Oliver Arnold

Dynamisches Task-Scheduling in heterogenen Mehrprozessorsystemen

Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 65

Oliver Arnold

Dynamisches Task-Scheduling in heterogenen Mehrprozessorsystemen

VOGT

Dresden 2014

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.

Bibliographic Information published by the Deutsche Bibliothek
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibiograpic data is available in the internet at http://dnb.ddb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2014

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Dynamisches Task-Scheduling in heterogenen Mehrprozessorsystemen“ von Oliver Arnold überein.

Gesetzt vom Autor

ISBN 978-3-938860-77-9

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

Technische Universität Dresden

Dynamisches Task-Scheduling in heterogenen Mehrprozessorsystemen

Oliver Arnold

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil. Leon Urbas
Gutachter:	Prof. Dr.-Ing. Dr. h.c. Gerhard Fettweis
	Prof. Dr.-Ing. Jürgen Teich
Tag der Einreichung:	10.10.2013
Tag der Verteidigung:	17.01.2014

Danksagung

Diese Arbeit entstand während meiner Zeit am Vodafone Stiftungslehrstuhl für Mobile Nachrichtensysteme der Technischen Universität Dresden. Ich möchte mich bei allen Mitgliedern der Hardware-Arbeitsgruppe bedanken, mit denen ich zusammengearbeitet habe und die immer zu einer erfrischenden Diskussion bereit waren. Für die fachlichen Anregungen, Betreuung und Unterstützung für diese Arbeit, möchte ich meinem Doktorvater Professor Gerhard Fettweis herzlich danken. Des Weiteren danke ich Professor Jürgen Teich von der Friedrich-Alexander-Universität Erlangen-Nürnberg sehr herzlich für die Begutachtung meiner Dissertation.

Ein großer Dank gilt meiner Familie für ihre Unterstützung meiner Arbeit und Tätigkeit an der Technischen Universität Dresden.

Des Weiteren bedanke ich mich beim Tomahawk2-Team, insbesondere bei Benedikt Nöthen, Steffen Kunze, Esther Pérez-Adeva, Emil Matúš, Tobias Seifert, Erik Fischer, Holger Eisenreich und Sebastian Höppner. Ihr unermüdliches Engagement haben dieses Projekt erst möglich gemacht.

Abstract

Simultaneously improving both, computational power and energy efficiency, represents one of the major challenges in the development of future hardware systems. In this regard, the systems' manufacturing process as well as the hardware architecture must be examined. Concerning architecture design, the combination of different types of processing elements (PEs) in a single chip, is a recent trend. A smart management of these PEs allows an increase in computational power as well as energy efficiency.

In this work a dedicated scheduling unit called CoreManager is proposed to control heterogeneous Multiprocessor Systems-on-Chips (MPSoCs). This unit is responsible for dynamically distributing atomic tasks on different PEs as well as managing the inherent data transfers. For this purpose, a runtime analysis of the data dependencies is firstly performed. Based on this analysis, a schedule is created to allocate the PEs and explicitly reserve and administrate the local memories. The results of the dynamic data dependency analysis are additionally reused in order to increase data locality. In particular, required data are kept in the local memories, thus reducing the number of transfers from the global memories.

The CoreManager has been profiled in order to expose the most time consuming components. Dynamic data dependency check was found to be the limiting factor regarding system scalability. An extension of the instruction set architecture of the CoreManager has been developed and integrated to solve this issue. This extension allows speeding-up the data dependency check process and other CoreManager components, thus increasing performance while reducing energy consumption.

A battery-aware mode of operation of the CoreManager is introduced, which allows extending the lifetime of the system. Furthermore, a failure-aware dynamic task scheduling approach for unreliable heterogeneous MPSoCs was integrated in the CoreManager. It enables a detection and isolation of erroneous PEs, connections and memories.

By applying these approaches the efficient management of heterogeneous MPSoCs is enabled.

