Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2 - SoSe 2011

Steffen Rempp

Entwurfstechniken für Parallelisierung auf MPSoC Plattformen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Steffen Rempp

Entwurfstechniken für Parallelisierung auf MPSoC Plattformen

Ausarbeitung eingereicht im Rahmen der Veranstaltung Anwendungen 2
im Studiengang Studiengang
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Gutachter : Prof. Dr.-Ing Birgit Wendholt
Gutachter : Prof. Dr. Franz Korf

Abgegeben am 1. September 2011

Inhaltsverzeichnis

1 Einführung **4**
1.1 FAUST High Performance Embedded Computing 5
1.2 Entwicklungsplattform TI OMAP 4433 mit ARM Cortex A9 Dual Core 6
1.2.1 ARM Cortex A9 Dual Core Prozessor 7
1.2.2 PandaBoard 7

2 Vergleichbare Arbeiten **8**
2.1 IMEC MPSoC Programming Approach 8
2.2 Efficient OpenMP Support and Extensions for MPSoCs with Explicitly Managed Memory Hierarchy 10
2.3 A Cross-Domain Multiprocessor System-on-a-Chip for Embedded Real-Time Systems 11

3 Schluss **14**

Abbildungsverzeichnis **15**

Literaturverzeichnis **16**

1 Einführung

Im September 2010 wurde anlässlich des 4. IT-Gipfels der Bundesregierung die Nationale Roadmap Embedded Systems [ZVEI (2010)] veröffentlicht. Sie soll als Leitfaden für die strategischen Forschungsfelder der Zukunft im Embedded Bereich dienen. Laut dem Herausgeber der Roadmap, dem Kompetenzzentrum Embedded Software & Systems im Zentralverband Elektrotechnik- und Elektronikindustrie e.V. seien “eingebettete Systeme in ihrem Softwareanteil nicht sichtbar und doch hoch relevant für den Wirtschaftsstandort Deutschland“. Zudem wurden zehn Thesen verabschiedet, was die Bedeutung der Embedded Systeme betrifft, sowie sechs Forschungsschwerpunkte der Zukunft genannt, um den Embedded Markt weiter voranzutreiben. Da es sich bei Embedded Systemen um eine Querschnittstechnologie handle und die einzelnen Anwendungsgebiete sich teilweise sehr stark in ihren spezifischen Anforderungen unterscheiden, ziehe dies einen hohen Forschungsbedarf [Ziegler (2010)] nach sich. Weiterhin werden auch immer mehr Stimmen lauter, die mit der 6. These der Roadmap übereinstimmen und sich eine Standardisierung über Branchengrenzen hinweg wünschen und einen gemeinsamen Nenner fordern. Der Embedded Markt in Deutschland habe sich in den vergangenen Jahren mit stabilen Zuwachsraten von bis zu acht Prozent entwickelt und lag 2010 bei ca. 20 Milliarden Euro. Auch in der Infrastruktur für die Elektromobilität, dem wohl wichtigsten strategischen Ziel der Automobilindustrie, liege viel Potenzial für Embedded Systems. Einer der sechs genannten Forschungsschwerpunkte lautet “Architekturprinzipien“. Darin wird formuliert, dass es absehbar sei, dass sich Prozessorarchitekturen in Embedded Systems auf ähnliche Weise weiterentwickeln werden, wie universelle Prozessorarchitekturen, die bereits heute mehrere Kerne aufweisen. Somit sei es an der Zeit, dass Single-Core Prozessoren von hochparallelen Multi-Core Architekturen abgelöst würden. Speziell für Embedded Systems wirft dies wissenschaftliche Herausforderungen und Fragestellungen auf, wie z.B. nach der Parallelisierung und Verteilung sicherheitskritischer oder echtzeitfähiger Anwendungen auf verteilte Multiprozessor-Architekturen.

Das Ziel meiner eigenen Arbeit ist es, eine Multiprozessor-Plattform für das Projekt High Performance Embedded Computing (HPEC) (vgl. Kap. 1.1) an der HAW Hamburg bereitzustellen und Methoden zu finden, wie bereits bestehende, serielle Software- und Hardwarefunktionen parallelisiert werden und auf die Plattform portiert werden können. Das Ziel dieser Ausarbeitung im Rahmen der Veranstaltung Anwendungen 2 im Masterstudiengang Informatik ist es, hierzu vergleichbare Arbeiten (vgl. Kap. 2) vorzustellen, wie Software für eine eingebettete Multiprozessor-Plattform parallelisiert werden kann, bzw. welche Möglichkeiten der Parallelisierung auf verschiedenen Hierarchieebenen innerhalb einer eingebetteten Multiprozessor-Plattform bestehen.

