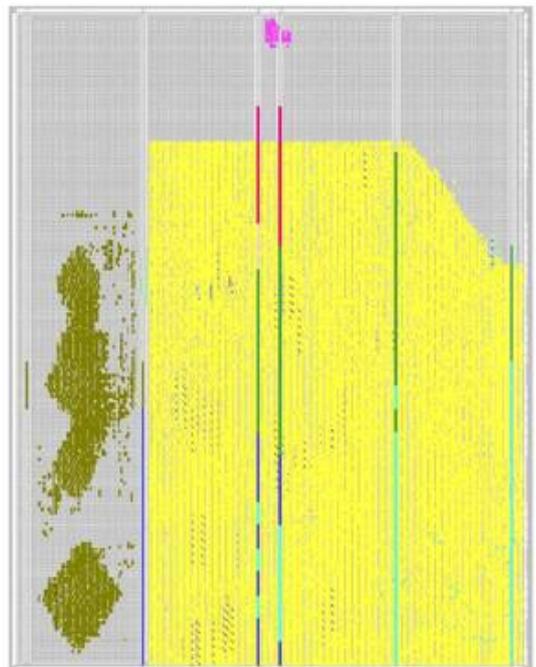
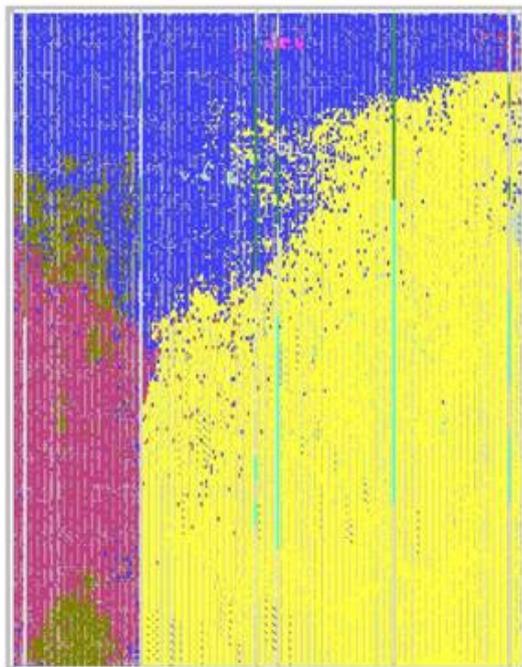


Stephen Schmitt

**Integrierte Simulation und
Emulation eingebetteter
Hardware/Software-Systeme**



Cuvillier Verlag Göttingen

Integrierte Simulation und Emulation eingebetteter Hardware/Software-Systeme

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Dipl.-Inform. Stephen Schmitt
aus Ruit auf den Fildern

Tübingen
2005

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

1. Aufl. - Göttingen : Cuvillier, 2005
Zugl.: Tübingen, Univ., Diss., 2005
ISBN 3-86537-511-1

Tag der mündlichen Prüfung: 08.06.2005
Dekan Prof. Dr. Michael Diehl
1. Berichterstatter: Prof. Dr. Wolfgang Rosenstiel
2. Berichterstatter: Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer

© CUVILLIER VERLAG, Göttingen 2005
Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

Alle Rechte vorbehalten. Ohne ausdrückliche Genehmigung des Verlages ist es nicht gestattet, das Buch oder Teile daraus auf fotomechanischem Weg (Fotokopie, Mikrokopie) zu vervielfältigen.

1. Auflage, 2005
Gedruckt auf säurefreiem Papier

ISBN 3-86537-511-1

Für meine Mutter und meinen Vater

Danksagung

Diese Arbeit entstand neben meiner Tätigkeit als Wissenschaftlicher Mitarbeiter im Arbeitsbereich Technische Informatik des Wilhelm-Schickard-Instituts für Informatik der Universität Tübingen. Mein besonderer Dank gilt meinem Doktorvater Herrn Professor Dr. Wolfgang Rosenstiel für die Betreuung der Arbeit und die exzellente Arbeitsumgebung, die er mir zur Durchführung meiner Arbeit zur Verfügung gestellt hat. Ein weiterer Dank gilt Herrn Professor Dr. Dr. E.h. Wolfgang Straßer für die Übernahme des Korreferats und seiner sorgfältigen Begutachtung der Arbeit. Darüber hinaus möchte ich mich bei meinen Projektpartnern von Infineon Technologies AG für die gute Zusammenarbeit bedanken.

