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-

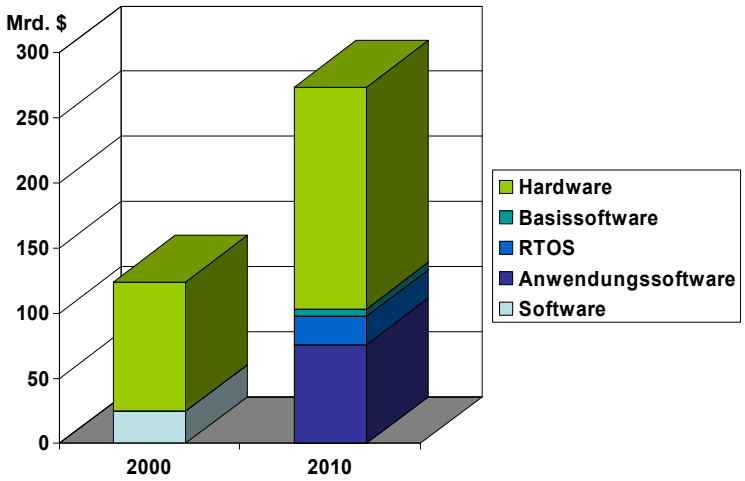

Abbildung 1.1: Entwicklung des Anteils der Software an den Ausgaben für Hardware und Software in der Automobilindustrie (Quelle: Mercer, 2003).

den auch dynamisch nachgeladen werden können.

Durch die Verwendung von Software lassen sich die Produkte mit unterschiedlichen Funktionen ausstatten, um sie von Konkurrenzprodukten abzuheben. Heute sind neue Produkte wie Mobiltelefone oder Digitalkameras für den Kunden nur noch dann interessant, wenn sie wesentliche Neuerungen gegenüber bereits bekannten Geräten bieten. Der Zeitpunkt, zu dem ein neues Produkt auf den Markt kommt, hat einen entscheidenden Einfluss auf den Erfolg. So verzeichnete Siemens 2004 einen Verlust von 152 Mio Euro unter anderem aufgrund der Tatsache, dass man die Entwicklung in Bezug auf die Integration von Digitalkameras in Mobiltelefonen nicht vorausgesehen hat [43].

Darüber hinaus spielt die Verifikation und Qualitätssicherung bei der zunehmenden Komplexität eingebetteter Systeme ein wichtige Rolle. Speziell im Bereich sicherheitskritischer Anwendungen sind ausgiebige Tests unter möglichst realistischen Umweltbedingungen unumgänglich. Treten bei solchen Systemen nach der Auslieferung Probleme auf, so sind die Firmen zu Rückrufaktionen gezwungen, die nicht nur sehr teuer werden, sondern auch ihrem Image schaden. Gleiches gilt auch für Produkte aus dem Verbraucherbereich wie beispielsweise Mobiltelefone, Digitalkameras oder MP3-Spieler.

Aus diesem Grund ist das *Prototyping* ein wichtiger Bestandteil des Entwurfsprozesses eingebetteter Systeme. Beim Prototyping wird versucht, die Funktionalität eines Systems anhand eines physikalischen Prototyps zu evaluieren. Oft lassen sich komplexe Effekte, die z.B. durch Umwelteinflüsse entstehen, nur anhand eines real existierenden physikalischen Modells des zu entwickelnden Systems genauer un-

tersuchen und bewerten. Deshalb wird das Prototyping unter anderem gerne in den Bereichen der Automobilentwicklung und Kommunikationstechnik eingesetzt.

Beim Prototyping moderner eingebetteter Systeme müssen unterschiedliche Entwicklungsmethoden und -werkzeuge kombiniert werden. Betrachtet man die Hardware-Komponenten eines Systems, so ist die Integration der verschiedenen funktionalen Blöcke sehr aufwändig. Beim Entwurf eingebetteter Software sind gegenüber der konventionellen Software-Entwicklung andere Anforderungen wichtig. Hier spielen Echtzeiteigenschaften, Speicherplatz, Energieverbrauch und Sicherheitsaspekte eine wichtige Rolle. Nicht zuletzt müssen die Schnittstellen zwischen den Hardware- und Software-Komponenten des eingebetteten Systems eine verlässliche und reibungslose Zusammenarbeit zwischen den einzelnen Komponenten garantieren.

