for uncorrelated vectors

FSM-Like Patterns

In FSMs with dozens of flip flops, only a small fraction of all possible states are actually reachable. Moreover, the state probabilities are usually extremely non-uniform. There are states that correspond to the regular operation mode of the FSM and other states that represent exceptional modes. The former have high probabilities, the latter probabilities near zero. This behavior was modeled in the experiments whose results are depicted in figure

4-19. The set from which the input patterns were chosen, had been restricted to 10% of all possible input patterns. The non-uniform state probabilities were modeled by a binomial distribution. Under these assumptions the limits for success are shifted to higher numbers of PIs. While *NoDup* is successful until $PIs=8$, *GPS* and *CS* succeed even for $PIs=32$.

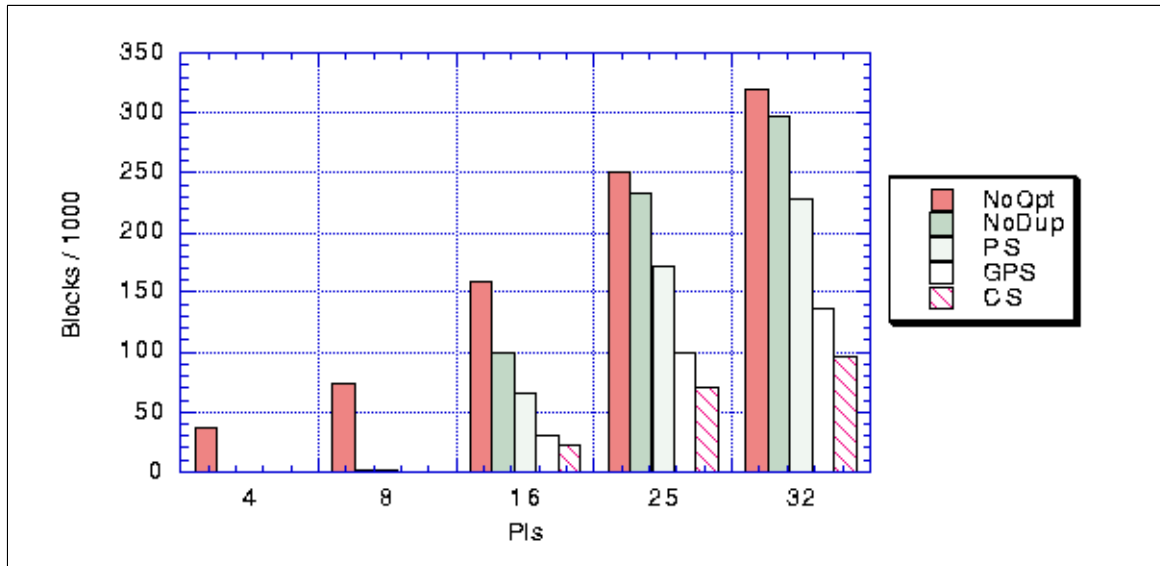


Figure 4-19: Optimization for FSM-like vectors

Spatially Correlated Patterns

For the experiments of this section the PIs were divided into groups of eight signals. For circuits with less than nine inputs, group size four was chosen. The test vectors were generated such that the second to the fourth or eighth signal of each group equal the first signal with probability 0.8. Signals of different groups are uncorrelated. The results for spatially corre-

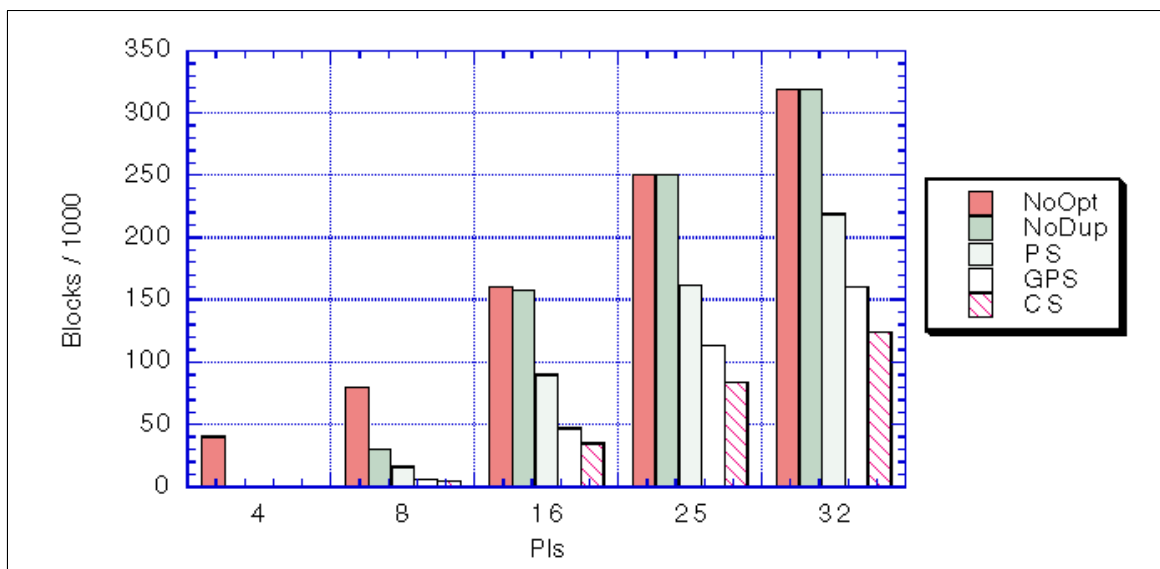


Figure 4-20: Optimization for spatially correlated vectors

lated patterns (figure 4-20) are slightly worse but comparable to those of the FSM-like patterns. Thus, the original assumption is confirmed that heuristic rearrangement is best suited for correlated patterns.

Low Active Signals

In many practical applications, only few signals have a switching activity as high as $\alpha_{x_i} = 0.5$. In order to investigate the effect of lower switching activities, α_{x_i} was reduced to 0.1 for the experiments whose results are depicted in figure 4-21. Obviously, low active signals can only be improved if $PIs \leq 8$. For circuits with more PIs the heuristics may even deteriorate the results. The block sizes in the original signal representation are so good that they cannot be improved by the heuristics which focus rather on local than on global optimizations.

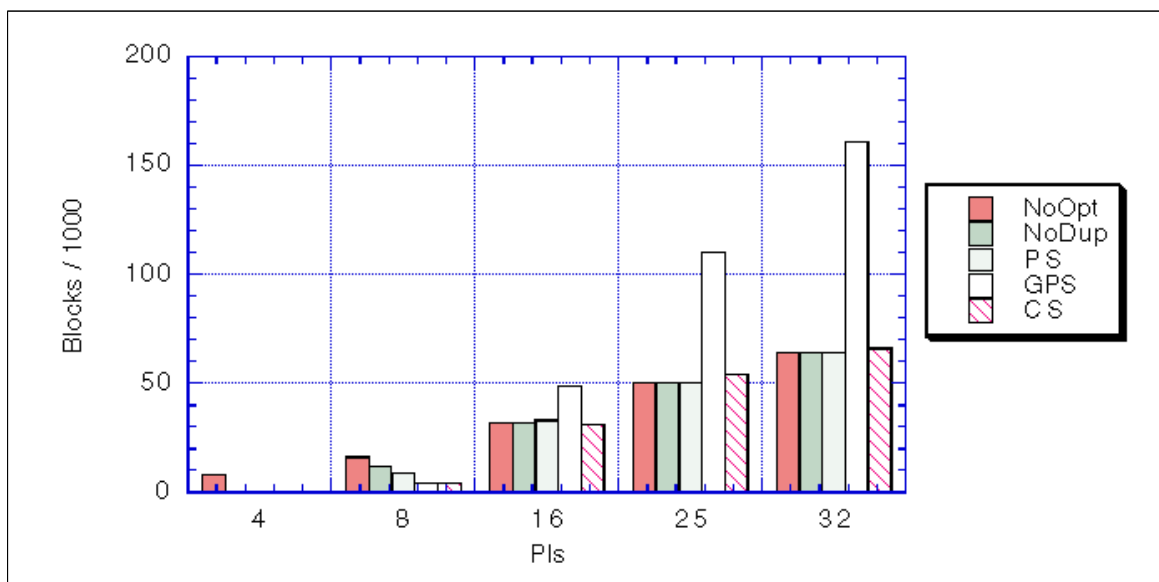


Figure 4-21: Optimization for low active vectors

CPU Time

This section provides the average runtimes of the previous experiments. They are depicted in figure 4-22. The optimization times for 4 PIs were omitted since they are negligible, just as the CPU times for *NoDup*. All pattern operations, like comparing two patterns or computing their hamming distance, become more expensive with an increasing number of input signals. Therefore, the optimization times increase with the number of PIs. However, it is surprising that *GPS* requires considerably less CPU time than *PS*. Obviously the pre-optimization with the Gray Code arrangement does not only improve the final optimization result but does also speed up the following *Pool Search* since similar patterns are already very close.

Discussion

The results show that it has to be clearly distinguished between low active and high active input streams. For the former, the heuristics cannot decrease the total number of blocks

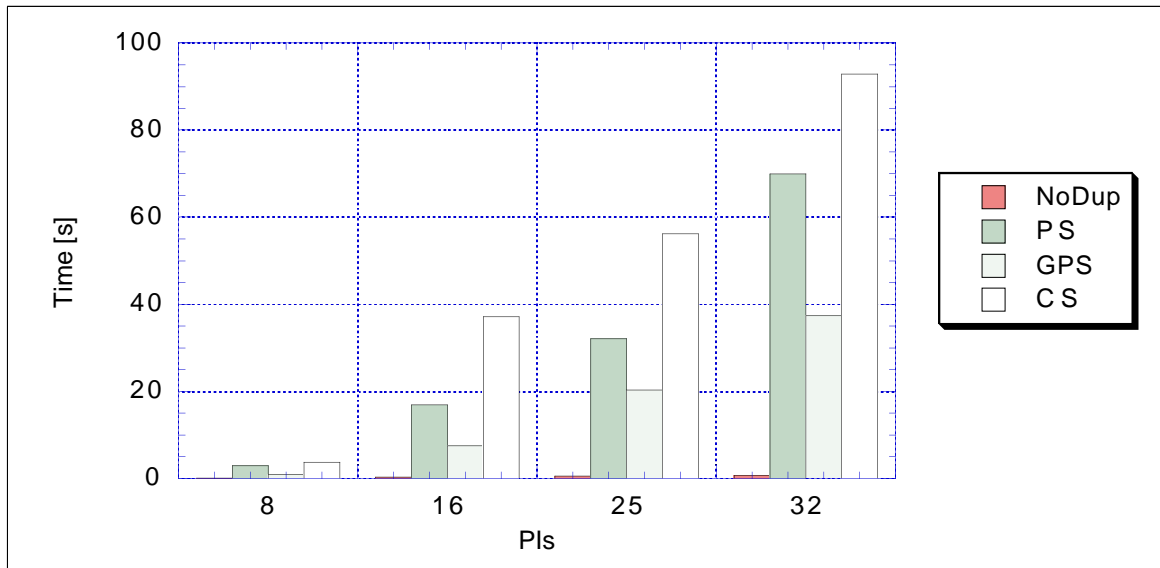


Figure 4-22: Average optimization times

since low active patterns consist of large blocks from the start. In that case, further optimization is only possible if the number of PIs is eight or less. Otherwise, the heuristics even tend to deteriorate the original arrangement. As a consequence, rearrangement as well as the delayed signals should be omitted for low active circuits.

In all other experiments the complete search algorithm (*CS*) yields the best results. Duplicate-removal (*NoDup*) is very fast, but as it was to be expected, it is only successful for circuits with a low number of PIs. The success of the pool search (*PS*) is rather modest. But it can be significantly improved with a preceding Gray code rearrangement (*GPS*). *GPS* is significantly faster and yields much better results than *PS*. *CS* still outperforms *GPS* in terms of block reduction by some 20-25% at the cost of 2-3 times increased CPU time.

GPS and *CS* are quite successful for smaller numbers of PIs, but all methods become less successful with an increasing number of PIs. This can be explained as follows. Consider the example of figure 4-16 where many PIs imply the same number of rows. If two columns are to be exchanged in order to optimize the block length in one row, it is very likely that a block in one or more other rows gets destroyed, thus nullifying the intended optimization.

Block optimization is only reasonable as long as the number of blocks can be reduced by half or more. Otherwise, a simple set simulation without rearrangement is to be preferred because the delayed signal can be omitted, which reduces the block number by 50%. Hence, according to the results of this section, block optimization for uncorrelated input patterns is promising only up to 25 PIs. The approach is better suited for spatially correlated PIs. There, the limit is between 25 and 32 PIs or even higher.

Some Comments Concerning Runtime of the Optimization Algorithms

Usually, a large number of typical input patterns is obtained through high level simulation. Several circuit alternatives are then investigated using the same patterns. However, rearrangement must only be performed once for all the alternatives. Under this assumption, the

runtime of the rearrangement is less crucial for the overall development process.

4.5.5 Simulation Results

After the optimization algorithms were empirically investigated, the experimental results that are presented in this section, focus on the effect of pattern rearrangement to the set based simulation. The following circuits were chosen for these experiments after mapping them on the static CMOS library from chapter 3.

circuit	f51m	alu2	alu4	c1908	c3540
gates	112	270	551	245	791
PIs	8	10	14	33	50

Table 4-4: Parameters of the benchmark circuits

The figures in the following subsections will give the simulation times for logic simulation, simple, unoptimized set simulation without delayed signals, and set simulation with *GPS* and *CS*. They will be denoted as *logic*, *NoDel*, *GPS*, and *CS*, respectively. The accuracy is consciously omitted since the results of the set simulations are always equal to the results of logic simulations. Unless stated otherwise, the test patterns were generated under the same assumptions as in section 4.5.4. The following figures won't be commented in detail, because the trends of the simulation times corresponds to the block number reductions that have been presented and discussed in the last chapter. Summarizing comments are left for a final discussion at the end of this section.

Uncorrelated Vectors

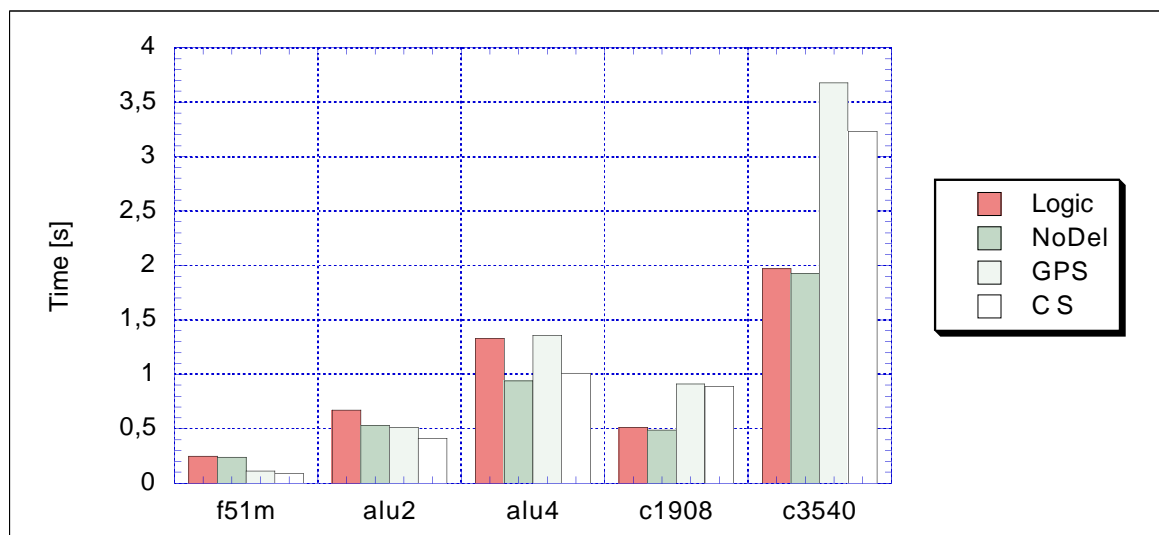


Figure 4-23: Simulation times for uncorrelated vectors

FSM-Like Vectors

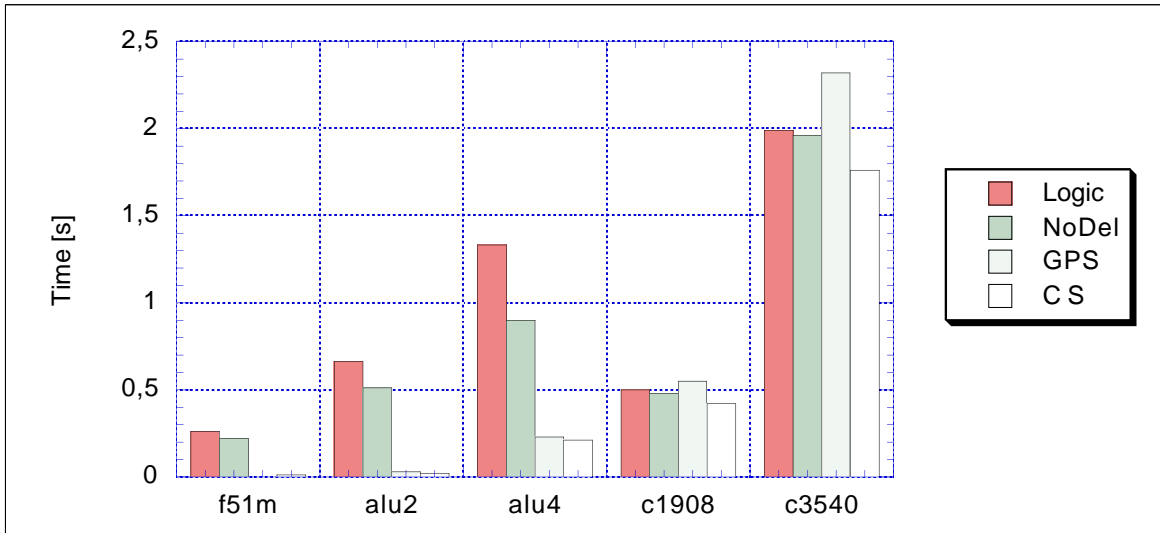


Figure 4-24: Simulation times for FSM-like vectors

Spatially Correlated Vectors

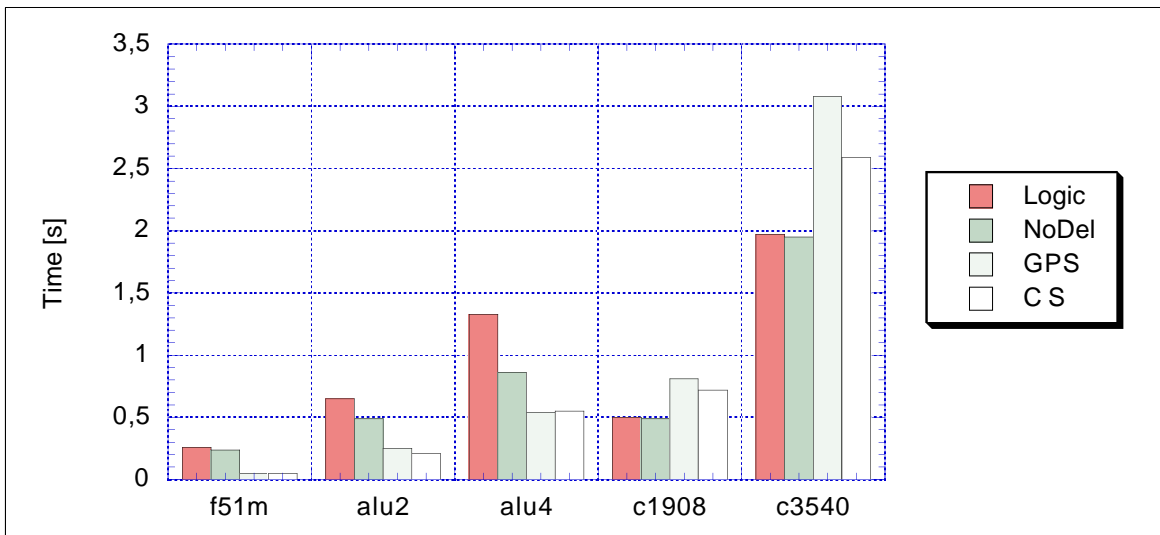


Figure 4-25: Simulation times for correlated vectors

Low Active Signals

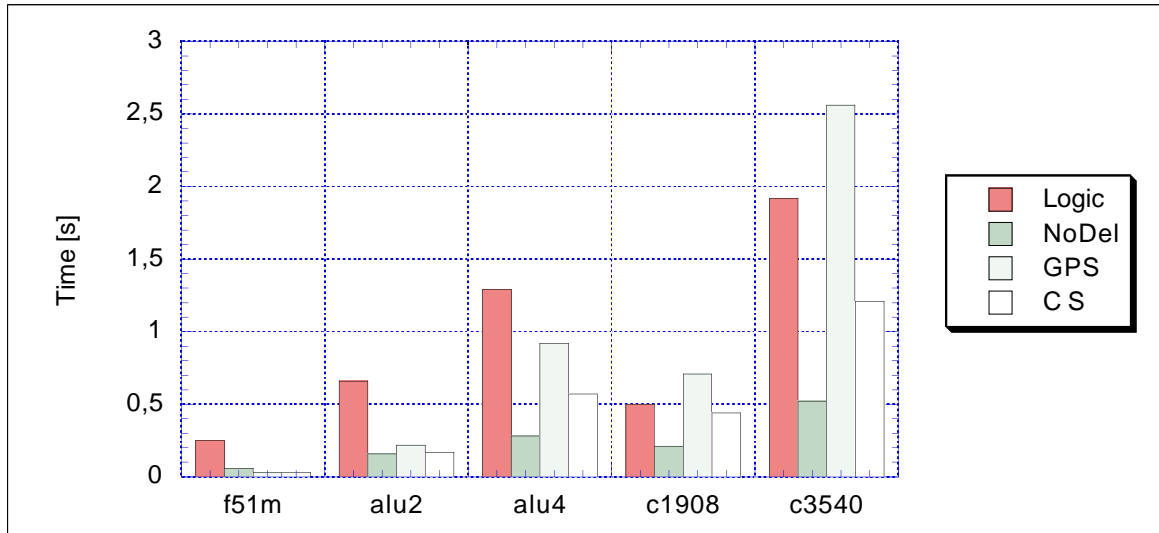


Figure 4-26: Simulation times for low active signals

Sequential Circuits

As a last experiment, some sequential circuits were simulated using the configuration of figure 4-7. The circuits were mapped on the GFN120 gate library [IMS 91]. The state bits were generated with the approach of figure 4-9. The next state logic was realized by RTL-simulation with random input patterns¹. Hence, the total number of PIs of each benchmark circuit is given by the number of PIs of the original sequential circuit plus the number of its registers. Table 4-5, figure 4-27, and figure 4-28 depict the major circuit properties, the optimization results, and the simulation times, respectively.

circuit	s382	s820	s1196	s1494	s5378	s9234
gates	162	293	553	649	2781	5584
PIs	3	18	14	8	35	36
FF	21	5	18	6	179	211
PIs+FF	24	23	32	14	214	247

Table 4-5: Parameters of the benchmark circuits

In particular, the results of the experiments with the large sequential circuits can be compared to those with low active patterns. In fact, as the number of flip flops increases, the number of flip flops with low active output signals increases as well. This can be explained as follows: while the typical operation mode of a sequential circuit requires usually only a small fraction of all possible states, a major part of all states must be reserved for excep-

¹. This pattern generation method will be referred to as *RTL-generated patterns* in the sequel.

tional states like error or reset modes. The flip flops that are used for coding these states have a low switching activity.

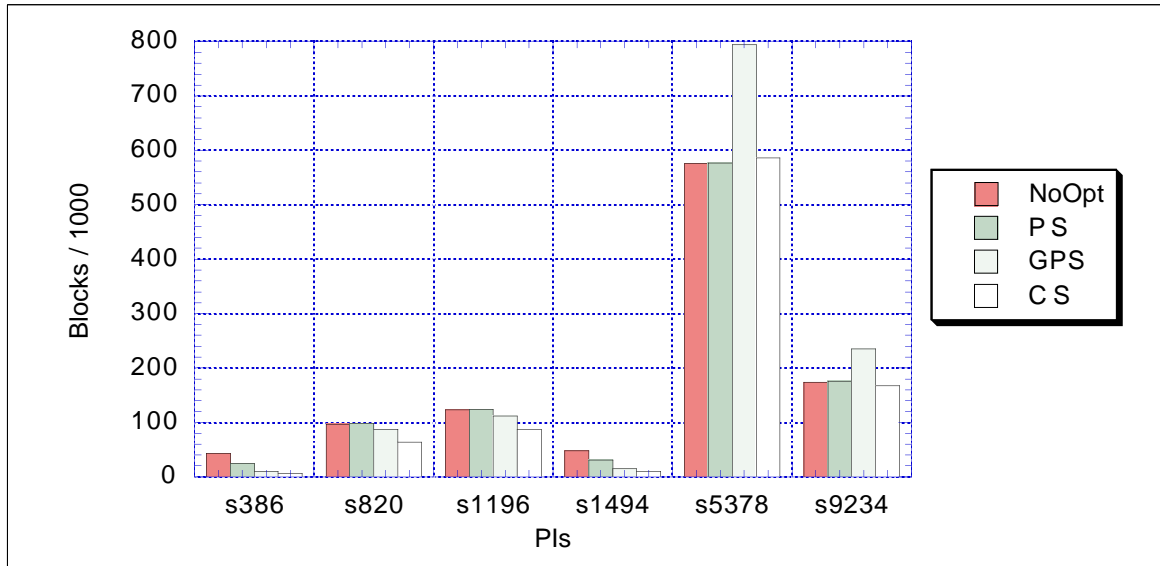


Figure 4-27: Block optimization for sequential circuits

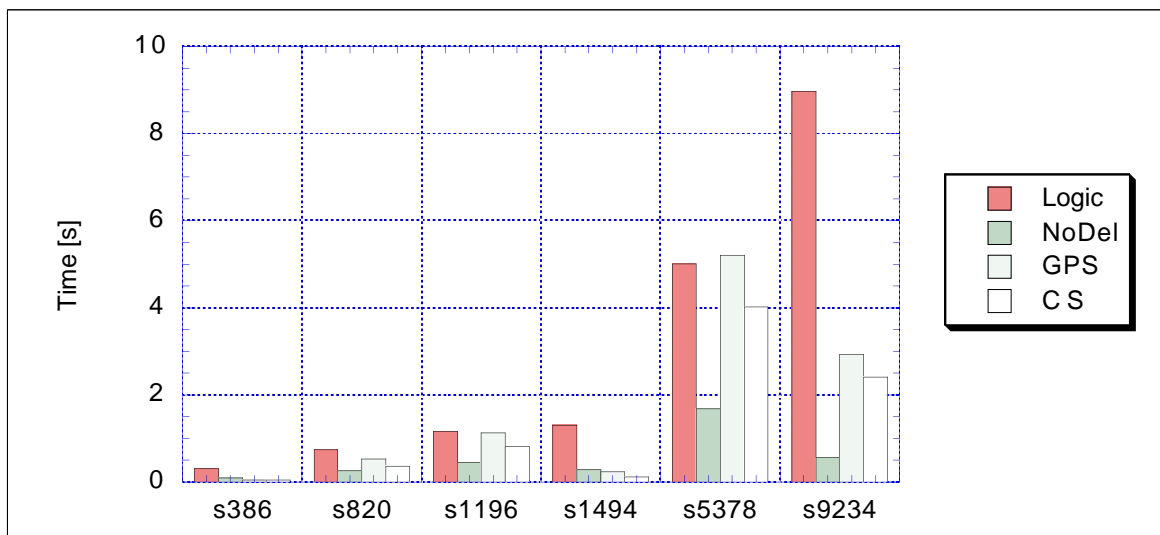


Figure 4-28: Simulation times of sequential circuits

Discussion

In general, the runtimes of the simulations reflect the results of the block optimization methods. However, the speedups are not as high as implied by the latter. The reason is that the blocks are partly destroyed as they propagate through the circuits. Thus, block optimization can only speedup the simulation process for circuits with a relatively low number of PIs.

On the other hand, set based simulation without any optimizations, which is referred to as *NoDel*, is very efficient if there are many low active PIs among the primary input signals. The simulation results show that the combinational parts of large sequential circuits belong

to this class. In that case, the CPU time of the simulation process can be importantly reduced compared to logic simulation without the need of time consuming optimization steps. Hence, set simulation is especially well suited for applications that are connected with the methods that were presented in section 4.4.

4.6 Trading Speed versus Accuracy

Pattern rearrangement accelerates the simulation only under certain conditions. In general, a high number of uncorrelated primary inputs prevents efficient rearrangement. On the other hand, in many applications it would be acceptable to sacrifice some accuracy for speed. In this section an approach will be proposed that makes such a trade-off possible.

4.6.1 Grouping

It was mentioned before that the input streams are either created by high level simulation or they are based on real data. If such streams are analyzed more closely, it reveals that there often exist certain groups of correlated signals, while the correlations between signals of different groups are negligible. Such groups of highly correlated signals will be called *correlation groups* in the sequel or simply *groups*, depending on the context.

Consider e.g. the model from figure 4-11. It implies *correlation groups* for the register outputs of FSMs. Further examples are protocol or code processing devices like bus interfaces, LAN adapters, or processors. In any code, the code words can be partitioned into fields with specific purposes. E.g. figure 4-29 depicts the typical format of a two-address machine instruction. It consists of three fields: the operation code, the address of the first operand, and the address of the second operand. While the three fields are independent, the bits within the fields maybe correlated since some opcodes are more likely than others [Patt 96] and most memory accesses have a local character [Raba 96].

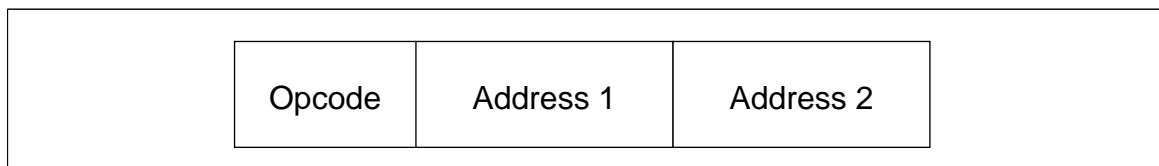


Figure 4-29: Two-address instruction

Figure 4-30 shows an abstract model of a circuit that has to process such an input stream. It has eight inputs $a-h$. It is assumed that there exist two correlation groups $G_1=\{a, b, c, d\}$ and $G_2=\{e, f, g, h\}$.

Because of the statistical independence between groups, the assignment of rows in G_1 and G_2 to each other is arbitrary and can be changed randomly. Thus, it seems likely to break the input vectors into partial vectors that are defined by the correlation groups (figure 4-31). The partial vectors are then optimized separately as depicted in figure 4-32. Thus, the two major requirements for efficient block optimization can be met: low number of rows and highly correlated patterns. However, the question arises how these partial vectors can be

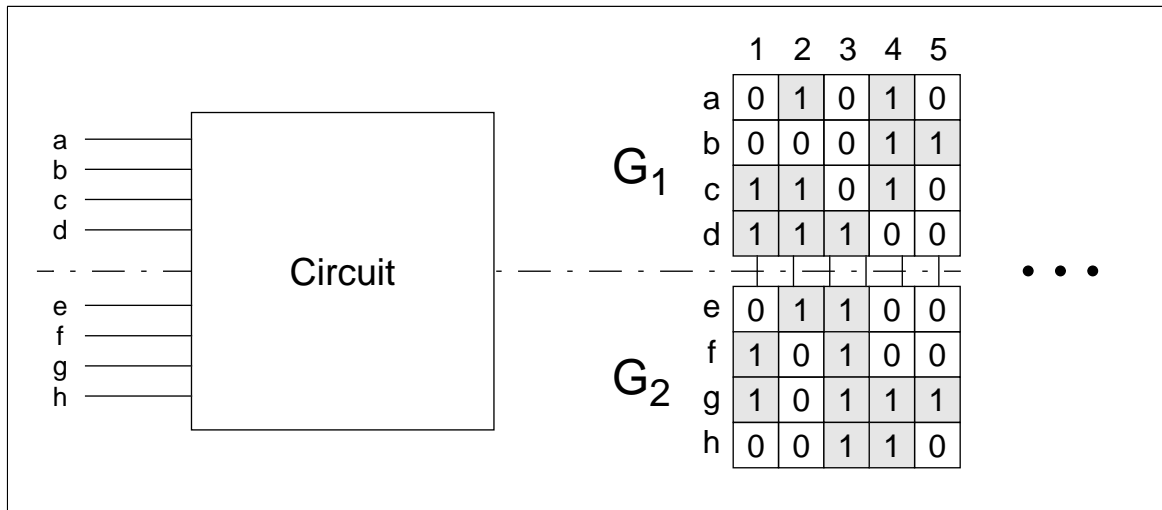


Figure 4-30: Correlation groups

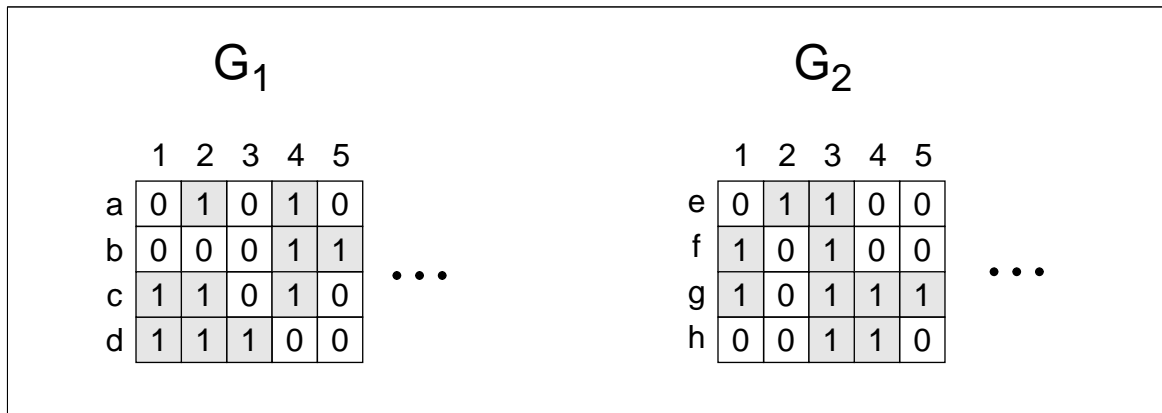


Figure 4-31: Partial vectors

correctly handled by the simulation algorithm. It would be an interesting possibility to introduce a multidimensional signal representation. Each dimension would represent one correlation group. While the operations within a group wouldn't change, processing of signals of different groups would lead to multidimensional set operations. Even in the two dimen-

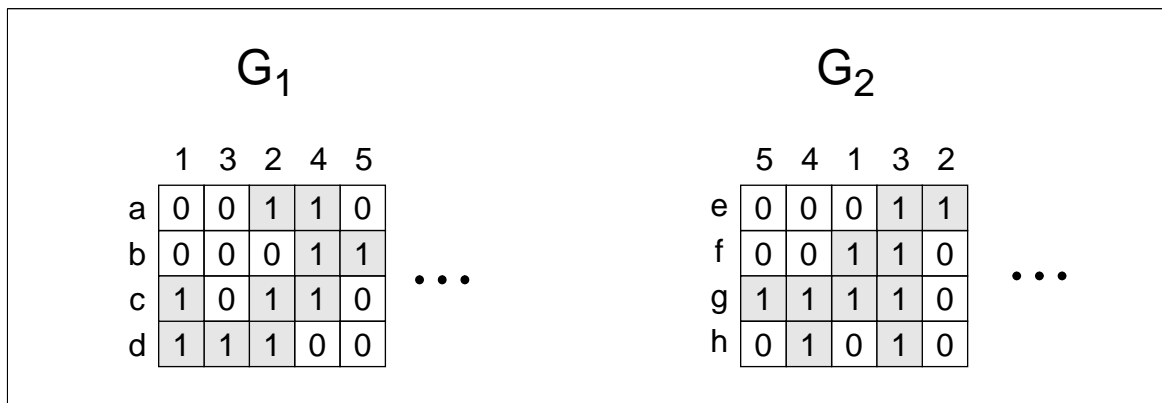


Figure 4-32: Separately optimized partial vectors

sional case, where efficient algorithms could be adopted from other applications like *design rule checking*, these operations are still quite costly in terms of CPU time and memory requirements; not to mention the requirements in case the data is to be modeled with three or more dimensions. Hence, there remain two open questions: how to determine the correlation groups and how to process signals of different correlation groups efficiently.

4.6.2 Determination of Correlation Groups

If corresponding information is available from system design, the correlation groups can be manually identified by the designer. However, an algorithmic solution is to be preferred in order to support today's highly automated design flows and for reliability reasons.

The algorithm that is proposed here, is based on the pairwise correlation coefficient (definition 2-54). It consists of three phases. During the first phase, the magnitudes c_{ij} of the correlation coefficients are computed. They are stored in the array SC . A complete entry in SC consists of the correlation coefficients c_{ij} plus references to the according variables x_i and x_j :

$$SC(k) = (c_{ij}, *x_i, *x_j). \quad (4-33)$$

```

For all variables  $x_i, x_j$  ( $i \leq j$ ) compute magnitude of the pairwise
correlation coefficients  $c_{ij} = |r_{ij}|$ 
Store the correlation coefficients in array SC:
  SC[k] = (cij, *xi, *xj)
Sort SC in decreasing order of SC[k].cij
k=0
while(SC[k] > L2) {
  if(xi belongs to a group)
    Gi := the group xi belongs to
  else
    Gi := {xi}
  if(xj belongs to a group)
    Gj := the group xj belongs to
  else
    Gj := {xj}
  if(|Gi| + |Gj| ≤ LG OR SC[k] > L1) {
    Gi := Gi ∪ Gj
    delete(Gj)
  }
  k := k+1
}

```

Listing 4-1: The grouping algorithm

During the second phase, signals are merged into groups. The vector SC is processed in descending order of its correlation coefficients $SC[k].c_{ij}$. The corresponding signal pairs are conceivably merged into one group using the following operations:

- If either x_i or x_j don't belong to any group, a new group G_n is created containing x_i and x_j .
- If x_i and x_j belong to two different groups G_i and G_j , respectively, the two groups are merged into one single group.
- If x_i and x_j already belong to the same group, no action is necessary.

The second phase ends when $SC[k].c_{ij}$ is below a certain limit L_1 . Then, during the third phase, merging is only allowed if the resulting groups do not exceed a limited *group size* L_G . This prevents the groups from becoming too large, in case that the signals are only weakly correlated.

As soon as $SC[k].c_{ij}$ is below a second limit L_2 the grouping algorithm stops, the remaining correlations are negligible. Listing 4-1 depicts the grouping algorithm in a C-like pseudocode notation.

For the experiments that are presented later, the limits have been chosen to $L_1 = 0.3$ and $L_2 = 0.1$, while the *group size* L_G has been a variable parameter.

4.6.3 Vector Recomposition

The original set simulator can be used to simulate the separately optimized correlation groups if the groups are arbitrarily recomposed to new, complete input patterns, as depicted in figure 4-33.

		1	2	3	4	5
G_1	a	0	0	1	1	0
	b	0	0	0	1	1
	c	1	0	1	1	0
	d	1	1	1	0	0
-----		...				
G_2	e	0	0	0	1	1
	f	0	0	1	1	0
	g	1	1	1	1	0
	h	0	1	0	1	0

Figure 4-33: Input patterns after recomposition

However, this recomposition is not of the random type that was required in section 4.6.1, since all the groups have passed identical optimization steps. Therefore, new correlations between previously uncorrelated signals of different groups have been introduced. For instance, in figure 4-33 signals a and f as well as b and e became identical. This effect can

be alleviated to some extent if the grouping algorithm of section 4.6.2 is extended by a fourth phase. During this phase, small groups and the remaining uncorrelated PIs are merged until their group size reaches L_G . Thus, the probability for equal signals in different groups and the correlations coefficients between groups are decreased at the expense of some optimization efficiency.

4.6.4 Results

For each experiment, 10 000 test patterns were generated. *Group sizes* L_G of 2, 4, 8, and 12 PIs were investigated with uncorrelated and correlated vectors. The experiments with correlated vectors were performed with the combinational part of sequential circuits¹. *GPS* was chosen for vector optimization as an acceptable trade-off between runtime and optimization quality. The experiments were carried out with six combinational and five sequential circuits. For the sake of clarity only the average results are depicted here. Detailed tables can be found in appendix C.2.3.

