




<b>Title</b>	Systematic delay-driven power optimisation and power-driven delay optimisation of combinational circuits
<b>Author(s)</b>	Mehrotra, Rashmi
<b>Publication date</b>	2013
<b>Original citation</b>	Mehrotra, R. 2013. Systematic delay-driven power optimisation and power-driven delay optimisation of combinational circuits. PhD Thesis, University College Cork.
<b>Type of publication</b>	Doctoral thesis
<b>Rights</b>	© 2013, Rashmi Mehrotra <a href="http://creativecommons.org/licenses/by-nc-nd/3.0/">http://creativecommons.org/licenses/by-nc-nd/3.0/</a> 
<b>Embargo information</b>	Restricted to everyone for one year
<b>Item downloaded from</b>	<a href="http://hdl.handle.net/10468/1111">http://hdl.handle.net/10468/1111</a>

Downloaded on 2017-02-12T13:08:25Z



# UCC

University College Cork, Ireland  
Coláiste na hOllscoile Corcaigh

Ollscoil na hÉireann

Systematic Delay-driven Power  
Optimisation and Power-driven Delay  
Optimisation of Combinational Circuits

A thesis presented to the

National University of Ireland, Cork

for the Degree of

Doctor of Philosophy

by

Rashmi Mehrotra

Supervisor: Dr. Emanuel. M. Popovici

Head of Department: Prof. Nabeel Riza

Department of Electrical and Electronic Engineering,

University College Cork, Ireland

Feb 4, 2013

## Abstract

With the proliferation of mobile wireless communication and embedded systems, the energy efficiency becomes a major design constraint. The dissipated energy is often referred as the product of power dissipation and the input-output delay. Most of electronic design automation techniques focus on optimising only one of these parameters either power or delay. Industry standard design flows integrate systematic methods of optimising either area or timing while for power consumption optimisation one often employs heuristics which are characteristic to a specific design. In this work we answer three questions in our quest to provide a systematic approach to joint power and delay optimisation. The first question of our research is: How to build a design flow which incorporates academic and industry standard design flows for power optimisation? To address this question, we use a reference design flow provided by Synopsys and integrate in this flow academic tools and methodologies. The proposed design flow is used as a platform for analysing some novel algorithms and methodologies for optimisation in the context of digital circuits. The second question we answer is: Is possible to apply a systematic approach for power optimisation in the context of combinational digital circuits? The starting point is a selection of a suitable data structure which can easily incorporate information about delay, power, area and which then allows optimisation algorithms to be applied. In particular we address the implications of a systematic power optimisation methodologies and the potential degradation of other (often conflicting) parameters such as area or the delay of implementation. Finally, the third question which this thesis attempts to answer is: Is there a systematic approach for multi-objective optimisation of delay and power? A delay-driven power and power-driven delay optimisation is proposed in order to have balanced delay and power values. This implies that each power optimisation step is not only constrained by the decrease in power but also the increase in delay. Similarly, each delay optimisation step is not only governed with the decrease in delay but also the increase in power. The goal is to obtain multi-objective optimisation of digital circuits where the two conflicting objectives are power and delay. The logic synthesis and optimisation methodology

is based on AND-Inverter Graphs (AIGs) which represent the functionality of the circuit. The switching activities and arrival times of circuit nodes are annotated onto an AND-Inverter Graph under the zero and a non-zero-delay model. We introduce then several reordering rules which are applied on the AIG nodes to minimise switching power or longest path delay of the circuit at the pre-technology mapping level. The academic Electronic Design Automation (EDA) tool ABC is used for the manipulation of AND-Inverter Graphs. We have implemented various combinatorial optimisation algorithms often used in Electronic Design Automation such as Simulated Annealing and Uniform Cost Search Algorithm. Simulated Annealing (SMA) is a probabilistic meta heuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. We used SMA to probabilistically decide between moving from one optimised solution to another such that the dynamic power is optimised under given delay constraints and the delay is optimised under given power constraints. A good approximation to the global optimum solution of energy constraint is obtained. Uniform Cost Search (UCS) is a tree search algorithm used for traversing or searching a weighted tree, tree structure, or graph. We have used Uniform Cost Search Algorithm to search within the AIG network, a specific AIG node order for the reordering rules application. After the reordering rules application, the AIG network is mapped to an AIG netlist using specific library cells. Our approach combines network re-structuring, AIG nodes reordering, dynamic power and longest path delay estimation and optimisation and finally technology mapping to an AIG netlist. A set of MCNC Benchmark circuits and large combinational circuits up to 100,000 gates have been used to validate our methodology. Comparisons for power and delay optimisation are made with the best synthesis scripts used in ABC. Reduction of 23% in power and 15% in delay with minimal overhead is achieved, compared to the best known ABC results. Also, our approach is also implemented on a number of processors with combinational and sequential components and significant savings are achieved.

## Acknowledgement

First and foremost, all thanks are due to God Almighty for the innumerable blessings through my life, one of which is this thesis.

My thesis supervisor, Dr. Emanuel M. Popovici, deserves a special acknowledgement for his vision, guidance and support throughout the research work.

I would like to express my sincere thanks to my parents and my brother who have installed in me the courage to continue with higher education.

I would like to thank my fellow Ph.D. students in the lab, Shraddha, Nasim, Richard, Tom, Chen, Stevan and Christian for the interesting discussions and support through these four years.

I gratefully acknowledge the support of IDA Ireland and Synopsys for the financial support of this work.

Lastly to my fianc Vivek who has always been a tremendous source of encouragement, confidence and love.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Binary Decision Diagrams (BDDs) . . . . .	5
2.2	AND-Inverter Graphs (AIGs) . . . . .	6
2.3	NAND-NOR-Inverter Graphs (NNIGs) . . . . .	7
2.4	Review of Academic Electronic Design Automation (EDA) tools . . . . .	9
2.4.1	CUDD package . . . . .	9
2.4.2	Folded Binary Decision Diagrams (FBDDs) . . . . .	9
2.4.3	A System for Sequential Circuit Synthesis (SIS) . . . . .	9
2.4.4	A System for Sequential Synthesis and Verification (ABC) . . . . .	10
2.5	Power and delay optimisation techniques . . . . .	10
<b>3</b>	<b>Digital power estimation flow</b>	<b>13</b>
3.1	Introduction . . . . .	13
3.2	Power dissipation in CMOS circuits . . . . .	14
3.3	Industrial Design Flow for Power Estimation . . . . .	15
3.4	Academic Design Flow for Power Estimation . . . . .	17
3.4.1	Structural Information . . . . .	18
3.4.2	Details of the design flow . . . . .	18
3.5	Integrated Design Flow . . . . .	20
3.5.1	Tool Flow . . . . .	21
3.5.2	Key features of the integrated design flow . . . . .	21
3.6	Experimental Results . . . . .	22
3.7	Conclusions . . . . .	22

<b>4</b>	<b>Graph manipulation for both power and delay analysis</b>	<b>24</b>
4.1	Introduction . . . . .	24
4.2	Background of data structures . . . . .	26
4.3	DSM tool . . . . .	27
4.4	Experimental Results for AIG vs. NNIG . . . . .	28
4.5	Probabilistic Timed NNIGs (PTNNIGs) . . . . .	30
4.5.1	Background . . . . .	30
4.5.2	NNIG and PTNNIG . . . . .	31
4.5.3	Mapping of Probability and Delay on PTNNIGs . . . . .	31
4.6	Experimental Results for PTNNIG . . . . .	34
4.7	Conclusions . . . . .	36
<b>5</b>	<b>Power and delay estimation and optimisation on AIGs</b>	<b>39</b>
5.1	Introduction . . . . .	39
5.2	Switching probability estimation under the zero-delay model . . . . .	40
5.3	Longest path delay and switching power estimation under a non-zero-delay model . . . . .	42
5.3.1	Longest path delay estimation . . . . .	42
5.3.2	Dynamic power estimation under given delay model . . . . .	43
5.4	Reordering rules for power optimisation . . . . .	46
5.4.1	Rule $Rp_1$ . . . . .	46
5.4.2	Rule $Rp_2$ . . . . .	49
5.4.3	Rule $Rp_3$ . . . . .	51
5.4.4	Rule $Rp_4$ . . . . .	53
5.5	Reordering rules for delay optimisation . . . . .	55
5.5.1	Rule $Rd_1$ . . . . .	55
5.5.2	Rule $Rd_2$ . . . . .	56
5.5.3	Rule $Rd_3$ . . . . .	57
5.6	Conclusions . . . . .	59
<b>6</b>	<b>Implementation of reordering rules for power and delay optimisation</b>	<b>61</b>
6.1	Implementation of low power optimisation rules using Greedy Algorithm	62
6.2	Architecture and application of AIG2Net . . . . .	63

6.2.1	Estimation/Optimisation Design Flow . . . . .	66
6.3	Delay-driven power optimisation and power-driven delay optimisation . . . . .	67
6.4	Simulated Annealing . . . . .	69
6.5	Switching nodes order for OPT-PT application . . . . .	75
6.5.1	Application of Uniform Cost Search algorithm . . . . .	75
6.5.2	Matrices and graphs during OPT-P and OPT-T . . . . .	78
6.5.3	Comparison of Uniform Cost Search (UCS) algorithm with previous methods in OPT-P tool application . . . . .	80
6.6	Conclusions . . . . .	82
<b>7</b>	<b>Experimental Results using OPT-PT</b>	<b>85</b>
7.1	Simulation Design Flow . . . . .	86
7.2	Results with OPT-P application . . . . .	87
7.3	Results with OPT-T application . . . . .	89
7.4	Results after technology mapping by Synopsys Design Compiler . . . . .	92
7.5	Power-Time (P-T) curve . . . . .	94
7.6	Experimental results on sequential circuits . . . . .	95
7.7	The impact on switching power corresponding to various input data sets . . . . .	99
7.8	Conclusions . . . . .	103
<b>8</b>	<b>Combinatorial optimisation techniques for dynamic power reduction</b>	<b>105</b>
8.1	Introduction . . . . .	105
8.2	Multi-objective power-delay optimisation is NP-Complete . . . . .	106
8.3	The Local Search Method . . . . .	108
8.4	The Tabu Search Method . . . . .	110
8.5	The Simulated Annealing Search . . . . .	110
8.6	The Partitioned Random Search . . . . .	113
8.7	Experimental Results . . . . .	113
8.8	Conclusions . . . . .	117
<b>9</b>	<b>Conclusions</b>	<b>118</b>
9.1	Contributions . . . . .	118
9.2	Future Work . . . . .	120





# List of Figures

2.1	A BDD example for a circuit with functionality $abc + a'b'$ . . . . .	6
2.2	An AIG example for a circuit with functionality $abc + a'b'$ . . . . .	7
2.3	A NNIG example for a circuit with functionality $abc + a'b'$ . . . . .	8
3.1	Industrial Design Flow . . . . .	16
3.2	Academic Design Flow . . . . .	19
3.3	Proposed Integrated Design Flow . . . . .	20
4.1	DSM tool flow . . . . .	27
4.2	A NNIG example for a circuit with functionality $((c + d)Xa)Xb$ . . . . .	32
4.3	BLIF file of the circuit representing the logic function $ab(c + d)$ . . . . .	33
4.4	Verilog file of the circuit representing the logic function $ab(c + d)$ . . . . .	33
4.5	SAIF . . . . .	35
4.6	SDF . . . . .	36
4.7	Mapping of Probability and Delay issues on PTNNIGs . . . . .	37
5.1	AIG of logic function $ab$ . . . . .	41
5.2	AIG of logic function $abc$ . . . . .	44
5.3	Rule $Rp_1$ on AIG . . . . .	47
5.4	Rule $Rp_2$ on AIG . . . . .	50
5.5	Rule $Rp_3$ on AIG . . . . .	52
5.6	Rule $Rp_4$ on AIG . . . . .	54
5.7	Rule $Rd_1$ on AIG . . . . .	56
5.8	Rule $Rd_2$ on AIG . . . . .	58
5.9	Rule $Rd_3$ on AIG . . . . .	60
6.1	AIG of logic function $a(b + c)$ . . . . .	64

6.2	Mapped Verilog netlist of an AIG network . . . . .	65
6.3	User defined switching activity file for the netlist shown in Figure 6.2 . . . . .	65
6.4	The design flow for power, delay and area estimation . . . . .	66
6.5	The graph $G(V, E)$ corresponding to the circuit <b>x2</b> . . . . .	79
6.6	The graph $G(V, E)$ corresponding to the circuit <b>count</b> . . . . .	81
6.7	The Uniform Cost Search Algorithm . . . . .	82
7.1	The ROM Look-Up Table . . . . .	86
7.2	The graph for power reduction among various circuits . . . . .	88
7.3	The graph for delay reduction among various circuits . . . . .	93
7.4	The P-T curve on $R_{i10,o12}$ . . . . .	96
7.5	High level view of a Processor . . . . .	97
7.6	Sequential AIG of a 3-bit shift register . . . . .	97
7.7	Flat <i>SAP</i> . . . . .	100
7.8	Decreasing $\text{Log}_2$ <i>SAP</i> . . . . .	100
7.9	Random <i>SAP</i> . . . . .	100
8.1	Local nodes within an AIG network . . . . .	108

# List of Tables

3.1	Power dissipation with and without SWF . . . . .	23
4.1	Power estimation on AIG and NNIG . . . . .	29
4.2	Power comparisons against ABC and DSM . . . . .	30
4.3	Power estimation against BLIF and optimised BLIF . . . . .	38
6.1	$SP(f)$ without and with the application of rewriting rules $Rp_1, Rp_2, Rp_3$ and $Rp_4$ . . . . .	63
6.2	Power dissipation with and without the RESWITCH tool . . . . .	68
6.3	The Matrix Order $init_{SNx}[y]$ . . . . .	80
6.4	The comparison of $T_1, T_2$ and $T_3$ used on DES using OPT-P application	82
6.5	The comparison of $T_1, T_2$ and $T_3$ used on $R_{i8,o10}$ using OPT-P application	83
6.6	The comparison of $T_1, T_2$ and $T_3$ used on <code>pair</code> using OPT-P application	83
7.1	The comparison of OPT-P with the best algorithm in ABC (65nm pro- cess) . . . . .	88
7.2	Computation time of OPT-P with respect to <code>resyn</code> script of ABC . . .	89
7.3	Dynamic and Static Power for the simulations stated in Table 7.1 (65nm process) . . . . .	90
7.4	The comparison of OPT-P with the best algorithm in ABC (90nm pro- cess) . . . . .	91
7.5	Dynamic and Static Power for the circuits stated in Table 7.4 (90nm process) . . . . .	92
7.6	The comparison of OPT-T with ABC . . . . .	93
7.7	Computation time of OPT-T with respect to <code>resyn</code> script of ABC . . .	94
7.8	Results using Synopsys Design Compiler mapping via OPT-P . . . . .	95

7.9	Sequential circuit synthesis results for selected MCNC circuits . . . . .	98
7.10	The comparison of OPT-P with ABC, (65nm process) . . . . .	99
7.11	Power reduction on the combinational sub-network for a set of sequential circuits . . . . .	99
7.12	Impact on power reduction under various input data sets (implemented on $6 \times 4$ bit multiplier) . . . . .	102
7.13	Impact on power reduction under various input data sets (implemented on 5 bit adder ) . . . . .	103
8.1	Power comparisons among LS, TS, SS and PS . . . . .	115
8.2	Delay comparison among LS, TS, SS and PS . . . . .	116
8.3	Comparison of computation time using LS, TS, SS and PS . . . . .	116

## Appendix

The thesis is based in part on the following publications:

1. R. Mehrotra, T. English, K.L. Man, E. Popovici and M. Schellekens, “Digital power estimation flow combining academic and industrial tools”, IEEE International Soc Design Conference (ISOCC) Busan, Nov 2008, pp. 89-92.
2. R. Mehrotra, K.L. Man, E. Popovici and M. Schellekens, “Data Structure Manipulation for NNIG and PTNNIG: Towards a Unified Power and Timing Analysis”, IEEE 3rd International Conference on Signals, Circuits and Systems Tunisia, Nov 2009, pp. 1-6.
3. R. Mehrotra, E. Popovici, K. L. Man and M. Schellekens, “Power reduction and technology mapping of digital circuits using AND-Inverter Graphs”, IEEE 27th International Conference on Microelectronics (MIEL) Serbia, May 2010, pp. 295-298.
4. R. Mehrotra, T. English, E. Popovici and M. Schellekens, “Delay dependent power optimisation of combinational circuits using AND-Inverter Graphs”, IEEE International SOC conference (SOCC) Las Vegas, Sept 2010, pp. 9-14.
5. R. Mehrotra, T. English, M. Schellekens, S. Hollands and E. Popovici, “Timing-driven Power Optimisation and Power-driven Timing Optimisation of Combinational Circuits”, Journal of Low Power Electronics (JOLPE), August 2011, Vol. 7, No. 3, pp. 364-380.

# Chapter 1

## Introduction

In recent years, with the development of electronic devices such as cellular phones, smart phones, laptops, tablets and mobile multimedia systems, energy dissipation has become a critical parameter in digital VLSI design. In battery operated system such as portable systems, the amount of energy stored within the battery is limited. Hence power dissipation is important as it defines the runtime of the device on a single battery charge.

With the advances in CMOS fabrication technology, the number of transistors per chip and operating frequency is increasing every year. Consecutively, the power dissipation per unit area is also growing thereby increasing the chip temperature. This excessive temperature reduces the reliability and lifetime of the circuit, and a large cooling system and expensive packaging is required to dissipate the extra heat. Hence low power design is crucial in today's circuit design in Deep Sub-Micron (DSM) technology.

However, although power dissipation is important in portable systems, the delay parameter is an equally important constraint for digital designers as the energy consumption is the product of power dissipation times delay (longest path delay of the circuit). Tight energy constraints are commonplace in many modern VLSI applications. Consumers expect higher speed, more functionality and higher levels of integration, from their cellular phones and hand-held devices. Low power design is not effective if it increases the delay (performance of the circuit) largely. Similarly fast circuit design (longest path delay optimisation) is not fruitful if it increases power dissipation significantly. Hence one needs to explore optimisation schemes for power with minimal overhead of delay, and optimisation schemes for delay with minimal overhead with re-

spect to power.

Market competition and wide-spread use of mobile equipment has resulted in ever-tightening constraints for power, delay and area. This has led to a drastic increase in design complexity. In order to handle the ever increasing complexity, Computer-Aided Design (CAD) tools have been developed to produce circuit design solutions for a wide range of applications. These tools are used for logic synthesis to obtain optimised logic circuits from some input-output specification. The logic synthesis process is split up into two different phases as explained in [35]. First logic optimisation is performed on the Boolean description of the circuit. Next, the technology mapping is performed on this optimised circuit that translates the generic Boolean description to logic gates or cells existing in the chosen library. This library is specified to the fabrication process that is used and has precise layout, area and delay information for each gate or cell.

There are various Electronic Design Automation (EDA) tools available, some of which are commercial tools (for e.g. Synopsys Design Compiler and Synopsys Prime Time) and some are open-source academic tools (ABC, SIS and FBDD) used in research for logic synthesis, power, delay and area optimisation, technology mapping, etc, of digital circuits. The academic tools are open-source tools and many new optimisation techniques for digital circuits are introduced by researchers within these tools. One objective of this dissertation is to introduce a set of rules (within the state of the art academic open-source EDA tool ABC [11]) applied on the nodes of the Boolean network representation of a digital circuit represented as AIG network such that power and delay is optimised. We have introduced the concept of delay-driven power optimisation and power-driven delay optimisation. For delay-driven power optimisation, firstly we assume that the increase in longest path delay shall not exceed a given constraint while optimising for dynamic power. Secondly for each application of the reordering rules on the AIG nodes, a weight is calculated which is directly proportional to the decrease in switching power and inversely proportional to the increase in delay. These weights keep a check on every bad move where delay increases largely in the process of switching power reduction. Similarly, power-driven delay optimisation is defined by assuming that firstly, the increase in dynamic power shall not exceed a given constraint while optimising for delay. Secondly for each application of the reordering rules on the AIG nodes, a weight is calculated which is directly proportional to the decrease in delay



and inversely proportional to the increase in dynamic power. Hence multi-objective optimisation of two conflicting objectives (namely power and delay) subjected to each other's constraints, is considered in our research. The optimisation method uses two optimisation algorithms namely Simulated Annealing and Uniform Cost Search Algorithm.

The synthesis techniques must not only optimise the above parameters (power and delay) but also lead to a better circuit (with respect to total power and longest path delay) in as little time as possible. The techniques introduced in our work for synthesis and optimisation have a low computation time. Their runtime ranges from 1.2 to 1.8 times when compared to the ABC synthesis scripts, with some significant reduction in power and delay.

The advances in process technology typically come with a host of new design variables and hence a new set of challenges to the designers (such as optimisation of the leakage power). Hence the CAD tools must handle such new design parameters as well. One can also introduce an area constraint during power or delay optimisation. The synthesis and optimisation methods introduced in this work give a constraint to the area (number of AND nodes of the AIG network) while optimising power and/or delay. This keeps a bound on the total silicon area and leakage power of the circuit while optimising for dynamic power or longest path delay.

The thesis is organised as follows: Chapter 2 gives some background on various data structures used within the CAD tools for the synthesis and optimisation of logic circuits. This is followed by a review of some academic EDA tools which use these data structures. Existing power and delay reduction techniques are also discussed in this chapter. Chapter 3 describes various design flows of digital circuits including industry and academic. This design flow converts the RTL description of the circuit to the gate-level netlist, followed by the generation of power, delay and area reports. The chapter presents an industrial design flow using industrial EDA tools, an academic design flow using academic research EDA tools and an integrated design flow. Although many techniques applied in the academic tools demonstrate robustness and optimality in EDA, the tools have not found application in industry and can only be considered theoretical frameworks. Hence we have introduced an integrated design flow which links the two design flows, for improved and simplified application of academic tools in the industrial

domain.

Various data structures used by academic EDA tools such as NAND-NOR-Inverter Graphs (NNIGs) are being presented. These graphs have gained much importance in recent years for power and delay optimisation. In Chapter 4, we introduce a novel tool for the manipulation of NNIGs. We also introduce a new data structure known as Probabilistic Timed NNIG (PTNNIG) in this chapter, which has switching probabilities and delay values annotated on each node of the NNIG. This data structure can be used for power and delay optimisation. The PTNNIG is the basis to achieve delay-driven power optimisation and power-driven delay optimisation.

In Chapter 5, firstly we estimated switching power and longest path delay of an AIG network representing the functionality of a digital circuit. We introduce then various reordering and re-structuring rules applied on each of the nodes of the AIG network for power and delay optimisation. The reordering rules applied on AIGs help in reducing the switching power and the delay of the network. The problem of glitches is also dealt in our work, while estimating the switching power on the AIG network.

Chapter 6 deals with the implementation of the reordering rules. Various optimisation algorithms are also introduced which implement these rules. A simulation design flow is also presented which allows us to obtain power, delay and area reports of the optimised AIG network using industrial tools. The chapter also introduces the application of two algorithms ‘Simulated Annealing’ and ‘Uniform Cost Search Algorithm’ which are used in delay-driven power optimisation and power-driven delay optimisation. We introduce a new tool OPT-PT which is sub-divided into two tools, OPT-P for delay-driven power optimisation and OPT-T for power-driven delay optimisation. The tool is implemented in C as a sub-package in ABC.

Chapter 7 presents a list of experimental results on large combinational circuits with node count ranging from 1,000 to 100,000 gates. OPT-PT tool is implemented on large combinational and sequential circuits and the optimised power and delay results are reported.

Chapter 8 gives a brief overview of the various combinatorial optimisation techniques that are incorporated in our tool OPT-PT. Finally, Chapter 9 presents conclusions and topics for future work.

# Chapter 2

## Background

Various data structures which are used within the Electronic Design Automation (EDA) tools for synthesis and optimisation of digital logic circuits are described in this chapter. The data structures represent the functionality of the logic circuit. A list of academic EDA tools are also covered. In the last section, some state of the art power and delay reduction techniques for digital circuits using data structures, are reviewed.

### 2.1 Binary Decision Diagrams (BDDs)

Binary Decision Diagrams (BDDs) [13] can be represented as a rooted, directed acyclic graph denoted as Graph  $G(V, E)$  where  $V$  is the set of vertices and  $E$  is the set of edges which link the vertices denoted in  $V$ . Any  $v \in V$  is either a non-terminal node or one of the two terminal nodes namely 0-terminal and 1-terminal. The non-terminal nodes are the decision nodes which are labeled by a Boolean variable. These nodes have two child nodes called low child and high child. Any  $e \in E$  from the decision nodes point to the low child or high child depending upon the assignment of the Boolean variable labeled on the decision node to be '0' or '1' respectively. The starting decision node of the graph is called the root node. This forms a BDD.

Such a BDD is called 'ordered' if all the paths from the root node to the terminal nodes have the same Boolean variable order. A BDD is said to be 'reduced' if the following two rules have been applied to its graph. Firstly, there exists no two sub-graphs expressing the same function i.e. no two sub-graphs are graph isomorphic. Secondly, there exist no two decision nodes whose two children are isomorphic.

In popular usage, the term BDD almost always refers to Reduced Ordered Binary Decision Diagram (ROBDD). One advantage of a ROBDD is that it is canonical (unique) for a particular function and variable order. This property makes it useful in functional equivalence checking and other operations like functional technology mapping.

Figure 2.1 shows a BDD graph  $G(V, E)$  where  $V$  is the set of decision nodes and terminal nodes. The decision nodes are represented with a circle and terminal nodes are represented with a rectangle. The decision nodes represent Boolean variables  $a$ ,  $b$  and  $c$  representing the functionality  $abc + a'b'$  of the circuit. In the figure, the light edges point to the low child i.e assignment of the Boolean variable to '0'. The highlighted edges point to the high child i.e. assignment of the Boolean variable to '1'. In the figure, the functionality at each decision node is also mentioned.

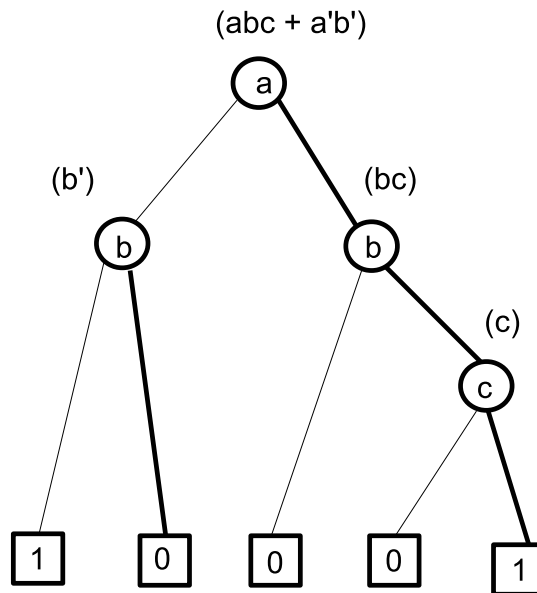


Figure 2.1: A BDD example for a circuit with functionality  $abc + a'b'$

## 2.2 AND-Inverter Graphs (AIGs)

An AND-Inverter Graph (AIG) [48] is a directed acyclic graph denoted as Graph  $G(V, E)$  where  $V$  is the set of vertices and  $E$  is the set of edges which link any two vertices. The graph represents the structural functionality of the digital logic circuit.

Any  $v \in V$  either represents two-input AND gate or terminal node labeled with a variable name. The variable names are the names of the input variables of the logic circuit. Any  $e \in E$  is the input to the two-input AND gate. Figure 2.2 shows an example of AIG, representing the functionality  $abc + a'b'$  of the circuit. The inputs are  $a$ ,  $b$  and  $c$ . The inputs are represented as the terminal nodes denoted within a triangle. The nodes which represent two-input AND gates are denoted within a circle.  $\times$  represent that those nodes are 2-Input AND nodes. Edges with a dot indicate negation i.e. inversion of that input. Here the term ‘vertices’ and ‘nodes’ imply the same. AIGs are simple and flexible data structures which promise improvements in quality and runtime of several applications. The runtime of the combinational synthesis and mapping is typically faster using AIGs. AIGs provide a much compact representation of digital circuits than BDDs. Formal verification is simpler as they can be easily mapped to a netlist.

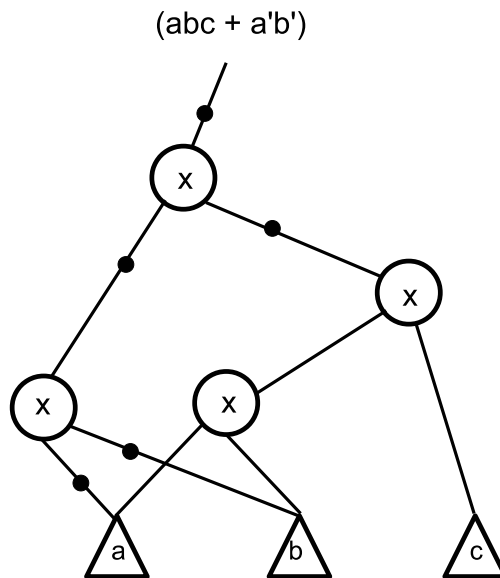


Figure 2.2: An AIG example for a circuit with functionality  $abc + a'b'$

### 2.3 NAND-NOR-Inverter Graphs (NNIGs)

NAND-NOR-Inverter Graph (NNIG) [3] is also a directed acyclic graph denoted as Graph  $G(V, E)$  where  $V$  is the set of vertices and  $E$  is the set of edges which link

any two vertices. The graph represents the structural functionality of the digital logic circuit. Any  $v \in V$  either represents two-input NAND gate, two-input NOR gate or a terminal node labeled with a variable name. The variable names are the names of the input variables of the logic circuit. Any  $e \in E$  is the input to the two-input NAND gate or two-input NOR gate. Figure 2.3 shows an example of NNIG, representing the functionality  $abc + a'b'$  of the circuit. The inputs are  $a$ ,  $b$  and  $c$ . The inputs are represented as the terminal nodes denoted within a triangle.  $\otimes$  and  $\oplus$  represent those nodes as 2-Input AND gates and 2-input OR gates respectively. The symbol  $\otimes$  linked with the notation  $\bullet$  indicates a NAND, similarly the symbol  $\oplus$  linked with the notation  $\bullet$  indicates a NOR. Edges with a dot indicate negation i.e. inversion of that input. NNIGs also provide a compact representation of the digital circuit and are immensely used in logic synthesis, technology mapping and formal verification.

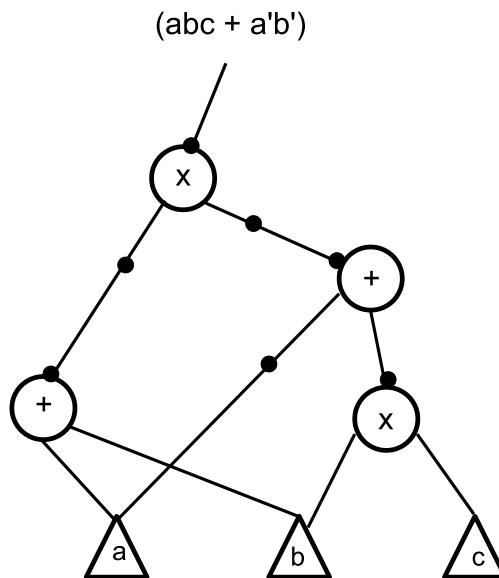


Figure 2.3: A NNIG example for a circuit with functionality  $abc + a'b'$

Another data structure is the NAND-Inverter Graphs (NIGs). It is represented as a directed acyclic graph consisting of 2-input NAND gates and Inverters only.

## **2.4 Review of Academic Electronic Design Automation (EDA) tools**

A number of academic EDA tools [49], [59], [9], [76], [77], [79] and [11] have been proposed in the literature. These open-source tools provide a programming environment and a solid platform for research in logic synthesis, technology mapping, power and delay estimation and optimisation. These academic tools represent the Boolean functionality of any digital circuit using a data structure. Manipulation for logic synthesis, optimisation and technology mapping are done on these data structures. Although many techniques applied in the tools demonstrate robustness and optimality in EDA, the tools have not found application in industry and can only be considered theoretical frameworks. Some academic EDA tools with their data structure are presented below.

### **2.4.1 CUDD package**

The CUDD package provides functions to manipulate Binary Decision Diagrams (BDDs) [13] and other decision diagrams [81]. The size of the BDD is largely governed by the order of the input variables. The package also provides a large assortment of variable reordering methods, to reduce the number of BDD nodes.

### **2.4.2 Folded Binary Decision Diagrams (FBDDs)**

FBDD [79] is an open-source logic synthesis package based on BDDs. The package employs several new techniques, including folded logic transformations and two-variable sharing extraction. The primary objective of the package is to scale the logic synthesis algorithm runtime by one order of magnitude, while producing competitive synthesis quality.

### **2.4.3 A System for Sequential Circuit Synthesis (SIS)**

SIS [59] is another interactive tool for synthesis and optimisation of sequential circuits. Given a state transition table, a signal transition graph, or a logic-level description of a sequential circuit, it produces an optimised netlist in the target technology while preserving the sequential input-output behavior. The input specifications in SIS are STG (State Transition Graph) and ASTG (Asynchronous Signal Transition Graph).

#### 2.4.4 A System for Sequential Synthesis and Verification (ABC)

ABC [11] is a logic synthesis and verification tool which performs scalable logic optimisation based on AND-Inverter Graphs (AIGs) [48]. In all of these academic tools, data structures and algorithms largely determine the efficiency of the tool in implementing new applications. Previous tools like SIS [59], BDD [13] and FBDD [79] get inefficient for large circuits and do not provide enough flexibility for binary synthesis, owing to their type of data structures. ABC deals with simple and flexible data structures, i.e. AIGs, which promise improvements in quality and runtime of several applications. Our major work is based on AND-Inverter Graphs, manipulated in ABC.

### 2.5 Power and delay optimisation techniques

There are three main sources of power dissipation in digital CMOS circuits [14] namely short-circuit power, leakage power and switching power. A detailed description of each of the components is given in Chapter 3. In our work we have mainly targeted switching power optimisation. The switching power component is often the dominant component however in deep sub-micron technology, leakage power is also quite significant. Therefore, in the experimental results we have tabulated leakage power component as well. The switching power dissipation in a CMOS gate is dependent on the switching activity at the circuit nodes, capacitive load, supply voltage and operating frequency. Various techniques which aim at the reduction of switching power and delay are presented below.

Previously, various probabilistic techniques have been used to estimate and minimise the switching activity of the circuit based on data structures and graphs. Binary Decision Diagrams (BDDs) have been used extensively in [78], [34], [38], [19], [36] and [74] for switching activity estimation and optimisation. In these works, the zero-delay model is assumed and reduced switching activity is obtained either by reducing the BDD size or by changing the order of the BDD. The main drawback of these techniques is their computational complexity. Computation time and memory footprint increase rapidly with the circuit size.