1.1 FAUST High Performance Embedded Computing

Das Projekt “FAUST“ an der HAW Hamburg beinhaltet u.a. den Forschungsschwerpunkt “High Performance Embedded Computing“ (HPEC). Ziel dieses Schwerpunktes ist es, eine FPGA-basierte Embedded Plattform zur Fahrzeugsteuerung für ein autonomes Fahrzeug zu entwickeln und zu implementieren. Für die Entwicklung und technische Realisierung einer solchen Plattform werden u.a. Spartan3 oder Virtex5 FPGAs verwendet. Dank dieser Technologie werden sowohl die Integration von Software Lösungen mit Hilfe des MicroBlazes als auch die Implementierung von Algorithmen in Hardware gewährleistet. Die Modellierung der Algorithmen für eine Implementierung in Hardware eignet sich insbesondere für die Echtzeit-Signalverarbeitung, z.B. in der digitalen Bildverarbeitung zur Fahrspurerkennung und -führung. Weiterhin bewährt sich die Implementierung von Algorithmen in Hardware ebenfalls bei großen Datenmengen, um dem stark begrenzten Speicherplatz auf einer Embedded Plattform zu begegnen. Der von einer Kamera gelieferte Datenstrom kann dank der parallelen Arithmetik-Ressourcen eines FPGAs auch bei erheblich niedrigeren Taktfrequenzen beherrschbar gemacht werden.
Die bisher verwendete SoC Plattform besteht aus einer Vielzahl von Intellectual Properties (IPs), die über ein PLB-Bussystem mit einem Softcore MicroBlaze auf dem FPGA verbunden sind. Des Weiteren werden die IPs auf der FPGA Plattform nach ihrer Echtzeitfähigkeit unterteilt, also in Realtime IPs und Non-Realtime IPs. Die Realtime IPs werden für die Erfassung der Kamera- u. Sensordaten verwendet, während die Non-Realtime IPs Debug- und Kontrollfunktionen implementieren. Nicht nur aufgrund dieser Unterteilung nach Echtzeitverhalten, sondern auch aufgrund der Aufteilung der Implementierung aller Funktionen zum einen in Software, zum anderen in Hardware soll die SoC Plattform durch eine ebenfalls FPGA-basierte MPSoC Plattform ersetzt werden. Ein prinzipieller Kernaspekt bei der Entwicklung einer MPSoC-Plattform ist es, eine Entscheidung zu treffen, welche Funktionalitäten in Software und welche Funktionalitäten in Hardware implementiert werden sollen. An dieser Stelle findet nun schon eine Hardware-Software Partitionierung statt, und damit eine Parallelisierung auf System- bzw. Plattformebene.

Meine Projektarbeit im 2. Semester des Masterstudiengangs Informatik führt nun allerdings weg von der Entwicklung und Implementierung einer FPGA-basierten MPSoC-Plattform. Parallel zur Entwicklung einer FPGA-basierten MPSoC Plattform soll nun eine weitere MPSoC Plattform für das FAUST-Projekt HPEC entwickelt und bereitgestellt werden, die nicht auf einem FPGA sondern auf einer ARM Cortex A9 DualCore Architektur basiert. Die Eckpfeiler und Herausforderungen bei der Entwicklung sind ebenfalls wie schon erwähnt die Unterteilung in Realtime und Non-Realtime Funktionalitäten, sowie die Partitionierung der Implementierungen aller Funktionalitäten in Hardware und Software, darüber hinaus die zusätzliche Parallelisierung auf Softwareebene durch das Ausnutzen der SMP-Fähigkeit des DualCore Prozessors. Es soll also Parallelisierung innerhalb dieser MPSoC Plattform auf mehreren Ebenen stattfinden, sowohl auf Softwareebene, als auch auf Plattformebene durch die Hardware-Software Partitionierung. Durch das Entwickeln und Bereitstellen dieser zusätzlichen MPSoC-Plattform soll in den kommenden Semestern das Verständnis und KnowHow im Bereich der Multiprozessor Technologie im Embedded Bereich weiter vertieft und erweitert werden. Die wissenschaftliche Motivation

hierbei liegt in der Verknüpfung der Themen Parallelisierung und Embedded Systems durch MPSoC Architekturen. Durch die Entwicklung mehrerer MPSoC-Plattformen bietet sich zusätzlich noch die Möglichkeit, die verschiedenen Hardwaretechnologien bezüglich verschiedener Aspekte zu vergleichen, wie z.B. in puncto Performance, Energieeffizienz oder Speicherplatz.

1.2 Entwicklungsplattform TI OMAP 4433 mit ARM Cortex A9 Dual Core

Bei der bereits erwähnten und im Projekt verwendeten, nicht FPGA-basierenden MPSoC Plattform handelt es sich hierbei um die TI OMAP 4433 (Open Multimedia Application Platform) von Texas Instruments (TI), eine proprietäre Prozessorarchitektur, welche auf der ARM-Architektur basiert. Das Kernstück dieser Plattform bildet ein ARM Cortex A9 Dual Core Prozessor mit SMP-Fähigkeit. Die OMAP Plattform wurde u.a. für den Einsatz in Smartphones und anderen mobilen Endgeräten entwickelt, um den wachsenden technischen Anforderungen heutiger Anwendungen an die Plattform im Bereich Mobile und Multimedia Computing zu begegnen. TI verspricht sich mit dieser Plattform-Architektur vor allem High Performance Multimedia Computing bei geringem Energieverbrauch. Ein weiteres Bestreben von TI besteht darin, mit der OMAP Plattform dank des integrierten ARM Multiprozessors SMP in die Welt des Mobile Computings zu führen.

