- Foreword p. v
- Preface p. vii
- Acknowledgments p. ix
- Acronyms p. xix
- Part I Introduction and Basic Concepts
- 1 Introduction p. 1
- 1.1 Overview p. 1
- 1.2 Origins p. 2
- 1.3 Amdahl's Law and Structured Parallel Design p. 4
- 1.4 Introduction to PPF Systems p. 4
- 1.5 Conclusions p. 8
- Appendix p. 10
- A.1 Simple Design Example: The H.261 Decoder p. 10
- 2 Basic Concepts p. 17
- 2.1 Pipelined Processing p. 20
- 2.2 Pipeline Types p. 24
- 2.2.1 Asynchronous PPF p. 25
- 2.2.2 Synchronous PPF p. 26
- 2.3 Data Farming and Demand-based Scheduling p. 27
- 2.4 Data-farm Performance Criteria p. 28
- 2.5 Conclusion p. 30
- Appendix p. 31
- A.1 Short case studies p. 31
- 3 PPF in Practice p. 37
- 3.1 Application Overview p. 38
- 3.1.1 Implementation issues p. 39
- 3.2 Parallelization of the Postcode Recognizer p. 39
- 3.2.1 Partitioning the postcode recognizer p. 40
- 3.2.2 Scaling the postcode recognizer p. 41
- 3.2.3 Performance achieved p. 43
- 3.3 Parallelization of the address verifier p. 47
- 3.3.1 Partitioning the address verifier p. 47
- 3.3.2 Scaling the address verifier p. 49
- 3.3.3 Address verification farms p. 50
- 3.3.4 Overall performance achieved p. 50
- 3.4 Meeting the Specification p. 51
- 3.5 Conclusion p. 53
- Appendix p. 53
- A.1 Other Parallel Postcode Recognition Systems p. 53
- 4 Development of PPF Applications p. 57
- 4.1 Analysis Tools p. 58
- 4.2 Tool Characteristics p. 59
- 4.3 Development Cycle p. 60
- 4.4 Conclusion p. 62

- Part II Analysis and Partitioning of Sequential Applications
- 5 Initial Development of an Application p. 67
- 5.1 Confidence Building p. 67
- 5.2 Automatic and Semi-automatic Parallelization p. 69
- 5.3 Language Proliferation p. 71
- 5.4 Size of Applications p. 72
- 5.5 Semi-automatic Partitioning p. 73
- 5.6 Porting Code p. 75
- 5.7 Checking a Decomposition p. 77
- 5.8 Optimizing Compilers p. 77
- 5.9 Conclusion p. 79
- 6 Graphical Simulation and Performance Analysis of PPFs p. 81
- 6.1 Simulating Asynchronous Pipelines p. 82
- 6.2 Simulation Implementation p. 82
- 6.3 Graphical Representation p. 84
- 6.4 Display Features p. 88
- 6.5 Cross-architectural Comparison p. 89
- 6.6 Conclusion p. 93
- 7 Template-based Implementation p. 95
- 7.1 Template Design Principles p. 96
- 7.2 Implementation Choices p. 99
- 7.3 Parallel Logic Implementation p. 100
- 7.4 Target Machine Implementation p. 101
- 7.4.1 Common implementation issues p. 102
- 7.5 'NOW' Implementation for Logic Debugging p. 104
- 7.6 Target Machine Implementations for Performance Tuning p. 109
- 7.7 Patterns and Templates p. 112
- 7.8 Conclusion p. 113
- Part III Case Studies
- 8 Application Examples p. 117
- 8.1 Case Study 1: H.261 Encoder p. 118
- 8.1.1 Purpose of parallelization p. 119
- 8.1.2 'Per macroblock's quantization without motion estimation p. 119
- 8.1.3 'Per picture' quantization without motion estimation p. 123
- 8.1.4 'Per picture' quantization with motion estimation p. 125
- 8.1.5 Implementation of the parallel encoders p. 126
- 8.1.6 H.261 encoders without motion estimation p. 128
- 8.1.7 H.261 encoder with motion estimation p. 129
- 8.1.8 Edge data exchange p. 131
- 8.2 Case Study 2: H263 Encoder/Decoder p. 132
- 8.2.1 Static analysis of H.263 algorithm p. 134
- 8.2.2 Results from parallelizing H.263 p. 135
- 8.3 Case Study 3: 'Eigenfaces'--Face Detection p. 139
- 8.3.1 Background p. 139