The probabilistic technique for switching activity estimation using BDD follows a BDD traversal method [74]. The BDD structure involved in this traversal method in-



creases rapidly with circuit size, hence making it cumbersome for large circuits. Many techniques in [69], [26], [27] and [25] work on reducing the BDD size as used in the BDD traversal method in order to reduce the memory usage and computation time. Several other power-aware BDD-based synthesis techniques are introduced in [54] and [73]. Another BDD based synthesis technique for low power is introduced in [70] which is based on split BDDs, which disables parts of the circuit under useless switching (parts of the circuit not being used but still dissipating power due to switching), however the delay parameter is again ignored.

Several new academic tools have been developed for switching activity and power estimation including BLAPE [78] and POSE [34]. These tools are mainly used for behavioural and gate-level designs. They assume the zero-delay model, resulting in inaccurate power estimation. The contribution of glitches - spurious transitions due to inertial delays and differential gate delays - is not captured. Further, when optimising power, these tools ignore other factors such as delay, area and power-delay product. Plotting a power-delay curve illustrates how power reduction has impacted on circuit delay. In this context, work has been done on delay-dependent power estimation, but optimisation remains still an issue [62].

In [23], a gate-level power estimation technique under a delay model, based on tagged probabilistic simulation (TPS) is introduced, which proved to be efficient and accurate even for large circuits. However, power optimisation is not considered. An extension of [74] is introduced in [75] which works on the estimation of switching activity under a delay model, followed by dynamic power estimation and optimisation. However, a unit-delay model has been assumed rather than a variable delay model and the overhead parameters are not discussed.

To improve the synthesis process, several efficient and compact data structures have been introduced to represent the functionality of combinational circuits, such as AIGs [48], NNIGs [3], [6] and NIGs [4] as discussed in the previous section. Each data structure focuses on capturing either power or delay in a graph representation for further manipulation. The academic tool ABC [11] is used for the manipulation of AIGs but offers minimal support for power estimation and optimisation. AIGs are used in [44] and [50] mainly to make the logic synthesis flow more scalable and faster, and to unify and enhance many phases of logic synthesis. AIGs are also used in [12] which intro-

duces various local two-level algebraic rules applied on the AIG network for reducing the number of AIG nodes. A tool was also introduced in [42] for NNIG manipulation. Works in [3] and [5] largely deal with AIGs, NNIGs and NIGs comparisons among each other with respect to power or delay. A power optimisation tool box which uses AIGs for logic synthesis and mapping introduced in [46] largely motivated us to use AIGs in our research work. Works in [71], [72], [53], [64] and [55] introduce methods for switching activity estimation using fast, accurate, delay-dependent methods under a variable-delay model. The above works underpin the research presented in this thesis.

For delay optimisation, many techniques have been developed [61], [15]. However they did not provide a clear link with the power dissipation due to the delay (critical path length) reduction. Work in [20] is based on delay- driven optimisation, however power values increase largely in the simulation results. Research in [21], [80], [39], [47] and [45] deal with delay and area optimisation, but the power is still left out. None of the techniques mentioned above focus on multi-objective optimisation such as joint power and delay optimisation.

Finally, many techniques used in [68], [2], [37], [51] and [16] which focused on power and delay optimisation, motivated us for the research indicated in this thesis. However, these techniques do not follow delay-driven power optimisation and power-driven delay optimisation as described earlier. Reviewing various power and delay reduction techniques introduced to date, new techniques and methodologies implemented on AIGs are introduced in our work for delay-driven power optimisation and power-driven delay optimisation. The synthesis method is based on AIGs which represent the functionality of the circuit.

## Chapter 3

# Digital power estimation flow

### 3.1 Introduction

Over the years, a number of academic EDA tools [49], [59], [9], [76], [77], [79] and [11] have been proposed in the literature. These open-source tools provide a programming environment and a solid platform for research in logic synthesis, power and delay optimisation and technology mapping. However, the tools do not support constructs from more modern HDLs such as Verilog-2001 and System Verilog. Only a restricted subset of language constructs in Verilog and VHDL are supported for input and output. Basic random-pattern simulation is supported, but complex test benches with user-supplied test vectors and assertions are not. The academic tools only include generic technology libraries, and data exchange between academic tools is complicated by occasional file format incompatibilities.

Several academic tools used for logic synthesis, power estimation and power optimisation are surveyed with the aim of building a better understanding of the theoretical foundations behind such tools as well as motivating research towards an integration of academic tools into an industrial design flow.

In this chapter, we propose a design flow which translates the RTL description of the circuit to a gate-level netlist, using several well-known academic tools, and is integrated into an industrial design flow. This integration will be beneficial to researchers in logic synthesis, power estimation and power optimisation. The aims of the integration are to relate academic tools to an industrial design flow and to bridge gaps between the academic and industrial design flows (new tools have been developed to achieve this). It

also intends to make the academic tools easier to deploy and to use by connecting them together with a set of customisable Unix shell scripts. This will create an open-source, academic tool flow within which new techniques can be developed and compared. Also data from the industrial design flow (such as event-driven simulation results) can be used to support the development of logic synthesis, power estimation and power optimisation techniques in the academic domain.

The remainder of the chapter is organised as follows. Section 3.2 gives an overview of power dissipation in CMOS circuits. A standard industrial design flow and the proposed academic design flow for power estimation are presented in Sections 3.3 and 3.4, respectively. Section 3.5 outlines an integrated design flow and the motivation for its development. Experimental results (enabling power saving of over 40%) are provided in Section 3.6 to demonstrate the applicability and effectiveness of such an integrated design flow. Finally, concluding remarks are made in Section 3.7.

## 3.2 Power dissipation in CMOS circuits

There are three main sources of power dissipation in digital CMOS circuits [14]:

- short-circuit power;
- leakage power;
- switching power;

The short-circuit power is due to the direct-path short circuit current, which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. Leakage power which arises from substrate injection and sub-threshold effects, is primarily determined by fabrication technology considerations.

The switching power dissipation in a CMOS gate is due to charging and discharging of load capacitance driven by gate. The capacitance consists of the internal capacitance of the gate, wire capacitance of the fanout net and the capacitances of the gate terminals of the transistors being controlled by the fanout net. The sum of switching power and short circuit power is referred as the dynamic power of the CMOS circuit. The switching

power dissipation referred as dynamic power is calculated by the following equation:

$$P_{dynamic} = \alpha C_l V_{dd}^2 f_{clk} \quad (3.1)$$

where  $\alpha$  is the switching activity factor (transition probability of switching from 0 to 1 or 1 to 0),  $C_l$  is the overall capacitance to be charged and discharged in a reference clock cycle,  $V_{dd}$  is the supply voltage and  $f_{clk}$  is the clock frequency. The dynamic power dissipation of a CMOS logic circuit with the set of gates  $S$  is given by the equation:

$$P_{dynamic} = V_{dd}^2 f_{clk} \cdot \sum_{G \in S} C_G \alpha_G \quad (3.2)$$

where  $C_G$  is the load capacitance at gate  $G$  and  $\alpha_G$  is the switching activity factor at gate  $G$ . When  $f_{clk}$  and  $V_{dd}$  are given,  $\sum_{G \in S} C_G \alpha_G$  needs to be computed for evaluating  $P_{dynamic}$ . Throughout the thesis, this product has been used as the parameter for power dissipation. Thus an accurate estimation of the transition density in a circuit (product of capacitive load and transition probability) will give an accurate estimation of dynamic power dissipated in the circuit. The transition probability or the switching probability  $\alpha_G$  is a function of transition probabilities of primary inputs, logical behaviour and design style of the CMOS circuit. Several low power techniques have concentrated on reducing dynamic power. Dynamic power (switching power) is classified into necessary transition for correct functioning of the circuit and unnecessary transition due to unbalanced paths in the circuit. This latter component of the power dissipated is known as glitching power. Glitches [18] are unnecessary transitions produced in the circuit due to the difference in signal arrival times at the inputs of a gate. The power dissipated due to glitches is glitching power.

### 3.3 Industrial Design Flow for Power Estimation

The industrial design flow is presented in Figure 3.1. The design flow outlines the path of a digital circuit from the RTL level to the gate-level netlist followed by power, delay and area reports. The tools used in this flow are all commercial EDA tools and hence the name industrial design flow. In the flow, the input is a RTL description of a circuit and the outputs are power, delay and area reports.

In this flow, RTL simulation is performed using VCS, a commercial tool from EDA vendor Synopsys [67]. VCS (like some other commercial simulators) supports Verilog

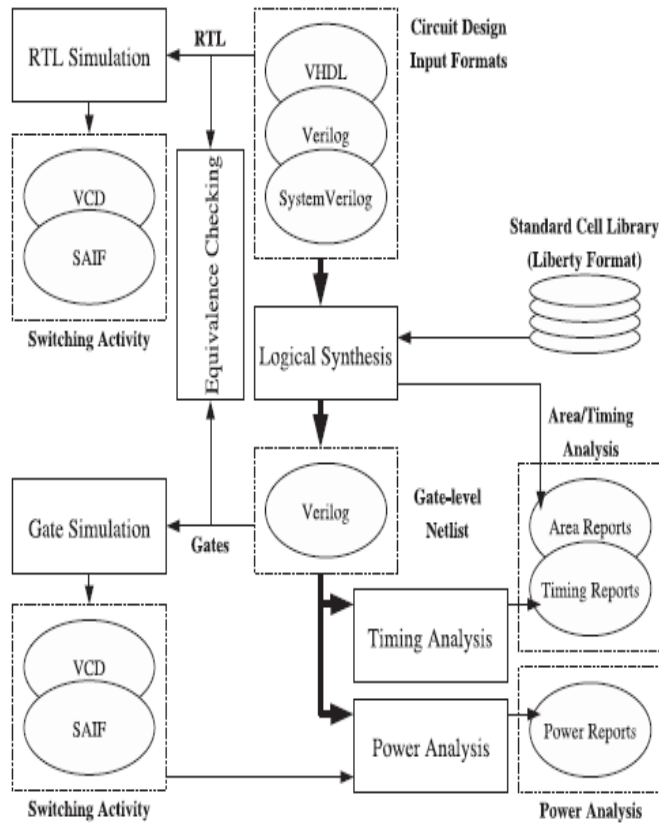


Figure 3.1: Industrial Design Flow

and VHDL as well as more recent IEEE-standard HDLs such as Verilog-2001 and SystemVerilog. The logic description is constrained and then synthesised to a gate-level netlist targeting our chosen standard cell library in Design Compiler, tool from Synopsys. Formal verification is performed in Formality, tool from Synopsys, ensuring equivalence between RTL and the resulting gate-level netlist. Reports generated by the synthesis tool detail the silicon area consumed by the logic design. Delay reports are generated using PrimeTime, tool from Synopsys determining whether the netlist meets constraints laid down prior to synthesis. PrimeTime also generates Standard Delay Format (SDF) [58] containing delay information for netlist annotation during gate-level simulation. The primary goal of constructing this flow was to perform power analysis on the pre-placement netlist using commercial tools. Switching probability information

is recorded by VCS during simulation as Switching Activity Information File (SAIF) [57]. Time domain value-change data are recorded as Value Change Dump (VCD) [33]. SAIF and VCD are used during power analysis to obtain realistic power figures for the simulated scenario. Two power analysis flows are used:

1. SAIF switching probabilities from RTL simulation are annotated onto the netlist in Power Compiler from Synopsys [67]. The tool propagates switching activity through the netlist and reports average power for the simulated scenario. (SAIF from gate-level simulation can also be used, if available, for more accurate results at the expense of processing time).
2. The netlist is simulated in PrimeTime PX, tool from Synopsys using value-change data from gate-level VCD. Using VCD allows both average and instantaneous (peak) power reports to be generated, along with time domain power waveforms from the simulation.

### 3.4 Academic Design Flow for Power Estimation

A typical academic logic synthesis tool first constructs a two-level or multi-level Boolean network corresponding to the RTL description of a design. This network is then optimised using various technology-independent techniques. Finally, technology mapping transforms the technology independent circuit into a network of gates in a given technology. Mapping is constrained by several factors including the available gates in the technology library, the drive sizes of each gate and the delay, power and area characteristics of each gate.

Binary Decision Diagrams (BDDs) [13] and AIGs [48] have become very effective representations for Boolean functions in VLSI/CAD and they are widely used in many applications like logic synthesis and formal verification. Berkeley Logic Interchange Format (BLIF) is the most popular format used in academic tools for describing logic-level hierarchical circuits in textual form. A circuit described in BLIF can be viewed as a directed graph of combinational logic nodes and sequential elements. Due to this, it is reasonably easy to transform a circuit described in BLIF into the corresponding BDD or a corresponding AIG. This has already been automated in many academic EDA tools (e.g. SIS [59], VIS [77], MVSIS [9] and ABC [11]). Furthermore, several (bi-directional) translators between BLIF and HDL/netlists are publicly available.

### 3.4.1 Structural Information

Figure 3.2 shows our proposed academic design flow. This design flow outlines a similar path as that of the industrial design flow. The input file is given in BLIF, PLA and BENCH file formats describing the digital circuit in textual form and the output is again the power, delay and area reports. However all the tools used in this flow are academic open-source tools used in research. Hence the name academic design flow. The flow is composed of three types of tools i) HDL translators, ii) logic synthesis and verification tools and iii) power analysis tools. The selection of tools are:

- HDL translators EDIF2BLIF [24], BLIF2VHDL [7] and several built-in translators in VIS, FBDD and ABC;
- SIS [59] - a classical interactive tool for logic synthesis and power estimation in academia;
- MVSIS [9] - a multi-level, multi-valued logic synthesis tool;
- FBDD [79] - a BDD logic synthesis system based on folded logic transformation and two-variable sharing extraction;
- BDS-PGA [76] - a practical logic synthesis system based on a BDD decomposition technique;
- ABC [11] - a logic synthesis and verification tool which performs scalable logic optimisation based on AND-Inverter Graphs (AIGs) [48];
- VIS [77] - a tool that integrates the verification, simulation and synthesis of hardware systems;

### 3.4.2 Details of the design flow

The input of the design flow can be a circuit description in BLIF, PLA, or BENCH formats. Restricted subsets of EDIF, Verilog, VHDL and Synopsys Equation Format are also accepted as input. The output of the design flow is an optimised circuit in BLIF or a restricted form of Verilog. The power analysis tools are then be used to calculate the



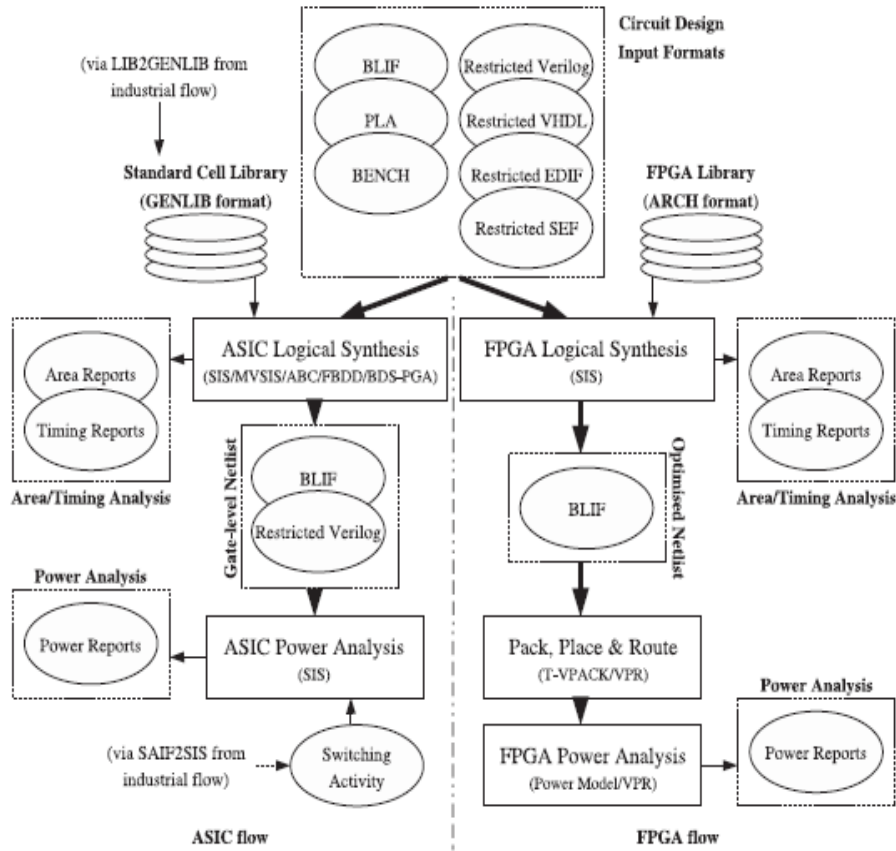


Figure 3.2: Academic Design Flow

circuit’s power dissipation. Note that this design flow also enables technology mapping which converts a circuit composed of simple gates into a circuit composed of lookup tables (LUTs) suitable for FPGA based implementation. Moreover, various transformations of circuit descriptions (e.g. Verilog to BLIF) at different stages of the flow can be done by using HDL translators available within the design flow. The correctness of such transformations is ensured by the equivalence checker in VIS and ABC.

The design flow is complete as it takes the RTL description of the circuit and synthesises it into a gate-level netlist after technology mapping. Power, delay and area estimations tools are also available within the design flow. The same flow is observed within the industrial design flow as well. However, there are several issues that can be improved within the academic tool. Firstly the technology libraries used in all logic synthesis tools available within the design flow are outdated. Due to this, the final optimised circuit cannot be interfaced with commercial EDA tools. Secondly, some

commercial EDA tool features are still missing in the proposed design flow. For example, switching activity from gate-level event-driven simulation cannot yet be annotated directly onto the circuit for power analysis. Due to these issues, an integrated design flow is essential. In this way information from the industrial domain can be used within the academic domain and vice versa.

### 3.5 Integrated Design Flow

The two design flows presented in Sections 3.3 and 3.4 (with some enhancements) are integrated, providing a solid platform for future research in low power design methodologies. This integrated design flow provides a flexible programming environment with new capabilities and improved performance. Our integrated design flow is depicted in Figure 3.3.

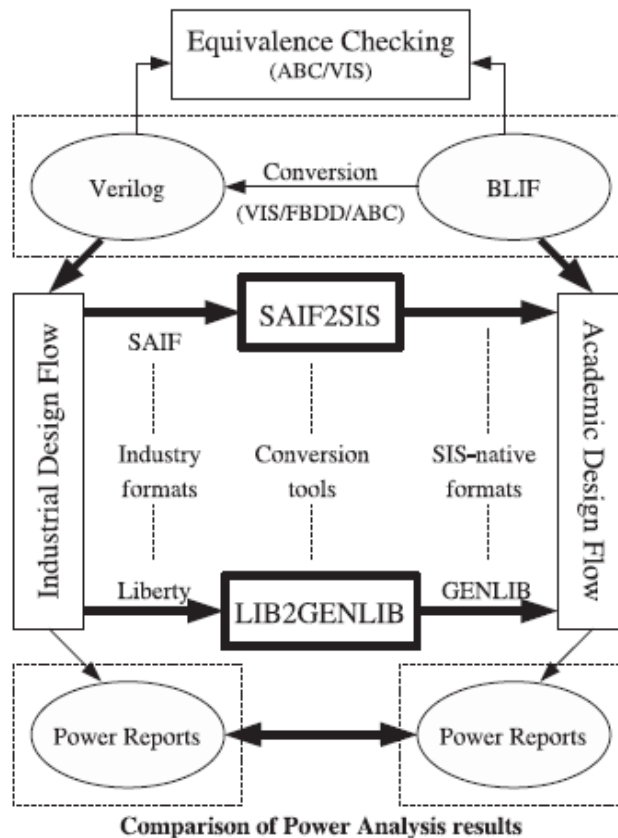


Figure 3.3: Proposed Integrated Design Flow

### 3.5.1 Tool Flow

The integrated design flow is composed of four different types of tools: i) existing academic tools, ii) commercial tools, iii) modified existing academic tools and iv) new tools. A short description of the new tools introduced within the integrated design flow is as follows:

- LIB2GENLIB is a new tool implemented in Java which converts an industrial technology library file (e.g. 65nm TSMC technology) to the corresponding genlib, the technology library file format used by SIS, MVSIS, ABC and VIS;
- SAIF2SIS is a new tool implemented in C which converts relevant switching activity information from an SAIF file (obtained from the industrial design flow) into the corresponding switching activity file used in academic tool SIS;

### 3.5.2 Key features of the integrated design flow

The integrated design flow consists of two sub-flows, the academic flow and the industrial flow (as shown in the Figure 3.3), and possesses the following features:

- Flexible programming environment - the academic flow provides an open and flexible programming environment in which new techniques for logic optimisation, power estimation and power optimisation can be developed; and possibly compared with similar techniques presented in the industrial flow.
- Up-to-date technology processes - the tool LIB2GENLIB provides up-to-date commercial technology processes that can be used for mapping in the academic flow.
- Interoperability - useful information (e.g. switching activity information from gate-level simulation) can be easily obtained from the industrial flow and plugged into the academic flow to obtain better and accurate power analysis results.
- Comparative study - the integrated design flow can serve as an optimal platform to evaluate the effectiveness and efficiency of techniques developed in the academic flow by means of comparative study amongst the academic flow and industrial flow.

### 3.6 Experimental Results

We performed experiments on several combinational MCNC benchmark circuits using an AMD 2258 MHz CPU with 256 MB RAM. Before running the experiments, the tool LIB2GENLIB translates the 65nm CMOS TSMC standard cell library (tcbn65gplustc.lib) to the corresponding genlib for mapping. In all the experiments, the updated genlib file is used for the mapping. Since we have used the updated technology library files, the results are accurate and can be used within the industrial domain. Also realistic switching activity information which is essential in power estimation is introduced in using SAIF2SIS tool. The tool SAIF2SIS converts the relevant switching activity information from SAIF files into the corresponding SIS switching activity files. (The SAIF files are obtained by running a RTL-level simulation of the equivalent benchmarks modeled in Verilog using VCS.) The selected benchmarks are run on ABC for logic synthesis and SIS is used to calculate their power dissipation with and without the *switching activity files* (SWF) provided by SAIF2SIS. The results shown in Table 3.1 are the estimated power values by SIS which uses the modified genlib library for mapping (using the tool LIB2GENLIB) and also uses the relevant switching activity information through the SWF (using the tool SAIF2SIS). The results (given in  $\mu W$ ) indicate that annotating realistic switching activity has reduced estimated dynamic power by 40% while compared to the power estimated values without switching activity information.

### 3.7 Conclusions

An integrated design flow consisting of both academic and commercial tools has been presented, in which new techniques can be developed and compared. In addition, this integrated design flow provides a solid platform for future research in low power design methodologies, with new capabilities and improved performance (see the experimental results given in Section 3.6 for details). This chapter outlines the foundation behind the academic EDA tools, and its integration with the commercial tools via the integrated design flow. This helped in the manipulation followed by simulation of any digital circuit represented as Boolean logic network (eg. AIG, NNIG and NIG) for power and delay estimation and optimisation, as discussed in the next chapters of the thesis. In the next chapter, we present the manipulation of NAND-NOR-Inverter Graphs (NNIGs)

Table 3.1: Power dissipation with and without SWF

Circuit	without SWF	with SWF	% reduction
-	( $\mu W$ )	( $\mu W$ )	-
cm138a	63.80	51.80	-18.80%
alu4	2519.30	1445.00	-42.64%
b1	58.30	31.90	-45.28%
c8	615.80	298.30	-51.55%
dalu	3880.70	1290.60	-66.74%
b9	417.60	113.30	-72.86%
count	654.50	115.40	-82.36%

which are largely being used in the academic domain for logic synthesis and technology mapping of digital circuits. Another data structure called Probabilistic Timed NAND-NOR-Inverter Graph is also introduced which has switching activity and delay information of a digital circuit introduced on each of the nodes of the network. These parameters are obtained using the integrated design flow we presented in this chapter.

## Chapter 4

# Graph manipulation for both power and delay analysis

The structural representation of any digital circuit into a Boolean network and technology mapping of the digital circuit using the Boolean network are important issues in the synthesis of digital circuits. In the literature, data structures such as ROBDDs, AIGs and NNIGs are commonly used for the structural representations of binary logic networks at the technology-independent stage. They are widely used in the EDA community and their main application domains are logic synthesis, testing and formal verification, technology mapping, power and delay optimisation, as stated in the previous chapter.

### 4.1 Introduction

Optimisation of multi-level logic networks after structural representation plays an equally important role in EDA. Various factorization methods such as Boolean and Algebraic factorisation are used in the optimisation of such Boolean networks. In [56], a decomposed algebraically factored form representation of a Boolean expression is introduced to guide the structural transformations of a Boolean network for low power. NAND-NOR-Inverter Graphs (NNIGs) are another multi-level Boolean network introduced for such structural representation. According to [3], NNIGs (functionality of the circuit represented and synthesised as NNIGs) are found to exhibit 3.97% less node count when compared to AIGs and consume 23.74% less library cells than AIGs for the representa-

tion of the same circuits. We therefore proceeded with NNIGs as our data structure for future manipulation and implementation. The tool ABC [11] has been developed for the logic synthesis and manipulation of AIGs. To automate the implementation of NNIGs, we present a new tool Data Structure Manipulation (DSM), which has been developed as a sub-package in ABC. The implementation of the tool is based on the algebraic factorisation of minimised SOP form of the combinational logic. For experimentation, power comparisons using NNIGs and AIGs of the same circuits is done using SIS [59]. A comparisons of DSM against the ABC tool is also done.

One difficulty with the data structures including NNIGs is that it is difficult to capture all information such as power, delay and area in the same data structure. Each data structure is optimised for a particular parameter optimisation (power or delay or area). Hence even after the implementation of the tool for NNIG manipulation, NNIGs are still not ideal for both power and delay analysis as they do not incorporate power and delay as parameters. Hence we propose a new data structure called Probabilistic Timed NAND-NOR-Inverter Graph (PTNNIG), which can be used for more accurate power estimation as well as joint delay and power analysis. Switching activity and delay of a circuit from the RTL simulation are represented in PTNNIGs as probability and delay parameters respectively. To the best of our understanding, PTNNIGs are the first data structures incorporating delay and switching activity for the graph representations of digital circuits. A one-to-one mapping is presented, between a Verilog netlist and NNIG structure to map switching activity and delay values using an integrated design flow. Some experimental results are performed for the implementation of PTNNIGs.

The remainder of the chapter is organised as follows. Section 4.2 gives an overview about different graph structures and their manipulation. Section 4.3 presents the tool DSM. Section 4.4 shows some experimental results which include comparisons made on NNIGs against AIGs on the basis of power estimation. It also includes comparisons of DSM against the ABC tool. Section 4.5 introduces the new data structure PTNNIG along with an example, followed by an explanation of how switching activity and delay of a circuit from RTL simulation can be obtained and mapped on NNIGs for joint power and delay analysis. Section 4.6 presents some experimental results using BLIF and optimised BLIF using realistic switching activity information. Finally, concluding remarks are made in Section 4.7.

## 4.2 Background of data structures

Compact multi-level binary networks can be efficiently and effectively represented using AIGs and NNIGs. AIGs consist of only two-input AND gates and inverters. An example of an AIG is also given in Chapter 2. For more details refer to [48].

Similar to AIG, a logic circuit can be conveniently represented using NNIG. A NNIG also corresponds to a DAG representation for a logic comprising of two-input NAND gates, two-input NOR gates and inverters. For details refer to Chapter 2.

During multilevel network synthesis, each node of a Boolean network can be represented by minimal ON- and/or OFF- covers and in some cases together with factored expressions derived from the minimal sum-of-products (SOPs). ESPRESSO [40] is an academic standard two-level logic minimiser which is being widely and most commonly used to obtain minimised SOPs. Factoring Boolean functions is also a basic operation in algorithmic logic synthesis. Factoring is the translation of a function in the SOP form (also called disjunctive form) having minimum number of literals. For instance  $a$ ,  $abc$  and  $a(b + c + d) + e$  are all factored forms. A factored form can be represented as a tree structure, where each internal node is an AND or OR operator and each leaf is a literal. There are mainly two methods to obtain the factored form of a two-level representation of the function. One is Algebraic division [10] and [8], also known as weak division which is quite fast. The other method is Boolean division [65], also known as strong division which is slower but capable of giving better results (shorter form) in many cases. In general, the algebraic methods are fast because the logic function is treated as a polynomial, and hence fast methods of manipulation are available. All the digital circuits structurally represented as multi-level logic networks (i.e. NNIGs, AIGs etc) are synthesised using these factorial methods.

Both AIG and NNIG can be built recursively using De Morgan's laws. NNIGs and AIGs are non-canonical structures. A Boolean function can have many functionally equivalent NNIG representations corresponding to different expressions at the two-level logic, typically two structures will be compact representations; one obtained from a factored minimum sum of products (MSOP) of the function and other based on factored minimum products of sum (MPOS) of the function.



### 4.3 DSM tool

This section presents the tool DSM implemented as a sub-package in ABC. The theoretical foundation behind the tool is based on [3] and [5]. The DSM tool allows us to build a NNIG for a given circuit. The input of DSM is given in Berkeley Logic Interchange Format (BLIF) which is the most popular format used in academic tools for describing logic-level hierarchical circuits in a textual form. By using some optimisation steps, the output as the NNIG of the digital circuit (given as input in BLIF) is obtained. The NNIG network is again translated to a BLIF file format. The flow for generating NNIGs using DSM is shown in Figure 4.1.

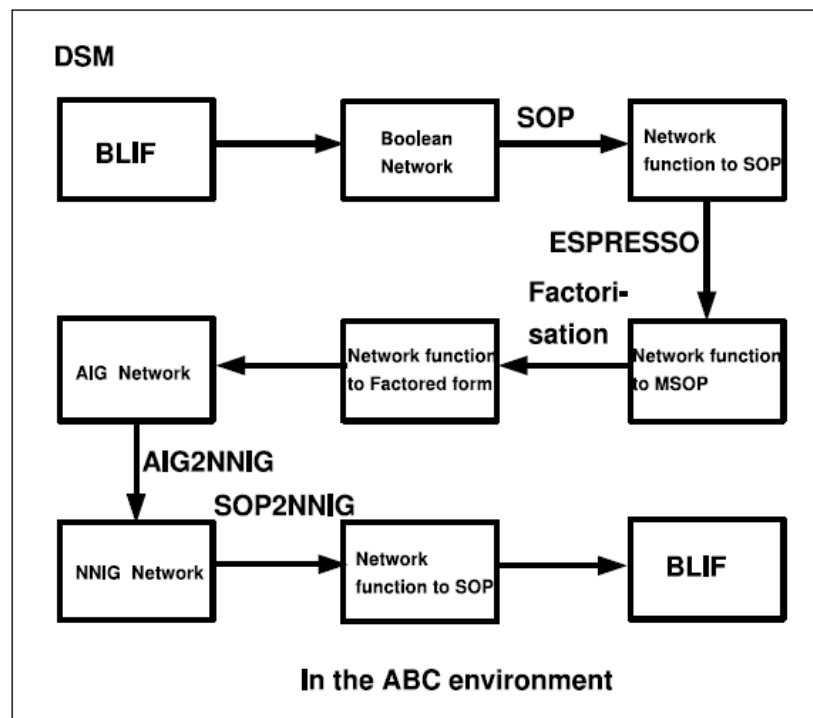


Figure 4.1: DSM tool flow

First, the BLIF file is read as a network in ABC whose functionality is converted to SOP. Using ESPRESSO, the SOP undergoes a two-level logic minimisation. If there is a sharing of variables in the reduced SOP expression of the function, the obtained minimised SOP is then factorised using the algebraic factorisation technique. The factorised-minimised SOP is converted to AIG in the environment of ABC and then further transformed to a NNIG network using a sub-command AIG2NNIG. The function-

ality of the NNIG network is converted back to SOP using a sub-command SOP2NNIG in the DSM tool and then to BLIF file format using the translator available in ABC. This tool is implemented in C as a sub-package in ABC. It can handle both sequential and combinational digital circuits as inputs.

#### 4.4 Experimental Results for AIG vs. NNIG

The BLIF files corresponding to AIG and NNIG are compared with respect to power dissipation in this experiment. The BLIF file corresponding to NNIG is obtained as shown in Section 4.3. The BLIF file corresponding to AIG is obtained following the same DSM tool flow (shown in Figure 4.1) after ESPRESSO optimisation [40]. The corresponding AIG structure is derived from the minimum SOP form of the function in the environment of ABC and then written to BLIF file format using ABC.

The BLIF files corresponding to AIG and NNIG networks are given as inputs in SIS (in the environment of ABC) for power estimation. Table 4.1 presents a list of randomly taken 10 combinational and 5 sequential circuits and their power estimation in  $\mu W$ . Some BLIF examples used in these results are from the MCNC benchmark circuits and some of the BLIF files used are those mentioned in [3]. We have considered the clock frequency and  $V_{dd}$  to be 20 MHz and 5 Volts respectively. The table clearly indicates that many combinational and sequential digital circuits synthesised using NNIGs (NNIG blif file) consume less power than AIG based synthesis. On average, the difference in power consumption is 22.06%.

Another set of experiments is performed on the BLIF files obtained from the DSM tool against the BLIF files obtained from ABC. The BLIF files from ABC considered in this experiment are those obtained from AIG network creation after using one of the standard scripts of ABC namely *resyn*. In this script, the series of synthesis commands are *balancing*, *rewriting*, *rewriting with zero cost replacement*, *balancing*, *rewriting with zero cost replacement* and *balancing*. These synthesis commands are used to reduce the number of nodes, number of logic levels and delay. Power estimation on the two BLIF files is done using SIS. Table 4.2 presents a list of combinational circuits and their power estimation in  $\mu W$ . The BLIF files used in these experimental results are those mentioned in [3]. We have considered the clock frequency and  $V_{dd}$  to be 20 MHz and 5 Volts respectively.

The table indicates that for some of the combinational circuits, the tool technique used in DSM can prove better than some of the synthesis techniques used in ABC, resulting in lower power dissipation. On average, the power reduction is 17.69%.

