Contents

List of publications iii

Contents v

List of Figures ix

List of Tables xiii

I Introduction 1

1 Introduction to hard real-time systems 5

1.1 Introduction 5

1.2 Real-time operating system 6

1.3 Processing elements 7

1.3.1 Field Programmable Gate Array 9

1.4 Hardware considerations 10

2 Common notions and models 13

2.1 Time model 13

2.2 Application model 13

2.3 Hardware model 17

2.3.1 Processor model 17

2.3.2 Platform model 18

2.3.3 Reconfigurable processors 21

2.3.4 Clustered platforms 22

2.4 Scheduler 24

2.5 Multi-mode application 32
2.5.1 Mode model 34
2.5.2 Protocol model 37
2.6 Memory considerations 37

3 Motivation and organisation **41**
3.1 Motivation 41
3.2 Related work 42
3.3 Outline of the thesis 44

II Global scheduling **45**

4 Introduction to Global scheduling on heterogeneous unrelated platform **49**
4.1 Motivation 49
4.2 Seminal model 49
4.3 Related works 50
4.4 Organisation and contributions 55

5 Workload assignment **59**
5.1 New model 59
5.1.1 Motivation 59
5.1.2 Empirical measurements 60
5.1.3 Model 62
5.2 Designing new LPs 62
5.3 LP-Feas and LP-CFeas 67
5.4 LP-Load and LP-CLoad 68
5.5 Minimal number of presences: ILP-CMig 69
5.6 Workload assignment evaluation 70
5.6.1 Experimental setup 70
5.6.2 Inter-cluster number of presences in excess 71
5.6.3 Run-time measurement 73

6 Flaw & correction in the schedule construction of [19, 29] **75**
6.1 Seminal algorithm from [19, 29] 76
6.2 Counter-example of the seminal algorithm 77
6.3 Correction of the matching algorithm 78

6.4 Proof of correctness of the algorithm . . . 80

7 Schedule construction optimisation 83

7.1 Pre-optimisation: minimising the number of schedule points . . . 83
7.2 LBAP experiments . . . 84
7.3 Post-optimisation: Reordering the template schedule . . . 87
7.4 TSP experiments . . . 89

8 Flaw in the sporadic scheduler of [29] 91

8.1 Seminal algorithm . . . 91
8.1.1 Seminal model . . . 91
8.1.2 Seminal algorithm offline phase . . . 91
8.1.3 Seminal algorithm run-time phase . . . 92
8.2 Counter-example . . . 92
8.2.1 Seminal offline phase . . . 93
8.2.2 Seminal run-time phase . . . 93

9 Conclusion 97

III Multi-mode applications 99

10 Introduction to multi-mode applications 103

10.1 Contributions and organisation . . . 104

11 Introducing a new multi-mode application model 107

11.1 Hardware model . . . 107
11.1.1 Clustered platforms . . . 108
11.2 Software model . . . 108
11.3 Multi-mode model . . . 109
11.4 Model example . . . 111

12 A first protocol for multi-mode applications: ACCEPTOR 117

12.1 Scheduling problem . . . 118
12.2 ACCEPTOR . . . 118
12.2.1 Offline phase: computing the required reconfigurations . . . 119
12.2.2 Run-time: scheduling and reconfiguring . . . 121
12.2.3 Note on the offline phase . . . 121

12.2.4 Mode change phase example 121
12.3 The upper-bound $\overline{\text{reconfigured}}$ 123
12.4 Validity test 124
12.5 Evaluation: Time complexity 126
12.6 Evaluation: empirical pessimism of $\overline{\text{reconfigured}}$ 127
12.7 Evaluation: Competitive analysis of ACCEPTOR 128
12.7.1 Preliminary definitions and notations 129
12.7.2 Competitive analysis 130
12.8 Handling mode independent tasks 134
12.9 Improving the upper-bound $\overline{\text{reconfigured}}$ 134
12.9.1 Changing the idle upper-bound 135
12.9.2 Time complexity 135
12.9.3 Evaluation 136
12.9.4 Last words about $\overline{\text{reconfigured}'}$ 137