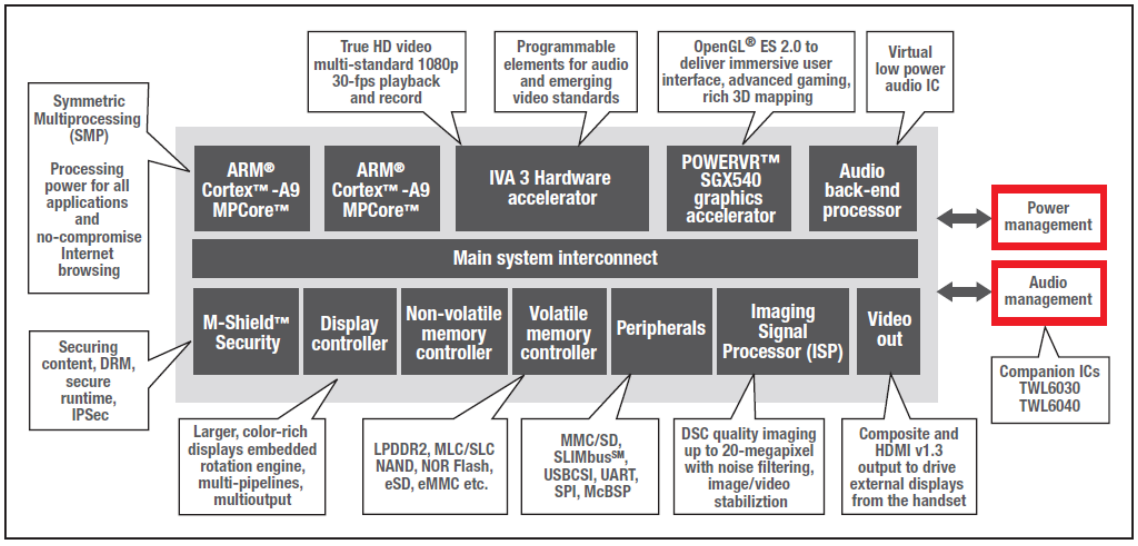

Abbildung 1.1: TI OMAP 4 Plattform

Neben vielen Multimedia Funktionalitäten und diversen Schnittstellen beinhaltet die TI OMAP 4433 Plattform folgende Subsysteme (vgl. Abbildung 1.1):

- Cortex A9 Mikroprozessor Subsystem mit zwei ARM Cortex A9 Kernen
- IVA 3 Hardwarebeschleuniger
- Digital Signal Prozessor (DSP)
- Image Signal Prozessor (ISP)
- Cortex M3 Mikroprozessor Subsystem mit zwei ARM Cortex M3 Prozessoren
- POWERVR 3D Grafikprozessor
- Audio backend (ABE) Prozessor

Daneben bietet OMAP 4430 Plattform Unterstützung für die High-Level Betriebssysteme Linux, Palm OS, Symbian OS und Windows CE.

1.2.1 ARM Cortex A9 Dual Core Prozessor

Aufgrund der SMP-Fähigkeit und der Unterstützung diverser High-Level Betriebssysteme ist der ARM Cortex A9 Dual Core Prozessor für die Projektarbeit von besonderem Interesse, nicht zuletzt deshalb, weil dadurch der Einsatz eines ebenfalls SMP-fähigen Betriebssystems möglich gemacht wird, und damit auch die Parallelisierung auf Softwareebene. Das Cortex A9 Multiprozessor Subsystem beinhaltet zwei Cortex A9 CPUs. Jeder CPU steht ein 32 KByte Level 1 Instruktions-Cache und ein 32 KByte Level 1 Daten-Cache zur Verfügung. Weiterhin teilen sich die beiden CPUs insgesamt 1 Mbyte verteilten Level 2 Cache in einem gemeinsam genutzten Adressraum.

1.2.2 PandaBoard

Die technische Realisierung dieser OMAP 4433 Architektur wird von einem Entwicklungsboard namens PandaBoard umgesetzt und für den Markt zugänglich gemacht. Im Rahmen dieser Projektarbeit wurden bereits drei PandaBoards angeschafft. Des Weiteren werden vom Hersteller des Entwicklungsboards bereits die vorkompilierten Betriebssysteme Ubuntu und Pandroid inklusive Board Support Packages und Hardware Support für die OMAP 4430 Plattform zur freien Verfügung angeboten.

