eXamen.press

eXamen.press ist eine Reihe, die Theorie und Praxis aus allen Bereichen der Informatik für die Hochschulausbildung vermittelt.

Christian Haubelt · Jürgen Teich

Digitale Hardware/ Software-Systeme

Spezifikation und Verifikation

Springer

Christian Haubelt
Universität Erlangen-Nürnberg
Lehrstuhl Hardware-Software-Co-Design
Am Weichselgarten 3
91058 Erlangen
Deutschland
haubelt@informatik.uni-erlangen.de

Jürgen Teich
Universität Erlangen-Nürnberg
Lehrstuhl Hardware-Software-Co-Design
Am Weichselgarten 3
91058 Erlangen
Deutschland
teich@informatik.uni-erlangen.de

ISSN 1614-5216
ISBN 978-3-642-05355-9 e-ISBN 978-3-642-05356-6
DOI 10.1007/978-3-642-05356-6
Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

© Springer-Verlag Berlin Heidelberg 2010
Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.
Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Einbandentwurf: KuenkelLopka GmbH

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort

Getrieben durch neue Technologien und Anwendungen wird der Entwurf eingebetteter Systeme zunehmend komplexer. Dabei ist eine Umsetzung als Hardware/Software-System heutzutage der Stand der Technik. Die Minimierung von Fehlern im Entwurf dieser Systeme ist aufgrund deren Komplexität eine der zentralen Herausforderungen unserer heutigen Zeit. Bereits heute wird mehr Aufwand in die Verifikation, also in die Überprüfung der Korrektheit, eines eingebetteten Systems gesteckt als in den eigentlichen Entwurf. Um so erstaunlicher ist es, dass das Thema Verifikation eingebetteter Systeme in der Ausbildung und Lehre nach wie vor keinen entsprechenden Stellenwert besitzt. Ein überwiegender Anteil der Lehrveranstaltungen zu diesem Thema konzentriert sich dabei ausschließlich auf die Verifikation von Hardware oder Software. Aber gerade das Zusammenspiel beider Bestandteile garantiert die Realisierbarkeit komplexer Systeme und bedarf selbstverständlich, wie andere Aspekte, auch der Verifikation.

Dieses Lehrbuch widmet sich der Verifikation digitaler Hardware/Software-Systeme. Es ist als zweiter Band zu dem Buch „Digitale Hardware/Software-Systeme – Synthese und Optimierung“ [426] konzipiert mit dem Ziel, grundlegende Methoden und Verfahren zur Verifikation digitaler Hardware, Software, aber auch von ganzen Systemen zu vermitteln. Der Leser wird damit in die Lage versetzt, die Komplexität und Grenzen der Verifikation zu verstehen sowie Verifikation durchzuführen und dabei ihre Effektivität einzuschätzen. Dies ist die Voraussetzung, um im praktischen Einsatz geeignete Verifikationsziele zu setzen, passende Verifikationsmethoden auszuwählen sowie die damit verbundenen Risiken abzuschätzen.

Im Gegensatz zu vielen anderen Lehrbüchern, welche oftmals den Schwerpunkt auf den Aspekt Hardware oder Software legen, zielt der vorliegende Band darauf ab, dem Leser genau die notwendigen Methoden an die Hand zu geben, um moderne eingebettete Systeme bestehend aus Hardware- und Software-Komponenten auf ihren korrekten Entwurf hin zu überprüfen. Hierzu gehört sowohl die funktionale Verifikation als auch die Verifikation des Zeitverhaltens. Der Stoff ist Lehrinhalt von Vorlesungen, die teilweise seit mehreren Jahren vom Lehrstuhl für Hardware-Software-Co-Design angeboten und mit großer Resonanz von den Studierenden der Friedrich-Alexander-Universität Erlangen-Nürnberg angenommen werden.

Beim Lesen dieses Buches kann man feststellen, dass Verifikationsmethoden für Hardware und Software auf identischen Prinzipien aufsetzen und ähnliche Lösungen hervorbringen. Dies ist eine analoge Erkenntnis zu der im ersten Band aufgezeigten Dualität für den Entwurf von Hardware und Software. Als gemeinsames Modell für beide Bände dient deshalb das Doppeldachmodell, welches den idealisierten Entwurfsfluss für digitale Hardware/Software-Systeme darstellt. In diesem Band wird ein Doppeldachmodell für den Verifikationsprozess vorgestellt, das zur Definition grundlegender Verifikationsaufgaben in der Entwicklung von digitalen Hardware/Software-Systemen dient. Die verwendeten Modellierungsansätze für digitale Systeme, namentlich Petri-Netze, endliche Automaten, Datenflussmodelle sowie ausgewählte heterogene Modelle, bilden eine weitere Gemeinsamkeit beider Bände. Eine dritte Gemeinsamkeit, die beide Bände verbindet, ist eine Entwurfsumgebung mit dem Namen SystemCoDesigner, welche als aktuelles Forschungsprojekt am Lehrstuhl für Hardware-Software-Co-Design an der Universität Erlangen-Nürnberg entwickelt wird. SystemCoDesigner setzt die im Band Synthese und Optimierung beschriebenen Synthese- und Optimierungsverfahren um und unterstützt die in diesem Band beschriebenen Spezifikations- und Verifikationsmethodiken.