Zusammenfassung

Eine Erhöhung der Rechenleistung bei einer gleichzeitigen Verbesserung der Energieeffizienz ist eine der Hauptaufgaben bei der Entwicklung zukünftiger Hardwaresysteme. Hierbei muss sowohl der Herstellungsprozess als auch die Hardware-Architektur betrachtet werden. Ein vielversprechender Ansatz für eine Verbesserung der Hardware-Architektur besteht darin, mehrere heterogene Prozessoren auf einem Chip zu integrieren. Hierbei kann, bei einer intelligenten Verteilung der Aufgaben, sowohl die Rechenleistung als auch die Energieeffizienz erhöht werden.

In dieser Arbeit wird ein dedizierter Task-Scheduler, genannt CoreManager, konzipiert und entworfen. Dessen Aufgabe ist es, atomare Tasks dynamisch in einer heterogenen Plattform auf Prozessorelemente (PE) zu verteilen. Die Datenabhängigkeiten der Tasks werden zur Laufzeit analysiert. Anhand dieser Abhängigkeiten wird ein Schedule aufgebaut. Prozessorelemente werden daraufhin allokiert und der lokale Speicher explizit reserviert und verwaltet. Ein Management der Datentransfers erlaubt eine Erhöhung der Datenlokalität. Dies wird durch eine Wiederverwendung der Resultate der dynamischen Datenabhängigkeitsanalyse erreicht.

Eine Analyse der Abarbeitung des CoreManagers hat ergeben, dass die dynamische Datenabhängigkeitsanalyse der zeitaufwendigste Teil der Abarbeitung ist. Um die Skalierung des CoreManagers zu verbessern, wurde dieser um einen anwendungsspezifischen Befehlssatz erweitert. Die weiteren Komponenten des CoreManagers wurden ebenfalls beschleunigt. Bei einer batteriebetriebenen Abarbeitung erlaubt es eine Anpassung der Verfahren im CoreManager, die Systemlaufzeit zu verlängern. Des Weiteren erfolgte eine Anpassung des CoreManagers, um eine fehlerfreie Abarbeitung von Tasks sicherzustellen. Hierbei werden fehlerhafte Prozessoren, Verbindungen und Speicher erkannt, markiert und gegebenenfalls isoliert.

Unter Verwendung dieser Ansätze und Verfahren ist ein effizientes Management von heterogenen MPSoCs möglich.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
AES	Advanced Encryption Standard
ADPLL	All Digital Phase-Locked Loop
APP	Applikationsprozessor
ASAP	As-Soon-As-Possible
ASIC	Application-Specific Integrated Circuit
ASIP	Application Specific Integrated Processor
ATE	Produkt aus Fläche, Zeit und Energie
AWGN	Additive white Gaussian noise
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	circa
Cell BE	Cell-Broadband-Engine
CM	CoreManager
CMOS	Complementary Metal Oxide Semiconductor
CUDA	CompUte Driver Architecture
DAG	gerichteter azyklischer Graph
DDAA	Dynamische Datenabhängigkeitsanalyse
DFG	Datenflussgraph

d. h.	das heißt
DMA	Direct Memory Access
DMAC	Direct Memory Access Controller
DSM	Distributed Shared Memory
DSP	Digitaler Signalprozessor
EIB	Element Interconnect Bus
FEC	Forward Error Correction
FFT	Schnelle Fourier-Transformation
FIFO	First In-First Out
FIR	Finite Impulse Response
FLIX	Flexible Length Instruction Extensions
FPGA	Field Programmable Gate Array
FSM	Finite State Machines
GPP	General Purpose Processor
GSM	Global System for Mobile Communication
GTU	globale Task-Unit
JPEG	Joint Photographic Experts Group
KPN	Kahn Prozessnetzwerke
LTE	Long Term Evolution
LTU	lokale Task-Unit
MeP	Media embedded Processor
MFC	Memory Flow Controller
MHz	Megahertz
MISP	Multiple Instructions Stream Prozessor
MPI	Message Passing Interface
MPSoC	Multi-Processor System-on-Chip
NoC	Network-on-Chip
NUMA	Non-Uniform Memory Access
oDL	Ohne Datenlokalität
OmpSs	OpenMP Superscalar
OMT	Object Mapping Tabellen
OpenCL	Open Compute Language
OpenMP	Open Multi-Processing

