Contents

1 Introduction 1
1.1 Design of Embedded Real-Time Systems 3
1.1.1 Industrial Practice for Meeting Timing Constraints 5
1.1.2 WCET-Aware Compilation 7
1.1.2.1 Challenges for WCET Minimization 8
1.2 Contribution of This Work 9
1.3 Outline 11

2 WCET Analysis Techniques 13
2.1 Introduction 13
2.2 Approaches for WCET Analysis 14
2.2.1 Measurement-Based Approach 14
2.2.2 Static Approach 14
2.2.3 Hybrid Approach 15
2.3 Basic Concepts for Static WCET Analysis 15
2.3.1 Control Flow 16
2.3.2 Processor Behavioral Analysis 17
2.3.2.1 Context-Dependent Timing Analysis 18
2.3.2.2 Timing Anomalies 18
2.3.3 Flow Facts 19
2.3.4 Bound Calculation 19
2.4 Static WCET Analyzer aiT 21

3 WCC—WCET-Aware C Compiler 23
3.1 Introduction 23
3.2 Related Work 24
3.3 Structure of the WCC Compiler 26
3.3.1 Compiler Frontend ICD-C IR 26
3.3.2 Standard Source Code Level Optimizations 29
3.3.3 Code Selector 29
3.3.4 Compiler Backend LLIR 30
3.3.5 Standard Assembly Level Optimizations 31

3.3.6 Code Generator . . . 32
3.4 Integration of WCET Analyzer . . . 33
3.4.1 Conversion from LLIR to CRL2 . . . 33
3.4.1.1 Operation Identification . . . 34
3.4.1.2 Exploitation of Memory Hierarchy Specification . 36
3.4.1.3 Loop Transformation . . . 37
3.4.2 Invocation of aiT . . . 38
3.4.3 Import of Worst-Case Execution Data . . . 39
3.5 Modeling of Flow Facts . . . 39
3.5.1 Specification of Flow Facts . . . 40
3.5.2 Translation and Transformation of Flow Facts . . . 41
3.6 Static Loop Analysis . . . 41
3.6.1 Related Work . . . 42
3.6.2 Abstract Interpretation . . . 43
3.6.2.1 Modified Abstract Interpretation . . . 44
3.6.3 Interprocedural Program Slicing . . . 46
3.6.4 Polyhedral Evaluation . . . 48
3.6.4.1 Polyhedral Condition Evaluation . . . 48
3.6.4.2 Polyhedral Loop Evaluation . . . 50
3.6.4.3 Preconditions for Loop Evaluation . . . 50
3.6.4.4 Ehrhart Polynomial Evaluation . . . 51
3.6.4.5 Static Statement Evaluation . . . 52
3.6.5 Experimental Results . . . 52
3.6.5.1 Determination of Loop Iteration Counts . . . 53
3.6.5.2 Analysis Time . . . 53
3.7 Back-Annotation . . . 54
3.7.1 Mapping of Low-Level to High-Level Components . . . 55
3.7.2 Back-Annotated Data . . . 57
3.8 TriCore Processor . . . 57

4 WCET-Aware Source Code Level Optimizations . . . 61
4.1 Introduction . . . 62
4.2 Existing Code Optimization Techniques . . . 63
4.3 Procedure Cloning . . . 64
4.3.1 Motivating Example . . . 65
4.3.2 Related Work . . . 68
4.3.3 Standard Procedure Cloning . . . 69
4.3.3.1 Selection of Functions to Be Cloned . . . 70
4.3.4 Impact of Standard Cloning on WCET . . . 71
4.3.5 Experimental Results for Standard Procedure Cloning . . . 72
4.3.5.1 WCET . . . 73
4.3.5.2 ACET . . . 74
4.3.5.3 Code Size . . . 74
4.3.6 WCET-Aware Procedure Cloning . . . 75
4.3.7 Experimental Results for WCET-Aware Procedure Cloning 77