Weiterhin bedanke ich mich bei Herrn Dr. Joachim Gerlach, Herrn Dipl.-Inform. Axel Braun und Herrn Dr. Klaus Beschorner für die fruchtbaren Diskussionen während meiner Arbeit sowie bei meiner Studentin Frau Katharina Weinberger und meinen Studenten Herrn Paulius Duplys und Herrn Johannes Brüggemann für die Zusammenarbeit. Ein ganz besonderer Dank gilt meinen beiden Kollegen Herrn Dipl.-Phys. Werner Dreher und Herrn Dr. Walter Lange für die vielen hilfreichen Diskussionen und Anregungen für meine Arbeit. Insbesondere danke ich Herrn Dr. Walter Lange für die Korrekturhinweise zu meiner Ausarbeitung.

Bei meinen Kolleginnen und Kollegen bedanke ich mich für die freundliche Atmosphäre am Arbeitsbereich und die angenehmen Gespräche auch abseits der Arbeit. Insbesondere gilt mein Dank den Administratorkollegen Herrn Dr. Marcus Ritt, Herrn Dipl.-Inform. Carsten Schulz-Key, Herrn Dipl.-Inform. Gerald Heim und Herrn Dipl.-Inform. Thomas Schweizer für die gute Zusammenarbeit und die stets einwandfreie Rechnerinfrastruktur.

Mein herzlichster Dank aber gilt meiner Mutter und meinem Vater, die mich auf meinem Lebensweg immer unterstützt und die mir diese Arbeit durch eine erstklassige schulische und universitäre Ausbildung erst ermöglicht haben. Bei meinen Eltern habe ich zu jeder Zeit den Rückhalt, den ich zur Erlangung meiner Ziele benötige. Ohne die vielen kleinen und großen Hilfen meiner Eltern wäre ich nie so weit gekommen und ohne sie wäre ich nie das geworden, was ich heute bin.

Nürtingen, im Juni 2005

Stephen Schmitt

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation und Problemstellung	3
1.2	Aufbau der Arbeit	5
2	Grundlagen	7
2.1	Entwurfsmethoden für eingebettete Systeme	7
2.1.1	Allgemeine Entwurfsmethoden für Software	7
2.1.2	Entwurfsmethoden für Hardware/Software-Systeme	8
2.1.3	Plattformbasierter Entwurf	10
2.1.4	Prototyping	12
2.2	Hardware-Entwicklung für eingebettete Systeme	14
2.2.1	Hardware-Beschreibungssprachen	14
2.2.2	Hardware-Synthese	15
2.2.3	Verifikation der Hardware	17
2.3	Software-Entwicklung für eingebettete Systeme	18
2.3.1	Entwicklungsumgebungen	18
2.3.2	Debugging der Software	19
2.4	Hardware-Emulation	20
2.4.1	Überblick über FPGAs	21
2.4.2	Funktionsweise SRAM-basierter FPGAs	21
2.4.3	Hardware-Architektur von FPGAs	23
2.4.4	Debugging mit integrierten Logikanalysatoren	28
3	Stand der Technik	31
3.1	Software-Entwicklung in frühen Entwurfsphasen	31
3.1.1	Entwicklung von Software-Algorithmen mit Befehlssatzsimulatoren	32
3.1.2	Einsatz von Java in eingebetteten Systemen	32
3.1.3	Transaktionsbasierte Modellierung auf hohen Abstraktionsebenen	33
3.1.4	Software-Entwicklung mit Evaluierungsboards	34
3.2	Entwicklung eingebetteter Hardware/Software-Systeme	35