2 Vergleichbare Arbeiten

2.1 IMEC MPSoC Programming Approach

Jean-Yves Mignolet und Roel Wuyts vom Interuniversity Microelectronics Centre IMEC in Belgien stellen in ihrer Arbeit [Jean-Ives Mignolet (2009)] eine Reihe von Software Werkzeugen für die automatische Parallelisierung von sequentiellem C Code vor. Die Motivation für ihre Arbeit sehen sie darin, den Embedded Entwickler weitestgehend darin zu unterstützen, der Herausforderung zu begegnen, ein Model für die parallele Programmierung zu benutzen, welches optimal auf die jeweilige MPSoC Plattform zugeschnitten ist und ihr volles Leistungspotential ausnutzen kann. Denn laut ihrer Meinung steht der Embedded Entwickler bei der parallelen Softwareentwicklung für eine MPSoC Plattform vor zwei grundsätzlichen Problemen. Zum einen ist das Parallelisieren von bestehendem sequentiellem C Code von Hand sehr umständlich, zeitaufwendig und fehlerträchtig, weil die üblichen Hürden wie Interprozess-Kommunikation und Synchronisation, Race-Conditions oder Deadlockfreiheit schon auf ihn warten. Zum anderen dem Begegnen der plattform- und architekturspezifischen Eigenschaften der MPSoC-Plattform, wie z.B. eine jeweils sehr spezielle Speicherhierarchie oder verschiedene Hardwarekomponenten und On-Chip-Ressourcen. Über Jahre hinweg arbeiten die Forscher am IMEC daran, Softwareentwicklung auf MPSoC Plattformen zu erleichtern. Das Ergebnis ist eine softwarebasierte Werkzeugkette, um Applikationen auf MPSoC Plattformen abzubilden.

Abbildung 2.1 veranschaulicht die Werkzeugkette, die ein sequentieller C-Code durchläuft bis daraus parallel ausführbarer Code für die jeweilige Zielplattform generiert wird. Generell werden die einzelnen Tools in Cleaning Tools und Mapping Tools unterteilt. Zuallererst durchläuft der sequentielle C-Code das Cleaning Tool CleanC, das ANSI C-Code als Input akzeptiert. CleanC beinhaltet eine Menge von Richtlinien und Regeln, wie C-Code geschrieben werden soll um bei der Analyse später die optimale Genauigkeit zu erlangen. Mit Hilfe von CleanC kann der Entwickler den sequentiellen C-Code gegen diese Richtlinien testen und “unsaubere“ wie auch für die Parallelisierung ungeeignete Codeabschnitte herausfiltern. Ein typisches Beispiel für “unsauberen“ Code sind globale Variablen. Anschließend durchläuft der nun saubere C-Code zwei verschiedene Mapping Tools, zum einen das Memory Hierarchy (MH) Tool und den MPSoC Parallelizing Assistant (MPA). Das MH Tool ist zuständig für das optimale Ausnutzen der Speicherhierarchie, indem die passenden Größen von Datenblöcken und Datenblock-Transfers ermittelt werden. Dazu benötigt das MH Tool zusätzlich zwei weitere Inputs, zum einen eine genaue Beschreibung der Speicherhierarchie (welche Speicherpartitionen existieren, ihre Eigenschaften und wie sie verbunden sind), zum anderen ein Profil der Applikation für optimales Scheduling durch

Prefetchten von Datenblöcken. Letztere Informationen werden während der Analysephase durch die Cleaning Tools übrigens automatisch ermittelt. Die eigentliche Parallelisierung findet mit Hilfe des MPA Tools statt. Der Kernaspekt dahinter ist, dass es sich hierbei nicht etwa um einen Compiler handelt, der alle Optimierungen automatisch vornimmt, sondern um ein Hilfswerkzeug für den Entwickler, mit dessen Hilfe er diejenigen Codeabschnitte identifizieren und kennzeichnen kann, die parallelisiert, also auf mehrere Threads aufgeteilt werden sollen. Solche Abschnitte werden Parsections genannt. Für jede Parsection legt der Entwickler die Anzahl der ausführenden Threads fest. Es bleibt ebenfalls dem Entwickler überlassen, über die Art der Parallelisierung zu entscheiden, z.B. ob es sich um mehrere voneinander unabhängige und parallel ablaufende Worker-Threads handelt (funktionale Parallelität) oder Datenparallelität, wie z.B. Schleifeniterationen. Mit Hilfe aller vom Entwickler angegebenen Direktiven wird das Programm vom MPA Tool parallelisiert. Leider findet sich keinerlei Information darüber, wie die automatische Parallelisierung technisch realisiert wird.

