Contents

INTRODUCTION xiii

PART 1. PROLOG: OPTIMIZING COMPILATION 1

CHAPTER 1. ON THE DECIDABILITY OF PHASE ORDERING IN OPTIMIZING COMPILATION 3

1.1. Introduction to the phase ordering problem 3
1.2. Background on phase ordering 5
1.2.1. Performance modeling and prediction 5
1.2.2. Some attempts in phase ordering 6
1.3. Toward a theoretical model for the phase ordering problem 7
1.3.1. Decidability results 8
1.3.2. Another formulation of the phase ordering problem 11
1.4. Examples of decidable simplified cases 12
1.4.1. Models with compilation costs 12
1.4.2. One-pass generative compilers 13
1.5. Compiler optimization parameter space exploration 16
1.5.1. Toward a theoretical model 17
1.5.2. Examples of simplified decidable cases 19
1.6. Conclusion on phase ordering in optimizing compilation 20

PART 2. INSTRUCTION SCHEDULING 23

CHAPTER 2. INSTRUCTION SCHEDULING PROBLEMS AND OVERVIEW . 25

2.1. VLIW instruction scheduling problems 25
2.1.1. Instruction scheduling and register allocation in a code generator . 25
2.1.2. The block and pipeline VLIW instruction scheduling problems . . . 27
2.2. Software pipelining 29
2.2.1. Cyclic, periodic and pipeline scheduling problems 29

2.2.2. Modulo instruction scheduling problems and techniques 32
2.3. Instruction scheduling and register allocation 35
2.3.1. Register instruction scheduling problem solving approaches 35

CHAPTER 3. APPLICATIONS OF MACHINE SCHEDULING TO INSTRUCTION SCHEDULING 39

3.1. Advances in machine scheduling 39
3.1.1. Parallel machine scheduling problems 39
3.1.2. Parallel machine scheduling extensions and relaxations 41
3.2. List scheduling algorithms 43
3.2.1. List scheduling algorithms and list scheduling priorities 43
3.2.2. The scheduling algorithm of Leung, Palem and Pnueli 45
3.3. Time-indexed scheduling problem formulations 47
3.3.1. The non-preemptive time-indexed RCPSP formulation 47
3.3.2. Time-indexed formulation for the modulo RPISP 48

CHAPTER 4. INSTRUCTION SCHEDULING BEFORE REGISTER ALLOCATION 51

4.1. Instruction scheduling for an ILP processor: case of a VLIW architecture 51
4.1.1. Minimum cumulative register lifetime modulo scheduling 51
4.1.2. Resource modeling in instruction scheduling problems 54
4.1.3. The modulo insertion scheduling theorems 56
4.1.4. Insertion scheduling in a backend compiler 58
4.1.5. Example of an industrial production compiler from STMicroelectronics 60
4.1.6. Time-indexed formulation of the modulo RCISP 64
4.2. Large neighborhood search for the resource-constrained modulo scheduling problem 67
4.3. Resource-constrained modulo scheduling problem 68
4.3.1. Resource-constrained cyclic scheduling problems 68
4.3.2. Resource-constrained modulo scheduling problem statement 69
4.3.3. Solving resource-constrained modulo scheduling problems 70
4.4. Time-indexed integer programming formulations 71
4.4.1. The non-preemptive time-indexed RCPSP formulation 71
4.4.2. The classic modulo scheduling integer programming formulation . 72
4.4.3. A new time-indexed formulation for modulo scheduling 73
4.5. Large neighborhood search heuristic 74
4.5.1. Variables and constraints in time-indexed formulations 74
4.5.2. A large neighborhood search heuristic for modulo scheduling . . . 74
4.5.3. Experimental results with a production compiler 75
4.6. Summary and conclusions 76