PE	Prozessorelement
PPE	Power Processing Element
QM	Queue Manager
RISC	Reduced Instruction Set Computer
RO	Read-Only
RTL	Register Transfer Level
RW	Read-Write
SD	Sphere Detection
SDF	Synchrone Datenflussgraphen
SDR	Software Defined Radio
SIMD	Single Instruction Multiple Data
SMP	symmetrisches Multiprozessorsystem
SMPS	SMP Superscalar
SoC	System-on-Chip
SPE	Synergistic Processing Elemente
SPMD	Single Program Multiple Date
SPU	Synergistic Processing Unit
StarSs	STAR superscalar
Szen.	Szenario
TGFF	Task Graphs For Free
TIE	Tensilica Instruction Extension
TL	Tasklaufzeit
TRS	Task-Reservation Station
TSMC	Taiwan Semiconductor Manufacturing Company
TUD	Technische Universität Dresden
TU Dresden	Technische Universität Dresden
UART	Universal Asynchronous Receiver Transmitter
UMC	United Microelectronics Corporation
UMTS	Universal Mobile Telecommunication System
VBN	Verbindungsnetzwerks
VCD	Value Change Dump
VDSP	Vektor-DSP
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
Vrgl.	Vergleich

WFI	Wait-for-Interrupt
XML	Extensible Markup Language
XPG	Xtensa Processor Generator
XTSC	XTensa SystemC
z. B.	zum Beispiel

Symbolverzeichnis

α	Batterie-Parameter: Batteriekapazität
β	Batterie-Parameter: Nichtlinearität
c_{Konst}	Konstanter Mehraufwand
I_k	Konstante Last
n_{IN}	Anzahl der Eingangsdatentransfers
n_{OUT}	Anzahl der Ausgangsdatentransfers
n_{TiS}	Anzahl der Tasks im System
$n_{Transfers}$	Anzahl der Eingangs- und Ausgangsdatentransfers
n_{ws}	Größe des Taskfensters
t_{Dep}	Zeit: Dynamische Datenabhängigkeitsanalyse
$t_{DMAC\_Allokation}$	Zeit: DMAC Allokation
t_{DMAC_Config}	Zeit: DMAC Konfiguration
$t_{Interrupt}$	Zeit: Interrupt Routine
$t_{Mehrbelastung}$	Zeit: Mehrbelastung
$t_{Nachfolger\_Bearbeiten}$	Zeit: Nachfolger-Tasks bearbeiten
$t_{PE\_Allokation}$	Zeit: PE Allokation
$t_{Speicher\_Allokation}$	Zeit: Allokation von lokalem Speicher
t_{Task_END}	Zeit: Bearbeitung eines Tasks nach dessen Start
t_{Task_START}	Zeit: Bearbeitung eines Tasks vor dessen Start
t_{Task_ZEIT}	Zeit: Bearbeitung eines Tasks
$t_{TaskBeschreibung}$	Zeit: Transfer der Taskbeschreibung
$t_{Tasklaufzeit}$	Tasklaufzeit
V_{Mode}	Spannung für die Aktivierung der CoreManager-Verfahren
V_{Tot}	Totspannung einer Batterie

Kapitel 1

Einleitung

1.1 Motivation

Zukünftige technische Systeme verlangen eine immer höhere Rechenleistung bei gleichzeitiger Reduzierung des Energieverbrauches. Dies kann einerseits durch eine Verbesserung in der Herstellung, z. B. durch eine reduzierte Strukturbreite der Fertigungstechnologie und durch spezielle Prozesse für niedrige Energieverbräuche, realisiert werden. Andererseits können Verbesserungen durch Anpassung der Komponenten, der Architektur und der Laufzeitumgebung des Systems erzielt werden.