- 8.3.2 Eigenfaces algorithm p. 140
- 8.3.3 Parallelization steps p. 141
- 8.3.4 Introduction of second and third farms p. 143
- 8.4 Case Study 4: Optical Flow p. 145
- 8.4.1 Optical flow p. 145
- 8.4.2 Existing sequential implementation p. 147
- 8.4.3 Gradient-based routine p. 147
- 8.4.4 Multi-resolution routine p. 150
- 8.4.5 Phase-based routine p. 154
- 8.4.6 LK results p. 156
- 8.4.7 Other methods p. 158
- 8.4.8 Evaluation p. 160
- 8.5 Conclusion p. 161
- 9 Design Studies p. 163
- 9.1 Case Study 1: Karhunen-Loeve Transform (KLT) p. 164
- 9.1.1 Applications of the KLT p. 164
- 9.1.2 Features of the KLT p. 165
- 9.1.3 Paralleization of the KLT p. 165
- 9.1.4 PPF parallelization p. 168
- 9.1.5 Implementation p. 171
- 9.2 Case Study 2: 2D-Wavelet Transform p. 171
- 9.2.1 Wavelet Transform p. 172
- 9.2.2 Computational algorithms p. 173
- 9.2.3 Parallel implementation of Discrete Wavelet Transform (DWT) p. 173
- 9.2.4 Parallel implementation of oversampled WT p. 176
- 9.3 Case Study 3: Vector Quantization p. 179
- 9.3.1 Parallelization of VQ p. 180
- 9.3.2 PPF schemes for VQ p. 181
- 9.3.3 VQ implementation p. 183
- 9.4 Conclusion p. 186
- 10 Counter Examples p. 189
- 10.1 Case Study 1: Large Vocabulary Continuous-Speech Recognition p. 190
- 10.1.1 Background p. 190
- 10.1.2 Static analysis of the LVCR system p. 191
- 10.1.3 Parallel design p. 193
- 10.1.4 Implementation on an SMP p. 195
- 10.2 Case Study 2: Model-based Coding p. 196
- 10.2.1 Parallelization of the model-based coder p. 196
- 10.2.2 Analysis of results p. 198
- 10.3 Case Study 3: Microphone Beam Array p. 202
- 10.3.1 Griffiths-Jim beam-former p. 202
- 10.3.2 Sequential implementation p. 203
- 10.3.3 Parallel implementation of the G-J Algorithm p. 204
- 10.4 Conclusion p. 206

- Part IV Underlying Theory and Analysis
- 11 Performance of PPFs p. 211
- 11.1 Naming Conventions p. 212
- 11.2 Performance Metrics p. 212
- 11.2.1 Order statistics p. 213
- 11.2.2 Asymptotic distribution p. 216
- 11.2.3 Characteristic maximum p. 217
- 11.2.4 Sample estimate p. 219
- 11.3 Gathering Performance Data p. 220
- 11.4 Performance Prediction Equations p. 221
- 11.5 Results p. 223
- 11.5.1 Prediction results p. 224
- 11.6 Simulation Results p. 225
- 11.7 Asynchronous Pipeline Estimate p. 227
- 11.8 Ordering Constraints p. 230
- 11.9 Task Scheduling p. 235
- 11.9.1 Uniform task size p. 236
- 11.9.2 Decreasing task size p. 236
- 11.9.3 Heuristic scheduling schemes p. 237
- 11.9.4 Validity of Factoring p. 238
- 11.10 Scheduling Results p. 238
- 11.10.1 Timings p. 238
- 11.10.2 Simulation results p. 240
- 11.11 Conclusion p. 241
- Appendix p. 242
- A.1 Outline derivation of Kruskal-Weiss prediction equation p. 242
- A.2 Factoring regime derivation p. 243
- 12 Instrumentation of Templates p. 247
- 12.1 Global Time p. 248
- 12.2 Processor Model p. 249
- 12.3 Local Clock Requirements p. 249
- 12.4 Steady-state Behavior p. 250
- 12.5 Establishing a Refresh Interval p. 253
- 12.6 Local Clock Adjustment p. 256
- 12.7 Implementation on the Paramid p. 257
- 12.8 Conclusion p. 259
- Part V Future Trends
- 13 Future Trends p. 263
- 13.1 Designing for Differing Embedded Hardware p. 265
- 13.2 Adapting to Mobile Networked Computation p. 265
- 13.3 Conclusion p. 267
- References p. 269
- Index p. 299