Table 4.1: Power estimation on AIG and NNIG

Circuit	AIG network	NNIG network	% power difference
-	( $\mu W$ )	( $\mu W$ )	-
lf3	29.50	22.80	-22.71%
lf6	36.30	26.70	-26.44%
lf11	41.00	37.80	-7.80%
lf8	56.00	32.50	-41.96%
lf5	60.10	44.10	-26.62%
cm85a	205.70	179.60	-12.68%
cm162a	249.80	162.50	-34.94%
cc	367.20	289.20	-21.24%
b9	639.90	540.10	-15.59%
adder	1242.50	998.80	-19.54%
bbara	162.00	94.60	-42.07%
tav	245.50	241.70	-2.08%
ex7	378.90	325.90	-14.09%
ex3	418.70	267.90	-35.04%
ex1	878.10	805.90	-8.25%

Table 4.2: Power comparisons against ABC and DSM

Circuit	ABC	DSM	% power difference
-	( $\mu W$ )	( $\mu W$ )	-
lf2	24.20	18.10	-25.65%
lf3	24.00	22.80	-5.08%
lf5	56.60	44.10	-22.45%
lf6	28.30	26.70	-6.28%
lf7	71.40	61.00	-14.62%
lf10	49.40	37.70	-23.68%
lf12	64.20	51.20	-20.07%
lf13	24.20	18.10	-25.04%

## 4.5 Probabilistic Timed NNIGs (PTNNIGs)

### 4.5.1 Background

In recent years, power consumption has become one of the biggest constraints in digital circuits. Power can be broken down into static and dynamic components. The dynamic power also known as switching power is caused by the charging and discharging of the capacitances in a CMOS circuit. Dynamic power optimisation is targeted in our work. The dynamic power is also described in Equation 3.1. In our work, *alpha* also known as switching activity factor is considered as the parameter for dynamic power optimisation. The critical path length or the longest path delay in the circuit network is considered as the parameter for delay optimisation.

Various data structures have been used for the manipulation of switching activity and delay such as those referred to in [38]. However they were not ideal for a combined power estimation and delay analysis. This section hence proposes the new data structure PTNNIG. Switching activity and delay both represented as probability and delay parameters respectively are specifically defined in the syntax of PTNNIGs for power and delay analysis. Optimising such graphs will imply power and delay minimisation.

Our ultimate goal is to provide a data structure which enables efficient power-driven delay optimisation and delay-driven power optimisation.

### 4.5.2 NNIG and PTNNIG

A NNIG network represents the functionality of the circuit using 2-input NANDs, 2-input NORs and Inverters.

A PTNNIG is defined as a *quadruple*( $V, E, P, T$ ), where  $V$  and  $E$  represent the same set of attributes as defined in the NNIG network. Refer to Chapter 2 Section 2.3. Two new elements are defined in this data structure namely  $P$  and  $T$ .  $P$  is a set of probabilities and  $T$  is a set of delay values. Switching probability at each logic node in the NNIG network ( probability of switching from logic state ‘1’ to ‘0’ and ‘0’ to ‘1’) is an important factor in dynamic power dissipation as described in Equation 3.1. Probability of a logic node to be at state ‘1’ represented as  $p_1$  also gives an estimate of switching probability. These  $p_1$  values associated at each logic node of the NNIG network are represented in the set  $P$ . Similarly the delay value from the input to the output of each logic node of the NNIG network is important to determine the longest path delay of the network. Let  $t_f$  represent the the delay at a logic node at the falling edge. These  $t_f$  values are represented in the set  $T$ . Hence the probability and the delay are two parameters added to existing NNIG data structures. The new parameters namely  $P$  and  $T$  are associated to each logic node of the NNIG network i.e. set  $V$ .

### 4.5.3 Mapping of Probability and Delay on PTNNIGs

For illustration purpose, this section presents a way to map probability and delay attributes  $P$  and  $T$  on NNIG to build PTNNIG. We introduce a tool written in C namely NNIG2Net. This tool converts the NNIG network (obtained from the DSM tool) to a NNIG netlist in Verilog. In this translation, each NAND node of the NNIG network is converted to 2-input NAND cell, each NOR node of the NNIG network is converted to 2-input NOR cell from a specific library. The complemented (inverted) edges are replaced with inverter cells and non-complemented edges are replaced with buffer cells. Even the name mapping of NNIG nodes and edges to the netlist cells and wires is kept in mind. In this way exact technology mapping of the NNIG network to a NNIG netlist (in Verilog) is possible. The two comparative structures are a NNIG network obtained

from the tool DSM and a NNIG netlist obtained from NNIG2Net. A one-to-one mapping is possible between the information associating the Verilog netlist to the nodes of the NNIG network. A NNIG network representing the functionality  $ab(c + d)$  i.e.  $((c + d)Xa)Xb$  is shown in Figure 4.2. A one to one mapping between this NNIG network and NNIG Verilog netlist is explained in Figure 4.7.

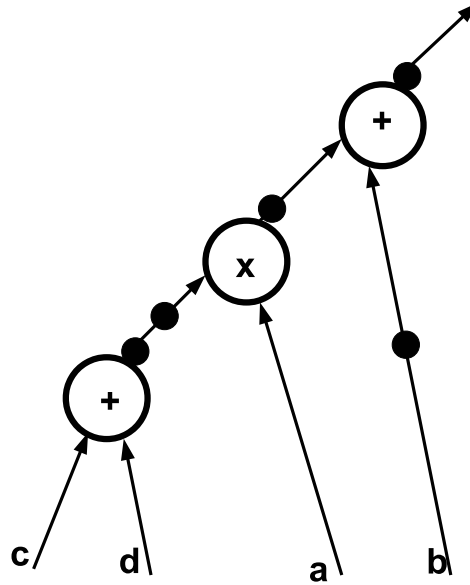


Figure 4.2: A NNIG example for a circuit with functionality  $((c + d)Xa)Xb$

The description of such a circuit in BLIF is shown in Figure 4.3. This NNIG network is converted to a NNIG netlist in Verilog using NNIG2Net as shown in top right side of the figure. The Verilog netlist is shown in Figure 4.4.

On performing RTL simulation on this Verilog design using VCS, a commercial tool from Synopsys, SAIF files are obtained. The probabilities in the set  $P$  of PTNNIG are derived from these SAIF files [57]. SAIF files contain information such as the switching activities for all the nodes of the digital circuit. Switching activity of a node in a digital circuit is highly dependent on the circuit inputs of the current clock cycle and the previous clock cycle. It indicates the number of times a particular circuit node switches from 0 – 1 and from 1 – 0 in a given clock cycle. SAIF files begin with a header and contain a set of activity metrics such as  $T0$ ,  $T1$ ,  $TX$ ,  $TC$ ,  $TS$  and  $DUR$  for each circuit

```

.model ptmnig
.inputs a b c d
.outputs e
.names a b c d e
1101 1
1110 1
1111 1
.end

```

Figure 4.3: BLIF file of the circuit representing the logic function  $ab(c + d)$

```

module ptmnig ( a, b, c, d, e );
input a, b, c, d;
output e;
wire n5, n6, n7, n8;
NR2D0HVT U6 ( .A1(c), .A2(d), .ZN(n8) );
NR2D0HVT U7 ( .A1(n5), .A2(n6), .ZN(e) );
INVD0HVT U8 ( .I(b), .ZN(n5) );
ND2D0HVT U9 ( .A1(a), .A2(n7), .ZN(n6) );
INVD0HVT U10 ( .I(n8), .ZN(n7) );
endmodules

```

Figure 4.4: Verilog file of the circuit representing the logic function  $ab(c + d)$

node.

- T0, T1 and TX represent the total time spent at logic 0, 1 and X respectively.
- TC represents the total number of logic transitions (0 to 1 and 1 to 0).
- TS represents the SAIF header ‘TIMESCALE’ field.
- DUR represents the SAIF header ‘DURATION’ field.

Hence the  $p_1$  values for the probability set  $P$  are defined as:

$$p_1 = \frac{T1}{T} \quad (4.1)$$

where the simulation duration  $T$  is defined as

$$T = TS * DUR \quad (4.2)$$

Using the one-to-one mapping, switching probabilities  $p_1$  for  $P$  for each circuit node is obtained from the SAIF file and mapped to each node of the NNIG network obtained from DSM.

Similarly, according to Figure 4.7, the delay values in the set  $T$  are obtained from the *Standard Delay Format* (SDF) files [58]. The SDF files are generated using PrimeTime from Synopsys.

The SDF files are used for the representation and interpretation of delay data used at any stage of the logic synthesis. It includes path delays, delay constraint values and interconnect delays for each of the cells. The files contain the cell names used in the mapping along with the individual delays from each input-output path of the corresponding cells. The term IOPATH represents the path referenced to, followed by the worst case and the best case values at rising edge, which is then followed by the worst case and the best case values at falling edge. The falling edge delay for each cell of the netlist defined as  $t_f$  is mapped on each node of the NNIG network and hence the delay set  $T$  is defined.

The corresponding SAIF and SDF files of the Verilog design are generated and shown in Figure 4.5 and Figure 4.6 respectively. Only parts of the SAIF and SDF files are shown.

$P$  and  $T$  are sets containing  $p_1$  and  $t_f$  information of each circuit node respectively. The  $p_1$  and  $t_f$  for the set  $P$  and  $T$  respectively are hence obtained from the SAIF and SDF files by one-to-one mapping. The conversion from NNIG to PTNNIG is described in the Figure 4.7. The final PTNNIG is shown in the bottom left side of the Figure 4.7. On each node of the PTNNIG network, the probability and delay parameters  $P$  and  $T$  are shown.  $p_1$  is a probability value hence less than 1 and  $t_f$  is measured in nanoseconds.

## 4.6 Experimental Results for PTNNIG

Power estimation using the academic EDA tool SIS allows the user to input a file with specifications of input probabilities, node capacitances and node delays of the



```

.....
(e
(T0 18081) (T1 1919) (TX 0)
(TC 127) (IG 0)
)
(n5
(T0 16820) (T1 3180) (TX 0)
(TC 126) (IG 0)
)
(n6
(T0 1920) (T1 18080) (TX 0)
(TC 191) (IG 0)
)
.....

```

Figure 4.5: SAIF

circuit. We have named this file as **SA** file. In our experiments for power estimation, the probability values of primary inputs and primary outputs are annotated in the **SA** file. The **C** tool, SAIF2SIS [41] reads the relevant switching activity information from a SAIF file, calculates  $p_1 \in P$  using Equation 4.1 and allows to specify these probabilities of the primary inputs and primary output in the **SA** file. For inputs, BLIF files of 10 combinational MCNC benchmark circuits are taken. Power estimation results in  $\mu W$  against a BLIF file and an optimised BLIF file obtained from the tool DSM are shown in Table 4.3. Same **SA** file is used with both the BLIF files. The clock frequency is 20 MHz and Vdd is 5 Volts for power estimation. The NNIG network corresponding to the BLIF file obtained from the DSM tool used in these results can be referred as the PTNNIG, with probability parameter annotated through the **SA** file and delay parameter assumed zero assuming the zero-delay model. From the results, it is concluded that the power dissipation is less in the case of BLIF files obtained from the tool DSM. The average reduction in the estimated power is 16.7%.

```

.....
(CELL
(CELLTYPE“ND2D0HVT”)
(INSTANCE U9)
(DELAY
(ABSOLUTE
(IOPATH A1 ZN (0.010::0.023)
(0.011::0.031))
(IOPATH A2 ZN (0.013::0.032)
(0.014::0.041))
)
)
)
(CELL
(CELLTYPE “INVD0HVT”)
(INSTANCE U8)
(DELAY
(ABSOLUTE
(IOPATH I ZN (0.009::0.022)
(0.006::0.015))
)
)
)
.....

```

Figure 4.6: SDF

## 4.7 Conclusions

This chapter presented a tool to build optimised NNIG networks of digital circuits within ABC. From the results, we obtained an average power reduction of 22.06% with NNIG networks while compared to AIG networks. An average power reduction of 17.69% on BLIF files from DSM compared to BLIF files from ABC tool, was also obtained. The

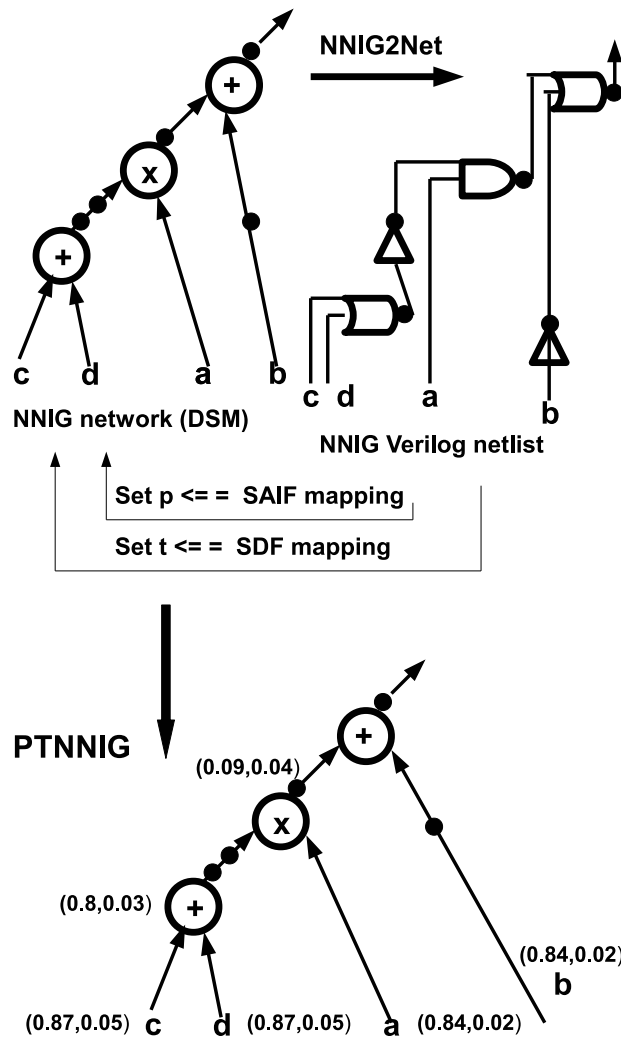


Figure 4.7: Mapping of Probability and Delay issues on PTNNIGs

node probabilities for power estimation are introduced in SIS using an integrated design flow as used in Chapter 3, Section 3.5. This implies that NNIG logic data structures can be used for enabling low power logic designs.

This chapter also introduces briefly the concept of concurrent power and delay optimisation. In the context of a unified framework for power and delay analysis and optimisation, the new data structure PTNNIG is presented which represents the functionality of any digital design along with switching activity and delay as parameters on each node of PTNNIG. Such data structures will also help us to apply a statistical

Table 4.3: Power estimation against BLIF and optimised BLIF

Circuit	BLIF	optimised BLIF	% power difference
-	( $\mu W$ )	( $\mu W$ )	-
b1	29.30	23.40	-20.05%
cm85a	126.40	119.30	-5.61%
cm162a	125.20	116.60	-6.90%
cmb	126.70	117.50	-7.38%
pm1	143.70	108.10	-24.79%
x2	143.40	121.70	-15.12%
mux	195.30	115.70	-40.72%
cc	217.00	206.70	-4.74%
b9	212.50	148.40	-30.21%
c8	497.40	414.40	-16.77%

approach to the power analysis with constraints on delay parameter, similarly a statistical approach to delay analysis with constraints on switching activity parameter. Such constraints will allow us to obtain delay-aware power optimisation and power-aware delay optimisation. Although our major work of synthesis and optimisation of digital circuits (in terms of dynamic power and delay) is based on AIGs implemented in ABC, as it will be discussed in the next chapters, the introduced PTNNIG structure gave the motivation to achieve our main objective of the thesis i.e. power-driven delay optimisation and delay-driven power optimisation. In the next chapter, we work on AIGs for the estimation and optimisation of dynamic power and delay of a digital logic circuit. Further, various reordering rules for delay-driven power optimisation and power-driven delay optimisation are introduced on AIGs.

## Chapter 5

# Power and delay estimation and optimisation on AIGs

With the advent of wireless communication and portable devices, energy efficiency has become a major constraint in digital circuit design. Tight energy constraints are common place in many modern VLSI applications. The energy dissipated, often referred as the power-delay product, is the product of power dissipation and the input-output delay. In this chapter, dynamic power and longest path delay of a digital logic combinational circuit are estimated and optimised. The logic synthesis and optimisation is based on AND-Inverter Graphs (AIGs) which capture the functionality of the circuit. The switching power and arrival times of circuit nodes are annotated onto a AND-Inverter Graph under the zero and a non-zero-delay model. We introduce several reordering and restructuring rules which are applied on the AIG nodes to minimise switching power or longest path delay of the circuit at the pre-technology mapping level. The academic Electronic Design Automation (EDA) tool ABC is used for the manipulation of AND-Inverter Graphs.

### 5.1 Introduction

In recent years, power dissipation and delay are being given increased importance in digital design. Average power dissipation in digital CMOS circuits can be expressed as the sum of three main components  $P_{short-circuit}$ ,  $P_{leakage}$  and  $P_{switching}$ , as stated in Chapter 3.  $P_{switching}$  is the switching power dissipation, also called the dynamic

power, and is given by Equation 3.1 of Chapter 3. The switching power is proportional to  $\alpha$  the switching activity factor (also called transition probability),  $C_l$  the overall capacitance to be charged and discharged in a reference clock cycle,  $V_{dd}$  the supply voltage and  $f_{clk}$  the clock frequency. Given the supply voltage and clock frequency, the product of capacitive load and switching activity factor can give an estimate of dynamic power dissipation (as illustrated in Equation 3.1). In this chapter we have estimated this product using AIGs. We have also estimated longest path delay of the network for delay optimisation by assuming a non-zero-delay model.

Our power optimisation method is based on AIGs. AIGs offer a much compact representation of the same circuit than BDDs. Further, it is difficult to incorporate delay information in BDDs which restricts them to power optimisation only. However AIGs allow to map the delay information along with the switching probability. In addition to that, formal verification and technology mapping is easier in AIGs owing to its simple and flexible structure. Hence power and delay analysis and optimisation, both are possible using such data structures.

The remainder of the chapter is organised as follows. Section 5.2 presents switching power estimation on AIG nodes under the zero-delay model. Section 5.3 presents longest path delay (critical path length of the network) and switching power estimation under a non-zero-delay model. Section 5.4 presents four reordering rules which are used to reduce the switching power of the AIG network. Similarly Section 5.5 presents three reordering rules used in the reduction of critical path length of the network. Finally concluding remarks are given in Section 5.6.

## 5.2 Switching probability estimation under the zero-delay model

The functionality of the digital logic circuit is structurally represented as an AIG network and switching probability is estimated on each of the AIG nodes. The switching probability estimation method on each of the AIG nodes used in our work is based on the technique used in [38]. For this estimation, circuit input signals are assumed to be statistically uncorrelated and completely independent. The zero-delay model is assumed, that is all gate output evaluations occur instantaneously. Consider a sub-graph

of an AIG shown in Figure 5.1, with AND nodes  $a$ ,  $b$  and  $c$ . The node  $c$  has 2 inputs (fanins) namely nodes  $a$  and  $b$ . The functionality at node  $c$  is denoted as  $c = ab$ .

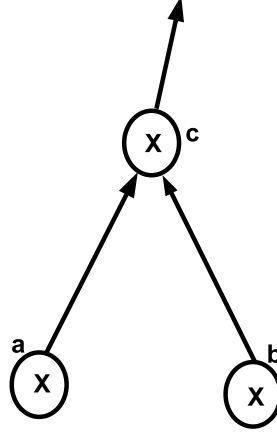


Figure 5.1: AIG of logic function  $ab$

The probability of a node  $c$  denoted as  $P(c)$  is the probability that  $c$  has a value of ‘1’ at some arbitrary time of observation. Similarly  $P(c')$  denotes the probability that  $c$  has a value of ‘0’ at some point of time. Consider the node  $c$  having the probability  $P(c)$  and probabilities  $P(a)$  and  $P(b)$  for the corresponding inputs  $a$  and  $b$  and the functionality given by  $c = ab$ . The switching probability  $P_{sw}(c)$  of  $c$  is to be determined. Switching occurs if and only if the value of  $c$  changes from ‘0’ to ‘1’ or ‘1’ to ‘0’, at two different time of observation. The output  $c$  is ‘1’ when both inputs are ‘1’.

$$P(c) = P(a)P(b) \quad (5.1)$$

The output  $c$  is ‘0’ when either one or both inputs are ‘0’.

$$P(c') = (1 - P(a))P(b) + P(a)(1 - P(b)) + (1 - P(a))(1 - P(b)) \quad (5.2)$$

Consider the value of  $c$  at two different observation times  $c_{t1}$  and  $c_{t2}$ . If the value of  $c$  at those instants is not the same,  $c$  is considered to switch. For more details refer [38]. Hence the switching probability  $P_{sw}(c)$  of  $c$  is:

$$P_{sw}(c) = P((c_{t1} = 0) \cap (c_{t2} = 1)) + P((c_{t1} = 1) \cap (c_{t2} = 0)) \quad (5.3)$$

$$P_{sw}(c) = 2P(a)P(b)(1 - P(a)P(b)) \quad (5.4)$$

Using the equation above, the switching probability on each of the AIG nodes can be calculated if the probabilities of the primary inputs of the AIG are known. As stated in Equation 3.1 of Chapter 3, given  $V_{dd}$  and  $f_{clk}$  the dynamic power is the product of switching activity factor and capacitive load. In our case, switching power at each node is the product of switching probability calculated at each node times the capacitive load at each node. The fanout at a given nodes is assumed to be the measure of capacitive load at that given node. Consider an AIG network of functionality  $f$  with  $n$  AND nodes namely  $N_1, N_2, \dots, N_n$  and fanout  $F_1, F_2, \dots, F_n$  at each of the nodes respectively. Let  $P_{sw}(N_1)$  denotes the switching probability at node  $N_1$ ,  $P_{sw}(N_2)$  denotes the switching probability at node  $N_2$  etc. The total switching power of this AIG network denoted as  $SP(f)$  is given by:

$$SP(f) = P_{sw}(N_1).F_1 + P_{sw}(N_2).F_2 + \dots P_{sw}(N_n).F_n \quad (5.5)$$

The switching power  $SP(f)$  is chosen as the parameter for dynamic power optimisation.

## 5.3 Longest path delay and switching power estimation under a non-zero-delay model

### 5.3.1 Longest path delay estimation

The circuit is structurally represented as an AIG graph and the longest path delay from one of the inputs to one of the outputs of the network is calculated under a given delay model. A non-zero-delay model (corresponding to the AIGs) is assumed which takes into account the delay due to AND gates, inverters and the fanout. Consider the AIG network shown in Figure 5.1 with three AND nodes namely  $a, b$  and  $c$ . The node  $c$  has two fanins namely  $a$  and  $b$  representing the functionality  $c = ab$ . The arrival time at the output of the AND node  $c$  with inputs  $a$  and  $b$  can be calculated as follows. Let  $t_{arr,a}$  represent the arrival time at node  $a$  and  $t_{arr,b}$  represent the arrival time at node  $b$ .  $IND$  is the parameter accounting for the inverter delay. If there is no inversion edge corresponding to that input, then  $IND$  is zero.  $t_{d,c}$  denotes the node delay at node  $c$ . Finally  $FAND$  is the parameter to measure the fanout delay and  $Fc$  represents the fanouts at node  $c$ . Let  $AT_0$  define the arrival time from fanin  $a$  to  $c$  and let  $AT_1$  define



the arrival time from fanin  $b$  to  $c$ , then:

$$AT_0 = t_{arr,a} + IND \quad (5.6)$$

$$AT_1 = t_{arr,b} + IND \quad (5.7)$$

$$t_{arr,c} = \max(AT_0, AT_1) + t_{d,c} + FAND.Fc \quad (5.8)$$

In the example of Figure 5.1,  $IND$  is zero with respect to both the fanins and fanouts of  $c$ .  $Fc$  is equal to one.

Once the delay model is fixed, the arrival time at each of the AIG nodes is calculated, assuming arrival time of primary inputs to be zero. The maximum arrival time (longest path delay) of the AIG network with functionality  $f$  (represented as  $MAT(f)$ ) is used for delay optimisation.

### 5.3.2 Dynamic power estimation under given delay model

The difference in arrival times of signals at a gate input (the difference in the arrival times of the fanins at the AND node), leads to spurious or unwanted transitions, also called glitches. These spurious transitions play a major role in dynamic power dissipation. Based on the fanins' arrival times and the delay of the node, time instants at which a possible signal transition can occur have been calculated and associated with each node of the graph. Let  $t_{sw,a}$  denote a switching instant at node  $a$ .

The time parameter hence plays a critical role in the power dissipation estimation. Hence, the exact description of a logic circuit shall include not only its logic behavior, but also the time dependency among the logic signals. Furthermore, since the glitch generation is strongly dependent on time, a modified Boolean function, which describes the logic and delay behavior of each signal, is needed. This is called the *Real Delay Boolean Function* (RDBF). For more details refer to [72] and [64]. The RDBF of each signal is built using AIG. Consider the AIG of the logic function  $f = abc$  as shown in Figure 5.2. The arrival time of the primary inputs namely  $a$ ,  $b$  and  $c$  is assumed to be zero.

The logic behavior of the node  $f$  is described in time domain by the following RDBF:

$$f = a(t - 5d)b(t - 5d)c(t - 3d) \quad (5.9)$$

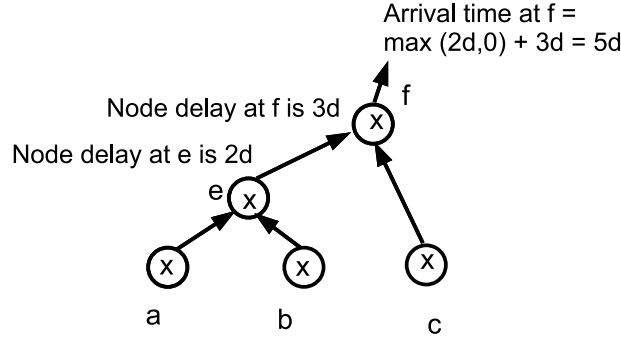


Figure 5.2: AIG of logic function  $abc$

This equation implies that the function  $f$  is dependent on three input signals namely  $a$ ,  $b$  and  $c$ . Node  $a$  is at  $5d$  delay from  $f$ , hence the term  $a(t - 5d)$  is used in the equation. Similarly, node  $b$  is also at  $5d$  delay from  $f$ , hence the term  $b(t - 5d)$  is used in the equation. Node  $c$  is at  $3d$  delay from  $f$ , hence the term  $c(t - 3d)$  is used in the equation. The signal  $f$  will switch at two time instants then i.e.  $t_{sw,f} = 3d$  and  $t_{sw,f} = 5d$ . These two time instants are actually the switching instants. The set of switching instants of node  $f$  is denoted as  $T_{sw,f}$ . In an AIG network these switching instants can be easily calculated by traversing the graph from lower levels to higher levels of the AIG network. Associating the arrival times and node delays corresponding to each node of the network, the switching instants can be calculated. Hence this method aims at a series of switching activity calculations at certain switching time points. Let  $P_{sw,t_{sw,x}}(x)$  denote the switching probability at node  $x$  at the switching instant  $t_{sw,x}$ . The sum of switching probabilities on all the switching instants of a given node is the total switching probability at that node. Let  $N1$  denotes one of the nodes of the AIG network and  $N1_{sp}$  denotes the total switching probability at node  $N1$ , given by the following equation.

$$N1_{sp} = \sum_{t_{sw}, N1 \in T_{sw}, N1} P_{sw, t_{sw}, N1}(N1) \quad (5.10)$$

For the switching probability calculation, consider the node  $c$  with fanins  $a$  and  $b$ , as in Figure 5.1. Let  $t1$  be one of the switching instants of node  $c$  in  $T_{sw,c}$  and  $dt$  be a small interval of time. Consider the value of  $c$  at two different observation times  $c_{t1}$  and  $c_{t1+dt}$ . If the value of  $c$  at those instants is not same,  $c$  is considered to switch. Hence the switching probability  $P_{sw,t1}(c)$  can be calculated as follows. Let the node delay at  $c$  be  $t_{d,c}$ .  $P_t(a)$  is the probability of  $a$  being '1' at time  $t$ .

$$P_{sw,t1}(c) = P(c_{t1} = 0).P(c_{t1+dt} = 1) + P(c_{t1} = 1).P(c_{t1+dt} = 0) \quad (5.11)$$

$$P(c_{t1} = 0) = 1 - P_{t1-t_{d,c}}(a)P_{t1-t_{d,c}}(b) \quad (5.12)$$

$$P(c_{t1+dt} = 1) = P_{t1+dt-t_{d,c}}(a)P_{t1+dt-t_{d,c}}(b) \quad (5.13)$$

$$P(c_{t1} = 1) = P_{t1-t_{d,c}}(a)P_{t1-t_{d,c}}(b) \quad (5.14)$$

$$P(c_{t1+dt} = 0) = 1 - P_{t1+dt-t_{d,c}}(a)P_{t1+dt-t_{d,c}}(b) \quad (5.15)$$

Design approaches, which take into account signal probability, show good results with respect to power optimisation. Many methods have been presented which primarily focus on the probabilistic behavior of the processed data, like bus encoding techniques [1]. This technique exploits redundancy in the data transmitted on the bus. According to the equations above, a probability profile or a waveform is needed corresponding to each primary input, with respect to each bit of an n-bit bus used at every interval of time. The Real Delay Boolean Function of each node is written in term of primary inputs whose probability profiles are known (at each unit interval of time) and hence switching probability is predicted.

Once the switching probability is estimated on each of the nodes, the switching power at each node (which is the product of the switching probability and capacitive load at that node) is estimated. Consider an AIG network of functionality  $f$  with  $n$  number of AND nodes namely  $N_1, N_2, \dots, N_n$ . Let  $N_{1(sp)}$  denotes the total switching probability at node  $N_1$ , similarly for  $N_2, \dots, N_n$ .  $F_1$  represents the number of fanouts at node  $N_1$  etc. The total switching power of the circuit which is the sum of switching power at all the nodes of the AIG network is given by:

$$SP(f) = N_{1(sp)} \cdot F_1 + N_{2(sp)} \cdot F_2 + \dots \dots \dots N_{n(sp)} \cdot F_n \quad (5.16)$$

The process of switching power calculation under a non-zero-delay model is slightly complex than the method under the zero-delay model. However the results are more accurate in this case due to consideration of unwanted and spurious transitions.

## 5.4 Reordering rules for power optimisation

So far we considered power estimation using AIGs. We consider the same graph structure for power optimisation as well. Work in [12] bases its synthesis and optimisation method on AIGs. A set of local two-level rewriting rules on the AIG network for node reduction is introduced. Similarly, work in [17] is based on two-level logic minimization technique for low power-driven synthesis based on local transformations. Motivated by these works, we introduced a set of low power-driven reordering rules to minimise the switching power at the AIG nodes. Four reordering and rewriting rules namely  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$  are introduced which help us reduce the overall switching power of the digital circuit structurally represented as an AIG network. These rules based on swapping and interchanging of variables change the structuring of the AIG network while maintaining the functionality using common rules of Boolean Algebra.

### 5.4.1 Rule $Rp_1$

Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which link the AND Nodes. Consider  $V_1$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Consider another node  $V_2$  in  $V$  such that  $V_1$  is a fanin of  $V_2$ . Let  $f((V_1, V_2))$  represent the complement attribute from edge  $V_1$  to  $V_2$  where  $(V_1, V_2)$  is in  $E$ . If there is no inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is zero. And if there is an inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is one. Rule  $Rp_1$  at node  $V_1$  is valid according to the following conditions:

- $F(V_1) = 1$
- $f((V_1, V_2)) = 0$

If the above conditions are valid, then the fanin of node  $V_2$  other than  $V_1$  can be swapped with the fanins of  $V_1$ . This swapping does not affect the functionality of the

AIG network but may reduce the total switching power of the network.

The rule is based on the associative rule i.e.  $c \times (a \times b) = (c \times a) \times b$  which is explained in Figure 5.3, showing a sub-graph of an AIG with AND nodes namely  $a, b, c, d$  and  $e$ . Rule  $Rp_1$  is applied on node  $d$  as the above conditions are met i.e.  $F(d) = 1$  and  $f((d, e)) = 0$ . Hence the fanin of node  $e$  i.e. node  $c$  gets swapped with the fanins of node  $d$  i.e.  $b$ . The functionality of node  $e$  does not change and since node  $d$  does not have any other fanouts, the swapping does not affect the overall AIG network.

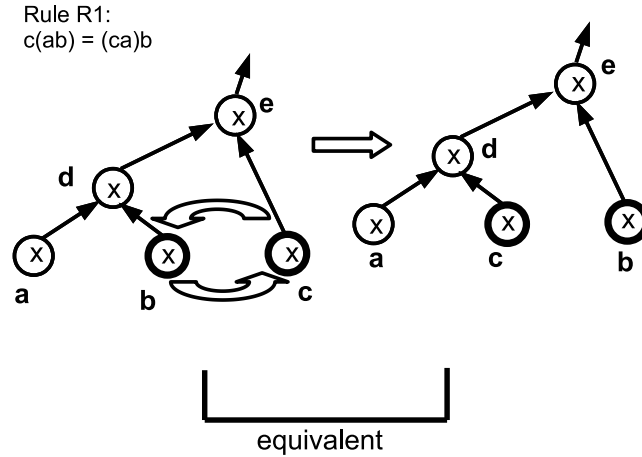


Figure 5.3: Rule  $Rp_1$  on AIG

Under a zero-delay model, due to the swapping of nodes and change in the fanins, the switching power ( $SP(d)$ ) at node  $d$  might decrease or increase. For this, cases are made to direct the swapping of a particular node only when the switching power is getting reduced. Denote the switching power at node  $d$  before swapping, when the fanins of node  $d$  are nodes  $a$  and  $b$ , as  $SP(d)_{ab}$ , switching power at node  $d$  on swapping  $a$  and  $c$  (nodes  $b$  and  $c$  are the fanins of node  $d$ ) as  $SP(d)_{bc}$  and switching power at node  $d$  on swapping  $b$  and  $c$  (nodes  $a$  and  $c$  are the fanins of node  $d$ ) as  $SP(d)_{ac}$ .

If ( $SP(d)_{ab}$  is less than  $SP(d)_{bc}$ ) and ( $SP(d)_{ab}$  is less than  $SP(d)_{ac}$ ), no swapping is required.

If ( $SP(d)_{bc}$  is less than  $SP(d)_{ab}$ ) and ( $SP(d)_{bc}$  is less than  $SP(d)_{ac}$ ), swap nodes  $a$  and  $c$ .

If ( $SP(d)_{ac}$  is less than  $SP(d)_{ab}$ ) and ( $SP(d)_{ac}$  is less than  $SP(d)_{bc}$ ), swap nodes  $b$  and  $c$ .

The same conditions are applied in the rest of the rules, to direct that particular rule application. An example for Rule  $Rp_1$  explanation is presented below.

