Advanced Techniques in Logic Synthesis, Optimizations and Applications

Sunil P. Khatri · Kanupriya Gulati
Editors

Advanced Techniques in Logic Synthesis, Optimizations and Applications

Springer

Editors
Sunil P. Khatri
Department of ECE
333F WERC, MS 3259
Texas A&M University
College Station, TX 77843-3259,
USA
sunilkhatri@tamu.edu

Kanupriya Gulati
Intel Corporation
2501 NW 229th Ave
Hillsboro, OR 97124,
USA
kanupriya.gulati@intel.com

ISBN 978-1-4419-7517-1 e-ISBN 978-1-4419-7518-8
DOI 10.1007/978-1-4419-7518-8
Springer New York Dordrecht Heidelberg London

Preface

The last few decades have seen a stupendous growth in the speed and complexity of VLSI integrated circuits. This growth has been enabled by a powerful set of electronic design automation (EDA) tools. The earliest EDA tools were two-level logic minimization and PLA folding tools. Subsequently, EDA tools were developed to address other aspects of the VLSI design flow (in addition to logic optimization) such as technology mapping, layout optimization, formal verification. However, research in logic synthesis and optimization continued to progress rapidly. Some of the research in logic synthesis tools saw broader application, to areas far removed from traditional EDA, and routinely continue to do so. While observing the recent developments and publications in logic synthesis and optimization, we felt that there was a need for a single resource which presents some recent significant developments in this area. This is how the idea of this edited monograph came about. We decided to cover some key papers in logic synthesis, optimization, and its applications, in an effort to provide an advanced practitioner a single reference source that covers the important papers in these areas over the last few years.

This monograph is organized into five sections, dealing with logic decomposition, Boolean satisfiability, Boolean matching, logic optimization, and applications of logic techniques to special design scenarios. Each of the chapters in any section is an expanded, archival version of the original paper by the chapter authors, with additional examples, results, and/or implementation details.

We dedicate this book to the area of logic synthesis and hope that it can stimulate new and exciting ideas which expand the contribution of logic synthesis to areas far beyond its traditional stronghold of VLSI integrated circuit design.

College Station, Texas — Sunil P. Khatri
Hillsboro, Oregon — Kanupriya Gulati

Contents

Contributors

Kia Bazargan University of Minnesota, Minneapolis, MN USA, kia@umn.edu

Anna Bernasconi Department of Computer Science, Universit‘a di Pisa, Pisa, Italy, annab@di.unipi.it

Kalyana C. Bollapalli NVIDIA Corporation, San Jose, CA, USA, kbollapalli@nvidia.com

Robert Brayton University of California, Berkeley, CA, USA, brayton@eecs.berkeley.edu

Chung-Ming Chan Fu Jen Catholic University, Taipei County, Taiwan, vicax95@csie.fju.edu.tw

Satrajit Chatterjee Strategic CAD Labs, Intel Corporation, Hillsboro, OR, USA, satrajit.chatterjee@intel.com

Valentina Ciriani Department of Information Technologies, Universit‘a degli Studi di Milano, Milano, Italy, valentina.ciriani@unimi.it

Jeff L. Cobb Texas Instruments, Sugar Land, TX USA, jcobb@ti.com

Xin Fan Innovations for High Performance Microelectronics, Frankfurt (Oder), Brandenburg, Germany, fan@ihp-microelectronics.com

Lei Fang Microsoft Corporation, Redmond, WA, USA, lei.fang@microsoft.com

Eugene Goldberg Northeastern University, Boston, MA, USA, eigold@ccs.neu.edu

Sivaram Gopalakrishnan Synopsys Inc., Hillsboro, OR, USA, sivaram.gopalakrishnan@synopsys.com

Eckhard Grass Innovations for High Performance Microelectronics, Frankfurt (Oder), Brandenburg, Germany, grass@ihp-microelectronics.com

Kanupriya Gulati Intel Corporation, Hillsboro, OR, USA, kanupriya.gulati@intel.com

Michael S. Hsiao Virginia Tech, Blacksburg, VA, USA, mhsiao@vt.edu

Wei-Lun Hung National Taiwan University, Taipei, Taiwan, b91076@csie.ntu.edu.tw

Jie-Hong Roland Jiang National Taiwan University, Taipei, Taiwan, jhjiang@cc.ee.ntu.edu.tw

Priyank Kalla University of Utah, Salt Lake City, UT, USA, kalla@ece.utah.edu

Hadi Katebi University of Michigan, Ann Arbor, MI, USA, hadik@eecs.umich.edu

Sunil P. Khatri Department of ECE, Texas A&M University, College Station, TX, USA, sunilkhatri@tamu.edu