13 SQUARER **139**
13.1 Protocol SQUARER description 140
13.1.1 Offline phase presentation 142
13.1.2 Run-time phase 143
13.1.3 Preventing deadline misses 144
13.2 Upper-bound and validity test 146
13.3 Execution time 149
13.4 Empirical performances evaluation of $\overline{\text{reconfigured}''}$ 149

14 Conclusion **153**

IV General conclusion and perspectives **155**

A Multi-mode protocol algorithms **165**
A.1 ACCEPTOR algorithm 165
A.2 SQUARER algorithm 165

Index **167**

Bibliography **169**

List of Figures

1.1 Layer hierarchy of an embedded system 7
1.2 A dual-core, dual-processor system 8
1.3 The Zynq UltraScale+™ EG processor block diagram. EG devices feature a quad-core ARM® Cortex-A53 platform running up to 1.5GHz, combined with dual-core Cortex-R5 real-time processors, a Mali-400 MP2 graphics processing unit, and a 16nm FinFET+ programmable logic [1]. 10

2.1 Visual representation of task utilisation 20
2.2 Two possible divisions of the example platform (based on the Zynq Ultra-Scale+™) into several clusters. 23
2.3 Example of a scheduling pattern 25
2.4 Visual representation of job preemptions and migrations 26
2.5 Visual representation of partitioned and global schedules 28
2.6 Visual representation of a clustered schedule 30
2.7 Visual representation of the idle instant and makespan 31
2.8 Multi-mode illustration: different functionalities of a plane 33
2.9 Multi-mode illustration: processor utilisation of the different functionalities 33
2.10 Graph transition example 35
2.11 Mode transition illustration 36
2.12 Example of a memory scheme 38
2.13 Notation summary 39

4.1 Step by step schedule construction 52
4.2 Resulting template schedule for the guideline example 55

5.1 Illustration of flat versus clustered platform model 60

5.2 Effect of migrations on execution time distribution 61
5.3 Proof sketch for Theorem 5.1 64
5.4 Rectangle schedule computed from LP-CFeas versus schedule favouring fast cores utilisation computed from LP-CLoad 68
5.5 Number of presences by workload assignment method for unrelated and consistent clusters 72

6.1 Bipartite graph corresponding to the workload assignment matrix at time $t = 1$. 76
6.2 Illustration of the application of the cleaning algorithm in [19, 29] 78
6.3 Illustration of the proposed matching algorithm 79
6.4 Illustration of the proposed matching algorithm 81
6.5 Illustration of two possible types of connected components 81

7.1 Results for LBAP ($\dot{m} = 2, 5$) 85
7.2 Example of LBAP pitfall 86
7.3 Non-optimised template schedule 87
7.4 Template schedule optimised with TSP 88
7.5 Scheduling windows through a TSP 88
7.6 TSP ($\dot{m} = 2$) 88
7.7 TSP ($\dot{m} = 5$) 88

8.1 Counter-example task set 93
8.2 Counter-example template schedule 93
8.3 Counter-example: reservation at $t = 0$ 94
8.4 Counter-example: reservation at $t = 1$ 95

11.1 Graph transition example 110
11.2 Notation summary 114
11.3 The Zynq UltraScale+™ EG processor block diagram. EG devices feature a quad-core ARM® Cortex-A53 platform running up to 1.5GHz, combined with dual-core Cortex-R5 real-time processors, a Mali-400 MP2 graphics processing unit, and a 16nm FinFET+ programmable logic [1]. 115

12.1 ACCEPTOR protocol illustration 122
12.2 Pessimism of $\overline{\text{reconfigured}}$ 127
12.3 A *squeezable* system with $m' = 3$ and $n = 5$ 131
12.4 Average ratio $\frac{\overline{\text{reconfigured}'}}{\overline{\text{reconfigured}}}$ 136

13.1 Same system with two different protocols 140
13.2 Illustration of non-parallel idle time 145
13.3 Pessimism comparison between $\overline{\text{reconfigured}}$ and $\overline{\text{reconfigured}''}$ 150
13.4 Total idle comparison between $\overline{\text{reconfigured}}$ and $\overline{\text{reconfigured}''}$ 151

A.1 ACCEPTOR run-time algorithm 166
A.2 SQUARER run-time algorithm 166