1.1 Motivation und Problemstellung

Der bisherige Entwurfsprozess für eingebettete Systeme und speziell der SoCs ist stark auf die Verifikation der Hardware ausgerichtet. Die Integration und der Test neu entwickelter Hardware-Komponenten wird heute vor allem mit hardware-spezifischen Entwicklungsplattformen und Verifikationsmethoden durchgeführt. Wie im vorangegangenen Abschnitt allerdings bereits angedeutet wurde, steigt der Anteil der Software in eingebetteten Systemen rasant an, weshalb die Notwendigkeit besteht, neue, auch software-zentrierte Entwicklungs- und Verifikationsmethoden für eingebettete Hardware/Software-Systeme zu entwickeln.

Diese Arbeit entstand im Rahmen des Forschungsprojektes „Spezifikation und Algorithmus/Architektur-Codesign für hochkomplexe Anwendungen der Automobil- und Kommunikationstechnik (SpeAC)“ [94] in Zusammenarbeit mit der Firma Infineon Technologies AG. Ziel des Forschungsprojektes ist eine Produktivitätssteigerung des Entwurfs eingebetteter Hardware/Software-Systeme durch die Entwicklung einer Algorithmus/Architektur-Codesign-Methodik, die auf hohem Abstraktionsniveau beginnend, einen durchgängigen Entwurfsablauf ermöglicht und auf der Basis vordefinierter Hardware/Software-Plattformen angelegt ist.

Innerhalb dieser Entwurfsmethodik konzentrieren sich die in dieser Arbeit vorgestellten Konzepte auf die Systemintegration von Hardware und Software, das Hardware/Software-Co-Debugging und Rapid Prototyping komplexer Systementwürfe. Die Ziele dieser Arbeit lassen sich daher auch in zwei Teilbereiche untergliedern:

1. Entwicklung einer Entwurfsmethodik für eingebettete Software für SoC-Entwürfe auf der Basis eines schnellen und architekturgenauen Modells der SoC-Hardware.

In heutigen Systemen spielt die Verkürzung der Entwurfszeit eine entscheidende Rolle. Hier liegt die Software-Entwicklung auf dem kritischen Pfad, da für deren Entwicklung ein Modell des Mikrokontrollers verfügbar sein muss. Da der Umfang der Programme stetig ansteigt, muss deren Ausführungszeit signifikant verkürzt werden. Dazu ist die Integration von Software-Entwicklungswerkzeugen und insbesondere die Anbindung einer Software-Debuggingumgebung an das beschleunigte Hardware-Modell zwingend notwendig.

Für die Evaluierung von Software-Algorithmen und Performanzabschätzungen des Zielsystems ist darüber hinaus ein möglichst architekturgenaues Modell der Hardware notwendig. An dieser Stelle weisen heutige Entwurfsmethoden Defizite auf, da die Ausführung der Software auf detaillierten Hardware-Modellen sehr viel Zeit in Anspruch nimmt. Da an der Software-Entwicklung darüber hinaus mehrere Entwickler arbeiten können, sollte die Entwicklungsplattform keine hohen Kosten verursachen, damit eine parallele Entwicklung unterschiedlicher Teile der Software mit mehreren Systemen möglich ist.

Teile dieser Software müssen dabei für neue Hardware-Komponenten entwickelt werden, die die Funktionalität des SoC anwendungsspezifisch erweitern. Ein zweites Ziel dieser Arbeit ist deshalb:

2. Bereitstellung einer Entwurfsmethodik, für die parallele Entwicklung und den Test komplexer Hardware-Komponenten und der dazugehörigen Software[1].

Durch die parallele Entwicklung von Hardware und Software lässt sich die Entwurfszeit drastisch reduzieren. Insbesondere entfällt durch das Testen der Hardware-Komponenten durch die Software auf dem Prozessor-Modell die Erstellung einer Hardware-Testumgebung. Die frühe Evaluierung von Hardware und dazugehöriger Software ist wichtig, da in dieser Entwicklungsphase Fehler in der Hardware-Komponente und der Hardware/Software-Schnittstelle noch einfach behoben werden können. Nach dem Tape-out des Chips ist dies nur durch eine neue Implementierung möglich, was Kosten in Millionenhöhe verursachen kann.

Die Entwurfsmethodik muss detaillierte Einblicke in die internen Abläufe der Hardware-Komponenten bis auf Signalebene zulassen. Ferner muss das Debugging der Hardware-Komponenten ermöglicht und die Turn-around-Zeiten für das Wiederaufsetzen der Entwicklungsumgebung nach Beseitigung eines Entwurfsfehlers verkürzt werden.

Das in dieser Arbeit vorgestellte Konzept ermöglicht es einem Hardware-IP-Anbieter darüber hinaus, Details seiner IP vor dem Kunden zu verstecken und so den

[1] Im Folgenden wird die Software, die für die Kontrolle einer Hardware-Komponente verwendete wird, als *Treiber* oder hardware-abhängige Software bezeichnet.

