Towards a Design Flow for Reversible Logic

Robert Wille • Rolf Drechsler

Towards a Design Flow for Reversible Logic

Springer

Robert Wille
Institute of Computer Science
University of Bremen
Bibliothekstr. 1
28359 Bremen
Germany
rwille@informatik.uni-bremen.de

Rolf Drechsler
Institute of Computer Science
University of Bremen
Bibliothekstr. 1
28359 Bremen
Germany
drechsle@informatik.uni-bremen.de

ISBN 978-90-481-9578-7 e-ISBN 978-90-481-9579-4
DOI 10.1007/978-90-481-9579-4
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2010932404

Cover design: eStudio Calamar S.L.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The development of computing machines found great success in the last decades. But the ongoing miniaturization of integrated circuits will reach its limits in the near future. Shrinking transistor sizes and power dissipation are the major barriers in the development of smaller and more powerful circuits. Reversible logic provides an alternative that may overcome many of these problems in the future. For low-power design, reversible logic offers significant advantages since zero power dissipation will only be possible if computation is reversible. Furthermore, quantum computation profits from enhancements in this area, because every quantum circuit is inherently reversible and thus requires reversible descriptions.

However, since reversible logic is subject to certain restrictions (e.g. fanout and feedback are not directly allowed), the design of reversible circuits significantly differs from the design of traditional circuits. Nearly all steps in the design flow (like synthesis, verification, or debugging) must be redeveloped so that they become applicable to reversible circuits as well. But research in reversible logic is still at the beginning. No continuous design flow exists so far.

In this book, contributions to a design flow for reversible logic are presented. This includes advanced methods for synthesis, optimization, verification, and debugging. Formal methods like Boolean satisfiability and decision diagrams are thereby exploited. By combining the techniques proposed in the book, it is possible to synthesize reversible circuits representing large functions. Optimization approaches ensure that the resulting circuits are of small cost. Finally, a method for equivalence checking and automatic debugging allows to verify the obtained results and helps to accelerate the search for bugs in case of errors in the design. Combining the respective approaches, a first design flow for reversible circuits of significant size results.

This book addresses computer scientists and computer architects and does not require previous knowledge about the physics of reversible logic or quantum computation. The respective concepts as well as the used models are briefly introduced.

All approaches are described in a self-contained manner. The content of the book does not only conveys a coherent overview about current research results, but also builds the basis for future work on a design flow for reversible logic.

Bremen

Robert Wille
Rolf Drechsler

Acknowledgements

This book is the result of more than three years of intensive research in the area of reversible logic. During this time, we experienced many support from different people for which we would like to thank them very much.

In particular, the Group of Computer Architecture at the University of Bremen earns many thanks for providing a comfortable and inspirational environment. Many thanks go to Stefan Frehse, Daniel Große, Lisa Jungmann, Hoang M. Le, Sebastian Offermann, and Mathias Soeken who actively helped in the development of the approaches described in this book.

Sincere thanks also go to Prof. D. Michael Miller from the University of Victoria, Prof. Gerhard W. Dueck from the University of New Brunswick, and Dr. Mehdi Saeedi from the Amirkabir University of Technology in Tehran for very fruitful collaborations. In this context, we would like to thank the German Academic Exchange Service (DAAD) which enabled the close contact with the groups in Canada.

Special thanks go to the German Research Foundation (DFG) which funded parts of this work under the contract number DR 287/20-1.

Finally, we would like to thank Marc Messing who did a great job of proofreading as well as Christiane and Shawn Mitchell who closely checked the manuscript for English style and grammar.