Consider  $P(a) = 0.2$ ,  $P(b) = 0.6$  and  $P(c) = 0.4$ . The switching probability  $SP(f)_{ab}$  at node  $d$  before swapping (i.e. when nodes  $a$  and  $b$  are the fanins of node  $d$ ) (using the Equation 5.4) is:

$$SP(d)_{ab} = 0.1056 \quad (5.17)$$

The switching probability  $SP(d)_{ac}$  at node  $d$  after swapping nodes  $b$  and  $c$  (i.e. when nodes  $a$  and  $c$  are the fanins of node  $d$ ) (using the Equation 5.4) is:

$$SP(d)_{ac} = 0.073 \quad (5.18)$$

The switching power at node  $d$  reduces by 30%. The estimations are made under a zero-delay model.

Under a non-zero-delay model, not only the fanins but the arrival times and switching instants at nodes  $d$ ,  $e$  and the rest of the nodes change due to swapping. After each swap the arrival times for the rest of the nodes are updated and new switching instants are calculated as described in the previous sections. The switching power at node  $d$  and node  $e$  might decrease or increase, due to changes in the fanins and changes in the switching instants. Consider  $t_1$  to be the switching instant at node  $d$  before swapping and  $t_2$  be the switching instant at node  $d$  after swapping. Consider nodes  $a$  and  $c$  to be the fanins of node  $d$  now. Let the delay at node  $d$  be  $t_{d,d}$ . The initial and final switching probabilities at node  $d$  namely  $P_{sw,t_1}(d)$  and  $P_{sw,t_2}(d)$  are as follows:

$$P_{sw,t_1}(d) = P(d_{t_1} = 0).P(d_{t_1+dt} = 1) + P(d_{t_1} = 1).P(d_{t_1+dt} = 0) \quad (5.19)$$

$$P(d_{t_1} = 0) = [1 - P_{t_1-t_{d,d}}(a)P_{t_1-t_{d,d}}(b)] \quad (5.20)$$

$$P(d_{t_1+dt} = 1) = P_{t_1+dt-t_{d,d}}(a)P_{t_1+dt-t_{d,d}}(b) \quad (5.21)$$

$$P(d_{t_1} = 1) = P_{t_1-t_{d,d}}(a)P_{t_1-t_{d,d}}(b) \quad (5.22)$$

$$P(d_{t_1+dt} = 0) = [1 - P_{t_1+dt-t_{d,d}}(a)P_{t_1+dt-t_{d,d}}(b)] \quad (5.23)$$

$$P_{sw,t_2}(d) = P(d_{t_2} = 0).P(d_{t_2+dt} = 1) + P(d_{t_2} = 1).P(d_{t_2+dt} = 0) \quad (5.24)$$

$$P(d_{t_2} = 0) = [1 - P_{t_2-t_{d,d}}(a)P_{t_2-t_{d,d}}(c)] \quad (5.25)$$

$$P(d_{t_2+dt} = 1) = P_{t_2+dt-t_{d,d}}(a)P_{t_2+dt-t_{d,d}}(c) \quad (5.26)$$

$$P(d_{t_2} = 1) = P_{t_2-t_{d,d}}(a)P_{t_2-t_{d,d}}(c) \quad (5.27)$$

$$P(d_{t_2+dt} = 0) = [1 - P_{t_2+dt-t_{d,d}}(a)P_{t_2+dt-t_{d,d}}(c)] \quad (5.28)$$

$$\begin{aligned} P_{sw,t_1}(d) = & [1 - P_{t_1-t_{d,d}}(a)P_{t_1-t_{d,d}}(b)] \\ & P_{t_1+dt-t_{d,d}}(a)P_{t_1+dt-t_{d,d}}(b) + \\ & P_{t_1-t_{d,d}}(a)P_{t_1-t_{d,d}}(b) \\ & [1 - P_{t_1+dt-t_{d,d}}(a)P_{t_1+dt-t_{d,d}}(b)] \end{aligned}$$

$$\begin{aligned} P_{sw,t_2}(d) = & [1 - P_{t_2-t_{d,d}}(a)P_{t_2-t_{d,d}}(c)] \\ & P_{t_2+dt-t_{d,d}}(a)(1 - P_{t_2+dt-t_{d,d}}(c)) + \\ & P_{t_2-t_{d,d}}(a)(1 - P_{t_2-t_{d,d}}(c)) \\ & [1 - P_{t_2+dt-t_{d,d}}(a)P_{t_2+dt-t_{d,d}}(c)] \end{aligned}$$

#### 5.4.2 Rule $Rp_2$

Consider the AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which link the AND Nodes. Let the conditions for Rule  $Rp_1$  be true and Rule  $Rp_1$  is being followed, such that  $V_1$  has different fanins than before. There can be a case when there already exists a node  $V_3$  in  $V$  with the same fanins and same functionality as that of  $V_1$ . Nodes  $V_1$  and  $V_3$  become redundant and one of the nodes gets removed. Due to reduction of a node, the total switching power of the network decreases. Let  $func(V_1)$  denote the functionality at node  $V_1$ . Let  $func(V_3)$  denote the functionality at node  $V_3$ . Rule  $Rp_2$  at node  $V_1$  is valid according to the following conditions:

- $F(V_1) = 1$
- $f((V_1, V_2)) = 0$
- $func(V_1) = func(V_3)$

If the above conditions are valid, then one of the redundant nodes between  $V_1$  and  $V_3$  is removed from the graph  $G(V, E)$ . Hence with no change in the functionality, one of the nodes gets removed from the AIG network followed by reduction in switching power.

The rule is explained in Figure 5.4, showing sub-graph of an AIG with AND nodes namely  $a, b, c, d, e$  and  $f$ . Rule  $Rp_2$  is applied on node  $d$  as the above conditions are met i.e.  $F(d) = 1$ ,  $f((d, e)) = 0$  and  $func(d) = func(f)$ . Hence the fanin of node  $e$  i.e. node  $c$  can be swapped with the fanin of node  $d$  i.e.  $b$ . The functionality of node  $e$  does not change and since node  $d$  does not have any other fanouts, the swapping does not affect the overall AIG network. Node  $d$  has fanins  $a$  and  $c$ . There already exists a node  $f$  with same fanins  $a$  and  $c$  and same functionality as that of node  $d$ . Hence node  $f$  is removed from the graph. The total switching power of the network gets reduced as the switching power at the redundant node (removed from the network) is not counted. Such swapping of nodes reduces the overall switching power of the AIG network and also reduces the number of nodes.

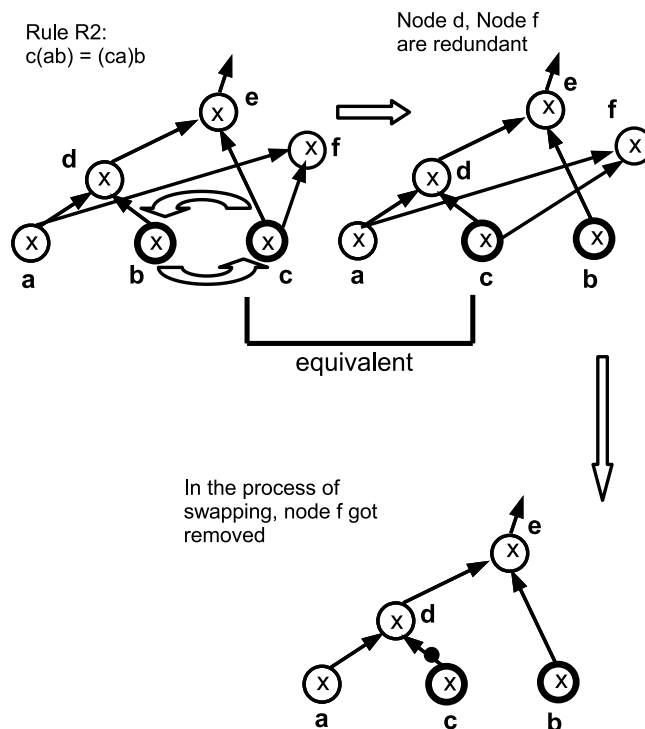


Figure 5.4: Rule  $Rp_2$  on AIG



### 5.4.3 Rule $Rp_3$

Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which links the AND Nodes. Consider  $V_1$  and  $V_3$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Let  $F(V_3)$  be the number of fanouts at node  $V_3$ . Consider another node  $V_2$  in  $V$  such that  $V_1$  and  $V_3$  are the fanins of  $V_2$ . Let  $f((V_1, V_2))$  represent the complement attribute from edge  $V_1$  to  $V_2$  where  $(V_1, V_2)$  is in  $E$ . Similarly, let  $f((V_3, V_2))$  represent the complement attribute from edge  $V_3$  to  $V_2$  where  $(V_3, V_2)$  is in  $E$ . If there is no inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is zero. And if there is an inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is one. Rule  $Rp_3$  at nodes  $V_1$  and  $V_3$  combine is valid according to the following conditions:

- $F(V_1) = 1$
- $F(V_3) = 1$
- $f((V_1, V_2)) = 0$
- $f((V_3, V_2)) = 0$

If the above conditions are valid, then the fanins of node  $V_1$  can be swapped with the fanins of  $V_3$ . This swapping does not affect the functionality of the AIG network but may reduce the total switching power of the network.

Rule  $Rp_3$  is based on the associative rule with four variables i.e.  $(a \times b) \times (c \times d) = a \times (b \times c) \times d$  and the commutative rule i.e.  $a \times b = b \times a$ . This implies that  $(a \times b) \times (c \times d) = a \times (b \times c) \times d = a \times (c \times b) \times d = (a \times c) \times (b \times d)$ . Rule  $Rp_3$  is explained in Figure 5.5 showing a part of an AIG network with AND nodes namely  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ ,  $f$  and  $g$ . Rule  $Rp_3$  is applied on node  $e$  and  $f$  as the above conditions are met i.e.  $F(e) = 1$ ,  $F(f) = 1$ ,  $f((e, g)) = 0$  and  $f((f, g)) = 0$ . Hence the fanin of node  $e$  i.e. node  $b$  can be swapped with the fanin of node  $f$  i.e.  $c$ . The functionality of node  $g$  does not change and since node  $e$  and  $f$  does not have any other fanouts, the swapping does not affect the overall AIG network.

In the Figure 5.5, on swapping nodes  $b$  and  $c$ , nodes  $b$  and  $d$  are the fanins of node  $f$ , and nodes  $a$  and  $c$  are the fanins of node  $e$ . This swapping creates redundant nodes which can be removed from the graph. On the other hand, consider  $SP(e)_{ab}$  and  $SP(f)_{cd}$  as the switching probability at node  $e$  and  $f$  respectively before swapping. If

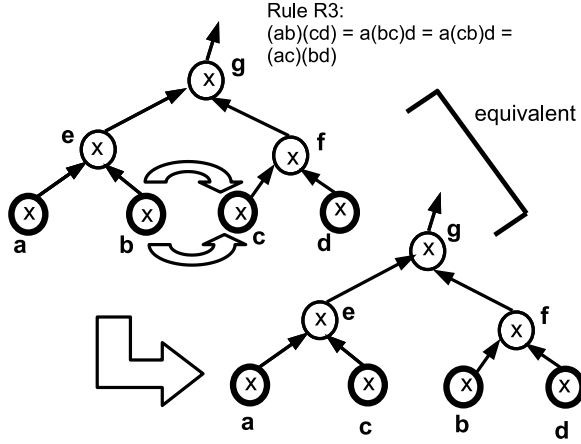


Figure 5.5: Rule  $Rp_3$  on AIG

nodes  $a$  and  $c$  are swapped, consider  $SP(e)_{bc}$  and  $SP(f)_{ad}$  as the switching probability at node  $e$  and  $f$  respectively after swapping nodes  $a$  and  $c$ . There can be a case when  $SP(e)_{ab} + SP(f)_{cd}$  is greater than  $SP(e)_{bc} + SP(f)_{ad}$ . In that case, switching power is getting reduced. Hence Rule  $Rp_3$  actually combines the options available by rules  $Rp_1$  and  $Rp_2$ . An example for Rule  $Rp_3$  implementation (under zero-delay model) is presented below.

Consider  $P(a) = 0.7$ ,  $P(b) = 0.3$ ,  $P(c) = 0.4$  and  $P(d) = 0.2$ . The switching power  $SP(e)_{ab}$  at node  $e$  before swapping (i.e. when nodes  $a$  and  $b$  are the fanins of node  $e$ ) is:

$$SP(e)_{ab} = 0.3318 \quad (5.29)$$

The switching power  $SP(f)_{cd}$  at node  $f$  before swapping (i.e. when nodes  $c$  and  $d$  are the fanins of node  $f$ ) is:

$$SP(f)_{cd} = 0.1472 \quad (5.30)$$

The switching power  $SP(e)_{bc}$  at node  $e$  after swapping (i.e. when nodes  $b$  and  $c$  are the fanins of node  $e$ ) is:

$$SP(e)_{bc} = 0.2112 \quad (5.31)$$

The switching power  $SP(f)_{ad}$  at node  $f$  after swapping (i.e. when nodes  $a$  and  $d$  are the fanins of node  $f$ ) is:

$$SP(f)_{ad} = 0.2408 \quad (5.32)$$

The sum of the switching power at nodes  $e$  and  $f$  before swapping is:

$$SP(e)_{ab} + SP(f)_{cd} = 0.479 \quad (5.33)$$

The sum of the switching power at nodes  $e$  and  $f$  after swapping is:

$$SP(e)_{bc} + SP(f)_{ad} = 0.452 \quad (5.34)$$

The switching power gets reduced by 5.6%.

Under a non-zero-delay model, this swapping changes the arrival times, switching instants and hence switching power at nodes  $e$ ,  $f$ ,  $g$  and the rest of the nodes as well. After every swap the arrival times and switching instants has to be updated. Due to change in the fanins and switching instants, the switching power at various nodes change without changing the functionality. The present and the final sum of switching power need to be compared for the implementation of Rule  $Rp_3$ . The swapping is not affecting the overall AIG network, however the swapping has the potential to either reduce the switching power or find redundant nodes which can be removed from the graph. Hence Rule  $Rp_3$  actually combines the options available by the rules  $Rp_1$  and  $Rp_2$ .

#### 5.4.4 Rule $Rp_4$

Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which links the AND Nodes. Consider  $V_1$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Consider another node  $V_2$  in  $V$  such that  $V_1$  is a fanin of  $V_2$ . Let  $f((V_1, V_2))$  represent the complement attribute from edge  $V_1$  to  $V_2$  where  $(V_1, V_2)$  is in  $E$ . If there is no inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is zero. And if there is an inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is one. Rule  $Rp_4$  at node  $V_1$  is valid according to the following conditions:

- $F(V_1) = 1$
- $f((V_1, V_2)) = 1$

If the above conditions are valid, then the fanin of node  $V_2$  other than  $V_1$  is swapped with one of the fanins of  $V_1$ . A new node  $V_3$  in  $V$  is also created with one of the fanins of  $V_1$  and one of the fanins of  $V_2$  following the distributive rule such that the functionality of the network does not change. This swapping creates an extra node but does not affect the functionality of the AIG network. The switching power may get reduced even after increment of a node.

The rule is based on the distributive rule i.e.  $a \times (b + c) = (a \times b) + (a \times c)$  and De Morgan's Theorem i.e.  $(a' \times b')' = a + b$  and  $(a' + b')' = a \times b$ . This implies that  $(a' \times b')' \times c = (a + b) \times c = (a \times c) + (b \times c) = [(a \times c)' \times (b \times c)']'$ . The rule is explained in Figure 5.6, showing a part of an AIG network with AND nodes namely  $a, b, c, d$  and  $e$ . Rule  $Rp_4$  is applied on node  $d$  as the above conditions are met i.e.  $F(d) = 1$  and  $f((d, e)) = 1$ . In this case,  $e = (a' * b')' * c$  as  $(a * c) + (b * c)$  can be written following the distributive rule. So an extra AND node  $f$  is created with fanins  $b$  and  $c$ , while node  $d$  has fanins  $a$  and  $c$ . Rule  $Rp_4$  increases the node count by one but may reduce the switching power of the network significantly. Under a non-zero-delay model, the new node has the same node delay as that of node  $d$ .

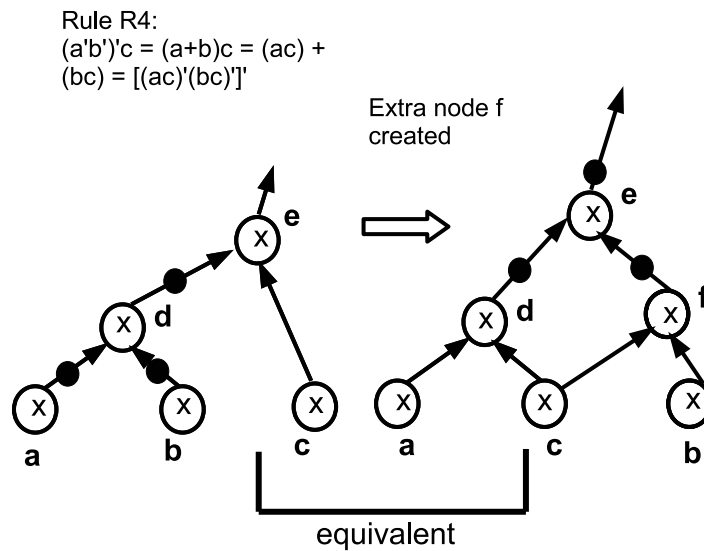


Figure 5.6: Rule  $Rp_4$  on AIG

These four rules (for switching power reduction) are applied on each of the AIG nodes, depending upon the conditions for the applicability of the rules. After each rule application, the switching power  $SP(f)$ , the critical path length  $MAT(f)$  and the total area might change.

## 5.5 Reordering rules for delay optimisation

We have introduced three reordering and rewriting rules namely  $Rd_1$ ,  $Rd_2$  and  $Rd_3$  which help us reduce the overall longest path delay of the digital circuit structurally represented as an AIG network. These rules are also based on swapping and interchanging of variables which change the structuring of the AIG network while maintaining the functionality.

### 5.5.1 Rule $Rd_1$

Rule  $Rd_1$  states that the last arriving signal (fanin with maximum arrival time) in an AIG tree must be closer to the output, or closer to the root node of a sub-graph [61]. Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which links the AND Nodes. Consider  $V_1$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Consider another node  $V_2$  in  $V$  such that  $V_1$  is a fanin of  $V_2$ . Also consider node  $V_2$  to be the root node of a sub-graph. Let  $f((V_1, V_2))$  represent the complement attribute from edge  $V_1$  to  $V_2$  where  $(V_1, V_2)$  is in  $E$ . If there is no inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is zero. And if there is an inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is one. Rule  $Rd_1$  at node  $V_1$  is valid according to the following conditions:

- $F(V_1) = 1$
- $f((V_1, V_2)) = 0$

If the above conditions are valid, then the fanin of node  $V_2$  other than  $V_1$  and the fanins of node  $V_1$  are compared and swapped among each other such that the last arriving signal coming from these fanins is collapsed to the root node of this sub-graph which is node  $V_2$ . This rearrangement reduces the arrival time at the root node  $V_2$  and hence reduces the longest path delay of the whole network. The functionality of the network

does not change if the above conditions are valid.

Rule  $Rd_1$  is explained in Figure 5.7, showing a part of an AIG network with AND nodes namely  $a$ ,  $b$ ,  $c$ ,  $d$  and  $e$ . Rule  $Rd_1$  is applied on node  $d$  as the above conditions are met i.e.  $F(d) = 1$  and  $f((d, e)) = 0$ . In this case, the fanins of node  $d$  and  $e$  are compared. Hence the arrival times of  $a$ ,  $b$  and  $c$  are compared. Node  $b$  has the highest arrival time among the three signals. Hence the signals are rearranged at this level such that the last arriving signal i.e.  $b$  is collapsed to the root node of this sub-graph i.e.  $e$ , this implies that now node  $b$  is the fanin of the root node  $e$ . This rearrangement is not changing the functionality of the network but reduces the arrival time at  $e$  and hence the longest path delay  $MAT(f)$  of the network.

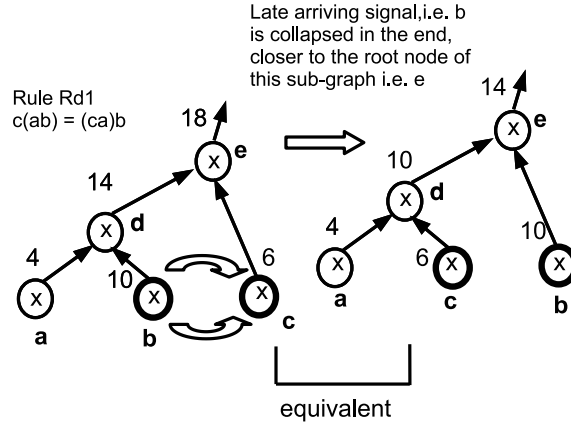


Figure 5.7: Rule  $Rd_1$  on AIG

In Figure 5.7, let the arrival time at nodes  $a$ ,  $b$ ,  $c$ ,  $d$  and  $e$  be 4, 10, 6, 14 and 18 respectively. All the AND nodes have delay of 4. After Rule  $Rd_1$ , the arrival times at nodes  $a$ ,  $b$ ,  $c$ ,  $d$  and  $e$  are 4, 10, 6, 10 and 14 respectively, using Equation 5.8. The arrival time at node  $e$  is reduced from 18 to 14 on the application of Rule  $Rd_1$ .

### 5.5.2 Rule $Rd_2$

Rule  $Rd_2$  is based on AIG node duplication [15]. Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which links the AND Nodes. Consider  $V_1$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Rule  $Rd_2$  at node  $V_1$  is valid according to the following conditions:

- $F(V_1) > 1$

If the above condition is valid, then node  $V_1$  is duplicated and another node  $V_2$  in  $V$  is created. Both nodes  $V_1$  and  $V_2$  have the same fanins and same functionality. Node  $V_2$  has one of the fanouts of node  $V_1$ . Due to this duplication, the fanouts at node  $V_1$  decreases. This reduction in the fanouts reduces the arrival time at node  $V_1$  according to Equation 5.8. This duplication of a node increases the node count by one but can reduce the arrival time at  $V_1$  and hence the longest path delay of the network. The reduction in the arrival time is explained in the following example.

Rule  $Rd_2$  is explained in Figure 5.8 showing a sub-graph of an AIG with AND nodes namely  $a, b, c, d, e, f$  and  $g$ . Rule  $Rd_2$  is applied on node  $c$  as the above condition is valid i.e.  $F(c) > 1$ . According to Equation 5.8, the arrival time on a node is directly proportional to the number of fanouts at that node. Hence if we reduce the number of fanouts at a given node, the arrival time of that node decreases. The same concept is applied in this rule. To reduce the number of fanouts at  $c$ , node  $c$  is duplicated as node  $d$ . Node  $d$  has the same node delay, same fanins and same functionality as that of node  $c$ . One of the fanins of node  $c$  i.e.  $e$  is linked as fanout of node  $d$ . This duplication is not changing the functionality, may lead to a slight area increase but can bring about some significant reduction in delay at node  $c$ . The delay at node  $c$  always decreases due to decrease in the number of fanouts at that node.

In the Figure 5.8, node  $c$  with three fanouts  $e, f$  and  $g$ . Consider the arrival time at node  $c$  to be  $x + FAND.F(c)$ , where  $x = 10$ ,  $FAND = 2$  and  $F(c) = 3$ , hence according to Equation 5.8,  $t_{arr,c} = 16$ . Once node  $c$  is duplicated with a new node  $d$  having the same fanins as node  $c$ , the arrival times at node  $c$  and  $d$  become 14 and 12 respectively. This is caused by the reduction in the number of fanouts. Eventually,  $c$  has two fanouts and  $d$  has one fanout.

### 5.5.3 Rule $Rd_3$

Rule  $Rd_3$  combines the use of Rule  $Rd_1$  and  $Rd_2$  in one step. Consider an AIG network represented by a graph  $G(V, E)$  where  $V$  (vertices) is the set of AND nodes and  $E$  is the set of edges which links the AND Nodes. Consider  $V_1$  in  $V$ . Let  $F(V_1)$  be the number of fanouts at node  $V_1$ . Consider another node  $V_2$  in  $V$  such that  $V_1$  is a fanin of  $V_2$ . Consider another node  $V_3$  in  $V$  such that  $V_1$  is a fanin of  $V_3$ . Let  $f((V_1, V_2))$

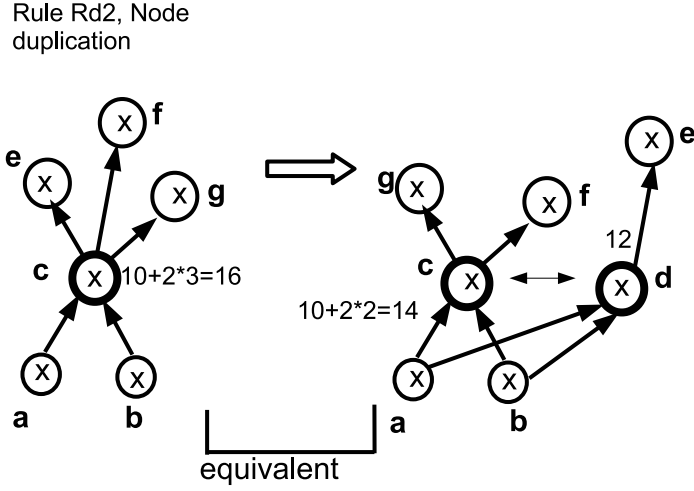


Figure 5.8: Rule  $Rd_2$  on AIG

represent the complement attribute from edge  $V_1$  to  $V_2$  where  $(V_1, V_2)$  is in  $E$ . If there is no inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is zero. And if there is an inversion on the edge  $(V_1, V_2)$ ,  $f((V_1, V_2))$  is one. Let  $f((V_1, V_3))$  represent the complement attribute from edge  $V_1$  to  $V_3$  where  $(V_1, V_3)$  is in  $E$ . Rule  $Rd_3$  at node  $V_1$  is carried out in three steps, with initial conditions as follows:

- $F(V_1) > 1$

If the above condition is valid, node  $V_1$  is duplicated as  $V_4$  in  $V$  (following  $Rd_2$ ). Node  $V_4$  gets linked to one of the fanouts of node  $V_1$  say  $V_3$  which implies  $F(V_4) = 1$ . The second step is carried out with the following conditions:

- $f((V_4, V_3)) = 0$

If the above condition is valid then the fanin of node  $V_3$  other than  $V_4$  and the fanins of node  $V_4$  are compared and swapped among each other (following Rule  $Rd_1$ ). The last arriving signal coming from these fanins is collapsed to the root node of this sub-graph which is node  $V_3$  (following  $Rd_1$ ). This rearrangement reduces the arrival time at the root node  $V_3$  and hence reduces the longest path delay of the whole network. The functionality of the network does not change if the above conditions are valid. The third step is carried out with the following conditions:



- $F(V_1) = 1$
- $f((V_1, V_2)) = 0$

If the above conditions are valid then the fanin of node  $V_2$  other than  $V_1$  and the fanins of node  $V_1$  are compared and swapped among each other such that the last arriving signal coming from these fanins is collapsed to the root node of this sub-graph which is node  $V_2$  (following  $Rd_1$ ). This rearrangement reduces the arrival time at the root node  $V_2$  and hence reduces the longest path delay of the whole network. The functionality of the network does not change if the above conditions are valid.

Rule  $Rd_3$  is explained in Figure 5.9 showing a sub-graph of an AIG with AND nodes namely  $a, b, c, d, e, f, g$  and  $h$ . The first step of node duplication is applied on node  $c$  as the above condition is valid i.e.  $F(c) > 1$ . A new node  $d$  is created with same fanins as that of  $c$  namely  $a$  and  $b$ . Node  $d$  has node  $e$  as its fanout. The second step of swapping signals on the duplicated node  $d$  is applied as the condition  $f((d, e)) = 0$  is valid. The signals  $a, b$  and  $g$  are rearranged such that the last arriving signal is collapsed closer the root node of this sub-graph which is node  $e$  (also stated in Rule  $Rd_1$ ). In the Figure 5.9, the arrival time at  $a, b$  and  $g$  are 6, 10 and 14 respectively. Signal  $b$  is collapsed with the root node  $e$  since it is the last arriving signal. The third step is also valid as the conditions  $F(c) = 1$  and  $f((c, f)) = 0$ . The signals  $a, b$  and  $h$  are rearranged such that the last arriving signal is collapsed closer the root node of this sub-graph which is node  $f$  (as stated in Rule  $Rd_1$ ). In the Figure 5.9, the arrival time at  $a, b$  and  $h$  are 6, 10 and 14 respectively. Signal  $h$  is the last arriving signal which is collapsed with the root node  $f$  already. The functionality of the network does not change with any of the above transformations. However the rule helps in some significant reduction in the longest path delay ( $MAT(f)$ ) of the network.

## 5.6 Conclusions

In this chapter, we have introduced a set of reordering rules for power and delay optimisation. The synthesis and optimisation method is based on AIGs. The digital logic circuit is structurally represented as an AIG network and switching power and longest path delay of the circuit is estimated on this network. The rules are applied on each of the AIG nodes. However there are two issues which have to be discussed with these

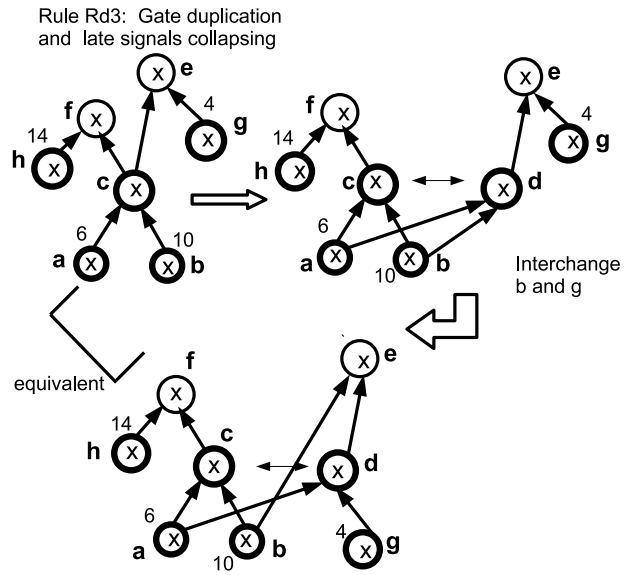


Figure 5.9: Rule  $Rd_3$  on AIG

rules and they are discussed in the next chapter. Firstly if more than one rule is possible on a given node, then to decide which rule is most beneficial for the power and delay optimisation, various algorithms are introduced in the next chapter. Secondly since the goal of our thesis is delay-driven power optimisation and power-driven delay optimisation, the rules must account for the increase in delay with every decrease in switching power and similarly the increase in power with every decrease in longest path delay. In the next chapter, we discuss various algorithms and techniques which are used in our work for the rule implementation on the AIG nodes. We used these algorithms to get better optimisation results and for the implementation of power-driven delay optimisation and delay-driven power optimisation.

## Chapter 6

# Implementation of reordering rules for power and delay optimisation

The previous chapter introduces a set of reordering rules for power and delay optimisation. This chapter presents the implementation of the introduced rules. Initially our method only targeted low power optimisation schemes. This implies decreasing the switching power ( $SP(f)$ ) with no constraints on the longest path delay ( $MAT(f)$ ) of the network. According to the conditions for the applicability of any rule, if all the four rules of power optimisation namely  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$  are possible on a given node, then the rule which gives maximum reduction of switching power is chosen. Same applies when any three rules are possible on a given node. The rules applied in this method may decrease the switching power greedily but may increase the  $MAT(f)$  of the network largely. The algorithm applied in this implementation is a Greedy Algorithm [32] i.e. the rules are applied recursively at locally optimal AIG nodes until the minimal sum of switching power is achieved. The optimisation scheme proposed in this method is local and more emphasis will be on global and multi-objective optimisation in the further optimisation schemes. Also Greedy Algorithm may not always yield global optimisation. This implies that if at each step we choose the locally optimal node (i.e. the local node which gives maximum reduction in switching power on the application of the reordering rules), then it may not always give the global minimum. Hence, we intro-

duced optimisation schemes which firstly do not stuck at local minimum and secondly they look globally into the AIG network to find an order or sequence of nodes in which the rules have to be applied such that it gives better results than Greedy Algorithm.

In Section 6.1, implementation of low power reordering rules (introduced in the previous chapter) under Greedy Algorithm is shown. Section 6.2 shows some simulation results on the application of the reordering rules on few benchmark circuits using a new tool AIG2Net. Section 6.3 introduces the concept of delay-driven power optimisation and power-driven delay optimisation. We introduce a new tool OPT-PT implemented as a sub-package in ABC for multi-objective optimisation of power and delay. Section 6.4 introduces the application of Simulated Annealing in multi-objective optimisation of power and delay. Section 6.5 introduces the application of Uniform Cost Search Algorithm for global search within the AIG network for reduced power and delay values. Finally, section 6.6 gives concluding remarks on the various optimisation schemes introduced in this chapter.

## 6.1 Implementation of low power optimisation rules using Greedy Algorithm

We have introduced a tool called RESWITCH implemented in C as a sub-package in ABC. The tool does power optimisation using the four rules of power optimisation. The RESWITCH tool takes a set of MCNC Benchmark circuits as inputs. The BLIF file of such a circuit is taken and converted to an AIG network in ABC. The AIG network is then synthesised and optimised in ABC using one of the best synthesis script *resyn*. This script is used for reducing the network complexity in terms of AIG nodes. In this script, the series of synthesis commands are *balancing*, *rewriting*, *rewriting with zero cost replacement*, *balancing*, *rewriting with zero cost replacement* and *balancing*. These synthesis commands are used for the reduction of nodes and complexity. After the application of the synthesis script, the switching power is estimated on each of the AIG nodes as mentioned in subsection 5.3.2, using Equation 5.16 followed by the RESWITCH application. The RESWITCH tool gives power optimised circuit as output. Table 6.1 presents the overall decrease in the sum of the switching power of the nodes. It also specifies the decrease in the number of nodes of the AIG network. In the

table,  $SP(f)_b$  and  $SP(f)_a$  stand for switching power sum before and after the application of RESWITCH (again estimated using Equation 5.16).  $NC_b$  and  $NC_a$  stand for node count (AIG nodes in the AIG network) before and after the RESWITCH application.  $\Delta SP(f)$  is the change in switching power in percentage.  $\Delta NC$  is the percentage change in AIG node count. The overall decrease in the switching power is 14.90%.