The following statistics were recorded during the experiments:

- *speedup* of the simulation with signal grouping over simulation without signal grouping.
- *global error*: the relative error in the overall toggle rate.
- *local error*: the average relative error per gate (equation 4-34). As suggested in [Schn 95], only signals with a switching activity $\alpha_i > 0.01$ were considered. The reason for disregarding signals with lower switching activities is that their contribution to the overall power consumption is very small but on the other hand, they cause large relative errors. Imagine a signal making two transitions during a 20k pattern simulation but it should only make one; the relative error is 100% but power dissipation of this signal can be neglected.

$$\text{error}_{\text{local}} = \text{average} \left(\left| \frac{\alpha_{i_{\text{exact}}} - \alpha_{i_{\text{estimated}}}}{\alpha_{i_{\text{exact}}}} \right| \right) \Bigg|_{\text{all gates } i \text{ with } \alpha_i > 0.01} \quad (4-34)$$

Figure 4-34 depicts the speedup through signal grouping. While the speedup is quite high for small *group sizes*, it decreases with increasing *group size* since the heuristic optimizations become less efficient. The speedups for the sequential circuits decrease slower than for combinational circuits. The reason for this behavior is in the grouping algorithm. The parameter *group size* limits the actual group size only for weakly correlated signals. Thus, if large groups of highly correlated PIs exist like in large sequential circuits, the grouping algorithm merges them into one single correlation group, regardless the *group size* parameter, resulting in a poor speedup for small L_G . On the other hand, these highly correlated groups can be much better optimized than similar groups of uncorrelated signals. This results into relatively high speedups for large L_G .

1. For more details see section 4.5.5, *sequential circuits*.

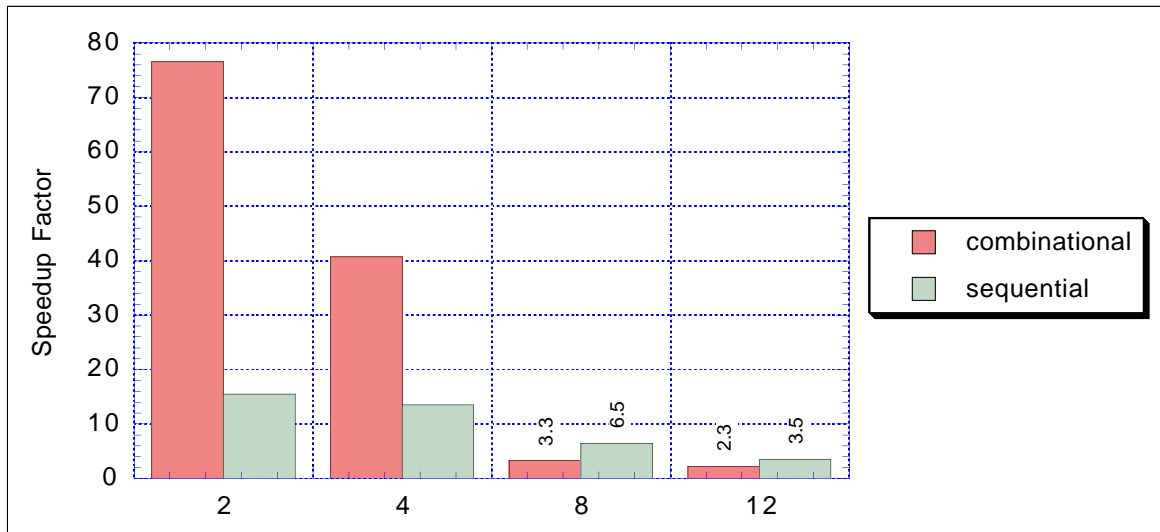


Figure 4-34: Speedup with signal grouping

The global errors are depicted in figure 4-35. As expected, larger *group sizes* yield more accurate results. But even for a *group size* as small as 2, the average global error is far below 5% for uncorrelated PIs. On the other hand, for the sequential circuits, the average errors are higher. This demonstrates the effect of the approximations made by the grouping heuristic of listing 4-1 where several correlations are neglected. However, the average errors are below 15% and thus still acceptable for many applications. The result for *group size* 4 is unexpected since the error is higher than for smaller groups. The major reason is the *random error compensation effect* that was first reported of in [Kapo 94]. When summing up the signal toggles over the whole circuit, underestimated nodes and overestimated nodes may partly compensate each other. If the average actual group size of two simulation runs differ only slightly, as indicated by the speedups for sequential circuits for *group size* 2 and 4, this random effect can become dominant for the global error.

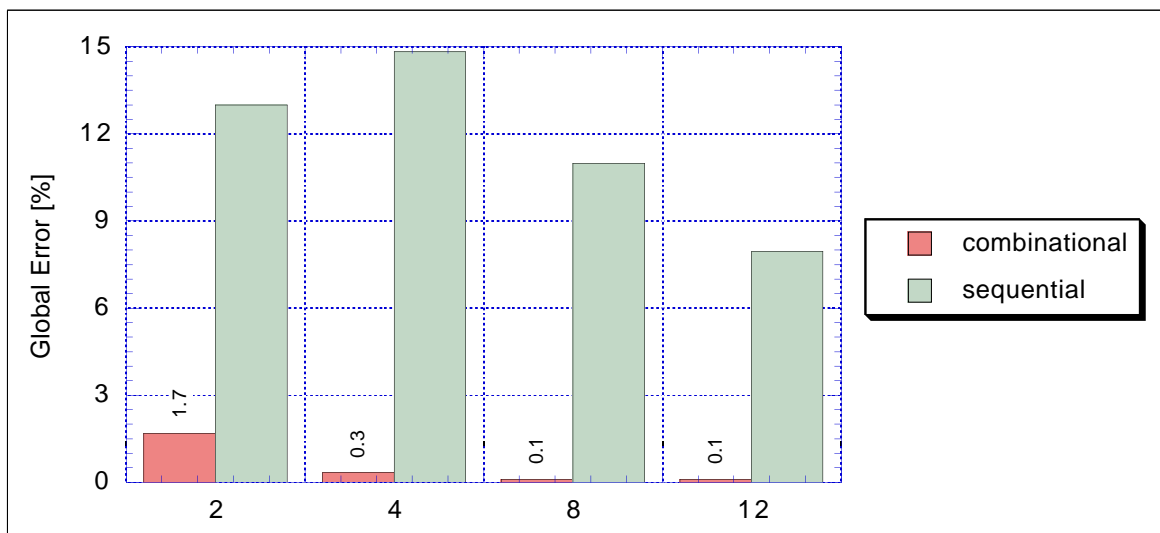


Figure 4-35: Global error with signal grouping

This theory is further supported by the average gate related errors which are depicted in figures 4-36. They clearly increase with decreasing *group size* since they don't profit from the *random error compensation effect*. The local errors are particularly important for logic optimization algorithms since those rely on accurate data for individual gates. For combinational circuits, the local errors are acceptable for *group size* 4 and higher. For the sequential circuits, even *group size* 12 yields an average local error of 14%.

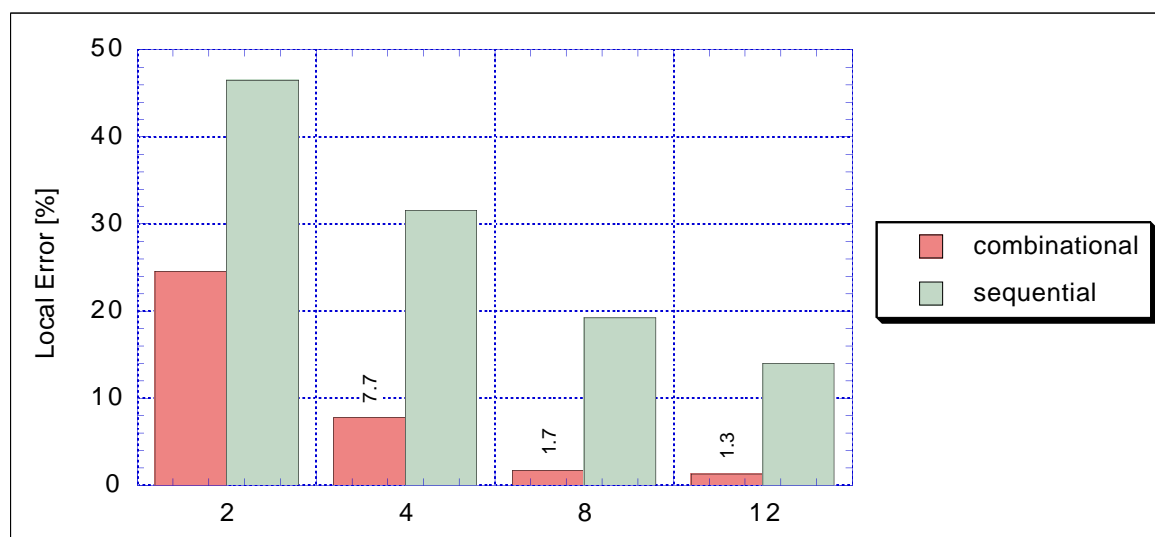


Figure 4-36: Local error with signal grouping

Conclusion

As a general conclusion of these experiments, two observations can be made. If the overall power consumption of a circuit is to be estimated, group oriented pattern optimization followed by arbitrary vector recomposition yields remarkable speedups and acceptable accuracy even for a *group size* as small as 2. However, as the input patterns get more correlated like in the combinational part of sequential circuits, larger groups between 8 and 12 signals are to be preferred, trading some speed for accuracy. However, the performance penalty due to increasing *group size* is relatively modest for highly correlated PIs compared to uncorrelated PIs. On the other hand, if high accuracy is required for each single gate in the circuit, the *group size* L_G must be further increased to at least 4 and 12 for combinational and sequential circuits, respectively.

4.7 Summary

This chapter started with a detailed overview over state-of-the-art techniques for power estimation on logic level. The focus was put on switching activity estimation under the zero delay assumption where two, fundamentally different approaches exist: logic and probabilistic simulation. While both suffer from long runtimes if high accuracy is required, there exist other problems that are specific to each approach. For logic simulation based approaches they are related to pattern generation, for probabilistic methods, accurate correlation handling poses the major challenge. It was shown which basic methods exist to

address these problems. Most of them are limited to purely combinational circuits. However, since sequential circuits represent an important circuit class, several methods from the literature were presented which allow to apply the estimation methods for combinational circuits to sequential ones.

In the second part a new, set based simulation approach was proposed. From a mathematical point of view, it fits in the gap between logic and probabilistic simulation. However, it still depends on input patterns. The basic approach has proven to be quite efficient for low active circuits like the combinational part of large sequential circuits. For other types of circuits several optimizations were proposed, which rely on vector rearrangement. The efficiency of these approaches depends strongly on the statistical properties of the input patterns. But in general, efficiency is decreased with the number of primary input signals of the circuit. Therefore, a method was presented in the last section that enables a trade-off between accuracy and CPU time. Its main idea relies on partitioning the PIs into smaller groups of correlated signals. These groups are then arbitrarily recomposed after separate optimization. It was shown by experimental results that average speedups as high as 75 times can be obtained over the accurate approach. The global accuracy loss depends upon the parameter *group size* and the statistical properties of the primary input streams. For uncorrelated PIs the global error is far below 10% even for a *group size* as small as 2. If the PIs are more correlated or if high gate oriented accuracy is required, larger *group sizes* must be chosen.

The ZDM is a convenient simplification during the development of a novel simulation algorithm. It can also be applied early in the design cycle. A ZDM simulator can yield a first rough estimate of the actual power consumption before technology mapping, when the circuit structure and gate delays are not available yet [Schn 95]. However, if gate delays are neglected, certain effects cannot be taken into account. This may lead to important estimation errors.

After some definitions and the detailed presentation of the limits of a ZDM based simulator, this chapter shows how the set based simulation approach can be extended to any desired delay model. A new signal representation is introduced that extends signal waveforms, which are usually represented by single dimensional vectors, to 2-dimensional arrays. The RDM simulation algorithm itself can be considered as an extension of common, event based logic simulators. The approach presented here can be applied to a whole class of simulation algorithms. This is shown by applying it to the bitparallel power estimator which was first presented in [Schn 95]. While being very efficient in runtime it has also been restricted to the ZDM so far. Beside the basic technique, two methods to improve the runtime are proposed: an exact approach and an approximation. Detailed experimental results prove the runtime efficiency of the novel algorithms and give new insights about the impact of the delay models on the accuracy of the simulation result.

5.1 Limits of the Zero Delay Model

The set based simulation method that was presented in chapter 4 has been based on the ZDM so far. Thus, it suffers from all the drawbacks of any ZDM based power estimator as it cannot model effects that are introduced by the gate delays like hazards and glitches. Consider the circuit of figure 5-1. Y_{ZDM} and Y_{UDM} depict the output waveform of the circuit under two different delay assumptions: the ZDM and the *unit delay model (UDM)*. Obviously, the ZDM does not capture the hazard on the output Y which is caused by the delays of the two inverters. As it is shown later, hazards have an important contribution to the overall power consumption. As the ZDM generally tends to underestimate the signal activity, it also underestimates the power consumption.

Most RDM logic simulators rely on an event based technique [Lehm 94]. An *event* is an ordered pair which consists of a signal value and the time when the according signal takes

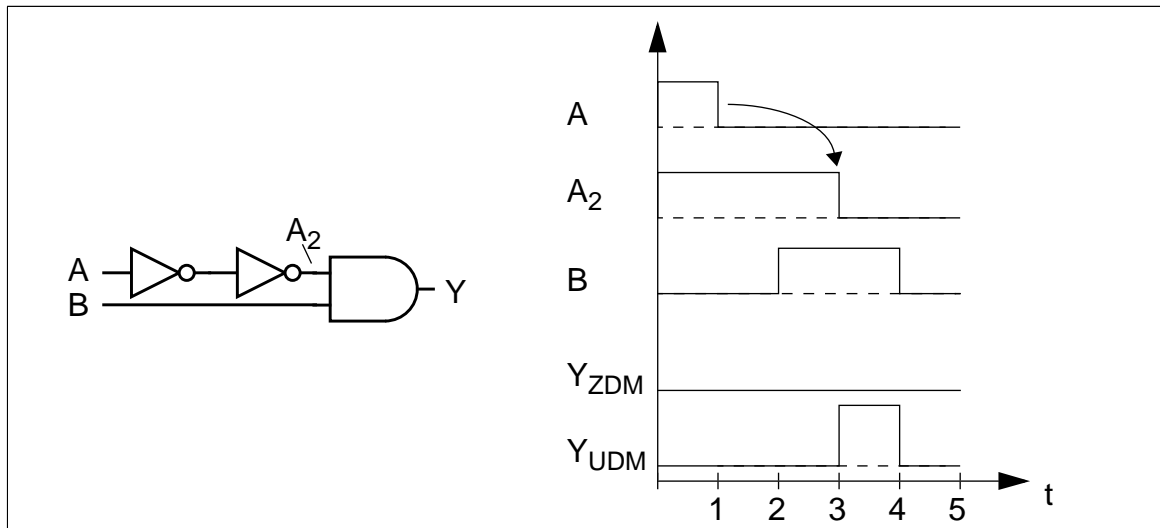


Figure 5-1: Circuit simulated with ZDM and UDM

this new value. Signal waveforms are stored in event lists. When the output waveform of a gate is to be computed, the events on its input waveforms must be processed in temporal order. The correct temporal order of the event processing is guaranteed by the global simulation time t which controls the simulation flow.

Probabilistic simulators can be extended to more sophisticated delay models with the concept of *probability waveforms* [Najm 90]. In this case, any event is defined by the new signal value, the event time, and the probability that the event actually occurs. These probabilities can then be propagated through the circuit using common probabilistic approaches. While today suitable techniques for logic and probabilistic simulation are state-of-the-art, no such method exists yet for the set based approach.

5.2 Real Delay Model

The target delay model of the algorithm that is presented in the sequel, is based on the models of table 2-2 with $n=0$ and $m=1$. It represents a commonly used model in existing simulators, like the *standard delay model* in [SyLi 96]. The algorithm can be easily adapted to other models, though. Furthermore, VHDL's inertial delay model is assumed, including different delays for falling and rising edges. This results into two additional requirements for the RDM simulation algorithm. It must detect and distinguish between falling and rising edges in order to apply the correct delays. Further it needs a glitch detection mechanism that filters output pulses that are shorter than the gate delay.

5.2.1 Signal Representation

Set representation is cycle oriented. Each signal takes one single value per clock cycle. Now consider the signal waveform of figure 5-2. It depicts an arbitrary example for the output waveform of a gate in a real delay simulator. The vertical dashed lines indicate the begin-

ning of a new clock cycle. Analysis of the waveform reveals two shortcomings of the set representation. Firstly, there may exist multiple signal changes during one clock cycle. Hence, the time resolution of ‘1 clock cycle’ is too coarse. Secondly, any signal change gets a time stamp attached to it.

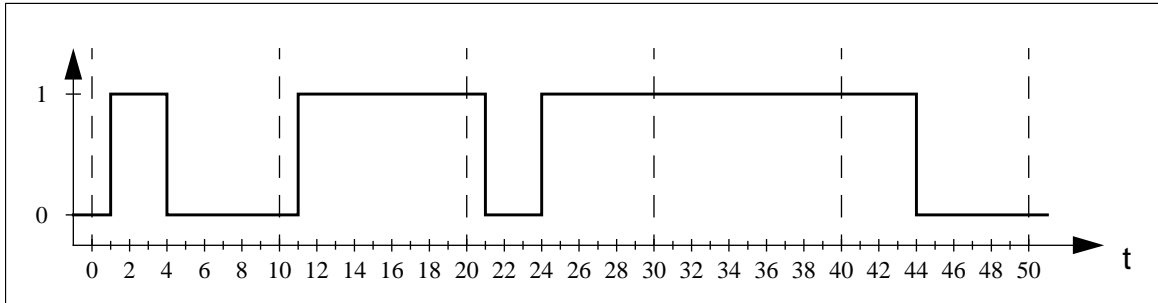


Figure 5-2: RDM signal

In ZDM simulators, signal waveforms are usually represented as single dimensional vectors. Each entry in the vector corresponds to the signal value during a specific clock cycle. It was outlined before that such a representation is not sufficient if real gate delays are to be taken into account. In order to manage the additional information that is introduced by the RDM, the time axis is divided into clock cycles. Instead of representing time as a single scalar value, it is represented by an ordered pair t_p , also called *CO-time*.

Definition 5-1: CO-Time

Given the scalar time $t_s \in T$ and a clock period τ , then the *CO-Time (Cycle-Offset Time)* is the ordered pair

$$t_p = (cc, \Delta t) \in \mathbf{Z} \times \mathbf{X} \text{ where } \mathbf{X} = \{x | x \in T \wedge 0 \leq x < \tau\}.$$

cc and Δt are given by

$$cc = \lfloor t_s / \tau \rfloor \text{ and } \Delta t = t_s \bmod \tau.$$

The set T is usually \mathbf{Z} or \mathbf{R} .

Figure 5-3 depicts the signal of figure 5-2 in the CO-time notation with $\tau=10$.

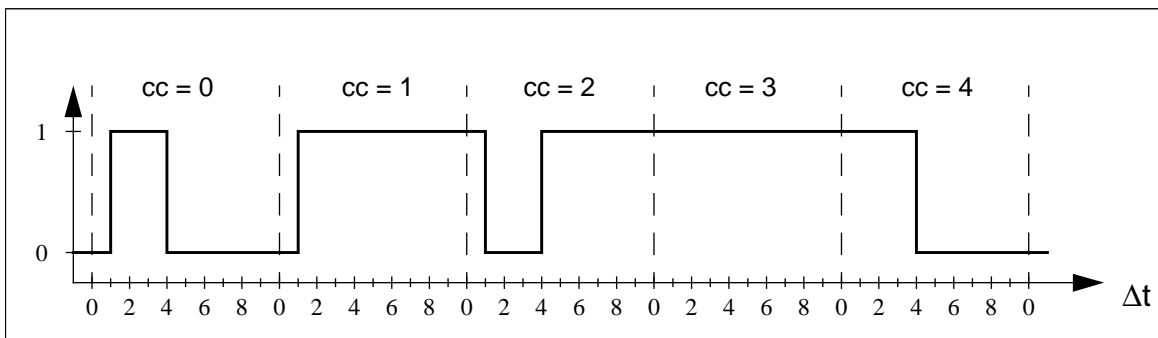


Figure 5-3: RDM signal in CO-time notation

Note that the labels at the y-axis are not unique anymore. Hence, the scalar time t_s can be computed from the CO-time by:

$$t_s = \tau \cdot cc + \Delta t. \quad (5-1)$$

Using the CO-time representation, the signal waveform can now be represented as a two dimensional array. The columns represent the signal values during one clock cycle cc while each row corresponds to a time offset Δt . The array notation of the signal waveform of figure 5-3 is depicted in table 5-1. Note the first row with $\Delta t < 0$. It represents the signal values at the end of the previous clock cycles. Obviously it corresponds to the last row at $\Delta t = 4$ shifted right by one position.

	cc				
Δt	0	1	2	3	4
<0	0	0	1	1	1
1	1	1	0	1	1
4	0	1	1	1	0

Table 5-1: Array representation of signal waveforms

The array representation of table 5-1 is called the *schedule of a signal* or simply *schedule*. A complete schedule consists of two arrays: the two dimensional array which contains the signal values, also called *value array*, and a single dimensional array, which stores the time offsets Δt that correspond to the rows in the value array. The latter is called the *offset vector*. It is depicted as the first column from the left in table 5-1. The rows of the value array including their time offset Δt are sometimes called a *vector event* or briefly *event*. The values of a vector event of signal X at Δt are referred to as $X(\Delta t)$. Consequently, $X(cc, \Delta t)$ denotes the signal value of X in a specific clock cycle cc at a specific time offset Δt .

The rows of the value array can now easily be represented with sets. The sets are already indicated in table 5-1 by the shaded and unshaded regions. In the following examples this kind of illustration is retained for easier comprehension.

5.2.2 The Simulation Algorithm

Based on the signal representation of the last section, the simulation algorithm can now be described. The algorithm computes for each gate the complete output waveform before it proceeds to the next gate. The waveforms are represented as schedules. The schedules that describe the input and output waveforms of a gate are called the *gate's input* and *output schedules*, respectively. The output schedule of each gate is computed by simultaneously scanning the gate's input schedules. The input events are processed in ascending order of their time offsets Δt . Complete processing of an input event requires four operations:

1. Perform the logic operation of the gate.
2. Delay determination.
3. Glitch detection and removal.

4. Count transitions and possibly distinguish hazards from useful transitions.

These operations are described in more detail in the following sections. Each step is illustrated by the following example. Consider the 2-input AND gate G of figure 5-4. Its input schedules are depicted on the left of the same figure. The according input waveforms are graphically illustrated in figure 5-5.

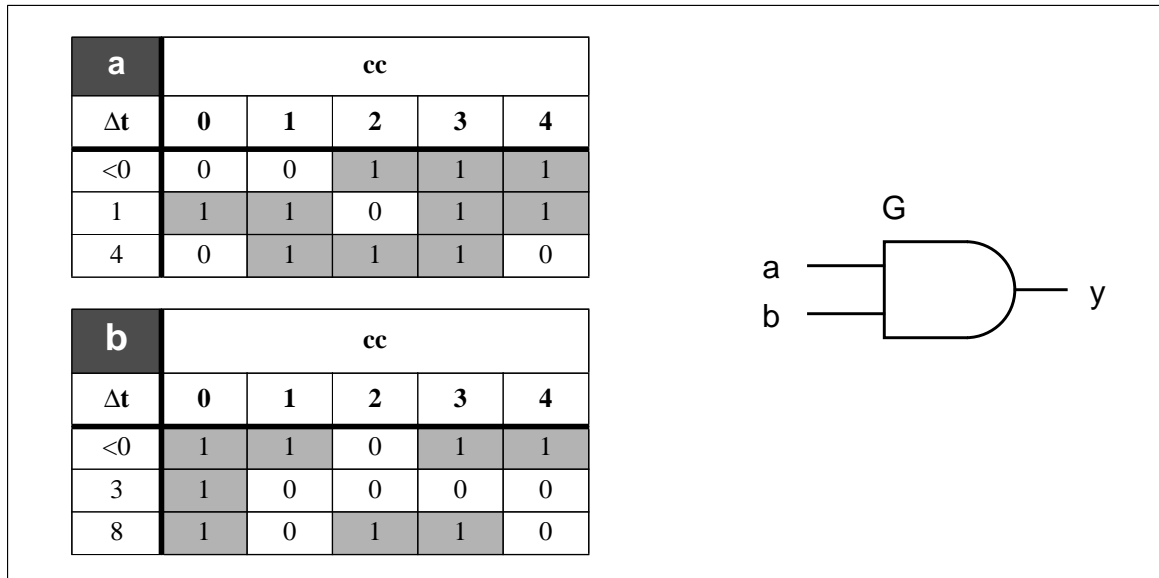


Figure 5-4: AND gate with input schedules

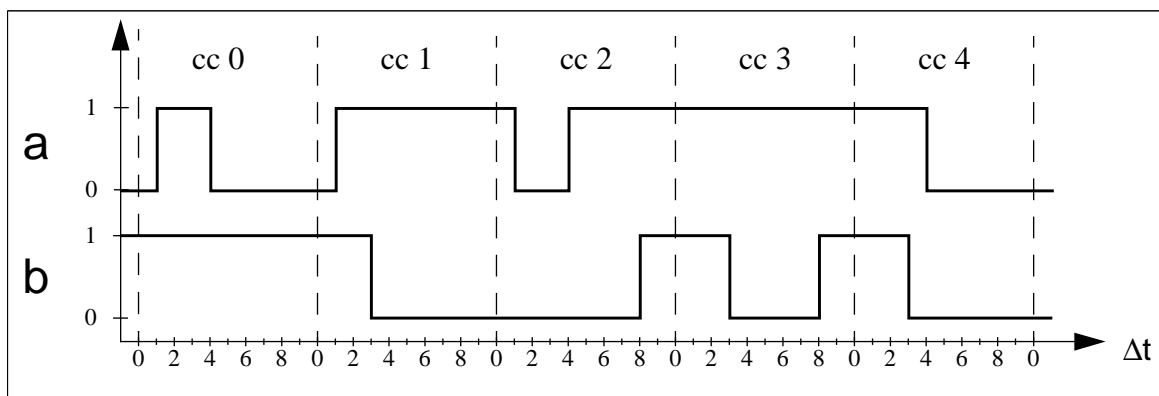


Figure 5-5: The input waveforms of figure 5-4

Before the details of the simulation algorithm can be explained, we need to define the following three terms.

Definition 5-2: Current Simulation Time Δt

The *current simulation time* Δt denotes the time offset of the input event that is currently being processed.

The current simulation time has a similar meaning as in traditional, event based logic simulators. The major difference is that it does not refer to one single point in time but it rather denotes a time offset that refers to all clock cycles simultaneously.

Definition 5-3: Transition $tr_y(\Delta t_i)$

The *transition* $tr_y(\Delta t_i)$ is a binary vector of size n where n is the number of clock cycles that are simulated and

$$tr_y(cc, \Delta t_i) = \begin{cases} 1 & \text{if signal } x \text{ changes its value at CF-time } (cc, \Delta t_i) \\ 0 & \text{otherwise} \end{cases}$$

$tr_y(\Delta t_i)$ is sometimes also referred to as tr_{y_i} or simply as tr_i if the reference variable y is obvious.

Definition 5-4: Falling and Rising Transition

The *falling transition* $tr_y^f(\Delta t_i)$ is a binary vector of size n . n denotes the number of clock cycles that are simulated. $tr_y^f(cc, \Delta t_i) = 1$ if the signal y has a 1 to 0 transition at CF-time $(cc, \Delta t_i)$, otherwise it is 0. The *rising transition* $tr_y^r(\Delta t_i)$ is accordingly defined for 0 to 1 transitions.

Sometimes the notion $tr_y^x(\Delta t_i)$ is used which denotes either a falling or a rising transition.

All vectors in definitions 5-3 and 5-4 are represented by sets.

Logic Operation

Firstly, the output schedule needs to be initialized by performing the logic operation of the gate on the initial values of its input schedules ($\Delta t < 0$). Since these values represent the steady states that have been reached in the previous clock cycles, the values are immediately available at the beginning of the current cycles and need not to be delayed.

Then Δt proceeds to the first event in the input schedules. Assume that this event occurs at Δt_I on input x . All other inputs are supposed to remain stable. The new output value of the gate $y(\Delta t_x)$ is computed and entered into the output schedule by applying the changed value of $x(\Delta t_I)$. The actual value of the time offset Δt_x of the new entry in the output schedule remains open for the time being. Concurrent events on several inputs can be processed accordingly.

In the example of figure 5-4 the initial value of the output y is given by

$$y_{<0} = a_{<0} \cdot b_{<0} = 00111 \cdot 11011 = 00011. \quad (5-2)$$

The first input event occurs at $\Delta t = I$ on input a . Thus we obtain

$$y_{\Delta t_x} = a_1 \cdot b_{<0} = 11011 \cdot 11011 = 11011 \quad (5-3)$$

and the preliminary schedule of table 5-2.

y	cc				
	0	1	2	3	4
Δt	0	0	0	1	1
Δt_x	1	1	0	1	1

Table 5-2: Preliminary schedule of y after two logic operations

Delay Determination

The simulation time Δt_x when the new entry is to be scheduled has been left open in the last section. Since the delay model allows different delays for rising and falling transitions t_{up} and t_{down} , respectively, the two different transition types need to be detected using the following three equations:

$$\text{tr}_y(\Delta t_x) = y(\Delta t_x) \oplus y(\Delta t_{i-1}) \quad (5-4)$$

$$\text{tr}_y^f(\Delta t_i + t_{down}) = \text{tr}_y(\Delta t_x) \cdot y(\Delta t_{i-1}) \quad (5-5)$$

$$\text{tr}_y^r(\Delta t_i + t_{up}) = \text{tr}_y(\Delta t_x) \cdot y(\Delta t_x), \quad (5-6)$$

where $y(\Delta t_{i-1})$ is the entry in the output schedule which directly precedes $y(\Delta t_x)$.

While equation 5-4 detects transitions of any type, equations 5-5 and 5-6 filter falling and rising edges, respectively. Now the correct delays can be applied. The falling and rising transition vectors tr_y^f and tr_y^r are temporarily stored in a second schedule, also called *transition schedule*. It is used during processing of succeeding events. The transition schedule can be removed after the complete output schedule of a gate has been computed, hence it doesn't require much additional memory.

If it is the first input event that is being processed, $y(\Delta t_x)$ can be immediately replaced by the final output values $y(\Delta t + t_{down})$ and $y(\Delta t + t_{up})$. Under the assumption that $t_{down} < t_{up}$, they can be computed by

$$y(\Delta t_i + t_{down}) = \text{tr}_y^f(\Delta t_i + t_{down}) \oplus y(\Delta t_{i-1}) \quad (5-7)$$

$$y(\Delta t_i + t_{up}) = y(\Delta t_x). \quad (5-8)$$

If $t_{down} > t_{up}$, t_{down} and t_{up} as well as the superscripts 'r' and 'f' must be exchanged in equations 5-7 and 5-8.

If equations 5-4 to 5-8 are applied to the given example, the following intermediate results will be obtained:

$$\text{tr}_y(\Delta t_x) = y(\Delta t_x) \oplus y(\Delta t_{<0}) = 11011 \oplus 00011 = 11000 \quad (5-9)$$

$$\text{tr}_y^f(1 + t_{down}) = \text{tr}_y(\Delta t_x) \cdot y(\Delta t_{<0}) = 11000 \cdot 00011 = 00000 \quad (5-10)$$

$$tr_y^r(1 + t_{up}) = tr_y(\Delta t_x) \cdot y(\Delta t_x) = 11000 \cdot 11011 = 11000. \quad (5-11)$$

Thus, we have only two rising events. Since we are processing the first event, we can immediately compute the output values:

$$y(1 + t_{up}) = y(\Delta t_x) = 11011, \quad (5-12)$$

which results in the schedule of table 5-3.

y	cc				
Δt	0	1	2	3	4
<0	0	0	0	1	1
$1+t_{up}$	1	1	0	1	1

tr_y	cc				
Δt	0	1	2	3	4
<0	---				
$1+t_{up}$	1	1	0	0	0

Table 5-3: Final schedule of y after two logic operations

Glitch Removal

Glitch detection and removal can only be omitted during the processing of the first input event. For all succeeding events it is indispensable. According to chapter 2.2.2, any output pulse that is shorter than the gate delay is a glitch. Since the contribution of glitches to the overall power consumption is small [Malu 98], they will be neglected¹. Hence, pulses shorter than the gate delay must be filtered out. In the transition schedule, a glitch can be identified by two criteria:

1. Two transition vectors are separated by less than the delay of the later transition and
2. the two transition vectors have a '1' in the same column.

Thus, as a new entry tr_i is to be entered into the output schedule, condition 2 must be checked for any previous entry tr_j with a scheduled offset Δt_j greater than the current simulation time Δt ($\Delta t_j > \Delta t$). Checking and filtering can be performed simultaneously with the following two equations:

$$tr_j' = (tr_j \oplus tr_i) \cdot tr_j \quad (5-13)$$

$$tr_i' = (tr_j \oplus tr_i) \cdot tr_i \quad (5-14)$$

The old values of tr_i and tr_j are then replaced by their filtered versions:

$$tr_i := tr_i' \text{ and } tr_j := tr_j', \quad (5-15)$$

y_j will only be recalculated when Δt becomes greater than Δt_j .

In the sequel these results shall be applied to our example. Table 5-4 shows the preliminary output schedule during processing of the second input event which occurs on input b at

1. However, it would be perfectly possible to incorporate the models of [Rabe 98] into the simulator.

$\Delta t=3$. Delay determination has already taken place. There exist three falling transitions in clock cycles 1, 3, and 4 which are entered into the last row of the transition schedule.

y	cc				
Δt	0	1	2	3	4
<0	0	0	0	1	1
$1+t_{up}$	1	1	0	1	1
$3+t_{down}$	1	0	0	0	0

tr_y	cc				
Δt	0	1	2	3	4
<0	---				
$1+t_{up}$	1	1	0	0	0
$3+t_{down}$	0	1	0	1	1

Table 5-4: Preliminary schedule of y after three logic operations

Let us assume that $1+t_{up} > 3$. Otherwise no glitch detection would be required since there would be no entry j in the transition schedule with $t_j > \Delta t$. Application of equations 5-13 and 5-14 yields:

$$tr'_{3+t_{down}} = (tr_{3+t_{down}} \oplus tr_{1+t_{up}}) \cdot tr_{3+t_{down}} = 10011 \cdot 01011 = 00011 \quad (5-16)$$

$$tr'_{1+t_{up}} = (tr_{3+t_{down}} \oplus tr_{1+t_{up}}) \cdot tr_{1+t_{up}} = 10011 \cdot 11000 = 10000. \quad (5-17)$$

The resulting schedule is depicted in table 5-5. Obviously, the pulse at $y(1, 1+t_{up})$ has been deleted. Note that $y(1+t_{up})$ has not been updated yet with the new value of $tr(1+t_{up})$.

y	cc				
Δt	0	1	2	3	4
<0	0	0	0	1	1
$1+t_{up}$	1	1	0	1	1
$3+t_{down}$	1	0	0	0	0

tr_y	cc				
Δt	0	1	2	3	4
<0	---				
$1+t_{up}$	1	0	0	0	0
$3+t_{down}$	0	0	0	1	1

Table 5-5: Final schedule of y after three logic operations

Transition and Hazard Counting

As the simulation proceeds and the current simulation time Δt increases, entries $tr_y(\Delta t_i)$ in the transition schedule become dispensable as Δt becomes greater than their scheduled time Δt_i . Before they can be removed from the memory, the total toggle number of the gate tg_y must be updated using the following formula:

$$tg_y := tg_y + \sum_{\text{for all sets } j \text{ of } tr_y(\Delta t_i)} e_j - s_j + 1 \quad (5-18)$$

where e_j and s_j denote the indexes of the end and the start of set j , respectively. At the same time the value array has to be updated by

$$y(\Delta t_i) = tr_y(\Delta t_i) \oplus y(\Delta t_{i-1}). \quad (5-19)$$

After all input events have been processed and the total number of toggles of the gate have been computed, the hazard rate of this gate h_y can be determined. Before, the useful transitions u_y need to be calculated by

$$u_y = y_{<0} \oplus y_n, \quad (5-20)$$

where y_n is the last entry in the output schedule. The number of hazards results then in

$$h_y = tg_y - tg_u. \quad (5-21)$$

The number of useful transitions tg_u can be determined by applying equation 5-18 on u_y .

In the example, we find $tg_y=6$. u_y results in

$$u_y = y_{<0} \oplus y_{11} = 00011 \oplus 00110 = 00101. \quad (5-22)$$

Thus, we have 2 useful transitions and 4 hazards.

Table 5-6 depicts the final output schedule after all input events have been processed. The gate delays were assumed to $t_{up}=3$ and $t_{down}=1$. Note that the events at $1+t_{up}$ and $3+t_{down}$ were joined to one single event since $1+t_{up} = 3+t_{down} = 4$.

y	cc				
	0	1	2	3	4
<0	0	0	0	1	1
4	1	0	0	0	0
5	0	0	0	0	0
11	0	0	1	1	0

Table 5-6: Final output schedule after processing of all input events

5.2.3 Results

The algorithm that was presented in the last section has been integrated into *ASSeT*. It was tested on several circuits from the ISCAS-85 [Brgl 85] and ISCAS-89 [Brgl 89] benchmark sets. The circuits were mapped on the GFN 120 gate forest cell library [IMS 91]. For each experiment, 25 000 random test patterns were generated using the same parameters as in chapter 4: the static probabilities p_i and the switching probabilities α_i of the PIs are 0.5. The sequential circuits were simulated with 20 000 *RTL-generated patterns*¹. The block optimization algorithms were omitted since they don't improve the simulation times for sequential or large combinational circuits. The runtimes of the set based simulator were compared to logic simulation. *ASSeT* in a logic simulation mode was used for the latter in order to guarantee a fair comparison. This ensures that exactly the same delay models are applied and avoids the influence of implementation specific details that can largely effect the runtimes, like programming languages, different compilers, and compiler options. The speedup of the

1. For details about pattern generation see chapter 4.5.5, section *Sequential Circuits*.

set based simulator over logic simulation and the commercial logic simulator of Synopsys [SyPo 96] are depicted in figure 5-6, labeled by *Logic* and *Synopsys*, respectively.

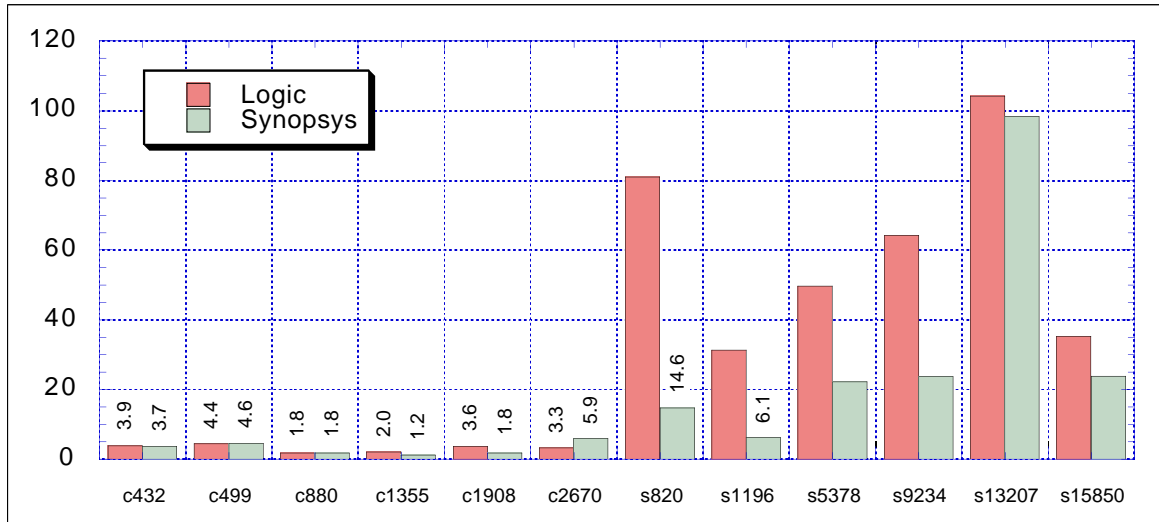


Figure 5-6: Speedup of set based over logic simulation

As in the zero delay simulations that were presented in the last chapter, the set oriented approach is better suited for sequential circuits. With average speedups of 3.2 and 61 for combinational and sequential circuits, respectively, the absolute acceleration is higher than for ZDM simulation, though. Obviously, the vectors in the schedule tend to be sparse, resulting in a more efficient set representation than in a zero delay simulation.

The speedups over Synopsys are slightly smaller, depending on the circuit. However, the two simulators are hardly comparable since they rely on completely different simulation methods. Synopsys requires the simulation to be preceded by an analysis step in which the original netlist is translated into native machine code of the machine, the simulator is running on. Figure 5-6 is based on pure simulation times which don't include the analysis step. Furthermore, the simulator version that was available for these experiments offers some more functionality, but did not count signal toggles on the other hand.