Contents

1 Introduction 1

2 Preliminaries 7
2.1 Background 7
2.1.1 Reversible Functions 7
2.1.2 Reversible Circuits 9
2.1.3 Quantum Circuits 13
2.2 Decision Diagrams 16
2.2.1 Binary Decision Diagrams 17
2.2.2 Quantum Multiple-valued Decision Diagrams 19
2.3 Satisfiability Solvers 21
2.3.1 Boolean Satisfiability 21
2.3.2 Extended SAT Solvers 22

3 Synthesis of Reversible Logic 27
3.1 Current Synthesis Steps 28
3.1.1 Embedding Irreversible Functions 28
3.1.2 Transformation-based Synthesis 30
3.2 BDD-based Synthesis 31
3.2.1 General Idea 32
3.2.2 Exploiting BDD Optimization 34
3.2.3 Theoretical Consideration 37
3.2.4 Experimental Results 39
3.3 SyReC: A Reversible Hardware Language 46
3.3.1 The SyReC Language 47
3.3.2 Synthesis of the Circuits 50
3.3.3 Experimental Results 53
3.4 Summary and Future Work 56

4 Exact Synthesis of Reversible Logic 57
4.1 Main Flow 58
4.2 SAT-based Exact Synthesis 61

4.2.1 Encoding for Toffoli Circuits . . . 61
4.2.2 Encoding for Quantum Circuits . . . 65
4.2.3 Handling Irreversible Functions . . . 68
4.2.4 Experimental Results . . . 70
4.3 Improved Exact Synthesis . . . 76
4.3.1 Exploiting Higher Levels of Abstractions . . . 77
4.3.2 Quantified Exact Synthesis . . . 81
4.3.3 Experimental Results . . . 84
4.4 Summary and Future Work . . . 91

5 Embedding of Irreversible Functions . . . 93
5.1 The Embedding Problem . . . 94
5.2 Don't Care Assignment . . . 96
5.2.1 Methods . . . 96
5.2.2 Experimental Results . . . 99
5.3 Synthesis with Output Permutation . . . 100
5.3.1 General Idea . . . 102
5.3.2 Exact Approach . . . 104
5.3.3 Heuristic Approach . . . 105
5.3.4 Experimental Results . . . 106
5.4 Summary and Future Work . . . 111

6 Optimization . . . 113
6.1 Adding Lines to Reduce Circuit Cost . . . 114
6.1.1 General Idea . . . 114
6.1.2 Algorithm . . . 116
6.1.3 Experimental Results . . . 117
6.2 Reducing the Number of Circuit Lines . . . 124
6.2.1 General Idea . . . 125
6.2.2 Algorithm . . . 127
6.2.3 Experimental Results . . . 130
6.3 Optimizing Circuits for Linear Nearest Neighbor Architectures . . . 131
6.3.1 NNC-optimal Decomposition . . . 133
6.3.2 Optimizing NNC-optimal Decomposition . . . 134
6.3.3 Experimental Results . . . 138
6.4 Summary and Future Work . . . 138

7 Formal Verification and Debugging . . . 143
7.1 Equivalence Checking . . . 144
7.1.1 The Equivalence Checking Problem . . . 145
7.1.2 QMDD-based Equivalence Checking . . . 145
7.1.3 SAT-based Equivalence Checking . . . 148
7.1.4 Experimental Results . . . 150
7.2 Automated Debugging and Fixing . . . 154
7.2.1 The Debugging Problem . . . 155
7.2.2 Determining Error Candidates . . . 157

7.2.3 Determining Error Locations 161
7.2.4 Fixing Erroneous Circuits 165
7.2.5 Experimental Results 167
7.3 Summary and Future Work 173

8 Summary and Conclusions 175

References 177

Index 183

Acronyms

BDDs	Binary Decision Diagrams
CMOS	Complementary Metal Oxide Semiconductor
CNF	Conjunctive Normal Form
CNOT	Controlled-NOT
d	Number of gates (depth) of a circuit
HDL	Hardware Description Language
LNN	Linear Nearest Neighbor
NNC	Nearest Neighbor Cost
MCF	Multiple control Fredkin
MCT	Multiple control Toffoli
P	Peres
QMDD	Quantum Multiple-valued Decision Diagram
QF_BV	Quantifier free bit-vector logic
QBF	Quantified Boolean Formulas
SAT	Boolean satisfiability
SMT	SAT Modulo Theories
SWOP	Synthesis with Output Permutation