CHAPTER 5. INSTRUCTION SCHEDULING AFTER REGISTER ALLOCATION 77

5.1. Introduction 77
5.2. Local instruction scheduling 79
5.2.1. Acyclic instruction scheduling 79
5.2.2. Scoreboard Scheduling principles 80
5.2.3. Scoreboard Scheduling implementation 82
5.3. Global instruction scheduling 84
5.3.1. Postpass inter-region scheduling 84
5.3.2. Inter-block Scoreboard Scheduling 86
5.3.3. Characterization of fixed points 87
5.4. Experimental results 87
5.5. Conclusions 89

CHAPTER 6. DEALING IN PRACTICE WITH MEMORY HIERARCHY EFFECTS AND INSTRUCTION LEVEL PARALLELISM 91

6.1. The problem of hardware memory disambiguation at runtime 92
6.1.1. Introduction 92
6.1.2. Related work 93
6.1.3. Experimental environment 94
6.1.4. Experimentation methodology 95
6.1.5. Precise experimental study of memory hierarchy performance . . . 95
6.1.6. The effectiveness of load/store vectorization 100
6.1.7. Conclusion on hardware memory disambiguation mechanisms . . . 103
6.2. Data preloading and prefetching 104
6.2.1. Introduction 104
6.2.2. Related work 105
6.2.3. Problems of optimizing cache effects at the instruction level 107
6.2.4. Target processor description 109
6.2.5. Our method of instruction-level code optimization 110
6.2.6. Experimental results 116
6.2.7. Conclusion on prefetching and preloading at instruction level . . . 117

PART 3. REGISTER OPTIMIZATION 119

CHAPTER 7. THE REGISTER NEED OF A FIXED INSTRUCTION SCHEDULE 121

7.1. Data dependence graph and processor model for register optimization 122
7.1.1. NUAL and UAL semantics 122
7.2. The acyclic register need 123
7.3. The periodic register need 125

7.3.1. Software pipelining, periodic scheduling and cyclic scheduling . . 125
7.3.2. The circular lifetime intervals 127
7.4. Computing the periodic register need 129
7.5. Some theoretical results on the periodic register need 132
7.5.1. Minimal periodic register need versus initiation interval 133
7.5.2. Computing the periodic register sufficiency 133
7.5.3. Stage scheduling under register constraints 134
7.6. Conclusion on the register requirement 139

CHAPTER 8. THE REGISTER SATURATION 141

8.1. Motivations on the register saturation concept 141
8.2. Computing the acyclic register saturation 144
8.2.1. Characterizing the register saturation 146
8.2.2. Efficient algorithmic heuristic for register saturation computation . 149
8.2.3. Experimental efficiency of Greedy-k 151
8.3. Computing the periodic register saturation 153
8.3.1. Basic integer linear variables 154
8.3.2. Integer linear constraints 154
8.3.3. Linear objective function 156
8.4. Conclusion on the register saturation 157

CHAPTER 9. SPILL CODE REDUCTION 159

9.1. Introduction to register constraints in software pipelining 159
9.2. Related work in periodic register allocation 160
9.3. SIRA: schedule independant register allocation 162
9.3.1. Reuse graphs 162
9.3.2. DDG associated with reuse graph 164
9.3.3. Exact SIRA with integer linear programming 166
9.3.4. SIRA with fixed reuse edges 168
9.4. SIRALINA: an efficient polynomial heuristic for SIRA 169
9.4.1. Integer variables for the linear problem 170
9.4.2. Step 1: the scheduling problem 170
9.4.3. Step 2: the linear assignment problem 172
9.5. Experimental results with SIRA 173
9.6. Conclusion on spill code reduction 175

CHAPTER 10. EXPLOITING THE REGISTER ACCESS
DELAYS BEFORE INSTRUCTION SCHEDULING 177

10.1. Introduction 177
10.2. Problem description of DDG circuits with non-positive distances . . . 179
10.3. Necessary and sufficient condition to avoid non-positive circuits . . . 180
10.4. Application to the SIRA framework 182

