Contents

1 Introduction 1
- 1.1 Aim of this Thesis . . . 2
- 1.2 Organisation of the Work . . . 3

2 Anatomy of a Device Driver 5
- 2.1 The Driver Inside the Operating System . . . 5
- 2.2 Resource Management . . . 8
  - 2.2.1 Device Identification . . . 8
  - 2.2.2 Communication Channel Mapping . . . 9
- 2.3 Control-Flow . . . 11
  - 2.3.1 User Process and Driver Interaction . . . 12
  - 2.3.2 Device and Driver Interaction . . . 12
- 2.4 Access to the Device Hardware . . . 16
- 2.5 Development Process and Lifeline of a Device Driver . . . 17
- 2.6 Problems in Device Driver Design . . . 18
- 2.7 Summary . . . 19

3 State of the Art 21
- 3.1 Books on Device Driver Design . . . 21
- 3.2 Driver Algorithm . . . 22
  - 3.2.1 Portable Driver Design . . . 22
  - 3.2.2 Domain Specific Languages . . . 22
  - 3.2.3 Automatic Interface Adaptor Synthesis . . . 23
- 3.3 Hardware Abstraction Layer . . . 24
- 3.4 Integrated Solutions . . . 25
  - 3.4.1 Hardware/Software Co-design . . . 25
  - 3.4.2 Code Generators and Software Development Kits . . . 26
- 3.5 Component/Object Oriented Solutions . . . 27
  - 3.5.1 Object Oriented Operating Systems . . . 27
  - 3.5.2 Java and JINI . . . 28
  - 3.5.3 Programming Languages . . . 28
  - 3.5.4 Object Orientation, UML and Design Patterns . . . 29
- 3.6 Summary . . . 30

4 Approach for Device Driver Design 33
- 4.1 Intercomponent Communication . . . 34
- 4.2 Coarse Grained Driver Structure . . . 38
  - 4.2.1 Driver Object Structure . . . 38
  - 4.2.2 Device Identification and Driver Structure set up . . . 40

4.2.3 Influence of the Operating System Channel . . . 44
4.3 Example . . . 46
4.4 Summary . . . 49

5 Generic Hardware Abstraction Layer **51**
5.1 Problem Analysis . . . 52
5.2 System Architecture Model . . . 53
5.3 Behaviour Modelling . . . 56
5.3.1 Attributed Grammar . . . 57
5.3.2 Register File Model . . . 59
5.4 Code Generation . . . 60
5.4.1 Grammar Construction . . . 60
5.4.2 Start String Generation . . . 61
5.4.3 Grammar Evaluation . . . 62
5.5 Case Study . . . 64
5.6 Optimisations . . . 68
5.6.1 Combining of Accesses . . . 68
5.6.2 Explicitly Caching . . . 70
5.7 Restrictions and Extensions of the Method . . . 71
5.8 Summary . . . 72

6 Deriving the Driver Algorithm **75**
6.1 Problem Analysis . . . 76
6.2 Structural Reflections . . . 77
6.2.1 Modelling and Method . . . 78
6.2.2 Example CAN-Controller . . . 79
6.2.3 Example Interrupt Systems . . . 80
6.2.4 Summary . . . 84
6.3 Classification of FSM elements . . . 84
6.3.1 Splitting of Automata . . . 84
6.3.2 Split of Data-paths . . . 87
6.3.3 Conclusions for Device Driver Design . . . 88
6.4 Control-Flow Analysis with Model Checker . . . 89
6.4.1 Modelling . . . 90
6.4.2 Use of Model Checker Results . . . 91
6.5 Summary . . . 94

7 Design Flow and Tool Integration **97**
7.1 Design Flow for Driver Design . . . 97
7.2 Tool Integration . . . 100

8 Advanced Topics **103**
8.1 Applying Compiler Techniques . . . 103
8.1.1 Techniques and Pre-conditions . . . 103
8.1.2 Traffic Optimisation . . . 104
8.1.3 Dead Code Elimination in Hardware . . . 105
8.1.4 Summary . . . 105
8.2 System Security . . . 105

9 Conclusion 109

10 Bibliography 111

A Graphic Representations of Interrupt Systems 121

B RABBIT Interrupt System 127