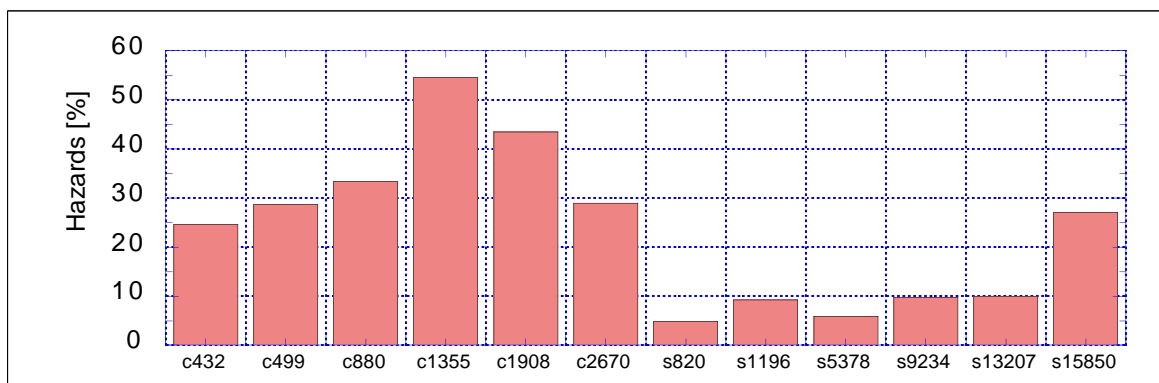


Figure 5-7: Hazard rate

Figure 5-7 shows the hazard rates which illustrate the error that is being made if gate delays are neglected. On the average, 23% of all transitions are hazards. Thus, gate delay modeling is indispensable for accurate power estimation.

For the experiments of figure 5-6 and figure 5-7, the load dependence of the gate delays had to be neglected since this dependence was not included in the available Synopsys library. Figure 5-8 depicts the speedup for *ASSeT* with load dependent delays. Obviously, there are slight differences but the general trend is comparable to figure 5-6.

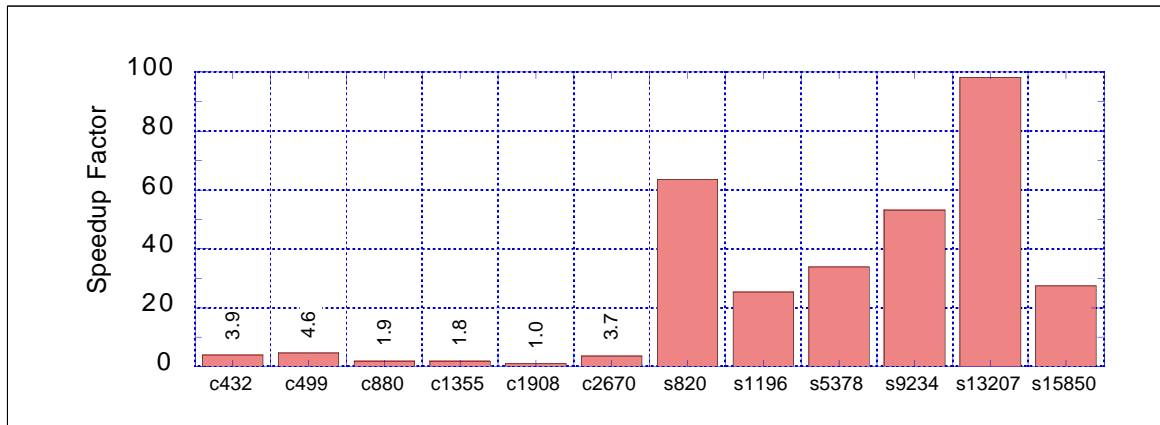


Figure 5-8: Speedup with load dependence

5.3 Application to a Bitparallel Simulator

The bitparallel approach from [Schn 95] was presented in chapter 4.2.1. Bitparallel simulation has been known since the mid 1980s in the test area [West 93], but it has been limited to the ZDM so far. The bitparallel and the set based approach have a common property: they both simulate several input patterns at once. While the bitparallel approach simulates packets of a fixed size, usually the machine word length, the packet sizes depend on the input patterns in the set approach. Nevertheless, the algorithm of section 5.2 can be directly applied to the bitparallel approach without major changes. Definitions 5-1 to 5-4 as well as equations 5-2 to 5-17 do not base on a specific signal representation model. Bitparallel signal representation can just as well be used as set representation. This emphasizes the general applicability of this approach to a whole class of simulation methods. Only during transition and hazard counting, the bitparallel approach enables some more optimizations that are discussed in the next section.

5.3.1 Transition and Hazard Counting

Schneider et al. propose a LUT based approach for transition determination. The LUT assigns n -bit words to the number of transitions in the word. Since the size of the LUT is exponential in the word width n , they partition the processor words into blocks of 8 bits. A transition between the last bit in the last 8-bit word and the first bit of the current 8-bit word cannot be recognized if the 8-bit words are considered individually. Thus, the last bit of the

last 8-bit word is appended to the current 8-bit word, resulting in $n=9$. 4 LUT accesses are then required in order to evaluate one 32-bit machine word.

In this work, a slightly more efficient approach was possible since the transition vectors tr_y are available. The machine words in the transition vectors are partitioned into 16-bit words which are evaluated with a LUT. But instead of mapping 16-bit words to transition numbers, the LUT maps 16-bit words to their hamming weight since each '1' in the transition vector corresponds to a signal transition. The higher efficiency results from only two LUT accesses per 32-bit word, from omitting the append-operation, and from the fact that most processors offer special machine instructions to access half words from machine registers [Pram 94].

5.3.2 Results

Similar experiments as in section 5.2.3 were carried out in order to test the bitparallel approach. However, the sequential circuits were also simulated with purely random patterns, so they can be considered here as large combinational circuits.

Figure 5-9 shows the speedup of the bitparallel simulator running on a 32-bit machine, compared to the same simulator in a bitwise mode¹. On the average 5.3 times speedup was obtained. This value is 65% times higher than for the set approach, but it is still far below the theoretical value of 32. Several methods to improve this value are addressed in the next section.

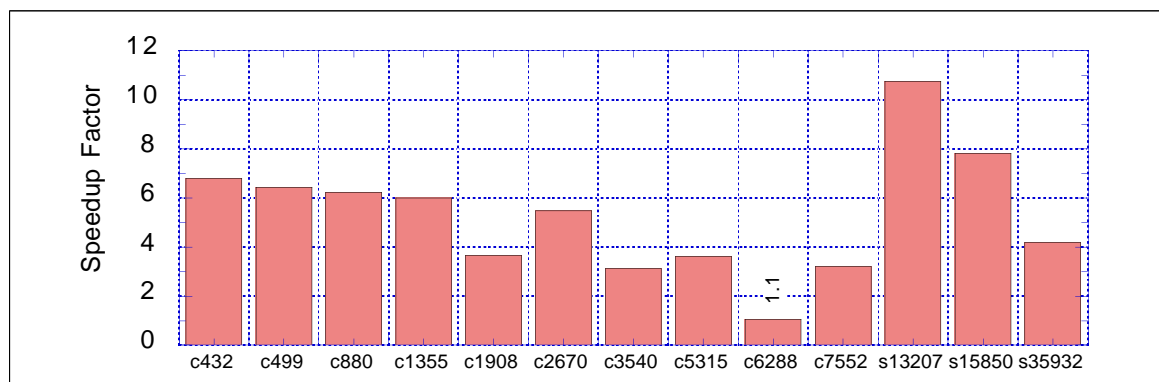


Figure 5-9: Speedup of bitparallel over bitwise simulation

5.4 Optimizations

The algorithm of the last section still offers some optimization potentials. Firstly, there is a certain overhead if only single processor words are propagated through the circuit. Memory needs to be dynamically allocated and deallocated. Further, some intermediate results are to be computed and stored for each gate output, e.g. the number of toggles that have occurred so far on a specific signal. Secondly, for some circuits the schedules tend to have an

1. The logic simulations of section 5.2.3 were performed using the same mode.

extremely high number of sparse vectors. While the speedup of the set approach is based on this sparseness, it increases the runtime and reduce the memory efficiency of the bitparallel simulator.

5.4.1 Dynamic Package Sizing

The simulation overhead can be reduced if packages of several words are propagated together, instead of propagating single words through the circuit. The overhead would be minimized if all the test patterns were processed in one single run. On the other hand, increased package sizes lead to increased schedule lengths and memory requirements. In the worst case, the available main memory mem_{max} can be exceeded, causing the operating system to use swapping space on the hard disk which in turn decreases the performance for several orders of magnitude. Hence, swapping must be absolutely avoided.

The optimum package size depends on the circuit structure, the cell library, as well as on the test vectors. Therefore, an algorithm has been developed for dynamically adjusting the optimum package size during the simulation process. This algorithm is called *dynamic package sizing (DPS)*. DPS consists of two phases. During the first phase it determines the optimum package size. This size is then used during the second phase for simulating the rest of the input patterns without any further checks.

Phase one is initialized with the minimum package size of one machine word, usually 32 or 64 bit. After each simulation run, the package size n is doubled. For each run the average CPU time per input pattern t_n^{pat} as well as the memory requirements mem_n are recorded. The latter is used to estimate the memory requirements of the next run with double package size $2n$, \hat{mem}_{2n} . Phase one ends under two conditions:

1. $t_n^{pat} > t_{n/2}^{pat}$: the optimum package size is $n/2$.
2. $\hat{mem}_{2n} > mem_{max}$: the optimum package size is n .

Results

The experimental results of DPS are depicted in the following two figures. Figure 5-10 shows the overall speedup of bitparallel simulation with DPS over bitwise simulation. Figure 5-11 focuses on the effect of DPS alone by illustrating the speedup of bitparallel simulation with DPS over bitparallel simulation without DPS.

The average speedup of bitparallel simulation with DPS is 23. Thus, the value of the pure bitparallel approach could be improved by factor 4.3 without any accuracy loss. As figure 5-11 indicates, some circuits profit considerably more from DPS than others. If the speedups of figure 5-11 are compared to those of figure 5-6, it becomes obvious that the higher the speedup of the pure bitparallel approach the higher is the profit from DPS as well.

The question, why some circuits perform worse than others and how the speedup can be improved for those circuits, is investigated in the next section.

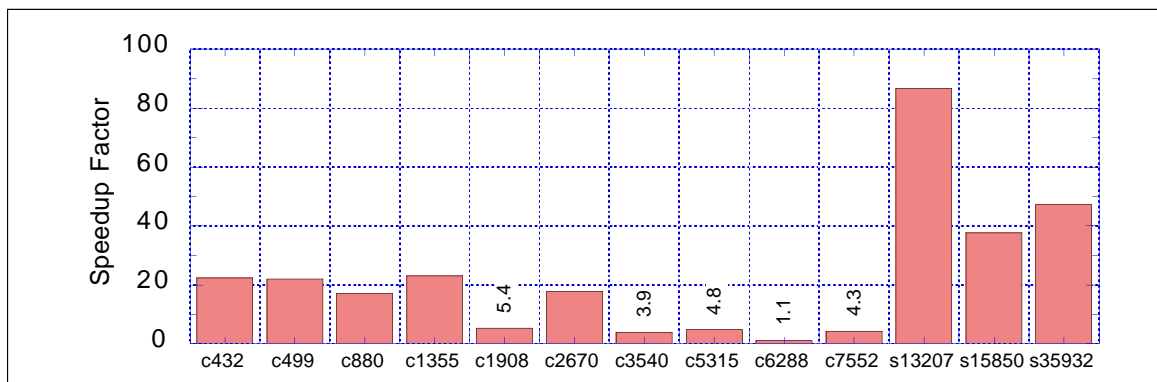


Figure 5-10: Speedup of bitparallel simulation with DPS over bitwise simulation

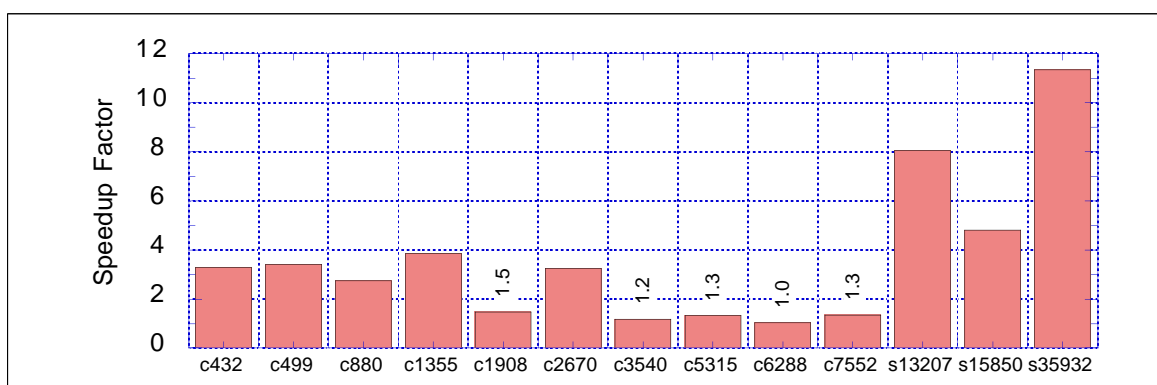


Figure 5-11: Speedup of bitparallel simulation with DPS over bitparallel simulation without DPS

5.4.2 Schedule Compression

Figure 5-12 depicts the package sizes that were chosen by DPS in the previous experiments. Obviously, the circuits for which DPS chooses large packages, result in a higher speedup. The reason is as follows: large package sizes can only reduce the simulation overhead if the schedules are dense. This happens if the output events of a specific gate concentrate on only few time offsets Δt_i . On the other hand, if the offsets of all events of a gate are different, each event requires a separate vector of full length in the schedule. In that case, as the package size is increased, the schedules become longer, resulting in longer simulation times since the degree of parallelism is decreased.

For this circuit type, further speedup is possible if some accuracy loss is acceptable using a technique called *schedule compression*. The basic idea behind schedule compression is to join events i with similar event times Δt_i by assigning the event times to equivalence classes $\Delta t'_i$

$$\Delta t'_i = \left[\left(\Delta t_i + \frac{\Theta}{2} \right) \text{div } \Theta \right] \cdot \Theta. \quad (5-23)$$

Equation 5-23 corresponds to rounding the event times Δt_i to multiples of Θ . Since glitch filtering heavily depends on the values of the gate delays and gate delays vary by more than

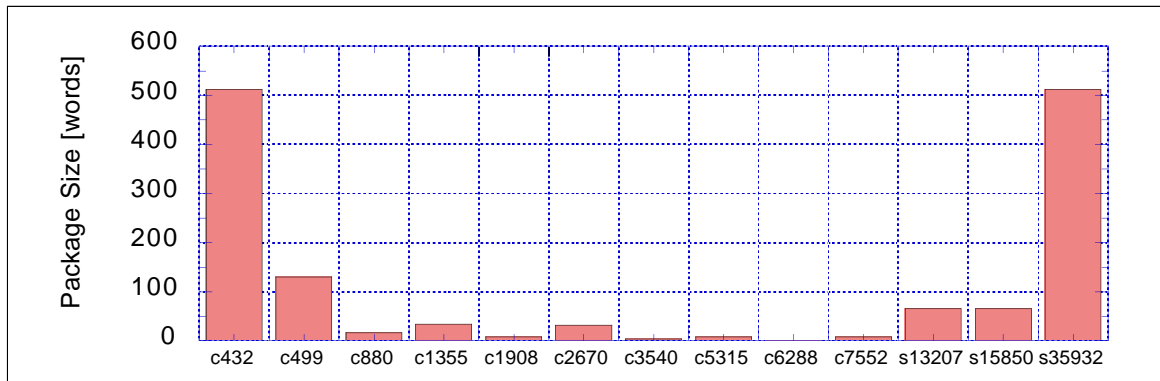


Figure 5-12: Packages sizes for DPS

one order of magnitude ([IMS 91] and [IMS 96]), Θ cannot be chosen as a static value. Instead, Θ is determined individually for each gate as a fraction δ of the gate delay. δ is also called the *compression rate* in the sequel.

Results

Figures 5-13 and 5-14 summarize the speedups that were obtained with schedule compression. 5%, 20%, and 50% denote the results for different compression rates δ , while in figure 5-13 0% repeats the result of bitparallel simulation with DPS only, for comparison. The average speedups over bitwise simulation result in 32, 69, and 131 for $\delta=5\%$, 20%, and 50%, respectively.

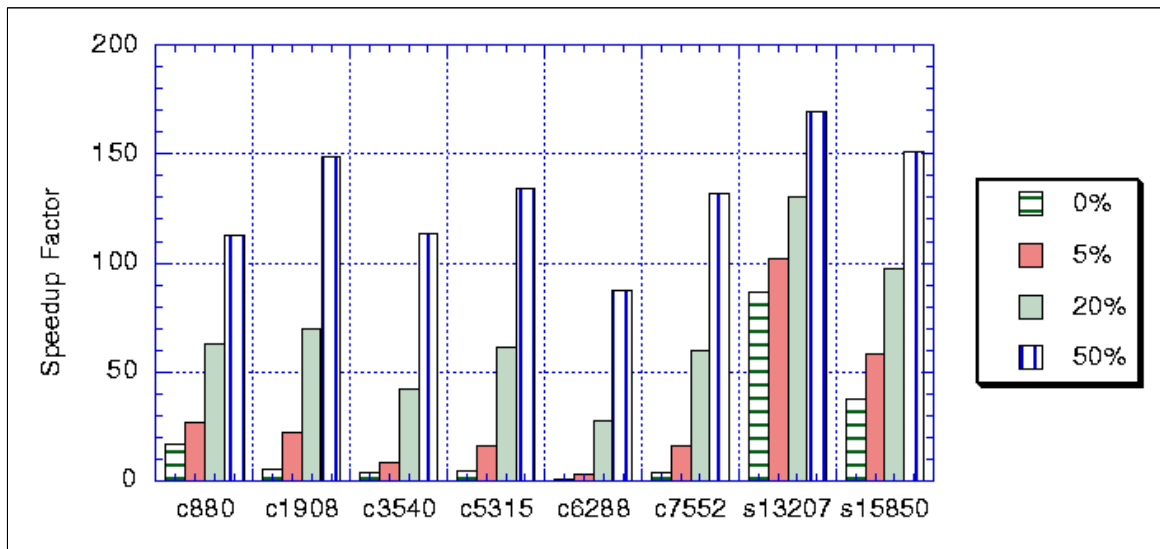


Figure 5-13: Speedup over bitwise simulation

Obviously, the circuits that got least effected by DPS profit most from schedule compression, so that the speedup figures become more and more homogeneous as the compression rate δ increases. Hence, the assumption that the poor speedups of bitparallel simulation with and without DPS is caused by long, sparse schedules could be justified and schedule compression offers a means to improve the simulation times for such circuits.

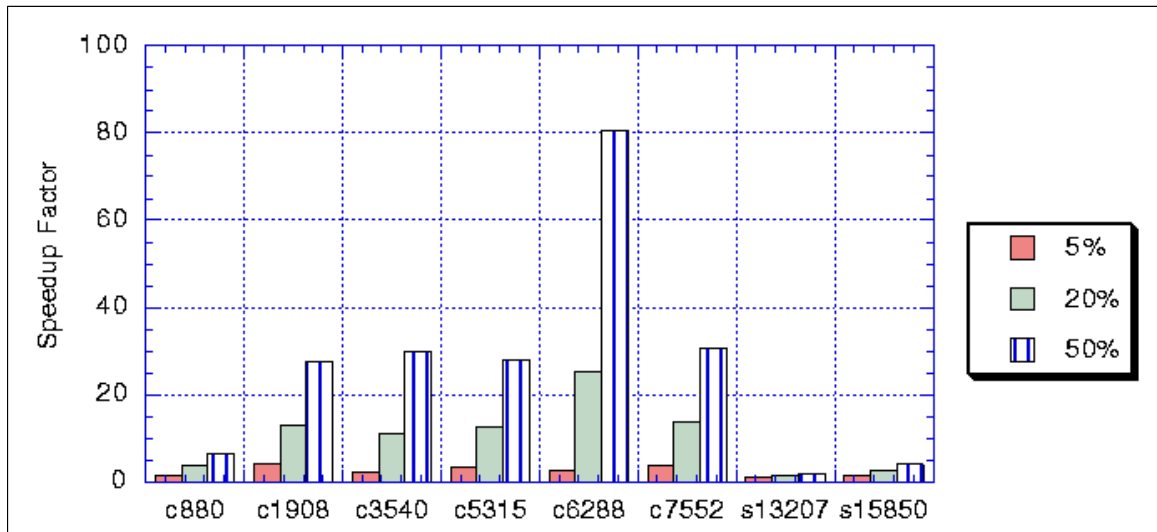


Figure 5-14: Speedup of schedule compression and DPS over DPS only

On the other hand, schedule compression introduces a certain estimation error because the event times are modified. The global and local errors (according to equation 4-34) are summarized in figures 5-15 and 5-16, respectively. The local errors are about two to three times higher than the global error and, as expected, the errors increase with the compression rate. But even the local error is below 4% for as high a compression rate as 50%. The errors that are introduced by other sources, e.g. the power models of the gates, are usually much higher. Hence, the error that is caused by schedule compression is negligible in most applications. The global errors for *c1908* and *s15850* are unexpected since they slightly decrease

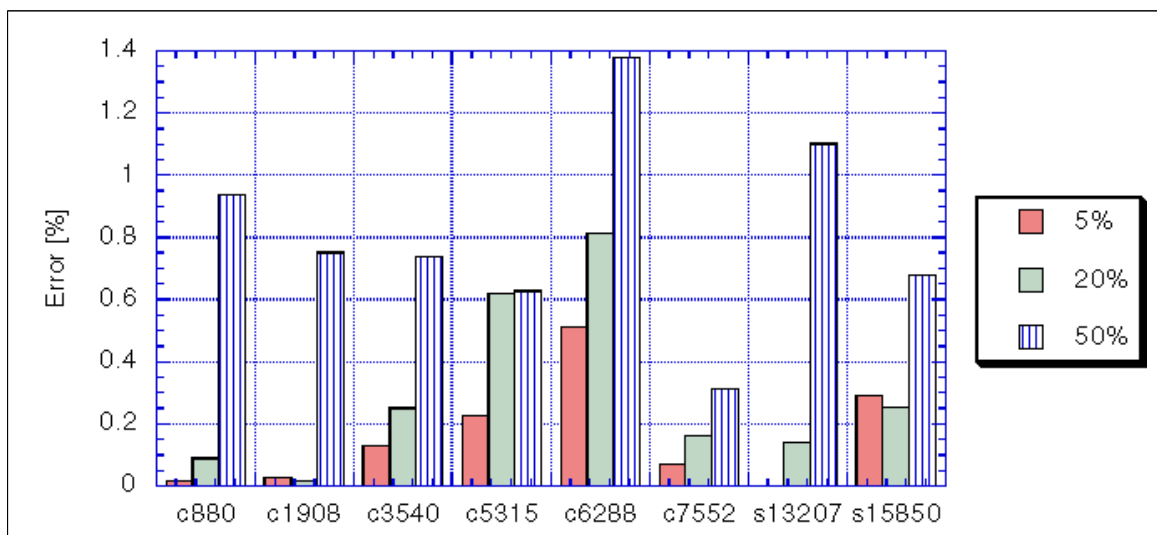


Figure 5-15: Global error

from compression rate 5% to 20%. Again, the cause may again be found in the *random error compensation effect* that was described in chapter 4.6.4.

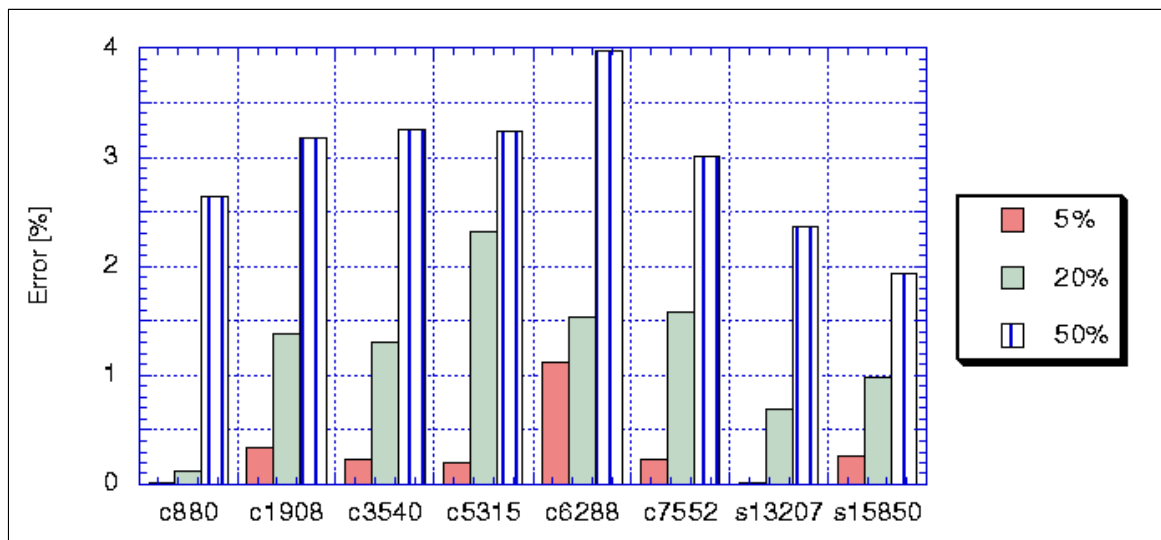


Figure 5-16: Local error

5.5 Summary

In this chapter, the set based simulation approach was extended to real delay models by combining it with a novel event based method. While in ordinary logic simulators the signal waveforms are represented as vectors, this representation was extended to a two dimensional array in order to take full advantage of the set based signal representation. The simulation algorithm allows different delays for falling and rising edges, and glitch detection. The latter are simply neglected like in most of today's commercial logic simulators [SyPo 96]. Additionally, a combination with more sophisticated glitch models like those of [Rade 96], would be possible. The resulting simulation algorithm was tested on several benchmarks from the ISCAS benchmark sets. The simulation times were compared to those of a pure logic simulation with the same simulator and to those of the commercial simulator from Synopsys. As in the ZDM approach, the speedups for combinational and sequential circuits differ remarkably. For the former, an average speedup of 3.2 was reached, while for the latter a value of 61 could be obtained. Hence, the tendency from chapter 4 that set simulation is best suited for sequential circuits, is even increased, if gate delays are taken into account. Comparisons with the commercial simulator, revealed slightly lower but comparable speedups.

The same real delay simulation concept could also be applied to the bitparallel simulator *PAPSAS* that was first presented in [Schn 95] and which was restricted to the ZDM so far. Experiments on several benchmark circuits revealed an average speedup of 5.3 over bitwise simulation. This value could be importantly improved by two optimization techniques: *dynamic package sizing (DPS)* and *schedule compression*. While DPS dramatically improves the simulation times for some circuits, its success is only modest for others. The simulation of the latter can be importantly accelerated by *schedule compression*. By combining both methods, an almost homogeneous speedup of up to 131 times over bitwise simulation could be obtained. However, schedule compression yields only an approximate

result since it is based on a reduced precision in the time scale. But even for the highest compression rate that was tested, the global and local errors were below 2% and 4%, respectively. These error rates are negligible in most applications.

On the other hand, it was found that the average hazard rates during all simulation experiments were between 25% and 50% of all transitions, depending on the circuit types and delay models. For one circuit (*c6288*), as many as 90% hazards were found. Since hazards can only be taken into account in a real delay simulation, gate delay modeling is mandatory for accurate power estimation. However, high precision of the delay models is less crucial for the estimation result as it could be shown by the experiments with schedule compression.

Pattern simulation based power estimation reaches its limits if high accuracy demands require the simulation of a huge amount of test patterns or if pattern generation is not feasible because of complex spatial correlations between the primary input signals. Then probabilistic methods offer a viable alternative. Just like pattern based approaches, probabilistic methods rely on a propagation technique. But instead of propagating complete signal waveforms in the form of long binary vectors, probabilistic methods propagate statistical informations. In the ideal case this information consists of only two values per PI: the static and the switching probability. Thus, probabilistic approaches exactly solve the estimation problem in one single “propagation run”. However, the problems of most of today’s probabilistic approaches are twofold: they are either fast and inaccurate or accurate and slow. The reason for both are either spatial or temporal signal correlations and concurrent switching of the primary inputs.

In this chapter two methods are presented which address these problems. The problem of correlation handling is solved by combining BDD based probabilistic methods with logic or set based simulation. Through the novel *correlation layer* technique, it becomes possible to correctly model the correlation groups, proposed in chapter 4. Runtime behavior and memory requirements are importantly improved by the concept of *local BDDs*. Detailed experimental results reveal that neglecting spatial correlations between the PIs can highly deteriorate the accuracy of the estimation result. Thus, the correlation layer technique can dramatically increase the accuracy. On the other hand, the concept of *local BDDs* reduces the memory requirements importantly while its influence on the estimation result is below 5% on the average. Hence, the local BDDs reduce the dependence on good variable orders for the BDDs, a problem commonly confronted with in any BDD application.

6.1 Analysis of the State-of-the-Art

Most of today’s developments in the area of probabilistic power estimation on gate and logic level rely basically on Najm’s algorithm [Najm 93]. Examples from the latest literature are [Lim 97] and [Mont 97]. Najm’s algorithm, as well as some extensions to it, were outlined in detail in chapter 4.3.2. The major results shall be briefly repeated here. While Najm’s BDD approach can correctly handle correlations caused by reconvergent fanout, it has still three shortcomings: the runtime behavior and the fact that concurrently switching inputs as well as spatial dependencies between the PIs are not taken into account. In most

publications the authors overcome the runtime problem by circuit partitioning, e.g. [Najm 93], [Chou 94]. However, they never mention the applied partitioning algorithm. The circuits are only said to be partitioned “cautiously”. Thus, it must be assumed that this task is performed manually. To the author’s knowledge, the only dedicated partitioning algorithm for power estimation purposes was published in [Kapo 94]. It is presented in more detail in section 6.3. Here it shall only be mentioned that the results are not satisfactory, especially in terms of accuracy.

The problem of concurrently switching PIs can be addressed by *XOR-BDDs* (chapter 4.3.2), while no extension for Najm’s approach has been published so far that takes into account spatially correlated PIs.

6.2 Probabilistic - Logic Simulation

The approach which is presented in this section, is based on the switching function tr_i of equation 4-2, represented by *XOR-BDDs*. Since the BDD-package [Some 98] that was used for the implementation of the BDD-algorithms, is based on negative edges, the following examples do also apply this technique.

6.2.1 Correlation Layers

It was outlined in chapter 4 that correlation groups can be identified in many practical applications. They can either be found through investigation of the system in which the circuit is embedded, e.g. with m-expanded networks, or through analysis of the input streams that are applied to the circuit, e.g. using the algorithm of listing 4-1.

For probabilistic simulation, the Boolean function y_i and the switching function tr_i of any circuit node i are represented by BDDs. Their variable order is free but it must be fixed before the BDDs are actually built. Usually the variable order is chosen such that the BDD representation results in a minimum number of nodes. The following example presents a novel extension for BDD-based probabilistic simulations that allows to take into account correlation groups if the requirement for minimum size BDDs is relaxed. Consider the BDD of figure 2-4, repeated here for convenience in figure 6-1. Figure 6-1a shows an optimal variable order (a, b, c) . Now assume that variables a and c are correlated but not b . The variables are now reordered such that a and c are adjacent, e.g. (a, c, b) . The resulting BDD is depicted in figure 6-1b¹ where all nodes indexed by the variables a or c appear now in one *layer*. Such a layer is called a *correlation layer* in the sequel.

If the signal probability $P(y)$ is to be calculated, equation 4-20 must be modified because the independence of x and f_x is not given within correlation layers. Instead, y is expanded until the end of the correlation layer according to the following equation:

$$P(y) = P(ac) \cdot P(f_{ac}) + P(\bar{a}c) \cdot P(f_{\bar{a}c}) + P(a\bar{c}) \cdot P(f_{a\bar{c}}) + P(\bar{a}\bar{c}) \cdot P(f_{\bar{a}\bar{c}}). \quad (6-1)$$

1. Of course a better variable order (c, a, b) would be possible and meet the constraints as well.

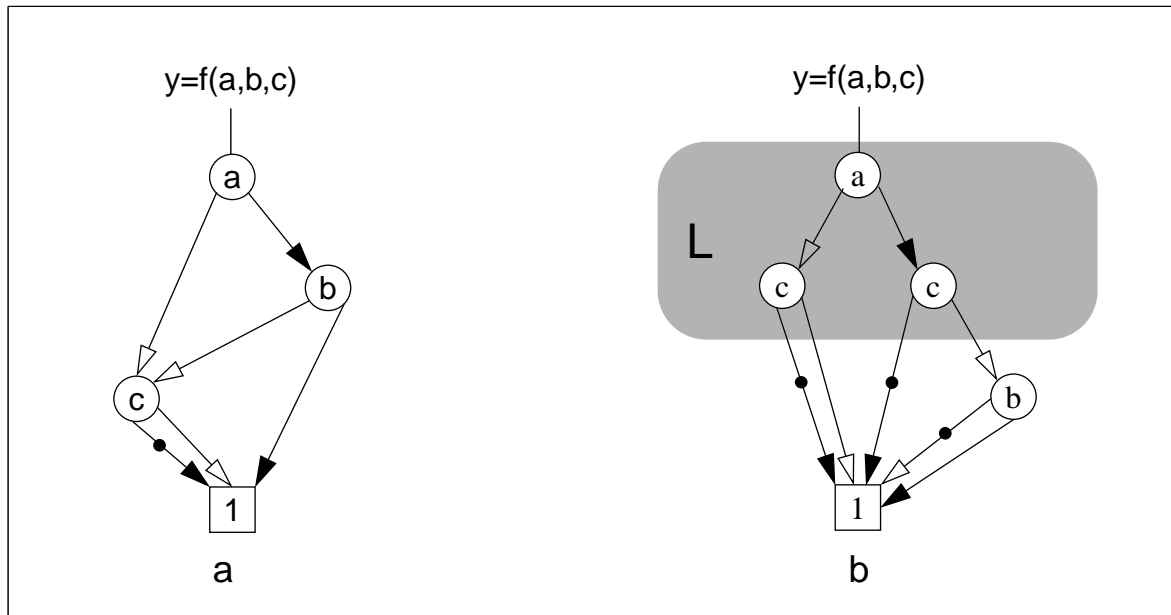


Figure 6-1: Two representations for $y = \bar{c} + ab$

Obviously, the formerly independent terms like $P_a \cdot P_c$ or $P_{\bar{a}} \cdot P_c$ have been replaced by the joint probabilities $P(ab)$, $P(\bar{a}b)$ etc. The joint probabilities are called *path probabilities* since each one refers to exactly one path through the correlation layer. Larger correlation groups can be taken into account if all variables of one group are ordered adjacently, thus building a larger correlation layer. The variables within a correlation layer as well as the correlation layers themselves can be arbitrarily reordered for optimization purposes. Nevertheless, the correlation layers reduce the total number of possible variable orders. Since just the variable order for minimum size BDDs maybe prohibited, some runtime and memory efficiency has to be sacrificed for accuracy.

6.2.2 The Simulation Algorithm

After highlighting the general idea, the simulation algorithm will be explained in more detail. In general, there are two types of node operations: probabilistic evaluation at the limits between correlation layers and evaluation within correlation layers. While the former has been introduced in chapter 4.3, the latter is explained in this section.

Vector Probability

The *vector probability* plays a key role during the evaluation within correlation layers. For easier comprehension, it is introduced by a concrete example. Assume the BDD of figure 6-2 which corresponds to an AND-gate. It has two input variables x_1 and x_2 , and two nodes A and B .

The variables x_1 and x_2 are supposed to belong to the same correlation layer. Therefore, four different input vectors and path probabilities can be defined. They are depicted in table 6-1.

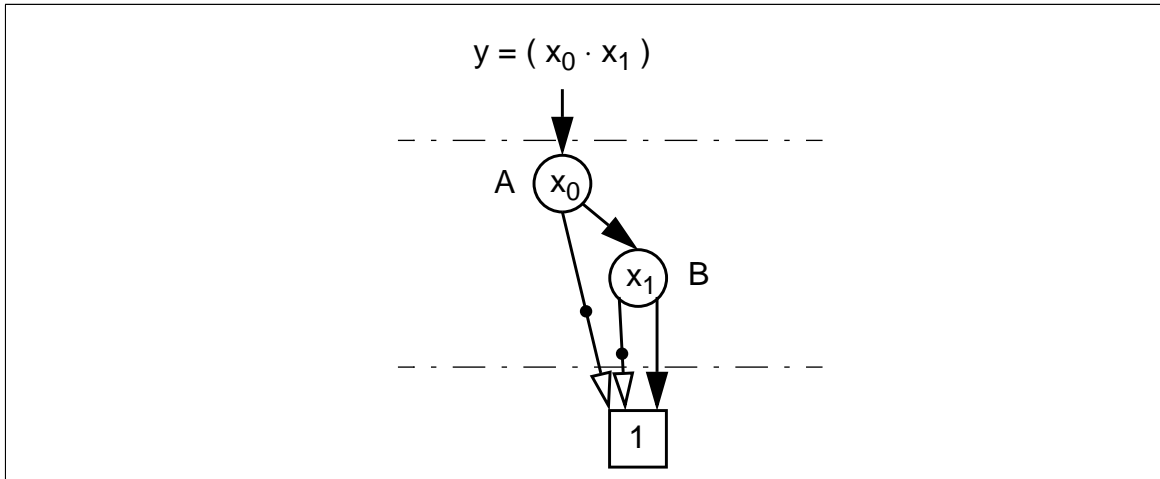


Figure 6-2: BDD of an AND-gate

Vector number	0	1	2	3
Path Probability	p_0	p_1	p_2	p_3
x_0	0	0	1	1
x_1	0	1	0	1

Table 6-1: Path probabilities and possible input vectors of a 2-input circuit

Each of the paths results into '1' with a certain probability. These probabilities are depicted in the *vector probability* (table 6-2).

Vector number	0	1	2	3
Vector probability	0.0	0.0	0.0	1.0

Table 6-2: Vector probability of a 2-input AND-gate

The elements of the vector probability give the probabilities that a subfunction which is selected by a certain input vector, results into '1'. In figure 6-2 there is only one such subfunction. It is the 1-terminal node. If it is reached via a positive edge (path 3), the 1-probability of this path is 1.0, otherwise it is 0.0.

Since the subfunctions always correspond to their root node X , we call them the *subfunction of node X*, $SF(X)$. The *subfunction probability* $P(SFP(X))$ denotes the probability that the subfunction that is defined by node X , results into '1'. In figure 6-2 $SF(X)$ is trivial, it is the 1-terminal node and we get $P(SFP('1')) = 1.0$.

After this illustrative introduction, a more formal definition of the terms will be given. Consider a correlation layer L that consists of n variables. There are $N=2^n$ possible patterns for these variables. The vector $\underline{p}(X)^T = [p_0 \dots p_{N-1}]$ contains the probabilities of all possi-

ble patterns and shall be given. $p(X)$ is a property of the specific correlation layer L . Therefore, it is also denoted as $p(L)$. $p(L)$ is called *path probability vector* or briefly *path probabilities*.

Definition 6-1: L-Path, T-Node

Suppose a correlation layer L and a node $X \in L$. Any path with head X , which contains exactly one node $T \notin L$, is called an *L-Path*. Obviously, the node T can only be the tail of the L-path. It is called *T-node*.

From any node $X \in L$ there exist $m \leq N$ L-paths.

Definition 6-2: Vector Probability

Suppose a correlation layer L and a node $X \in L$ with m L-paths. T_0, \dots, T_{m-1} are the T-nodes of the L-paths. Then the vector probability of X is given by the vector

$$\underline{v}(X) = [\text{SFP}(T_0) \dots \text{SFP}(T_{m-1})].$$

Definition 6-3: Selection Matrix

Suppose a correlation layer L and a node $X \in L$. The *selection matrix* $S(X)=(s_{ij})$ of X is the $m \times N$ matrix defined by:

$$s_{ij} = \begin{cases} 1 & \text{if input pattern } i \text{ selects L-path } j \\ 0 & \text{otherwise} \end{cases}$$

The subfunction probability of any node $X \in L$ is then given by

$$\text{SFP}(X) = \underline{v}(X) \cdot \underline{S}(X) \cdot \underline{p}(L) \quad (6-2)$$

In the example of figure 6-2 the previously defined terms result in

$$\underline{p}(L)^T = [p_0 \ p_1 \ p_2 \ p_3]$$

$$\underline{S}(B) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \underline{S}(A) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{v}(B) = [0.0 \ 1.0] \quad \underline{v}(A) = [0.0 \ 0.0 \ 1.0]$$

and application of equation 6-2 yields $\text{SFP}(B) = p_1 + p_3$ and $\text{SFP}(A) = p_3$.