Laszlo B. Kish Department of ECE, Texas A&M University, College Station, TX, USA, Laszlo.Kish@ece.tamu.edu

Nathan B. Kitchen University of Berkeley, Berkeley, CA, USA, nbk@eecs.berkeley.edu

Victor N. Kravets IBM TJ Watson Research Center, Yorktown Heights, NY, USA, kravets@us.ibm.com

Smita Krishnaswamy IBM TJ Watson Research Center, Yorktown Heights, NY, USA, skrishn@us.ibm.com

Miloš Krstić Innovations for High Performance Microelectronics, Frankfurt (Oder), Brandenburg, Germany, krstic@ihp-microelectronics.com

Andreas Kuehlmann Cadence Design Systems, Inc., San Jose, CA, USA, kuehl@cadence.com

Ruei-Rung Lee National Taiwan University, Taipei, Taiwan, r95943156@ntu.edu.tw

David J. Lilja University of Minnesota, Minneapolis, MN USA, lilja@umn.edu

Hsuan-Po Lin National Taiwan University, Taipei, Taiwan, centauricog@hotmail.com

Jung-Chang Liu Fu Jen Catholic University, Taipei County, Taiwan, binize97@csie.fju.edu.tw

Panagiotis Manolios Northeastern University, Boston, MA, USA, bjchamb@ccs.neu.edu

Igor Markov University of Michigan, Ann Arbor, MI, USA, imarkov@eecs.umich.edu

Joao Marques-Sila University College Dublin, Dublin, Ireland, jpms@ucd.ie

Alan Mishchenko Department of EECS, University of California, Berkeley, CA, USA, alanmi@eecs.berkeley.edu

Nilesh Modi IBM TJ Watson Research Center, Yorktown Heights, NY, USA, nilesh@ece.ucsb.edu

Nasir Mohyuddin Department of Electrical Engineering – Systems, University of Southern California, Los Angeles, CA, USA, mohyuddi@usc.edu

Ehsan Pakbaznia Department of Electrical Engineering – Systems, University of Southern California, Los Angeles, CA, USA, pakbazni@usc.edu

Massoud Pedram Department of Electrical Engineering – Systems, University of Southern California, Los Angeles, CA, USA, pedram@usc.edu

Jordi Planes University de Lleida, Lleida, Spain, jplanes@diei.udl.cat

Ruchir Puri IBM TJ Watson Research Center, Yorktown Heights, NY, USA, ruchir@us.ibm.com

Weikang Qian University of Minnesota, Minneapolis, MN, USA, qianx030@umn.edu

Haoxing Ren IBM TJ Watson Research Center, Yorktown Heights, NY, USA, haoxing@us.ibm.com

Marc D. Riedel University of Minnesota, Minneapolis, MN, USA, mriedel@umn.edu

Alberto Sangiovanni-Vincentelli University of Berkeley, Berkeley, CA, USA, alberto@eecs.berkeley.edu

Subarna Sinha Synopsys Inc., Mountain View, CA, USA, subarna.sinha@synopsys.com

Duncan Smith Vennsa Technologies, Inc., Toronto, ON, Canada, duncan.smith@vennsa.com

Gabriella Trucco Department of Information Technologies, Universit‘a degli Studi di Milano, Milano, Italy, gabriella.trucco@unimi.it

Andreas Veneris University of Toronto, Toronto, ON, Canada, veneris@eecg.utoronto.ca

Tiziano Villa Department of Computer Science, Universit‘a degli Studi di Verona, Verona, Italy, tiziano.villa@univr.it

Kuo-Hua Wang Fu Jen Catholic University, Taipei County, Taiwan, khwang@csie.fju.edu.tw

Yu-Shen Yang University of Toronto, Ontario, Canada, terry.yang@utoronto.ca

Qi Zhu Intel Corporation, Hillsboro, OR, USA, qi.dts.zhu@intel.com

Chapter 1
Introduction

Sunil P. Khatri and Kanupriya Gulati

With the advances in VLSI technology, enhanced optimization techniques are required to enable the design of faster electronic circuits that consume less power and occupy a smaller area. In the VLSI design cycle, significant optimization opportunities exist in the logic design stage. In recent times, several research works have been proposed in the area of logic synthesis, which can prove to be very valuable for VLSI/CAD engineers. A solid understanding and sound implementation of these advanced techniques would enable higher levels of optimization, and thus enable better electronic design. This text is a systematic collection of important recent work in the field of logic design and optimization.

Conventional logic synthesis consists of the following phases. Given an initial netlist, technology-independent optimizations [1, 3, 4] are first carried out in order to optimize various design criteria such as gate, literal, and net count. Both Boolean and algebraic techniques are used, such as kernel and cube extraction, factorization, node substitution, don't care-based optimizations [2]. During the logic decomposition phase, large gates are decomposed into smaller gates, which allows for efficient technology mapping and technology-dependent optimizations. Finally, technology mapping is applied on the decomposed netlist which is followed by technology-dependent optimizations. This book presents recent research in some of the above areas.