Die Autoren möchten Ihren Dank den anonymen Gutachtern des Springer-Verlags aussprechen, die maßgeblich geholfen haben, die Idee zu diesem Buch zu fokussieren. Darüber hinaus möchten wir uns bei den wissenschaftlichen Mitarbeitern bedanken, die durch ihre Ideen und ihr Mitwirken geholfen haben, die Entwurfsumgebung SystemCoDesigner zu verwirklichen. Insbesondere möchten wir uns bei Dipl.-Inf. Michael Glaß, Dipl.-Inf. Martin Streubühr und Dipl.-Inf. Christian Zebelein bedanken, die durch ihre zahlreichen Vorschläge geholfen haben, das vorliegende Buch zu verbessern. Unserer besonderer Dank gilt Prof. Dr. rer. nat. Rolf Wanka, der uns in langen Diskussionen geholfen hat, Ergebnisse aus dem Bereich der Theoretischen Informatik zu interpretieren. Schließlich möchten wir uns bei Dipl.-Inf. Jens Gladigau bedanken, der durch seine Kommentare und Anregungen maßgeblich zum Gelingen dieses Buches beigetragen hat.

Erlangen, im Frühjahr 2010 C. Haubelt und J. Teich

Inhaltsverzeichnis

1 Einleitung 1
1.1 Motivation 1
1.2 Der Verifikationsprozess 11
1.2.1 Das V-Modell 13
1.2.2 Das Doppeldachmodell des Entwurfsprozesses 14
1.2.3 Das Doppeldachmodell des Verifikationsprozesses 18
1.3 Eine kurze Geschichte der Verifikation 22
1.4 Beispiele 29
1.5 Ausblick 34
1.6 Literaturhinweise 34

2 Spezifikation digitaler Systeme 37
2.1 Wie spezifiziert man ein System? 37
2.2 Formale Verhaltensmodelle 41
2.2.1 Petri-Netze 41
2.2.2 Endliche Automaten 47
2.2.3 Datenflussgraphen 51
2.2.4 Heterogene Modelle 56
2.3 Ausführbare Verhaltensmodelle 59
2.3.1 SystemC 60
2.3.2 SysteMoC 68
2.4 Formale Spezifikation funktionaler Anforderungen 72
2.4.1 Temporale Strukturen 73
2.4.2 Temporale Aussagenlogik 75
2.4.3 Die Zusicherungssprache PSL 83
2.5 Formale Spezifikation nichtfunktionaler Anforderungen 88
2.6 Literaturhinweise 91

3 Verifikation 95
3.1 Verifikationsaufgabe, -ziel und -methode 95
3.1.1 Verifikationsziel 97

3.1.2 Verifikationsmethode 99
3.2 Beobachtbarkeit und Steuerbarkeit 107
3.3 Gesteuerte zufällige Simulation 111

4 Äquivalenzprüfung 115
4.1 Implizite Äquivalenzprüfung 117
4.1.1 Kanonische Funktionsrepräsentationen 117
4.1.2 Taylor-Expansions-Diagramme 120
4.1.3 Reduktion und Normalisierung von TEDs 120
4.1.4 Kanonizität von TEDs 124
4.1.5 Implizite Äquivalenzprüfung mit TEDs 125
4.2 Explizite Äquivalenzprüfung 129
4.2.1 Regressionstest 131
4.2.2 Bereichstest 133
4.2.3 Pfadbereichstest 134
4.2.4 Fehleroffenbarende Unterbereiche 139
4.3 Sequentielle Äquivalenzprüfung 141
4.3.1 Automaten-Äquivalenz 142
4.3.2 Zustandsraumtraversierung 144
4.3.3 Symbolische Zustandsraumtraversierung 148
4.3.4 Erzeugung von Gegenbeispielen 151
4.4 Strukturelle Äquivalenzprüfung 152
4.5 Literaturhinweise 154