6.2.3 Probability Propagation

Suppose the BDD of figure 6-3 is given. The part of the BDD that is being considered here, consists of the three nodes A , B , and C . They belong to the same correlation layer L . R denotes the multi-BDD where the nodes B and C fanout to. It is of no further interest.

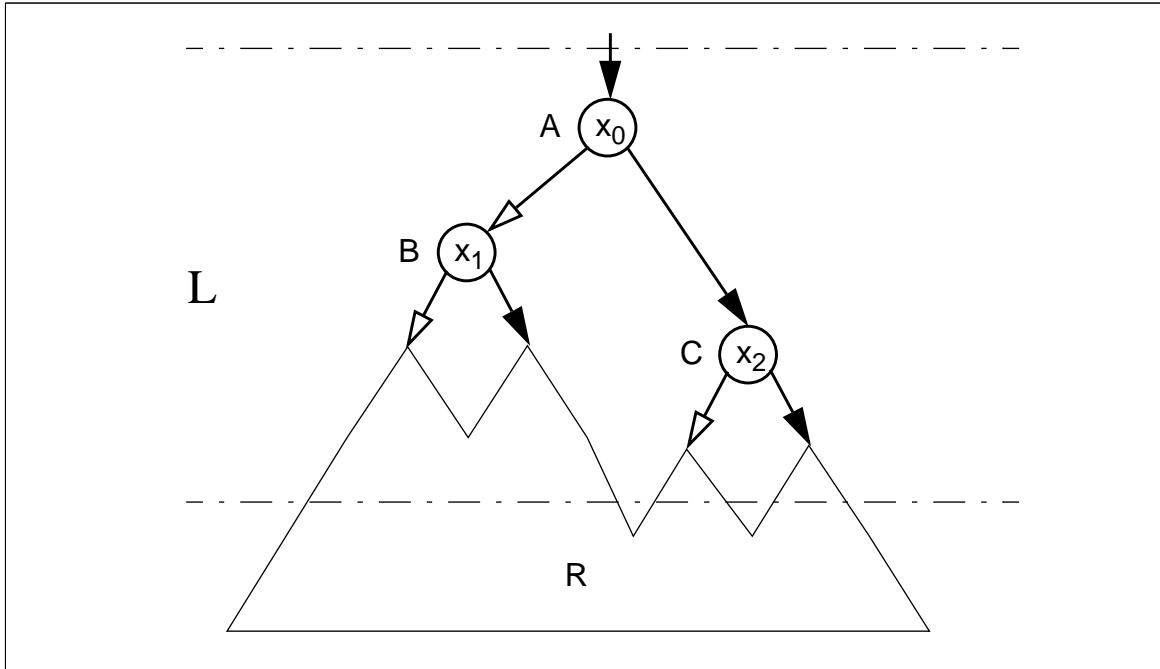


Figure 6-3: Arbitrary BDD

Now suppose that the path probabilities $p(L)$ of the layer L , the vector probabilities $\underline{v}(B)$, $\underline{v}(C)$, and the selection matrices $\underline{S}(B)$ and $\underline{S}(C)$ are given or have been computed before. Thus, all parameters are known for nodes B and C that are required to compute their subfunction probabilities $SFP(B)$ and $SFP(C)$. This section deals with the question, how $\underline{v}(A)$, $\underline{S}(A)$, and $SFP(A)$ can be computed if all parameters of the child nodes of A are known.

According to the evaluation rules for BDDs (definition 2-37) $SF(A)$ is given by

$$SF(A) = \bar{x}_1 \cdot SF(B) \oplus x_1 \cdot SF(C). \quad (6-3)$$

Thus, $x_1=0$ selects $SF(B)$ and $x_1=1$ selects $SF(C)$. Consequently, the selection matrix of A $\underline{S}(A)$ can be composed from $\underline{S}(B)$ and $\underline{S}(C)$ with the following rule:

$$\underline{S}(A) = \begin{bmatrix} \underline{S}(B) \otimes \bar{x}_0 \\ \text{-----} \\ \underline{S}(C) \otimes x_0 \end{bmatrix}, \quad (6-4)$$

where x_0 and \bar{x}_0 correspond to the signal waveforms according to table 6-1. The mask operator \otimes is defined by the following definition.

Definition 6-4: Mask Operator \otimes

Given an $m \times n$ binary matrix \underline{M} with m rows M_i and a binary vector x of size n , the result of the mask operation $\underline{M} \otimes x$ is defined as the $m \times n$ binary matrix \underline{R} where the rows r_i of \underline{R} are given by

$$r_i = M_i \cdot x.$$

Note that ‘ \cdot ’ denotes the bitwise AND-operation.

The vector probability $v(A)$ of A is given by

$$\underline{v}(A) = [\underline{v}'(B) \mid \underline{v}'(C)] \quad (6-5)$$

where

$$\underline{v}'(X) = \begin{cases} \underline{v}(X) & \text{if edge (A,X) is positive} \\ \mathbf{1} - \underline{v}(X) & \text{if edge (A,X) is negative} \end{cases} \quad (6-6)$$

and $\mathbf{1}$ denotes a 1-matrix with the same dimensions as $\underline{v}(X)$.

If node X does not belong to the same correlation layer as A , the selection matrix and vector probability of X are chosen as

$$\underline{S}(X) = \mathbf{1} \quad (6-7)$$

$$\underline{v}(X) = [\text{SFP}(X)]. \quad (6-8)$$

Complexity Reduction

In general, the sizes of the selection matrix $\underline{S}(A)$ and of the vector probability $\underline{v}(A)$ grow linearly with the corresponding sizes of the child nodes. $\underline{S}(A)$ and $\underline{v}(A)$ can become very large since the number of L-paths is exponential in the number of input variables of the layer L . However, if two entries i and j in the vector probability are equal:

$$v_i(A) = v_j(A), \quad (6-9)$$

the two corresponding rows in the selection matrix $\underline{S}(A)$ can be joined by:

$$S_i' = S_i + S_j. \quad (6-10)$$

S_i can then be replaced by S_i' . S_j and v_j can be removed from $\underline{S}(A)$ and $\underline{v}(A)$, respectively.

The fulfillment of equation 6-9 seems to be very unlikely. But beside a few accidental hits it is always fulfilled if two L-paths share their T-nodes and their numbers of negative edges have the same parity. This limits the number of rows in the selection matrices and the size of the vector probabilities to two times the number of T-nodes of a layer. Thus, BDDs with a high node sharing rate also result in compact selection matrices and vector probabilities, which can be processed efficiently in terms of CPU time and memory.

Example

The following example will illustrate the use of the definitions and rules that have been introduced so far. Consider the following BDD:

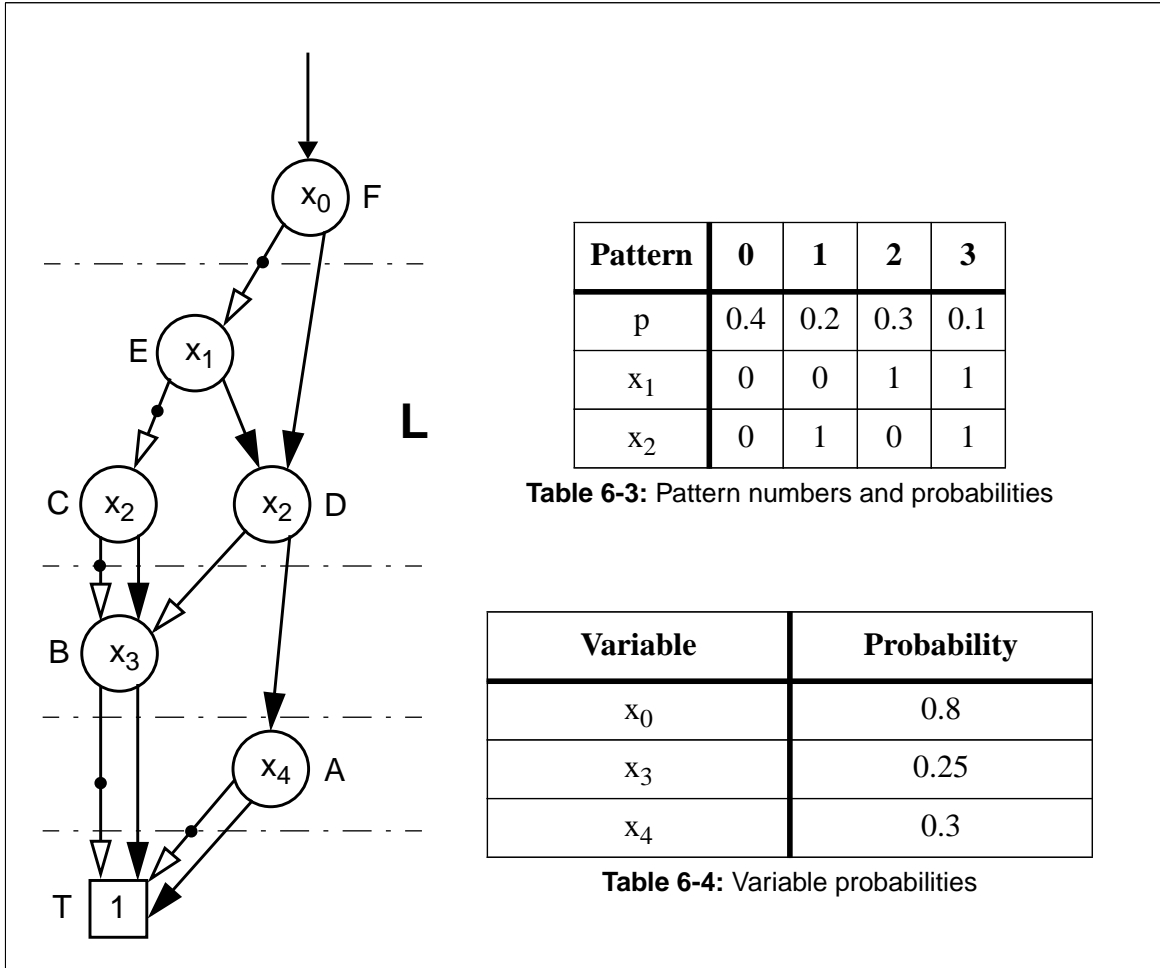


Figure 6-4: Example BDD

Nodes A and B can be easily evaluated because they are uncorrelated to any other variable:

$$\text{SFP}(A) = 0.3 \tag{6-11}$$

$$\text{SFP}(B) = 0.25 \tag{6-12}$$

The nodes C, D, and E belong to the same correlation layer L. Its path probability vector $\underline{p}(L)$ can be deduced from table 6-3:

$$\underline{p}(L)^T = [0.4 \ 0.2 \ 0.3 \ 0.1] \tag{6-13}$$

The vector probabilities and selection matrices of nodes C and D are given by

$$\underline{v}(C) = [1 - \text{SFP}(B) \ \text{SFP}(B)] = [0.75 \ 0.25] \tag{6-14}$$

$$\underline{v}(D) = [\text{SFP}(B) \text{ SFP}(A)] = [0.25 \ 0.3]. \quad (6-15)$$

With x_2 derived from table 6-3, their node matrices result in

$$\underline{S}(C) = \underline{S}(D) = \begin{bmatrix} \mathbf{1} \otimes \bar{x}_2 \\ \text{-----} \\ \mathbf{1} \otimes x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}. \quad (6-16)$$

The subfunction probabilities of node C and D can now be computed:

$$\text{SFP}(C) = \underline{v}(C) \cdot \underline{S}(C) \cdot \underline{p}(L) = 0.6 \quad (6-17)$$

$$\text{SFP}(D) = \underline{v}(D) \cdot \underline{S}(D) \cdot \underline{p}(L) = 0.265. \quad (6-18)$$

The next node to be evaluated is E . It belongs to the same layer as its two child nodes C and D . Therefore

$$\underline{v}(E) = [\mathbf{1} - \underline{v}(C) \mid \underline{v}(D)] = [0.25 \ 0.75 \ 0.25 \ 0.3] \quad (6-19)$$

$$\underline{S}(E) = \begin{bmatrix} \underline{S}(C) \otimes \bar{x}_1 \\ \text{-----} \\ \underline{S}(D) \otimes x_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6-20)$$

$v_0(E)$ and $v_2(E)$ are equal, therefore rows 0 and 2 in $\underline{S}(E)$ can be joined:

$$\underline{v}(E) = [0.25 \ 0.75 \ 0.3] \text{ and } \underline{S}(E) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6-21)$$

Hence

$$\text{SFP}(E) = \underline{v}(E) \cdot \underline{S}(E) \cdot \underline{p}(L) = 0.355. \quad (6-22)$$

Finally the root node F can be processed:

$$\underline{v}(F) = [1 - \text{SFP}(E) \ \text{SFP}(D)] = [0.645 \ 0.265] \quad (6-23)$$

Formally we can introduce the patterns for x_0 and the according probabilities:

Pattern	0	1
p	0.2	0.8
x_0	0	1

Table 6-5: Pattern numbers and probabilities for the single variable x_0

The pattern probability of the layer L_F with the single variable x_0 results then in

$$\underline{p}(L_F)^T = [0.2 \ 0.8] \quad (6-24)$$

and the selection matrix is given by

$$\underline{S}(F) = \begin{bmatrix} \mathbf{1} \otimes \bar{x}_0 \\ \mathbf{1} \otimes x_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (6-25)$$

Now the final probability of the BDD which is equal to $SFP(F)$ can be computed

$$SFP(F) = \underline{v}(F) \cdot \underline{S}(F) \cdot \underline{p}(L_F) = 0.341. \quad (6-26)$$

In this case, equation 6-26 can be simplified since the correlation layer consists of a single variable. The computation of $\underline{v}(F)$, $S(F)$, and $\underline{p}(L_F)$ can be avoided by the following formula:

$$SFP(F) = [1 - p(x_0)] \cdot [1 - SFP(E)] + p(x_0) \cdot SFP(D). \quad (6-27)$$

6.2.4 Logic Operations

There appear two logic operations during layer evaluation¹: the mask operations in equation 6-4 and the OR-operation in equation 6-10. For these operations either the bitparallel approach from chapter 5.3 or the set approach of chapter 4 can be applied. The input vectors in table 6-1 and the rows of the selection matrices \underline{S} are then represented by processor words or by sets, respectively.

For the set approach the pattern probabilities $\underline{p}(L)$ must be converted into block sizes with the following trick: each column j is repeated n_j times where n_j is given by

$$n_j = \text{rnd}(p_j(L) \cdot K). \quad (6-28)$$

$\text{rnd}()$ is the rounding operator, and K is a constant that should be chosen sufficiently large in order to obtain accurate approximations of the real pattern probabilities. These new patterns can now be represented by sets. Equation 6-2 degenerates than to

1. This is the reason for the name *Probabilistic - Logic Simulation*.

$$\text{SFP}(X) = \frac{1}{K} \cdot \underline{v}(X) \cdot \begin{pmatrix} \text{wt}(S_0(X)) \\ \dots \\ \text{wt}(S_{m-1}(X)) \end{pmatrix} = \frac{1}{K} \cdot \sum_{i=0}^{m-1} v_i \cdot \text{wt}(S_i) \quad (6-29)$$

where $\text{wt}(S_i)$ is the Hamming weight of the i th row of the selection matrix $\underline{S}(X)$. Hence, the probabilistic-logic simulation can be considered as a more sophisticated alternative to the vector recomposition method proposed in chapter 4.6. However, in contrast to vector recomposition, this approach handles uncorrelated signals correctly and does not suffer from the unwanted correlations that are caused by recomposition. Thus, very small and efficient signal groups must not be avoided anymore.

6.2.5 Results

The algorithms presented in the previous sections were tested on several benchmark circuits. All the experiments were executed on a Sparc Ultra 2, 300 MHz, Sun Solaris 2.6 with 1.4 GB main memory. However, the latter had been limited to 1 GB. If the memory requirements of an experiment were higher, it was cancelled. The benchmark circuits were not mapped on a specific library but were simulated with the generic gates as published in [Brgl 85] and [Brgl 89]. Unless stated otherwise, the switching activity as well as the static probabilities of the PIs were chosen to 0.5, the switching activity for the sequential circuits was chosen to 0.25. For the logic operations the bitparallel approach was used because it has proven to be more efficient in the ZDM ([Schn 95] and chapter 4, respectively).

In a first set of experiments the influence of the variable order on memory requirements and runtime is considered. For all later experiments, accuracy is the major issue. The second set of experiments deals with the question, how a reference result for the accuracy investigations can be obtained. Thirdly, Najm's approximation [Najm 93] is compared to XOR-BDDs [Schn 94], assuming uncorrelated PIs in order to estimate the effect of Najm's simplifications. Finally, several experiments with correlated input patterns demonstrate the accuracy improvements due to the probabilistic-logic approach compared to other methods that neglect the spatial correlations between the PIs.

Variable Order

It was already mentioned before that the size of a BDD is highly influenced by the variable order. In order to quantify this effect, several medium sized circuits were simulated using a random and an optimized variable order. Since algorithmic determination of good variable orders is very time consuming, optimized orders from [Some 98] and [Dors 98] were used here. Note that these orders may not represent the actual optimum for the XOR-BDDs because they were originally generated for the logic functions of the circuits and not for their switching functions. But it can be assumed that they are a rather good approximation.

Figures 6-5 and 6-6 show the results in terms of the runtime and the number of generated BDD nodes, respectively. The latter is a measure for the memory requirements. The number of BDD nodes was preferred here to the actual memory requirements because the latter are highly biased for smaller circuits by the program code itself and other static variables. The

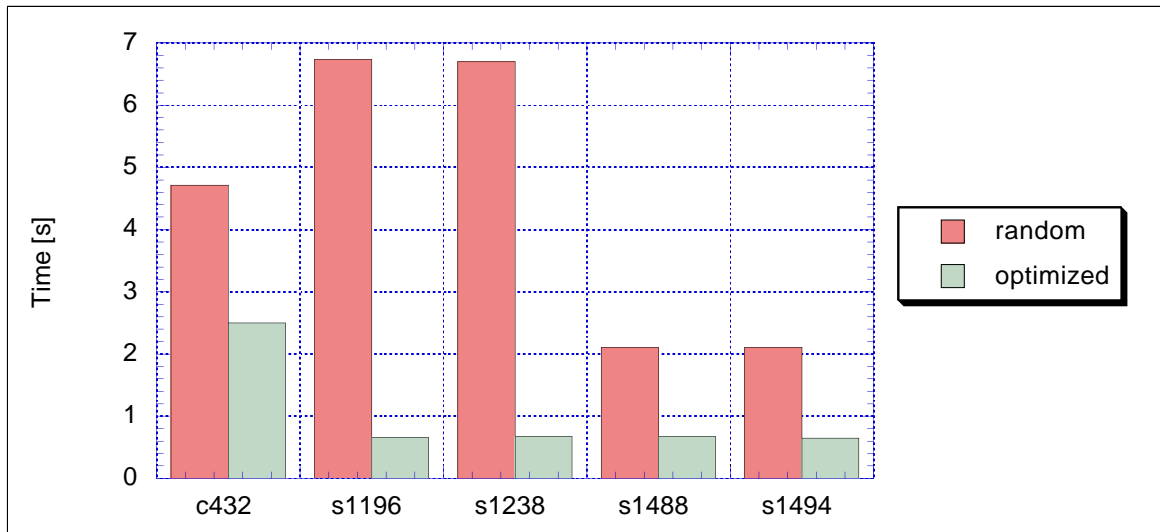


Figure 6-5: Simulation times for random and optimized variable orders

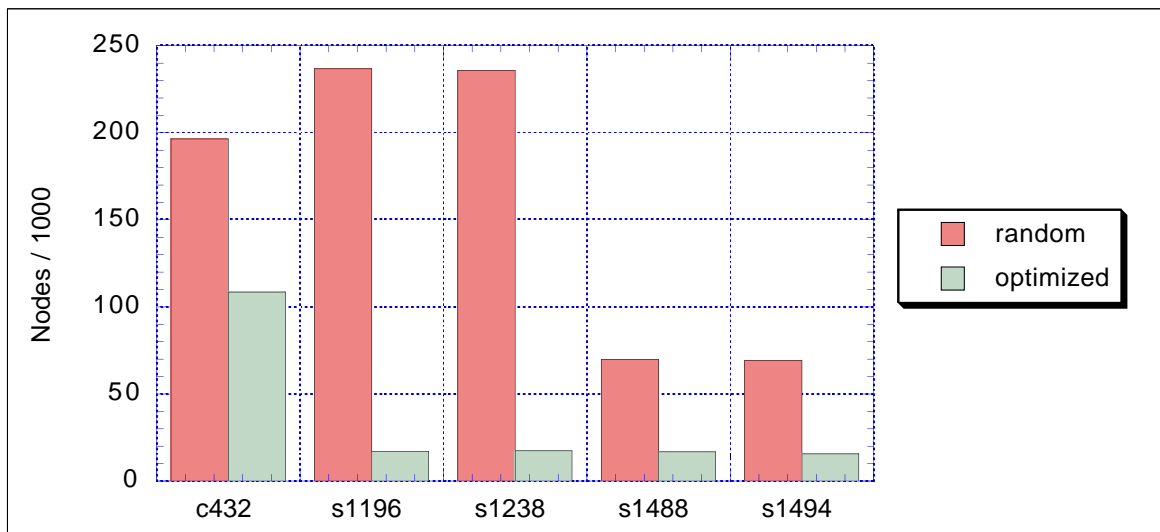


Figure 6-6: Number of BDD nodes for random and optimized variable orders

optimized versions simulate 2-10 times faster than those with a random variable order. The differences in the node numbers are even more significant. It should be noted here that larger circuits could not be used for these experiments since simulations with random orders exceeded the 1 GB memory limit. Thus, the availability of good variable orders is crucial for large circuits.

Accuracy of Logic Simulation

In the following, the previously presented probabilistic simulation approaches, i.e. Najm's approximation, XOR-BDDs, and the probabilistic-logic approach are investigated for the three properties that are important from a practical point of view: runtime of the simulation, memory requirements, and most important, accuracy of the results. Especially the latter imposes a serious problem: it implies the knowledge of the exact results which are usually not available. By definition, the exact result equals the limit that is approached by a logic

simulation with an infinite number of typical, random input patterns. Obviously, infinite simulation is not a practical solution. Alternatively, probabilistic simulation yields the exact result as well, as long as the PIs are uncorrelated. However, for arbitrarily spatially correlated input streams exact probabilistic methods don't exist. In that case, it is only possible to obtain an approximation through logic simulation of a high number of input patterns. Monte Carlo simulation [Najm 93] offers one possibility. But the number of test patterns that are required for a certain accuracy heavily depends on the circuit structure and the signal statistics [Hill 95]. It cannot be estimated before the actual simulation.

In order to get an idea of the quality of logic simulation, the following experiment was carried out on seven combinational circuits from the ISCAS-85 benchmark set [Brgl 85]. Two types of simulations were performed: probabilistic simulations with XOR-BDDs, assuming spatially uncorrelated PIs, and logic simulations with different numbers of spatially uncorrelated test patterns. Under these assumptions, the results of the probabilistic simulations represent the exact values.

Figure 6-7 depicts the deviations between logic and probabilistic simulation which were averaged over the seven circuits. As before, the global and local deviations are depicted. Even for as few as 1000 input patterns the global results of logic and probabilistic simulations are very close, while the local deviations are one order of magnitude higher, hence requiring substantially more patterns for the same accuracy. But the results of the logic simulation clearly converge versus the probabilistic simulation with increasing number of test

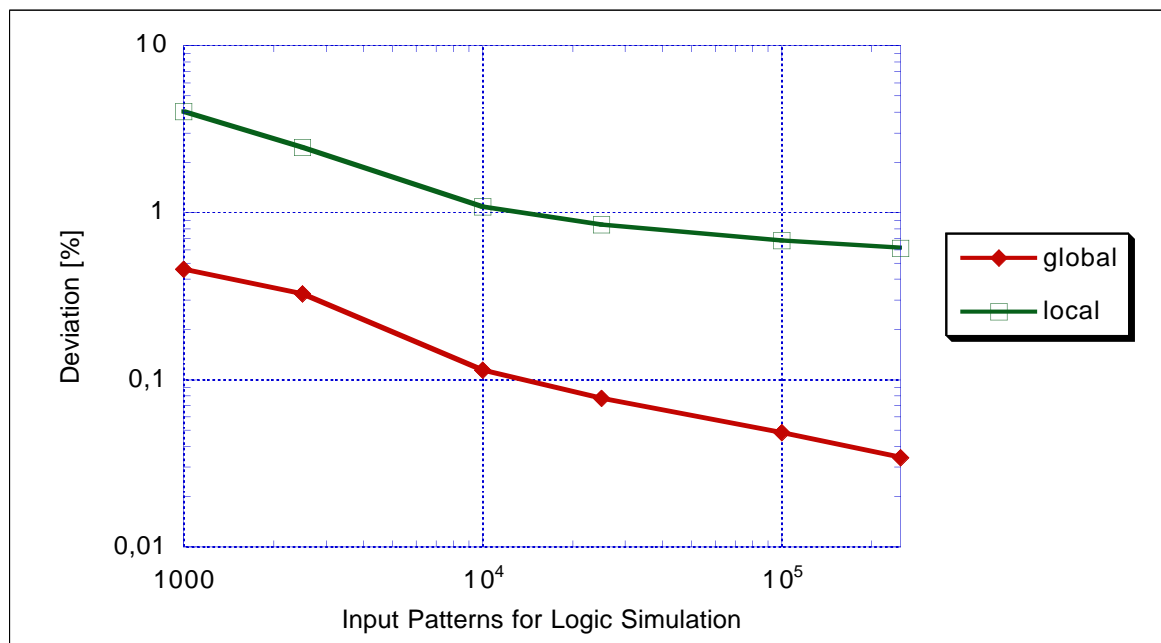


Figure 6-7: Deviations between logic and probabilistic simulation

patterns for both types of deviations. This justifies the assumption that probabilistic simulation yields the exact result if the correlations are accurately modeled. However, if accurate modeling is not possible, logic simulation with 20 000 vectors or more will be used as refer-

ence in agreement with [Schn 95]. But one has always to be aware of the accuracy limits of this approach. Therefore the term *error* will be consciously avoided in the sequel in favor of the term *deviation*.

Najm's Approximation versus XOR-BDDs

The concept of the correlation layers can be applied to any BDD-based signal activity estimation approach. Equation 4-14 indicates that Najm's approximation leads to more but smaller BDDs because Najm's BDDs require only half as many variables as the XOR-BDDs. The following two figures depict the results that were obtained by a comparison of Najm's approximation and the XOR-BDDs on several benchmark circuits.

Figure 6-8 summarizes the runtimes (*CPU*) and total number of BDD nodes that were created during the simulations with XOR-BDDs. The values are normalized to the according values of simulations with Najm's approximation. Thus, values greater than 1 indicate that

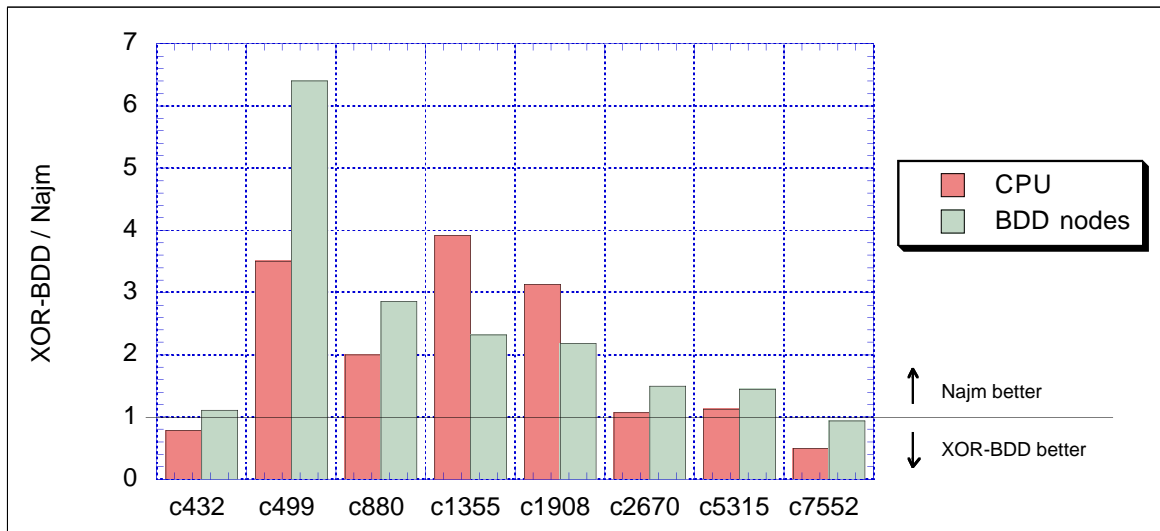


Figure 6-8: CPU and memory requirements of Najm's approximation versus XOR-BDDs

the requirements for XOR-BDDs were higher. The results reveal that Najm's approximation requires less resources in most cases, only for *c432* and *c7552* the XOR-BDDs are more economical. However, the global errors, which are depicted in figure 6-9, nullify this advantage for any practical application. Error rates of more than 100% make the estimation result absolutely useless, a fact that was already indicated by equation 4-17.

As a consequence of these results, only XOR-BDDs are used for all following experiments where the influence of the probabilistic-logic simulation approach on the simulation of spatially correlated input streams is investigated. Two types of spatial correlations are considered: the first are created by the random test pattern generator [Dall 98] and the second by RTL simulation of sequential circuits. They are called *correlated signals* and *FSM signals* in the following. For details about these signal types and test pattern generation see chapter 4.5.5. The experiments with correlated signals are usually performed with the combinational benchmark circuits from [Brgl 85] (e.g. *c499*), while the experiments with FSM signals are carried out with the sequential benchmark circuits from [Brgl 89] (e.g. *s298*).

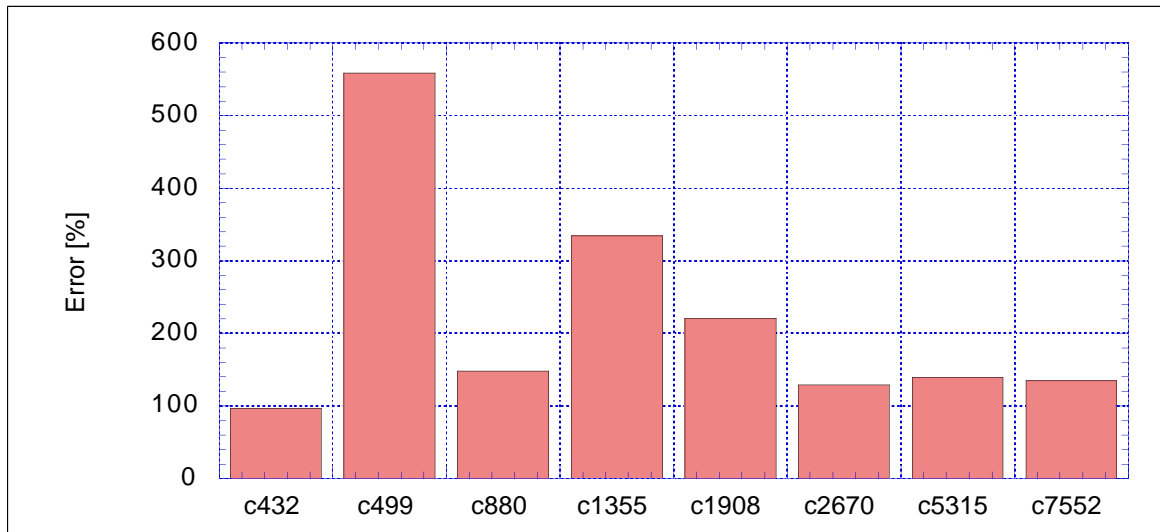


Figure 6-9: Global error of Najm's approximation

For both correlation types the global and local deviations are depicted in detail. Firstly if correlations are neglected and secondly if probabilistic-logic simulation is applied. The correlation groups for the latter were identified with the algorithm of listing 4-1.

Correlated Signals

For this set of experiments 25 000 test vectors were generated and the PIs were partitioned into groups of four correlated signals. From these test vectors the statistic properties and the correlation groups were derived for probabilistic and probabilistic-logic simulation. The test vectors were directly used to obtain the reference results through logic simulation.

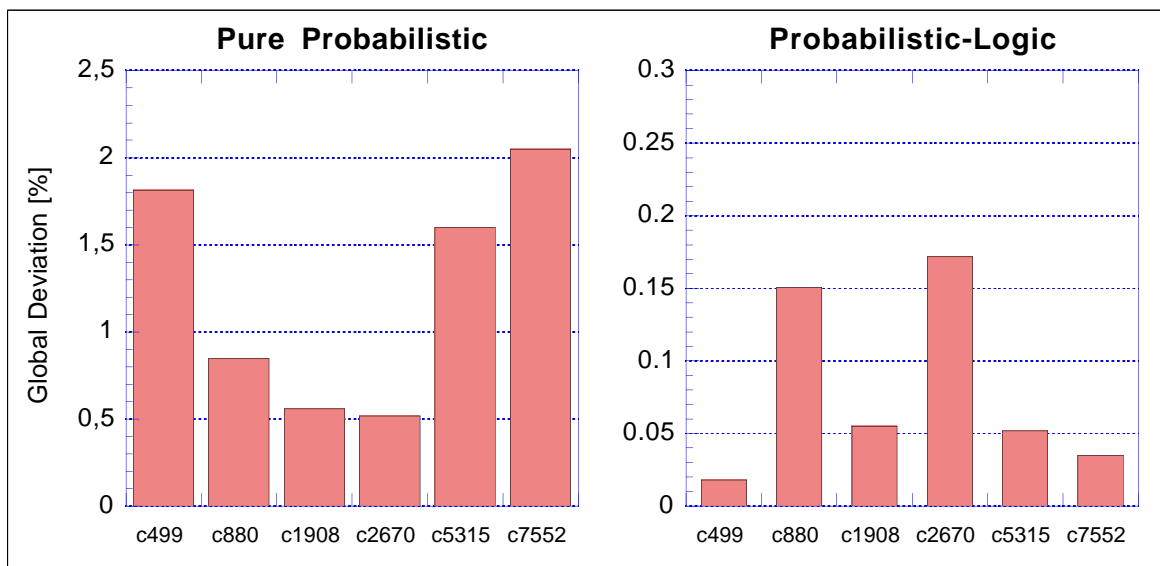


Figure 6-10: Global deviations for probabilistic and probabilistic-logic simulation

Figures 6-10 and 6-11 compare the local and global deviations of pure probabilistic and probabilistic-logic simulation, respectively. Obviously, the probabilistic-logic approach yields significant improvements. Its deviations to logic simulation are one to two orders of magnitude lower as for the pure probabilistic method which neglects any spatial correlations.

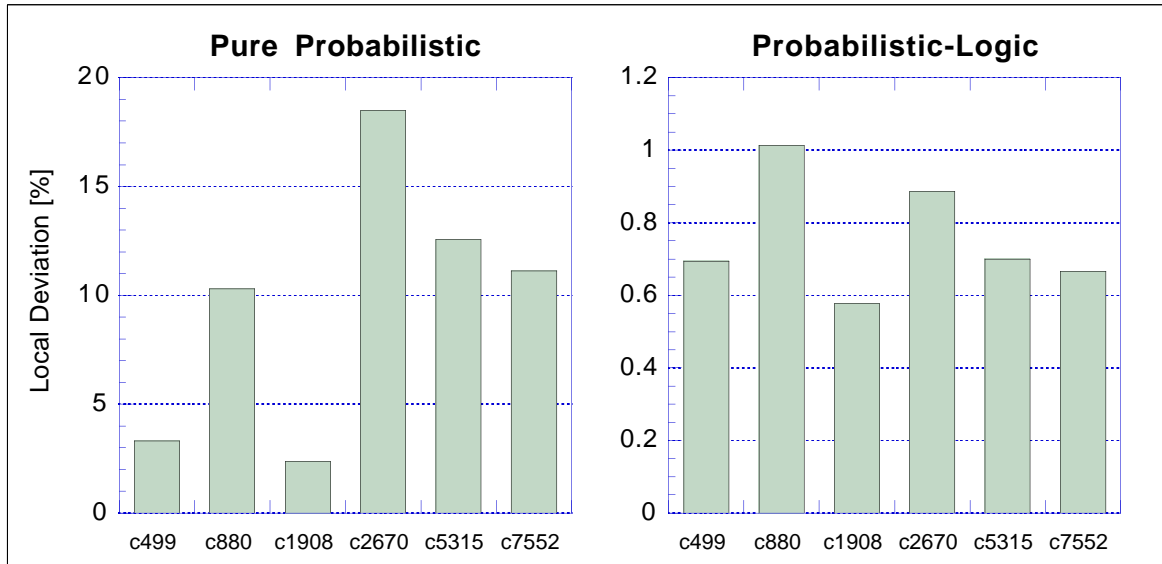


Figure 6-11: Local deviations for probabilistic and probabilistic-logic simulation

FSM signals

The same experiments as before were carried out on sequential circuits with FSM signals. Figure 6-12 shows the large estimation errors if the correlations are neglected by pure probabilistic simulation. In some examples the global deviations are as high as 30% and the local errors as high as 65%.

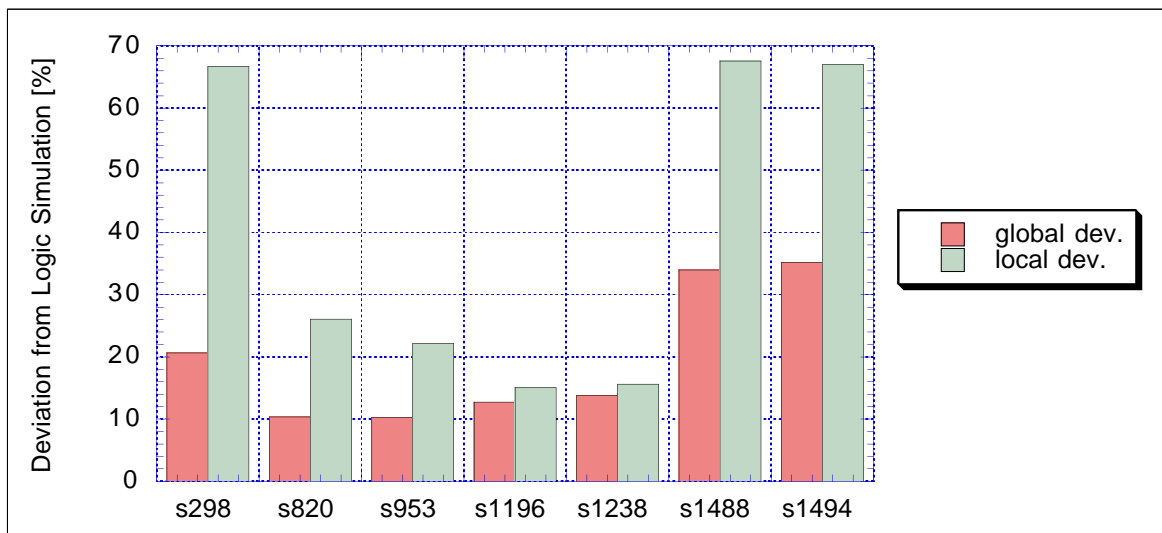


Figure 6-12: Global and local deviation if correlations are neglected

For the results in figure 6-13 probabilistic-logic simulation was applied. As in the previous experiments, the estimation results are importantly improved; note the different scales of the y-axis in figures 6-12 and 6-13. The average global and local deviations are reduced from 20% to 5% and from 40% to 8%, respectively. The absolute deviation rates become not as

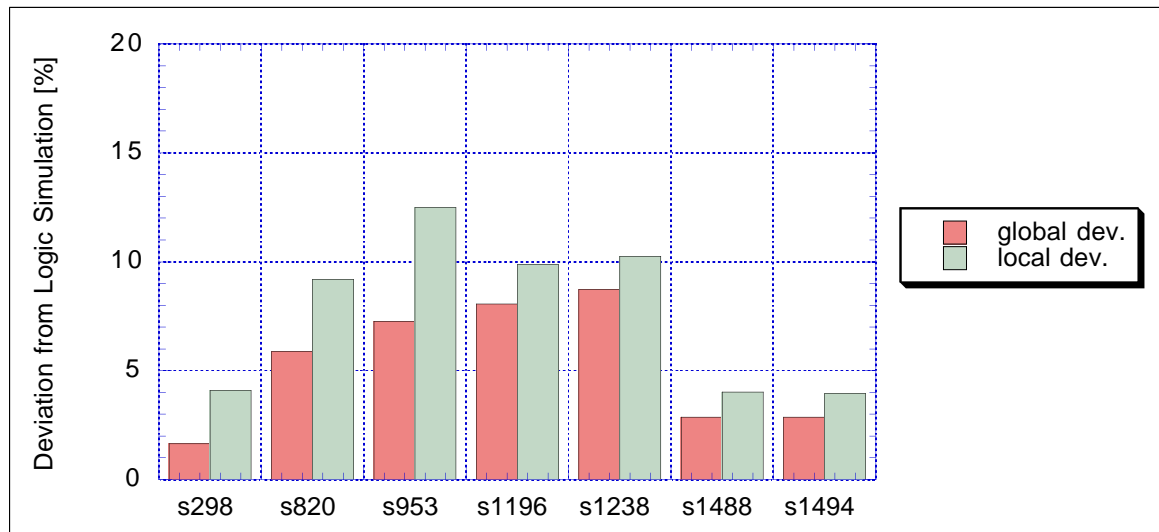


Figure 6-13: Global and local deviation if correlations are taken into account

low as in the experiments of the previous subsection because the probabilistic-logic approach can only accept limited group sizes. Thus, minor correlations must be neglected, which leads to higher estimation errors.