3.2.1	Simulation von Hardware-Komponenten	36
3.2.2	Kombination von BSS mit HDL-Simulator	38
3.2.3	Emulation von Hardware-Komponenten	39
3.2.4	Kombination von BSS mit Hardware-Emulator	41
3.2.5	Kombination aus Prozessor-ASIC und FPGA	42
3.2.6	Integration von Prozessoren in FPGAs	43
3.2.7	Formale Verifikation	44
4	Bewertung des Stands der Technik	45
4.1	Bewertung der Entwurfsmethoden für die Software-Entwicklung . .	45
4.2	Bewertung der Entwurfsmethoden für die Entwicklung komplexer Hardware-Komponenten	46
4.3	Zielsetzung einer neuen Entwurfsmethodik für eingebettete Hardware/- Software-Systeme	48
5	Konzept einer Entwicklungsumgebung für den Entwurf eingebet- teter Hardware/Software-Systeme	53
5.1	Grundkonzept der Entwurfsmethodik	53
5.2	Beschleunigung der Anwendungsentwicklung durch Emulation . . .	55
5.2.1	Prototyping von ASIC-IP-Kernen auf FPGAs	55
5.2.2	Methodik zur Lösung des Ressourcenproblems	59
5.2.3	Architektur der rekonfigurierbaren Entwicklungsumgebung	60
5.2.4	Integration eines System-on-Chip auf dem System	63
5.2.5	Hardware-nahe Software-Entwicklung unter harten Ressour- ceneinschränkungen	67
5.3	Parallele Entwicklung von Hardware-Komponenten und hardware- naher Software	69
5.3.1	Überblick	70
5.3.2	Simulation des eingebetteten Hardware/Software-Systems .	72
5.3.3	Entwicklung von Hardware-Komponenten auf einem Board	74
5.3.4	Hardware-Entwicklung mit Hilfe von Erweiterungsplattformen	76
5.3.5	Emulation und Hardware-Debugging des SoC	78
5.3.6	Software-Entwicklung mit einer Cross-Entwicklungsumge- bung	80
5.4	Zusammenfassung	82
6	Hardware/Software-Entwicklung mit einem emulationsbasierten Mikrocontroller-IP-Kern	85
6.1	TriCore1-Mikrocontroller	85

6.1.1	Überblick	86
6.1.2	Architektur	86
6.1.3	Anbindung von Peripheriemodulen	88
6.1.4	Programmiermodell und Unterbrechungsbehandlung	89
6.1.5	TriCore1-Plattformen	90
6.2	Rapid-Prototyping-System Spyder	91
6.3	Emulation des TriCore1-Mikrokontroller-IP-Kerns	93
6.3.1	Technologische Anpassung des ASIC-IP-Kerns	93
6.3.2	Realisierung komplexer SoC-Prototypen unter harten Ressourcenbeschränkungen	96
6.3.3	Zwischenergebnis	104
6.3.4	Integration des TriCore1-Mikrokontroller-IP-Kerns auf dem Spyder-System	107
6.3.5	Zwischenergebnis	113
6.4	Entwicklung eingebetteter Hardware/Software-Systeme	115
6.4.1	Beispiel eines typischen SoC-Bussystems	115
6.4.2	Entwicklung und Integration neuer Hardware-Komponenten	117
6.4.3	Plattformbasierter Entwurf mit Erweiterungsplattformen	120
6.4.4	Verifikation der Hardware-Komponenten	128
6.4.5	Zwischenergebnis	128
6.5	Zusammenfassung	129
7	Ergebnisse	131
7.1	Entwicklung eingebetteter Software auf Hardware-Modellen	131
7.1.1	Implementierung einer TC1MP-S-Plattform	132
7.1.2	Laufzeitmessungen für eingebettete Software	137
7.1.3	Architekturgenaue Emulation des TriCore1-Mikrokontroller-Kerns	144
7.1.4	Zusammenfassung	147
7.2	Entwicklung eingebetteter Hardware/Software-Systeme	147
7.2.1	Überblick über die Hardware-Komponenten	148
7.2.2	Syntheseergebnisse	149
7.2.3	Debugging von Hardware und Software	153
7.2.4	Laufzeitmessungen	156
7.2.5	Zusammenfassung	159
8	Zusammenfassung	161
A	Abkürzungen	163

