

Contents

Abstract	iii
Zusammenfassung	v
Contents	vii
1 Introduction	1
1.1 Quantum computing	1
1.2 Qubits and gates	3
1.3 Programming quantum computers	4
1.4 Classical simulation of quantum circuits	6
1.5 Outline	9
I Compilation and optimization of quantum programs	11
2 A software methodology for compiling quantum programs	15
2.1 Quantum programs	15
2.2 A toolchain for quantum programming	18
2.2.1 Providing abstractions	18
2.2.2 Enabling performance	21
2.2.3 From logical operations to hardware	22
2.2.4 Software and hardware backends	23
2.2.5 Implementation: ProjectQ	24
3 Using Hoare logic to optimize quantum programs	27
3.1 Quantum programs	28
3.2 Hoare types and their use for optimization	29
3.3 Compiler optimization via Hoare logic	30
3.4 Practical example: Optimizing floating-point arithmetic	33
3.5 Formalization and generalization	37
3.5.1 Formalization of our basic methodology	38

3.5.2	Generalized optimization methodology	39
3.6	Implementation using ProjectQ and Z3	42
3.7	Results and comparison	44
3.8	Summary and future work	47
4	Reducing resource requirements by optimizing error tolerances	49
4.1	Compilation and approximation	50
4.2	Error-propagation in quantum circuits	52
4.3	Example application	56
4.4	Implementation and numerical results	58
4.5	Conclusion	62
II	Improved simulation of quantum computers	63
5	0.5 petabyte simulation of a 45-qubit quantum circuit	67
5.1	Optimizations	70
5.1.1	Standard optimizations	70
5.1.2	Single-core	70
5.1.3	Single-node	72
5.1.4	Multi-node	73
5.1.5	Global gate specialization	74
5.1.6	Circuit optimizations: Gate scheduling and qubit mapping	75
5.2	Implementation and results	79
5.2.1	Cori II	79
5.2.2	Edison	83
5.3	Conclusion	85
6	Emulation of quantum circuits	87
6.1	Quantum computer emulation	88
6.1.1	Arithmetic operations and mathematical functions	89
6.1.2	Quantum Fourier transform	90
6.1.3	Quantum phase estimation	90
6.1.4	Measurements	91
6.2	Performance results	92
6.2.1	Experimental setup	92
6.2.2	Arithmetic operations and mathematical functions	92
6.2.3	Quantum Fourier transform	93
6.2.4	Quantum phase estimation	94
6.2.5	Comparison against other simulators	96
6.3	Conclusion	98

III	Quantum circuits for mathematical functions	99
7	Factoring using $2n+2$ qubits with Toffoli based modular multiplication	103
7.1	Toffoli based in-place addition	105
7.1.1	Serial implementation	107
7.1.2	Runtime analysis of the serial implementation	109
7.1.3	Parallel / low-depth version	109
7.2	Modular multiplication	110
7.3	Implementation and simulation results	110
7.4	Advantages of Toffoli circuits	113
7.4.1	Single-qubit rotation gate synthesis	113
7.4.2	Design for testability	113
7.5	Conclusion	114
8	Optimizing quantum circuits for arithmetic	115
8.1	Learning from classical libraries	116
8.2	Evaluation of piecewise polynomial approximations	117
8.3	Software stack module for piecewise smooth functions	120
8.4	Inverse square root	121
8.5	Arcsine	123
8.6	Conclusion	124
	Implementation details	125
8.A	Basic circuit building blocks for fixed-point arithmetic	125
8.B	Resource estimates for polynomial evaluation	126
8.C	(Inverse) Square root	128
8.C.1	Reversible implementation	128
8.C.2	Resource estimates	130
8.D	Arcsine	131
8.D.1	Reversible implementation	132
8.D.2	Resource estimates	133
8.E	Simulation results	134
8.E.1	Piecewise polynomial approximation	134
8.E.2	(Inverse) Square root	134
8.E.3	Arcsine	135
9	Conclusion and outlook	139
	List of publications	141
	Bibliography	143

CONTENTS

Acknowledgments	155
Curriculum Vitae	157