6.3 Covering

6.3.1 Problems of BDD approaches

The probabilistic-logic approach, which was presented in the last section, shows very accurate results. However, the size of the BDDs and the runtime depend heavily on the variable order. For unfavorable orders the memory requirements of the BDDs easily exceed 1GB even for smaller circuits like C432. Thus, good variable orders are crucial. On the other hand, the problem of finding the best variable order is NP-complete [Sasa 96] and all the approximate, heuristic solutions proposed so far are very time consuming. Further, there are classes of circuits where it has been proven that the size of their BDDs is exponential in the number of variables for any variable order, e.g. multipliers (C6288) [Brya 86].

The problem of the variable orders is even increased by the correlation layers where the variable order is given by signal properties rather than circuit properties. Thus, the search space for good variable orders is limited and just the best orders may be prohibited. For this reason only medium sized circuits could be chosen for the experiments in section 6.2.5. The practical significance of any BDD based approach could be importantly improved if the influence of the variable order was decreased.

6.3.2 Local BDDs as Speed - Accuracy Trade-off

It was already observed in the past [Marc 95] that the influence of spatial signal correlations becomes less significant if the correlated signals pass many gates before they reach the gate where they are actually linked together. For gates that are far away from the PIs, there exist three effects which reduce the influence of the spatial correlations between the PIs. Firstly, there are many other variables which influence the switching activity of such a gate as well and thus reduces the influence of each single PI. Second, there are over- and underestimation effects that compensate each other. Finally, new correlations develop which are caused by reconvergent fanout. They may dominate the correlations between the PIs.

Thus, it seems to be useful to partition the circuit into smaller subcircuits with a limited number of primary inputs. In fact, this approach was taken by most researchers who presented BDD based approaches for switching activity estimation, e.g. [Najm 93] or [Chou 94]. However, only in [Kapo 94] a systematic dedicated partitioning algorithm was proposed. The heuristic that was presented there, splits the circuit into partitions with a limited number of inputs and a maximum number of gates. The partitions are then simulated separately under the assumption of spatial independence of the partition inputs. This can lead to important estimation errors at the partition borders as the following example illustrates. Assume that the two partitions which are depicted in figure 6-14 are parts of a larger circuit. The signal activity of the output of gate Y is to be estimated. Its input signals are

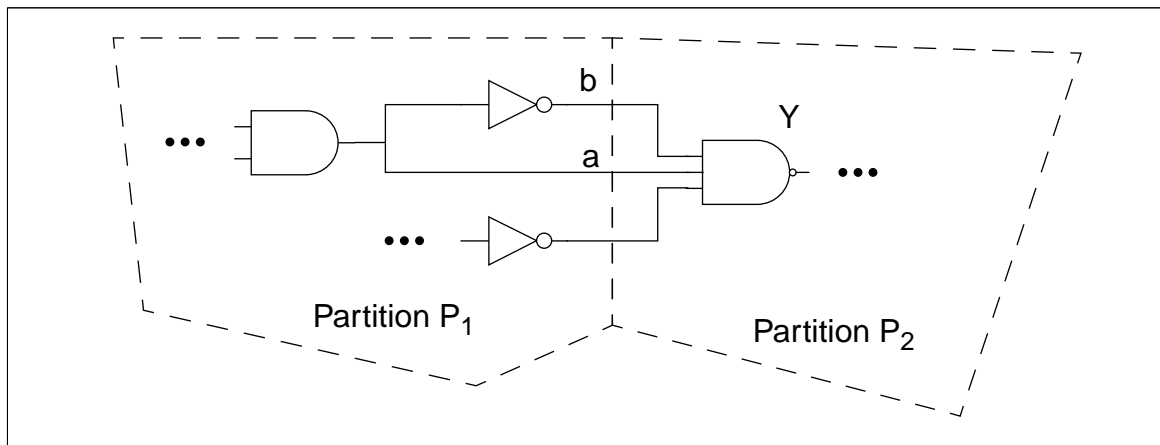


Figure 6-14: Circuit with two partitions

considered spatially independent because all three inputs of gate Y originate from gates that belong to another partition. However, this assumption cannot be made since signals a and b are strongly connected ($b = \bar{a}$), which results in a high estimation error for this specific gate. Consequently, Kapoor didn't mention absolute estimation errors of his approach but only "improvements in the number of low error nodes" compared to other approaches.

Local BDDs

The switching activity of a gate that is connected to the output of gate Y , may result into a smaller estimation error because the gate's input signals have already passed another gate since the partition borders. Thus, a certain error compensation effect exists already. In order

to obtain the same compensation effect for all gates it seems useful to build an independent “partition“ for each gate with a defined maximum number of inputs. However, these circuit parts don’t represent a mathematical partition anymore because they are not disjoint. They rather form a *cover*. In connection with BDDs the blocks of the cover are also called *local BDDs* and the input variables of the local BDDs are called *secondary inputs (SI)*. Figure 6-15 shows a circuit with two 3-input blocks, for gate *a* and gate *c*, respectively.

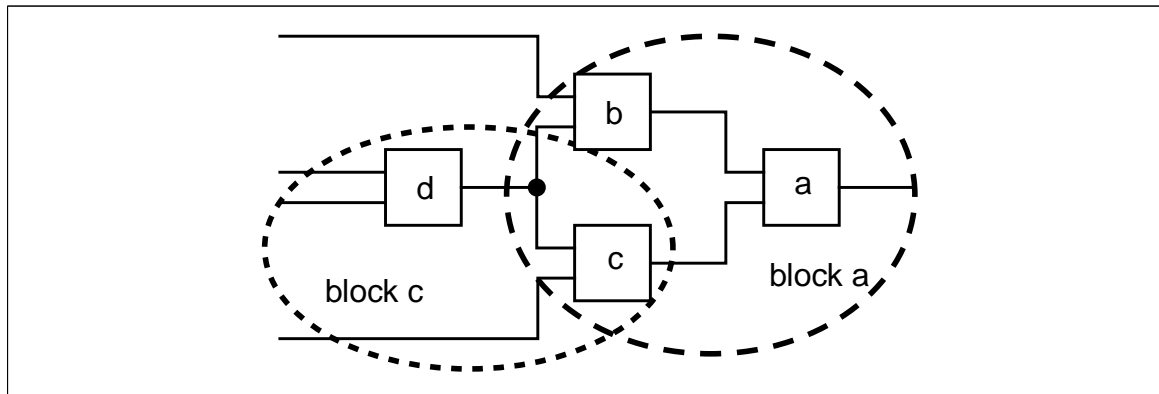


Figure 6-15: Circuit with 3 input blocks for gate *a* and *c*

For our purpose, the following three properties are required for the local BDDs, which are called *accuracy preserving requirements*:

1. The maximum number of SIs is limited to an upper bound L .
2. The local BDDs should cover a maximum number of gates.
3. Reconvergent fanouts should be hidden in the local BDDs as far as possible.

While the first claim guarantees reasonable limited BDD sizes, claims two and three ensure the maximum accuracy of the estimation algorithm.

6.3.3 The Covering Algorithm

Assume an arbitrary circuit, e.g. the circuit of figure 6-16. The task of the *covering algorithm* is to find the largest possible block for gate *a* with L inputs (e.g. $L=3$). For this purpose we define three sets, B , I_B , and E_X :

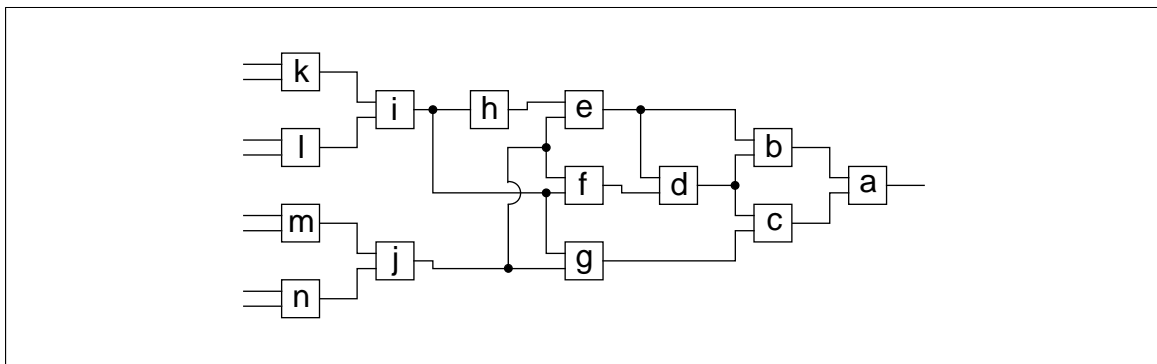


Figure 6-16: Circuit for which a local BDD is searched

1. B contains all gates that have been merged to the current block so far. It is initialized with the gate for which the local BDD is to be built.
2. I_B contains the input signals of the current block.
3. The extension set E_X contains the additional SIs that would occur if a set of gates X was added to the current block B .

For the covering algorithm we will use the following convention: all signals get the same name as the gate they fanout from. Thus, $B \cup I_B$ is the union of the gates of B and the gates where the signals from I_B fanout from.

In the example of figure 6-16 B and I_B are initialized as:

$$B = \{a\}$$

$$I_B = \{b, c\}.$$

The number f_B of free SIs for the current block is given by

$$f_B = L - |I_B|. \quad (6-30)$$

The algorithm tries to add subsets X_i from I_B to the current block B . The maximum size of the subsets X_i is limited to two for runtime reasons. Hence, only single gates or gate pairs can be added to B during one step of the algorithm. For each candidate X_i the input set I_{X_i} and the extension sets $E_{X_i} = I_{X_i} \setminus I_B$ are computed. The number of additional inputs n_+ that would result from adding X_i to the current block B is given by

$$n_+ = |E_{X_i}| - |X_i|. \quad (6-31)$$

In order to account for reconvergent fanouts, it is important to compute the number of input signals n_c that are shared between the gates in X_i or between the gates in X_i and B :

$$n_c = \left(\sum_{\text{all gates } x \text{ from } X_i} |I_x| \right) - |E_{X_i}|. \quad (6-32)$$

In the given example, there are three possible subsets X_i . They are stored together with their properties in a sorted cost list. This list is depicted in table 6-6 for the initial block $\{a\}$.

i	X_i	n_c	n_+	E_X
0	b,c	1	1	d,e,g
1	b	0	1	d,e
2	c	0	1	d,g

Table 6-6: Properties of possible extensions for initial block $\{a\}$

In order to meet the *accuracy preserving requirements*, the cost list is sorted according to the following criteria in ascending order of their priority:

Priority	Criteria
1	Descending in the number of common signals n_c .
2	Ascending in the number n_+ of additional secondary inputs.
3	Descending in the number of gates to be added to the block: $ X_i $.

Table 6-7: Sort criteria for the cost list

If two entries have equal sort criteria they are sorted in a random way. While criterion 1 causes reconvergent fanouts to get hidden in blocks, criteria 2 and 3 ensure blocks with largest possible size.

The cost list is then scanned from top to bottom and the first element that meets the constraint

$$|E_x| \leq f_B \quad (6-33)$$

is added to the current block. In the example $f_B=1$, i.e. element X_0 is added. This element is removed from the cost list and all other entries are updated. The gates that have just been added to the block B are removed from all remaining entries in the cost list and their properties (n_c , n_+ , E_x) are recalculated. Possibly, some entries must be removed from the cost list since they became empty. Then, the next element is chosen from the cost list and added to the block. Only if the cost list becomes empty or if no element of the cost list meets the SI constraint, the cost list is refilled by investigating the SIs of the current block B . So, all paths in the block to become similar in depth.

The largest possible block is found if no more gates can be added to the block without violating the SI constraint. Listing 6-1 details the covering algorithm in pseudo code. An elaborate example can be found in appendix D.

6.3.4 Results

Block Size

Obviously, the estimation error decreases as the number of SIs for the local BDDs increases because more correlations through reconvergent fanout can be taken into account. But runtime and memory requirements will increase at the same time. This increase is very likely to be exponential since good variable orders are not available for the local BDDs. Hence, it is necessary to find an acceptable trade-off between accuracy on the one hand, and memory and CPU requirements on the other hand. The number of SIs of a block is referred to as *block size* in the sequel.

```

1. Build the initial block containing the root gate a: B={a}
2. Build the input set IB containing the SIs of block B
   and compute the number of remaining free inputs fB=L-|IB|
3. For each possible subset X of IB with |X|≤2
   Begin
       Compute the input set IX =  $\bigcup_{\text{all gates } x_i \text{ of } X} I_{x_i}$ 
       Compute the extension set EX=IX\IB
       Compute n+ and nc according to equations 6-31 and 6-32
       Insert the entry (X,nc,n+,EX) into the cost list
   End
4. Sort the cost list according to the criteria of table 6-7
5. Select the highest element Xs from the cost list with n+≤fB
6. If no element is found that fulfills requirement 5. then
   If cost list hasn't changed since last execution of 3. then
       return B as the maximum block with |IB| ≤ L
   Else
       continue with 3.
7. Add Xs to the current block: B = B∪Xs
   IB=(IB\Xs)∪E(Xs)
   fB = |IB|
   Remove Xs from the cost list
   For all remaining elements Xi of the cost list
   Begin
       Xi = Xi\Xs
       Recompute the cost criteria nc, n+, and EXi of Xi
   End
8. continue with 4.
    
```

Listing 6-1: The covering algorithm

In order to find an acceptable trade-off between accuracy and resource requirements, nine combinational circuits¹ were simulated using XOR-BDDs without correlation layers. The reference results were obtained from a logic simulation with 25 000 uncorrelated test vectors. The block size of the probabilistic simulation was varied between 4 and 19 for all circuits. Figure 6-17 depicts the resource requirements. They are normalized to the values of the initial block size 4. The accuracy results are summarized in figure 6-18. All results in figures 6-17 and 6-18 are averaged over the nine benchmark circuits.

1. c432, c499, c880, c1355, c1908, c2670, c3540, c5315, and c7552. c6288 had to be omitted here because it could not be simulated with *block size*>15.

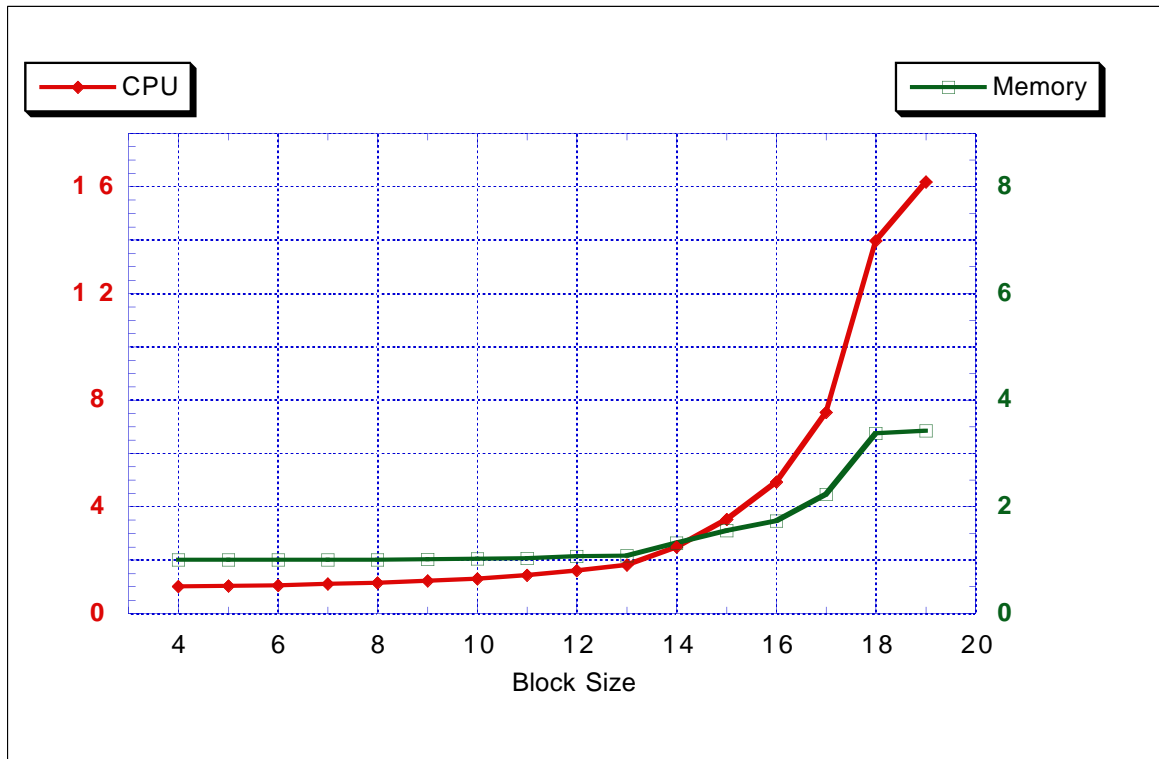


Figure 6-17: Increase of CPU and memory requirements with block size

The resource requirements in figure 6-17 show clearly the exponential behavior that could have been expected before, because optimized variable orders are not available for the local BDDs. Note that the actual figures of the CPU and memory requirements are related to the left and right axis, respectively. Both curves remain relatively flat until block size 12 or 13. Then they start to rise considerably. The dents at block size 19 are caused by one single circuit (c2670). It results by chance into very efficient local BDDs for this specific block size. Larger block sizes could not be simulated since some circuits exceeded the memory limit of 1 GB.

The accuracy curves which are depicted in figure 6-18 are not as smooth as the curves of the resource requirements. A larger block size cannot always guarantee more accurate results because of the unpredictable random error compensation effect (chapter 4.6.4). However, as the curves of figure 6-18 reveal clearly, the probability of estimation errors decreases with increasing block size.

For block sizes greater than 12, the global and local deviations reach values that are comparable to those of an accurate probabilistic simulation. Thus, the effect of the local BDDs on the accuracy becomes negligible if the block size is greater than 12, at least as long as the PIs are spatially uncorrelated. For all following experiments block size 13 was chosen because of its modest resource requirements and high accuracy.

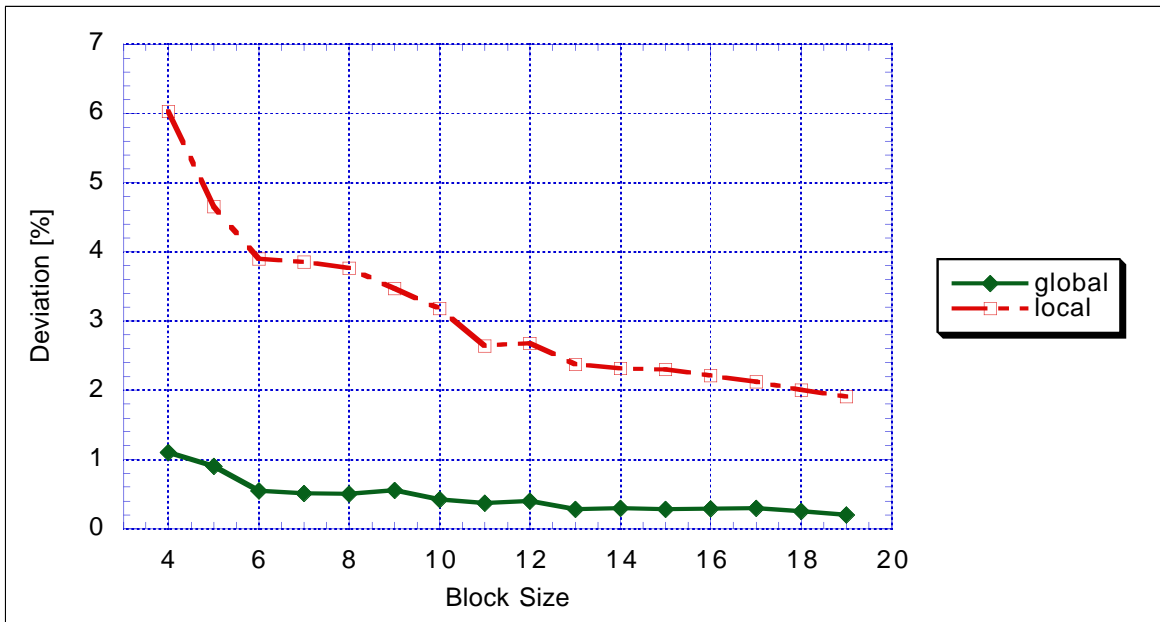


Figure 6-18: Influence of block size to estimation accuracy

Combinational Circuits with Uncorrelated and Correlated Signals

This section details the results that were obtained by simulating several combinational circuits with uncorrelated and correlated inputs, and the probabilistic-logic approach with local BDDs. Figure 6-19 depicts the global deviations from logic simulation. They are well below 5% in all cases with average values of 0.5% and 0.7% for uncorrelated and correlated signals, respectively.

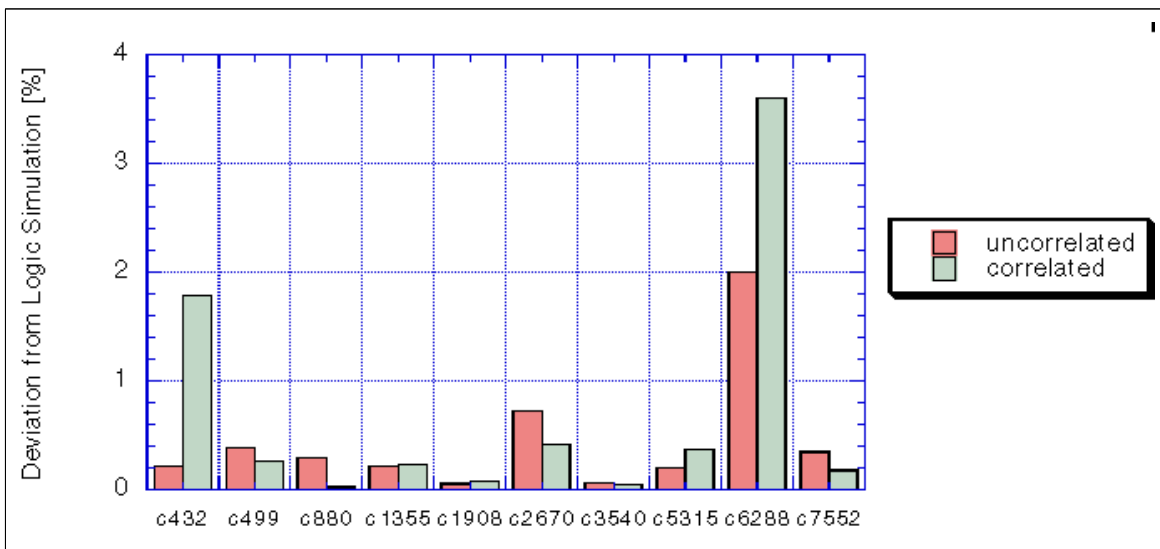


Figure 6-19: Global deviations for uncorrelated and correlated PIs with local BDDs

The local deviations are depicted in figure 6-20. They average to 2.6% and 3.4%, respectively. Consequently, they are some 5 times higher than the global deviations but most values are still below 5%. In general, the average deviations are higher for correlated signals

than for uncorrelated signals. This demonstrates the influence of the spatial correlations between the PIs which are neglected at the block inputs. However, this influence is moderate, which justifies the assumption that spatial correlations between the PIs only influence the switching activity of gates close to the circuit inputs.

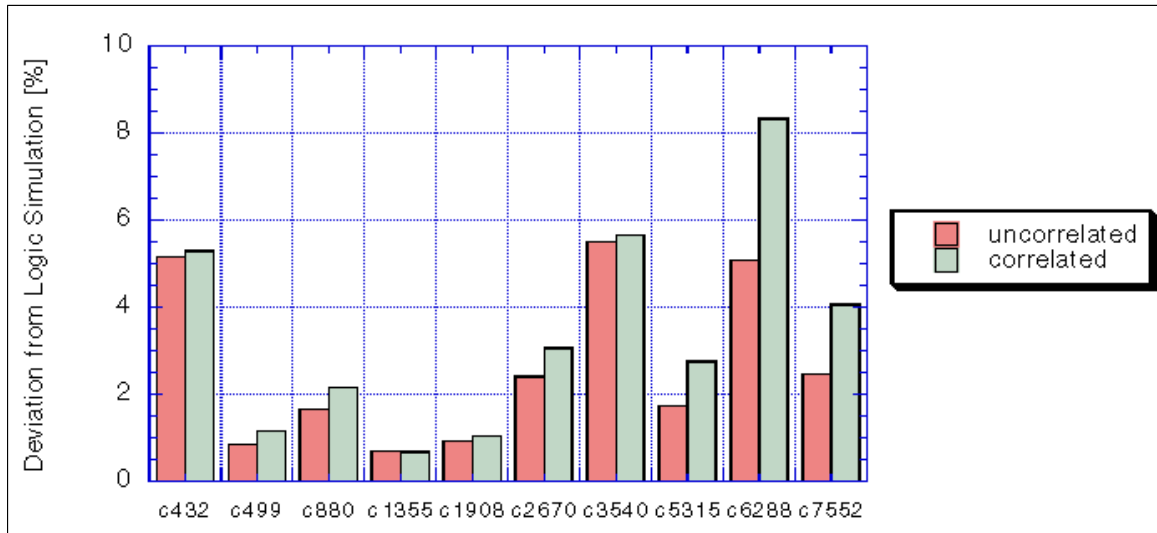


Figure 6-20: Local deviations for uncorrelated and correlated PIs with global BDDs

FSM Signals

The experiments with sequential circuits and FSM signals from section 6.2.5 were also carried out with local BDDs. The latter enabled the simulation of much larger circuits, though. The accuracy results are summarized in figure 6-21. The global and local deviations average to 5.9% and 9.1%, respectively. Like in previous experiments, the deviations to logic simulations are greater than for combinational circuits with correlated inputs because minor correlations must be neglected. Compared to figure 6-13, the increase in the deviations reaches from negligible for s298, s820, and s1238 to a factor 2 for s1488 and s1494. Obviously, the

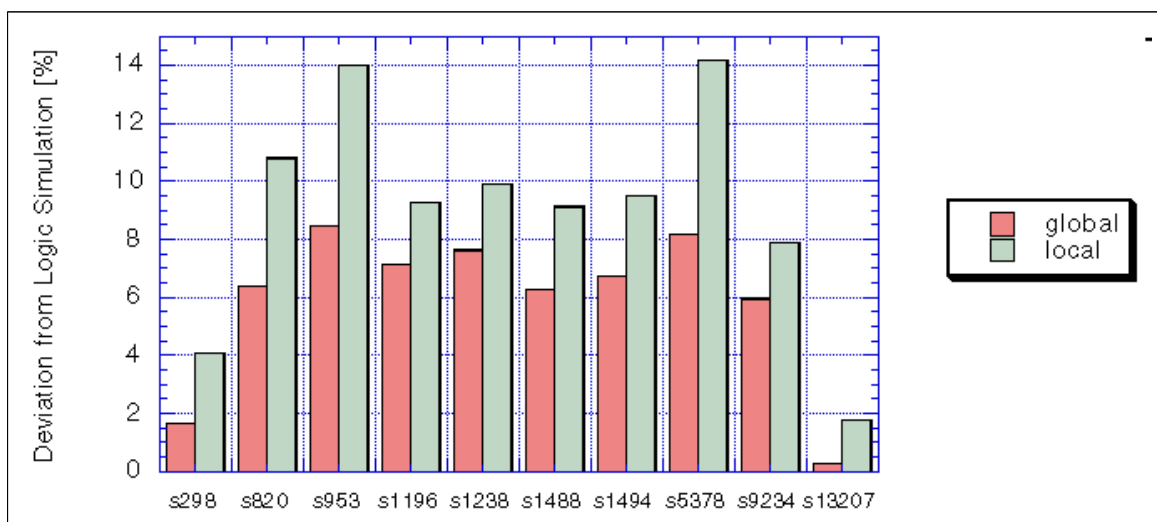


Figure 6-21: Global and local deviation from logic simulation

latter is very sensitive to correct correlation handling. A fact that can also be observed by comparing figures 6-12 and 6-13. Nevertheless, the global deviations remain below 10% for all circuits and the local deviations are also around 10% in most cases.

Runtimes

Finally, figure 6-22 summarizes the speedups that were obtained with the local BDDs. Note that not all circuits from the previous subsections can be presented here. Some larger circuits like *c6288* could not be simulated with the exact approach since their memory requirements exceeded the 1 GB limit.

Local BDD based simulation is 65 times faster than the exact approach, on the average. However, there is a high variation in the speedup figures, ranging from 1 for *s298* to more than 500 for *c499*. This variation has the following reason: the simulation times of the exact

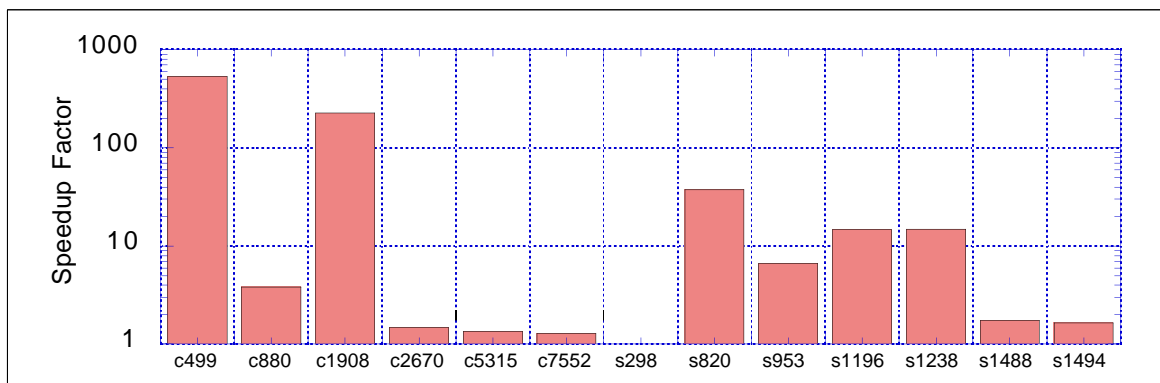


Figure 6-22: Speedup with local BDDs over pure probabilistic-logic simulation

approach depend on the BDD sizes, rather than on the circuit size, while the local BDD approach depends more on the actual circuits size because the sizes of all local BDDs are roughly constant. Thus, circuits that have an efficient BDD representation are less likely to be accelerated by local BDDs. This effect is even increased if optimized variable orders are used, as it was necessary for the combinational circuits in these experiments in order to remain within the memory limits. However, it was already mentioned that in practice good variable orders are not known in most cases. Thus, the higher figures of figure 6-22 are more likely to represent the relevant case.

6.4 Summary

Today, two major approaches for probabilistic switching activity estimation exist: *Najm's approximation* and *XOR-BDDs*. Both are based on function representation by BDDs. It was shown that *Najm's approximation* is less resource consuming but produces estimation errors of more than 100%. On the other hand, it could be demonstrated that *XOR-BDDs* are accurate as long as the primary input variables are spatially uncorrelated. However, this method becomes inaccurate if spatial correlations between the PIs are not negligible. A novel approach was presented that addresses that problem. Its basic idea relies on choosing the variable order of the BDDs such that correlated variables are adjacent. Thus, there exist lay-

ers of correlated variables in the XOR-BDDs, also called *correlation layers*. Inside the correlation layers signal correlations are taken into account by replacing probabilistic computations through path probabilities. For the operations inside the correlation layers, ordinary logic operations play an important role. Therefore, the new method has been called *probabilistic-logic simulation*.

The problem of finding an accurate reference result was addressed next. If all correlations can be correctly modeled, the result of probabilistic simulation is exact and corresponds to logic simulation with an infinite number of test vectors. This could be justified by comparing probabilistic simulation to logic simulation with an increasing number of uncorrelated test vectors. However, as the test vectors get spatially correlated, exact probabilistic methods do not exist anymore. Therefore, it was decided to use the result of a logic simulation with 25 000 test patterns as the reference.

The probabilistic-logic simulation method was implemented into a prototype simulator. Several experiments were carried out on combinational circuits with correlated inputs as well as on sequential circuits. In both cases the deviations from the reference result could be reduced by one to two orders of magnitude compared to approaches that neglect spatial correlations between the PIs.

However, any BDD based approach depends on the availability of good variable orders in order to keep CPU and memory requirements within reasonable limits. Finding good variable orders is a very time consuming task, though. This problem is even aggravated by the probabilistic-logic approach because the correlation layers restrict the number of possible variable orders. Thus, in its pure form it is limited to small and medium sized circuits. In order to overcome this serious restriction, the concept of *local BDDs* was developed. It is based on the assumption that the influence of spatial correlations between the PIs on a gate's switching activity decreases as the distance of the gate from the circuit's PIs increases, and can finally be neglected. On the other hand, the size of a BDD depends on the number of variables rather than on the number of gates that are covered by the BDD. Therefore, a novel technique, similar to traditional circuit partitioning algorithms, was developed. For each gate, a specific block with a limited number of inputs and a maximum number of gates is searched. The major difference to circuit partitioning is that the blocks are not disjoint but rather build a cover of the circuit. For these blocks the local BDDs are built. As long as some of the inputs of a block are PIs, their spatial correlations are taken into account. All other inputs are assumed to be spatially uncorrelated. Hence, probabilistic-logic simulation is only applied for gates which are close to the PIs. The simulation degenerates to the traditional probabilistic approach as it proceeds to gates that are buried more deeply in the circuit.

The local BDD option was integrated into the probabilistic-logic simulator. In a first set of experiments, resource requirements and accuracy was recorded over the block size. As expected, the resource requirements increase with increasing block size while the deviations decrease. Block size 13 has proven to be a good trade-off between resource requirements and accuracy. The resource requirements remain almost stable until this limit while the accuracy does not much further increase for larger block sizes.

Experiments with many circuits revealed that much larger circuits can be simulated with local BDDs at only a moderate decrease in accuracy, despite the 1GB limit. In fact, all benchmark circuits of the ISCAS-85 and ISCAS-89 set completed successfully regardless the variable order that was used. Further, most simulations were considerably accelerated. Hence, the concept of local BDDs does also defuse the importance of good variable orders for power estimation purposes.

After bipolar, static PMOS, and NMOS technologies have been widely replaced by static CMOS, static current has practically disappeared in digital circuits. Thus, the problem of power consumption was thought to be solved. However, with increasing integration densities and operation frequencies, combined with the advent of complex portable devices, design for low power has regained its importance as the third design goal, beside delay and area consumption. But in contrast to the past, today, dynamic power consumption is dominant by far. The domain of *low power design* can be divided into two major subdomains: power estimation and actual circuit design for low power, the latter including all efforts for power optimization and low power synthesis. In this thesis, specific aspects of both subdomains were treated on different levels.

The starting point for the design aspects was given by a new technology, which is currently being developed at the Institute for Microelectronics Stuttgart (IMS) in the frame of the European project *Hiperlogic*. It combines three recent technological steps: 3D-structure, T-gate transistors, and Silicon on Insulator (SOI). It is intended to use this technology in a semi-custom design environment on the basis of a gate array master for high performance applications at low power consumption. For this purpose, a suitable basic circuit technique had to be found. The following five constraints could be derived from the project goals and the specific structural properties of the *Hiperlogic* technology:

- Equal numbers of PMOS and NMOS transistors.
- Use of minimum width transistors.
- The stacked transistors should share as many contacts as possible.
- Low power consumption.
- Functional robustness against varying gate loads.

From all existing circuit techniques, the structural constraints are best fulfilled by the *double pass logic* (DPL). A complete small library was implemented in DPL on transistor level. As a reference, a similar library was designed in static CMOS. Performance and power consumption of both techniques were compared on the basis of several 32-bit *carry lookahead adders* (CLA), which were implemented with both libraries. In order to more deeply investigate specific DPL properties like the buffer problem and the need for complementary signals, six different DPL versions were implemented. Analog simulations of all CLAs revealed that the performance of the fastest DPL implementations is comparable to static CMOS. However, on the power consumption side even the most economical DPL version

consumes 50% more power than static CMOS. It could be shown by some additional experiments that DPL heavily depends on efficient driver optimization, which is not possible in a gate array technology. Hence, if the potentials of *Hiperlogic* are to be fully exploited, static CMOS should be preferred. However, static CMOS cannot be optimally supported by a pre-fabricated master based on stacked transistor pairs. As a consequence, a standard cell approach promises to yield significantly better results.

The main focus of this thesis was put on power estimation techniques for the logic and gate level. The major source for power consumption in digital, static CMOS circuits are dynamic capacitive and short circuit currents. Both depend on the circuit structure, the operation frequency, some technological parameters, and the switching activities of the circuit gates. While all parameters except the switching activity are static and can be derived from the circuit layout and the technology, the latter does also depend on the typical input patterns that are applied to the circuit. Accordingly, the task of power estimation was more precisely confined on signal activity estimation. Most of today's switching activity estimation methods are based on either pattern simulation or probabilistic simulation. Both approaches suffer from long runtimes if high accuracy is required.

This problem was addressed by a new, set based simulation method. From a mathematical point of view, it classifies between logic and probabilistic simulation. The fundamental idea relies on set based signal representation. Consecutive '1's in the logic signal waveforms are represented by sets. During circuit simulation, all logic operations of the logic gates are then being replaced by according set operations. Since the switching algebra and the set algebra are both implementations of a Boolean algebra, it can be guaranteed that this is always possible. This basic approach was implemented into the simulator *ASSeT*. It has proven to be quite efficient for low active circuits like the combinational part of large sequential circuits. An average speedup of factor 5 over logic simulation could be obtained at the same accuracy. For other circuit types, several optimization algorithms were proposed, which improve the set representation. However, the efficiency of these approaches depends strongly on the statistics of the input patterns. In general, the efficiency decreases with the number of primary inputs of the circuits. Therefore, a method was developed that allows to trade some accuracy for speed. Its main idea relies on partitioning the PIs into smaller groups of correlated input signals. These groups are then separately optimized and arbitrarily recomposed. However, group recomposition introduces a certain estimation error, caused by additional signal correlations. The speedup as well as the estimation error depend upon the group size and the statistical properties of the input streams. For uncorrelated patterns, an average speedup as high as 40 times could be obtained at global and local estimation errors below 8%. For patterns of sequential circuits the speedup is only modest at much higher error rates.

The first implementation of *ASSeT* was based on the zero delay model (ZDM). Thus, effects like hazards and glitches were neglected. They can only be modeled if the gate delays are taken into account. Consequently *ASSeT* was extended by a real delay model (RDM). This extension is based on a combination of set based simulation and a novel event based scheduling algorithm. The traditional vector based signal representation was extended to an array representation, also called *schedule*. In the schedule the columns denote the signal values in

a specific clock cycle, while the rows represent different time offsets during the clock cycles. All operations like event and glitch detection, and distinction between falling and rising edges were implemented through basic logic operators. Thus, they can take full advantage of the set based signal representation. The underlying delay model enables load dependent delays with different values for falling and rising edges. Glitches are filtered out. Their power consumption is neglected. With average speedups of 3.2 and 61 over logic simulation for patterns of combinational and sequential circuits, respectively, the set based approach has proven to be best suited for RDM simulations with sequential patterns. On the other hand, it was found that between 25% and 50% of all transitions are hazards, depending on the circuit type and the delay models. So, gate delay modeling is mandatory for accurate power estimation.