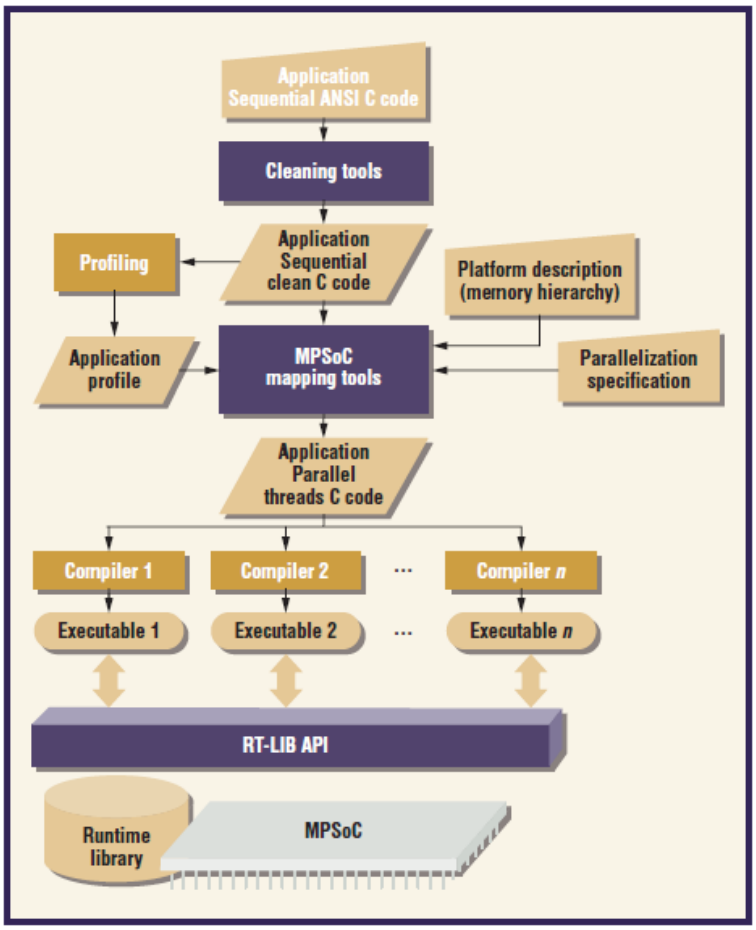

Abbildung 2.1: TI OMAP 4 Workflow IMEC Toolchain

2.2 Efficient OpenMP Support and Extensions for MPSoCs with Explicitly Managed Memory Hierarchy

In der Arbeit [Andrea Marongiu (2009)] von Andrea Marongiu und Luca Benini an der Universität von Bologna wird eine Erweiterung des parallelen Programmiermodells OpenMP speziell für MPSoC-Architekturen mit Scratchpad Speicher (SPM) und ohne Cache-Kohärenz vorgestellt. Dieses Framework ist eine optimierte Implementierung der Compiler- und Laufzeitinfrastruktur des herkömmlichen OpenMP-Modells mit spezifischen Erweiterungen, die die Speicherhierarchie der Zielplattform optimal und vorteilhaft nutzen. OpenMP wird normalerweise auf SMP-fähigen Maschinen mit Shared-Memory Architekturen genutzt. Doch Marongiu und Benini haben festgestellt, dass eine wirklich effektive Implementierung auf MPSoC-Plattformen mit OpenMP aus folgenden drei Gründen nicht immer möglich sei. Erstens seien MPSoCs typischerweise heterogene Architekturen mit verschiedenen Prozessorkomponenten, die nicht alle zusammen von einem einzigen Betriebssystem verwendet werden. Zweitens sei die Speicherhierarchie viel komplexer, z.B. bei einer Distributed Shared Memory Architektur. Schließlich sollte die Implementierung von Synchronisierungsdirektiven sehr sorgfältig optimiert werden, weil sie die Performance des parallelisierten Codes stark beeinflussen. Des Weiteren wurde die OpenMP Runtime Library (libgomp) nicht nur erweitert sondern in einigen Teilen auch umgeschrieben. Während bei der herkömmlichen OpenMP-Implementierung die Direktive #pragma omp parallel in die POSIX-Funktion pthread_create umgesetzt wird, verfolgt die hier vorgestellte Erweiterung ein Master-Worker Schema, wobei der Master-Thread die main() der Applikation aufruft, den seriellen Codeabschnitt ausführt und die Workers solange an einer Grenze (barrier) warten. Die selbe o.g. Direktive wird umgesetzt in die neue Funktion do_parallel(), und es werden ihr zwei Argumente übergeben, die Adresse des parallel auszuführenden Codes und den Adressbereich, in dem die verteilten Variablen platziert sind. Ab diesem Punkt hebt der Master die barrier auf, und die Worker-Threads führen den parallelen Code aus. Die Master- und Slave-Threads basieren auf den Prozessor-IDs. Der technische Kernaspekt dieser Arbeit liegt im Benutzen des Scratchpad Speichers (SPM) für die Parallelisierung. Da der SPM sich in unmittelbarer Nähe zur CPU befindet, und damit in der Speicherhierarchie ganz oben, noch vor dem Level 1 Cache liegt, garantiert er die geringste Zugriffslatenzzeit.