4.3.7.1 WCET . . . 77
4.3.7.2 Code Size . . . 78
4.3.7.3 Optimization Run Time . . . 79
4.4 Superblock Optimizations . . . 79
4.4.1 Motivating Example . . . 80
4.4.2 Related Work . . . 81
4.4.3 WCET-Aware Source Code Superblock Formation . . . 82
4.4.3.1 Trace Selection . . . 83
4.4.3.2 Concepts of Superblocks . . . 85
4.4.3.3 Superblock Formation . . . 88
4.4.4 WCET-Aware Superblock Optimizations . . . 91
4.4.4.1 Static Program Analysis . . . 91
4.4.4.2 WCET-SB Common Subexpression Elimination . 92
4.4.4.3 WCET-SB Dead Code Elimination . . . 93
4.4.5 Experimental Results for WCET-Aware Superblock Optimizations . . . 94
4.4.5.1 WCET . . . 94
4.4.5.2 ACET . . . 96
4.4.5.3 Code Size . . . 97
4.4.5.4 Optimization Run Time . . . 97
4.5 Loop Unrolling . . . 97
4.5.1 Motivating Example . . . 98
4.5.2 Related Work . . . 99
4.5.3 Standard Loop Unrolling . . . 100
4.5.4 WCET-Aware Loop Unrolling . . . 102
4.5.4.1 Worst-Case Loop Iteration Counts . . . 102
4.5.4.2 I-Cache and Memory Constraints . . . 103
4.5.4.3 Prediction of Unrolling Effects . . . 105
4.5.4.4 Determination of Final Unrolling Factor . . . 107
4.5.4.5 WCET-Aware Unrolling Heuristics . . . 108
4.5.5 Experimental Results for WCET-Aware Loop Unrolling . . 109
4.5.5.1 WCET . . . 109
4.5.5.2 ACET . . . 111
4.5.5.3 Code Size . . . 112
4.5.5.4 Optimization Run Time . . . 113
4.6 Accelerating Optimization by the Invariant Path . . . 113
4.6.1 Motivating Example . . . 114
4.6.2 Related Work . . . 115
4.6.3 Invariant Path Paradigm . . . 116
4.6.3.1 *IF-THEN* Structure . . . 117
4.6.3.2 *IF-THEN-ELSE* Structure with Statically Evaluable Condition . . . 118
4.6.3.3 *IF-THEN-ELSE* Structure with Statically Non-evaluable Condition . . . 119
4.6.4 Construction of the Invariant Path . . . 119

4.6.5 Invariant Path Ratio . . . 120
4.6.6 Case Study: WCET-Aware Loop Unswitching . . . 121
4.6.7 Experimental Results for WCET-Aware Loop Unswitching 125
4.6.7.1 Optimization Run Time . . . 125
4.6.7.2 WCET . . . 126
4.6.7.3 Code Size . . . 127
4.7 Summary . . . 128

5 WCET-Aware Assembly Level Optimizations . . . 131
5.1 Introduction . . . 131
5.2 Existing Code Optimization Techniques . . . 132
5.3 Procedure Positioning . . . 133
5.3.1 Motivating Example . . . 134
5.3.2 Related Work . . . 136
5.3.3 Standard Procedure Positioning . . . 137
5.3.4 WCET-Centric Call Graph-Based Positioning . . . 138
5.3.4.1 Greedy WCET-Aware Positioning Approach . . . 139
5.3.4.2 Heuristic WCET-Aware Positioning Approach . . 142
5.3.5 Experimental Results for WCET-Aware Procedure Positioning . . . 142
5.3.5.1 WCET . . . 142
5.3.5.2 ACET . . . 144
5.3.5.3 Optimization Run Time . . . 144
5.4 Trace Scheduling . . . 145
5.4.1 Motivating Example . . . 145
5.4.2 Related Work . . . 147
5.4.3 Local Instruction Scheduling . . . 148
5.4.3.1 List Scheduling . . . 149
5.4.4 WCET-Aware Trace Scheduling . . . 151
5.4.5 Experimental Results for WCET-Aware Trace Scheduling . 154
5.4.5.1 WCET . . . 154
5.4.5.2 ACET . . . 155
5.4.5.3 Code Size . . . 156
5.4.5.4 Optimization Run Time . . . 156
5.5 Summary . . . 156