The same event based scheduling algorithm could also be applied to the bitparallel simulator *PAPSAS* that was published in the literature. It was restricted to the ZDM so far and so suffered from the same accuracy problems as *ASSeT* in its first implementation. Experiments with *PAPSAS* on a wide variety of benchmark circuits revealed an average speedup of 5 over bitwise simulation on purely random patterns. While *PAPSAS* cannot profit from special statistics of the input streams, it is 65% faster than *ASSeT* for white noise patterns. The efficiency of *PAPSAS* could be further improved by two optimization techniques: *dynamic package sizing (DPS)* and *schedule compression*. *DPS* is an exact method, while *schedule compression* yields an approximation. The former optimizes the number of test patterns that are simulated in one simulation run while the latter relies on a gate specific reduction of the precision of the time scale. *DPS* could dramatically improve the efficiency on some circuits but had almost no effect on others. However, the simulation times of the latter can be importantly reduced by *schedule compression*. Combining both methods, an average speedup of 131 over bitwise simulation could be obtained with global and local errors below 2% and 4%, respectively. Hence, it could be shown that although gate delay modeling is crucial for accurate power estimation, highly accurate gate delay models play only a minor part. In its current implementation, *ASSeT* is limited to purely combinational circuits. For the ZDM model, extensions to sequential circuits were proposed for bitparallel simulators. In subsequent works it should be checked how these extensions can be incorporated into *ASSeT*. Further improvements could focus on more sophisticated glitch and delay models as well as higher valued logic types.

As a last topic of this work, probabilistic simulation was addressed. The basic idea of probabilistic simulation relies on propagating statistical values through the circuit. Ideally two values per input signal are sufficient for an exact estimation of the average toggle rates: the signal and the switching probability. The problems of accurate probabilistic simulation are introduced by signal correlations. From an algorithmic point of view, they can be classified into three categories: temporal correlations, spatial correlations of the primary inputs, and spatial correlations caused by reconvergent fanout. The latter can be covered by a BDD based algorithm that was first proposed by Najm. The technique of *XOR-BDDs* allows to extend Najm's basic approach to correct handling of concurrently switching inputs. However, spatial correlations between the primary inputs were still an open issue that was addressed in this work with the novel concept of *correlation layers*. Input variables which

belong to the same *correlation group* are adjacently arranged in the *XOR-BDDs*, thus forming a layer of correlated variables. Within such a *correlation layer* the probabilistic operations, which rely on spatial independence, are replaced by path probabilities. Since the processing of the path probabilities is based on logic operations, this approach has been called *probabilistic-logic simulation*. The logic operations can be efficiently implemented through set or bitparallel methods. It could be shown by experiments on several benchmark circuits that the *probabilistic-logic* approach improves the accuracy by at least one order of magnitude compared to previous approaches which neglect spatial correlations.

For any BDD based approach, the availability of good variable orders is crucial in order to keep the CPU and memory requirements within reasonable limits. This problem is aggravated in the *probabilistic-logic* approach since the number of admissible variable orders is restricted by the *correlation layers*. Thus, the basic approach allows only to simulate small and medium sized circuits without exceeding conventional memory limits. Therefore, circuit partitioning was proposed in the past, in order to limit the number of BDD variables. However, traditional partitioning produces large estimation errors at the block borders. These errors can be reduced with the novel concept of *local BDDs*. It relies on the basic assumption that the influence of spatial correlations between the primary inputs decreases as a gate's distance from the circuit inputs increases. Just like circuit partitioning, the *local BDDs* are built for subsets of the circuit. However, these subsets need not to be disjoint. Thus, the *local BDDs* form a circuit cover. A dedicated covering algorithm searches for each gate a specific block with a limited number of input variables which covers a maximum number of gates. For the primary input signals the correlation groups are taken into account; all other variables are supposed to be uncorrelated. Several experiments demonstrated that the concept of *local BDDs* enables the simulation of circuits with more than 10 000 gates without exceeding conventional memory limits and independent of the variable order, while the global and local estimation errors are below 10% on the average. Thus, the *local BDDs* can significantly defuse the problem of variable orders for power estimation purposes. For some 50% of all circuits they could also importantly reduce the simulation times.

The current implementation of the probabilistic-logic simulator is limited to the zero delay model. Thus, future work should include the extension of the delay model. Further accuracy improvements would be possible if correlations between the inputs of the *local BDDs* were more accurately modeled.

Design for low power has become a major issue for mobile and stationary devices. This work addressed low power design aspects under specific technological conditions and power estimation on the logic level. Especially for the latter, the state-of-the-art could be significantly improved in terms of runtime, memory requirements, and accuracy by several novel approaches. An overview over the complete power estimation system that was developed during this work is depicted in figure 7-1.

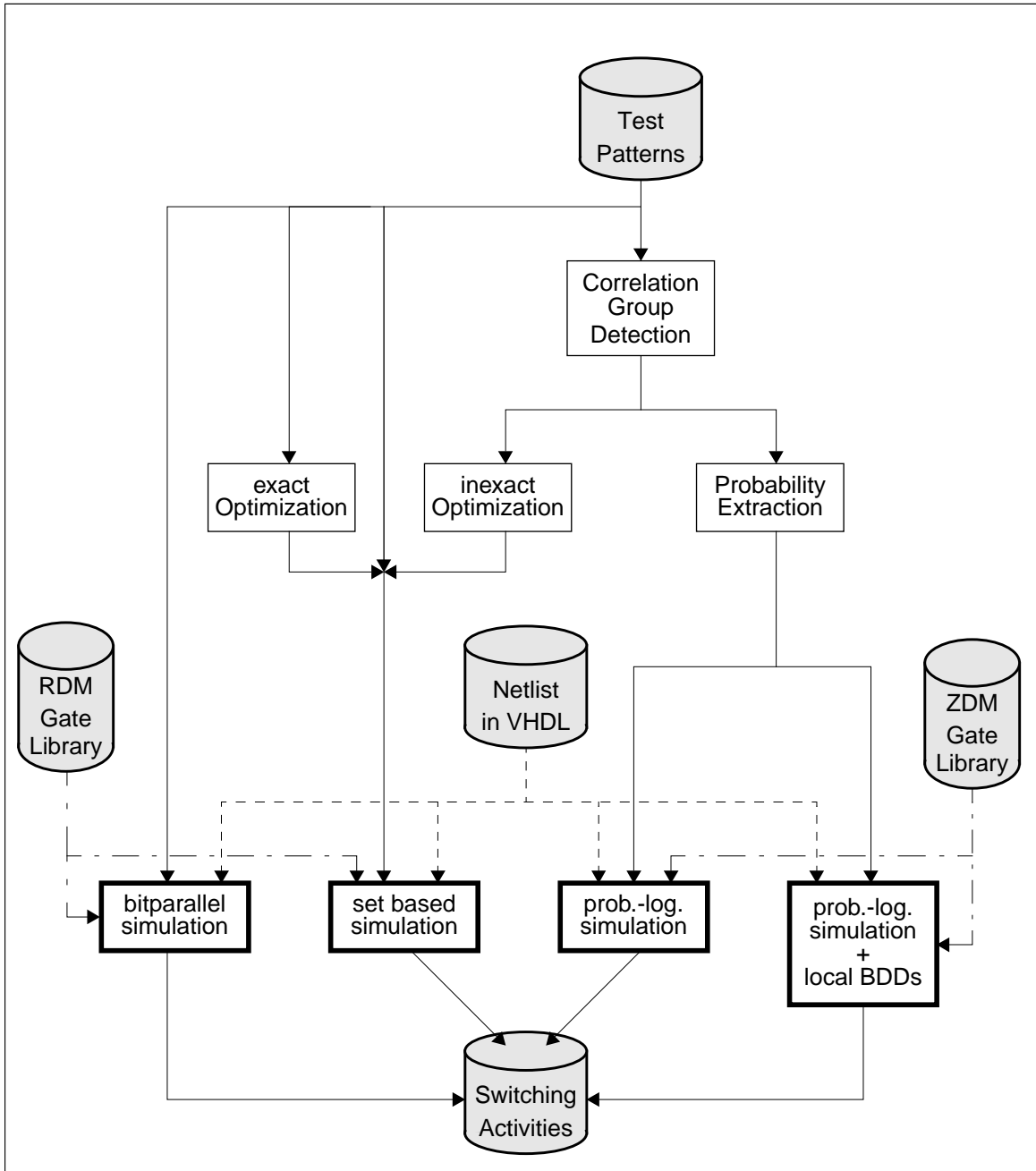


Figure 7-1: The complete power estimation system

Seit mehr als 20 Jahren wird der Fortschritt in der Mikroelektronik getragen von immer kleineren Strukturgrößen und damit ständig wachsender Komplexität von Schaltungen, die auf einem einzigen Chip integriert werden können. Diese Entwicklung wurde schon 1970 von Gordon Moore, Mitbegründer der Intel Corporation, vorhergesehen. In dem als *Moore's Law* bekannt gewordenen Zitat sagte er schon damals [Moor 75]: „Die Anzahl aktiver Transistoren pro Chip und damit die Entwurfskomplexität verdoppelt sich alle 18 Monate“. Dieser Trend spiegelt sich auch in der SIA-Vorhersage über Trends in der Mikroelektronik [SIA 97] wieder, wie Tabelle 8-1 belegt.

Jahr	1997	2001	2006	2012
Minimale Strukturgröße [nm]	250	150	100	50
Maximale Taktfrequenz [MHz]	750	1500	3500	10000
Transistoren pro Chip	11M	40M	200M	1,4G
Versorgungsspannung [V]	1,8-2,5	1,2-1,5	0,9-1,2	0,5-0,6
Verlustleistung (stationäre Geräte) [W]	70	110	160	175
Verlustleistung (mobile Geräte) [W]	1,2	1,7	2,4	3,2

Tabelle 8-1: Trends in der Mikroelektronik

Neben der Entwurfskomplexität tritt heute ein anderer Gesichtspunkt immer mehr in den Vordergrund: die Verlustleistung. Nachdem NMOS, PMOS und bipolare Technologien weitestgehend durch statisches CMOS verdrängt worden waren, hatte man lange Zeit geglaubt, dieses Problem sei endgültig überwunden. Zwar ist die statische Verlustleistung in statischem CMOS in der Tat vernachlässigbar, wachsende Integrationsdichten und Betriebsfrequenzen, vor allem in Verbindung mit dem Wunsch nach immer komplexeren mobilen Geräten, führen inzwischen dazu, daß der Entwurf von Schaltungen mit reduzierter Verlustleistung, sog. *Low Power Design*, seine Bedeutung als drittes Entwurfsziel, neben Fläche und Verzögerungszeit, wiedergewonnen hat. Im Gegensatz zu früheren Technologien wird die Verlustleistung jedoch heute von dynamischen Komponenten dominiert.

Im Bereich des Low Power Design lassen sich zwei wesentliche Forschungsbereiche erkennen: Entwicklung von Methoden zur frühzeitigen Abschätzung der Verlustleistung sowie die Erforschung von Verfahren zum Entwurf von Schaltungen mit geringer Verlustleistung. Letztere schließt alle Bemühungen der Verlustleistungsoptimierung und der Synthese von Schaltungen mit niedriger Verlustleistung mit ein. In der hier vorliegenden Arbeit wurden spezifische Aspekte beider Forschungsbereiche auf unterschiedlichen Entwurfsebenen behandelt.

8.1 Minimierung der Verlustleistung auf der Schaltkreisebene

Den Ausgangspunkt der Verlustleistungsoptimierungen auf Schaltkreisebene wurde durch eine neue Technologie gegeben, die derzeit am Institut für Mikroelektronik Stuttgart (IMS) im Rahmen des europäischen Projekts *Hiperlogic*¹ entwickelt wird. Hiperlogic soll erstmals drei technologische Entwicklungen aus jüngster Zeit miteinander verbinden: 3D-Integration, T-Gate Transistoren und Silicon on Insulator (SOI). Abbildung 8-1 zeigt die Draufsicht und den Querschnitt eines Transistorpaares in der Hiperlogic-Technologie. Deutlich zu

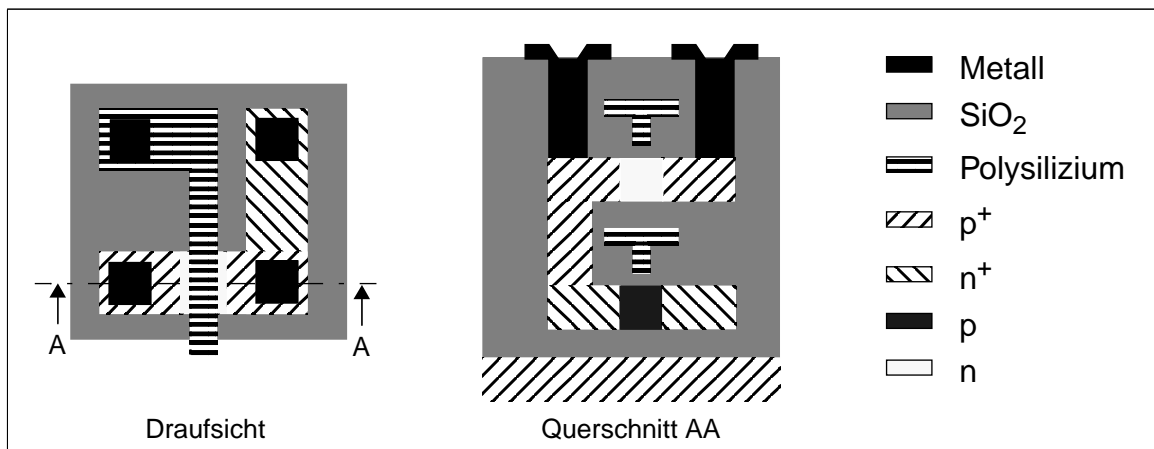


Abbildung 8-1: Die Hiperlogic-Technologie

erkennen sind die übereinandergestapelten PMOS- und NMOS-Transistoren sowie die T-förmigen Transistor Gates. In Verbindung mit der SOI-Technologie führen erhöhte Integrationsdichten, verminderte parasitäre Kapazitäten und erhöhte Treiberleistung der T-Gate Transistoren zu extrem schnellen Schaltungen mit sehr niedriger Verlustleistung. Im Rahmen des Projekts Hiperlogic ist beabsichtigt, diese Technologie in einer Semi-Custom Entwurfsumgebung auf der Basis eines Gate Array Masters einzusetzen, mit dem Ziel, den Entwurf extrem schneller, verlustarmer Schaltungen zu ermöglichen. Eine der Grundlagen, die zum Erreichen dieses Ziels erforderlich sind, ist der Einsatz einer geeigneten Schaltungstechnik, die optimal an die Hiperlogic-Technologie angepaßt ist. Ausgehend von den

1. Hiperlogic wird unterstützt von der Europäischen Kommission unter dem Titel *HIPERLOGIC ES-PRIT IV, Project No. 200023*

Projektzielen und den speziellen strukturellen Eigenschaften der Technologie müssen die folgenden fünf Bedingungen von einer Schaltungstechnik erfüllt werden:

1. Gleiche Anzahl von PMOS- und NMOS-Transistoren.
2. Einsatz von Transistoren minimaler Breite.
3. Die zwei übereinanderliegenden Transistoren sollten so viele Kontakte wie möglich gemeinsam nutzen.
4. Niedrige Verlustleistung.
5. Funktionale Robustheit gegenüber unterschiedlicher Gatterlast.

Die strukturellen Bedingungen 1. bis 3. werden am besten von der Double Pass Logic (DPL) erfüllt. Vor allem die Verlustleistung von DPL Schaltungen wird jedoch in der existierenden Literatur unterschiedlich beurteilt [KoBa 95] bzw. [Zimm 97]. Um die Bedingungen 4. und 5. genauer zu untersuchen, wurde eine kleine Zellbibliothek in DPL auf Transistorebene entworfen. Sie enthält alle für den Entwurf beliebiger Schaltungen notwendigen kombinatorischen und sequentiellen Gatter. Alle Gatter wurden auf der Basis einer einzigen Grundzelle, einem 2:1 Multiplexer, aufgebaut. Als Referenz hierzu wurde eine ähnliche Bibliothek in statischem CMOS entworfen, deren Gatter die gleichen Grundfunktionen wie die der DPL-Bibliothek realisieren. Die typischen Verzögerungszeiten und Verlustleistungen beider Schaltungstechniken wurden anhand von 32-Bit *Carry-Look-Ahead Addierern* verglichen, die auf der Basis beider Zellbibliotheken implementiert wurden. Um DPL-spezifische Besonderheiten, wie das Pufferproblem oder die Erzeugung komplementärer Signale, genauer zu erforschen, wurden sechs verschiedene Variationen des DPL-Addierers aufgebaut. Die Verzögerungszeiten sowie der Energieverbrauch pro Operation wurden aufgrund von Analogsimulationen auf Transistorebene ermittelt. Hierbei stellte sich heraus, daß die schnellsten DPL-Varianten zwar ähnliche Verzögerungszeiten aufweisen wie statisches CMOS, ihr Energieverbrauch, und damit ihre Verlustleistung, ist jedoch um mindestens 50% höher. Durch weitere Experimente konnte gezeigt werden, daß DPL-Schaltungen durch Treiberoptimierungen wesentlich optimiert werden können. Da dies in der Hiperlogic Technologie aus verschiedenen Gründen nicht möglich ist, sollte statischem CMOS der Vorzug gegeben werden, falls man die Vorteile der Hiperlogic Technologie voll ausschöpfen möchte. Statisches CMOS kann jedoch auf einem vorgefertigten Master mit übereinander gestapelten Transistoren nicht optimal realisiert werden, da sich zwei übereinander liegende Transistoren nur ihren Gateanschluß teilen können. Dies bedeutet, daß pro Transistorpaar fünf Kontakte benötigt werden. Damit ist die platzsparende Anordnung von Abbildung 8-1 nicht mehr möglich. Unter diesem Gesichtspunkt sollte deshalb von Seiten des IMS darüber nachgedacht werden, ob ein Ansatz mit Standardzellen nicht erfolgversprechender wäre. Dies wurde auch schon in [Abou 98] angedeutet.

8.2 Abschätzung der Verlustleistung

Das Hauptaugenmerk wurde in dieser Arbeit auf die Entwicklung neuartiger Methoden zur Abschätzung der Leistungsaufnahme in digitalen Schaltungen gelegt. In einem hierarchisch gegliederten Entwurf werden auf höheren Ebenen Entwurfsentscheidungen getroffen, die

den Verlauf der Entwicklung auf den unteren Ebenen vorausbestimmen. Entwurfsfehler oder ungünstige Entscheidungen auf höheren Entwurfsebenen lassen sich deshalb auf den nachfolgenden Ebenen nur sehr eingeschränkt korrigieren und erfordern meist aufwendige und teure Nachbesserungen. Aus diesem Grund ist es wichtig, dem Entwerfer rechnerbasierte Hilfsmittel zur Hand zu geben, die ihn zuverlässig bei der Entscheidungsfindung unterstützen. Die in dieser Arbeit vorgestellten Methoden zur Abschätzung der Leistungsaufnahme konzentrieren sich deshalb auf die Logik- und Gatterebene und erlauben so, verschiedene Schaltungsalternativen zu untersuchen, lange bevor die Maskendaten oder gar die fertigen Schaltungen zur Verfügung stehen. Kapazitive Ströme und Kurzschlußströme stellen die wichtigsten Ursachen dar für die Entstehung von Verlustleistung in digitalen, statischen CMOS Schaltungen. Beide hängen ab von der Schaltungsstruktur, der Betriebsfrequenz der Schaltung, einigen technologischen Parametern und der Schaltaktivität der einzelnen Gatter innerhalb der Schaltung. Alle Parameter außer der Schaltaktivität sind statischer Natur und können entweder von den Technologiedaten oder der Schaltung selbst abgeleitet werden. Die Schaltaktivität hängt jedoch neben der Schaltungsstruktur auch von den Testmustern ab, mit denen die Schaltung typischerweise betrieben wird. Sie ist also nicht nur eine Eigenschaft der Schaltung selbst, sondern wird auch von der Umgebung der Schaltung beeinflußt und ist damit eine statistische Größe. Somit läßt sich die Leistungsabschätzung auf Gatterebene in zwei Bereiche untergliedern: eine technologische Komponente, die sich hauptsächlich mit der Modellierung der auftretenden Ströme befaßt, und eine technologieunabhängige Komponente, deren Ziel es ist, die Schaltaktivitäten innerhalb einer Schaltung zu bestimmen. Der Schwerpunkt der hier vorliegenden Arbeit beschränkt sich auf letzteres.

Die meisten, heute bekannten Methoden zur Abschätzung der Schaltaktivität beruhen entweder auf expliziter Simulation vorgegebener Testmuster oder auf probabilistischer Simulation. Für beide Ansätze stellt sich jedoch das Problem langer Rechenzeiten, falls hohe Anforderungen an die Genauigkeit gestellt werden. Um die Rechenzeiten zu verkürzen wurde deshalb ein neuartiges, mengenbasiertes Simulationsverfahren vorgestellt. Aus mathematischer Sicht kann dieses Verfahren zwischen logischer und probabilistischer Simulation eingeordnet werden. Aufeinanderfolgende Einsen in der logischen Signaldarstellung werden hierbei durch Mengen repräsentiert (Abbildung 8-2). Während der Simula-

Takt	1	2	3	4	5	6	7	8	9	...							
Logisches Signal	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	1	1
Mengendarstellung		2-3					7-11					14-17					

Abbildung 8-2: Mengenbasierte Signaldarstellung

tion werden dann alle logischen Operationen durch entsprechende Mengenoperationen ersetzt. Da sowohl die Mengenlehre als auch die Schaltalgebra spezielle Implementierungen der Booleschen Algebra darstellen, ist sichergestellt, daß dieser Übergang von der Logik- zur mengenbasierten Simulation stets möglich ist. Die Transformationen der logischen Grund-

funktionen in ihre Mengenäquivalente sind in Tabelle 8-2 zusammengestellt. Daraus können beliebige logische Operationen bzw. Mengenoperationen zusammengesetzt werden.

Logische Operation		Mengenoperation	
Operation	Symbol	Symbol	Operation
UND	$x \cdot y$	$x \cap y$	Schnittmenge
ODER	$x + y$	$x \cup y$	Vereinigung
NICHT	\bar{y}	\bar{y}	Inversion

Tabelle 8-2: Zusammenhang zwischen logischen Operationen und Mengenoperationen

Die grundlegende Idee der mengenbasierten Simulation wurde prototypisch in einen Simulator namens *ASSeT (Activity Simulator based on Set Theory)* implementiert. Experimente mit einer Vielzahl von Schaltungen aus den ISCAS- bzw. LGSynth-Benchmark Sammlungen ([Brgl 85], [Brgl 89] bzw. [LGSY 91]) zeigten, daß sich ASSeT besonders für die Simulation von Schaltungen mit niedriger Schaltaktivität eignet wie z.B. der kombinatorische Teil großer sequentieller Schaltungen. Hier konnten im Vergleich zur Logiksimulation die Simulationszeiten auf ein fünftel reduziert werden bei gleicher Genauigkeit. Für andere Schaltungen wurden verschiedene Heuristiken vorgestellt, um die Mengendarstellung zu optimieren. Sie beruhen auf einer Umordnung der Signale, um größere Teilmengen zu erzielen. Da eine Umordnung der Eingangssignale ganz erheblich die Schaltaktivität beeinflussen kann, wurde das sogenannte *physikalische Signal* eingeführt. Es besteht aus einem originalen Signal und dem um einen Takt verschobenen originalen Signal. Dadurch stehen in jedem Takt neben den Informationen über die Signalwerte auch die Informationen über die Signalaktivitäten zur Verfügung. Die Testmuster können damit umgeordnet werden, ohne das Simulationsergebnis zu beeinflussen. Die Effizienz dieser Ansätze ist jedoch stark abhängig von den statistischen Eigenschaften der zu optimierenden Signale. Im allgemeinen nimmt die Effizienz ab, je mehr Primäreingänge eine Schaltung hat. Deshalb wurde eine Methode entwickelt, die einen Kompromiß zwischen Genauigkeit und Simulationszeit erlaubt. Sie beruht auf einer Partitionierung der Eingangssignale in Gruppen korrelierter Signale, sogenannter *Korrelationsgruppen*. Zum Auffinden geeigneter Korrelationsgruppen wurde ein heuristischer Algorithmus vorgestellt. Die Korrelationsgruppen werden dann getrennt optimiert und danach wieder willkürlich zu kompletten Testvektoren zusammengesetzt. Während der Optimierungen bleiben zwar alle Korrelationen innerhalb der Korrelationsgruppen korrekt erhalten, durch das Zusammensetzen werden jedoch Korrelationen zwischen Signalen aus unterschiedlichen Korrelationsgruppen erzeugt, was die Simulationsergebnisse verfälscht. Es wurde experimentell gezeigt, daß sowohl die Beschleunigung der Simulation als auch der Schätzfehler von der Gruppengröße und den statistischen Eigenschaften der Eingangssignale abhängen. Gerade für Schaltungen mit unkorrelierten Signalen, die dem reinen mengenbasierten Ansatz nur schlecht zugänglich waren, konnten erhebliche Beschleunigungen um den Faktor 40 erreicht werden, wobei die globalen und lokalen Fehler unter 8% blieben. Für Testmuster sequentieller Schaltungen ist dieser Ansatz

jedoch weniger geeignet. Die erzielte Beschleunigung ist recht gering und die Fehlerraten sind um den Faktor fünf bis zehn höher.

Die erste Implementierung von ASSeT beschränkte sich der Einfachheit halber auf ein verzögerungsfreies Gattermodell (engl. *Zero Delay Model, ZDM*). Dadurch konnten Effekte wie Hasards und Glitches, die durch die Gatterverzögerungen verursacht werden, nicht berücksichtigt werden. Da gerade Hasards einen sehr hohen Anteil der gesamten Signalaktivität ausmachen können, hat dies natürlich negative Auswirkungen auf die Genauigkeit des Simulationsergebnisses. Deshalb wurde ASSeT so erweitert, daß beliebige, realitätsnähere Gatterverzögerungen (engl. *Real Delay Model, RDM*) während der Simulation berücksichtigt werden können. Diese Erweiterung beruht auf einem neuartigen Algorithmus zur ereignisbasierten Ablaufplanung. Die herkömmliche, vektorbasierte Signaldarstellung wurde zu einer zweidimensionalen Matrix, dem sogenannten *Schedule*, erweitert. Abbildung 8-3

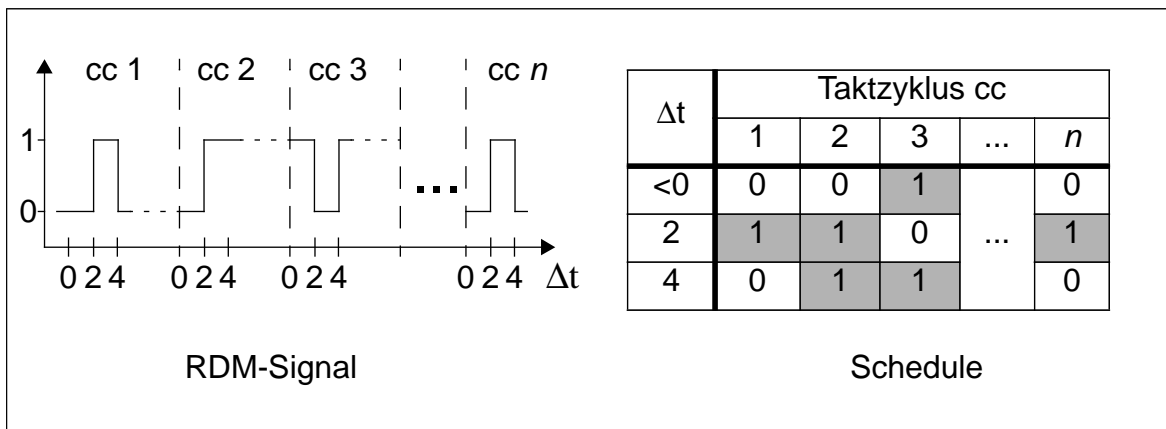


Abbildung 8-3: RDM-Signal mit *Schedule*

zeigt ein RDM-Signal mit dazugehörigem *Schedule*. Die Spalten repräsentieren die logischen Signalwerte während eines Taktzyklus *cc*, während die Zeilen die Signalwerte zu einem bestimmten Zeitversatz Δt seit Beginn der Taktzyklen darstellen. Genau wie bei der ZDM-Simulation können nun die Zeilen des *Schedule* durch Mengen dargestellt werden. Alle Operationen wie Ereignis- und Glitcherkennung sowie die Differenzierung zwischen fallenden und steigenden Flanken wurde mittels logischer Operatoren realisiert. Damit kommen die Vorteile der Mengendarstellung voll zum Tragen. Das der prototypischen Implementierung zugrunde liegende Verzögerungsmodell erlaubt lastabhängige Verzögerungen mit unterschiedlichen Werten für fallende und steigende Flanken. Glitches werden ausgefiltert, ihre Verlustleistung wird vernachlässigt. Experimente mit den schon zuvor genannten Benchmarkschaltungen ergaben mittlere Beschleunigungen um die Faktoren 3,2 für unkorrelierte Testmuster bzw. 61 für Testmuster sequentieller Schaltungen gegenüber einer Logiksimulation. Damit ist die Beschleunigung zum Teil wesentlich höher als bei der ZDM-Simulation, aber auch noch stärker abhängig von der Statistik der Eingangssignale. Gleichzeitig stellte sich heraus, daß im Mittel je nach Schaltungstyp und Verzögerungsmodell ca. 25% bis 50% aller Transitionen durch Hasards verursacht werden. Hieraus folgt,

daß die Laufzeitmodellierung unverzichtbar ist zur zuverlässigen Abschätzung der Verlustleistung auf der Gatterebene.

Der gleiche, ereignisgesteuerte Algorithmus zur Ablaufsteuerung konnte auch auf den bitparallelen Simulator *PAPSAS* angewendet werden, der in [Schn 95] veröffentlicht wurde. *PAPSAS* ist ein Logiksimulator. Er beruht auf der Idee, die gesamte Wortbreite w der Maschine auszunutzen, auf der die Logiksimulation ausgeführt wird, indem stets w Testvektoren gleichzeitig simuliert werden. Genau wie *ASSeT* in seiner ersten Implementierung war *PAPSAS* bisher auf verzögerungsfreie Gattermodelle beschränkt. Prinzipbedingt kann *PAPSAS* zwar nicht von speziellen statistischen Eigenschaften der Eingangssignale profitieren, aber gerade bei rein zufälligen, unkorrelierten Testmustern konnte eine durchschnittliche Beschleunigung der Simulation um den Faktor fünf gegenüber einer bitweisen Logiksimulation nachgewiesen werden. Damit ist *PAPSAS* bei dieser Signalklasse etwa 65% schneller als *ASSeT*. Die Effizienz von *PAPSAS* konnte durch zwei Optimierungsmaßnahmen weiter verbessert werden: *Dynamic Package Sizing (DPS)* und *Schedule Compression*. *DPS* ist eine exakte Methode, die die Anzahl der in einem Schaltungsdurchlauf simulierten Testvektorwörter optimiert. Bei einigen Schaltungen konnte *DPS* die Simulationszeiten erheblich verkürzen, während bei anderen sein Einfluß praktisch vernachlässigbar ist. Gerade bei solchen Schaltungen brachte jedoch *Schedule Compression* erhebliche Verbesserungen. Es basiert darauf, die Auflösung der Zeitachse gatterspezifisch zu reduzieren. Dadurch werden die Schedules kompakter, was zu kürzeren Bearbeitungszeiten führt aber unter Umständen auch das Simulationsergebnis leicht verfälscht. Durch die Kombination von *DPS* und *Schedule Compression* wurde eine relativ homogene, mittlere Beschleunigung um den Faktor 131 gegenüber einer bitweisen Logiksimulation erreicht, wobei die globalen und lokalen Schätzfehler unter 2% bzw. 4% blieben. Damit konnte gezeigt werden, daß zur Ermittlung der Signalaktivität Gatterverzögerungen zwar nicht vernachlässigbar sind, die Genauigkeit der Verzögerungsmodelle jedoch eine untergeordnete Rolle spielt.

In ihren momentanen Implementierungen sind sowohl *ASSeT* als auch *PAPSAS* auf rein kombinatorische Schaltungen beschränkt. Anschließende Arbeiten könnten eine Erweiterung auf sequentielle Schaltungen betreffen. Verfahren hierzu wurden für *PAPSAS* unter Vernachlässigung der Verzögerungszeiten schon in [Goud 91] vorgeschlagen. Weitere mögliche Verbesserungen betreffen das Verzögerungs- und das Glitchmodell sowie die Implementierung mehrwertiger Logik.

Als letztes Thema dieser Arbeit wurden probabilistische Simulationsverfahren behandelt. Probabilistische Verfahren bieten eine interessante Alternative zu Verfahren, die auf der expliziten Simulation von Testmustern basieren. Im Idealfall genügt es für eine exakte Ermittlung der Schaltaktivitäten, zwei Werte pro Eingangssignal durch die Schaltung zu propagieren: die statische Signalwahrscheinlichkeit und die Schaltaktivität. Die genaue Kenntnis der Testmuster ist nicht notwendig, was in manchen Fällen von Vorteil sein kann. Damit ist die probabilistische Simulation auf den ersten Blick wesentlich schneller als testmusterbasierte Simulationsverfahren. Probabilistische Methoden werden allerdings sehr aufwendig, sobald Signalkorrelationen in die Berechnungen mit einbezogen werden sollen. Von algorithmischer Seite her gesehen, können drei Arten von Korrelationen unterschieden werden: zeitliche Korrelationen, räumlich korrelierte primäre Eingangssignale und räumli-

che Korrelationen, die durch rekonvergentes Fanout hervorgerufen werden. Zeitliche Korrelationen werden durch die Signalaktivität ausgedrückt. Durch rekonvergentes Fanout verursachte Korrelationen können mit dem von Najm vorgestellten, BDD basierten Algorithmus [Najm 93] korrekt behandelt werden. Hierbei werden alle Funktionen, die zur Berechnung der Schaltaktivitäten benötigt werden, auf BDDs mit ausschließlich primären Eingangsvariablen zurückgeführt. Diese BDDs werden auch *globale BDDs* genannt. Mittels *XOR-BDDs* [Schn 94], [Schl 95] kann Najms grundlegender Algorithmus so erweitert werden, daß auch gleichzeitig schaltende Signale korrekt miteinbezogen werden können. Das Problem räumlich korrelierter primärer Eingangssignale konnte bislang allerdings mit Najms Algorithmus nicht zufriedenstellend gelöst werden. Zu diesem Zweck wurde im Rahmen dieser Arbeit das neuartige Konzept der *Korrelationsschichten* entwickelt. Ähnlich wie bei der optimierten Mengensimulation werden Gruppen korrelierter Eingangsvariablen gebildet. Variablen, die zur gleichen Korrelationsgruppe gehören, werden in den XOR-BDDs benachbart angeordnet. Die Korrelationsgruppen erscheinen damit im BDD als

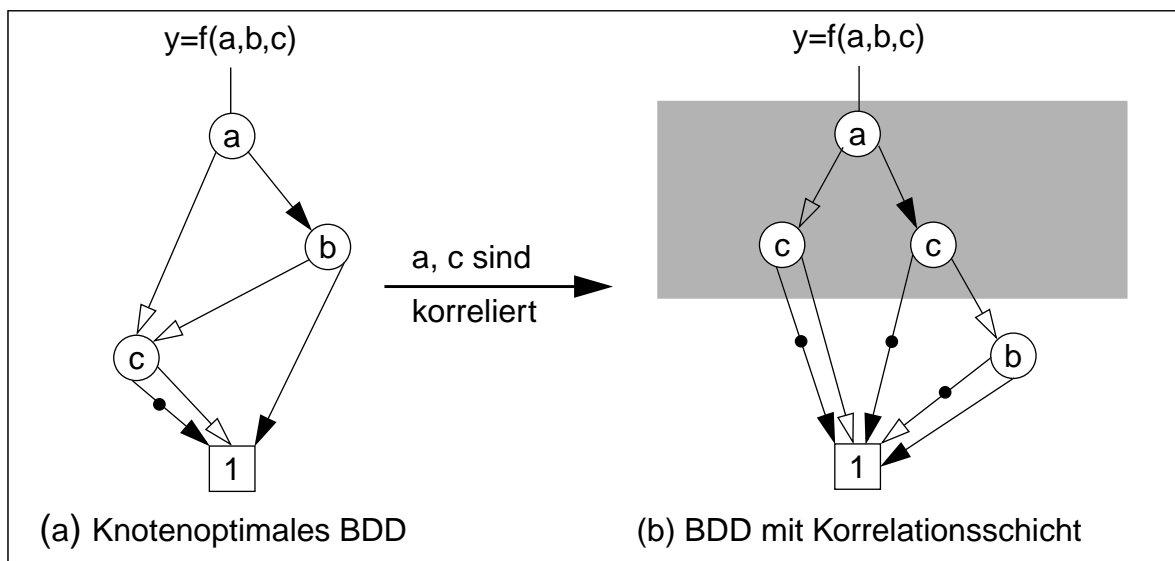


Abbildung 8-4: BDD für $y = \bar{c} + ab$ ohne und mit Korrelationsschicht

Schichten korrelierter Variablen (Abbildung 8-4). Während an den Übergängen zwischen den Korrelationsgruppen herkömmliche, probabilistische Verfahren angewandt werden, werden diese innerhalb der Korrelationsgruppen durch Pfadwahrscheinlichkeiten ersetzt. Zur inkrementellen Auswertung von BDDs mit Korrelationsschichten wurde ein auf der Matrizenrechnung beruhendes Verfahren entwickelt. Da bei der Auswertung der Matrizen logische Operationen eine wichtige Rolle spielen, wurde das Simulationsverfahren als *probabilistisch-logische Simulation* bezeichnet. Die logischen Operationen können effizient mittels mengenbasierter oder bitparalleler Methoden implementiert werden. Damit kann die probabilistisch-logische Simulation auch als Verfeinerung der korrelationsgruppenorientierten Mengensimulation betrachtet werden. Anhand von ausführlichen Experimenten wurde zunächst gezeigt, daß Najms ursprünglicher Ansatz selbst bei räumlich unkorrelierten Variablen zu Schätzfehlern von weit über 100% führen kann. Deshalb wurden für die probabilistisch-logische Simulation ausschließlich XOR-BDDs eingesetzt. Betrachtet man

Anwendungen mit räumlich korrelierten Primäreingängen, so stellt sich heraus, daß mittels probabilistisch-logischer Simulation die Abweichung vom Referenzergebnis um den Faktor fünf bis zehn reduziert werden kann gegenüber herkömmlichen Verfahren, die solche Korrelationen prinzipbedingt vernachlässigen. So wurden beispielsweise für Signale sequentieller Schaltungen die mittleren globalen und lokalen Abweichungen vom Referenzergebnis von 20% auf 5% bzw. von 40% auf 8% reduziert.

Für BDD-basierte Anwendungen ist die Variablenordnung in den BDDs von zentraler Bedeutung, was die CPU- und Speicheranforderungen angeht. Dieses Problem wird durch den probabilistisch-logischen Ansatz noch verstärkt, da die Anzahl zulässiger Variablenordnungen durch die Korrelationsschichten eingeschränkt wird. Damit beschränkt sich die grundlegende probabilistisch-logische Methode auf die Simulation kleiner und mittlerer Schaltungen. Für große Schaltungen werden meist die heute üblicherweise verfügbaren Hauptspeichergrößen von ca. 1 GB überschritten. Deshalb wurden in der Vergangenheit Algorithmen zur Schaltungspartitionierung vorgeschlagen. Damit kann die Anzahl der BDD-Variablen und damit die BDD-Größe begrenzt werden. Herkömmliche Schaltungspartitionierung hat jedoch evtl. große Schätzfehler zur Folge, da mit den heute verfügbaren Methoden die räumlichen Korrelationen an den Partitions Grenzen vernachlässigt werden müssen. Diese Fehler können mit dem neuartigen Ansatz der *lokalen BDDs* verringert werden. Er beruht auf der Annahme, daß der Einfluß der Korrelationen zwischen den primären Eingangssignalen auf die Schaltaktivität eines Gatters abnimmt, je weiter dieses Gatter von den Primäreingängen der Schaltung entfernt ist, wohingegen der Einfluß durch rekonvergentes Fanout zunimmt. Um dieser Annahme Rechnung zu tragen, werden lokale BDDs aus Untermengen der zu betrachtenden Schaltung gebildet. Anders als bei der Partitionierung können sich diese Untermengen jedoch überlappen, sie bilden also eine Schaltungsüberdeckung. Der Überdeckungsalgorithmus, der im Rahmen dieser Arbeit entwickelt wurde, sucht für jedes Gatter einen eigenen Block mit einer vorgegebenen Zahl an Eingangsvariablen, der eine maximale Anzahl von Gattern überdeckt. Sollte ein Block primäre Eingangsvariablen überdecken, so werden deren Korrelationsgruppen mit einbezogen, andere Variablen werden hingegen als statistisch unabhängig betrachtet. D.h., nahe den Primäreingängen einer Schaltung wird probabilistisch-logisch simuliert. Die probabilistisch-logische Simulation wird allmählich durch eine rein probabilistische Simulation ersetzt, je tiefer die Simulation in die Schaltung eindringt. Mittels umfangreicher Experimente konnte demonstriert werden, daß das Konzept der lokalen BDDs den Anwendungsbereich der probabilistisch-logischen Simulation auf Schaltungen mit mehr als 10.000 Gattern erweitert, ohne den verfügbaren Hauptspeicher von 1 GB zu überschreiten. Gleichzeitig bleiben die globalen und lokalen Abweichung vom Ergebnis der Logiksimulation im Mittel unter 10%. Damit konnte für BDD-Anwendungen im Bereich der Signalaktivitätsbestimmung das Problem der Ermittlung guter Variablenordnungen bedeutend entschärft werden. Gleichzeitig führte der Einsatz lokaler BDDs bei ca. 50% der betrachteten Schaltungen zu einer signifikanten Reduzierung der Laufzeit der Simulationen.