10.4.1. Recall on SIRALINA heuristic 183
10.4.2. Step 1: the scheduling problem for a fixed II 183
10.4.3. Step 2: the linear assignment problem 184
10.4.4. Eliminating non-positive circuits in SIRALINA 184
10.4.5. Updating reuse distances 186
10.5. Experimental results on eliminating non-positive circuits 187
10.6. Conclusion on non-positive circuit elimination 188

CHAPTER 11. LOOP UNROLLING DEGREE MINIMIZATION FOR PERIODIC REGISTER ALLOCATION 191

11.1. Introduction 191
11.2. Background 195
11.2.1. Loop unrolling after SWP with modulo variable expansion 196
11.2.2. Meeting graphs (MG) 197
11.2.3. SIRA, reuse graphs and loop unrolling 200
11.3. Problem description of unroll factor minimization for unscheduled loops 204
11.4. Algorithmic solution for unroll factor minimization: single register type 205
11.4.1. Fixed loop unrolling problem 206
11.4.2. Solution for the fixed loop unrolling problem 207
11.4.3. Solution for LCM-MIN problem 209
11.5. Unroll factor minimization in the presence of multiple register types 213
11.5.1. Search space for minimal kernel loop unrolling 217
11.5.2. Generalization of the fixed loop unrolling problem in the presence of multiple register types 218
11.5.3. Algorithmic solution for the loop unrolling minimization (LUM, problem 11.1) 219
11.6. Unroll factor reduction for already scheduled loops 221
11.6.1. Improving algorithm 11.4 (LCM-MIN) for the meeting graph framework 224
11.7. Experimental results 224
11.8. Related work 226
11.8.1. Rotating register files 226
11.8.2. Inserting `move` operations 227
11.8.3. Loop unrolling after software pipelining 228
11.8.4. Code generation for multidimensional loops 228
11.9. Conclusion on loop unroll degree minimization 228

PART 4. EPILOG: PERFORMANCE, OPEN PROBLEMS 231

CHAPTER 12. STATISTICAL PERFORMANCE ANALYSIS: THE SPEEDUP-TEST PROTOCOL 233

12.1. Code performance variation 233
12.2. Background and notations 236
12.3. Analyzing the statistical significance of the observed speedups 239
12.3.1. The speedup of the observed average execution time 239
12.3.2. The speedup of the observed median execution time, as well as individual runs 241
12.4. The Speedup-Test software 244
12.5. Evaluating the proportion of accelerated benchmarks by a confidence interval 246
12.6. Experiments and applications 248
12.6.1. Comparing the performances of compiler optimization levels 249
12.6.2. Testing the performances of parallel executions of OpenMP applications 250
12.6.3. Comparing the efficiency of two compilers 251
12.6.4. The impact of the Speedup-Test protocol over some observed speedups 253
12.7. Related work 253
12.7.1. Observing execution times variability 253
12.7.2. Program performance evaluation in presence of variability 254
12.8. Discussion and conclusion on the Speedup-Test protocol 255

CONCLUSION 257

APPENDIX 1. PRESENTATION OF THE BENCHMARKS USED IN OUR EXPERIMENTS 263

APPENDIX 2. REGISTER SATURATION COMPUTATION ON STAND-ALONE DDG 271

APPENDIX 3. EFFICIENCY OF SIRA ON THE BENCHMARKS 279

APPENDIX 4. EFFICIENCY OF NON-POSITIVE CIRCUIT ELIMINATION IN THE SIRA FRAMEWORK 293

APPENDIX 5. LOOP UNROLL DEGREE MINIMIZATION: EXPERIMENTAL RESULTS 303

APPENDIX 6. EXPERIMENTAL EFFICIENCY OF SOFTWARE DATA PRELOADING AND PREFETCHING FOR EMBEDDED VLIW 315

APPENDIX 7. APPENDIX OF THE SPEEDUP-TEST PROTOCOL 319

BIBLIOGRAPHY 327

LISTS OF FIGURES, TABLES AND ALGORITHMS 345

INDEX 353