Table 6.1:  $SP(f)$  without and with the application of rewriting rules  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$

Circuit	$SP(f)_b$	$SP(f)_a$	$NC_b$	$NC_a$	$\Delta SP(f)$	$\Delta NC$
-	( $\mu W$ )	( $\mu W$ )	-	-	-	-
b1	2.56	2.32	10	10	-9.37%	-
cc	12.67	10.47	51	53	-17.36%	3.90%
cm162a	6.57	5.73	33	34	-12.78%	3.02%
mux	15.40	13.24	81	83	-14.02%	2.54%
alu2	63.41	55.75	361	370	-12.00%	2.56%
alu4	117.97	101.53	657	675	-14.00%	2.78%
x2	7.53	6.30	45	47	-16.33%	4.49%
cmb	5.86	4.43	37	39	-23.37%	5.47%

## 6.2 Architecture and application of AIG2Net

The above results are estimated within ABC at pre-technology mapping level. Technology mapping refers to the final step of the logic synthesis and optimisation task. In technology mapping, library gates are selected to realise the design that has been structurally optimised, in our case by the four rewriting rules. We introduce a new tool AIG2Net implemented in C. This tool converts the AIG network of each Benchmark circuit obtained within ABC to a netlist form. The netlist is actually a technology mapped design written in Verilog, which can be read and compiled in Design Compiler DC. In this translation, each AND node of the AIG network is converted to 2-input

AND cell from a specific library. The complemented (inverted) edges are replaced by inverter cells and non complemented edges are replaced by buffer cells. The names of AIG nodes and edges to the netlist cells and wires respectively is also kept same. In this way exact technology mapping of the AIG network to a AIG netlist (in Verilog) is possible. Figure 6.2 shows a mapped Verilog AIG netlist of an AIG network representing functionality  $a(b + c)$  obtained from AIG2Net. The corresponding AIG network of functionality  $a(b + c)$  is shown in Figure 6.1.

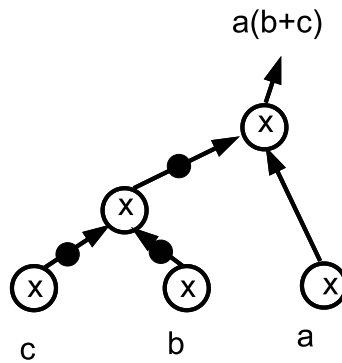


Figure 6.1: AIG of logic function  $a(b + c)$

With the netlist obtained, DC from Synopsys is used to report the power. For the dynamic power dissipation, a user defined switching activity file SAIF is also provided. The SAIF files are used and generated by various commercial tools. These files define the switching activity of all the primary inputs, primary outputs and internal cells of the netlist. A similar user defined SAIF file is created for the circuits specifying the switching activity approximated as switching probabilities annotated at all the AND nodes (replaced by AND cells). This SAIF file helps map the earlier calculated and minimised switching probabilities on the netlist for exact dynamic power estimation. Figure 6.3 shows a part of the user defined SAIF file given as an input to the netlist (shown in Figure 6.2) for power calculation. This SAIF file specify the switching probabilities estimated and optimised on the AIG network representing functionality  $a(b + c)$ .

```

module peg (a, b, c, f );
input a, b, c ;
output f ;
wire n2;
INVD0 U2z (.I(b), .ZN(n2));
wire n5;
AN2D0 U5 (.A1(n2), .A2(c), .Z(n5));
wire n5p5;
INVD0 U5z (.I(n5), .ZN(n5p5));
wire n6;
AN2D0 U6 (.A1(n2), .A2(n5p5), .Z(n6));
wire n6p6;
INVD0 U6z (.I(n6), .ZN(n6p6));
wire n7;
AN2D0 U7 (.A1(a), .A2(n6p6), .Z(n7));
BUFFD0 U4 (.I(n7), .Z(f));
endmodule

```

Figure 6.2: Mapped Verilog netlist of an AIG network

```

.....

set_switching_activity n2 -static_prob
0.032000 -toggle_rate 0.061952
set_switching_activity n5 -static_prob
0.138807 -toggle_rate 0.239079
set_switching_activity n6 -static_prob
0.138807 -toggle_rate 0.239079
.....

```

Figure 6.3: User defined switching activity file for the netlist shown in Figure 6.2

### 6.2.1 Estimation/Optimisation Design Flow

For technology-mapped power results, the flow diagram as shown in Figure 6.4 is followed. The input files are BLIF files. The BLIF format is the most common format used in academic tools. The MCNC Benchmark circuits are read as BLIF files into ABC. ABC converts this file into an AIG network. There are two sets of networks being compared - the ABC-optimised AIG network and the corresponding RESWITCH-optimised AIG network.

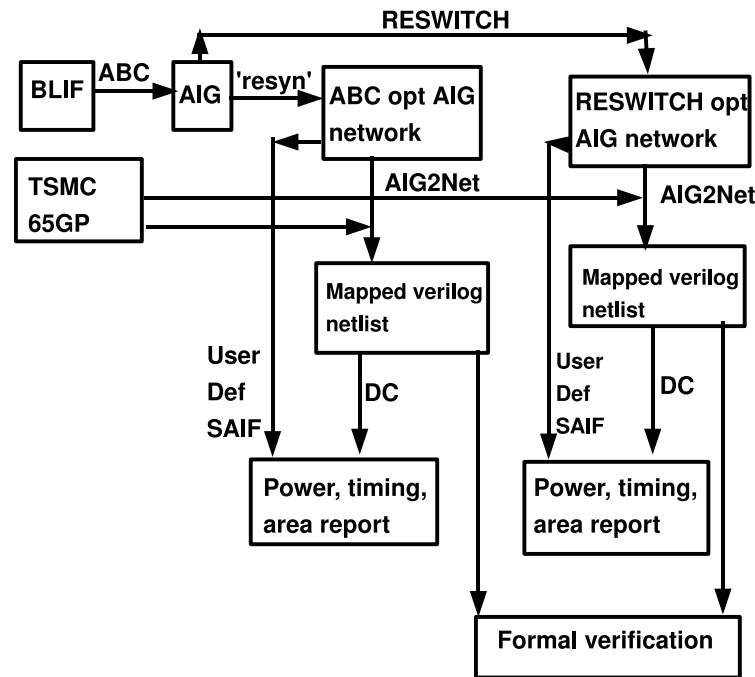


Figure 6.4: The design flow for power, delay and area estimation

The tool AIG2Net is used for the mapping of the AIG network to a Verilog netlist using 2-input AND, NOT and buffer cells from the TSMC 65GP CMOS standard cell library. Two sets of netlists are obtained - the ABC-optimised AIG netlist and RESWITCH-optimised AIG netlist. A user-defined SAIF (as shown in the Figure 6.4) is also provided with respect to the two networks. The SAIF file corresponding to the ABC-optimised AIG netlist relates to the initial switching probabilities. The SAIF file corresponding to the RESWITCH-optimised AIG netlist relates to the reduced switching probabilities after applying the reordering rules. At the end of the process, Synopsys Design Compiler (DC) is used for power, delay and area estimation on the two netlists. Also the two networks' netlists are verified for consistency by Synopsys



Formality.

The basis of comparison in the experimental results is as follows. All the power estimations and optimisations are performed on an AIG network. Firstly, the AIG network is optimised using the *resyn* script in ABC, which is the most efficient synthesis script available for reducing complexity of the AIG network with respect to node count. Secondly, the AIG network is optimised using our tool i.e. RESWITCH. The two optimised AIG networks are converted to mapped Verilog netlists (consisting of 2-input ANDs, Inverters and Buffer cells, still representing the same AIG network). Power reports are then generated in Design Compiler (Synopsys) and the results are compared.

Circuits used are MCNC Benchmark circuits which are generally used in literature. Table 6.2 presents few examples. Power results are in  $\mu W$ , corresponding to the AIG netlist with and without RESWITCH.  $SP(f)$  stands for power.  $\Delta SP(f)$  denotes the change in the switching power.

After power optimisation and technology mapping of various MCNC Benchmark Circuits, we obtained an average power reduction of 17.52%. AIG is used in our work as it gives much more compact representation than BDD and it allows both delay and power to be analysed and optimised, as dealt with in later sections of the chapter. The technology mapping tool AIG2Net is fast, accurate and makes exact name mapping of AIG network (nodes and edges) to the AIG netlist (cells and wires) possible. Use of AIGs at this stage helped in easy and fast technology mapping for realistic observation.

### 6.3 Delay-driven power optimisation and power-driven delay optimisation

In the previous section, we dealt with low power optimisation schemes using Greedy Algorithm. However the impact of decrease in switching power on the longest path delay is not considered. The main objective of our thesis is delay-driven power optimisation and power-driven delay optimisation. As discussed in Chapter 1, we introduced the concept of delay-driven power optimisation and power-driven delay optimisation. Delay-driven power optimisation implies two things, firstly the increase in longest path delay shall not exceed a given constraint while optimising for dynamic power. Secondly for each application of the reordering rules on the AIG nodes, a weight is calculated

Table 6.2: Power dissipation with and without the RESWITCH tool

Circuit	$SP(f)_{beforeRESWITCH}$	$SP(f)_{afterRESWITCH}$	$\Delta SP(f)$
-	( $\mu W$ )	( $\mu W$ )	-
cmb	31.86	25.48	-20.01%
b1	9.00	6.52	-27.03%
c8	75.18	64.91	-13.55%
cm162a	8.65	7.00	-19.07%
alu2	94.35	83.97	-11.09%
x2	27.96	23.31	-16.80%
cc	20.76	17.56	-15.42%

which is directly proportional to the decrease in switching power and inversely proportional to the increase in delay. These weights hence keep a check on every bad move where delay increases largely in the process of switching power reduction. Similarly power-driven delay optimisation is defined implying the same two things, firstly the increase in dynamic power shall not exceed a given constraint while optimising for delay. Secondly for each application of the reordering rules on the AIG nodes, a weight is calculated which is directly proportional to the decrease in delay and inversely proportional to the increase in dynamic power. The following method is introduced using two optimisation algorithms namely Simulated Annealing and Uniform Cost Search Algorithm. Hence multi-objective optimisation of two conflicting objectives (namely power and delay) subjected to each other's constraints, is dealt using these algorithms. We have introduced a new tool called OPT-PT (implemented in C as a sub-package in ABC) for the multi-objective optimisation of power and delay. OPT-PT has two components - OPT-P for delay-driven power optimisation and OPT-T for power-driven delay optimisation.

The OPT-P tool is based on the four reordering rules of power optimisation namely  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$ . However the rule's application is based on the two algorithms namely Simulated Annealing and Uniform Cost Search Algorithm. This will reduce

the overall switching power  $SP(f)$  such that the maximum arrival time or longest path delay ( $MAT(f)$ ) in the AIG network does not exceed a given constraint. This delay-driven power optimisation facilitates a decrease in power while constraining the tool to a slight increase in delay and/or area. Also the order of rule application on the AIG nodes of the network is determined according to Uniform Cost Search Algorithm.

The OPT-T for delay optimisation is based on three reordering rules  $Rd_1$ ,  $Rd_2$  and  $Rd_3$ . However the rule's application is based on the two algorithms namely Simulated Annealing and Uniform Cost Search Algorithm. This will reduce the critical path length  $MAT(f)$  such that the switching power ( $SP(f)$ ) in the AIG network does not exceed a given constraint. This power-driven delay optimisation facilitates a decrease in delay while constraining the tool to a slight increase in power and/or area. Also the order of rule application on the AIG nodes of the network is determined according to Uniform Cost Search Algorithm.

## 6.4 Simulated Annealing

The reordering rules for power and delay optimisation can be applied on each node of the AIG network depending upon the conditions mentioned above. Simulated Annealing (SMA) [31] is used as the optimisation algorithm, to decide which of the rules (if any) on a given AIG node is appropriate for delay-driven power optimisation or power-driven delay optimisation. The cost function  $SP(f)$  (which is the total switching power of the AIG network representing the functionality  $f$  of the circuit) and  $MAT(f)$  (which is the longest path delay or maximum arrival time of the AIG network) has to be reduced for power and delay optimisation respectively. At each step, the Simulated Annealing method considers some neighbour  $s'$  of the current state  $s$ . In our application,  $s$  is the initial AIG network prior to rule application and  $s'$  is the final AIG network after rule application. Our goal is to probabilistically decide between moving the system to state  $s'$  or staying in state  $s$  such that the cost function is minimised. Instead of probabilities, weights have been defined for each movement at each node. These weights depend upon the difference in the cost function before and after rule implementation. The weights also depend on a control parameter  $T$ .  $Wp_1$ ,  $Wp_2$ ,  $Wp_3$  and  $Wp_4$  are the weights corresponding to the four rules for power optimisation i.e.  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$  respectively.  $Wd_1$ ,  $Wd_2$  and  $Wd_3$  are the weights corresponding to the three rules

for delay optimisation i.e.  $Rd_1$ ,  $Rd_2$  and  $Rd_3$  respectively. The weights are determined in accordance with the method in [31], with slight modifications. In the formulation of the Simulated Annealing method [31], the acceptance function of any move from  $s$  to the neighbouring state  $s'$  is defined as  $\exp((e - e')/T)$  where  $e$  is the system energy at the present state,  $e'$  is the system energy of the neighbouring state and  $T$  is the temperature or the control parameter in the Simulated Annealing method. Hence the function is directly proportional to the decrease in the system energy and inversely proportional to the control parameter. A similar technique has been used while defining the weights of the Simulated Annealing method used in our case.

While applying the OPT-P tool, the cost function is  $SP(f)$  (which has to be minimised) and the control parameter  $T$  is the difference between the initial and the final MAT of the network obtained at each step. This allows us to perform delay-driven power optimisation. The weights are described below. Note that the weights are positive even when the cost function increases slightly. This is mainly performed in the simulated annealing method to avoid local minimum.  $\Delta p_{OPT-P}$  and  $\Delta t_{OPT-P}$  for the weight calculation are defined as follows:

$$\Delta p_{OPT-P} = SP_i(f) - SP_f(f) \quad (6.1)$$

where  $SP_i(f)$  is the total switching power of the network before the given rule application.  $SP_f(f)$  is the total switching power of the network after the given rule application.

$$\Delta t_{OPT-P} = MAT_f(f) - MAT_i(f) \quad (6.2)$$

where  $MAT_i(f)$  is the MAT of the network before the given rule application.  $MAT_f(f)$  is the MAT of the network after the given rule application. The weights are defined as follows:

If  $\Delta p_{OPT-P} > 0$  and  $\Delta t_{OPT-P} > 0$ , then  $W = e^{\Delta p_{OPT-P}/\Delta t_{OPT-P}}$

If  $\Delta p_{OPT-P} > 0$  and  $\Delta t_{OPT-P} < 0$ , then  $W = e^{\Delta p_{OPT-P}*(-1*\Delta t_{OPT-P})}$

If  $\Delta p_{OPT-P} > 0$  and  $\Delta t_{OPT-P} == 0$ , then  $W = e^{\Delta p_{OPT-P}}$

If  $\Delta p_{OPT-P} < 0$  and  $\Delta t_{OPT-P} > 0$ , then  $W = 1/e^{(-1*\Delta p_{OPT-P})*\Delta t_{OPT-P}}$

If  $\Delta p_{OPT-P} < 0$  and  $\Delta t_{OPT-P} < 0$ , then  $W = 1/e^{(-1*\Delta p_{OPT-P})/\Delta t_{OPT-P}}$

If  $\Delta p_{OPT-P} < 0$  and  $\Delta t_{OPT-P} == 0$ , then  $W = 1/e^{-1*\Delta p_{OPT-P}}$

The stop condition weights are defined as follows:

If  $\Delta p_{OPT-P} = 0$ , then  $W = 0$

If  $\Delta t_{OPT-P}/MAT_i(f) > 0.1$ , then  $W = 0$

Algorithm 1 describes the technique as used in OPT-P.

The stop condition weights indicate two things. Firstly, if there is no change in the switching power, on the application of that rule, the weight corresponding to that rule is set to zero. Secondly, the weight of a given rule is set to zero if the application of that rule exceeds delay limitations. This implies that if there is more than 10% increase in delay, on the application of that rule, the weight is set to zero. The definition of weights (i.e. directly proportional to the decrease in switching power and inversely proportional to the increase in delay) and the second stop condition clearly shows the concept of delay-driven power optimisation. In our case, we have set the constraint limits on delay to be 10%.

The weights are directly proportional to the decrease in switching power and inversely proportional to the increase in MAT, as indicated in the weight equations above. Consider a case, when Rule  $Rp_x$  ( $x \in 1, 2, 3, 4$ ) is applied on a given node on a given AIG network of a circuit. Assume that on applying  $Rp_x$ ,  $\Delta p_{OPT-P} = 0.2$  and  $\Delta t_{OPT-P} = 0.1$ . Consider another case, when Rule  $Rp_y$  ( $y \in 1, 2, 3, 4$ ) is applied on the given node of the given AIG network of a circuit. Assume that on applying Rule  $Rp_y$ ,  $\Delta p_{OPT-P} = 0.3$  and  $\Delta t_{OPT-P} = 0.3$ . Although Rule  $Rp_y$  gives more reduction in switching power, it increases the MAT significantly. The weights corresponding to  $Rp_x$  and  $Rp_y$  are  $exp(2)$  and  $exp(1)$ . Rule  $Rp_x$  has higher weight and hence it is chosen in the process. These weights facilitate a decrease in switching power without much increase in MAT. This also shows the concept of delay-driven power optimisation. While considering power optimisation, the overall delay constraint for each circuit is relaxed step by step ranging from 1% to 10%. The process stops when there is no further reduction in power even after relaxing the delay constraint.

While applying the OPT-T tool, the cost function is  $MAT(f)$  and the control parameter  $T$  is the difference between the initial  $SP(f)$  of the network and the final  $SP(f)$  of the network obtained at each step. This allows us to perform power-driven delay optimisation. The weights for OPT-T are defined in a similar way as defined in the OPT-P method. Since OPT-T is for delay optimisation, the exponential weights are directly proportional to the decrease in delay and inversely proportional to the increase

---

**Algorithm 1** Simulated Annealing in OPT-P

---

**Require:**  $N$ ,  $RN$

**Ensure:**  $R$

```
1: {  $N$  is the total number of nodes in the AIG network }
2: {  $RN$  is the number of rules in OPT-P }
3: for  $i = 1$  to  $N$  do
4:   for  $j = 1$  to  $RN$  do
5:     if  $Node_i$  follows Rule  $R_j$  then
6:       { calculation of Simulated Annealing weights }
7:       { calculation of  $W_p$  }
8:     end if
9:      $W_j = W_p$ 
10:  end for
11:  { Comparisons of  $W_j$  i.e.  $W_{p1}$ ,  $W_{p2}$ ,  $W_{p3}$  and  $W_{p4}$  }
12:   $W_{max} = \max(W_{p1}, W_{p2}, W_{p3}, W_{p4})$ 
13:  if  $W_{max} == W_{p1}$  then
14:     $R = Rp_1$ 
15:  else if  $W_{max} == W_{p2}$  then
16:     $R = Rp_2$ 
17:  else if  $W_{max} == W_{p3}$  then
18:     $R = Rp_3$ 
19:  else if  $W_{max} == W_{p4}$  then
20:     $R = Rp_4$ 
21:  end if
22:  {  $R$  is the rule chosen at that step, for OPT-P application }
23:  { Implement OPT-P on Node  $Node_i$  using Rule  $R$  }
24: end for
```

---

in switching power.  $\Delta p_{OPT-T}$  and  $\Delta t_{OPT-T}$  for the weight calculation are defined as follows:

$$\Delta p_{OPT-T} = SP_f(f) - SP_i(f) \quad (6.3)$$

where  $SP_i(f)$  is the total switching power of the network before the given rule application.  $SP_f(f)$  is the total switching power of the network after the given rule application.

$$\Delta t_{OPT-T} = MAT_i(f) - MAT_f(f) \quad (6.4)$$

where  $MAT_i(f)$  is the MAT of the network before the given rule application.  $MAT_f(f)$  is the MAT of the network after the given rule application. The weights are defined as follows:

If  $\Delta p_{OPT-T} > 0$  and  $\Delta t_{OPT-T} > 0$ , then  $W = e^{\Delta t_{OPT-T}/\Delta p_{OPT-T}}$

If  $\Delta p_{OPT-T} > 0$  and  $\Delta t_{OPT-T} < 0$ , then  $W = e^{\Delta t_{OPT-T}*(-1*\Delta p_{OPT-T})}$

If  $\Delta p_{OPT-T} > 0$  and  $\Delta t_{OPT-T} == 0$ , then  $W = e^{\Delta t_{OPT-T}}$

If  $\Delta p_{OPT-T} < 0$  and  $\Delta t_{OPT-T} > 0$ , then  $W = 1/e^{(-1*\Delta t_{OPT-T})*\Delta p_{OPT-T}}$

If  $\Delta p_{OPT-T} < 0$  and  $\Delta t_{OPT-T} < 0$ , then  $W = 1/e^{(-1*\Delta t_{OPT-T})/\Delta p_{OPT-T}}$

If  $\Delta p_{OPT-T} < 0$  and  $\Delta t_{OPT-T} == 0$ , then  $W = 1/e^{-1*\Delta t_{OPT-T}}$

The stop condition weights are:

If  $\Delta t_{OPT-T} == 0$ , then  $W == 0$

If  $\Delta p_{OPT-T}/SP_i(f) > 0.1$ , then  $W = 0$

Similarly, the Algorithm 2 describes the technique as used in OPT-T.

The stop condition weights indicate two things. Firstly, if there is no change in the  $MAT(f)$ , on the application of that rule, the weight corresponding to that rule is made zero. Secondly, the weight of a given rule is set to zero if the application of that rule exceeds the power limitations. This implies that if there is more than 10% increase in switching power, on the application of that rule, the weight is set to zero. The second stop condition clearly shows the concept of power-driven delay optimisation. In our case, we have set the constraint limits on switching power to be 10%.

Also the weights (in OPT-T) are directly proportional to the decrease in  $MAT(f)$  and inversely proportional to the increase in switching power. This is to obtain a significant reduction in delay without much increase in switching power. While performing delay optimisation, the overall power constraint for each circuit is relaxed step by step ranging

---

**Algorithm 2** Simulated Annealing in OPT-T

---

**Require:**  $N, RN$ **Ensure:**  $R$ 

```
1: {  $N$  is the total number of nodes in the AIG network}
2: {  $RN$  is the number of rules in OPT-T}
3: for  $i = 1$  to  $N$  do
4:   for  $j = 1$  to  $RN$  do
5:     if  $Node_i$  follows Rule  $R_j$  then
6:       {calculation of Simulated Annealing weights}
7:       {calculation of  $W_t$ }
8:     end if
9:      $W_j = W_t$ 
10:  end for
11:  {Comparisons of  $W_j$  i.e.  $Wd_1, Wd_2$  and  $Wd_3$  }
12:   $W_{max} = max(Wd_1, Wd_2, Wd_3)$ 
13:  if  $W_{max} == Wd_1$  then
14:     $R = Rd_1$ 
15:  else if  $W_{max} == Wd_2$  then
16:     $R = Rd_2$ 
17:  else if  $W_{max} == Wd_3$  then
18:     $R = Rd_3$ 
19:  end if
20:  {  $R$  is the rule chosen at that step, for OPT-T application}
21:  { Implement OPT-T on Node  $Node_i$  using Rule  $R$ }
22: end for
```

---



from 1% to 10%. The process stops when there is no further reduction in delay even after relaxing the power constraint.

## 6.5 Switching nodes order for OPT-PT application

Consider an AIG network with  $n$  nodes representing the functionality  $f$  of a combinational circuit. Let us denote the nodes as  $N_1, N_2, \dots, N_n$ . Among these nodes, the reordering rules can be applied on a subset depending upon the conditions mentioned in Section 5.4 and Section 5.5. A term called *switching nodes* is defined. Among the nodes  $N_1, N_2, \dots, N_n$ , the *switching nodes* are those on which any of the reordering rules can be applied (satisfying the conditions for the applicability of the reordering rules) such that  $SP(f)$  or  $MAT(f)$  is getting decreased. Let  $SN$  denote a set of such switching nodes among the total  $n$  nodes and  $SNx$  be an element in  $SN$ . With the calculation of the eligible switching nodes for the set  $SN$ , we introduce a technique which determines a favourable switching nodes order or sequence say  $R$ . This order is the sequence of switching nodes in which OPT-PT has to be implemented. This order gives better results in terms of  $SP(f)$  and  $MAT(f)$  for power and delay optimisation respectively, as compared to the previous ordering techniques discussed in the following section. Uniform Cost Search algorithm [66] is used to obtain a better switching nodes order for OPT-PT implementation.

### 6.5.1 Application of Uniform Cost Search algorithm

Uniform Cost Search (UCS) [66] is a tree search algorithm used for traversing or searching a weighted tree or graph. The search begins at the root node, then continues by visiting the next node which has the least total cost from the root. Typically, the search algorithm involves expanding nodes by adding all unexpanded neighbouring nodes that are connected by directed paths to a priority queue. In the queue, each node is associated with its total path cost from the root, where the least-cost paths are given highest priority. The Uniform Cost Search is complete and optimal. To implement such an algorithm, a graph  $G(V, E)$  is constructed such that:

- the vertices represent the switching nodes chosen from the total  $n$  nodes
- edges represent or link two consecutive vertices (representing switching nodes)

meant for sequential application of OPT-PT. Any two switching nodes which are linked in the Graph  $G(V, E)$  may not necessarily be linked in the AIG network. This linkage only implies that these two switching nodes located globally in the AIG network are used consecutively for OPT-PT application.

- it is a weighted graph with weights (total path cost) associated to each edge.

Suppose there is a weight  $W$  on the edge  $(V_1, V_2)$ . For the weight calculations of  $W$  in OPT-P, certain variables are defined as follows:

$$\Delta p_{OPT-P} = SP_{V_1}(f) - SP_{V_2}(f) \quad (6.5)$$

where  $SP_{V_1}(f)$  is the total switching power of the network, when OPT-P has been applied on all of the nodes up to node  $V_1$  following the calculated order of switching nodes.  $SP_{V_2}(f)$  is the total switching power of the network, when OPT-P has been applied on all of the nodes up to node  $V_1$  following the same calculated order of switching nodes and then on node  $V_2$ .

$$\Delta t_{OPT-P} = MAT_{V_2}(f) - MAT_{V_1}(f) \quad (6.6)$$

$MAT_{V_1}(f)$  is the MAT of the network, when OPT-P has been applied on all of the nodes up to node  $V_1$  following the calculated order of switching nodes.  $MAT_{V_2}(f)$  is the MAT of the network, when OPT-P has been applied on all of the nodes up to node  $V_1$  following the same calculated order of switching nodes and then on node  $V_2$ .

Both,  $\Delta p_{OPT-T}$  and  $\Delta t_{OPT-T}$  can be defined similarly. The weights are defined as in the Simulated Annealing method described in Section 6.4.

In the Simulated Annealing method, weights decide which rule is most beneficial at a particular node of the network. However, in the Uniform Cost Search algorithm, these weights decide a better switching nodes order for OPT-PT application sequentially. Application of OPT-PT in this order gives minimal switching power or minimal delay for power or delay optimisation respectively. For this algorithm application, some matrices are defined corresponding to the weights on the edges of the graph.

- $Orderinit[x]$  is a one dimensional array of order  $1 \times sn$ .  $sn$  is the number of elements in the set  $SN$ . The array represents initial weights  $W_p$  or  $W_t$  of the circuit corresponding to OPT-PT application on every *switching node* mentioned in the array.

- $Orderinit_{SNx}[y]$  is another one-dimensional array of order  $1 \times (sn - 1)$ . The node  $SNx$  where  $SNx \in SN$  is chosen to be the root node among the  $sn$  number of switching nodes. It has the maximum weight in the array  $Orderinit[x]$ . This array represents the weights corresponding to OPT-PT application on the rest of the switching nodes, after the application on node  $SNx$ . Hence these weights correspond to the edges from  $SNx$  to the rest of the switching nodes. The size is reduced by one as the root node has already been chosen from the previous array as  $SNx$ .
- Similarly, another array  $Orderinit_{SNx,SNy}[z]$  of order  $1 \times (sn - 2)$  is defined. This array represents the weights corresponding to OPT-PT application on the rest of the switching nodes, after the application of OPT-PT on nodes  $SNx$  and  $SNy$ . Hence these weights correspond to the edges from  $SNy$  to the rest of the switching nodes. The size is reduced by two as the two nodes have already been chosen from the previous array.

The rest of the matrices used for determining the rest of the order (i.e. after  $SNx$ ,  $SNy$  and  $SNz$ ) are computed accordingly. These matrix elements correspond to the weights on the above created graph  $G(V, E)$  traversing from the root node to the leaf nodes. Hence a particular node is paired with an adjacent node among all the neighbouring nodes depending upon the maximum weight with respect to that edge as mentioned in the adjacency matrix. The path cost with respect to each edge is inversely proportional to the weights calculated, hence the edge with maximum weight is chosen so as to obtain minimal  $SP(f)$  or minimal  $MAT(f)$ .

$R$  is a set to store the ordered *switching node* sequence.  $Q$  is a set to store  $SN - R$ . This set depicts the switching nodes left to be ordered. The root node is obtained by the array  $Orderinit[x]$ . This root node  $SNx$  is connected to all the rest of the nodes with weights corresponding to  $Orderinit_{SNx}[y]$  matrix. Further nodes are expanded by adding the unexpanded neighbouring nodes. The process continues until all the switching nodes are covered or reach a minimal without exceeding the constraint limits of power or delay.

Algorithm 3 describes this technique.

---

**Algorithm 3** Uniform Cost Search in OPT-PT

---

**Require:**  $sn$ **Ensure:**  $R$ 

```
1: { $sn$  is the total number of switching nodes in the AIG network}
2: for  $i = 1$  to  $sn$  do
3:   for  $j = 1$  to  $sn$  do
4:     if  $Node_j \neq SNx$  then
5:        $Orderinit[j] = W_j$ 
6:       {  $W_j$  corresponds to weights when OPT-PT is applied on Node  $Node_j$  }
7:     end if
8:   end for
9:   {  $Max(Orderinit[x])$  is at Node  $SNx$  }
10:  { Apply OPT-PT on Node  $SNx$  }
11:  {  $R$  is the set to store ordered switching node sequence }
12:   $R[i] = SNx$ 
13: end for
```

---

### 6.5.2 Matrices and graphs during OPT-P and OPT-T

To exemplify, the above optimisation algorithm is explained by implementing it on a MCNC Benchmark circuit **x2**. The OPT-P tool is implemented and the associated graph  $G(V, E)$  is shown in Figure 6.5.

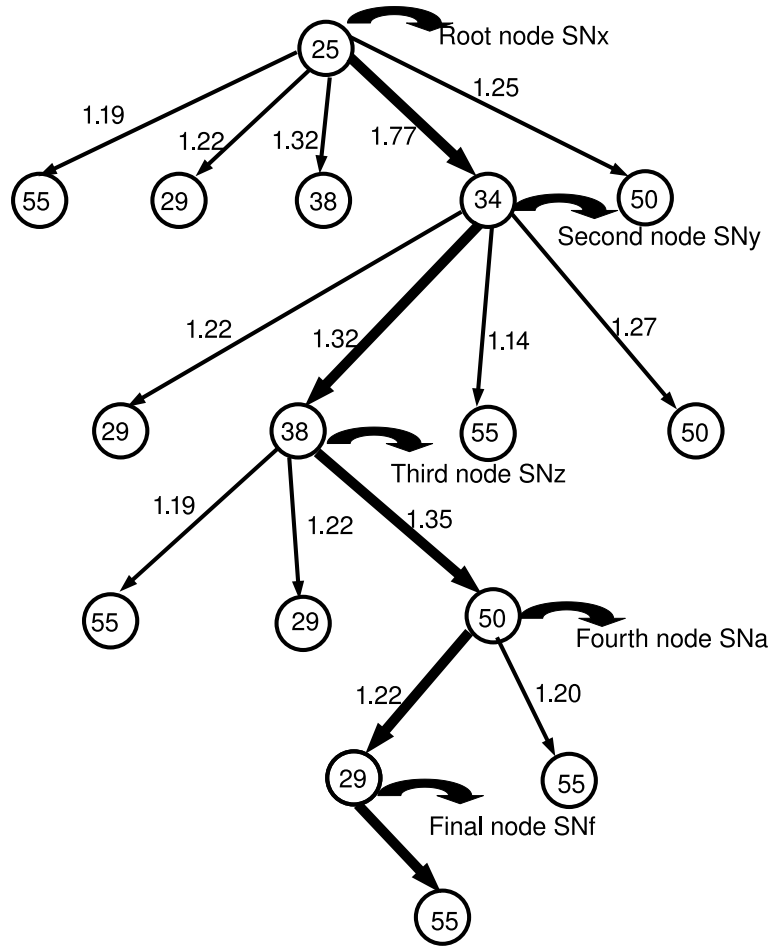


Figure 6.5: The graph  $G(V, E)$  corresponding to the circuit  $x_2$

The nodes shown in Figure 6.5 are the chosen switching nodes and the edges link two switching nodes meant for sequential OPT-PT application. In the graph, the weights on the edges correspond to the matrix values of  $Orderinit_{SN_x}[y]$ ,  $Orderinit_{SN_x, SN_y}[z]$  and so on. In the graph, the node with id ‘25’  $NodeId_{25}$  is chosen as the root node, using  $Orderinit[x]$ . The weights in the matrix  $Orderinit_{SN_x}[y]$  (shown in Table 6.3) are for the edges from  $NodeId_{25}$  to the rest of the remaining switching nodes. The weights are determined as explained in the sub-section above. From the graph  $G(V, E)$ , it is visible that  $NodeId_{25}$ ,  $NodeId_{55}$ ,  $NodeId_{29}$ ,  $NodeId_{38}$ ,  $NodeId_{34}$  and  $NodeId_{50}$  are the switching nodes chosen for the circuit  $x_2$ . In the first step, the root node is chosen which is  $NodeId_{25}$ . In the second step, weights in the matrix  $Orderinit_{SN_x}[y]$  are compared. Hence weights 1.19, 1.22, 1.32, 1.77 and 1.25 with respect to nodes  $NodeId_{55}$ ,  $NodeId_{29}$ ,  $NodeId_{38}$ ,  $NodeId_{34}$  and  $NodeId_{50}$  are compared. The maximum weight of

1.77 with respect to  $NodeId_{34}$  is chosen. This implies the OPT-PT gets applied initially on  $NodeId_{25}$  and then on  $NodeId_{34}$ . In the third step, weights from  $NodeId_{34}$  to the remaining switching nodes namely  $NodeId_{29}$ ,  $NodeId_{38}$ ,  $NodeId_{55}$  and  $NodeId_{50}$  are compared. Similarly, rest of the weights in the following matrices are compared and a sequence or order of nodes is obtained, which when followed gives minimum and optimised power or delay values.