Die momentane Implementierung der probabilistisch-logischen Simulation beschränkt sich auf ein verzögerungsfreies Gattermodell (ZDM). Es ist jedoch zu erwarten, daß sich die Methode der *Probability Waveforms* [Najm 90] problemlos auf die probabilistisch-logische

Simulation anwenden läßt. Die Genauigkeit der Simulation mit lokalen BDDs ließe sich weiter erhöhen, wenn Korrelationen zwischen den Eingangsvariablen der lokalen BDDs genauer modelliert werden könnten. Hierbei ist jedoch noch nicht ganz klar, wie diese in die Simulation miteingebunden werden können, so daß hier noch weiterer Forschungsbedarf besteht.

Die Verlustleistung ist heute ein wichtiger Gesichtspunkt beim Entwurf von mobilen und stationären digitalen Systemen. Die in dieser Arbeit vorgestellten Methoden befassen sich mit Entwurfsaspekten unter speziellen technologischen Randbedingungen sowie mit der Abschätzung der Verlustleistung auf der Logikebene. Besonders im Bereich der Verlustleistungsbestimmung konnten bedeutende Fortschritte gegenüber dem aktuellen Stand der Technik erzielt werden was die Laufzeit- und die Speicheranforderungen sowie die Genauigkeit angeht. Abbildung 8-5 gibt einen abschließenden Überblick über das komplette, in dieser Arbeit entstandene System zur Abschätzung der Signalaktivität und damit der Verlustleistung.

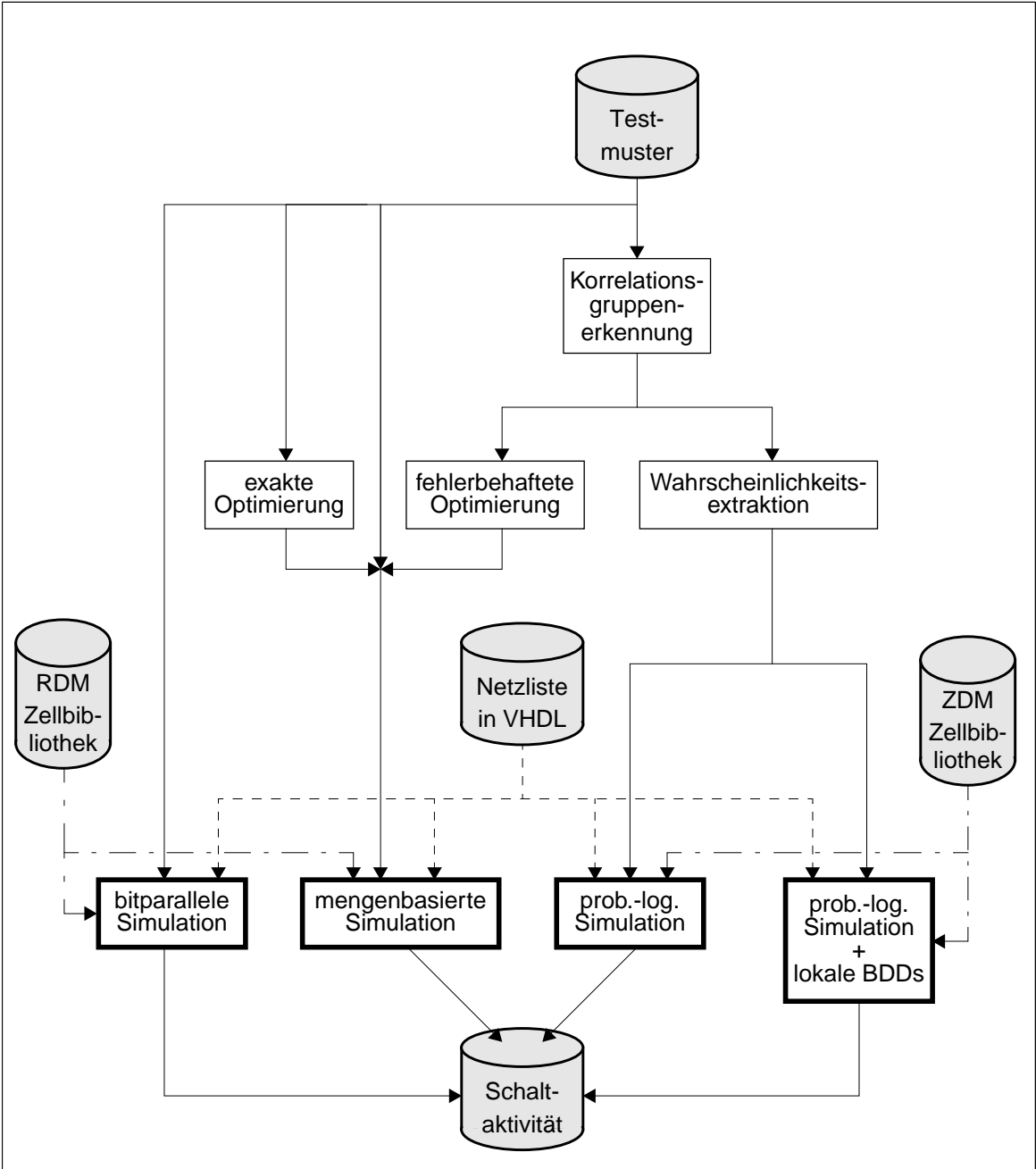


Abbildung 8-5: Das komplette System zur Ermittlung der Schaltaktivität

Hiperlogic Transistor Parameters

Appendix

A

This appendix gives the most recent transistor parameters for the Hiperlogic technology in the SPICE format. They were used for all analog simulations in chapter 3.

NMOS Transistor

```
.MODEL nmos NMOS LEVEL = 3
+ TOX = 5E-9          NSUB = 1.55596E17   GAMMA = 0.65815
+ PHI = 0.837411     VTO = 0.447166     DELTA = 0
+ UO = 130.566       ETA = 0           THETA = 0.49758
+ VMAX = 1.41532E5   KAPPA = 0.864159   JS = 100u
+ NFS = 1.4086E12    XJ = 1.90546E-9    LD = 0
+ CGDO = 0.3E-9     CGSO = 0.3E-9     CGBO = 0.00
+ CJ = 17E-4        PB = 0.8           MJ = 0.5
+ FC = 0.5          CJSW = 2.5E-10    MJSW = 0.3
+ RSH = 630
```

PMOS Transistor

```
.MODEL pmos PMOS LEVEL = 3
+ TOX = 5E-9          NSUB = 6.9346e16   GAMMA = 0.494853
+ PHI = 1            VTO = -0.34         DELTA = 0
+ UO = 36.97        ETA = 1.770645e-4   THETA = 0.5239953
+ VMAX = 5.178528E4 KAPPA = 1.6525353  RD = 0
+ RS = 0            RSH = 576           JS = 100u
+ NFS = 9.969E11    XJ = 1.1e-9        LD = 0
+ WD = 0            CGDO = 3E-10       CGSO = 3E-10
+ CGBO = 0.0        CJ = 6E-4         PB = 0.8
+ MJ = 0.5          FC = 0.5           CJSW = 6E-10
+ MJSW = 0.3
```


The Benchmark Circuits

Appendix

B

For all experiments in this work benchmark circuits from [Brgl 85], [Brgl 89] and [LGSY 91] were used. They are referred to as ISCAS-85, ISCAS-89 and LGSYNTH, respectively. Generic netlists of all circuits are available via the internet at [BENCH]. The ISCAS-85 circuits are purely combinational, while circuits from ISCAS-89 are sequential. In the following tables, *Generic*, *HL* and *GFN120* denote the number of gates in the original, generic description, when mapped on the Hiperlogic cell library, and when mapped on the GFN120 library [IMS 96], respectively.

ISCAS-85

Circuit	Function	Inputs	Outputs	Gates		
				Generic	HL	GFN120
c17	unknown	6	2	6	7	6
c432	Priority Decoder	37	7	160	126	163
c499	ECAT	42	32	202	270	202
c880	ALU + Control	61	26	383	287	383
c1355	ECAT	42	32	514	300	514
c1908	ECAT	34	25	880	245	880
c2670	ALU + Control	234	140	997	400	997
c3540	ALU + Control	51	22	1446	791	1446
c5315	ALU + Selector	179	123	1994	1203	1996
c6288	16-Bit Multiplier	33	32	2461	2339	2461
c7552	ALU + Control	208	108	2978	1319	2978

Table B-1: Statistics of the ISCAS-85 benchmark circuits

ISCAS-89

Circuit	Inputs	Flip Flops	Inputs + Flip Flops	Outputs	Gates	
					Generic	GFN120
s27	4	3	7	1	17	14
s298	3	14	17	6	137	123
s344	9	15	24	11	179	164
s382	3	21	24	6	183	162
s386	7	6	13	7	167	161
s820	18	5	23	19	298	293
s1196	14	18	32	14	551	553
s1238	14	18	32	14	530	512
s1488	8	6	14	19	661	655
s1494	8	6	14	19	655	649
s5378	35	179	214	49	2960	2781
s9234	19	228	247	22	5812	5584
s13207	31	669	700	121	8593	7924
s15850	14	597	611	87	10310	9713
s35932	35	1728	1763	320	17796	16068

Table B-2: Statistics of the ISCAS-89 benchmark circuits

LGsynth91

Circuit	Inputs	Outputs	Gates	
			Generic	HL
misex2	25	18	36	69
f51m	8	8	16	112
bw	5	25	56	135
alu2	10	6	59	270
misex3	14	14	73	300
alu4	15	8	112	551

Table B-3: Statistics of the LGSYNTH benchmark circuits

All the results that were graphically presented in the main text will be repeated here in a tabular form. In each section the corresponding section in the main text will be specified in italics.

All test patterns were generated with the test vector generators *tvgen* from [Dall 98] or *kktvgen* from [KapD 98]. They allow the generation of pseudo random test vectors with a variety of statistical properties. All the experiments were executed on a Sun Ultra Sparc 2, 300 MHz, 1.4 GB main memory, running Sun Solaris 2.6.

C.1 Power Minimization on the Circuit Level

Related to chapter 3.

C.1.1 aCPL with $V_{thp}=0V$ and $V_{thp}=0.5V$

Related to section 3.2.2.

V_{thp} [V]	Delay [ns]	Energy per Clock Cycle [pJ]
0.0	0.44	0.86
0.5	1.40	1.10

Table C-1: CPL with $V_{thp}=0V$ and $V_{thp}=0.5V$ for the pass transistors

The threshold voltage of the inverters was chosen to $V_{thp}=0.5V$, the inputs of the NAND-gates were driven by inverters.

C.1.2 CPL and DPL versus CMOS

Related to section 3.2.3.

Circuit	Delay [ps]			Energy per Clock Cycle [fJ]		
	CMOS	CPL	DPL	CMOS	CPL	DPL
nor4	700	298	220	45	435	106
nand4	534	243	195	43	490	101
xor4	525	462	200	206	1932	144

Table C-2: CPL and DPL versus CMOS

C.1.3 32-Bit CLAs

Related to sections 3.3.3 and 3.4.

Circuit	Transistors	Delay [ns]	Energy per Clock Cycle [pJ]
cadder	2008	5.46	14.3
dadder	2648	5.79	22.7
dsadder	3340	5.62	25.5
dbadder	3426	8.00	31.4
ddadder	4748	7.91	38.4
ddbadder	5176	8.44	41.6
madder	3548	6.47	27.6

Table C-3: Results from the 32-bit CLAs

C.1.4 4-Bit CLAs with Varying Transistor Sizes

Related to section 3.4.

$\frac{W_{\text{CMOS}}}{W_{\text{DPL}}}$	Delay [ns]		Energy per Clock Cycle [pJ]	
	CMOS	DPL	CMOS	DPL
1	2.21	2.54	1.73	3.10
2	2.08	2.10	3.11	3.83
3	2.04	2.00	4.48	4.65
4	2.02	1.99	5.85	5.55
5	2.00	2.02	7.23	6.42

Table C-4: 4-bit CLAs with varying transistor sizes

C.2 Zero Delay Simulation

Related to chapter 4.

C.2.1 Optimization Results

Related to section 4.5.4.

Uncorrelated Patterns

PIs	NoOpt	NoDup		PS		GPS		CS	
	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks
4	39 802	0.03	516	0.04	246	0.04	133	0.04	147
8	80 067	0.08	69 055	7.19	39 772	1.89	13 959	9.62	10 678
16	160 206	0.40	160 206	20.77	111 649	12.52	75 117	50.28	57 337
25	289 648	0.58	249 260	33.68	189 122	27.37	150 934	58.39	119 984
32	319 955	0.71	319 955	73.83	250 972	48.55	212 232	96.49	172 094

Table C-5: Optimization times and results for uncorrelated PIs

FSM-like Patterns

PIs	NoOpt	NoDup		PS		GPS		CS	
	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks
4	36 510	0.03	333	0.04	167	0.04	117	0.03	104
8	73 412	0.07	1 075	0.07	537	0.09	343	0.08	280
16	158 528	0.35	99 641	8.73	66 294	3.89	29 908	18.82	22 190
25	249 878	0.60	232 834	29.33	172177	18.49	99 743	50.71	69 879
32	319 505	0.76	297 562	63.90	227 910	39.23	136 862	82.66	96 507

Table C-6: Optimization times and results for FSM-like PIs

Spatially Correlated Patterns

PIs	NoOpt	NoDup		PS		GPS		CS	
	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks
4	40 069	0.03	500	0.03	221	0.04	131	0.04	136
8	79 740	0.07	30 165	1.39	15 964	0.70	5 900	1.71	4 864
16	160 166	0.41	158 412	21.25	90 075	6.13	47 286	42.42	35 109
25	249 775	0.57	249 775	33.31	162 236	15.30	112 974	59.55	84 456
32	318 611	0.70	318 611	72.22	218 611	24.34	160 566	99.42	124 471

Table C-7: Optimization times and results for spatially correlated PIs

Low Active Signals

PIs	NoOpt	NoDup		PS		GPS		CS	
	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks	Time [s]	Blocks
4	9 038	0.02	383	0.03	212	0.03	140	0.02	125
8	16 092	0.06	11 589	0.54	9 399	0.46	4 362	0.94	3 523
16	32 397	0.34	32 390	10.08	32 825	8.93	48 686	39.49	31 190
25	50 163	0.44	50 162	164.10	50 355	24.87	110 205	59.76	53 940
32	64 062	0.52	64 062	315.14	64 240	50.99	160 869	126.49	65 990

Table C-8: Optimization times and results for low active PIs

C.2.2 Simulation Results

Related to section 4.5.5.

Uncorrelated Patterns

Circuit Name	Gates	PIs	Toggles/ 1000	Simulation Time [s]			
				Logic	NoDel	GPS	CS
misex2	69	25	249	0.17	0.15	0.19	0.18
f51m	112	8	80	0.25	0.24	0.11	0.09
bw	135	5	789	0.33	0.23	0.01	0.01
c1908	245	33	1 666	0.51	0.49	0.91	0.89
alu2	270	10	1 488	0.67	0.53	0.51	0.41
misex3	300	14	140	0.75	0.53	0.58	0.47
c1355	300	41	410	0.71	0.81	1.48	1.32
alu4	551	14	2 819	1.33	0.94	1.36	1.01
c3540	791	50	4 956	1.97	1.93	3.68	3.23
c6288	2 239	32	18 449	5.65	5.09	10.04	9.46

Table C-9: Simulation times for uncorrelated PIs

FSM-Like Patterns

Circuit Name	Gates	PIs	Toggles/ 1000	Simulation Time [s]			
				Logic	NoDel	GPS	CS
misex2	69	25	285	0.18	0.16	0.12	0.09
f51m	112	8	798	0.26	0.22	0.01	0.01
bw	135	5	255	0.33	0.07	0.01	0.01
c1908	245	33	1 658	0.50	0.48	0.55	0.42
alu2	270	10	1 462	0.66	0.51	0.03	0.03
misex3	300	14	1 336	0.80	0.50	0.13	0.10
c1355	300	41	2 342	0.64	0.76	0.90	0.73
alu4	551	14	2 801	1.32	0.93	0.24	0.22
c3540	791	50	4 932	1.99	1.96	2.32	1.76
c6288	2 239	32	18 393	5.48	4.92	6.12	4.89

Table C-10: Simulation times for FSM-like PIs

Spatially Correlated Patterns

Circuit Name	Gates	PIs	Toggles/ 1000	Simulation Time [s]			
				Logic	NoDel	GPS	CS
misex2	69	25	310	0.17	0.17	0.16	0.12
f51m	112	8	862	0.26	0.24	0.05	0.05
bw	135	5	723	0.33	0.21	0.01	0.01
c1908	245	33	1 634	0.50	0.49	0.81	0.72
alu2	270	10	1 360	0.65	0.49	0.25	0.21
misex3	300	14	1 431	0.80	0.55	0.42	0.29
c1355	300	41	2 186	0.65	0.71	1.25	1.15
alu4	551	14	2 550	1.33	0.86	0.54	0.55
c3540	791	50	4 975	1.97	1.95	3.08	2.59
c6288	2 239	32	17 598	5.52	4.75	8.13	7.34

Table C-11: Simulation times for spatially correlated PIs

Low Active Patterns

Circuit Name	Gates	PIs	Toggles/ 1000	Simulation Time [s]			
				Logic	NoDel	GPS	CS
misex2	69	25	75	0.16	0.03	0.11	0.07
f51m	112	8	261	0.25	0.06	0.03	0.03
bw	135	5	255	0.33	0.07	0.01	0.01
c1908	245	33	828	0.50	0.21	0.71	0.44
alu2	270	10	544	0.66	0.16	0.22	0.17
misex3	300	14	420	0.77	0.13	0.35	0.26
c1355	300	41	1 146	0.61	0.29	1.12	0.65
alu4	551	14	1 043	1.29	0.28	0.92	0.57
c3540	791	50	1 881	1.92	0.52	2.56	1.21
c6288	2 239	32	10 870	5.25	2.31	8.26	5.63

Table C-12: Simulation times for low active PIs

Sequential Circuits

Circuit Name	Gates	PIs	FF	Simulation Time [s]			Blocks	
				Logic	NoDel	CS	before	after
S382	162	3	21	0.32	0.06	0.01	35 060	2 023
S386	161	7	6	0.30	0.09	0.04	394 336	7 861
S820	293	18	5	0.74	0.26	0.36	97 630	64 173
S1196	533	14	18	1.16	0.45	0.81	121 138	87 150
S1494	649	8	6	1.30	0.28	0.12	48 563	10 007
S5378	2781	68	179	5.01	1.68	4.02	576 726	586 610
S9234	5601	19	228	8.97	0.56	2.41	174 408	167 112

Table C-13: Simulation times for patterns of sequential circuits

C.2.3 Signal Grouping

Related to section 4.6.4

The experiments in this section were performed on a Sun Ultra Sparc 1, 170 MHz, 192 MB main memory, running Sun Solaris 2.5.

Legend

Gr parameter *group size*
 Gr = '-' without any optimization
 Gr = 'opt' optimized with *CS*

Blocks number of blocks

Time [s]: simulation time in seconds

Combinational Circuits

Circuit	Gr	Uncorrelated Patterns					FSM-like Patterns				
		Blocks	Time [s]	Error [%]			Blocks	Time	Error [%]		
				global	local	max.			global	local	max.
c499	-	410 191					401 910				
	opt	292 058	1.05				273 957	0.96			
	12	90 504	0.73	0.12	0.84	18.9	12 256	0.20	0.17	1.32	21.64
	8	44 778	0.57	0.26	1.26	35.9	8 152	0.12	0.34	1.75	37.15
	4	1 166	0.03	0.39	4.38	28.4	941	0.02	0.19	5.27	57.29
	2	192	0.01	1.21	22.15	270.7	197	0.01	0.47	12.99	201.22

Table C-14: Combinational circuits with different group sizes

Circuit	Gr	Uncorrelated Patterns					FSM-like Patterns					
		Blocks	Time [s]	Error [%]			Blocks	Time	Error [%]			
				global	local	max.			global	local	max.	
c880	-	598 610					588 854					
	opt	465 453	1.86				445 326	1.74				
	12	126 823	0.67	0.03	2.22	40.6	26 877	0.19	0.07	5.15	66.22	
	8	53 993	0.37	0.33	2.24	23.4	16 137	0.14	0.53	4.24	35.05	
	4	1 602	0.03	0.04	11.53	125.5	1 429	0.03	0.13	13.36	100.00	
	2	258	0.02	2.96	33.63	137.7	287	0.02	0.30	31.15	101.76	
c1908	-	329 256					324 321					
	opt	220 935	3.00				204 093	2.76				
	12	80 345	1.76	0.17	0.96	23.6	10 509	0.33	0.05	0.89	34.69	
	8	38 386	1.17	0.07	1.37	39.9	6 422	0.22	0.27	2.19	28.96	
	4	981	0.08	0.10	5.67	100.0	693	0.07	1.39	7.90	97.79	
	2	134	0.06	2.20	17.33	108.3	140	0.05	1.04	14.93	327.87	
c6288	-	319 955					314 206					
	opt	212 232	12.03				195 235	11.98				
	12	62 822	8.13	0.10	1.17	17.0	6 838	1.56	0.26	2.97	34.56	
	8	21 985	4.31	0.01	1.85	16.0	6 430	1.48	0.08	3.08	44.10	
	4	758	0.35	0.19	8.67	98.7	664	0.29	0.31	10.00	83.10	
	2	143	0.17	2.40	25.99	236.6	129	0.17	0.90	23.69	258.79	
c7552	-	2 071 326					2 036 899					
	opt	1 852 682	19.62				1 807 714	19.63				
	12	502 340	11.79	0.09	1.22	23.5	43 201	2.05	0.08	3.37	56.13	
	8	202 622	5.56	0.10	1.79	31.6	53 314	2.06	0.04	3.26	93.48	
	4	5 147	0.91	0.56	8.42	95.6	5 071	0.88	0.40	9.90	280.65	
	2	926	0.76	0.09	23.80	337.3	908	0.80	0.05	22.70	221.89	

Table C-14: Combinational circuits with different group sizes

FSMs

Circuit	Gr	Blocks	Time [s]	Error [%]		
				global	local	max.
s328	-	35 060				
	opt	3 071	0.06			
	12	518	0.02	8.28	16.53	144.9
	8	370	0.02	7.82	22.35	162.8
	4	108	0.02	24.40	46.30	475.8
	2	76	0.02	18.90	41.42	539.4
s820	-	97 630				
	opt	87 427	0.53			
	12	25 252	0.14	4.60	11.03	86.4
	8	11 215	0.05	7.63	15.03	150.7
	4	527	0.02	5.08	33.00	246.9
	2	101	0.02	0.22	54.13	509.8
s1196	-	123 138				
	opt	112 972	1.17			
	12	23 003	0.28	11.17	18.14	110.8
	8	6 313	0.09	16.73	23.78	179.8
	4	572	0.04	18.27	37.68	661.5
	2	203	0.03	21.82	71.88	940.0
s1494	-	48 563				
	opt	15 440	0.27			
	12	2 231	0.06	4.50	0.85	10.1
	8	2 272	0.07	3.86	4.16	13.9
	4	253	0.04	6.75	2.11	13.6
	2	96	0.04	6.75	21.67	36.1
s9234	-	174 408				
	opt	235 701	3.18			
	12	40 196	1.60	14.88	14.23	126.9
	8	26 370	1.55	18.63	20.99	121.4
	4	16 765	1.46	24.25	27.36	175.2
	2	16 448	1.46	2.36	29.11	220.2

Table C-15: The logic parts of some FSMs with different group sizes

C.3 Real Delay Power Estimation

Related to chapter 5.

Legend

bitwise:	<i>ASSeT</i> in a bitwise mode
pure:	pure set simulation without <i>DPS</i> and <i>schedule compression</i>
DPS:	simulation with <i>DPS</i>
5%, 20%, 50%	<i>schedule compression</i> with $\delta=5\%$, 20% or 50%
word:	<i>PAPSAS</i> without <i>DPS</i> and <i>schedule compression</i>
Max. Error:	maximum gate related error
Std. Deviation:	standard deviation of the gate related errors

C.3.1 ASSeT versus Synopsys

Related to section 5.2.3.

Circuit Name	Load Independent Delays				Load Dependent Delays		
	Hazard Rate [%]	Simulation Times [s]			Hazard Rate [%]	Simulation Times [s]	
		bitwise	ASSeT	Synopsys		bitwise	ASSeT
c432	24.6	136.5	35.4	130.6	29.0	139.3	35.4
c499	28.6	173.0	39.0	178.0	28.6	176.6	38.2
c880	33.3	301.5	164.8	298.0	34.2	305.1	160.7
c1355	54.5	484.0	238.9	282.3	33.2	430.5	236.2
c1908	43.5	741.1	206.7	369.2	45.0	749.3	776.5
c2670	29.0	988.5	301.4	1769.3	45.2	480.8	299.1
s820	4.9	137.8	1.7	24.9	13.9	137.9	2.2
s1196	9.2	232.1	7.4	45.5	15.4	233.9	9.3
s5378	6.0	1167.5	23.5	523.0	5.8	1168.7	34.5
s9234	9.8	2299.1	35.8	844.9	9.6	2300.4	43.4
s13207	10.0	3485.7	33.4	3285.4	11.3	3519.2	35.9
s15850	27.0	4227.3	119.9	2851.3	25.1	4218.0	154.1

Table C-16: Simulation times of ASSeT and Synopsys

C.3.2 ASSET, PAPSAS and Optimizations

Related to 5.3.2, 5.4.1, and 5.4.2.

In these experiments the load dependence of the gate delays was taken into account. Since a suitable Synopsys library was not available, the compare to Synopsys had to be omitted. The sequential circuits *s13207*, *s15850*, and *s35932* were simulated with purely random patterns. Hence, they can be considered as large combinational circuits here.

Runtimes

In the following table the runtimes of bitwise simulation, ASSET and PAPSAS are depicted. For ASSET *pure* denotes set based simulation without schedule compression. DPS was omitted for the smaller circuits, since it does not improve the performance of ASSET. However, for the larger circuits DPS was used to keep the memory requirements below 1GB. Those circuits are marked with an asterisk.

Circuit Name	Hazards [%]	Runtime [s]									
		Bit-wise	ASSET				PAPSAS				
			pure	5%	20%	50%	word	DPS	5%	20%	50%
c432	29.0	139	35	26	13	9	21	6	3	1.6	1.0
c499	28.6	177	38	11	8	6	27	8	2	1.1	0.9
c880	34.2	305	161	92	46	26	49	18	11	4.8	2.7
c1355	33.2	431	236	95	44	30	72	19	10	4.6	2.9
c1908	45.0	749	776	250	120	64	204	139	34	10.7	5.0
c2670	45.2	1091	299	183	118	82	199	61	28	14.0	9.4
c3540*	51.5	1408	722	670	537	329	432	369	170	33.5	12.4
c5315*	50.8	2221	1276	1087	895	303	611	465	139	36.4	16.6
c6288*	92.5	7758	11680	9140	5357	3064	7301	7136	2519	282.1	88.4
c7552*	53.5	3410	2186	1869	1483	879	1066	797	212	57.1	25.9
s13207*	13.1	5679	484	435	357	253	528	66	17	15.6	33.2
s15850*	25.7	7190	1736	1836	1518	643	919	191	124	73.4	47.4
s35932*	55.3	16585	8788	4652	2528	1926	3966	349	237	176.7	143.3

Table C-17: Hazard rates and runtimes of real delay simulations with random patterns

Errors of Schedule Compression

Circuit Name	Global Error [%]			Local Error [%]			Max. Error [%]			Std. Deviation [%]		
	5%	20%	50%	5%	20%	50%	5%	20%	50%	5%	20%	50%
c432	0.0	0.1	0.3	0.0	0.1	0.7	0	1	5	0.1	0.2	1.3
c499	0.5	0.2	0.4	0.7	1.3	0.7	5	9	5	0.9	2.1	1.1
c880	0.0	0.1	0.9	0.0	0.1	2.6	0	3	55	0.1	0.3	8.1
c1355	6.7	1.5	2.5	6.6	2.4	4.8	54	14	43	13.0	3.2	6.5
c1908	0.0	0.0	0.8	0.3	1.4	3.2	9	17	25	0.8	2.5	5.3
c2670	0.1	0.3	3.5	0.1	2.7	7.6	27	963	1677	0.9	28.4	49.3
c3540	0.1	0.2	0.7	0.2	1.3	3.3	29	70	46	1.0	3.5	6.1
c5315	0.2	0.6	0.6	0.2	2.3	3.2	22	35	33	0.9	4.8	5.1
c6288	0.5	0.8	1.4	1.1	1.5	4.0	7	25	25	1.3	1.9	4.7
c7552	0.1	0.2	0.3	0.2	1.6	3.0	42	115	110	1.2	6.6	5.3
s13207	0.0	0.1	1.1	0.0	0.7	2.4	8	83	78	0.2	3.8	8.2
s15850	0.3	0.3	0.7	0.3	1.0	1.9	56	68	89	2.3	4.2	5.2
s35932	1.7	0.3	0.3	1.9	0.8	1.4	27	21	12	4.5	2.7	2.3
average	0.2	0.3	0.7	0.3	1.1	2.5	22	110	169	2.1	4.9	8.4

Table C-18: Error rates of schedule compression for different δ

Sequential Circuits

The patterns for the following experiment were generated by high level simulation of the sequential circuits.

Circuit Name	Hazards [%]	Bit-wise	Runtime [s]								
			ASSeT				PAPSAS				
			pure	5%	20%	50%	word	DPS	5%	20%	50%
s820	13.9	139	2.2	2.1	1.9	1.5	2.2	9.4	0.9	0.9	0.8
s1196	15.4	235	9.3	8.1	6.0	4.3	21.5	4.4	3.5	2.4	1.6
s5378	5.8	1171	34.5	28.6	20.9	15.0	79.9	11.9	9.5	6.9	5.2
s9234	9.6	1563	43.2	38.8	29.8	22.3	121.7	17.9	15.9	12.4	9.3
s13207	11.3	3510	35.8	35.2	33.1	29.4	189.3	19.3	19.1	18.4	17.2
s15850	25.1	2883	154.7	106.1	73.5	51.7	240.7	46.9	41.6	31.4	24.3

Table C-19: Hazard rates and runtimes of real delay simulations with RTL generated patterns

C.4 Probabilistic Simulation

Related to chapter 6.

For all simulations with BDDs a memory limit of 1GB was used. C6288 could only be simulated with local BDDs. Otherwise its BDDs exceed the memory limit.

Legend

vectors/1000: number of test vectors for logic simulation divided by 1000

prob.: probabilistic simulation with XOR-BDDs

max: maximum gate related deviation from/to logic simulation.

C.4.1 Correlation Layers

Related to section 6.2.5.

Accuracy of Logic Simulation

Circuit Name	Vectors/1000	CPU [s]	Memory [MB]	Deviation [%]		
				global	local	max
c432	prob.	2.81	14.681	---	---	---
	1	0.07	1.418	1.006	5.135	23.171
	2.5	0.12	1.770	1.028	3.050	10.361
	10	0.34	3.564	0.072	0.931	4.353
	25	0.87	7.112	0.039	0.752	3.576
	100	3.56	24.888	0.003	0.736	3.459
	250	9.16	60.409	0.013	0.793	3.266
	1000	42.27	238.060	0.043	0.846	3.689
c499	prob.	163.41	365.069	---	---	---
	1	0.09	1.508	0.297	3.729	48.387
	2.5	0.13	1.910	0.138	2.168	26.452
	10	0.41	3.941	0.053	0.995	14.839
	25	0.96	7.996	0.113	0.701	7.871
	100	3.99	28.280	0.036	0.504	4.516
	250	10.56	68.879	0.005	0.410	2.872
	1000	47.67	271.893	0.023	0.320	2.502

Table C-20: Probabilistic compared to logic simulation

Circuit Name	Vectors/ 1000	CPU [s]	Memory [MB]	Deviation [%]		
				global	local	max
c880	prob.	31.20	95.913	---	---	---
	1	0.17	1.779	0.500	4.714	34.783
	2.5	0.24	2.344	0.289	2.969	28.000
	10	0.68	5.260	0.357	1.777	21.333
	25	1.56	11.052	0.126	1.039	11.373
	100	6.55	40.109	0.078	0.890	5.097
	250	17.01	98.215	0.002	0.758	4.850
	1000	78.82	388.883	0.034	0.820	3.707
c1355	prob.	530.04	836.019	---	---	---
	1	0.21	1.811	0.102	3.484	48.387
	2.5	0.28	2.229	0.176	2.165	26.452
	10	0.72	4.302	0.056	0.903	14.839
	25	1.70	8.447	0.146	0.732	7.871
	100	6.31	29.156	0.096	0.621	4.516
	250	16.09	70.616	0.070	0.545	2.872
	1000	78.69	277.947	0.074	0.477	2.502
c1908	prob.	57.93	111.338	---	---	---
	1	0.29	2.016	0.802	3.519	50.000
	2.5	0.39	2.393	0.305	2.011	41.176
	10	0.98	4.261	0.150	0.759	17.647
	25	2.17	7.972	0.117	0.716	8.267
	100	8.42	26.592	0.050	0.575	6.615
	250	21.78	63.825	0.056	0.542	6.646
	1000	105.22	249.947	0.066	0.530	4.640
c2670	prob.	12.44	74.859	---	---	---
	1	0.71	3.597	0.213	3.672	46.667
	2.5	1.02	5.834	0.171	2.496	25.333
	10	2.74	16.942	0.052	1.173	10.667
	25	6.29	39.159	0.011	1.074	7.467
	100	25.80	150.324	0.013	0.789	6.167
	250	69.54	372.655	0.000	0.702	4.267
	1000	memory limit exceeded				

Table C-20: Probabilistic compared to logic simulation

Circuit Name	Vectors/ 1000	CPU [s]	Memory [MB]	Deviation [%]		
				global	local	max
c5315	prob.	7.61	46.916	---	---	---
	1	1.22	4.326	0.380	3.767	47.541
	2.5	1.58	6.129	0.218	2.368	31.818
	10	3.62	15.148	0.036	1.052	12.667
	25	7.74	33.154	0.014	0.820	7.297
	100	29.75	123.192	0.047	0.678	4.848
	250	75.58	303.318	0.084	0.596	4.293
	1000	memory limit exceeded				
c7552	prob.	9.57	52.446	---	---	---
	1	1.93	5.530	0.379	4.115	84.615
	2.5	2.39	7.652	0.287	2.453	40.645
	10	5.48	18.269	0.140	1.077	22.029
	25	10.92	39.486	0.053	0.924	13.231
	100	41.26	145.597	0.062	0.635	6.492
	250	104.71	357.770	0.043	0.557	4.696
	1000	memory limit exceeded				

Table C-20: Probabilistic compared to logic simulation

Najm's Approximation versus XOR-BDDs

Circuit Name	XOR-BDDs		Najm's Approximation				
	CPU [s]	Memory [MB]	CPU [s]	Memory [MB]	Error [%]		
					global	local	max
c432	21.751	3.48	19.522	4.45	96.714	114.415	565.5
c499	372.409	160.33	58.172	45.73	558.698	6.580	48.4
c880	107.685	32.34	37.750	16.20	147.845	193.730	6513.3
c1355	840.041	520.18	362.341	132.78	334.412	56.531	303.2
c1908	117.712	57.71	54.084	18.40	220.759	111.742	8183.3
c2670	120.333	16.79	80.250	15.68	128.798	63.833	1386.2
c5315	81.683	10.94	56.304	9.72	138.624	37.595	15157.8
c7552	92.882	13.44	99.206	27.12	134.751	48.315	3074.2

Table C-21: Najm's approximation versus XOR-BDDs

XOR-BDDs with and without Correlation Layers

Circuit Name	without Correlation Layers					with Correlation Layers				
	CPU [s]	Memory [MB]	Deviation [%]			CPU [s]	Memory [MB]	Deviation [%]		
			global	local	max			global	local	max
c499	162.39	373.6	1.81	3.32	59.1	1948.33	952.6	0.02	0.69	9.5
c880	32.30	107.7	0.85	10.29	109.5	128.67	115.1	0.15	1.01	13.0
c1908	57.85	117.8	0.56	2.37	75.6	1036.73	248.9	0.05	0.58	12.2
c2670	16.80	120.4	0.52	18.48	433.2	197.75	103.3	0.17	0.89	14.8
c5315	10.89	81.6	1.60	12.56	385.7	107.81	68.5	0.05	0.70	12.4
c7552	13.35	92.8	2.05	11.12	459.7	147.80	77.6	0.04	0.67	28.1
s298	0.39	8.6	20.67	66.70	795.2	0.50	7.4	1.65	4.09	63.9
s820	25.02	68.1	10.34	26.08	341.4	185.49	72.5	5.86	9.19	93.0
s953	3.28	15.3	10.25	22.15	107.9	56.41	22.1	7.26	12.48	114.7
s1196	6.97	27.1	12.73	15.02	49.8	265.43	204.2	8.05	9.88	34.3
s1238	6.88	27.1	13.85	15.68	49.8	263.64	200.7	8.73	10.26	34.3
s1488	2.23	10.8	33.97	67.57	667.9	21.23	13.7	2.87	4.01	46.2
s1494	2.27	10.8	35.26	67.03	684.4	20.61	13.0	2.86	3.95	44.5

Table C-22: XOR-BDDs with and without correlation layers

C.4.2 Local BDDs

Related to section 6.3.4.

Block Size for Local BDDs

The following 9 circuits were used to investigate the dependence of resource requirements and accuracy on block size: c432, c499, c880, c1355, c1908, c2670, c3540, c5315 and c7552. Because of the huge amount of data, table C-23 presents only the averages over all five circuits.

Block size	CPU [s]	Memory [MB]	Deviation from Logic Simulation[%]		
			global	local	max
4	4.48	58.3	1.10	6.03	224.6
5	4.60	58.4	0.90	4.66	200.1
6	4.75	58.5	0.54	3.90	174.3
7	4.94	58.7	0.51	3.85	215.2
8	5.18	59.0	0.50	3.77	220.8
9	5.49	59.3	0.55	3.47	222.0

Table C-23: Resource requirements and accuracy for different block sizes

Block size	CPU [s]	Memory [MB]	Deviation from Logic Simulation[%]		
			global	local	max
10	5.85	59.7	0.42	3.18	148.9
11	6.46	60.5	0.37	2.64	111.6
12	7.22	62.5	0.40	2.68	99.4
13	8.13	63.3	0.28	2.37	124.8
14	11.16	77.1	0.29	2.31	124.9
15	15.76	90.5	0.28	2.30	118.9
16	22.09	101.2	0.28	2.21	109.3
17	33.76	130.2	0.29	2.12	106.7
18	62.52	197.0	0.25	2.01	99.2
19	72.47	199.7	0.19	1.91	94.0

Table C-23: Resource requirements and accuracy for different block sizes

Local BDDs for Combinational Circuits and FSMs

Circuit Name	CPU [s]	Memory [MB]	Deviation from Logic Simulation [%]		
			global	local	max
c432	3.09	15.4	1.79	5.29	63.8
c499	3.69	16.7	0.26	1.15	18.1
c880	33.85	57.0	0.02	2.16	26.2
c1355	5.59	25.4	0.23	0.68	17.6
c1908	4.61	24.2	0.08	1.04	77.3
c2670	135.34	79.8	0.42	3.06	141.4
c3540	16.81	50.6	0.05	5.66	252.1
c5315	80.78	97.5	0.37	2.75	245.6
c6288	138.01	673.4	3.60	8.32	299.3
c7552	118.37	115.4	0.18	4.05	1188.8
s298	0.53	7.9	1.65	4.09	63.9
s820	4.94	11.0	6.39	10.80	262.6

Table C-24: Local BDDs for combinational circuits and FSMs

Circuit Name	CPU [s]	Memory [MB]	Deviation from Logic Simulation [%]		
			global	local	max
s953	8.49	12.7	8.46	14.01	114.5
s1196	18.00	20.2	7.14	9.29	42.2
s1238	17.70	20.2	7.62	9.91	42.1
s1488	12.10	20.1	6.26	9.13	447.8
s1494	12.52	20.1	6.72	9.49	454.7
s5378	104.21	134.8	8.18	14.17	288.9
s9234	169.50	381.2	5.96	7.88	123.8
s13207	116.70	345.4	0.29	1.76	529.5

Table C-24: Local BDDs for combinational circuits and FSMs

Example for the Covering Algorithm

This appendix gives an elaborate example of the covering algorithm that was presented in chapter 6. Consider the circuit of figure 6-16, which is repeated here for convenience.