Abbildungsverzeichnis

1.1	Entwicklung des Anteils der Software an den Ausgaben für Hardware und Software in der Automobilindustrie.	2
2.1	Entwicklung eingebetteter Hardware/Software-Systeme.	9
2.2	Entwurfsmethoden integrierter Schaltungen und SoCs.	10
2.3	Hardware/Software-Entwicklung beim plattformbasierten Entwurf. .	12
2.4	Systementwicklung mit Rapid Prototyping.	13
2.5	VHDL-basierter ASIC-Entwurfsablauf.	16
2.6	Anwendungsentwicklung mit einer Cross-Entwicklungsumgebung. .	18
2.7	Aufbau der oberen Hälfte eines FPGA-Logikblocks.	22
2.8	Architektur des Virtex-II-Pro-FPGAs von Xilinx.	23
2.9	Architektur des Virtex-II-FPGAs von Xilinx.	24
2.10	Aufbau und Schnittstellen eines Virtex-II Block-RAM-Bausteins. .	25
2.11	Aufbau eines Virtex-II E/A-Blocks.	25
2.12	Konkurrierende Ansteuerung der vier Quadranten eines FPGAs durch Taktmultiplexer.	26
2.13	Debugging von Hardware-IP-Komponenten mit in FPGAs integrierten Logikanalysatoren.	28
3.1	Evaluierungsboard TriBoard TC1920A.	35
3.2	Entwicklungsumgebung für die TriCore1-DesignWare-Komponente. .	37
3.3	Entwicklung eingebetteter Hardware/Software-Systeme durch Kombination aus BSS und HDL-Simulator.	38
3.4	Hardware-Emulations-System von Aptix.	40
3.5	Entwicklung eingebetteter Hardware/Software-Systeme durch Kombination aus BSS und Hardware-Emulator.	41
3.6	Spyder-System mit Hitachi-SH3-Mikrokontroller-Board (oben) und Xilinx Virtex-FPGA-Board.	42
3.7	Architektur des Virtex-II-Pro-FPGAs von Xilinx.	43
4.1	Laufzeitvergleich verschiedener Modelle für die Entwicklung eingebetteter Software.	47

5.1	Klassische Entwicklungsumgebung für eingebettete Software und neue Entwicklungsumgebung für eingebettete Hardware/Software-Systeme.	54
5.2	Klassifikation des Prototypings von ASIC-IP-Kernen mit FPGAs.	56
5.3	Abschalten des Taktes mit Hilfe eines <i>gate</i> -Signals in ASICs.	57
5.4	Umwandlung eines bedingten Taktes für FPGAs.	57
5.5	Entwicklung der Transistorkapazitäten von ASICs und FPGAs.	58
5.6	Kosteneinsparung bei der Prototyping-Plattform durch Partialemulation.	59
5.7	Generische Prototyping-Plattform zur Integration komplexer SoCs.	61
5.8	Aufbau eines generischen SoC mit Mikrokontroller, Speicherhierarchie und Peripheriekomponenten.	63
5.9	Abbildung einer Menge S_K von SoC-Komponenten auf die Ressourcen R_{PS} eines Prototyping-Systems.	64
5.10	Abbildung eines SoC auf ein FPGA-basiertes Prototyping-System.	67
5.11	Software-Entwicklung unter harten Ressourceneinschränkungen.	68
5.12	Erweiterung der rekonfigurierbaren Entwicklungsumgebung um Erweiterungsplattformen.	70
5.13	Entwurfsablauf für die Entwicklung eingebetteter Hardware/Software-Systeme.	71
5.14	Integration des Objektcodes in das SRAM-HDL-Modell der SRAMs des generischen Prototyping-Systems.	74
5.15	Hierarchie des generischen SoC.	75
5.16	Architektur des SoC-Prototypen mit dem Peripherie-Submodul.	75
5.17	Integration komplexer Hardware-Komponenten mit einer Erweiterungsplattform.	77
5.18	Partitionierung des SoC mit Erweiterungsplattform.	78
5.19	Anbindung einer Cross-Entwicklungsumgebung an ein generisches Prototyping-System.	80
5.20	Software-Entwicklung mit der Cross-Entwicklungsumgebung.	81
6.1	Architektur des TriCore1-Mikrokontroller-IP-Kerns.	87
6.2	Beispiel eines FPI-Bus-basierten SoC mit unterschiedlichen Peripheriemodulen.	88
6.3	Aufteilung des Adressraums des TriCore1-Prozessors.	89
6.4	Blockschaltbild des Spyder Rapid-Prototyping-Systems in der Version 2.	92
6.5	Modulstruktur des TriCore1 TC1MP-S-Softmakros.	94
6.6	Synthese des TriCore1 VHDL-Quelltextes für Xilinx VirtexII-FPGAs.	95
6.7	TriCore1-CPU-Modul mit unterschiedlichen Speicheranschlüssen.	98

