Contents

List of Figures ix
List of Tables xiv
Acknowledgments xvii
Foreword xix

1. INTRODUCTION 1
   1.1 Why Source Code Optimization? 3
      1.1.1 Abstraction Levels of Code Optimization 4
      1.1.2 Survey of the traditional Code Optimization Process 5
      1.1.3 Scopes for Code Optimization 8
   1.2 Target Application Domain 10
   1.3 Goals and Contributions 11
   1.4 Outline of the Book 12

2. EXISTING CODE OPTIMIZATION TECHNIQUES 15
   2.1 Description Optimization 16
   2.2 Algorithm Selection 17
   2.3 Memory Hierarchy Exploitation 17
   2.4 Processor-independent Source Code Optimizations 19
   2.5 Processor-specific Source Code Optimizations 20
   2.6 Compiler Optimizations 21
      2.6.1 Loop Optimizations for High-Performance Computing 21
      2.6.2 Code Generation for embedded Processors 23

3. FUNDAMENTAL CONCEPTS FOR OPTIMIZATION AND EVALUATION 25
   3.1 Polyhedral Modeling 25

3.2 Optimization using Genetic Algorithms 29
3.3 Benchmarking Methodology 34
3.3.1 Profiling of Pipeline and Cache Performance 34
3.3.2 Compilation for Runtime and Code Size Measurement 35
3.3.3 Estimation of Energy Dissipation 37
3.4 Summary 38

4. INTERMEDIATE REPRESENTATIONS 41
4.1 Low-level Intermediate Representations 42
4.1.1 GNU RTL 42
4.1.2 Trimaran ELCOR IR 43
4.2 Medium-level Intermediate Representations 44
4.2.1 Sun IR 44
4.2.2 IR-C / LANCE 45
4.3 High-level Intermediate Representations 47
4.3.1 SUIF 47
4.3.2 IMPACT 48
4.4 Selection of an IR for Source Code Optimization 48
4.5 Summary 51

5. LOOP NEST SPLITTING 53
5.1 Introduction 53
5.1.1 Control Flow Overhead in Data dominated Software 54
5.1.2 Control Flow Overhead caused by Data Partitioning 55
5.1.3 Splitting of Loop Nests for Control Flow Optimization 57
5.2 Related Work 60
5.3 Analysis and Optimization Techniques for Loop Nest Splitting 62
5.3.1 Preliminaries 64
5.3.2 Condition Satisfiability 67
5.3.3 Condition Optimization 69
5.3.3.1 Chromosomal Representation 70
5.3.3.2 Fitness Function 73
5.3.3.3 Polytope Generation 79
5.3.4 Global Search Space Construction 80
5.3.5 Global Search Space Exploration 82
5.3.5.1 Chromosomal Representation 83
5.3.5.2 Fitness Function 84
5.3.6 Source Code Transformation 90
5.3.6.1 Generation of the splitting If-Statement 90

5.3.6.2 Loop Nest Duplication 92
5.4 Extensions for Loops with non-constant Bounds 94
5.5 Experimental Results 100
5.5.1 Stand-alone Loop Nest Splitting 101
5.5.1.1 Pipeline and Cache Performance 102
5.5.1.2 Execution Times and Code Sizes 107
5.5.1.3 Energy Consumption 109
5.5.2 Combined Data Partitioning and Loop Nest Splitting for energy-efficient Scratchpad Utilization 110
5.5.2.1 Execution Times and Code Sizes 111
5.5.2.2 Energy Consumption 113
5.6 Summary 116
6. ADVANCED CODE HOISTING 119
6.1 A motivating Example 119
6.2 Related Work 124
6.3 Analysis Techniques for Advanced Code Hoisting 129
6.3.1 Common Subexpression Identification 130
6.3.1.1 Collection of equivalent Expressions 130
6.3.1.2 Computation of Live Ranges of Expressions 132
6.3.2 Determination of the outermost Loop for a CSE 139
6.3.3 Computation of Execution Frequencies using Polytope Models 141
6.4 Experimental Results 154
6.4.1 Pipeline and Cache Performance 154
6.4.2 Execution Times and Code Sizes 158
6.4.3 Energy Consumption 160
6.5 Summary 161
7. RING BUFFER REPLACEMENT 163
7.1 Motivation 164
7.2 Optimization Steps 169
7.2.1 Ring Buffer Scalarization 170
7.2.2 Loop Unrolling for Ring Buffers 171
7.3 Experimental Results 173
7.3.1 Pipeline and Cache Performance 173
7.3.2 Execution Times and Code Sizes 176
7.3.3 Energy Consumption 178
7.4 Summary 179

8. SUMMARY AND CONCLUSIONS 181
8.1 Summary and Contribution to Research 181
8.2 Future Work 184

Appendices 188
Experimental Comparison of SUIF and IR-C / LANCE 189
Benchmarking Data for Loop Nest Splitting 191
B.1 Values of performance-monitoring Counters 191
B.1.1 Intel Pentium III 191
B.1.2 Sun UltraSPARC III 193
B.1.3 MIPS R10000 194
B.2 Execution Times and Code Sizes 195
B.3 Energy Consumption of an ARM7TDMI Core 197
B.4 Combined Data Partitioning and Loop Nest Splitting 198
B.4.1 Execution Times and Code Sizes 198
B.4.2 Energy Consumption 199
Benchmarking Data for Advanced Code Hoisting 201
C.1 Values of performance-monitoring Counters 201
C.1.1 Intel Pentium III 201
C.1.2 Sun UltraSPARC III 202
C.1.3 MIPS R10000 203
C.2 Execution Times and Code Sizes 203
C.3 Energy Consumption of an ARM7TDMI Core 205
Benchmarking Data for Ring Buffer Replacement 207
D.1 Values of performance-monitoring Counters 207
D.1.1 Intel Pentium III 207
D.1.2 Sun UltraSPARC III 208
D.1.3 MIPS R10000 208
D.2 Execution Times and Code Sizes 208
D.3 Energy Consumption of an ARM7TDMI Core 209

References 211

About the Authors 221

Index 223