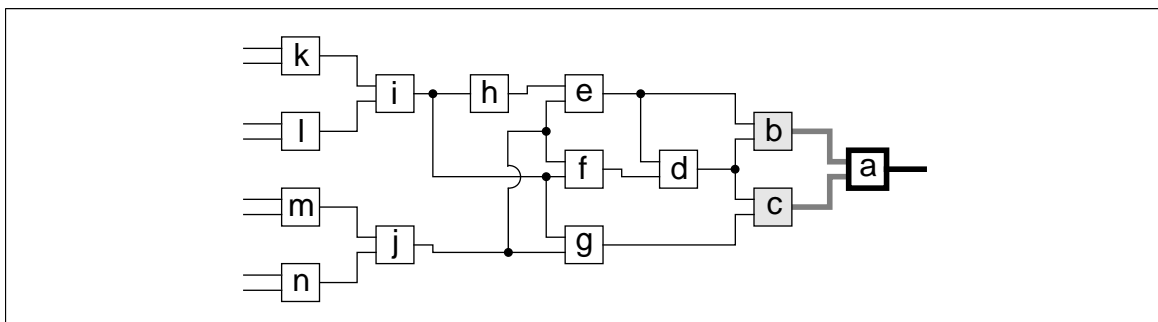


Figure D-1: Circuit to find a local BDD for

A subcircuit B with a maximum number of gates and a limited number of secondary inputs L for gate a is to be found.

Initial Parameters

- $L = 3$
- $B = \{a\}$
- $I_B = \{b, c\}$
- $f_B = L - |I_B| = 1$

Step 1

We start with building the subsets X_i from the input set I_B and build the according cost list:

i	X_i	n_c	n_+	E_X
0	b, c	1	1	d, e, g
1	b	0	1	d, e
2	c	0	1	d, g

Table D-1: Cost list of gates b and c

X_0 is added to the current block since it has highest priority and $n_{+0} \leq f_B$. X_0 is removed from the cost list and the other two entries are updated, resulting in an empty cost list.

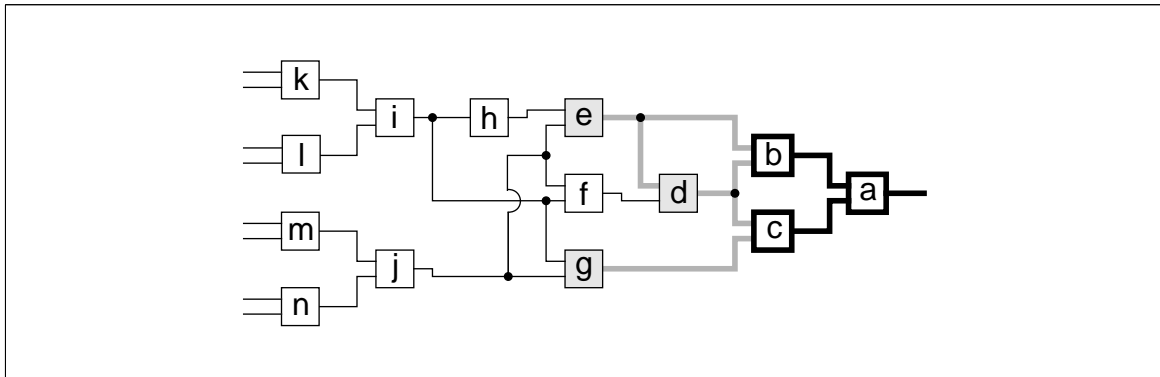


Figure D-2: Gates b and c have been added

Current Parameters

- $B = \{a, b, c\}$
- $I_B = \{d, e, g\}$
- $f_B = 0$

Note that the algorithm continues although $f_B = 0$.

Step 2

The cost list needs to be recomputed on the basis of the current I_B since it is empty:

i	X_i	n_c	n_+	E_X
0	d	1	0	f
1	d, g	1	1	f, i, j
2	e, g	1	1	h, i, j
3	e, d	1	1	f, h, j
4	g	0	1	i, j
5	e	0	1	h, j

Table D-2: Cost list of gates d, e and g

The covering algorithm chooses X_0 to be added to the current block B .

Current Parameters

- $B = \{a, b, c, d\}$
- $I_B = \{e, f, g\}$
- $f_B := f_B - n_+(X_0) = 0$

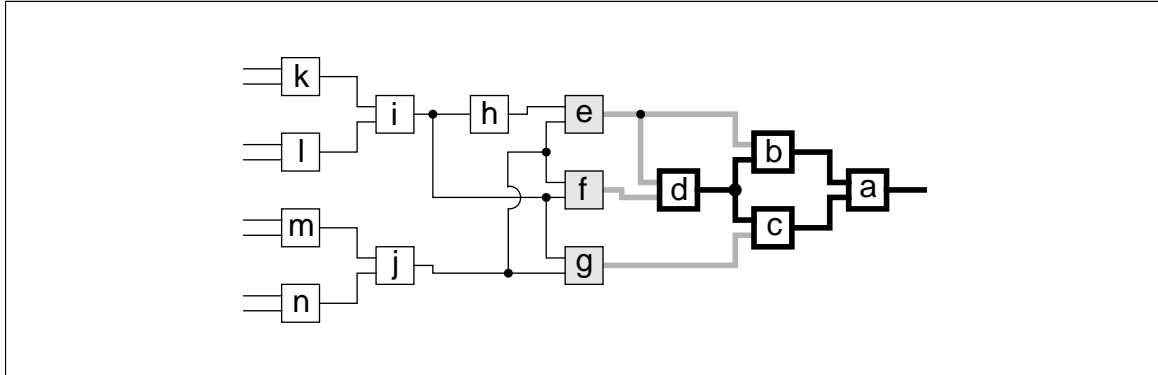


Figure D-3: Gate d has been added

Step 3

X_0 is removed from the cost list and the remaining entries are updated:

i	X_i	n_c	n_+	E_X
0	e, g	1	1	h, i, j
1	g	0	1	i, j
2	e	0	1	h, j

Table D-3: Cost list after eliminating gate d from all entries

$f_B=0$ but all remaining elements of the cost list require an additional SI. Therefore, the cost list is recomputed on the basis of the current I_B .

i	X_i	n_c	n_+	E_X
0	f, g	2	0	i, j
1	e, f	1	1	h, i, j
2	e, g	1	1	h, i, j
3	f	0	1	i, j
4	g	0	1	i, j
5	e	0	1	h, j

Table D-4: Cost list of gates e, f and g

Now, another pair $X_0 = \{f, g\}$ can be added to the block without increasing the number of SIs.

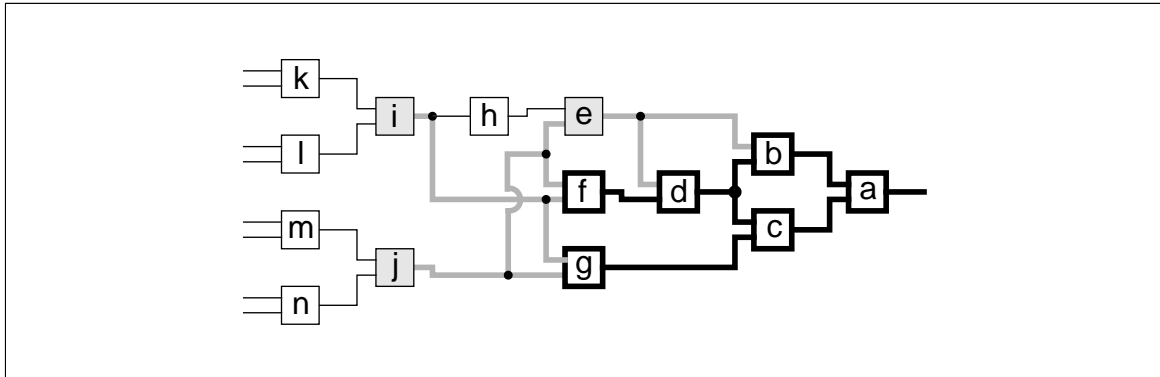


Figure D-4: Gates f and g have been added

Current Parameters

- $B = \{a, b, c, d, f, g\}$
- $I_B = \{e, i, j\}$
- $f_B = 0$

Step 4

Updating the cost list yields

i	X_i	n_c	n_+	E_X
0	e	1	0	h, j

Table D-5: Cost list after gates f and g have been removed

Now, gate e can be added without any additional cost.

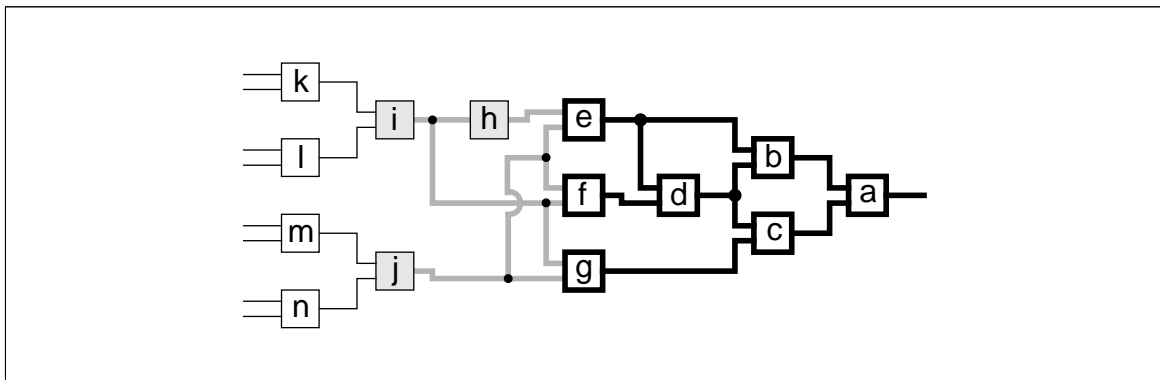


Figure D-5: Gates f and g have been added

Current Parameters

- $B = \{a, b, c, d, e, f, g\}$
- $I_B = \{h, i, j\}$

- $f_B = 0$

Step 5

Since gate e has been removed, the cost list is empty and must be recomputed on the basis of the current I_B .

i	X_i	n_c	n_+	E_X
0	h	1	-1	
1	h, i	1	0	k, l
2	h, j	1	0	m, n
3	i	0	1	k, l
4	j	0	1	m, n
5	i, j	0	2	k, l, m, n

Table D-6: Cost list of gates h, i and g

X_0 is added to the current block, since it has highest priority and it even increases f_B .

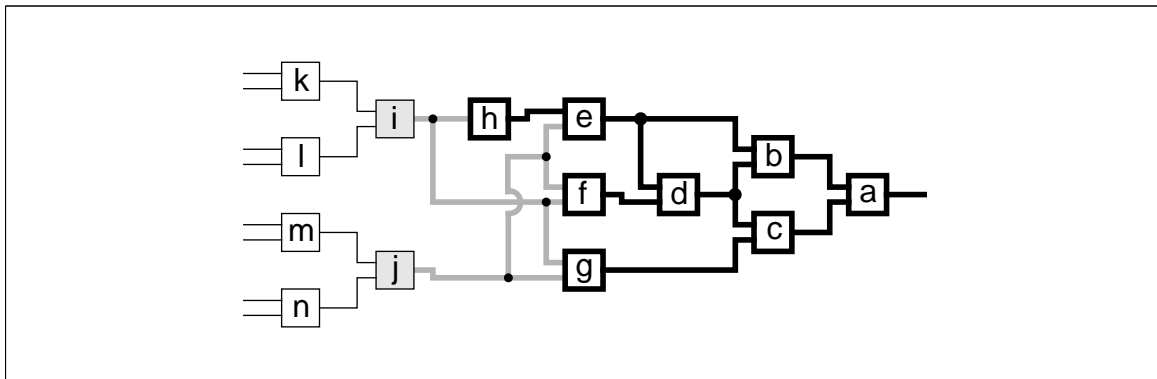


Figure D-6: Gate h has been added

Current Parameters

- $B = \{a, b, c, d, e, f, g, h\}$
- $I_B = \{i, j\}$
- $f_B = 1$

Step 3

The cost list then results in

i	X_i	n_c	n_+	E_X
0	i	0	1	k, l
1	j	0	1	m, n
2	i, j	0	2	k, l, m, n

Table D-7: Cost list after gate h has been removed from all entries

The order of X_0 and X_1 has been chosen arbitrarily since both entries have equivalent properties. Now X_0 is added to B and we obtain the final block that cannot be extended any further:

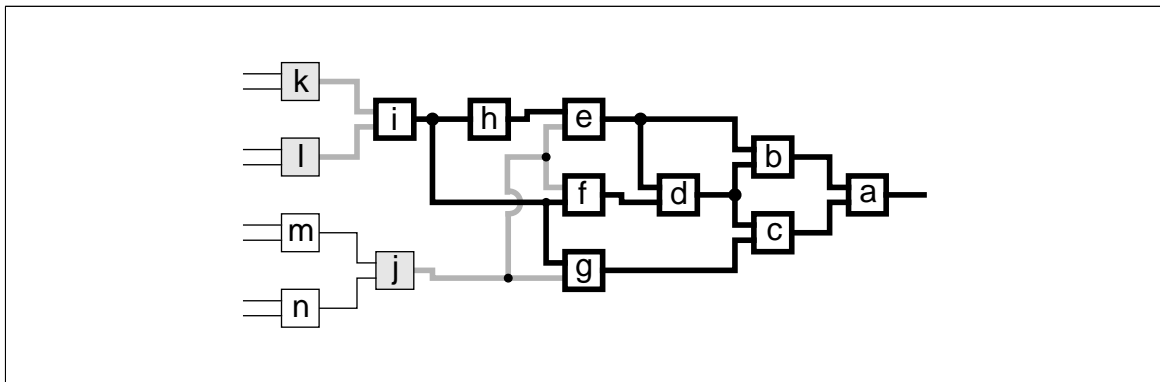


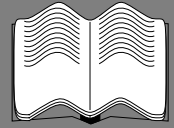
Figure D-7: Final block

Final Parameters

- $B = \{a, b, c, d, e, f, g, h, i\}$
- $I_B = \{j, k, l\}$
- $f_B = 0$

References

References



Publications

- [Abou 98] S. J. Abou-Samra: *Conception pour la Faible Consommation en Technologies SOI 2D et 3D: Application à l'Arithmétique*, Ph. D. Thesis at the Institut National Polytechnique de Grenoble, France (in French)
- [Aker 78] S. Akers: *Binary Decision Diagrams*, IEEE Transactions on Computers, Vol. C-27, pp. 509-516, June 1978
- [Alle 97] M. L. Alles: *Thin-film SOI Emerges*, IEEE Spectrum, June 1997, pp. 37-45
- [Bait 87] U. G. Baitinger: *Skriptum zur Vorlesung "Grundlagen der Digitaltechnik"*, Universität Karlsruhe, 1987 (in German)
- [Beun 88] M. Beunder, B. Höfflinger, J. Kernhof, *New Directions in Semicustom Arrays*, IEEE Journal of Solid-State Circuits, Vol. 23, No. 3, June 1988, pp. 728-735
- [Boll 98] B. Bollobàs: *Modern Graph Theory*, Springer, 1998, ISBN 0-387-98491-7
- [Boss 92] M. Bossert: *Kanalcodierung*, B.G. Teubner, 1992, ISBN 3-519-06143-0 (in German)
- [Birt 95] G. Birtwistle, A. Davis: *Asynchronous Digital Circuit Design*, Springer, 1995, ISBN 0-387-19901-2
- [Brgl 85] F. Brglez, H. Fujiwara: *A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran*, IEEE International Symposium on Circuits and Systems, ISCAS'85, June 1985
- [Brgl 89] F. Brglez, D. Bryan, K. Kozminski: *Combinational Profiles of Sequential Benchmark Circuits*, IEEE International Symposium on Circuits and Systems, ISCAS'89, May 1989, pp. 1929-1934
- [Bron 87] I. N. Bronstein, K. A. Semendjajew: *Taschenbuch der Mathematik*, Verlag Harri Deutsch, 1987, ISBN 3-87144-492-8 (in German)

- [Brya 86] Bryant R.: *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on computers, Vol. C-35, No. 8, August 1986
- [Brzo 95] J. A. Brzozowski, C.-J. H. Seger: *Asynchronous Circuits*, Springer, 1995, ISBN 0-387-94420-6
- [Burch 88] R. Burch, F. N. Najm, P. Yang, D. Hocevar: *Pattern-Independent Current Estimation for Reliability Analysis of CMOS Circuits*, 25th ACM/IEEE Design Automation Conference, DAC'88, 1988, pp. 294-299
- [Burch 93] R. Burch, F. N. Najm, P. Yang, T. N. Trick: *A Monte Carlo Approach for Power Estimation*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 1, no. 1, March 1993, pp. 63-71
- [Bueh 98] M. Bühler, D. Dallmann, U. G. Baitinger: *Switching Activity Analysis Using a Set Theoretical Approach*, GI/ITG/GME Workshop 1998, Paderborn, Germany, ISBN 3-931466-35-3, March 9-11, 1998
- [Bueh 98a] M. Bühler, Utz G. Baitinger: *VHDL-Based Development of a 32-Bit Pipelined RISC Processor for Educational Purposes*, 9th Mediterranean Electro-technical Conference (Melecon'98), Tel-Aviv, Israel, May 18-20, 1998
- [Bueh 98b] M. Bühler, B. Stöhr, A. Gerstlauer, U. G. Baitinger: *Eine geeignete Schaltungstechnologie für einen 3D-SOI-Prozeß mit T-Gate Transistoren*, Mikroelektronik für die Informationstechnik, Hannover, Germany, ISBN 3-80072325-5, March 3-4, 1998 (in German)
- [Bueh 98c] M. Bühler, K. Kapp, D. Dallmann, U. G. Baitinger, *TESA: Timeparallel Estimation of Switching Activity under a Real Delay Model*, 5th IEEE International Conference on Electronics, Circuits and Systems (ICECS'98), Lisbon, Portugal, September 7-10, 1998
- [Bueh 99] M. Bühler, M. Papesch, K. Kapp, U. G. Baitinger: *Efficient Switching Activity Simulation Under a Real Delay Model Using a Bitparallel Approach*, Design, Automation and Test in Europe, DATE'99, Munich, March 1999
- [Bueh 99a] M. Bühler, M. Papesch, U. G. Baitinger: *Accurate and Approximate Methods for Speeding up Signal Activity Estimation on Gate Level*, PATMOS'99, Kos Island, Greece, October 6-8, 1999
- [Carl 94] L. Carley, I. Lys: *QuadRail: A Design Methodology for Ultra-Low Power ICs*, 1994 International Workshop on Low Power Design, ISWLPD'94, pp. 225-230
- [Chan 95] A. P. Chandrakasan, R. W. Brodersen: *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995

- [Chou 94] T.-L. Chou, K. Roy, S. Prasad: *Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching*, IEEE/ACM International Conference on Computer Aided Design, ICCAD'94, pp. 300-330
- [Coli 91] J.P. Colinge: *Silicon-on-Insulator*, Kluwer Academic Publishers, 1991, ISBN 0-792-39150-0
- [Cong 96] J. Cong, M. Pedram: *Performance & Power Optimization in Synthesis & Layout of VLSI Circuits & Systems*, Documentation Tutorial 3, EURO-DAC'96, 1996
- [Corr 96] P. Correia et al.: *Low-Power CMOS Digital Cell Libraries: Performance and Testability*, PATMOS'96, Bologna, Italy, pp. 307-316
- [DeMi 94] G. De Micheli: *Synthesis and Optimization of Digital Circuits*, McGraw-Hill Inc., 1994, ISBN 0-07-016333-2
- [Deva 94] S. Devadas, J. Monteiro, B. Lin, C.Y. Tsui, M. Pedram: *Exact and Approximate Methods of Switching Activity Estimation in Sequential Logic Circuits*, International Workshop on Low Power Design, IWLPD'94, 1994, pp.117-122
- [Drec 96] R. Drechsler: *Ordered Kronecker Functional Decision Diagrams und ihre Anwendung*, Dissertation, Modell Verlag Göckel/Drechsler, 1996, ISBN 3-9805033-0-5 (in German)
- [Dres 93] F. Dresig, Ph. Lanches, O. Rettig, U. G. Baitinger: *Simulation and Reduction of CMOS Power Dissipation at Logic Level*, European Conference on Design Automation (EDAC), 1993, pp. 341-346
- [Dude 96] V. Dudek, W. Appel, L. Beer, G. Digele, B. Höfflinger: *Lithography-Independent Nanometer Silicon MOSFETs on Insulator*, IEEE Transactions on Electron Devices ED-43(10), 1996, pp. 1626-1632
- [Dude 97] V. Dudek: *Lithographie-unabhängige MOS-Transistoren mit Kanallängen kleiner 100 nm*, Fortschr.-Ber. VDI Reihe 9 Nr.248, Düsseldorf: VDI-Verlag 1997, Ph. D. Thesis, ISBN 3-18-324809-3 (in German)
- [Ells 97] J. Ellsberger, D. Hogrefe, A. Sarma: *SDL Formal Object-oriented Language for Communication Systems*, Prentic Hall Europe, 1997, ISBN 0-13-621384-7
- [Farn 94] C. Farnsworth, D. Edwards, S. Sikand: *Utilising Dynamic Logic for Low Power Consumption in Asynchronous Circuits*, Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Technical Committee on VLSI, Salt Lake City, Utah, USA, Nov. 3-5th, 1994

- [Gajs 83] D. D. Gajski, R. H. Kuhn: *New VLSI Tools*, IEEE Computer 16 (12/1983), pp. 11-14, 1983
- [Gosh 92] A. Ghosh, S. Devadas, K. Keutzer, J. White: *Estimation of Average Switching Activity in Combinational and Sequential Circuits*, 29th ACM/IEEE Design Automation Conference, DAC'92, pp. 253-259
- [Goud 91] N. Gouders, R. Kaibel: *PARIS A Parallel Pattern Fault Simulator for Synchronous Sequential Circuits*, IEEE International Conference on Computer-Aided Design, ICCAD, pp. 542-545, 1991
- [Gumm 96] M. Gumm, M. Bühler, U. G. Baitinger: *Design Education in Microelectronics at the University of Stuttgart: VHDL and Synthesis Based Design of a 32-bit RISC Processor in a Four Months Course*, Proceedings of the European Workshop Microelectronics Education, Grenoble, France, ISBN 981-02-2653-5, February 5-6, 1999.
- [Hach 96] G. D. Hachtel, F. Somenzi: *Logic Synthesis and Verification Algorithms*, Kluwer Academic Publishers, 1996
- [Hare 87] D. Harel: *Statecharts: A Visual Formalism for Complex Systems*, Science on Computer Programming, Vol. 8, 1987, pp. 231-274
- [Haus 77] M. Hausner: *Elementary probability theory*, Plenum-Rosetta, 1977, ISBN 0-306-20026-0
- [Hill 95] A. M. Hill, S.-M. Kang: *Determining Accuracy Bounds for Simulation-Based Switching Activity Estimation*, International Symposium on Low Power Design, ISLPD'95, 1995, pp. 215-220
- [Huan 95] C. X. Huang, B. Zhang, A-C. Deng, B. Swirski: *The Design and Implementation of Power Mill*, Proceedings, 1995 International Symposium on Low Power Design, April 23-26, 1995, pp. 105-110
- [Hünt 90] F. Hüntemann: *Anwendungsspezifischer Entwurf von DOMINO-CMOS-Schaltnetzen im Gate-Matrix-Entwurfstil*, VDI Verlag, Nr. 107, PhD. Thesis, ISBN 3-18-140709-7 (in German)
- [Kapo 94] B. Kapoor, *Improving the Accuracy of Circuit Activity Measurement*, International Workshop on Low Power Design, IWLPD'94, 1994, pp. 111-116
- [KoBa 95] U. Ko, P. Balsara, W. Lee: *Low Power Design Techniques for High-Performance CMOS Adders*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.3, No.2, June 1995, pp.327-333
- [Kund 95] K. S. Kundert: *The Designers Guide to Spice and Spectre*, Kluwer Academic Publishers, 1995, ISBN 0-792-39571-9

- [Lanc 95] Ph. Lanches: *Parallele Logiksimulation - effiziente Methoden für Multicomputer*, VDI Verlag GmbH, 1995, ISBN 3-18--315420-X (in German)
- [Lane 67] S. Mac Lane, G. Birkhoff: *Algebra*, The Mac Millan Company, 1967
- [Lava 93] L. Lavagno, A. Sangiovanni-Vincentelli: *Algorithms for Synthesis and Testing of Asynchronous Circuits*, Kluwer Academic Publishers, 1993, ISBN 0-7923-9364-3
- [Lee 59] C. Lee: *Representation of Switching Circuits by Binary-Decision Programs*, Bell System Technical Journal, Vol. 38, pp. 985-999, July 1959
- [Lehm 94] G. Lehmann, B. Wunder, M. Selz: *Schaltungsdesign mit VHDL*, Franzis-Verlag GmbH, 1994, ISBN 3-7723-6163-0
- [Lim 97] Y. J. Lim, M. Soma: *Statistical Estimation of Delay-Dependent Switching Activities in Embedded CMOS Combinational Circuits*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 5, No. 3, September 1997, pp. 309-319
- [Lipp 92] H. M. Lipp: *Skriptum zur Vorlesung Entwurf Digitaler Schaltungen IA, Formale Hilfsmittel*, Universität Karlsruhe, 1992 (in German)
- [Marc 66] M. Marcus, H. Minc: *Modern University Algebra*, The Mac Millan Company, 1967
- [Marc 94] R. Marculescu, D. Marculescu, M. Pedram: *Switching Activity Analysis Considering Spatiotemporal Correlations*, International Conference on Computer Aided Design, ICCAD'94, 1994, pp. 628-634
- [Marc 95] R. Marculescu, D. Marculescu, M. Pedram: *Efficient Power Estimation for Highly Correlated Input Streams*, 32nd Design Automation Conference, DAC'95, 1995, pp. 628-634
- [Mark 87] G. Markowsky: *Bounding Signal Probabilities in Combinational Circuits*, IEEE Transactions on Computers, Vol. C-36, No. 10, October 1987, pp. 1247-1251
- [Mina 90] S. Minato, N. Ishiura, S. Yajima: *Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation*, Proc. of 27th ACM/IEEE Design Automation Conference, DAC'90, 1990, pp. 52-57
- [Mina 96] S. Minato: *Binary Decision Diagrams and Applications for VLSI CAD*, Kluwer Academic Publishers, 1996

- [Mont 95] J. Monteiro, S. Devadas: *Techniques for the Power Estimation of Sequential Logic Circuits Under User-Specified Input Sequences and Programs*, 1995 International Symposium on Low Power Design, ISLPD'95, pp. 33-38
- [Mont 97] J. Monteiro, S. Devadas, A. Ghosh, K. Keutzer, J. White: *Estimation of Average Switching Activity in Combinational Logic Circuits Using Symbolic Simulation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, No. 1, January 1997, pp. 121-127
- [Moor 75] G. E. Moore: *Progress in Digital Integrated Electronics*, IEEE IEDM, pp. 11-13, 1975
- [Najm 90] F. N. Najm, R. Burch, P. Yang, I. N. Hajj: *Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits*, IEEE Transactions on Computer Aided Design, Vol. 9, No. 4, April 1990, pp. 439-450
- [Najm 93] F. N. Najm: *Transition Density: A New Measure of Activity in Digital Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, No. 2, February 1993, pp. 310-323
- [Nebe 97] W. Nebel, J. Mermet: *Low Power Design in Deep Submicron Electronics*, Kluwer Academic Publishers, 1997, ISBN 0-7923-4569-X
- [Papo 65] A. Papoulis: *Probability, Random Variables, and Stochastic processes*, McGraw-Hill, 1965
- [Park 75] K. P. Parker, E. J. McCluskey: *Probabilistic Treatment of General Combinational Networks*, IEEE Transactions on Computers, Vol. C-24, June 1975, pp. 668-670
- [Patt 96] D. A. Patterson, J. L. Hennessy: *Computer Architecture A Quantitative Approach*, Morgan Kaufmann Publishers Inc., Second Edition, 1996, ISBN 1-55860-329-8
- [Pram 94] A. Prameswar, H. Hara, T. Sakurai: *A High Speed, Low Power, Swing Restored Pass-Transistor Logic Based Multiply and Accumulate Circuit for Multimedia Applications*, IEEE 1994 Custom Integrated Circuits Conference, CICC'94, pp. 278-281
- [Raba 96] J. Rabaey: *Basics of Low Power Design*, EUROCHIP Course on Methods and Tools for Digital Systems Design, Sept. 1995, Leuven, Belgium
- [Rabe 98] D. Rabe, G. Jochens, L. Kruse, W. Nebel: *Power-Simulation of Cell Based ASICs: Accuracy and Performance Trade-Offs*, Design, Automation and Test in Europe, DATE'98, 1998, pp. 356-361

- [Rade 96] M. Radetzki, B. Timmermann, D. Rabe, W. Nebel: *Generation of Binary Patterns with Given Spatiotemporal Correlations*, Sixth International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS'96, 1996, pp. 199-208
- [Retti 96] O. Rettig: *Technologieabhängige Entwurfsverfahren für spezielle Strukturen integrierter Schaltungen*, VDI Verlag GmbH, 1996, ISBN 3-18-322909-9 (in German)
- [Roos 92] G. Roos, B. Höfflinger: *Complex 3D-CMOS Circuits based on a Triple-Decker Cell*, IEEE Solid-State Circuits SC-27 (7), 1992, pp. 1067-1072
- [Roos 93] G. Roos: *Zur drei-dimensionalen Integration von CMOS-Schaltungen*, VDI Verlag GmbH, Nr. 168, PhD. Thesis, ISBN 3-18-146809-6 (in German)
- [Sasa 96] T. Sasao, M. Fujita: *Representation of Discrete Functions*, Kluwer Academic Publishers, 1996
- [Savi 84] J. Savir, G. S. Ditlow, P. H. Bardell: *Random Pattern Testability*, IEEE Transactions on Computers, Vol. C-33, Jan. 1984, pp. 79-90
- [Schl 95] P. H. Schneider, U. Schlichtmann: *Decomposition of Boolean Functions for Low Power Based on a New Power Estimation Technique*, International Symposium on Low Power Design ISLPD'95, 1995, pp. 123-128
- [Schn 94] P. H. Schneider, U. Schlichtmann, K. Antreich: *A New Power Estimation Technique with Application to Decomposition of Boolean Functions for Low Power*, European Design Automation Conference, EuroDAC'94, 1994, pp. 388-393
- [Schn 95] P. H. Schneider, *PAPSAS: A Fast Switching Activity Simulator*, Fifth International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS'95, 1995, pp. 351-360
- [Seth 85] S. C. Seth, L. Pan, V. D. Agrawal: *PREDICT - Probabilistic Estimation of Digital Circuit Testability*, IEEE 15th Annual International Symposium on Fault-Tolerant Computing, 1985, pp. 220-225
- [Shan 38] C. E. Shannon: *A Symbolic Analysis of Relay and Switching Circuits*, Trans. AIEE (57), pp. 713-723, 1938
- [Stoe_98] B. Stöhr, M. Bühler, S. Weberruß, Utz G. Baitinger: *Eine Silicon-on-Insulator Sea-of-Gates Architektur für Low Power Anwendungen*, Mikroelektronik für die Informationstechnik, Hannover, Germany, ISBN 3-80072325-5, March 3-4, 1998 (in German)

- [Suzu 93] M. Suzuki et al.: *A 1.5-ns 32-b CMOS ALU in Double Pass-Transistor Logic*, IEEE Journal of Solid State Circuits, Vol. 28, No. 11, November 1993, pp. 1145-1150
- [Suzu 95] M. Suzuki et al.: *A 44. ns CMOS 54 x 54-b Multiplier Using Pass-Transistor Multiplexer*, IEEE Journal of Solid State Circuits, Vol. 30, No. 3, March 1995, pp. 251-256
- [Tcho 95] V. Tchoumatchenko, T. Vassileva, A. Guyot: *Timing Modeling for Adders Optimization*, 5th PATMOS, Oldenburg, Germany, October 1995, pp. 93-105
- [Tsui 94] C.-Y. Tsui, M. Pedram, A. M. Despain: *Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs*, 31st ACM/IEEE Design Automation Conference, DAC'94, pp. 18-22
- [Tsui 95] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, B. Lin: *Power Estimation Methods for Sequential Logic Circuits*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 3, No. 3, September 1995, pp. 404-416
- [Turg 95] S. Turgis, N. Azemard, D. Auvergne: *Explicit Evaluation of Short Circuit Power Dissipation for CMOS Logic Structures*, 1995 International Symposium on Low Power Design, ISLPD'95, 1995, pp.129-134
- [Tutt 84] W. T. Tutte: *Graph Theory*, Addison-Wesley Publishing Company, 1984, ISBN 0-201-13520-5
- [Veen 84] H. J. M. Veendrik: *Short -Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits*, IEEE Journal of Solid State Circuits, sc-19(4), August 1984
- [Waic 85] J. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom, T. McCarthy: *Fault Simulation for Structured VLSI*, VLSI Systems, 1985, pp. 20-32
- [West 93] N. Weste, K. Eshraghian: *Principles of CMOS VLSI Design*, Addison Wesley, 1993, ISBN 0-201-53376-6
- [Xake 94] M. G. Xakellis, F. N. Najm: *Statistical Estimation of the Switching Activity in Digital Circuits*, 31st ACM/IEEE Design Automation Conference, DAC'94, 1994, pp. 728-733
- [Zimm 97] R. Zimmermann, W. Fichtner: *Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic*, IEEE Journal of Solid-State Circuits, Vol. 32, No. 7, July 1997, pp. 1079-1090

Technical Documentations and Unpublished References

- [BENCH] <http://www.cbl.ncsu.edu/benchmarks/Benchmarks-upto-1996.html>
- [Bueh 97] M. Bühler, B. Stöhr: *Hiperlogic Workpackage 3.1 Cell Library & Design Kit Development Annual Review Report 1996/97*, Hiperlogic, ESPRIT IV Project No. 20023
- [Bueh 97a] M. Bühler: *Hiperlogic Workpackage 3.2 Low Power Synthesis and Signal Activity Management*, Annual Review Report 1996/97, Hiperlogic, ESPRIT IV Projekt Nr. 20023
- [Bueh 98] M. Bühler: *Hiperlogic Workpackage 3.1 Cell Library & Design Kit Development*, Annual Review Report 1997/98, Hiperlogic, ESPRIT IV Projekt Nr. 20023
- [Bueh 98a] M. Bühler: *Hiperlogic Workpackage 3.2 Low Power Synthesis and Signal Activity Management*, Annual Review Report 1997/98, Hiperlogic, ESPRIT IV Projekt Nr. 20023
- [Cade 95] *Spectre Reference Manual*, Cadence Design Framework II, Release 9502, Online Documentation
- [Dors 98] R. Dorsch: *Variable Orders for Some ISCAS Benchmarks*, email correspondence, July, 1998
- [Hipe 95] *Hiperlogic: Thousand MOPS Per Milliwatt CMOS High Performance Logic*, Esprit Project 20.023, Technical Annex, 1995
- [IMS 91] W. Beller, R. Hainbucher, T. Schwederski: *IMS 1.2 μm GATE FOREST Zellenkatalog*, Technical Report IMS-TB-3/91, Institute for Microelectronics Stuttgart, IMS, October 1991 (in German)
- [IMS 96] W. Beller, E. Bernath, R. Hainbucher, T. Schwederski: *IMS 0.8 μm GATE FOREST Library GFN 120*, Institute for Microelectronics Stuttgart, IMS, January 1996, <http://www.ims-chips.de/world/gfn120/gfn120.html>
- [IMS 97] *Preliminary Geometrical Design Rules (PGDR) IMS-TGSOI0.8 (0.8 μm technology with 0.1 μm CMOS transistors T-Gate on SOI)*, Version 1.0, July 28th 1998, Technical Documentation, Institute for Microelectronics, Stuttgart (IMS).
- [LGSY 91] Benchmark Examples of MCNC International Workshop on Logic Synthesis, 1991

- [Maye 98] G. Mayer, IBM R&D Laboratory Böblingen, Germany, private correspondence.
- [Rade 97] M. Radetzki: *Pattern Generator*, email correspondence, September 4th, 1997
- [SIA 97] *The National Technology Roadmap for Semiconductors*, SIA Semiconductor Industry Association
- [Some 98] F. Somenzi: *CUDD Package*, University of Colorado at Boulder, ECE Dept. Boulder, Release 2.3.0, Sep. 1998, <http://vlsi.colorado.edu/~fabio>
- [SyPo 96] Synopsys Inc., *Power Products Reference Manual*, Synopsys Online Documentation, Version 3.5, 1996
- [SyLi 96] Synopsys Inc., *Library Compiler User Guide*, Synopsys Online Documentation, Version 3.5, 1996

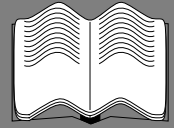
Study and Master Thesis (in German)

- [Dall 98] D. Dallmann: *Entwicklung eines symbolischen Schaltungssimulators*, Master Thesis, University of Stuttgart, IND and IPVR-ISE, 1998.
- [FucD 98] R. Fuchs: *Entwicklung eines Werkzeugs zur Logiksynthese mittels ROBDDs*, Master Thesis No. 1684, University of Stuttgart, IPVR-ISE, 1998.
- [FucS 98] R. Fuchs: *Analyse von ROBDDs für Werkzeuge zur Logiksynthese*, Study Thesis No. 1666, University of Stuttgart, IPVR-ISE, 1998.
- [Gers 97] A. Gerstlauer: *Entwicklung einer Zellbibliothek in CMOS und Passtransistorlogik für Low-Power-Anwendungen*, Master Thesis No. 1483, University of Stuttgart, IND and IPVR-ISE, 1997.
- [Heit 98] G. Heitsch: *Automatische Integration eine Zellbibliothek in ein kommerzielles Werkzeug zur Logiksynthese*, Master Thesis No. 1577, University of Stuttgart, IPVR-ISE, 1998 (in English).
- [KapD 98] K. Kapp: *Ein Simulator zur Bestimmung der Schaltaktivität basierend auf einer Kombination probabilistischer und logischer Methoden*, Master Thesis No. 1672, University of Stuttgart, IPVR-ISE, 1998.
- [KapS 98] K. Kapp: *Entwicklung eines Zeitmodells für einen symbolischen Simulator*, Study Thesis No. 1677, University of Stuttgart, IPVR-ISE, 1998.

- [Malu 98] C. Maluck: *Untersuchung von Hazards und Glitches in CMOS Gattern*, Study Thesis, University of Stuttgart, IND and IPVR-ISE, 1998.
- [Mess 96] F. Messicci: *Erweiterung des DLXS-Prozessormodells um eine 5-stufige Pipeline*, Study Thesis No. 1550, University of Stuttgart, IPVR-ISE, 1996.
- [Pape 98] M. Papesch: *Implementierung bitparalleler Signalverarbeitung für den AS-SeT-Simulator*, Software Practical, University of Stuttgart, IPVR-ISE, 1998.
- [Schl 98] T. Schlenker: *Redesign eines DLX-Prozessormodells mit Pipeline*, Master Thesis No. 1586, University of Stuttgart, IPVR-ISE, 1998.
- [Webe 97] L. Weberruß: *Erweiterung des DLXS-Prozessormodells um eine 5-stufige Pipeline*, Study Thesis No. 1625, University of Stuttgart, IPVR-ISE, 1997.
- [Weit 97] F. Weitmann: *Entwicklung eines Werkzeugs zur automatischen Charakterisierung von Bibliothekszellen*, Study Thesis No. 1618, University of Stuttgart, IPVR-ISE, 1997.

Curriculum Vitae

Curriculum
Vitae

**Name**

Markus Bühler

Date/Place of Birth

June 25th 1967 in Offenburg (Germany)

Education

August 1973 - June 1977

Grundschule Fessenbach, Primary School

August 1977 - June 1986

Schiller-Gymnasium Offenburg, High School Diploma (Abitur) 1986

October 1987 - November 1993

University of Karlsruhe, study of electrical engineering

May 1993 - November 1993

Master thesis> “Vergleich verschiedener Bildverarbeitungsalgorithmen zur Lokalisierung eines mobilen Roboters” (in German), University of Nancy, France, Prof. Husson

November 1993

Graduation with distinction as a Dipl.-Ing.

December 1993 - May 1995

Different industrial positions

June 1995 - November 1999

Ph.D. student at the University of Stuttgart, IPVR-ISE, Prof. Baitinger