Abbildung 2.2 beschreibt die gesamte Werkzeugkette. Das Prinzip dahinter ist, dass größere verteilte Speicherabschnitte wie z.B. ein Array bei Bedarf partitioniert und auf verschieden große phyikalische Speicheradressen abgebildet werden. Während eines ersten Probedurchlaufs in einem Simulator wird jeder Zugriff von jedem Prozessor beobachtet. Das Ergebnis dieser Beobachtungsphase ist eine Tabelle mit der Menge und Frequenz an Zugriffen für jeden Speicherabschnitt von jedem beteiligten Prozessor aus. Basierend auf diesen Zugriffsinformationen wird das Array im Allokierungsalgorithmus in die am meisten geeigneten verschiedenen Speicherabschnitte partitioniert, sodass jedem Prozessor genau derjenige Speicherabschnitt zugeteilt wird, auf den er im vorherigen Probedurchlauf am meisten zugegriffen hat. Das Ergebnis dieser Zuteilung wird ebenfalls in einer Tabelle festgehalten, mit den jeweiligen Anfangs- u. Endadressen für jeden Prozessor. Erst da-

nach wird geprüft, ob das gesamte Array in die SPM Blöcke der beteiligten Prozessoren passen. Ist das der Fall, wird das gesamte Array in den SPM geladen, falls nein, werden die zuvor ermittelten einzelnen Speicherblöcke an die jeweiligen SPM verteilt. Der gesamte Mechanismus geschieht automatisch und bleibt dem Benutzer verborgen.

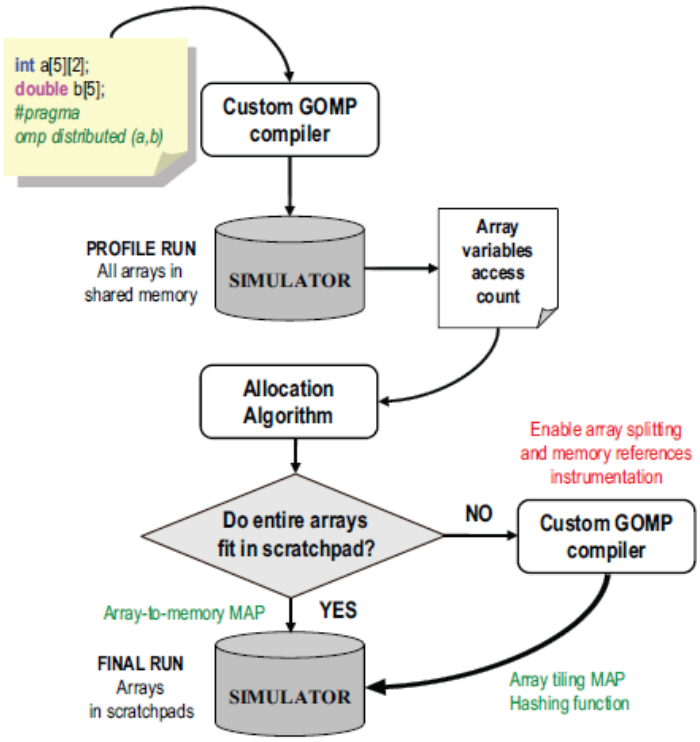

Abbildung 2.2: OpenMP for MPSoC Workflow

2.3 A Cross-Domain Multiprocessor System-on-a-Chip for Embedded Real-Time Systems

Hermann Kopetz, Roman Obermaisser und Christian Paukovits von der Technischen Universität Wien stellen in ihrer Arbeit GENESYS MPSoC [Kopetz (2010)] ein Modell für eine generische, branchenübergreifende MPSoC Plattform vor für die komponentenbasierende Entwicklung von Embedded Echtzeitsystemen. Im Vordergrund steht hierbei die komponentenorientierte Architketur, indem verschiedene Hardware-Ressourcen mit Software in Komponenten zusammengefasst werden, und alle Komponenten zusammen sich zu einer Multiprozessor-Architektur ergeben. Die Motivation für ihre Arbeit sahen sie darin, dem nahezu unübersichtlichen Wald von zu vielen branchenspezifischen Standards im Embedded Bereich durch Standardisierung und Konsolidierung zu begegnen. Ihr Ziel ist es, mit einer branchenübergreifenden Plattform eine Standardisierung zu erreichen, um nicht ständig

das Rad neu erfinden zu müssen in den einzelnen Branchen. Ihrer Meinung nach stehen die Embedded Entwickler in den einzelnen Branchen vor denselben Problemen und Herausforderungen in vielen Bereichen, was den Wunsch nach einem gemeinsamen Nenner hervorruft. Diese Arbeit soll eine gemeinsame Basis schaffen und wurde in interdisziplinärer Zusammenarbeit mit Vertretern aus verschiedenen betroffenen Industriezweigen wie Automobilindustrie, Avionics, Anlagen- u. Maschinenbau, Medizintechnik, Verbraucherelektronik und Mobile Devices gemeinsam entworfen.
In diesem Kontext wird eine Komponente geformt, indem ein Software Image, z.B. eine Funktionalität oder ein Service der Applikation in eine ausführbare Hardware-Einheit geladen wird. Hierbei ergibt sich noch die Möglichkeit, innerhalb einer solchen Komponente eine auf der Hardware aufsetzende hierarchische Struktur zu bilden, z.B. ein lokales Betriebssystem, eine Runtime Library, Middleware und schließlich die Applikation. Aus architektonischer Sichtweise betrachtet führt eine solche Komponente typischerweise einen Service, also eine logisch zusammenhängende, in sich abgeschlossene funktionale Einheit aus.