In order to enhance the scalability and performance of logic synthesis approaches, newer optimization styles are continually investigated. Boolean satisfiability (SAT) plays a big role in some of the recent logic optimization methodologies. Therefore, this edited volume also includes some of the latest research in SAT techniques. Further, several non-CAD systems can be viewed as an instance of logic optimization, and thus these too can take advantage of the rich body of recent research in logic synthesis and optimization. Such non-traditional applications of logic synthesis to specialized design scenarios are also included in this volume.

S.P. Khatri (✉)
Department of ECE, Texas A&M University, College Station, TX, USA
e-mail: sunilkhatri@tamu.edu

S.P. Khatri, K. Gulati (eds.), *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, DOI 10.1007/978-1-4419-7518-8_1,

The approaches described in this text are enhanced and archival versions of the corresponding original conference publications. The modifications include enhancements to the original approach, more experimental data, additional background and implementation details, along with as yet unpublished graphs and figures.

The different sections of this volume are described next.

1.1 Logic Decomposition

This section discusses the latest research in logic decomposition. The first chapter investigates restructuring techniques based on decomposition and factorization. In this chapter the authors describe new types of factorization that extend Shannon cofactoring, using projection functions that change the Hamming distance of the original minterms, to favor logic minimization of the component blocks.

The next chapter uses reachable state analysis and symbolic decomposition to improve upon the synthesis of sequential designs. The approach described uses under-approximation of unreachable states of a design to derive incomplete specification of the combinational logic. The resulting incompletely specified functions are decomposed to optimize technology-dependent synthesis. The decomposition choices are implicitly computed by using recursive symbolic bi-decomposition.

The third chapter in the topic of Boolean decomposition employs fast Boolean techniques to restructure logic networks. The techniques used are a cut-based view of a logic network, and heuristic disjoint-support decompositions. Local transformations to functions with a small number of inputs allow fast manipulations of truth tables. The use of Boolean methods reduces the structural bias associated with algebraic methods, while still allowing for high-speed.

The fourth chapter investigates Ashenhurst decomposition wherein both single and multiple output decomposition can be formulated with satisfiability solving, Craig interpolation, and functional dependency. In comparison to existing BDD-based approaches for functional decomposition, Ashenhurst decomposition does not suffer from memory explosion and scalability issues. A key feature of this approach is that variable partitioning can be automated and integrated into the decomposition process without the bound-set size restriction. Further, the approach naturally extends to nondisjoint decomposition.

The last chapter in the Boolean decomposition section focuses on scalability and quality of Boolean function bi-decomposition. The quality of a bi-decomposition is mainly determined by its variable partition. Disjoint and balanced decompositions reduce communication and circuit complexity and yield simple physical design solutions. Furthermore, finding a good or feasible partition may require costly enumeration, requiring separate decomposability checks. This chapter uses interpolation and incremental SAT solving to address these problems.

1.2 Boolean Satisfiability

In the area of Boolean satisfiability, this book presents some key ideas to make SAT more effective. The first chapter studies resolution proofs using boundary points elimination. Given a CNF formula F, boundary points are complete assignments that falsify only certain clauses of the formula. Since any resolution proof has to eventually eliminate all boundary points of F, this approach focuses on resolution proofs from the viewpoint of boundary point elimination. The authors use equivalence checking formulas to compare unsatisfiability proofs built by a conflict-driven SAT-solver. They show how every resolution of a specialized proof eliminates a boundary point, and how this enables building resolution SAT-solvers that are driven by elimination of cut boundary points.

The next chapter presents a methodology called SAT sweeping for simplifying And-Inverter Graphs (AIGs) by systematically merging graph vertices in a topological fashion starting from the inputs, using a combination of structural hashing, simulation, and SAT queries. This chapter presents the details of a SAT-sweeping approach that exploits local observability don't cares (ODCs) to increase the number of vertices merged. In order to enhance the efficiency and scalability of the approach, the authors bound the ODCs and thus the computational effort to generate them. They demonstrate that the use of ODCs in SAT sweeping results in significant graph simplification, with great benefits for Boolean reasoning in functional verification and logic synthesis techniques.