Table 6.3: The Matrix  $Orderinit_{SNx[y]}$

<b>Edge</b>	(25,29)	(25,34)	(25,38)	(25,50)	(25,55)
<b>Weight</b>	1.22	1.77	1.32	1.25	1.19

Similarly the OPT-T tool is implemented on a combinational MCNC Benchmark circuit count and the graph  $G(V, E)$  is shown in Figure 6.6. In this graph, since the weights had very small decimal values, they are multiplied by 1000 for clear visibility.

The global nature of our algorithm is indicated in Figure 6.7. The figure shows an AIG network with inputs  $a, b, c, d, e, f, g, h$  and  $i$  and outputs  $O_1, O_2$  and  $O_3$ . The highlighted nodes are the switching nodes chosen among the total nodes. Arrows indicate the order of switching nodes followed for OPT-PT application, to obtain better results in terms of power and delay.

### 6.5.3 Comparison of Uniform Cost Search (UCS) algorithm with previous methods in OPT-P tool application

Initially, the sequence of OPT-P applications on the AIG nodes proceeded from lower level nodes to higher level nodes of the network. Let this technique be called  $T_1$ . Next, the sequence of OPT-P application on the AIG nodes proceeded from ‘hotter’ nodes to the ‘cooler’ nodes of the AIG network (higher  $SP(f)$  to lower  $SP(f)$ ). Let this technique be called  $T_2$ . Finally, the Uniform Cost Search Algorithm is adopted for the sequence of OPT-P applications on the AIG nodes. Let this technique be called  $T_3$ . In all these three techniques, 10% delay (increase in  $MAT(f)$ ) and 10% area constraint (increase in node count) are allowed. In our case, we have set the constraint limits to be 10% on power, delay and area. Table 6.4, Table 6.5 and Table 6.6 represent the

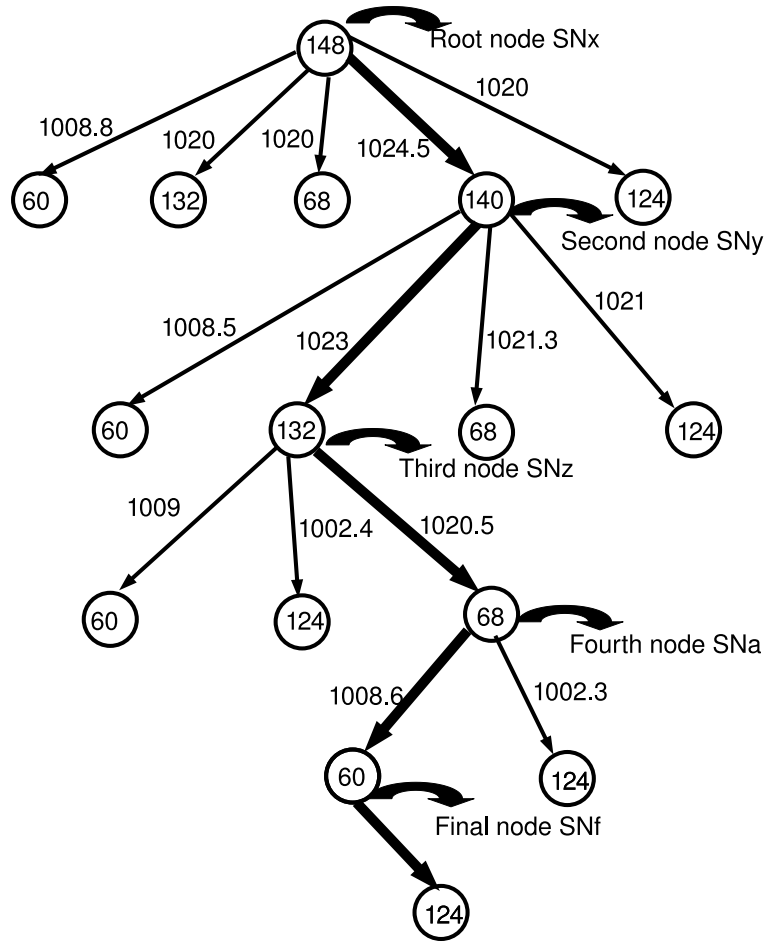


Figure 6.6: The graph  $G(V, E)$  corresponding to the circuit count

switching power reduction on combinational circuits `DES`, `Ri8,o10` and `pair` respectively. The circuits `DES` and `pair` are MCNC benchmark circuits. Circuit `Ri8,o10` actually represents a ROM circuit with 8 inputs and 10 outputs. The best results are obtained when the Uniform Cost Search (UCS) algorithm ( $T_3$ ) is applied. The total switching power  $SP(f)$  is obtained by the product of the switching probability at each node and number of fanouts at that node of the AIG network, as derived in Section 5.3.2.  $N$  stands for node count i.e. number of AIG nodes in the network as built by ABC.  $MAT(f)$  stands for the longest path delay in the AIG network or the maximum arrival time, as derived in Section 5.3.1.  $\Delta SP(f)$  stands for percentage reduction in switching power.

We used the Uniform Cost Search (UCS) algorithm on several Benchmark circuits and in most of the circuits the technique  $T_3$  prove to be better than the previous

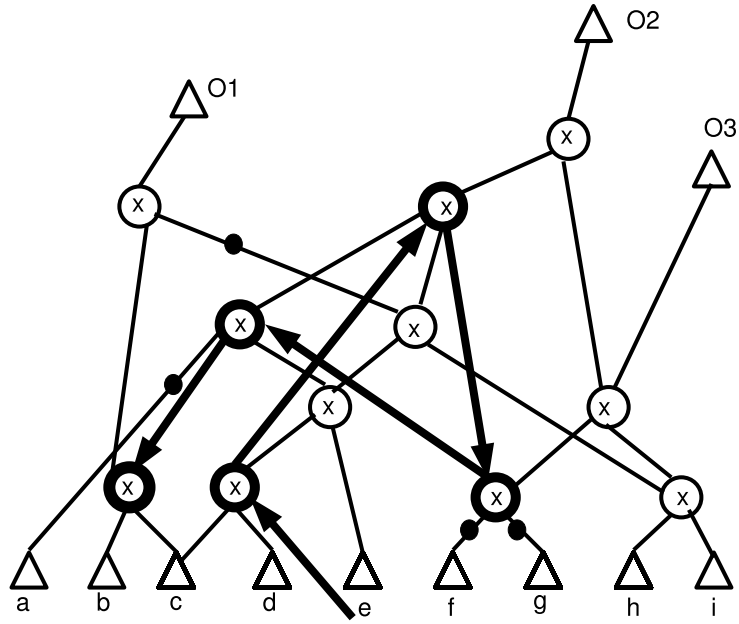


Figure 6.7: The Uniform Cost Search Algorithm

Table 6.4: The comparison of  $T_1$ ,  $T_2$  and  $T_3$  used on DES using OPT-P application

Technique	$SP(f)$	$N$	$MAT(f)$	$\Delta SP(f)\%$
Original	798.80	3612	74.00	-
$T_1$	749.27	3616	85.00	-6.27%
$T_2$	681.25	3615	85.00	-14.71%
$T_3$	594.34	3678	82.00	-25.53%

techniques  $T_1$  and  $T_2$ .

## 6.6 Conclusions

We present an efficient power and delay optimisation tool for digital circuits. This tool is motivated by the fact that power reductions can sometimes increase delay significantly. Similarly, delay reductions can increase power. In order to get a trade-off between the power and delay values, the new tool introduced namely OPT-PT performs a joint optimisation of power and delay. The tool is implemented as a sub-package in the



Table 6.5: The comparison of  $T_1$ ,  $T_2$  and  $T_3$  used on  $R_{i8,o10}$  using OPT-P application

Technique	$SP(f)$	$N$	$MAT(f)$	$\Delta SP(f)\%$
<b>Original</b>	327.16	1107	65.00	-
$T_1$	324.15	1108	72.00	-0.95%
$T_2$	295.13	1105	73.00	-10.87%
$T_3$	281.34	1123	72.00	-17.28%

Table 6.6: The comparison of  $T_1$ ,  $T_2$  and  $T_3$  used on **pair** using OPT-P application

Technique	$SP(f)$	$N$	$MAT(f)$	$\Delta SP(f)\%$
<b>Original</b>	167.60	1292	93.00	-
$T_1$	138.92	1292	104.00	-17.04%
$T_2$	124.26	1292	101.00	-25.88%
$T_3$	100.11	1309	102.00	-40.03%

popular academic synthesis tool ABC.

The tool can be directed to optimise either delay or power. The delay optimisation mode, referred to as OPT-T, reduces critical path length in a combinatorial circuit as much as possible, while honouring a constraint on the switching power. In so-called OPT-P mode, the tool takes the converse approach, reducing switching power as much as possible within a constraint on maximum delay degradation.

In this chapter, we introduced two algorithms which were used within our OPT-PT tool to give reduced and balanced power and delay values. The idea of delay-driven power optimisation and power-driven delay optimisation is introduced using Simulated Annealing method. Further ordering of nodes for OPT-PT application such that it gives reduced power and delay values is brought by the Uniform Cost Search Algorithm. Uniform Cost Search Algorithm searches globally within the AIG network and calculate the order of nodes in which OPT-PT has to be applied such that it gives far better values than previous ordering techniques. In the next chapter, a set of experimental

results (obtained by OPT-PT) is presented on large digital combinational and sequential circuits

## Chapter 7

# Experimental Results using OPT-PT

The tool OPT-PT is implemented on a set of MCNC Benchmark circuits. To complement the MCNC benchmark circuits, a set of large-scale random combinational logic circuits were prepared. These were created using a Verilog case statement filled with random data values implementing a Lookup Table (LUT). The address and the data word lengths were parameterised, allowing the complexity of the resulting LUT logic to be varied as illustrated in Figure 7.1. The random logic lookup tables (ROM circuits) have the names specifying the number of input variables and output lines. For example,  $R_{i6,o8}$  specifies that there are 6 input variables and 8 output lines.

The rest of the chapter is organised as follows. Section 7.1 explains the simulation design flow used to obtain power, delay and area reports on the optimised AIG network by commercial tools. Section 7.2 presents various experimental results of power optimisation using OPT-P application. The results are presented with respect to two technology library files namely TSMC 65nm GP CMOS and TSMC 90nm GP CMOS. Section 7.3 presents various experimental results of delay optimisation using OPT-T application. Some experiments are also performed on few processors and the results are tabulated in Section 7.4. Section 7.5 presents some results interpreted with respect to various input data sets i.e. with varying switching profiles. Finally Section 7.6 makes some concluding remarks.

Inputs = 14, Outputs = 16  
 Address Wordlength =  $2^{14}$   
 Data Wordlength = 16

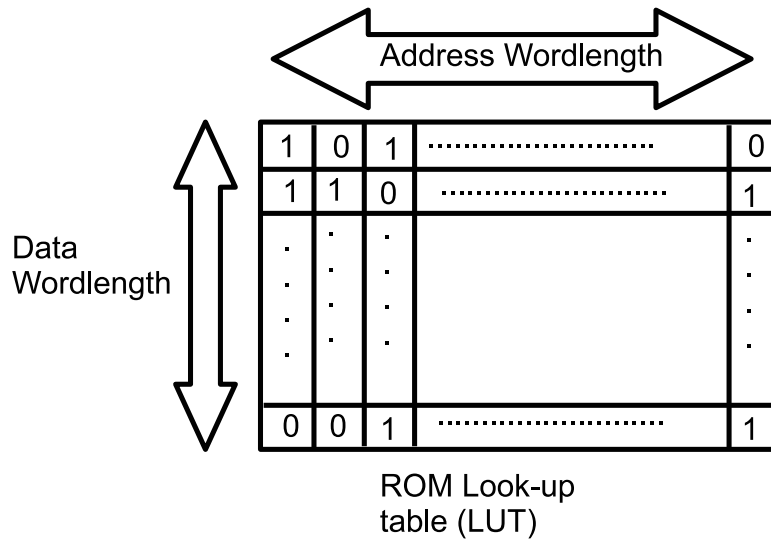


Figure 7.1: The ROM Look-Up Table

## 7.1 Simulation Design Flow

For technology-mapped power and delay results, the flow diagram as shown in Figure 6.4 is followed. The tool used is OPT-PT for power and delay optimisation. There are two sets of networks being compared - the ABC-optimised AIG network and the corresponding OPT-PT-optimised AIG network. The tool AIG2Net [43], as described in Chapter 6 is used for the mapping of the AIG network to a Verilog netlist using 2-input AND, NOT and buffer cells from the TSMC 65GP CMOS standard cell library. The concept behind AIG2Net is referenced in [43]. Two sets of netlists are obtained - the ABC-optimised AIG netlist and the OPT-PT-optimised AIG netlist. At the end of the process, Synopsys Design Compiler (DC) is used for power, delay and area estimation on the two netlists. Also the two networks' netlists are verified for consistency by Synopsys Formality.

All the power and delay estimations and optimisations are performed on an AIG network. Firstly, the AIG network is optimised using the *resyn* script in ABC, which is the most efficient synthesis script available for reducing complexity of the AIG network

with respect to node count. Secondly, the AIG network is optimised using our tool i.e. OPT-PT. The two optimised AIG networks are converted to mapped Verilog netlists (consisting of 2-input ANDs, Inverters and Buffer cells, still representing the same AIG network). Power, delay and area reports are then generated in DC and the results are compared.

## 7.2 Results with OPT-P application

Table 7.1 shows some experimental results based on the application of OPT-P tool. In the table,  $P_{ABC}$ ,  $T_{ABC}$  and  $A_{ABC}$  are the power, delay and area results from the netlist obtained from the ABC-optimised AIG network.  $P_{PO}$ ,  $T_{PO}$  and  $A_{PO}$  are the power, delay and area results from the netlist obtained by the OPT-P-optimised AIG network. In all the tables below, power values are in  $\mu W$  and delay values are in  $ns$ . The power values shown in the table are the total power values i.e. sum of static and dynamic power. In the table, a negative sign indicates a decrease and positive indicates an increase. The average power decrease is 23.46% with slight increase in delay and area. The power reduction ranges from 14% to 33%.

Table 7.2 shows the computation time required to run the synthesis scripts within ABC, on the same circuits mentioned in Table 7.1. *Gates* represent the number of standard cells, that is the number of 2-Input AND, Inverter and Buffer cells used in the mapping of the network.  $RT_{ABC}$  is the run time (given in seconds) resulting from the synthesis script *resyn* used in ABC to optimise the AIG network.  $RT_{PO}$  is the run time (given in seconds) resulting from the optimisation script used in OPT-P to optimise the AIG network. Observing the run time, it can be stated that as the number of nodes are doubled, the run time doubles.

In Figure 7.2, a graph for the percentage power reduction among various circuits on the application of OPT-P is shown.  $R_{8,10}$  implies ROM circuit with 8 inputs and 10 outputs.

Considering circuit  $R_{i8,o10}$  from the Table 7.1, the total power reduction of 25% is obtained by the four reordering rules namely  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$  used in OPT-P. The percentage usage of each rule in the experiment is 38.82% of  $Rp_1$ , 8.24% of  $Rp_2$ , 7.37% of  $Rp_3$  and 45.56% of  $Rp_4$ .

Table 7.3, presents the static and dynamic power component for the simulation result

Table 7.1: The comparison of OPT-P with the best algorithm in ABC (65nm process)

Circuit	$P_{ABC}$	$T_{ABC}$	$A_{ABC}$	$P_{PO}$	$T_{PO}$	$A_{PO}$	$\Delta P$	$\Delta T$	$\Delta A$
-	( $\mu W$ )	( $ns$ )	( $um^2$ )	( $\mu W$ )	( $ns$ )	( $um^2$ )	-	-	-
<b>dalu</b>	160.76	1.82	3464.64	107.42	1.82	3620.28	-33.20%	0%	4.51%
<b>frg2</b>	318.55	0.78	2279.88	243.35	0.79	2358.08	-21.72%	1.33%	3.44%
<b>vda</b>	123.20	0.49	1566.00	84.92	0.51	1560.00	-30.95%	4.16%	0%
<b>x3</b>	329.87	1.21	1999.80	249.58	1.24	2084.28	-22.97%	2.58%	4.79%
<b>des</b>	1460.50	1.95	10674.36	1025.33	2.04	11056.68	- 29.80%	4.61%	3.62%
<b>i8</b>	302.70	1.17	3261.60	256.28	1.20	3358.84	-15.13%	2.64%	3.05%
<b>i9</b>	84.59	1.14	1707.48	56.67	1.19	1735.56	-33.36%	4.47%	1.68%
<b>R<sub>i8,o10</sub></b>	512.50	0.86	3260.52	384.36	0.90	3368.50	-25.09%	4.61%	3.32%
<b>R<sub>i7,o9</sub></b>	306.44	0.62	1557.36	222.19	0.65	1623.24	-27.43%	4.84%	4.25%
<b>R<sub>i10,o12</sub></b>	1868.90	1.42	14509.80	1516.33	1.48	14509.80	-19.46%	4.27%	0
<b>R<sub>i11,o13</sub></b>	3818.09	2.36	30217.00	3226.44	2.39	30821.30	-15.58%	1.39%	2.01%
<b>R<sub>i10,o10</sub></b>	1899.92	1.57	15872.70	1523.69	1.59	16348.03	-19.84%	1.30%	3.06%
<b>R<sub>i12,o14</sub></b>	6585.28	3.52	66266.40	5649.40	3.54	66928.60	-14.27%	0%	0%
<b>R<sub>i11,o11</sub></b>	3646.70	2.20	31555.44	2905.26	2.17	32280.70	-20.38%	-1.49%	2.30%
<b>Average</b>	-	-	-	-	-	-	-23.51%	2.42%	2.63%

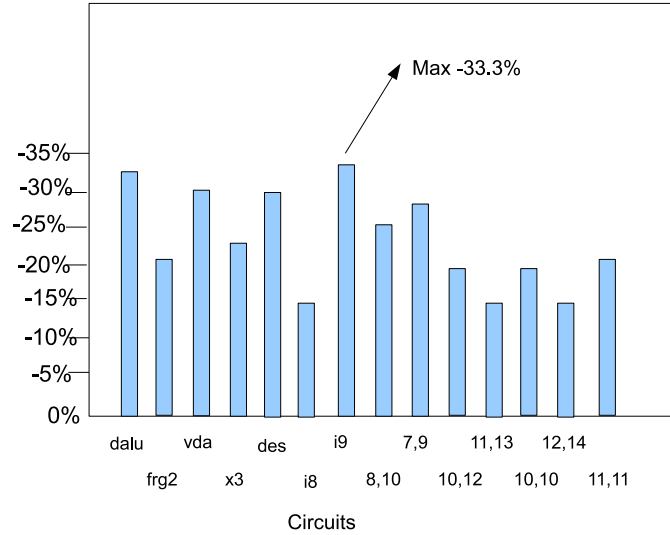


Figure 7.2: The graph for power reduction among various circuits

Table 7.2: Computation time of OPT-P with respect to *resyn* script of ABC

Circuit	Gates	$RT_{ABC}$	$RT_{PO}$
-	-	(s)	(s)
<b>dalu</b>	2097	0.50	0.69
<b>frg2</b>	1358	0.45	0.56
<b>vda</b>	850	0.32	0.46
<b>x3</b>	1180	0.33	0.47
<b>des</b>	6233	1.68	2.85
<b>i8</b>	1900	0.76	1.05
<b>i9</b>	1025	0.31	0.45
<b>R<sub>i8,o10</sub></b>	1911	0.51	0.66
<b>R<sub>i7,o9</sub></b>	918	0.21	0.31
<b>R<sub>i10,o12</sub></b>	8533	2.19	3.02
<b>R<sub>i11,o13</sub></b>	17864	4.38	8.38
<b>R<sub>i10,o10</sub></b>	9379	2.94	4.47
<b>R<sub>i12,o14</sub></b>	36270	8.61	15.95
<b>R<sub>i11,o11</sub></b>	18704	4.39	8.19

stated in Table 7.1. In the tables,  $DP$  represents dynamic power,  $LP$  represents leakage power and  $TP$  represents the total power.

Table 7.4 shows similar power, delay and area results as shown in Table 7.1. All the notations and variables are same as in Table 7.1, except here TSMC 90GP CMOS standard library cells of 2-Input ANDs, Inverters and Buffers are used. The leakage power in both technologies increased due to the addition of extra nodes. However in most of the examples, the leakage power increase ranges from 1% to 4%. Table 7.5 presents the static, dynamic and total power for the circuits stated in Table 7.4.

### 7.3 Results with OPT-T application

Table 7.6 shows some experimental results based on the application of the OPT-T tool.  $P_{TO}$ ,  $T_{TO}$  and  $A_{TO}$  are the power, delay and area results from the netlist obtained by the OPT-T-optimised AIG network. The average delay (critical path length) decrease is 15.53% with slight increase in power and area. The technology used is the TSMC

Table 7.3: Dynamic and Static Power for the simulations stated in Table 7.1 (65nm process)

Circuit	Gates	$DP_{ABC}$	$LP_{ABC}$	$TP_{ABC}$	$DP_{PO}$	$LP_{PO}$	$TP_{PO}$	$\Delta DP$	$\Delta LP$
-	-	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	-	-
<b>dalu</b>	2097	149.01	11.75	160.76	95.20	12.22	107.42	-36.11%	4.00%
<b>frg2</b>	1358	310.81	7.74	318.55	235.38	7.97	243.35	-24.26%	2.97%
<b>vda</b>	805	117.79	5.41	123.20	79.60	5.32	84.92	-32.42%	-1.66%
<b>x3</b>	1180	323.02	6.85	329.87	242.60	7.05	249.58	-24.89%	3.00%
<b>des</b>	6233	1423.80	36.25	1460.50	987.84	37.49	1025.33	-30.61%	3.42%
<b>i8</b>	1917	291.79	10.91	302.70	245.10	11.18	256.28	-16.05%	2.47%
<b>i9</b>	1025	78.73	5.86	84.59	50.66	6.01	56.67	-35.65%	2.55%
<b>R<sub>i8,o10</sub></b>	1911	501.53	10.97	512.50	373.01	11.35	384.36	-25.62%	3.46%
<b>R<sub>i7,o9</sub></b>	918	301.21	5.23	306.44	216.74	5.45	222.19	-28.04%	4.20%
<b>R<sub>i10,o12</sub></b>	8533	1820.30	48.63	1868.93	1467.70	48.63	1516.33	-19.37%	0
<b>R<sub>i11,o13</sub></b>	17864	3716.80	101.29	3818.09	3122.11	104.33	3226.44	-16.05%	3.06%
<b>R<sub>i10,o10</sub></b>	9379	1846.70	53.21	1899.91	1469.69	54.54	1523.70	-20.41%	2.49%
<b>R<sub>i12,o14</sub></b>	36270	6373.80	211.48	6585.28	5436.85	212.53	5651.50	-14.70%	0%
<b>R<sub>i11,o11</sub></b>	18704	3541.10	105.74	3646.84	2797.46	107.85	2905.31	-20.99%	1.99%

65GP CMOS standard library cells of 2-Input ANDs, Inverters and Buffers. The delay reduction ranges from 9% to 32%.

Table 7.7 shows the computation time required to run the synthesis scripts within ABC, on the same circuits mentioned in Table 7.6.  $RT_{ABC}$  is the run time (given in seconds) resulting from the synthesis script *resyn* used in ABC to optimise the AIG network.  $RT_{TO}$  is the run time (given in seconds) resulting from the optimisation script used in OPT-T to optimise the AIG network.  $\Delta RT$  is the change in the run time.

Considering the circuit count from Table 7.6, the delay reduction of 31% is obtained



Table 7.4: The comparison of OPT-P with the best algorithm in ABC (90nm process)

Circuit	$P_{ABC}$	$T_{ABC}$	$A_{ABC}$	$P_{PO}$	$T_{PO}$	$A_{PO}$	$\Delta P$	$\Delta T$	$\Delta A$
-	( $\mu W$ )	( $ns$ )	( $um^2$ )	( $\mu W$ )	( $ns$ )	( $um^2$ )	-	-	-
<b>dalu</b>	292	3.05	6006.77	202.41	3.02	6270.07	-30.50%	0%	4.40%
<b>frg2</b>	577.35	1.36	3953.47	422.08	1.38	4119.02	-26.98%	1.59%	4.20%
<b>vda</b>	220.63	0.82	2615.65	161.05	0.85	2607.19	-27.01%	3.62%	0%
<b>x3</b>	606.73	2.08	3467.31	467.18	2.11	3588.34	-23.03%	1.44%	3.50%
<b>des</b>	2707.87	3.92	18373.05	1949.66	4.06	19050.06	- 28.07%	3.60%	3.90%
<b>i8</b>	570.25	2.15	5614.45	467.60	2.22	5754.35	-18.08%	3.29%	2.50%
<b>i9</b>	150.02	2.08	2954.34	103.51	2.17	3001.62	-31.01%	4.32%	1.63%
<b>R<sub>i8,o10</sub></b>	958.21	1.43	5609.52	721.85	1.48	5790.85	-24.74%	3.55%	3.26%
<b>R<sub>i7,o9</sub></b>	564.59	1.02	2683.39	411.65	1.06	2796.29	-27.17%	3.98%	4.20%
<b>R<sub>i10,o12</sub></b>	3709.60	2.45	24981.00	2938.03	2.54	25105.00	-20.80%	3.71%	0%
<b>R<sub>i11,o13</sub></b>	7654.82	52089.02	4.38	6441.89	4.21	53130.03	-15.94%	1.45%	2.06%
<b>R<sub>i10,o10</sub></b>	3722.40	2.75	27358.06	2986.93	2.79	28178.00	-19.87%	1.40%	3.08%
<b>R<sub>i12,o14</sub></b>	13395.90	6.22	114595.00	11281.09	6.22	115167.00	-15.80%	0%	0%
<b>R<sub>i11,o11</sub></b>	7245.63	3.88	54429.88	5670.82	3.86	55790.62	-21.71%	0%	2.52%

by the three reordering rules namely  $Rd_1$ ,  $Rd_2$  and  $Rd_3$  used in OPT-T. The percentage of usage for each rule in the experiment is 32.36% for  $Rd_1$ , 11.99% for  $Rd_2$  and 55.65% for  $Rd_3$ . Similarly, consider another circuit **frg1** from Table 7.6, the delay reduction of 12% is obtained by the three reordering rules namely  $Rd_1$ ,  $Rd_2$  and  $Rd_3$  used in OPT-T. The percentage of usage for each rule in the experiment is 79.75% for  $Rd_1$ , 8.54% for  $Rd_2$  and 11.71% for  $Rd_3$ . In Table 7.6, in some of the cases, area increases marginally. This means there is a small node count increase during the application of OPT-T, implying little usage of  $Rd_2$  and  $Rd_3$ . However, the delay reduction obtained from the usage of these rules even once is high and hence the rules cannot be eliminated. In Figure 7.3, a graph for the percentage delay reduction among various circuits on the application of OPT-T is shown.

Table 7.5: Dynamic and Static Power for the circuits stated in Table 7.4 (90nm process)

<b>Circuit</b>	<b>Gates</b>	$DP_{ABC}$	$LP_{ABC}$	$TP_{ABC}$	$DP_{PO}$	$LP_{PO}$	$TP_{PO}$	$\Delta DP$	$\Delta LP$
-	-	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	-	-
<b>dalu</b>	2097	282.81	9.19	292.00	192.86	9.55	202.41	-31.80%	3.91%
<b>frg2</b>	1358	571.73	5.75	577.35	416.16	5.92	422.08	-27.21%	2.95%
<b>vda</b>	805	216.84	3.79	220.63	157.33	3.72	161.05	-27.44%	-1.84%
<b>x3</b>	1180	601.75	4.98	606.73	461.91	5.07	467.18	-23.23%	1.80%
<b>des</b>	6233	2681.40	26.47	2707.87	1922.00	27.66	1949.66	-28.33%	4.51%
<b>i8</b>	1917	562.03	8.22	570.25	459.18	8.42	467.60	-18.29%	2.43%
<b>i9</b>	1025	145.18	4.84	150.02	98.56	4.95	103.51	-32.11%	2.27%
<b>R<sub>i8,o10</sub></b>	1911	950.18	8.03	958.21	713.51	8.35	384.36	-24.90%	3.98%
<b>R<sub>i7,o9</sub></b>	918	560.79	3.80	564.59	407.72	3.93	411.65	-27.30%	3.42%
<b>R<sub>i10,o12</sub></b>	8533	3672.90	36.16	3709.60	2901.59	36.44	2938.03	-21.02%	0%
<b>R<sub>i11,o13</sub></b>	17864	7578.70	76.12	7654.82	6441.89	77.64	6519.53	-15.03%	1.99%
<b>R<sub>i10,o10</sub></b>	9379	3682.50	39.94	3722.40	2946.00	40.93	2986.93	-20.04%	2.47%
<b>R<sub>i12,o14</sub></b>	36270	13228.44	167.46	13395.93	11111.88	169.13	11281.05	-16.06%	0%
<b>R<sub>i11,o11</sub></b>	18704	7165.50	80.13	7245.63	5589.09	81.73	5670.82	-22.07%	2.08%

## 7.4 Results after technology mapping by Synopsys Design Compiler

The netlists considered in all of the above simulation results are composed of standard library cells of 2-Input ANDs, Inverters and Buffers, hence still representing the exact AIG networks. Another set of experiments were also performed. In the first experiment, one ABC-optimised and one OPT-PT-optimised netlist were written out of ABC as Verilog files in Generic Technology (GTECH) (i.e. without being mapped to any technology). These Verilog files and the associated SAIF files are read, synthesised and

Table 7.6: The comparison of OPT-T with ABC

Circuit	$P_{ABC}$	$T_{ABC}$	$A_{ABC}$	$P_{TO}$	$T_{TO}$	$A_{TO}$	$\Delta P$	$\Delta T$	$\Delta A$
-	( $\mu W$ )	( $ns$ )	( $um^2$ )	( $\mu W$ )	( $ns$ )	( $um^2$ )	-	-	-
count	76.85	0.91	414.72	78.68	0.62	436.32	2.40%	-31.91 %	5.22%
frg2	354.51	0.78	2279.88	338.88	0.66	2290.68	-4.43%	-15.44%	0%
x3	307.18	1.21	1999.80	297.56	1.10	2290.68	-3.15%	-9.16%	0%
frg1	53.37	0.64	380.52	54.91	0.56	388.13	2.37%	-12.58%	2.09%
dalu	144.12	1.82	3464.64	150.32	1.63	3468.80	4.30%	-10.41%	0%
toolarge	176.18	1.09	1359.72	179.50	0.92	1376.20	2.02%	-15.63%	1.24%
sct	27.22	0.31	185.04	26.36	0.25	186.70	0%	-19.45%	0%
term1	60.23	0.52	534.24	63.63	0.47	546.40	5.60%	-9.60%	2.30%
i2	184.53	0.42	721.44	184.11	0.34	721.44	0%	-19.06%	0%
k2	136.86	0.67	3150.36	148.58	0.55	3152.52	8.57%	-17.98%	0%
DES	1156.66	1.95	10674.36	1126.40	1.73	15879.24	-2.59%	-11.30%	2.01%
$R_{i10,o10}$	1587.32	1.57	15872.76	1497.70	1.37	15879.24	-5.62%	-12.73%	0%
$R_{i9,o11}$	783.48	1.05	6906.60	781.41	0.87	7010.40	0%	-17.24%	1.55%
Average	-	-	-	-	-	-	-	-15.65%	-

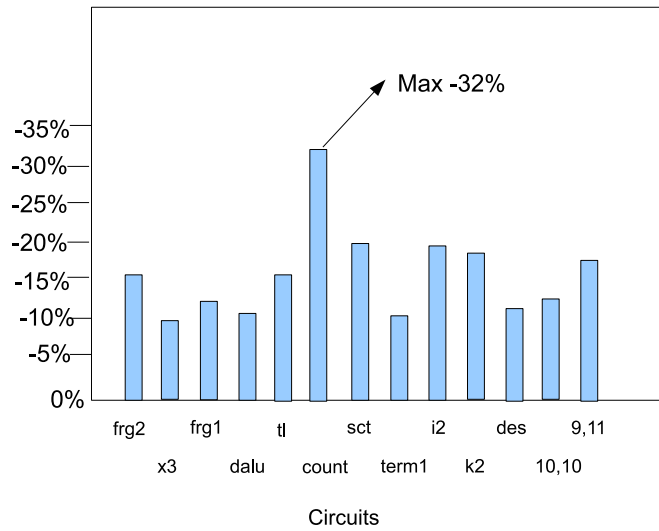


Figure 7.3: The graph for delay reduction among various circuits

Table 7.7: Computation time of OPT-T with respect to *resyn* script of ABC

Circuit	Gates	$RT_{ABC}$	$RT_{TO}$
-	-	(s)	(s)
<b>count</b>	257	0.05	0.09
<b>frg2</b>	1355	0.49	0.88
<b>x3</b>	1180	0.33	0.52
<b>frg1</b>	236	0.07	0.11
<b>dalu</b>	2097	0.46	0.82
<b>toolarge</b>	795	0.21	0.39
<b>sct</b>	114	0.02	0.025
<b>term1</b>	317	0.14	0.19
<b>i2</b>	447	0.12	0.19
<b>k2</b>	1638	0.61	1.1
<b>DES</b>	6233	1.68	3.12
<b>R<sub>i10,o10</sub></b>	9379	2.29	4.15
<b>R<sub>i9,o11</sub></b>	4063	1.05	1.95

compiled in Synopsys DC. Power, delay and area results obtained after actual technology mapping using all the standard cells of TSMC 65GP CMOS library, are reported in Table 7.8. There are two experiments denoted by *o1* and *o2*. In *o1*, the Verilog file from the ABC-optimised AIG network is analysed. In *o2*, the Verilog file from the OPT-P-optimised AIG network is analysed. Table 7.8 shows power, timing and area results.  $P_{o1}$ ,  $T_{o1}$ ,  $A_{o1}$  and  $G_{o1}$  are power, delay, area and gate count obtained by the *o1* experiment.  $P_{o2}$ ,  $T_{o2}$ ,  $A_{o2}$  and  $G_{o2}$  are power, delay, area and gate count obtained by *o2* experiment.

The power reductions using Synopsys technology mapping are marginally better than the power reductions using AIG2Net mapping on OPT-P optimised netlist. The average percentage reduction in power using AIG2Net mapping on OPT-P optimised netlist is 23% and using Synopsys technology mapping is 19%.