6.8	TriCore1-CPU-Modul mit einfacher LMB-Schnittstelle.	100
6.9	TriCore1-CPU-Modul mit LMBh-Anbindung und einfacher Speicherhierarchie.	101
6.10	Entwicklungsumgebung für ein eingeschränktes TriCore1-basiertes SoC.	103
6.11	Laufzeitvergleich von Software-Benchmarks zwischen Befehlssatzsimulator und dem TriCore1 auf dem Spyder-Board.	105
6.12	Synthesergebnisse verschiedener TriCore1-Varianten für Xilinx Virtex-2000E-FPGAs.	106
6.13	Schnittstellen eines minimalen TC1MP-S-Systems.	108
6.14	Implementierung der TC1MP-S-Plattform auf dem Spyder-System.	109
6.15	Software-Entwicklung mit der rekonfigurierbaren Entwicklungsumgebung.	111
6.16	Software-Debugger-Anbindung für Mikrokontroller von Infineon Technologies AG.	112
6.17	Hardware-Module des TriCore1 für das Software-Debugging.	113
6.18	Synthesergebnisse einiger beim TriCore1-Mikrokontroller verfügbarer Hardware-Konfigurationen.	114
6.19	Implementierung des FPI-Busses auf dem Spyder-System.	117
6.20	Implementierungsmöglichkeiten für die Einbindung neuer Hardware-Komponenten.	118
6.21	Implementierung eines TriCore1-basierten SoC mit zwei getrennten Submodulen für Peripheriekomponenten.	119
6.22	Implementierung eines verteilten SoC-Entwurfs mit einer TriCore1-Plattform und unidirektionalen Verbindungen auf der Backplane.	122
6.23	Verbrauch von Anschlussressourcen bei einer direkten Abbildung des FPI-Busses auf die Backplane.	123
6.24	Abbildung des FPI-Busses auf DDR-E/A-Anschlüsse des FPGAs und die Spyder-Backplane.	124
6.25	Abbildung des FPI-Busses auf Tristate-E/A-Anschlüsse des FPGAs und die Spyder-Backplane.	127
6.26	Ressourcenverbrauch unterschiedlicher FPI-Bus-Realisierungen auf der Backplane des Spyder-Systems.	129
7.1	Implementierung einer minimalen TC1MP-S-Plattform.	132
7.2	Synthesergebnisse für die Implementierung der TC1MP-S-Speicher in Block-RAMs und SRAMs.	134
7.3	Platzierung der TAG-RAMs für 4KB P/D-Caches mit Logikressourcen bzw. Block-RAMs.	135

7.4	Vergleich der Ressourcenauslastung verschiedener TAG-RAM Implementierungen.	136
7.5	Befehlsanzahl und -verteilung für EEMBC-Benchmarks.	141
7.6	Laufzeitergebnisse für EEMBC-Benchmarks lokalisiert in Speicherbereichen, die nicht in den Cache geladen werden.	142
7.7	Laufzeitergebnisse für EEMBC-Benchmarks lokalisiert in Speicherbereichen, die in den Cache geladen werden.	143
7.8	Laufzeitvergleich für EEMBC-Benchmarks lokalisiert in den Scratchpad-RAMs.	144
7.9	Vergleich der geschätzten Laufzeiten durch den BSS mit tatsächlich gemessenen Laufzeiten auf der Hardware.	145
7.10	Laufzeitvergleiche unterschiedlicher Realisierungen der TAG-RAMs im Spyder-FPGA.	146
7.11	Implementierung der RSK-Komponente auf dem Prototyping-System.	150
7.12	Auslastung des Spyder-FPGAs XC2V8000 für die Integration der drei Hardware-Komponenten zusammen mit der TC1MP-S-Plattform.	151
7.13	Auslastung des Spyder-Erweiterungsboards für die drei zusätzlichen Hardware-Komponenten.	152
7.14	Vergleich der Synthesedauer zwischen Implementierungen mit (EP) und ohne (EB) Erweiterungsplattformen.	153
7.15	Ansicht eines Software-Debuggers für einen SSC-Initialisierungs- und Sendetest.	154
7.16	Interne Signale der SSC-Schnittstelle aufgezeichnet mit dem internen Logikanalysator ChipScope Pro.	155
7.17	Ausgangssignale der SSC-Schnittstelle aufgenommen mit einem externen Logikanalysator.	156
7.18	Software-Entwicklung mit einem HDL-Simulator.	157