SAT-solvers find a satisfiable assignment for a propositional formula, but finding the "optimal" solution for a given function is very expensive. The next chapter discusses MIN-ONE SAT, an optimization problem which requires the satisfying assignment with the minimal number of ones, which can be easily applied to minimize an arbitrary linear objective function. The chapter proposes an approximation algorithm for MIN-ONE SAT that is efficient and achieves a tight bound on the solution quality. RelaxSAT generates a set of constraints from the objective function to guide the search, and then these constraints are gradually relaxed to eliminate the conflicts with the original Boolean SAT formula until a solution is found. The experiments demonstrate that RelaxSAT is able to handle very large instances which cannot be solved by existing MIN-ONE algorithms. The authors further show that RelaxSAT is able to obtain a very tight bound on the solution with one to two orders of magnitude speedup.

The last chapter in the Boolean satisfiability category presents an algorithm for MaxSAT that improves existing state-of-the-art solvers by orders of magnitude on industrial benchmarks. The proposed algorithm is based on efficient identification of unsatisfiable subformulas. Moreover, the new algorithm draws a connection between unsatisfiable subformulas and the maximum satisfiability problem.

1.3 Boolean Matching

Three research works are presented under the Boolean matching category. The first work proposes a methodology for Boolean matching under permutations of inputs and outputs that enables incremental logic design by identifying sections of netlist that are unaffected by incremental changes in design specifications. Identifying and reusing the equivalent subcircuits accelerates design closure. By integrating graph-based, simulation-driven, and SAT-based techniques, this methodology makes Boolean matching feasible for large designs.

The second approach in the Boolean matching category is DeltaSyn, a tool and methodology for generating the logic difference between a modified high-level specification and an implemented design. By using fast functional and structural analysis techniques, the approach first identifies equivalent signals between the original and the modified circuits. Then, by using a topologically guided dynamic matching algorithm, reusable portions of logic close to the primary outputs are identified. Finally, functional hash functions are employed to locate similar chunks of logic throughout the remainder of the circuit. Experiments on industrial designs show that together, these techniques successfully implement incremental changes while preserving an average of 97% of the pre-existing logic.

The last approach discussed in the Boolean matching section proposes an incremental learning-based algorithm, along with a Boolean satisfiability search, for solving Boolean matching. The proposed algorithm utilizes functional properties like unateness and symmetry to reduce the search space. This is followed by the simulation phase in which three types of input vector generation and checking methods are used to match the inputs of two target functions. Experimental results on large benchmark circuits demonstrate that the matching algorithm can efficiently solve the Boolean matching for large Boolean networks.

1.4 Logic Optimization

The first advanced logic optimization approach presents algebraic techniques to enhance common sub-expression extraction to allow circuit optimization. Common sub-expression elimination (CSE) is a useful optimization technique in the synthesis of arithmetic datapaths described at the RTL level.

The next chapter investigates a comprehensive methodology to automate logic restructuring in combinational and sequential circuits. This technique algorithmically constructs the required transformation by utilizing Set of Pairs of Function to be Distinguished (SPFDs). SPFDs can express more functional flexibility than traditional don't cares and have been shown to provide additional degrees of flexibility during logic synthesis. In practice, however, computing SPFDs may suffer from memory or runtime problems. This approach presents Approximate SPFDs (ASPFDs) that approximate the information contained in SPFDs by using the results

of test-vector simulation, thereby yielding an efficient and robust optimization platform.

The third chapter presents an approach to enhance the determinization of a Boolean relation by using interpolation. Boolean relations encapsulate the flexibility of a design and are therefore an important tool in system synthesis, optimization, and verification to characterize solutions to a set of Boolean constraints. For physical realization, a deterministic function often has to be extracted from a relation. Existing methods are limited in their handling of large problem instances. Experimental results for the interpolation-based relation determinization approach show that Boolean relations with thousands of variables can be effectively determinized and the extracted functions are of reasonable quality.

In this section, the fourth approach presented is a scalable approach for dual-node technology-independent optimization. This technique scales well and can minimize large designs typical of industrial circuits. The methodology presented first selects the node pairs to be minimized that are likely to give gains. For each node pair, a window or a subnetwork is created around the nodes. This windowing is done in order to allow the approach to scale to larger designs. Once the subnetwork is created, the Boolean relation, which represents the flexibility of the nodes, is computed. During this process, early quantification is performed which further extends the scalability of the approach. The Boolean relation is minimized, and the new nodes replace the original nodes in the original circuit. These steps are repeated for all selected node pairs. The authors experimentally demonstrate that this technique produces minimized technology-independent networks that are on average 12% smaller than networks produced by state-of-the-art single-node minimization techniques.

1.5 Applications to Specialized Design Scenarios

This volume presents applications of logic synthesis in non-traditional CAD areas. The first approach investigates techniques for synthesizing logic that generates new arbitrary probabilities from a given small set of probabilities. These ideas can be used in probabilistic algorithms. Instead of using different voltage levels to generate different probability values, which can be very expensive, the technique presented in the chapter alleviates this issue by generating probabilities using combinational logic.