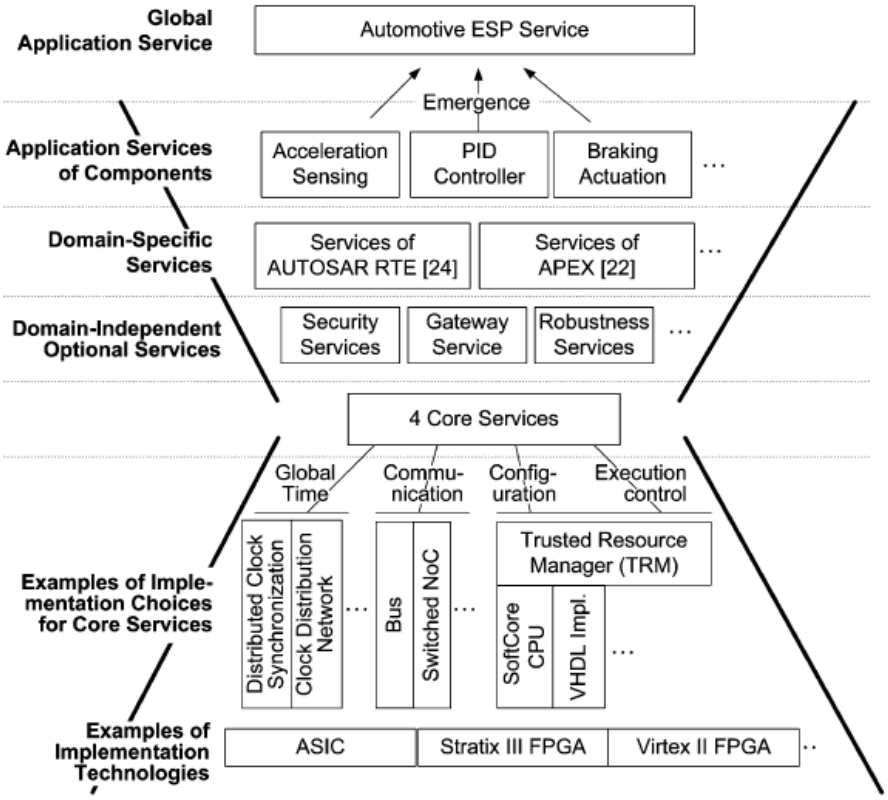

Abbildung 2.3: GENESYS MPSoC

Abbildung 2.3 zeigt den strukturellen Aufbau der GENESYS MPSoC Architektur am Beispiel einer Implementierung eines Antiblockiersystems (ABS) im Automotive Bereich. In der oberen Hälfte sind die Services hierarchisch angeordnet, die für die Implementierung des globalen ABS-Services definiert und eingesetzt werden, während in der unteren Hälf-

te einige Beispiele von Komponenten zu sehen sind, die die Services implementieren und auf der MPSoC Plattform technisch realisieren. Die “Taille“ in der Mitte werden von 4 Core Services gebildet. Diese Services sind in allen beteiligten Industriebranchen erforderlich, was als Resultat der interdisziplinären Zusammenarbeit hervorging. Sie bilden die Basis für die Implementierung aller übergeordneter Services. Global Time ist für die globale Clocksynchronisation und Zeitgebung auf der gesamten Plattform zuständig. Die Kommunikation zwischen Komponenten findet über Interfaces durch den Communication Service statt. Mit dem Configuration Service können Software Images auf Hardwarekomponenten geladen werden. Schließlich wird mit dem Execution Control Service die Ausführung von Komponenten gesteuert und verwaltet. Eine Ebene über den Core Services befinden sich die Plattform-Services. Diese werden in zwei Typen unterteilt: Die Domain-independent optional services zum einen bieten erweiterte Funktionalitäten an, die in vielen aber nicht allen Industriebranchen benötigt werden. Sie werden in einer “Ready to use“-Form bereitgestellt. Es obliegt dem Entwickler, zu entscheiden, welche dieser Services verwendet werden sollen. Die Domain-specific optional Services sind spezialisierte Services der jeweiligen Industriebranchen, wie z.B. die AUTOSAR RTE [GbR (2006)] oder ARTEMIS. Die GENESYS MPSoC Architektur bietet die Möglichkeit, die Plattform von den Core Services nach oben ausgehend für die benötigten Anforderungen immer weiter zu spezifizieren. In wird dieses Prinzip Bottom-up-custumizing bezeichnet.