Ein Ansatzpunkt, um gleichzeitig sowohl eine Erhöhung der Rechenleistung als auch die Reduktion des Energieverbrauchs zu erreichen, ist die Veränderung der Chiparchitektur. Hierbei werden anstatt eines Prozessors mehrere auf dem Chip integriert. Dieser Schritt folgt logisch aus der Tatsache, dass die dynamische Verlustleistung eines Prozessors quadratisch mit der Versorgungsspannung, aber nur linear mit der Taktfrequenz, anwächst. Eine Verteilung der Rechenleistung auf mehrere Prozessoren erlaubt demzufolge eine Reduzierung der Taktfrequenz bei gleichzeitiger Senkung der Versorgungsspannung. Eine Reduzierung der Leistungsaufnahme ist somit möglich. Der Trend, mehrere Prozessoren auf einem Chip zu integrieren, wird sich in zukünftigen Systemen weiter fortsetzen [ITR08]. Im Zeitraum zwischen 2010 und 2022 wird erwartet, dass sich die Anzahl der Prozessoren auf einem Chip, um ungefähr den Faktor 20 erhöht. Hinsichtlich der Leistungsaufnahme von portablen Geräten, wird eine Verdreifachung im selben Zeitraum prognostiziert. Des Weiteren steigen die Anforderungen an die Wiederverwendung von Komponenten eines Chips. Daher wird auch eine Zunahme der Wiederverwendbarkeit der Bauelemente von 50 Prozent im Jahr 2010 auf 94 Prozent im Jahr 2022 erwartet. Allerdings wirft der praktische Übergang, bezüglich der Verwendung von einem Prozessor hin zu einem Mehrprozessorensystem auf einem Chip (MPSoC), viele Fragen auf, so z. B.: Wie können mehrere Prozessoren auf einem Chip integriert werden? Müssen die Prozessoren homogen sein? Wie kann eine Synchronisation und ein Datenaustausch zwischen den Prozessoren realisiert werden? Kann man existierende Prozessoren wiederverwenden oder ist eine Anpassung dieser notwendig? Wie kann die vorhandene Rechenleistung der Applikation so zur Verfügung gestellt werden,

dass keine Kenntnis der Hardware benötigt wird? Wie skalieren sowohl die Hardware als auch die Software? Wie kann die Leistungsaufnahme reduziert und reguliert werden? Diese Liste von Fragen kann beliebig erweitert werden. Sie zeigt gleichzeitig die Komplexität der Aufgabe und deren inhärenten Problemstellungen, welchen sich die Forschung nun stellen muss. Gerade und insbesondere hinsichtlich der Weiterentwicklung bestehender Systeme hin zu zukunftsweisenden.

Die Laufzeitumgebung des Systems bestimmt die Abarbeitung der Applikation. Hierzu gehören Verfahren, die Daten und Aufgaben nach zeitlichen Vorgaben an die Prozessoren zu verteilen. Eine Verbesserung der Verfahren ist z. B. möglich, indem die Datenlokalität erhöht wird. Hierbei kann, ohne einen Eingriff in die Applikation oder die Hardware, eine Steigerung der Leistungsfähigkeit des Systems erreicht werden. Der in einigen Systemkonfigurationen auftretende Flaschenhals beim Zugriff auf den Speicher, welcher durch das Memory Wall beschrieben wird [ABC+06], verringert sich.

Auf der Ebene der Applikation für Terminals ist es möglich, Modem- und Nutzeranwendungen auf einem Chip zu vereinen und parallel abzuarbeiten. Es wird beiden Seiten ein kontrollierter Zugriff auf alle Ressourcen des Systems gewährt. Im EU Projekt *ICT-eMuCo* [EMU10] wurde dieser Ansatz untersucht. Resultat war, dass sowohl Nutzer-Applikationen wie z. B. die JPEG-Dekodierung als auch die Verarbeitung von Basisband-Algorithmen in gleichzeitiger Ausführung ermöglicht wird. Hierbei kann eine Abarbeitung mit einer Priorisierung bestimmter Tasks verwendet werden. Die Anforderung an die Laufzeitumgebung steigt in diesem Fall, da sich die Parallelitätsprofile [RLAF09] der Applikation unterscheiden und deren Startzeitpunkte ungewiss sind. Dadurch wird für diese eine dynamische Zuordnung von Ressourcen notwendig, um eine effiziente Nutzung des Systems zu ermöglichen. Des Weiteren kann für bestimmte Applikationen keine Vorhersage bezüglich der Laufzeit eines Task gemacht werden. Für batteriebetriebene Geräte ist eine Adaption notwendig, um deren Laufzeit zu verlängern.