The next chapter presents a gate-level probabilistic error propagation model which takes as input the Boolean function of the gate, the signal and error probabilities of the gate inputs, and the gate error probability and produces the error probability at the output of the gate. The presented model uses Boolean difference calculus and can be applied to the problem of calculating the error probability at the primary outputs of a multilevel Boolean circuit. The time complexity of the approach is linear in the number of gates in the circuit, and the results demonstrate

the accuracy and efficiency of the approach compared to the other known methods for error calculation in VLSI circuits.

In the third chapter in the applications category, a novel realization of combinational logic circuit is presented. In this approach, logic values 0 and 1 are implemented as sinusoidal signals of the same frequency that are phase shifted by π. The properties of such sinusoids can be used to identify a logic value without ambiguity, and hence a realizable system of logic is created. The chapter further presents a family of logic gates that can operate using such sinusoidal signals. In addition, due to orthogonality of sinusoid signals with different frequencies, multiple sinusoids could be transmitted on a single wire simultaneously, thereby naturally allowing the approach to implement multilevel logic. One advantage of such a logic family is its immunity from external additive noise and an improvement in switching (dynamic) power.

The last chapter focuses on asynchronous circuit design issues. In Systems-on-a-Chip (SOCs) and Networks-on-a-Chip (NoCs), using globally asynchronous locally synchronous (GALS) system design for pausible clocking is widely popular. This chapter investigates throughput reduction and synchronization failures introduced by existing GALS pausible clocking schemes and proposes an optimized scheme for more reliable GALS system design with higher performance. The approach minimizes the acknowledge latency and maximizes the safe timing region for inserting the clock tree.

References

1. Brayton, R.K., Hachtel, G.D., Sangiovanni-Vincentelli, A.L.: Multilevel logic synthesis. In: Proceedings of IEEE, 78(2):264–270 (1990)
2. Hassoun, S. (ed.): Logic Synthesis and Verification. San Jose, CA, USA (2001)
3. Sinha, S., Brayton, R.K.: Implementation and use of SPFDs in optimizing Boolean networks. In: Proceedings of International Conference on Computer-Aided Design, pp. 103–110. Paris, France (1998)
4. Wurth, B., Wehn, N.: Efficient calculation of Boolean relations for multi-level logic optimization. In: Proceedings of European Design and Test Conference, pp. 630–634. (1994)

Part I
Logic Decomposition

Under logic decomposition this book presents five research works. The first chapter proposes hypergraph partitioning and Shannon decomposition-based techniques for logic decomposition. The second chapter uses reachable state analysis and symbolic decomposition to improve upon the synthesis of sequential designs. Fast Boolean decomposition techniques employing a cut-based view of a logic network and heuristic disjoint-support decompositions are presented in the third work. The fourth approach performs Ashenhurst decomposition formulated using satisfiability, Craigs interpolation, and functional dependency. This last chapter in this category uses interpolation and incremental SAT solving to improve the quality of Boolean function decomposition.

Chapter 2
Logic Synthesis by Signal-Driven Decomposition

Anna Bernasconi, Valentina Ciriani, Gabriella Trucco, and Tiziano Villa

Abstract This chapter investigates some restructuring techniques based on decomposition and factorization, with the objective to move critical signals toward the output while minimizing area. A specific application is synthesis for minimum switching activity (or high performance), with minimum area penalty, where decompositions with respect to specific critical variables are needed (the ones of highest switching activity, for example). In order to reduce the power consumption of the circuit, the number of gates that are affected by the switching activity of critical signals is maintained constant. This chapter describes new types of factorization that extend Shannon cofactoring and are based on projection functions that change the Hamming distance among the original minterms to favor logic minimization of the component blocks. Moreover, the proposed algorithms generate and exploit don't care conditions in order to further minimize the final circuit. The related implementations, called P-circuits, show experimentally promising results in area with respect to classical Shannon cofactoring.

2.1 Introduction

In recent years, power has become an important factor during the design phase. This trend is primarily due to the remarkable growth of personal computing devices, embedded systems, and wireless communications systems that demand high-speed computation and complex functionality with low power consumption. In these applications, average power consumption is a critical design concern.

Low-power design methodologies must consider power at all stages of the design process. At the logic synthesis level, logic transformations proved to be an effective technique to reduce power consumption by restructuring a mapped circuit through permissible signal substitution or perturbation [1]. A fundamental step in VLSI design is logic synthesis of high-quality circuits matching a given specification. The

A. Bernasconi (✉)
Department of Computer Science, Università di Pisa, Pisa, Italy
e-mail: annab@di.unipi.it

Based on [5], pp.1464–1469, 20–24 April 2009 © [2009] IEEE.