## 7.5 Power-Time (P-T) curve

A P-T curve of circuit  $R_{i10,o12}$  with 8500 gates, undergoing OPT-P optimisation is also presented. This is shown in Figure 7.4. The x-axis indicates the delay constraint. For

Table 7.8: Results using Synopsys Design Compiler mapping via OPT-P

<b>Circuit</b>	$P_{o1}$	$T_{o1}$	$A_{o1}$	$P_{o2}$	$T_{o2}$	$A_{o2}$	$\Delta P$
-	( $\mu W$ )	( $ns$ )	( $um^2$ )	( $\mu W$ )	( $ns$ )	( $um^2$ )	-
<b>x3</b>	97.50	0.70	692.64	78.21	0.73	725.04	-19.79%
<b>frg2</b>	137.16	0.71	686.16	112.47	0.74	706.52	-18.03%
<b>i9</b>	30.21	0.74	479.52	23.86	0.71	486.28	-21.04%
<b>dalu</b>	24.80	0.95	679.32	20.50	0.99	713.28	-17.33%
<b>vda</b>	37.97	0.63	551.16	30.07	0.65	559.16	-20.80%
<b>des</b>	428.52	0.85	3213.36	334.98	0.89	3293.48	-21.82%
<b>k2</b>	32.34	0.77	943.56	26.54	0.80	978.84	-17.93%
<b>R<sub>i8,o10</sub></b>	131.47	0.60	1104.84	109.12	0.58	1137.88	-17.04%

example, 3% delay constraint implies that the increase in MAT shall not be greater than 0.03 times the original. On the y-axis, the optimised switching power corresponding to the given delay increase is presented. The figure indicates the maximum reduction in switching power obtained when allowing the specified relaxation in delay constraint. In the above experiments there was no area constraint hence area increased by 6.5%. The power, delay and area values are estimated by Synopsys DC following the flow diagram mentioned in Figure 6.4. The curve shown in Figure 7.4 indicates delay-driven power optimisation using the OPT-P tool.

## 7.6 Experimental results on sequential circuits

We performed another set of experimental results on sequential circuits (with both combinational plus memory (sequential) component). This section presents results on few circuits (used as differential equation solver, elliptic equation solver etc.) having both combinational sub network plus registers and flip flops.

The Figure 7.5 shows a high-level description of a processor with combinational and sequential components. The inputs  $X_1, \dots, X_m$  are the primary inputs and the

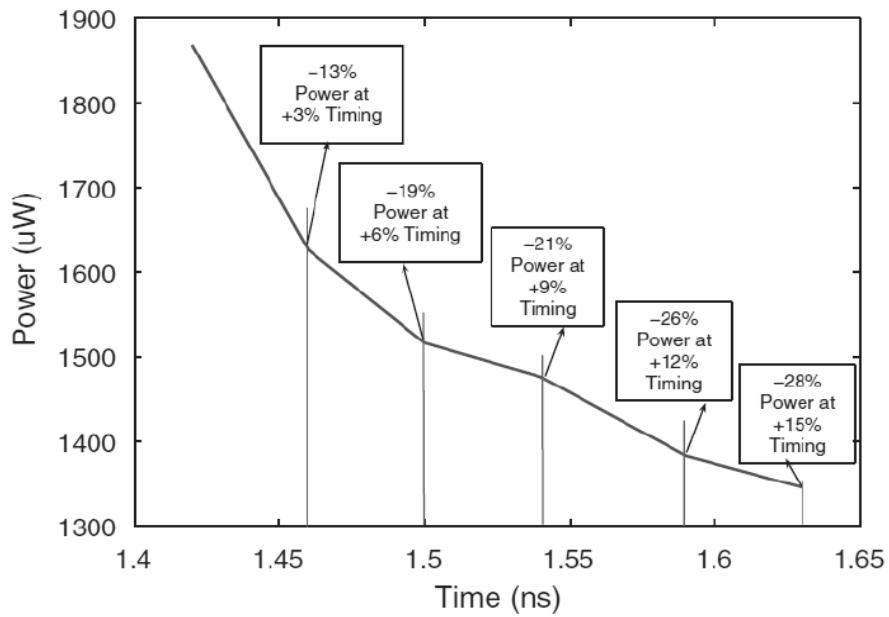


Figure 7.4: The P-T curve on  $R_{i10,o12}$

outputs  $Z_1, \dots, Z_n$  are the primary outputs. The ports  $Q_1, \dots, Q_k$  and  $Q_1^+, \dots, Q_k^+$  are the flip-flop/registers outputs and flip-flop/registers inputs respectively, and are usually called as pseudo primary outputs/inputs. The sequential component of a processor is composed of registers and flip flops represented as storage (Memory). The sub-circuit which links the primary inputs and flip-flop/register outputs with primary outputs and

flip-flop/registers inputs is the combinational sub network of the circuit. Using such module, we can easily identify the combinational part of the circuit.

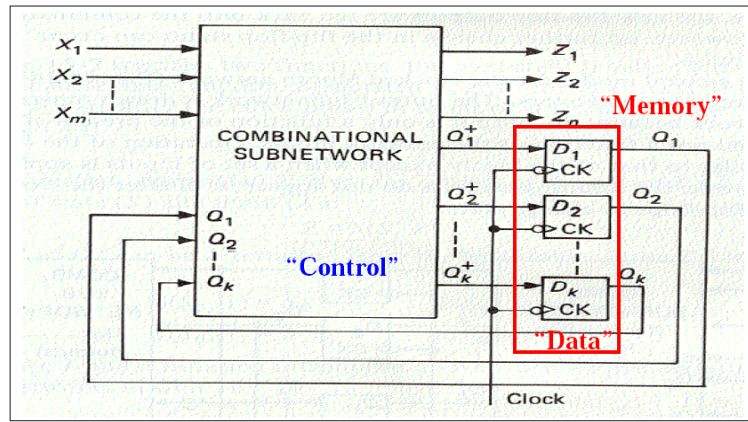


Figure 7.5: High level view of a Processor

A similar concept is performed on the sequential AIG network. A sequential AIG network is shown in Figure 7.6. The flip-flop/register inputs  $v1L - in$ ,  $v2L - in$  and  $v3L - in$  and flip-flop/register outputs  $v1L$ ,  $v2L$  and  $v3L$  are removed and replaced with pseudo Primary Inputs (PIs) and Primary Outputs (POs). The network between Pseudo PIs and Pseudo POs represents the combinational component of the circuit and hence OPT-PT rules can be applied on this combinational sub network.

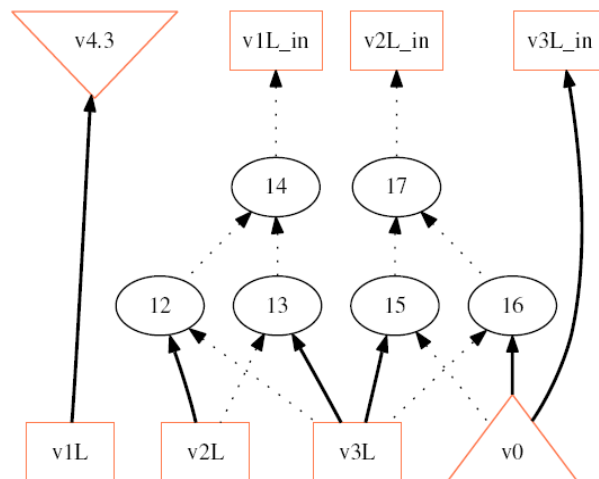


Figure 7.6: Sequential AIG of a 3-bit shift register

After the OPT-PT application, the network is again mapped to a Verilog netlist

using AIG2Net where AND nodes are replaced with AND cells, complemented edges with inverter cells and non-complemented edges with buffer cells chosen from a specific library (TSMC 65nm GP CMOS). The pseudo PIs and POs are reconnected to their sequential component. On mapping, the flip-flop/registers are replaced with D-flip flops chosen from the specified library. A table for the sequential circuit synthesis results is given in Table 7.9. The table presents six sequential circuits followed by their function description, number of input-output ports, number of nodes and edges, number of latches and latch area (non-combinational memory area). These values are tabulated after the technology mapping is done by tool AIG2Net. Table 7.10 presents the power, delay and area results on these circuits and the results are compared with the ABC synthesis scripts. Also note that the results mentioned in Table 7.10 are with respect to the complete circuit i.e. including combinational sub-network and latches. However in Table 7.11, power results are presented while only considering the combinational sub-network of the six sequential circuits considered in the experiment. In Table 7.10 and Table 7.11,  $P_{ABC}$ ,  $T_{ABC}$  and  $A_{ABC}$  are the total power, delay and area results obtained using the ABC synthesis scripts. And  $P_{PO}$ ,  $T_{PO}$  and  $A_{PO}$  are the power, delay and area results obtained using OPT-P script for power optimisation.

Table 7.9: Sequential circuit synthesis results for selected MCNC circuits

Circuit	Description	I/O ports	Cells	Nets	nDFF	Latches Area
-	-	-	-	-	-	( $\mu m^2$ )
<b>dsip</b>	Sequential Encryption	427	4541	4770	224	1532.16
<b>tseng</b>	Bus Controller	175	3964	4016	385	2633.40
<b>diffeq</b>	Differential Equation Solver	104	4743	4807	377	2578.68
<b>bigkey</b>	Sequential Key Encryption	461	5779	6008	224	1532.16
<b>elliptic</b>	Elliptic Equation Solver	246	11260	11391	1122	7674.00
<b>frisc</b>	-	137	11137	11157	886	6060.01

From the results we can infer that the average power reduction of 20% is obtained when only considering the combinational sub network of the circuit, as shown in Table 7.11. However a significant reduction in the average power of 14% is obtained even when considering the complete circuit including combinational plus sequential components as shown in Table 7.10.



Table 7.10: The comparison of OPT-P with ABC, (65nm process)

<b>Circuit</b>	$P_{ABC}$	$T_{ABC}$	$A_{ABC}$	$P_{PO}$	$T_{PO}$	$A_{PO}$	$\Delta P$	$\Delta T$	$\Delta A$
-	( $\mu W$ )	( $ns$ )	( $um^2$ )	( $\mu W$ )	( $ns$ )	( $um^2$ )	-	-	-
<b>dsip</b>	823.57	1.27	8979.84	716.00	1.32	9338.16	-13.01%	3.92%	4.03%
<b>tseng</b>	938	0.19	8481.96	806.04	0.19	8777.80	-14.05%	0%	3.56%
<b>diffeq</b>	1223	0.19	9770.06	1002.07	0.19	10209.08	-18.09%	0%	4.50%
<b>bigkey</b>	1499	1.23	10926.01	1319.02	1.25	11199.03	-12.04%	2.05%	2.50%
<b>elliptic</b>	2721	0.19	24468.06	2312.07	0.19	25202.00	-15.08%	1.50%	3.09%
<b>frisc</b>	1350	0.18	23308.00	1200.01	0.18	23890.02	- 11.03%	2.54%	2.55%
<b>Average</b>	-	-	-	-	-	-	-13.86%	-	-

Table 7.11: Power reduction on the combinational sub-network for a set of sequential circuits

<b>Circuit</b>	$P_{ABC}$	$P_{PO}$	$\Delta P$
-	( $\mu W$ )	( $\mu W$ )	-
<b>dsip</b>	367.00	289.02	-21.04%
<b>tseng</b>	621.06	496.08	-20.10%
<b>diffeq</b>	732.11	563.13	-23.15%
<b>bigkey</b>	1227.17	1018.19	-17.20%
<b>elliptic</b>	1805.22	1425.24	- 21.26%
<b>frisc</b>	760.28	615.30	- 19.02%
<b>Average</b>	-	-	-20.04%

## 7.7 The impact on switching power corresponding to various input data sets

In all of the above experimental results, for the estimation of switching power at each node, we assumed random input probabilities of the primary inputs. Next we observe the impact on the switching power corresponding to various input probability data sets. Various switching activity profiles (*SAP*) have been considered with respect to the primary inputs. Inputs with flat *SAP* denoted as *SAP1*, inputs with  $\log_2$  decreasing *SAP* denoted as *SAP2* and inputs with random *SAP* denoted as *SAP3* are considered in our work. Figure 7.7, Figure 7.8 and Figure 7.9 represent examples of *SAP1*, *SAP2*

and *SAP3* respectively.

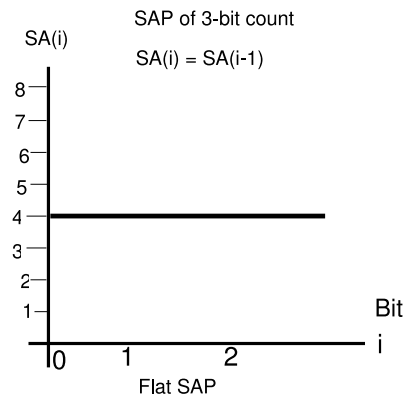
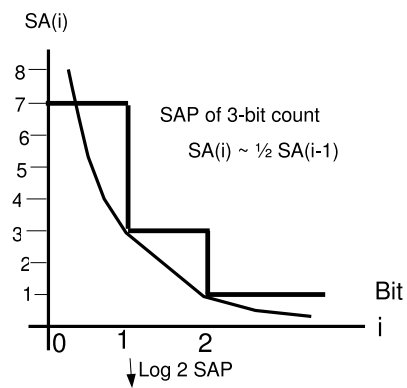


Figure 7.7: Flat *SAP*



SAP looks linear on a logarithmic scale,  
hence referred as "log<sub>2</sub>" SAP

Figure 7.8: Decreasing  $Log_2$  *SAP*

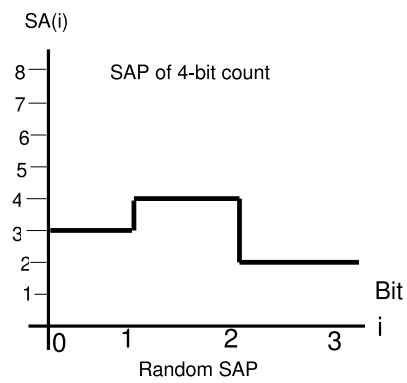


Figure 7.9: Random *SAP*

The experiment corresponding to different *SAPs* is performed on a  $6 \times 4$  bit multiplier. The two inputs of the multiplier are  $A$  and  $B$  with  $A$  of 6 bits and  $B$  of 4 bits. Hence the primary inputs of the multiplier are  $A(0)$ ,  $A(1)$ ,  $A(2)$ ,  $A(3)$ ,  $A(4)$ ,  $A(5)$  and  $B(0)$ ,  $B(1)$ ,  $B(2)$ ,  $B(3)$ . Flat *SAP* implies all the ten primary inputs of the multiplier circuit have same switching activity as implied in Figure 7.7.

$\log_2$  *SAP* implies that the switching activity of one input is half with respect to another input, as implied in Figure 7.8. Hence the switching activity decreases by half from  $A(0)$  to  $A(5)$  and from  $B(0)$  to  $B(3)$ .

Random *SAP* implies that the switching activities of primary inputs from  $A(0)$  to  $A(5)$  and from  $B(0)$  to  $B(3)$  is randomly chosen as implied in Figure 7.9. For the random *SAP* profile (*SAP3*), another set of experiments are performed. The randomly generated switching activities corresponding to the primary inputs  $A(0)$ ,  $A(1)$ ,  $A(2)$ ,  $A(3)$ ,  $A(4)$ ,  $A(5)$ ,  $B(0)$ ,  $B(1)$ ,  $B(2)$  and  $B(3)$  of the multiplier circuit, are stored in a set  $R$ . The switching activity of any one of the inputs is altered, while the rest of the switching activities are unchanged. We did this altering on each of the inputs namely and reported the change in switching power corresponding to each altering. Table 7.12 shows the switching power reduction on the  $6 \times 4$  bit multiplier under the various switching activity profiles.

In the Table 7.12,  $SAP_{1_{0.48}}$  implies flat *SAP* with switching activity set at 0.48.  $SAP_{2_{0.48}}$  implies  $\log_2$  decreasing *SAP* with maximum switching activity set at 0.48.  $SAP_{3_{r_1}}$  and  $SAP_{3_{r_2}}$  are two random *SAPs*.  $SAP_{3_{A(0)}}$  implies random switching activities taken from  $SAP_{3_{r_1}}$  where the switching activity of input  $A(0)$  is altered while rest of the activities are same. For example,  $SAP_{3_{A(1)}}$  implies random switching activities taken from  $SAP_{3_{r_1}}$  where the switching activity of input  $A(1)$  is altered while rest of the activities are same.  $SAP_{3_{A(2)}}$ ,  $SAP_{3_{A(3)}}$ ,  $SAP_{3_{A(4)}}$ ,  $SAP_{3_{A(5)}}$ ,  $SAP_{3_{B(0)}}$ ,  $SAP_{3_{B(1)}}$ ,  $SAP_{3_{B(2)}}$  and  $SAP_{3_{B(3)}}$  are described in a similar fashion.

The results in the Table 7.12 have been tabulated for various switching activity profiles on a multiplier circuit. From the results, the  $\log_2$  decreasing *SAP* given significant reduction of 19%. Also in the random *SAP*, there are instances when the reduction is as low as 9% and instances when the reduction is as high as 23%. In the random *SAP*, the results alter when we change the switching activity of one of the inputs keeping the rest same. Hence altering the switching activity at inputs  $A(1)$ ,  $A(4)$ ,  $A(5)$  and  $B(1)$

Table 7.12: Impact on power reduction under various input data sets (implemented on  $6 \times 4$  bit multiplier)

$SAP$	$P_{ABC}$	$P_{PO}$	$\Delta P$
-	( $\mu W$ )	( $\mu W$ )	-
$SAP1_{0.48}$	523.10	453.12	-13.14%
$SAP2_{0.48}$	431.16	350.18	-19.20%
$SAP3_{r2}$	477.11	400.13	-16.15%
$SAP3_{r1}$	380.17	307.19	-19.01%
$SAP3_{A(0)}$	380.17	338.19	-11.20%
$SAP3_{A(1)}$	380.17	304.21	-20.02%
$SAP3_{A(2)}$	380.17	326.22	-14.04%
$SAP3_{A(3)}$	380.17	323.23	-15.06%
$SAP3_{A(4)}$	380.17	300.24	-21.08%
$SAP3_{A(5)}$	380.17	292.25	-23.11%
$SAP3_{B(0)}$	380.17	345.26	-9.13%
$SAP3_{B(1)}$	380.17	296.27	-22.15%
$SAP3_{B(2)}$	380.17	342.28	-10.17%
$SAP3_{B(3)}$	380.17	329.29	-13.18%

gives higher reduction than the rest ones.

A similar experiment corresponding to different  $SAPs$  is performed on a 5 bit adder. The two inputs of the adder are  $A$  and  $B$  with  $A$  of 5 bits and  $B$  of 5 bits. Hence the primary inputs of the adder are  $A(0)$ ,  $A(1)$ ,  $A(2)$ ,  $A(3)$ ,  $A(4)$  and  $B(0)$ ,  $B(1)$ ,  $B(2)$ ,  $B(3)$ ,  $B(4)$ . Table 7.13 shows the power reduction on the 5 bit adder under the various switching activity profiles. The notations are the same as described above.

The results in the Table 7.13 have been tabulated for various switching activity profiles on an adder circuit. From the results, the  $\log_2$  decreasing  $SAP$  and flat  $SAP$  gives equivalent reduction of 24%. Also in the random  $SAP$ , there are instances when the reduction is as low as 12% and instances when the reduction is as high as 38%. In the random  $SAP$ , the results alter when we change the switching activity of one of the inputs keeping the rest same. Hence altering the switching activity on inputs  $A(0)$ ,  $A(3)$ ,  $A(4)$  and  $B(2)$  gives highest reduction.

Table 7.13: Impact on power reduction under various input data sets (implemented on 5 bit adder )

$SAP$	$P_{ABC}$	$P_{PO}$	$\Delta P$
-	( $\mu W$ )	( $\mu W$ )	-
$SAP1_{0.48}$	178.00	134.04	-24.18%
$SAP2_{0.48}$	141.01	107.05	-24.18%
$SAP3_{r2}$	163.02	120.06	-26.19%
$SAP3_{r1}$	117.03	84.07	-28.20%
$SAP3_{A(0)}$	117.03	82.08	-30.21%
$SAP3_{A(1)}$	117.03	93.09	-20.22%
$SAP3_{A(2)}$	117.03	96.10	-18.23%
$SAP3_{A(3)}$	117.03	83.11	-29.24%
$SAP3_{A(4)}$	117.03	72.12	-38.25%
$SAP3_{B(0)}$	117.03	84.13	-28.26%
$SAP3_{B(1)}$	117.03	103.14	-12.05%
$SAP3_{B(2)}$	117.03	82.15	-30.10%
$SAP3_{B(3)}$	117.03	97.16	-17.15%
$SAP3_{B(4)}$	117.03	93.17	-20.20%

## 7.8 Conclusions

The OPT-PT tool is applied on a set of large MCNC Benchmark circuits. It is also applied on large, randomly-created ROM circuits with gate counts ranging from 1,000 to 100,000. Across MCNC benchmark circuits and a set of large, randomly-generated combinatorial ROM circuits, OPT-PT achieved promising power and delay optimisation results. In OPT-P mode, the average power reduction is 23.46% with respect to the ABC synthesis (also minimal overhead with respect to delay and area). The power reduction ranged from 14% to 33%. On applying OPT-T for power-driven delay optimisation, the average critical path length is reduced by 15.53% with respect to the ABC synthesis (also with minimal overhead with respect to power and area). The delay reduction ranged from 9% to 32%. The networks before and after reordering are verified by Synopsys Formality.

Another experiment is also performed on digital circuits with combinational plus sequential component. The experiment is performed on processors used as differential

equation solver, elliptic equation solver etc., having both combinational sub network plus flip-flops/registers. The results show an average power reduction of 20% when only considering the combinational sub network of the circuit. However a significant reduction in the average power of 14% is obtained even when considering the complete circuit including combinational plus sequential components.

Finally, a set of experiments are performed to study the impact on power reduction with respect to various switching activity profiles. The power reduction is observed under flat switching activity profile, logarithmic decreasing switching activity profile and random switching activity profile.

In the next chapter, we use several combinatorial optimisation techniques to determine the order of rule application on the AIG nodes to achieve improved results in terms of power and delay optimisation.

## Chapter 8

# Combinatorial optimisation techniques for dynamic power reduction

Power optimisation is a crucial problem in modern circuit design, as discussed in the previous chapters. One way of power optimisation is to reduce the switching activity of circuit nodes by annotating it on AIG graphs, followed by reordering and restructuring rules on the AIG network to alter the switching probability and switching power, as dealt in Chapter 5. In this chapter, several combinatorial optimisation algorithms namely Local Search, Tabu Search, Simulated Annealing Search and Partitioned Random Search, are used and incorporated in our previous tool OPT-P for a systematic power optimisation. A comparison among these optimisation algorithms on the basis of dynamic power reduction and computation time is presented. Some significant results of dynamic power reduction are shown on digital circuits such as multipliers, which are a key part of many circuits.

### 8.1 Introduction

In the previous chapter, the switching activity of the circuit nodes is annotated as switching probabilities onto the nodes of AND-Inverter Graphs (AIGs) using a specific delay model. Various reordering and restructuring rules are applied using OPT-P to alter the switching probabilities on the AIG nodes, to reduce the switching power. The

four reordering rules, namely  $Rp_1$ ,  $Rp_2$ ,  $Rp_3$  and  $Rp_4$  used in OPT-P are described in Section 5.4 of Chapter 5. Each rule can be applied on each node of the AIG network if the node satisfies the condition required for the application of that rule. On the application of these rules for switching power reduction, either the *maximum arrival time*  $MAT(f)$  of the AIG network or the node count of the AIG network may increase.

Consider the AIG network to be a graph with AIG nodes as the vertices. The tool OPT-P is applied on some of the AIG nodes on traversing through the AIG network. The order of the nodes scheduled for OPT-P application is determined by the four optimisation algorithms used in this chapter. The combinatorial optimisation methods used are namely *Local Search* (LS), *Tabu Search* (TS), *Simulated Annealing Search* (SS) and *Partitioned Random Search* (PS). These algorithms are extensively used in Electronic Design Automation (EDA). The algorithms traverses the AIG network, to find the node order (path followed) for sequential OPT-P application, to optimise a cost function namely switching power. A comparison is made among the four optimisation algorithms on the basis of switching power and computation time. The switching power is estimated on some combinational circuits namely multipliers.

## 8.2 Multi-objective power-delay optimisation is NP-Complete

Finding the best possible order of AIG nodes for sequential rule application, which gives maximum reduction in the switching power  $SP(f)$  and minimum increase in delay  $MAT(f)$  (such that the product of power and delay gets minimised) is an NP-Complete problem.

It is well known that the complexity of the VLSI dynamic power optimization [60] and power-delay optimization [52] are NP-Complete. The generalized low power binding problem (i.e. the synthesis of a low power architecture based on a fixed number of resources from a Data Flow Graph) is formulated as an Integer Linear Programming (ILP) problem which is shown to be an NP-complete task to solve, as presented in [22]. Dynamic power minimisation during combinational testing is also formulated as a Traveling Salesman Problem (another well known NP-Complete Problem) as presented in [63].

In order to analyse the complexity of our research problem, we also consider the Trav-



eling Salesman Problem (TSP). The Traveling Salesman Problem can be formulated as follows:

Given an undirected weighted graph  $G(V, E)$  where the cities are the graph's vertices  $V$ , paths between cities are the graph's edges  $E$ , and path's distances are the weights  $W$  associated with each edge, find a path/order including every vertex in the set  $V$  that is a path  $p = (v_1, v_2, v_3, \dots, v_n)$  containing the vertices in  $V$  such that:

$$\forall v_i \in V$$

$$(\sum_{1 \leq i \leq n-1} W(v_i, v_{i+1})) + W(v_n, v_1) \text{ is minimised.}$$

The multi-objective power-delay optimisation problem dealt in our work can be re-framed as follows. Consider the set of AIG nodes on which the reordering rules have to be applied as  $AN$ . Let  $E$  denote the product of  $\Delta P$  (decrease in switching power) and  $\Delta T$  (increase in delay) of the circuit when the optimisation reordering rules are applied consecutively first on  $AN_i$  and then on  $AN_j$  where  $AN_i \in AN$  and  $AN_j \in AN$ . Given  $AN$  and  $E$ , the problem is to find AIG nodes order such that  $\sum_{AN_i, AN_j \in AN} 1/E(AN_i, AN_j)$  is minimised.

The Traveling Salesman Problem can be reduced to our multi-objective power-delay optimisation problem. The set of vertices  $V$  of the Traveling Salesman Problem can be replaced with the set of AIG nodes selected for rule application  $AN$ . The function  $W$  associated with each edge of the graph can be replaced with the the function  $E$  as described above. This reduction implies that a solution to the Traveling Salesman Problem exists if and only if there exists a solution to the multi-objective power-delay optimisation problem. Since TSP is NP-Complete, hence it can be concluded that multi-objective power-delay optimisation problem mentioned in our work is NP-Complete. The problem being NP-Complete, several heuristics methods are used in this chapter to find better node ordering with respect to each other. Work in [28] presents implementation and comparison among various heuristics (Tabu Search, Simulated Annealing Search etc) to solve the Traveling Salesman Problem

The rest of the chapter is organised as follows. Section 8.3, 8.4, 8.5 and 8.6 presents the implementation of Local Search, Tabu Search, Simulated Annealing Search and Partitioned Random Search methods respectively. Section 8.7 presents some experimental results using some multipliers and Section 8.8 presents some concluding remarks.

### 8.3 The Local Search Method

Local Search (LS) is a combinatorial optimisation algorithm [29] which starts with a candidate solution and moves iteratively to a neighbouring solution. Termination of a LS can be based on a time bound, number of iterations or any other parametric constraints set by the user. To proceed with any such search algorithms, a graph  $G(V, E)$  is defined. In reference to our context, the graph has vertices as the AIG nodes (solution space) and edges represent two AIG nodes meant for reordering rules application sequentially. In LS, if two nodes  $x$  first and then  $y$  are chosen, it implies that the reordering rules are applied on node  $x$  first and then on node  $y$ . There are two aims of these optimisation algorithms. Firstly the total switching power  $SP(f)$  of the AIG network gets reduced by finding the correct order node order for sequential rule application. Secondly, the computation time of the solution search gets reduced.

Any AIG node can have several neighbouring (local) nodes in the AIG network, the best neighbouring (local) node will be the node which gives maximum reduction of switching power when the OPT-P tool is applied on that node. The process continues until the percentage of  $MAT(f)$  or the node count of the AIG network exceeds the given limitations set as 10%. In the AIG network, nodes which we have defined local to each other, are shown in Figure 8.1. The highlighted nodes are neighbouring (local) nodes with respect to  $node_i$ .

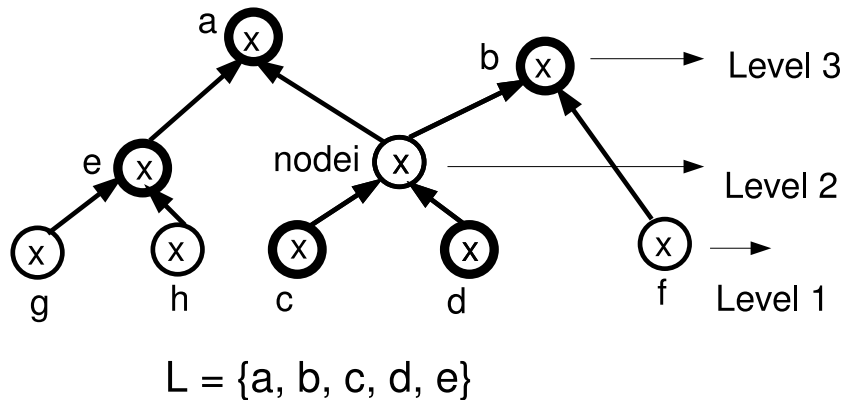


Figure 8.1: Local nodes within an AIG network

Some variables common to all the presented search methods below are declared here.

Let  $pNtk$  be the AIG network considered.  $\#$  denotes the condition for a node to be local with respect to another node. Let the ordered node sequence calculated by the search methods be stored in set  $R$ . Let  $node_i$  be the starting node of any search. Let  $L$  be the set of  $l$  local nodes with respect to Node  $node_i$  satisfying the condition  $\#$ .  $N_i$ ,  $N_f$  and  $\Delta n$  represent the initial, final and fractional node count increase of the AIG network respectively.  $MAT_i(f)$ ,  $MAT_f(f)$  and  $\Delta t$  represent the initial, final and fractional MAT increase after OPT-P application on a chosen node of the AIG network respectively. Let  $SP_i(f)$ ,  $SP_f(f)$  and  $\Delta p$  represent the initial, final and fractional switching power decrease of the AIG network respectively. The pseudo code description of LS is shown in Algorithm 4.

---

**Algorithm 4** Local Search Method

---

**Require:**  $N_i$ ,  $pNtk$ ,  $MAT_i(f)$

**Ensure:**  $R$

- 1:  $MAT_f(f) = MAT_i(f)$  and  $N_f = N_i$
  - 2: { Compute  $\Delta t$  and  $\Delta n$  }
  - 3:  $\Delta t = (MAT_f(f) - MAT_i(f))/MAT_f(f)$
  - 4:  $\Delta n = (N_f - N_i)/N_f$
  - 5: **while**  $\Delta n < 0.10$  **and**  $\Delta t < 0.10$  **do**
  - 6:   **for**  $j = 1$  **to**  $l$  **do**
  - 7:     { Let  $L$  be the set of  $l$  local nodes with respect to Node  $node_i$  satisfying the condition  $\#$  }
  - 8:     { current node is represented as  $node_i$  }
  - 9:     {  $SP_j(f)$  denotes the total switching power of the network when OPT-P is applied on node  $node_j$  after node  $node_i$  }
  - 10:     { Compute Min  $SP_m(f)$  at node  $node_m$  }
  - 11:   **end for**
  - 12:    $R[i] = node_m$  and  $i = i + 1$
  - 13:   { Apply OPT-P on node  $node_m$  }
  - 14:   { Compute  $MAT_f(f)$ ,  $N_f$ ,  $\Delta t$  and  $\Delta n$  }
  - 15:    $MAT_i(f) = MAT_f(f)$  and  $N_i = N_f$
  - 16:    $node_i = R[i]$
  - 17: **end while**
-

## 8.4 The Tabu Search Method

Tabu Search (TS) is another optimisation algorithm [30] which enhances the performance of LS by using memory structures. Once a potential solution has been determined, it is marked as ‘taboo’ so that the algorithm does not re-visit that solution. This method prepares a tabu list, which contains the solutions that have been visited in the recent past (less than  $k$  iterations ago, where  $k$  is the number of previous solutions to be stored). Hence TS excludes the solutions present in the tabu list, when looking for the next neighbouring solution. This is mainly done to overcome the drawback of LS. Sometimes LS gets stuck in a cycle (local minimum) and repeatedly chooses the same set of solutions. This may either lead to very large computation time or a poor/false minimum. The stopping conditions are same as in the LS. Let  $\alpha$  represent the condition that the new node solution is not among the last  $k$  solutions. The pseudo code for TS is shown in Algorithm 5.

## 8.5 The Simulated Annealing Search

Simulated Annealing Search (SS) is an optimisation algorithm [31] which probabilistically determines between moving to another neighbouring state  $s'$  or remaining in state  $s$ . In our SS, the cost function  $SP(f)$  has to be minimised. Instead of probabilities, weights are defined for the movement at each node. These weights depend upon the difference in the cost function before and after rules application, and a control parameter  $T$ . The control parameter  $T$  is the difference between the initial MAT of the network and the final MAT of the network obtained after the application of the rules. The weights chosen are directly proportional to the decrease in switching power and inversely proportional to the increase in MAT. To proceed, the weights are described below.

If  $\Delta p > 0$  and  $\Delta t > 0$ , then  $W = e^{\Delta p/\Delta t}$

If  $\Delta p > 0$  and  $\Delta t < 0$ , then  $W = e^{\Delta p*(-1*\Delta t)}$

If  $\Delta p > 0$  and  $\Delta t = 0$ , then  $W = e^{\Delta p}$

The pseudo code for SS is shown in Algorithm 6.