5 Eigenschaftsprüfung 155
5.1 Prüfung funktionaler Eigenschaften 156
5.1.1 Eigenschaftsprüfung auf Erreichbarkeitsgraphen 158
5.1.2 Strukturelle Eigenschaftsprüfung von Petri-Netzen 167
5.1.3 Partialordnungsreduktion 172
5.2 Explizite Modellprüfung 178
5.2.1 CTL-Modellprüfung 179
5.2.2 LTL-Modellprüfung 185
5.2.3 Zusicherungsbasierte Eigenschaftsprüfung 188
5.3 Symbolische Modellprüfung 197
5.3.1 BDD-basierte CTL-Modellprüfung 197
5.3.2 SAT-basierte Modellprüfung 199
5.4 Prüfung nichtfunktionaler Eigenschaften 207
5.4.1 Zeitbehaftete Petri-Netze 207
5.4.2 Zeitbehaftete Automaten 214
5.4.3 Zeitbehaftete SDF-Graphen 222
5.5 Literaturhinweise 232

6 Hardware-Verifikation 235
6.1 Äquivalenzprüfung kombinatorischer und sequentieller Schaltungen 236
6.1.1 Implizite Äquivalenzprüfung auf der Logikebene 236
6.1.2 Explizite Äquivalenzprüfung auf der Logikebene 246
6.1.3 Formale explizite Äquivalenzprüfung von Schaltwerken 258
6.1.4 Strukturelle Äquivalenzprüfung auf der Logikebene 263
6.2 Äquivalenzprüfung arithmetischer Schaltungen 273
6.2.1 Implizite Äquivalenzprüfung auf der Architekturebene 273
6.2.2 Äquivalenzprüfung zwischen Architektur- und Logikebene . 280
6.2.3 Äquivalenzprüfung auf der Architekturebene 283
6.3 Formale Verifikation von Prozessoren 291
6.3.1 Äquivalenzprüfung für Prozessoren mit Fließbandverarbeitung 293
6.3.2 Berücksichtigung von Multizyklen-Funktionseinheiten, Ausnahmebehandlung und Sprungvorhersage 308
6.3.3 Äquivalenzprüfung für Prozessoren mit dynamischer Instruktionsablaufplanung 313
6.4 Funktionale Eigenschaftsprüfung 323
6.4.1 Zusicherungsbasierte Eigenschaftsprüfung 323
6.4.2 SAT-basierte Modellprüfung 331
6.5 Zeitanalyse 345
6.5.1 Zeitanalyse synchroner Schaltungen 345
6.5.2 Zeitanalyse latenzinsensitiver Systeme 351
6.6 Literaturhinweise 356

7 Software-Verifikation 361
7.1 Formale Äquivalenzprüfung eingebetteter Software 362
7.1.1 Äquivalenzprüfung von Assemblerprogrammen 362
7.1.2 Strukturelle Äquivalenzprüfung von Assemblerprogrammen 368
7.1.3 Äquivalenzprüfung von C-Programmen 373
7.2 Testfallgenerierung zur simulativen Eigenschaftsprüfung 391
7.2.1 Funktionsorientierte Testfälle 391
7.2.2 Kontrollflussorientierte Testfälle 400
7.2.3 Datenflussorientierte Testfälle 410
7.3 Formale funktionale Eigenschaftsprüfung von Programmen 416
7.3.1 Statische Programmanalyse 416
7.3.2 SAT-basierte Modellprüfung von C-Programmen 422
7.3.3 Modellprüfung durch Abstraktionsverfeinerung 425
7.4 Zeitanalyse 431
7.4.1 BCET- und WCET-Analyse 432
7.4.2 Echtzeitanalyse für Einprozessorsysteme 438
7.5 Literaturhinweise 448

8 Systemverifikation 451
8.1 Funktionale Eigenschaftsprüfung von SystemC-Modellen 452
8.1.1 Symbolische CTL-Modellprüfung von SysteMoC-Modellen 452
8.1.2 Modellprüfung von SystemC-Modellen 466
8.1.3 Formale Modellprüfung von Transaktionsebenenmodellen . . 476
8.1.4 Zusicherungsbasierte Eigenschaftsprüfung für Transaktionsebenenmodelle 484
8.2 Zeitanalyse auf Systemebene 490
8.2.1 Simulative Zeitbewertung 492
8.2.2 Kompositionale Zeitanalyse über Ereignisströme 499
8.2.3 Modulare Zeitanalyse mit RTC 508
8.3 Literaturhinweise 520

Anhang 523

Notation 523
A.1 Mengen 523
A.2 Relationen und Funktionen 524
A.3 Aussagenlogik 527
A.4 Prädikatenlogik erster Ordnung 528
A.5 Graphen 529

Binäre Entscheidungsdiagramme 533
B.1 Entscheidungsdiagramme 533
B.2 Binäre Entscheidungsdiagramme 534
B.3 Verallgemeinerte binäre Entscheidungsdiagramme 537

Algorithmen 541
C.1 Klassifikation von Algorithmen 541
C.2 SAT-Solver 542
C.3 SMT-Solver 551
C.4 CTL-Fixpunktberechnung 556

Literatur 561

Sachverzeichnis 587