S.P. Khatri, K. Gulati (eds.), *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, DOI 10.1007/978-1-4419-7518-8_2,

performance of the circuit can be expressed in terms of several factors, such as area, delay, power consumption, and testability properties. Unfortunately, these factors often contradict each other, in the sense that it is very difficult to design circuits that guarantee very good performances with respect to all of them. In fact, power consumption is often studied as a single minimization objective without taking into account important factors such as area and delay.

In CMOS technology, power consumption is characterized by three components: dynamic, short-circuit, and leakage power dissipation, of which dynamic power dissipation is the predominant one. Dynamic power dissipation is due to the charge and discharge of load capacitances, when the logic value of a gate output toggles; switching a gate may trigger a sequence of signal changes in the gates of its output cone, increasing dynamic power dissipation. So, reducing switching activity reduces dynamic power consumption. Previous work proposed various transformations to decrease power consumption and delay (for instance [11, 14, 16] for performance and [1, 13, 15] for low power), whereby the circuit is restructured in various ways, e.g., redeploying signals to avoid critical areas, bypassing large portions of a circuit. For instance, if we know the switching frequency of the input signals, a viable strategy to reduce dynamic power is to move the signals with the highest switching frequency closer to the outputs, in order to reduce the part of the circuit affected by the switching activity of these signals. Similarly for performance, late-arriving signals are moved closer to the outputs to decrease the worst-case delay.

The aim of our research is a systematic investigation of restructuring techniques based on decomposition/factorization, with the objective to move critical signals toward the output and avoid losses in area. A specific application is synthesis for minimum switching activity (or high performance), with minimum area penalty. Differently from factorization algorithms developed only for area minimization, we look for decompositions with respect to specific critical variables (the ones of highest switching activity, for example). This is exactly obtained by Shannon cofactoring, which decomposes a Boolean function with respect to a chosen splitting variable; however, when applying Shannon cofactoring, the drawback is that too much area redundancy might be introduced because large cubes are split between two disjoint subspaces, whereas no new cube merging will take place as the Hamming distance among the projected minterms do not change.

In this chapter we investigate thoroughly the more general factorization introduced in [5], a decomposition that extends straightforward Shannon cofactoring; instead of cofactoring a function f only with respect to single variables as Shannon does, we cofactor with respect to more complex functions, expanding f with respect to the orthogonal basis $\overline{x}_i \oplus p$ (i.e., $x_i = p$) and $x_i \oplus p$ (i.e., $x_i \neq p$), where $p(x)$ is a function defined over all variables except x_i. We study different functions $p(x)$ trading-off quality vs. computation time. Our factorizations modify the Hamming distance among the on-set minterms, so that more logic minimization may be performed on the projection of f onto the two disjoint subspaces $x_i = p$ and $x_i \neq p$, while signals are moved in the circuit closer to the output. We then introduce and study another form of decomposition, called *decomposition with intersection*, where a function f is projected onto three overlapping subspaces of the Boolean

space $\{0, 1\}^n$ in order to favor area minimization avoiding cube fragmentation (e.g., cube splitting for the cubes intersecting both subspaces $x_i = p$ and $x_i \neq p$). More precisely, we partition the on-set minterms of f into three sets: $f|_{x_i=p}$ and $f|_{x_i\neq p}$, representing the projections of f onto the two disjoint subspaces $x_i = p$ and $x_i \neq p$, and a third set $I = f|_{x_i=p} \cap f|_{x_i\neq p}$, which contains all minterms of f whose projections onto $x_i = p$ and $x_i \neq p$ are identical. Observe that each point in I corresponds to two different points of f that could be merged in a cube, but are split into the two spaces $x_i = p$ and $x_i \neq p$. Thus, we can avoid cube fragmentation keeping the points in I unprojected. Moreover, given that the points in the intersection I must be covered, we can project them as *don't cares* in the two spaces $f|_{x_i=p}$ and $f|_{x_i\neq p}$ to ease the minimization of $f|_{x_i=p} \setminus I$ and $f|_{x_i\neq p} \setminus I$. Observe that, while classical don't care sets are specified by the user or are derived from the surrounding environment, our don't cares are dynamically constructed during the synthesis phase.

The circuits synthesized according to these decompositions are called *Projected Circuits*, or *P-circuits*, without and with intersection. We provide minimization algorithms to compute optimal P-circuits and argue how augmenting P-circuits with at most a pair of multiplexers guarantees full testability under the single stuck-at-fault model. We also show that the proposed decomposition technique can be extended and applied to move all critical signals, and not just one, toward the output, still avoiding losses in area.

The chapter is organized as follows. Section 2.2 describes the new theory of decomposition based on generalized cofactoring, which is applied in Section 2.3 to the synthesis of Boolean functions as *P-circuits*. Section 2.4 extends the decomposition from single to multiple variables. Experiments and conclusions are reported in Sections 2.5 and 2.6, respectively.

