

# Contents

<b>1. Introduction</b> .....	1
1.1 Analysis of Parallel Discrete Systems .....	3
1.2 Preliminary Remarks .....	5
1.3 The Scope of the Book .....	6
<b>2. Main Notions, Problems and Methods</b> .....	9
2.1 Models and Specifications of Parallel Discrete Systems .....	9
2.1.1 Parallel Discrete Systems (General Definition) .....	9
2.1.2 Petri Nets and Their Extensions .....	10
2.1.3 Parallel and Sequent Automata .....	16
2.1.4 Sequential Function Charts .....	17
2.1.5 Statecharts .....	17
2.1.6 Finite State Machines and FSM Networks .....	20
2.2 The Tasks of Analysis .....	21
2.3 The Methods of Analysis .....	23
<b>3. Reduced Reachability Graphs</b> .....	27
3.1 Review of Known Methods .....	27
3.1.1 Persistent Set Methods .....	27
3.1.2 Other Methods .....	32
3.2 A Generalization of Stubborn Set Method .....	33
3.3 Weak Persistent Sets .....	34
3.4 On Combining the Persistent Set Approach and Concurrent Simulation .....	37
3.4.1 Concurrent Simulation and Persistent Sets .....	37
3.4.2 Comparison with the Janicki-Koutny's Method .....	39
3.4.3 Conflicts, State Explosion and Decomposition .....	41
3.5 Analysis of Special Classes of Petri Nets .....	42
3.5.1 Properties and Analysis of $\alpha$ -Nets .....	42
3.5.2 A Hypothesis on EFC-Nets .....	53
3.5.3 Analysis of s-Nets .....	53
3.6 Minimization of Space .....	56

3.6.1	Dynamic Reduction of Reachability Graphs .....	57
3.6.2	Reducing the Space for Various Analysis Tasks .....	59
3.6.3	Example .....	60
3.6.4	Conclusive Notes on the Method .....	61
<b>4.</b>	<b>Decomposition for Analysis .....</b>	<b>63</b>
4.1	Block Decomposition .....	63
4.1.1	Operational Petri Nets .....	63
4.1.2	Analysis of Operational Petri Nets .....	64
4.1.3	Analysis of a Class of Cyclic Nets .....	66
4.1.4	Example and Experimental Results .....	67
4.2	Hierarchical Decomposition .....	69
4.2.1	A Conception of Hierarchical Decomposition of Petri Nets .....	70
4.2.2	Properties of P-Decomposition .....	74
4.2.3	Finding P-Blocks .....	75
4.3	Decomposition and Persistent Sets .....	78
4.4	Parallel Analysis .....	79
4.5	Distributed Analysis .....	82
4.5.1	A Method of Distributed Analysis .....	82
4.5.2	Implementation of the Method .....	84
4.5.3	Experimental Results and Concluding Remarks .....	85
<b>5.</b>	<b>Analysis by Solving Logical Equations – Calculation of Siphons and Traps .....</b>	<b>87</b>
5.1	Known Methods of Calculation of Siphons and Traps .....	88
5.1.1	Calculation of Siphons and Traps by Means of Solving Logical Equations .....	88
5.1.2	Other Approaches to Calculation of Siphons and Traps .....	89
5.2	Algorithm to Find Siphons and Traps .....	90
5.3	Example .....	91
5.3.1	The Proposed Method .....	91
5.3.2	Some Other Symbolic Methods .....	92
5.3.3	The Linear Algebraic Method .....	92
5.4	Concluding Remarks .....	93
<b>6.</b>	<b>Verification of Detailed System Descriptions .....</b>	<b>95</b>
6.1	Application of the Described Approaches to Other Parallel Discrete Models .....	95
6.1.1	Interpreted Petri Nets and Sequent Automata .....	95
6.1.2	Statecharts .....	102
6.1.3	FSM Networks .....	108
6.2	Verification of Parallel Automata Implementation .....	113
6.2.1	Testing Approach .....	113
6.2.2	Analytical Approach .....	117

**7. Conclusion** ..... 123

**Acknowledgments** ..... 127

**A. A Theorem on the Stubborn Set Method** ..... 129

**B. Decyclization of the Oriented Graphs** ..... 131

**C. Intersecting P-Blocks** ..... 135

**D. Improvements of Thelen's Prime Implicant Method** ..... 137

    D.1 Introduction ..... 137

    D.2 Thelen's Method ..... 138

    D.3 Heuristics for Thelen's Method ..... 139

**References** ..... 145

**Index** ..... 161

and structures. For that reason practically all modern hardware design languages and formalisms used for system specifications (such as VHDL, Verilog, etc.) allow describing concurrency. Design of complex, VLSI-based electronic devices is possible only with the help of CAD systems, so the design and verification methods have to be (and mostly are) formalized. Formalization and automation of system design requires developing of formal models for parallel discrete systems and low-level description languages based on those models.

Specifications of devices and systems described in VHDL, Verilog or other popular languages of logical control, as LD, IL or ST [159], are very difficult for formal verification, because it is practically impossible to create adequate and at the same time simple formal models for such specifications (if these languages are used without restrictions). The problem can be solved by using of restricted specifications based on models which are easy to analyze and have enough expressive power.

There are two main directions of developing such models, each having its good and bad aspects. Both of them are, in a sense, extensions of the finite state machines (FSM) - the basic model of sequential discrete devices, which is, of course, in its "pure" version not convenient for practical needs of specifying of complex systems.

One direction is the composition of FSMs in various ways. The simplest representation of this approach is the FSM network - a system of communicating automata [23, 147]. Studies on the automata networks have started in 1960-s, but rapid development of the methods of behavior specification by means of such networks began in 1980-s. Adding hierarchy to FSM networks leads to obtaining the model known as HCFSM (Hierarchical Concurrent Finite State Machines) [71]. One of the most popular and well-adapted to HCFSM languages has been developed within a frame of the universal specification language UML (Unified Modeling Language [209]), describing hierarchical objects and dependencies between them. We talk about the Statecharts, invented by D. Harel [56, 83]. There exist several other models and languages based on automata networks such as