3 Schluss

Die OpenMP-Erweiterung aus der Arbeit von [Andrea Marongiu (2009)] wird in den folgenden Projektarbeiten zur Parallelisierung von Software eingesetzt, denn glücklicherweise gehört die OMAP-Plattform zu den geeigneten Zielplattformen für die OpenMP-Erweiterung. Scratchpad-Speicherblöcke sind ebenfalls Bestandteil der ARM-Architektur. Vorausgesetzt wird die genaue Kenntnis der Speicherhierarchie der OMAP-Plattform, um die Erweiterung nutzen zu können. Für die eigene Arbeit verdeutlicht diese Arbeit, dass es bei der Entwicklung auf einer MPSoC-Plattform von großer Wichtigkeit und Notwendigkeit ist, einen genauen Überblick über die Speicherhierarchie der Zielplattform zu gewinnen, z.B. welche Speicherblöcke und Speicherarten vorhanden sind, welcher Komponente sie zur Verfügung stehen oder über welchen Adressbereich sie ansprechbar sind. Die Speicherhierarchie wird in einer sog. Pagetable mit vorgegebener Syntax dokumentiert. Das Erstellen einer solchen Pagetable von Hand ist mühsam, jedoch gibt es eine Reihe von ARM-Tools, die hierbei die Arbeit erleichtern. Eine Reihe von Testläufen, in denen die OpenMP-Erweiterung mit der herkömmlichen OpenMP-Implementierung bei identischer Parallelisierungsdirektive verglichen wird, wird Aufschluss darüber geben, wie groß die Unterschiede in puncto Speedup sind. Leider wird der Leser darüber im Unklaren gelassen.

Das Prinzip der komponentenbasierten und Service-orientierten Architektur in GENESYS MPSoC [Kopetz (2010)] wird ebenfalls in die kommenden Projektarbeiten mitgenommen, denn es erweist sich im Bereich der Embedded Entwicklung als vorteilhaft, nicht zuletzt aufgrund der engeren Verschmelzung von Software und Hardware in Embedded Systemen oder dem Vorhandensein mehrer IPs auf einer MPSoC-Plattform. Jedoch wird es schwierig bis unmöglich sein, eine Architektur wie GENESYS vollständig auf eine schon bestehende Plattform zu portieren, wie in diesem Fall die OMAP-Plattform. Die realisierende Hardware und die Architektur müsssen aufeinander abgestimmt sein, um z.B. sicherheitskritische Aspekte oder Robustheit zu gewährleisten und Performancegewinn zu erreichen. Ebenfalls ist in der eigenen Projektarbeit die Ausgangssituation verschieden, denn bereits bestehende Software muss parallelisiert werden und auf eine bereits bestehende MPSoC-Plattform portiert werden. Dennoch bieten die vorhandenen IPs auf der OMAP-Plattform die Möglichkeit, verschiedene Komponenten zu definieren und in einer Architektur zu strukturieren, die die Applikation und die Plattform zusammenbringt.

Abbildungsverzeichnis

1.1 TI OMAP 4 Plattform 6

2.1 TI OMAP 4 Workflow IMEC Toolchain 9
2.2 OpenMP for MPSoC Workflow 11
2.3 GENESYS MPSoC 12

Literaturverzeichnis

[Andrea Marongiu 2009] ANDREA MARONGIU, Luca B.: *Efficient OpenMP Support and Extensions for MPSoCs with Explicitly Managed Memory Hierarchy*. 2009. – URL https://portal.acm.org/citation.cfm?id=1874620.1874819&coll=DL&dl=ACM&CFID=20908893&CFTOKEN=23003158

[GbR 2006] GBR, AUTOSAR: *AUTOSAR Specification of RTE Software V1.0.1*. 2006. – URL http://www.autosar.org

[Jean-Ives Mignolet 2009] JEAN-IVES MIGNOLET, Roel W.: *Embedded Multiprocessor Systems-on-Chip Programming*. 2009. – URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4814956&tag=1

[Kopetz 2010] KOPETZ, Hermann: *A Cross-Domain Multiprocessor System-on-a-Chip for Embedded Real-Time Systems*. 2010. – URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5570994

[openmp.org 2009] OPENMP.ORG: *Open MP*. 2009. – URL http://openmp.org/wp/

[Ziegler 2010] ZIEGLER, Stephan: *Hoher Forschungsbedarf bei Embedded Systems*. 2010. – URL http://www.produktion.de/maerkte/hoher-forschungsbedarf-bei-embedded-systems/

[ZVEI 2010] ZVEI: *Nationale Roadmap Embedded Systems*. 2010. – URL http://www.bitkom.org/files/documents/NRMES_2009_einseitig.pdf

ZVEI (2010) Ziegler (2010) Jean-Ives Mignolet (2009) Andrea Marongiu (2009) openmp.org (2009) Kopetz (2010) GbR (2006)