---

**Algorithm 5** Tabu Search Method

---

**Require:**  $N_i, pNtk, MAT_i(f), k$ **Ensure:**  $R$ 

- 1:  $\{k$  represents the number of last iterative solutions in the tabu list  $\}$
  - 2:  $MAT_f(f) = MAT_i(f)$  and  $N_f = N_i$
  - 3:  $\{$  Compute  $\Delta t$  and  $\Delta n$   $\}$
  - 4:  $\Delta t = (MAT_f(f) - MAT_i(f))/MAT_f(f)$
  - 5:  $\Delta n = (N_f - N_i)/N_f$
  - 6: **while**  $\Delta n < 0.10$  **and**  $\Delta t < 0.10$  **do**
  - 7:   **for**  $j = 1$  **to**  $l$  **do**
  - 8:      $\{$  Let  $L$  be the set of  $l$  local nodes with respect to Node  $node_i$  satisfying the condition  $\#$  $\}$
  - 9:      $\{$   $i$  represents the current number of nodes chosen in the tabu list, current node is represented as  $node_i$  $\}$
  - 10:     **if**  $(i > k$  and  $\alpha == 1)$  or  $i \leq k$  **then**
  - 11:        $\{$  Compute Min  $SP_m(f)$  at node  $node_m$   $\}$
  - 12:     **end if**
  - 13:   **end for**
  - 14:    $R[i] = node_m$  and  $i = i + 1$
  - 15:    $\{$  Apply OPT-P on node  $node_m$  $\}$
  - 16:    $\{$  Compute  $MAT_f(f), N_f, \Delta t$  and  $\Delta n$   $\}$
  - 17:    $MAT_i(f) = MAT_f(f)$  and  $N_i = N_f$
  - 18:    $node_i = R[i]$
  - 19: **end while**
-

---

**Algorithm 6** Simulated Annealing Search

---

**Require:**  $N_i, pNtk, MAT_i(f)$

**Ensure:**  $R$

- 1:  $MAT_f(f) = MAT_i(f)$  and  $N_f = N_i$
  - 2: { Compute  $\Delta t$  and  $\Delta n$  }
  - 3:  $\Delta t = (MAT_f(f) - MAT_i(f))/MAT_f(f)$
  - 4:  $\Delta n = (N_f - N_i)/N_f$
  - 5: **while**  $\Delta n < 0.10$  **and**  $\Delta t < 0.10$  **do**
  - 6:   **for**  $j = 1$  **to**  $l$  **do**
  - 7:     { Let  $L$  be the set of  $l$  local nodes with respect to Node  $node_i$  satisfying the condition  $\#$  }
  - 8:     { current node is represented as  $node_i$  }
  - 9:     { Compute  $W_j$  corresponding to each node  $node_j$  }
  - 10:     { Compute  $maxW_m$  corresponding to node  $node_m$  }
  - 11:   **end for**
  - 12:    $R[i] = node_m$  and  $i = i + 1$
  - 13:   { Apply OPT-P on node  $node_m$  }
  - 14:   { Compute  $MAT_f(f), N_f, \Delta t$  and  $\Delta n$  }
  - 15:    $MAT_i(f) = MAT_f(f)$  and  $N_i = N_f$
  - 16:    $node_i = R[i]$
  - 17: **end while**
-

## 8.6 The Partitioned Random Search

Partitioned Random Search (PS) is an optimisation algorithm [82] with low complexity and low computation time. This method divides the search space into various partitions called sub-sets. It proceeds by looking into all the solutions of the first partition and then proceeds to the next sub-set (partition). Inside the partition, the search is random. In our method, the solution space is the set of AIG nodes of the network  $N$ . The number of subsets is  $ns = 10$ . Each node has a switching power assigned to it, which is the product of the switching probability and number of fanouts (capacitive load) at that node. Let the switching power at node  $n$  be represented by  $SP_n(f)$ . Let the maximum switching power among all the nodes be denoted by  $SP_{max}(f)$ . From the set  $N$ , nodes with switching power greater than or equal to  $0.9.SP_{max}(f)$  fall in the sub-set  $T_9$ . Nodes with switching power greater than or equal to  $0.8.SP_{max}(f)$  and less than  $0.9.SP_{max}(f)$  fall in sub-set  $T_8$ . Similarly, nodes with switching power less than  $0.1.SP_{max}(f)$  fall in sub-set  $T_0$ . Let the number of nodes in set  $T_9$  be  $n_9$ , in set  $T_8$  be  $n_8, \dots$ , in set  $T_0$  be  $n_0$ . If *exit* gets equal to 1, the search process stops there itself. The pseudo code for PS is shown in Algorithm 7.

## 8.7 Experimental Results

Technology mapped power results (after the application of OPT-P tool combined with the combinatorial optimisation algorithms on AIG nodes of the AIG network) are obtained with the same design flow as explained in Section 6.2.1 of Chapter 6. The input file is BLIF file format. The ABC converts this file to an AIG network. This AIG network is optimised using various synthesis scripts and optimisation algorithms. The networks are mapped to a Verilog netlist using AIG2Net [43]. This tool uses TSMC CMOS 65nm GP library cells of 2-input ANDs, Inverters and buffers in order to have almost similar mapping of AIG network to the netlist. A user defined SAIF file is also provided corresponding to the switching probabilities of AIG nodes of the AIG networks (netlist). Power is then estimated using Design Compiler (DC). In our experiments, four sets of AIG networks optimised using the four optimisation methods are considered. The four networks are then mapped to a netlist and power is estimated using DC. The computation time required to run these scripts on combinational circuits

---

**Algorithm 7** Partitioned Random Search Method

---

**Require:**  $N_i, pNtk, MAT_i(f), T_9, T_8, T_7, T_6, T_5, T_4, T_3, T_2, T_1, T_0$

**Ensure:**  $R$

```
1:  $MAT_f(f) = MAT_i(f)$  and  $N_f = N_i$ 
2:  $exit = 0$ 
3: { Compute  $\Delta t$  and  $\Delta n$  }
4:  $\Delta t = (MAT_f(f) - MAT_i(f))/MAT_f(f)$ 
5:  $\Delta n = (N_f - N_i)/N_f$ 
6: for  $j = 1$  to  $n_9$  do
7:    $k = Random(n_9)$ 
8:   { Apply OPT-P on node  $T_9[k]$  }
9:   { Compute  $MAT_f(f), N_f, \Delta t$  and  $\Delta n$  }
10:   $MAT_i(f) = MAT_f(f)$  and  $N_i = N_f$ 
11:  if  $\Delta n > 0.07$  and  $\Delta t > 0.07$  then
12:     $exit = 1$ 
13:  end if
14:  { Search process stops }
15:   $R[i] = T_9[k]$  and  $i = i + 1$ 
16: end for
17: .....
18: { Compute for rest of the subsets  $ns$  }
19: .....
20: for  $j = 1$  to  $n_0$  do
21:    $k = Random(n_0)$ 
22:   { Apply OPT-P on node  $T_0[k]$  }
23:   { Compute  $MAT_f(f), N_f, \Delta t$  and  $\Delta n$  }
24:    $MAT_i(f) = MAT_f(f)$  and  $N_i = N_f$ 
25:   if  $\Delta n > 0.07$  and  $\Delta t > 0.07$  then
26:      $exit = 1$ 
27:   end if
28:   { Search process stops }
29:    $R[i] = T_0[k]$  and  $i = i + 1$ 
30: end for
```

---



is estimated in ABC.

Experiments on some multiplier circuits are performed and three tables of comparison among the four search algorithms are presented on the basis of power, delay and computation time. Table 8.1 and Table 8.2 present power and delay comparisons. Table 8.3 presents the comparison of the computation time required on eight combinational multiplier circuits with respect to the four search algorithms. In the tables,  $m4 - 6$  represent a multiplier circuit with a four bit and six bit input. In the tables,  $P$  represents power values,  $T$  stands for delay values and  $CT$  stands for the computation time. Also  $P_{In}$  and  $T_{In}$  stand for initial power and delay values. Power values are in  $\mu W$ , delay values are in  $ns$  and computation time is given in  $s$ .

Table 8.1: Power comparisons among LS, TS, SS and PS

<b>Circuit</b>	$P_{In}$	$P_{LS}$	$P_{TS}$	$P_{SS}$	$P_{PS}$	$\Delta P_{LS}$	$\Delta P_{TS}$	$\Delta P_{SS}$	$\Delta P_{PS}$
-	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	( $\mu W$ )	-	-	-	-
$m4 - 6$	503.14	410.21	398.33	392.39	392.39	-18.47%	-20.83%	-22.01%	-22.01%
$m4 - 7$	683.49	573.78	576.38	577.68	553.76	-16.05%	-15.67%	-15.48%	-18.98%
$m4 - 8$	908.84	787.14	790.87	781.60	760.78	-13.39%	-12.98%	-14.01%	-16.29%
$m5 - 6$	988.84	861.97	847.23	882.34	831.31	-12.83%	-14.32%	-10.77%	-15.93%
$m4 - 9$	1090.67	963.17	958.15	963.17	936.88	-11.69%	-12.15%	-11.69%	-14.10%
$m5 - 7$	1696.22	1485.21	1503.52	1476.22	1441.44	-12.44%	-11.36%	-12.94%	-15.02%
$m6 - 6$	2105.36	1856.08	1863.06	1823.45	1811.87	-11.84%	-11.51%	-13.39%	-13.94%
$m5 - 8$	2713.61	2396.67	2326.78	2276.89	2265.88	-11.67%	-14.26%	-16.10%	-16.51%
<b>Average</b>	-	-	-	-	-	-13.54%	-14.13%	-14.53%	- 16.59%

In Table 8.3, comparing the computation time of LS and TS, it is observed that the computation time is almost similar for smaller circuits, but the difference increases significantly for larger circuits. This is mainly because large circuits imply large solution searching space. In the LS method with large solution space, there might be many cases when the search is stuck in a local minimum thereby increasing the computation time significantly. In Table 8.3, computation time is  $\infty$  in one the cases of Local Search. This implies that the search is stuck in an infinite loop (stuck at local minimum) and

Table 8.2: Delay comparison among LS, TS, SS and PS

<b>Circuit</b>	$T_{In}$	$T_{LS}$	$T_{TS}$	$T_{SS}$	$T_{PS}$	$\Delta T_{LS}$	$\Delta T_{TS}$	$\Delta T_{SS}$	$\Delta T_{PS}$
-	(ns)	(ns)	(ns)	(ns)	(ns)	-	-	-	-
$m4 - 6$	0.95	1.02	1.03	0.98	1.01	6.36%	7.42%	2.15%	5.31%
$m4 - 7$	1.03	1.10	1.09	1.07	1.08	5.79%	4.82%	2.88%	3.85%
$m4 - 8$	1.19	1.26	1.26	1.24	1.22	4.88%	4.88%	3.20%	1.52%
$m5 - 6$	1.20	1.25	1.24	1.23	1.25	3.16%	2.33%	1.50%	3.16%
$m4 - 9$	1.22	1.34	1.31	1.28	1.26	8.83%	6.37%	4.64%	2.27%
$m5 - 7$	1.49	1.61	1.57	1.55	1.61	7.05%	4.36%	3.02%	7.05%
$m6 - 6$	1.57	1.58	1.58	1.58	1.58	0.63%	0.63%	0.63%	0.63%
$m5 - 8$	1.87	1.91	1.97	1.91	1.98	2.13%	5.34%	2.13%	5.88%

Table 8.3: Comparison of computation time using LS, TS, SS and PS

<b>Circuit</b>	<b>Gates</b>	$CT_{LS}$	$CT_{TS}$	$CT_{SS}$	$CT_{PS}$
-	-	(s)	(s)	(s)	(s)
$m4 - 6$	2451	1.69	1.74	1.94	1.29
$m4 - 7$	3345	1.87	1.73	1.70	1.97
$m4 - 8$	4821	5.44	5.28	6.40	5.18
$m5 - 6$	5049	6.39	6.69	6.71	5.17
$m4 - 9$	6082	6.69	6.92	7.18	6.32
$m5 - 7$	8955	40.86	25.99	28.38	27.44
$m6 - 6$	11263	44.16	30.89	46.92	26.10
$m5 - 8$	15119	$\infty$	118.53	164.97	102.47

hence the computation time tends to  $\infty$ . On the other hand, TS prepares a list of last solutions which prevents it from getting stuck in cycles. Hence in the Table 8.3, in the last three circuits, the computation time for LS increases significantly. In one case, the method gets stuck in an infinite loop. In the smaller circuits, i.e. the first five circuits in Table 8.3, either the computation time for TS is low with respect to LS or if high, then the corresponding power results are better. This can be observed in Table 8.1.

Observing the computation time of SS in Table 8.3, it can be noticed that SS is a slow and gradual process and is based on delay dependent power optimisation. Although the computation time is higher for all the circuits, it has a significant decrease in power with a very slight increase in delay. This can be observed in Table 8.2. In Table 8.1 and Table 8.2, the power reductions due to SS are moderate but with that reduction, there is very little increase in the delay. The delay values in Table 8.2 for SS are least in the majority of the circuits.

The computation time in PS as shown in Table 8.3, is low in majority of the cases. The method has a fixed procedure of partitioning the search space and choosing randomly all solutions in each partition, thereby reducing the computation time significantly. However the criteria of partitioning is very subjective to the problem considered.

In Table 8.1, the maximum reduction in power is obtained in PS. TS and SS have almost similar power reduction values. LS method has the minimum average power reduction among the four search methods as shown in Table 8.1. LS is a basic and convenient method of search and provides good results with less time if the search space is small.

## 8.8 Conclusions

The chapter presents a comparison of four combinatorial optimisation algorithms namely Local Search, Tabu Search, Simulated Annealing Search and Partitioned Random Search. A methodology of how these optimisation algorithms can be incorporated in the OPT-P tool to reduce power on some combinational multiplier circuits is presented. In our experimental results it is shown that choosing the correct optimisation method is a trade-off between computation time, power reduction, overhead factors like delay, and complexity of the method. According to the results obtained, PS provided good results with good reductions in power, low computation time, low overhead with respect to delay. SS provided good power reductions, least overhead with respect to delay but with high computation time. TS is an enhanced version of LS and hence improved the results significantly with respect to LS. Power reduction values range from 13.5% to 16.6% as one shifts from LS to PS.

## Chapter 9

# Conclusions

Low power design is becoming increasingly important in today's technology as wireless communication becomes increasingly desirable. Although power dissipation is important in portable systems, delay issue (performance) is an equally important target for digital designers. Energy consumption is the product of power dissipation and runtime. Tight energy constraints are commonplace in many modern VLSI applications. Consumers expect higher speed, more functionality and higher levels of integration, from their cellular phones and hand-held devices. Low power design shall not be done at the expense of delay (performance) of the circuit. Similarly fast circuit design shall not be done at the expense of increased power dissipation. Hence the motive of the thesis was to explore optimisation schemes for power with minimal overhead of delay, and optimisation schemes for delay with minimal overhead with respect to power.

### 9.1 Contributions

A number of academic EDA synthesis tools proposed in the literature were studied. These open-source tools provide a programming environment and a solid platform for research in logic synthesis, technology mapping, power and delay estimation and optimisation. These academic tools represent the Boolean functionality of any digital circuit using a data structure. Data structure manipulations for logic synthesis, optimisation and technology mapping are done on these data structures. In our approach, AND-Inverter Graph (AIG) was used as the data structure for synthesis and optimisation. The academic EDA synthesis tool called ABC was selected and used for the implemen-

tation of various optimisation tools used in our work. The ABC package builds and manipulates AIGs.

The thesis introduced new methodologies and new synthesis methods for low power digital design. In our approach, the switching power and critical path lengths of digital circuits were estimated on AND-Inverter Graphs both under the zero-delay model and then under a real-delay model. A series of graph reordering rules were applied on these graphs. These rules were based on common rules of Boolean Algebra and worked on the restructuring of the AIG network keeping the functionality the same. This reordering and restructuring of the AIG nodes achieved significant reduction in switching power or delay.

The work also focused on concurrent power and delay optimisation. Delay-driven power optimisation and power-driven delay optimisation methodologies were introduced to have balanced power and delay values. A new tool was introduced called OPT-PT, implemented in C as a sub-package within ABC. OPT-PT has two components - OPT-P for delay-driven power optimisation and OPT-T for power-driven delay optimisation. This was motivated by the fact that optimisation of one of the factors (say switching power) might increase the other factor (like delay). Similarly, reduction of delay might increase the switching power.

To overcome this problem, two optimisation algorithms namely Simulated Annealing and Uniform Cost Search Algorithm were introduced. These algorithms were mainly used in delay-driven power optimisation and power-driven delay optimisation. Several other combinatorial optimisation techniques were also discussed and compared, which were used as a graph search engine, in reference to our work.

For power and delay simulation results, a design flow was introduced which converted the AIG network to an AIG netlist using TSMC GP CMOS library cells of 2-input ANDs, Inverters and Buffers. The comparisons were made with respect to the best synthesis scripts (used in reducing the complexity of the AIG network) of the most popular academic state of the art synthesis tool ABC. The two AIG networks, one optimised by the ABC synthesis scripts and one optimised by the OPT-PT synthesis scripts, were mapped to AIG netlist. Power, delay and area reports on these AIG netlists were generated using Design Compiler. Our tool was implemented on MCNC Benchmark Circuits and on large ROM circuits with gates count ranging from 1,000 to 100,000.

A power reduction of 23.46% is obtained using OPT-PT with respect to the ABC synthesis (also minimal overhead with respect to delay and area). The average critical path length was reduced by 15.53% using OPT-PT with respect to ABC synthesis (also with minimal overhead with respect to power and area). Also, the leakage power increased very marginally ranging from 1% to 4%.

## 9.2 Future Work

While this work introduced some solutions for the existing problems, it also opened the door for new questions and new methodologies for multi-objective optimisation in digital circuits.

NAND-NOR-Inverter Graphs proved to be another promising data structures which could be used for logic synthesis, technology mapping, power and delay estimation and optimisation. A tool can be introduced not only for implementing these data structures but also manipulating them for logic synthesis, technology mapping and optimisation.

In the simulation design flow, the AIG network is mapped to an AIG netlist using only the 2-input ANDs, Inverter and Buffer cells of TSMC GP CMOS library. A technology mapper tool can be developed as a sub-package in ABC which maps the AIG network to a better netlist (using all the cells of TSMC GP CMOS library) which can give further delay and power reduction. The technology mapper shall consider power as the optimisation metric or delay as the optimisation metric while mapping.

# Bibliography

- [1] Y. Aghaghiri, F. Fallah and M. Pedram, “Irredundant Address Bus Encoding for Low Power”, *International Symposium on Low Power Electronics and Design, Huntington Beach, CA*, Aug. 2001, pp. 182–187.
- [2] M. W. Allam, “New Methodologies for Low-Power High-Performance Digital VLSI Design”, *Thesis submitted to Department of Electrical and Computer Engineering, University of Waterloo, Ontario*, 2000.
- [3] P. Balasubramanian and K. Anantha, “Power and delay optimised graph representation for combinational logic circuits”, *International Journal of Computer Science*, Vol. 2, No. 1, 2007, pp. 47–53.
- [4] P. Balasubramanian and D. A. Edwards, “Synthesis of Power and Delay optimized NIG structures”, *Proceedings of 20th IEEE Canadian Conference on Electrical and Computer Engineering*, Apr. 2007, pp. 239–242.
- [5] P. Balasubramanian, R. T. Naayagi, A. Karthik and B. Raghavendra, “Evaluation of Logic Network Representations For Achilles’ Heel Boolean Functions”, *International Journal of Computers, Systems and Signals*, Vol. 9, No. 1, 2008, pp. 14–22.
- [6] P. Balasubramanian, C. H. Narayanan and K. Anantha, “Low Power Design of Digital Combinatorial Circuits with Complementary CMOS Logic”, *International Journal of Electronics, Circuits and Systems*, Vol. 1, No. 1, 2006, pp. 10–18.
- [7] BLIF2VHDL, “Blif to vhdl translator”, <http://tams-www.informatik.uni-hamburg.de/vhdl/tools/blif2vhdl>.
- [8] R. Brayton, “Factoring logic functions”, *IBM Journal of Research and Development*, Vol. 31, No. 2, 1987, pp. 187–198, .

- [9] R. K. Brayton, M. Gao, J. H. R. Jiang, Y. Jiang, Y. Li, A. Mishchenko, S. Sinha and T. Villa, “Optimization of multivalued multi-level networks”, *Proceedings of 32nd IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, May 2002, pp. 168–177.
- [10] R. Brayton and C. McMullen, “The decomposition and factorisation of Boolean expressions”, *Proceedings of IEEE International Symposium on Circuits and Systems*, 1982, pp. 49–54.
- [11] R. Brayton and A. Mishchenko, “ABC: An academic industrial-strength verification tool”, *Proceedings of CAV’10, Springer, LNCS 6174*, 2010, pp. 24–40.
- [12] R. Brummayer and A. Biere, “Local Two-Level AND-Inverter Graph Minimization without Blowup”, *Proceedings of 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, Oct. 2006, pp. 32–38.
- [13] R. Bryant, “Graph-based algorithms for Boolean function manipulation”, *IEEE Transactions on Computers*, Vol. 35, No. 8, 1986, pp. 677–691.
- [14] A. P. Chandrakasan and R. Brodersen, “Minimizing power consumption in digital CMOS circuits”, *Proceedings of the IEEE*, Vol. 83, No. 4, Apr. 1995, pp. 498–523.
- [15] C. Chen and C. Tsui, “Timing Optimization of Logic Network Using Gate Duplication”, *Asia and South Pacific Design Automation Conference 1999 (ASP-DAC’99)*, 1999, pp. 233–236.
- [16] H. Choi and S. H. Hwang, “Power Reduction and Power-Delay Trade-Offs Using Logic Transformations”, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 4, No. 1, Jan. 1999, pp. 97–121.
- [17] H. Choi and S. H. Hwang, “Improving Two-Level Logic Minimization Technique for Low Power Driven Multi-Level Logic Re-Synthesis”, *Proc. 40th Midwest Symposium on Circuits and systems*, 1997, pp. 1026–1029.
- [18] H. Choi and S. H. Hwang, “Low Power Logic Synthesis under a General Delay Model”, *ISLPED 98, Monterey, CA USA*, Aug. 1998, pp. 209–214.



- [19] H. Choi and S. H. Hwang, “Reducing the size of a BDD in the combinational circuit power estimation by using the dynamic size limit”, *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, Vol. 3, No. 1, Jun. 1997, pp. 1520–1523.
- [20] M. Choudhury and K. Mohanram, “Timing-driven optimization using lookahead logic circuits”, *Proceedings of 46th Annual Design Automation Conference, NY*, 2009, pp. 390–395.
- [21] W. Chuang, S. S. Sapatnekar and I. N. Hajj, “Timing and Area Optimization for Standard-Cell VLSI Circuit Design”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 3, 1995, pp. 308–320.
- [22] A. Davoodi and A. Srivastava, “Effective graph theoretic techniques for the generalized low power binding problem”, *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED)*, 2003, pp. 152–157.
- [23] C. Ding, C. Tsui and M. Pedram, “Gate-Level Power Estimation Using Tagged Probabilistic Simulation”, *IEEE Transactions On Computer-Aided Design Of Integrated Circuits and Systems*, Vol. 17, No. 11, Nov. 1998, pp. 1099–1107.
- [24] EDIF2BLIF, “An EDIF to BLIF conversion utility”, <http://www.eecg.toronto.edu/~jayar/software/edif2blif/edif2blif.html>.
- [25] M. Felipe, R. Teresa and T. Yago, “Disjoint Region Partitioning for Probabilistic Switching Activity Estimation at Register Transfer Level”, *PATMOS*, 2008, pp. 399–408.
- [26] M. Felipe, T. Yago and R. Teresa, “A BDD Proposal for Probabilistic Switching Activity Estimation”, *International Conference on Design of Circuits and Integrated Systems (DCIS), Grenoble, France*, Nov. 2008, pp. 54–62.
- [27] M. Felipe, T. Yago and R. Teresa, “Exploiting VHDL-RTL features to reduce the complexity of power estimation in combinational circuits”, *Research in Microelectronics and Electronics*, Jul. 2005, pp. 111–114.

- [28] D. Garg, “Choosing the best heuristic for a NP-Problem”, *International Journal of Information Technology and Knowledge Management*, Vol. 1, No. 2, 2008, pp. 537-547.
- [29] S. H. Gerez, “Local Search”, *Algorithms for VLSI Design Automation*, John Wiley and Sons, pp. 69–71.
- [30] S. H. Gerez, “Tabu Search”, *Algorithms for VLSI Design Automation*, John Wiley and Sons, pp. 73–74.
- [31] S. H. Gerez, “Simulated annealing”, *Algorithms for VLSI Design Automation*, John Wiley and Sons, pp. 71–72.
- [32] Greedy Algorithm, “Approximation and learning by greedy algorithms”, *Annals of Statistics - ANN STATIST*, Vol. 36, No. 1, 2008, pp. 64–94.
- [33] IEEE, “IEEE standard hardware description language based on the Verilog hardware description language”, (*IEEE Std 1364-1995*), 1995.
- [34] S. Iman and M. Pedram, “POSE: Power Optimization and Synthesis Environment”, *33rd Annual Conference on Design Automation*, 1996, pp. 21–26.
- [35] J. H. R. Jiang and S. Devadas, “Logic Synthesis in a Nutshell”, *Chapter 6 of Electronic Design Automation: Synthesis, Verification, and Test*, Elsevier 2009.
- [36] M. Kerttu, “Low Power Synthesis of BDD Mapped Circuits”, *Thesis submitted to Department of Computer Science and Electrical Engineering, Lulea University of Technology, Lulea, Sweden*, 2000.
- [37] S. N. Kumar and J. G. J. Gnannamal, “Delay and Power Optimization of Sequential Circuits through DJP Algorithm”, *Proceedings of the World Congress on Engineering*, Vol. 1, No. 2, 2008, pp. 24–28.
- [38] M. T. P. Lindgren, M. Kerttu and R. Drechsler, “Low power optimisation technique for BDD mapped circuits”, *ASP-DAC*, 2001, pp. 615–621.
- [39] A. C. Ling, J. Zhu and S. D. Brown, “Delay Driven AIG Restructuring using Slack Budget Management”, *ACM/IEEE Great Lakes Symposium on VLSI*, 2008, pp. 163–166.

- [40] P. C. McGeer, J. V. Sanghavi, R. K. Brayton and A. L. Sangiovannivincentelli, “ESPRESSO-SIGNATURE: A new exact minimiser for logic functions”, *IEEE Transactions on VLSI*, Vol. 1, No. 4, 1996, pp. 432–440.
- [41] R. Mehrotra, T. English, K. L. Man, E. Popovici and M. Schellekens, “Digital power estimation flow combining academic and industrial tools”, *IEEE Proceedings of the 5th IEEE International SoC Design Conference*, 2008, pp. 89–92.
- [42] R. Mehrotra, K. L. Man, E. Popovici and M. Schellekens, “Data Structure Manipulation for NNIG and PTNNIG: Towards a Unified Power and Timing Analysis”, *3rd International conference on Signals, Circuits and Systems*, Nov. 2009, pp. 1–6.
- [43] R. Mehrotra, E. Popovici, K. L. Man and M. Schellekens, “Power reduction and technology mapping of digital circuits using AND-Inverter Graphs”, *27th International conference on Microelectronics (MIEL 2010)*, Nis, Serbia, May 2010, pp. 295–298.
- [44] A. Mishchenko and R. Brayton, “Scalable Logic Synthesis using a Simple Circuit Structure”, *Proceedings of IWLS*, 2006, pp. 15–22.
- [45] A. Mishchenko, R. Brayton and S. Jang, “Global delay optimization using structural choices”, *Proceedings of FPGA ’10*, 2010, pp. 181–184.
- [46] A. Mishchenko, R. Brayton, S. Jang, and K. Chung, “A power optimization toolbox for logic synthesis and mapping”, *IEEE International Workshop on Logic Synthesis, San Francisco, CA*, 2009, pp. 1–8.
- [47] A. Mishchenko, R. Brayton, S. Jang and V. Kravets, “Delay Optimization Using SOP Balancing”, *Proceedings of IWLS’11*, 2011, pp. 75–82.
- [48] A. Mishchenko, S. Chatterjee and R. Brayton, “Dag-aware AIG rewriting a fresh look at combinational logic synthesis”, *Proceedings of the 43rd annual conference on Design automation*, 2006, pp. 532–535.
- [49] A. Mishchenko, S. Chatterjee, R. Brayton and P. Pan, “Integrating logic synthesis, technology mapping and retiming”, *Proceedings of IWLS ’05*, 2005, pp. 383–390.

- [50] A. Mishchenko, S. Chatterjee, R. Jiang and R. Brayton, “FRAIGs: A Unifying Representation for Logic Synthesis and Verification”, *ERL Technical Report, EECS Dept., UC Berkeley*, Mar. 2005.
- [51] K. Moiseev and A. Kolodny, “Power-Delay Optimization in VLSI Microprocessors by Wire Spacing”, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 14, No. 4, Aug. 2009, pp. 34–40.
- [52] K. Moiseev, A. Kolodny and S. Wimer, “The complexity of VLSI power-delay optimization by interconnect resizing”, *Journal of Comb. Optim.*, Vol. 23, No. 2, 2012, pp. 292–300.
- [53] J. Monteiro, S. Devadas, A. Ghosh, K. Keutzer and 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, 1997, pp. 121–127.
- [54] S. N. Pradhan, G. Paul, A. Pal and B. B. Bhattacharya, “Power Aware BDD-based Logic Synthesis Using Adiabatic Multiplexers”, *4th International Conference on Electrical and Computer Engineering, Bangladesh*, Dec. 2006, pp. 149–152.
- [55] S. S. Ramani, “Graphical Probabilistic Switching Model: Inference and Characterization for Power Dissipation in VLSI Circuits”, *Ph. D. Thesis submitted to Department of Electrical Engineering, College of Engineering, University of South Florida*, 2004.
- [56] S. Roy, A. Harm and P. Banerjee, “PowerShake: A Low Power Driven Clustering and Factoring Methodology for Boolean Expressions”, *Proceedings of Design, Automation and Test in Europe Conference*, Feb. 1998, pp. 967–968.
- [57] SAIF, “Switching Activity Interchange Format”, <http://www.synopsys.com/partners/tapin/saif.html>.
- [58] SDF, “Delay Format (SDF) for the electronic design process”, (*IEEE Std 1497-2004*), 2004.
- [59] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, “SIS: A system for

- sequential circuit synthesis”, *Technical Report UCB/ERL M92/41, EECS Department, University of California, Berkeley*, 1992.
- [60] G. Singh, “Optimization and Verification Techniques for Hardware Synthesis from Concurrent Action-Oriented Specifications”, *Ph. D. Thesis submitted to Department of Computer Engineering, Virginia Polytechnic Institute and State University*, Sept. 2008.
- [61] K. J. Singh, A. R. Wang, R. K. Brayton and A. Sangiovanni-Vincentelli, “Timing optimization of combinational logic”, *IEEE International conference on Computer-Aided Design*, Nov. 1988, pp. 282–285.
- [62] D. Sinha, D. Khalil, Y. Ismail and H. Zhou, “A Timing-Dependent Power Estimation Framework Considering Coupling”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 17, No. 6, Jun. 2009, pp. 843–847.
- [63] A. Sokolov, A. Sanyal, D. Whitley and Y. Malaiya, “Dynamic power minimization during combinational circuit testing as a traveling salesman problem”, *Proceedings of IEEE Congress on Evolutionary Computation*, Vol. 2, No. 2, Sept. 2005, pp. 1088–1095.
- [64] V. Srinivasan, “Real Delay Graphical Probabilistic Switching Model for VLSI Circuits”, *Ph. D. Thesis submitted to Department of Electrical Engineering, College of Engineering, University of South Florida*, 2000.
- [65] T. Stanion and C. Sechen, “Boolean division and factorisation using binary decision diagrams”, *IEEE Transactions on CAD of Integrated Circuits and Systems*, Vol. 13, No. 9, 1994, pp. 1179–1184.
- [66] J. R. Stuart and P. Norvig, “Artificial Intelligence: A Modern Approach”, (*2nd ed.*), *Upper Saddle River, New Jersey*, Prentice Hall.
- [67] Synopsys, “Synopsys design platforms”, <http://www.synopsys.com/products/products.html>.
- [68] C. H. Tan, “Optimisation of Power and Delay in VLSI Circuits Using Transistor Sizing and Input Ordering”, *Ph. D. Thesis submitted to Department of Com-*

*puter Science and Electrical Engineering, Massachusetts Institute of Technology, May 1994.*

- [69] S. Tani, K. Hamaguchi and S. Yajima, “The Complexity of the Optimal Variable Ordering Problems of A Shared Binary Decision Diagram ”, *Proceedings of the 4th International Symposium on Algorithms and Computation*, 1993, pp. 389–398.
- [70] R. Tavares, K. V. Eijk and M. Berkelaar, “BDD Techniques to Reduce Switching Activity in Logic Circuits”, *IEEE/ProRISC99*, 1999, pp. 497–502.
- [71] G. Theodoridis, S. Theoharis, D. Soudris, C. Goutis, “Switching activity estimation under real-gate delay using timed Boolean functions”, *IEE Proceedings on Computers and Digital Technique*, Vol. 147, No. 6, 2000, pp. 444–450.
- [72] S. Theoharis, G. Theodoridis, D. Soudris, C. Goutis and A. Thanailakis, “A fast and accurate delay dependent method for switching estimation of large combinational circuits”, *Journal of Systems Architecture*, Vol. 48, No. 4, 2002, pp. 113–124.
- [73] K. O. Tinmaung, D. Howland and R. Tessier, “Power-Aware FPGA Logic Synthesis Using Binary Decision Diagrams”, *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays, NY USA*, 2007, pp. 148–155.
- [74] H. Ueda and K. Kinoshita, “Low power design and its testability”, *Proceedings of the Fourth Asian Test Symposium, India*, Nov. 1995, pp. 361–366.
- [75] H. Ueda and K. Kinoshita, “Power Estimation And Reduction Of Cmos Circuits Considering Gate Delay”, *IEICI Transactions on Information and System*, Vol. E82-D, No. 1, Jan. 1999, pp. 301–308.
- [76] N. Vemuri, P. Kalla, K. O. TinMaung, and R. Tessier, “BDD based logic synthesis system for lut-based FPGAs”, *Design Automation of Electronic Systems*, Vol. 7, No. 4, 2002, pp. 501–525.
- [77] VIS, “VIS: Verification interacting with synthesis”, *Proceedings of CAV96. LNCS 1102, Springer-Verlag*, 1996, pp. 423–427.

- [78] R. L. Wright, M. A. Shanblatt, DCS Corp and V. A. Alexandria, “Improved switching activity estimation for behavioral and gate level designs”, *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, 2000, pp. 172–175.
- [79] D. Wu and J. Zhu, “FBDD: A folded logic synthesis system”, *In Proceedings of the International Conference on ASIC (ASICON)*, Oct. 2005, pp. 746–751.
- [80] C. Yang, M. Ciesielski and V. Singhal, “BDS: A BDD-Based Logic Optimization System”, *Proceedings of DAC*, 2000, pp. 92–97.
- [81] S. N. Yanushkevich, D. M. Miller, V. P. Shmerko and R. S. Stankovic, “Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook”, *CRS Press*, 2006, pp. 429–445.
- [82] H. Q. Ye and Z. B. Tang, “Partitioned Random Search for Global Optimization with Sampling Cost and Discounting Factor”, *Journal of optimisation theory and application*, Vol. 110, No. 2, 1998, pp. 445–455.