2.2 Decomposition Methods

How to decompose Boolean functions is an ongoing research area to explore alternative logic implementations. A technique to decompose Boolean functions is based on expanding them according to an orthogonal basis (see, for example [8], section 3.15), as in the following definition, where a function f is decomposed according to the basis $(g, \overline{g})$.

Definition 2.1 Let $f = (f_{\text{on}}, f_{\text{dc}}, f_{\text{off}})$ be an incompletely specified function and g be a completely specified function, the *generalized cofactor* of f with respect to g is the incompletely specified function $\text{co}(f, g) = (f_{\text{on}}.g, f_{\text{dc}} + \overline{g}, f_{\text{off}}.g)$.

This definition highlights that in expanding a Boolean function we have two degrees of freedom: choosing the basis (in this case, the function g) and choosing one completely specified function included in the incompletely specified function $\text{co}(f, g)$. This flexibility can be exploited according to the purpose of the expansion. For instance, when $g = x_i$, we have $\text{co}(f, x_i) = (f_{\text{on}}.x_i, f_{\text{dc}} + \overline{x}_i, f_{\text{off}}.x_i)$. Notice that the well-known Shannon cofactor $f_{x_i} = f(x_1, \ldots, (x_i = 1), \ldots, x_n)$ is a

completely specified function contained in $\mathrm{co}(f, x_i) = (f_{\mathrm{on}}.x_i, f_{\mathrm{dc}} + \overline{x}_i, f_{\mathrm{off}}.x_i)$ (since $f_{\mathrm{on}}.x_i \subseteq f_{x_i} \subseteq f_{\mathrm{on}}.x_i + f_{\mathrm{dc}} + \overline{x}_i = f_{\mathrm{on}} + f_{\mathrm{dc}} + \overline{x}_i$); moreover, f_{x_i} is the unique cover of $\mathrm{co}(f, x_i)$ independent from the variable x_i.

We introduce now two types of expansion of a Boolean function that yield decompositions with respect to a chosen variable (as in Shannon cofactoring), but are also area-efficient because they favor minimization of the logic blocks so obtained. Let $f(X) = (f_{\mathrm{on}}(X), f_{\mathrm{dc}}(X), f_{\mathrm{off}}(X))$ be an incompletely specified function depending on the set $X = \{x_1, x_2, \ldots, x_n\}$ of n binary variables. Let $X^{(i)}$ be the subset of X containing all variables but x_i, i.e., $X^{(i)} = X \setminus \{x_i\}$, where $x_i \in X$. Consider now a completely specified Boolean function $p(X^{(i)})$ depending only on the variables in $X^{(i)}$. We introduce two decomposition techniques based on the projections of the function f onto two complementary subsets of the Boolean space $\{0, 1\}^n$ defined by the function p. More precisely, we note that the space $\{0, 1\}^n$ can be partitioned into two sets: one containing the points for which $x_i = p(X^{(i)})$ and the other containing the points for which $x_i \neq p(X^{(i)})$. Observe that the characteristic functions of these two subsets are $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, respectively, and that these two sets have equal cardinality. We denote by $f|_{x_i=p}$ and $f|_{x_i \neq p}$ the projections of the points of $f(X)$ onto the two subsets where $x_i = p(X^{(i)})$ and $x_i \neq p(X^{(i)})$, respectively. Note that these two functions only depend on the variables in $X^{(i)}$. The first decomposition technique, already described in [12] and [6], is defined as follows.

Definition 2.2 Let $f(X)$ be an incompletely specified function, $x_i \in X$, and $p(X^{(i)})$ be a completely specified function. The (x_i, p)-*decomposition* of f is the algebraic expression

$$f = (\overline{x}_i \oplus p) f|_{x_i=p} + (x_i \oplus p) f|_{x_i \neq p}.$$

First of all we observe that each minterm of f is projected onto one and only one subset. Indeed, let $m = m_1 m_2 \cdots m_n$ be a minterm of f; if $m_i = p(m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n)$, then m is projected onto the set where $x_i = p(X^{(i)})$, otherwise m is projected onto the complementary set where $x_i \neq p(X^{(i)})$. The projection simply consists in eliminating m_i from m. For example, consider the function f shown on the left side of Fig. 2.1 with $f_{\mathrm{on}} = \{0000, 0001, 0010, 0101, 1001, 1010, 1100, 1101\}$ and $f_{\mathrm{dc}} = \{0111\}$. Let p be the simple Boolean function x_2, and x_i be x_1. The Boolean space $\{0, 1\}^4$ can be partitioned into the two sets $x_1 = x_2$ and $x_1 \neq x_2$ each containing 2^3 points. The projections of f onto these two sets are $f_{\mathrm{on}}|_{x_1=x_2} = \{000, 001, 010, 100, 101\}$, $f_{\mathrm{dc}}|_{x_1=x_2} = \emptyset$, and $f_{\mathrm{on}}|_{x_1 \neq x_2} = \{101, 001, 010\}$, $f_{\mathrm{dc}}|_{x_1 \neq x_2} = \{111\}$.