Quelltext seiner IP zu schützen. Dies stellt ein nicht unerhebliches Kriterium für eine Entwicklungsplattform für SoC-Entwürfe dar, da die Kosten für die IP selbst im siebenstelligen Bereich liegen können.

Insgesamt wird in dieser Arbeit ein *ganzheitlicher* Ansatz entwickelt und umgesetzt, der eine architekturgenaue und detaillierte Sicht auf ein zu entwickelndes System zulässt und dennoch die Ausführungszeiten für die zu entwickelnde Software auf dem Hardware-Modell beschleunigt. Diese Beschleunigung der Hardware-Modelle, die in einer Hardware-Beschreibungssprache wie z.B. VHDL [7] vorliegen, ist durch Emulation möglich. Heutige Emulations-Systeme sind allerdings sehr teuer und stellen dadurch eine Hürde für die Evaluierung neuer Technologien durch einen IP-Kunden dar.

Die **Idee** dieser Arbeit besteht deshalb darin, für den Entwurf eingebetteter Hardware/Software-Systeme eine *kostengünstige* rekonfigurierbare Entwicklungsumgebung zu verwenden, die auf einer FPGA-basierten Emulationsplattform basiert. Die Entwicklungsumgebung bietet dem Entwickler ein *Soft*-Evaluierungsboard, auf dem die SoC-Hardware integriert und durch die Kopplung mit einem Software-Debugger darüber hinaus auch die Entwicklung hardware-naher Software ermöglicht wird. Dadurch lässt sich eine signifikante Beschleunigung des Hardware-Modells und damit der Anwendungsentwicklung insgesamt bei dennoch moderaten Kosten für die Entwicklungsumgebung erreichen.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit lässt sich in vier Teile untergliedern. Neben den Grundlagen und einer Diskussion des Stands der Technik wird ein neues Konzept zur Entwicklung eingebetteter Hardware/Software-Systeme vorgestellt. Dieses Konzept wird anhand eines ausführlichen Beispiels aus der Praxis näher erläutert und die Relevanz der entwickelten Methoden wird anhand von Ergebnissen gezeigt.

In Kapitel 2 werden zunächst Grundlagen behandelt, die für das Verständnis der in dieser Arbeit angesprochenen Begriffe und Methoden notwendig sind. Kapitel 3 enthält eine Zusammenfassung des Stands der Technik bei der Entwicklung von Hardware und Software von SoCs. Abschnitt 3.1 behandelt zunächst den Entwurf eingebetteter Software auf Hardware-Modellen bevor in Abschnitt 3.2 bisherige Verfahren für die parallele Entwicklung von Hardware-Komponenten und deren Software für SoCs vorgestellt werden. In Kapitel 4 findet sich eine Bewertung des Stands der Technik. Es werden Defizite aktueller Methoden identifiziert und in Abschnitt 4.3 die Ziele der vorliegenden Arbeit definiert.

Im zweiten Teil dieser Arbeit wird in Kapitel 5 ein neues Konzept des Entwurfs eingebetteter Hardware/Software-Systeme für SoCs vorgestellt. Dort wird zunächst

in Abschnitt 5.1 ein Überblick über die Grundideen des neuen Konzepts gegeben. Abschnitt 5.2 enthält eine Methodik für die Beschleunigung der Software-Entwicklung für SoCs auf der Basis eines emulationsbasierten Mikrokontroller-IP-Kerns. Dieses Konzept wird in Abschnitt 5.3 zu einer Methodik zur parallelen Entwicklung neuer Hardware-Komponenten und deren hardware-naher Software ausgebaut.

Kapitel 6 zeigt mit Hilfe eines praxisrelevanten Beispiels, wie die in dieser Arbeit vorgestellten neuen Konzepte umgesetzt werden können. Es wird anhand des TriCore1®-Mikrokontrollers gezeigt, wie dieser auf einem FPGA-basierten Prototyping-System integriert und als SoC-Plattform für die Entwicklung neuer Hardware-Komponenten und ihrer dazugehörigen hardware-nahen Software und Anwendungen verwendet werden kann.

In Kapitel 7 werden Ergebnisse präsentiert, die mit dem vorgestellten Konzept erzielt werden können. Diese beziehen sich zum Einen auf Laufzeitmessungen, die mit einem architekturgenauen Modell einer SoC-Hardware gemacht wurden und zum Anderen auf die Darstellung der Entwicklung und Verifikation von Hardware-Komponenten und deren Software. Kapitel 8 enhält eine Zusammenfassung der im Rahmen dieser Arbeit entwickelten Konzepte und Methoden und der daraus gewonnenen Erkenntnisse.