6 Machine Learning Techniques in Compiler Design . . . 159
6.1 Introduction . . . 159
6.2 Related Work . . . 161
6.3 Machine Learning Based Heuristic Generation . . . 162
6.3.1 Supervised Learning . . . 162
6.3.2 Heuristic Generation Based on Supervised Learning . . . 163
6.3.3 Integration and Use of MLB Heuristics in Compilers . . . 163
6.4 Function Inlining . . . 165
6.4.1 Motivating Example . . . 165

6.4.2 Standard Function Inlining . . . 166
6.4.2.1 Related Work . . . 167
6.4.3 MLB Heuristic Generation at Source Code Level . . . 168
6.4.3.1 Feature Extraction . . . 168
6.4.3.2 Label Determination . . . 169
6.4.3.3 Model Induction . . . 170
6.4.3.4 Application of WCET-Aware Function Inlining . 171
6.4.4 Experimental Results for WCET-Aware Function Inlining . 172
6.4.4.1 Accuracy of Classification . . . 172
6.4.4.2 Variable Importance Measure . . . 173
6.4.4.3 WCET . . . 174
6.4.4.4 ACET . . . 177
6.4.4.5 Code Size . . . 177
6.4.4.6 Compilation Run Time . . . 178
6.5 Loop-Invariant Code Motion . . . 178
6.5.1 Motivating Example . . . 179
6.5.2 Standard Loop-Invariant Code Motion . . . 180
6.5.3 MLB Heuristic Generation at Assembly Level . . . 182
6.5.3.1 Automatic Model Selection . . . 183
6.5.3.2 Feature Extraction . . . 187
6.5.3.3 Label Determination . . . 187
6.5.3.4 Application of WCET-Aware LICM . . . 188
6.5.4 Experimental Results for WCET-Aware LICM . . . 188
6.5.4.1 WCET . . . 189
6.5.4.2 ACET . . . 193
6.5.4.3 Compilation run time . . . 193
6.6 Summary . . . 194

7 Multi-objective Optimizations . . . 197
7.1 Introduction . . . 197
7.2 Motivation . . . 199
7.3 Related Work . . . 201
7.4 Compiler Optimization Sequence Exploration . . . 202
7.4.1 Adaptive Compilers . . . 203
7.4.2 Adaptability in WCC . . . 204
7.4.2.1 Internal Structure of WCC's Optimizer . . . 204
7.4.2.2 Available Compiler Optimizations . . . 205
7.4.3 Encoding of Optimization Sequences . . . 206
7.4.4 Objective Functions . . . 208
7.5 Multi-objective Exploration of Compiler Optimizations . . . 208
7.5.1 Multi-objective Optimization . . . 209
7.5.1.1 Pareto Front Approximation . . . 209
7.5.2 Evolutionary Multi-objective Optimization Algorithms . . 210
7.5.3 Statistical Performance Assessment . . . 211
7.5.3.1 Dominance Ranking . . . 214

7.5.3.2 Hypervolume Indicators . . . 214
7.5.3.3 Statistical Hypothesis Testing . . . 215
7.6 Experimental Results for Optimization Exploration . . . 216
7.6.1 Statistical Performance Assessment . . . 218
7.6.2 Analysis of Pareto Front Approximations . . . 220
7.6.3 Analysis of the Optimization Sequences . . . 222
7.6.4 Cross Validation . . . 223
7.6.5 Optimization Run Time . . . 225
7.7 Summary . . . 225

8 Summary and Future Work . . . 229
8.1 Research Contributions . . . 229
8.1.1 Extensions to WCC Framework . . . 230
8.1.2 WCET-Aware Source Code Level Optimizations . . . 230
8.1.3 WCET-Aware Assembly Level Optimizations . . . 231
8.2 Future Work . . . 233

Appendix A Abstract Interpretation . . . 235
A.1 Concrete Semantics . . . 235
A.2 Abstract Interpretation . . . 237
A.2.1 Abstract Semantics . . . 237
A.2.2 Abstract Domain . . . 238
A.2.3 Calculation of Abstract Semantics . . . 239

Appendix B Transformation of Conditions . . . 241

References . . . 243

Index . . . 257

http://www.springer.com/978-90-481-9928-0

Worst-Case Execution Time Aware Compilation Techniques for Real-Time Systems

Lokuciejewski, P.; Marwedel, P.

2011, XVIII, 262 p., Hardcover
ISBN: 978-90-481-9928-0