Tabellenverzeichnis

2.1	Platzierung eines Entwurfs mit 16 Taktdomänen mit acht primären und acht sekundären Taktmultiplexern.	27
4.1	Bewertung der Entwurfsverfahren für eingebettete Software.	49
4.2	Bewertung der Entwurfsverfahren für Hardware-Komponenten und hardware-nahe Software.	50
5.1	Einteilung der Komponenten eines generischen SoC in Prioritätsklassen.	65
6.1	Synthesergebnisse für den TC1MP-S und Submodule für Xilinx Virtex-2000E-FPGAs.	97
6.2	Einteilung der Komponenten des TC1MP-S-Kerns in Prioritätsklassen.	102
6.3	Laufzeitvergleich von Software-Benchmarks zwischen RTL-Simulator und dem TriCore1 auf dem Spyder-Board der ersten Generation. . .	104
6.4	FPI-Bus-Signale des TriCore1-Mikrokontrollers.	116
7.1	Synthesergebnisse für die Implementierung des TC1MP-S mit 4KB und 8KB Programm- und Datencaches in den Slices des FPGAs. . .	135
7.2	Laufzeitvergleich für ein einfaches <i>Hello world!</i> -Programm.	137
7.3	Laufzeiten für die Entwicklung hardware-naher Software mit HDL-Simulator und FPGA-Emulation.	159

Programmverzeichnis

5.1	Ansteuerung des Debugging-Moduls.	69
6.1	Implementierung eines FPGA-E/A-Anschlusses mit doppelter Datenrate.	125
7.1	Integration von Debugging-Marken für die HDL-Simulation.	158

1 Einleitung

Nach dem Erfolg der Informationstechnologie in der Büro- und Arbeitswelt werden eingebettete Systeme als das wichtigste Anwendungsgebiet der Informatik in den kommenden Jahren betrachtet. In diesem Zusammenhang wird auch gerne von der *Nach-PC-Ära* gesprochen. Beispielsweise ist heute jede moderne Küche mit mehr Rechenkapazität ausgestattet als der Steuerrechner der Apollo Mondlandefähre im Jahre 1969.

Bedingt durch den stetigen Technologiefortschritt, der es ermöglicht, durch kleiner werdende Strukturgrößen immer mehr Funktionalität auf einem Chip unterzubringen, steigt die Anwendungsvielfalt und Komplexität der Systeme bei gleichzeitig sinkenden Entwurfszeiten und -kosten. Daraus ergibt sich ein komplexer Systementwurf der die Integration vieler unterschiedlicher Funktionen in ein Gesamtsystem auf einem Chip (*System-on-a-Chip (SoC)*) unterstützen muss.

Insbesondere spielt die Kombination eines oder mehrerer Mikrokontroller mit zusätzlichen Peripheriekomponenten eine zentrale Rolle. Aufgrund der Komplexität zukünftiger eingebetteter Systeme wird die Wiederverwendung möglichst vieler Komponenten an Bedeutung gewinnen. Diese werden in Form von *Intellectual Property (IP)* als Hardware und Software zu einem Gesamtsystem kombiniert. Bei der Vernetzung eingebetteter Systeme können beispielsweise die Schnittstellen in Hardware realisiert und das System anwendungsspezifisch damit erweitert werden. Im Automobilbereich erlangt z.B. das Echtzeitbussystem FlexRay [32] zunehmend an Bedeutung, mit dem es zukünftig möglich sein wird, sicherheitskritische Komponenten miteinander zu vernetzen und die Sicherheit zu erhöhen.

Daneben kommt der Software in zukünftigen eingebetteten Systemen eine wichtige Rolle zu. In Abbildung 1.1 ist die Entwicklung des Anteils der Software an den Ausgaben für Hardware und Software in der Automobilindustrie für die nächsten Jahre dargestellt. In diesem Zusammenhang kann eine Parallele zum Moore'schen Gesetz formuliert werden, die besagt, dass sich für viele Produkte aus dem Verbraucherbereich die Größe des Codes alle zwei Jahre verdoppelt [100].

Insbesondere ist aus dem Bild ersichtlich, dass sich die Software zukünftiger Systeme modularer zusammensetzt. Wurde früher die Software noch als Ganzes gesehen, wird in Zukunft auf einem Betriebssystem (RTOS) *Basissoftware*, wie etwa Treiber für Hardware-Komponenten oder Middleware für die Vernetzung, zusammen mit unterschiedlichen Anwendungen implementiert werden, die unter Umstän-