Second, observe that these projections do not preserve the Hamming distance among minterms, since we eliminate the variable x_i from each minterm, and two minterms projected onto the same subset could have different values for x_i. The Hamming distance is preserved only if the function $p(X^{(i)})$ is a constant, that is when the (x_i, p)-decomposition corresponds to the classical Shannon decomposition. The fact that the Hamming distance may change could be useful when f is

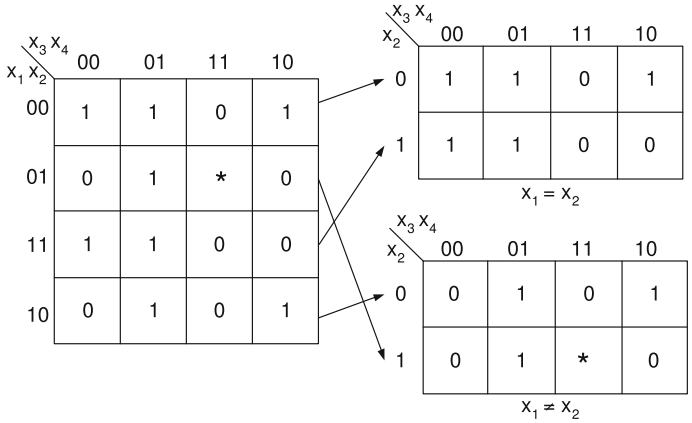

Fig. 2.1 An example of projection of the incompletely specified function f onto the spaces $x_1 = x_2$ and $x_1 \neq x_2$

represented in SOP form, as bigger cubes could be built in the projection sets. For example, consider again the function f shown on the left side of Fig. 2.1. The points 0000 and 1100 contained in f_{on} have Hamming distance equal to 2, and thus cannot be merged in a cube, while their projections onto the space $f_{\text{on}}|_{x_1=x_2}$ (i.e., 000 and 100, respectively) have Hamming distance equal to 1, and they form the cube $\overline{x}_3\overline{x}_4$.

On the other hand, the cubes intersecting both subsets $x_i = p(X^{(i)})$ and $x_i \neq p(X^{(i)})$ are divided into two smaller subcubes. For instance, in our running example, the cube $\overline{x}_3 x_4$ of function f_{on} is split into the two sets $x_1 = x_2$ and $x_1 \neq x_2$ forming a cube in $f_{\text{on}}|_{x_1=x_2}$ and one in $f_{\text{on}}|_{x_1\neq x_2}$, as shown on the right side of Fig. 2.1.

Observe that the cubes that end up to be split may contain pairs of minterms, whose projections onto the two sets are identical. In our example, $\overline{x}_3 x_4$ is the cube corresponding to the points $\{0001, 0101, 1001, 1101\}$, where 0001 and 1101 are projected onto $f_{\text{on}}|_{x_1=x_2}$ and become 001 and 101, respectively, and 0101 and 1001 are projected onto $f_{\text{on}}|_{x_1\neq x_2}$ and again become 101 and 001, respectively. Therefore, we can characterize the set of these minterms as $I = f|_{x_i=p} \cap f|_{x_i\neq p}$. Note that the points in I do not depend on x_i. In our example $I_{\text{on}} = f_{\text{on}}|_{x_1=x_2} \cap f_{\text{on}}|_{x_1\neq x_2} = \{001, 010, 101\}$, and $I_{\text{dc}} = \emptyset$.

In order to overcome the splitting of some cubes, we could keep I unprojected and project only the points in $f|_{x_i=p} \setminus I$ and $f|_{x_i\neq p} \setminus I$, obtaining the expression $f = (\overline{x}_i \oplus p)(f|_{x_i=p} \setminus I) + (x_i \oplus p)(f|_{x_i\neq p} \setminus I) + I$.

However, we are left with another possible drawback: some points of I could also belong to cubes covering points of $f|_{x_i=p}$ and/or $f|_{x_i\neq p}$, and their elimination could cause the fragmentation of these cubes. Thus, eliminating these points from the projected subfunctions would not be always convenient. On the other hand, some points of I are covered only by cubes entirely contained in I. Therefore keeping them both in I and in the projected subfunctions would be useless and expensive. In our example, since $I_{\text{on}} = \{001, 010, 101\}$, in $f_{\text{on}}|_{x_1=x_2}$ the points 001 and 101