Für den zellularen Mobilfunk werden Basisstationen benötigt, deren Betrieb einen hohen Energiebedarf aufweist [FZ08]. Zudem ist der relative Anteil des Energieverbrauches für die Basisbandverarbeitung größer, je kleiner die Basisstation ist [ARFB10]. Die zu erwartende Steigerung in der Nutzung von kleinen Basisstationen für Femto- und Mikrozellen, wird eine effiziente Abarbeitung des Basisbandes unumgänglich werden lassen. In Basisstationen führt ein dynamisches Management der Prozessoren und Speicherinhalte zu einer Reduzierung der Leistungsaufnahme. Innerhalb des Projektes *Cool BaseStations* wurde dieser Ansatz verfolgt und erfolgreich umgesetzt [COO11]. Das Resultat ist eine adaptive Anpassung des Energieverbrauches an die Last der Basisstation. Die Energieeffizienz wird hierbei gesteigert.

1.2 Ziele der Arbeit

Die Ziele dieser Arbeit sind:

- Konzept und Entwurf eines dedizierten dynamischen Task-Schedulers für heterogene Mehrprozessor-Architekturen; Analyse des Verhaltens und Bestimmung der Limitie-

rung wesentlicher Systemparameter wie z. B. der Skalierung über die Anzahl der Prozessoren

- Entwicklung einer modularen Simulationsplattform und dazugehörige Werkzeuge für experimentelle Analysen des taskbasierten Schedulers
- Konzept und Entwurf einer funktional korrekten, abstrakten Simulationsumgebung, um Applikationen mit einem taskbasierten Programmiermodell zu entwerfen
- Entwurf von Entwicklungswerkzeugen für die Bearbeitung des Quellcodes, der Analyse der Abarbeitung und für die Visualisierung des Verhaltens von Applikationen
- Analyse wesentlicher Komponenten eines dynamischen Task-Schedulers und Beschleunigung dieser unter Verwendung eines anwendungsspezifischen Befehlssatzes
- Konzept, Entwurf und Untersuchung eines Task-Schedulers mit verteilter Architektur
- Konzept und Entwurf einer universellen Debug-Umgebung sowohl für Simulationen, für FPGA Prototypen als auch für Chip-Implementierungen
- Entwicklung und Untersuchung eines Task-Scheduling Verfahrens, welches den Batteriezustand mit einbezieht
- Konzept, Entwurf und Untersuchung eines Task-Schedulers, welcher ein statisches Schedule abarbeiten kann
- Untersuchung und Entwurf eines ausfallsicheren Task-Schedulingverfahrens hinsichtlich Fehlern in der Abarbeitung von Tasks; des Weiteren ein Ansatz um dynamisch fehlerhafte Prozessoren und Speicher zu erkennen, zu markieren und gegebenenfalls zu isolieren
- Beschreibung des Zeitverhaltens wesentlicher Komponenten des Task-Schedulers mit analytischen Methoden
- Konzept und Entwurf eines abstrakten Modells des Task-Schedulers

1.3 Gliederung der Arbeit

Diese Arbeit untergliedert sich wie folgt: Die zum Verständnis benötigten Grundlagen der verwendeten Hardware-Architektur, das TaskC Programmiermodell und die Task-Schedulingeinheit, genannt CoreManager, werden im zweiten Kapitel vorgestellt. Im darauf folgenden Kapitel befindet sich ein Vergleich der hier gewählten Ansätze mit dem Stand der Technik. Im vierten Kapitel werden die verwendeten Entwicklungswerkzeuge vorgestellt, von den ein Großteil neu geschrieben wurde, und neben ihren Einsatzzwecken auch ihre Arbeitsweisen erläutert. Im fünften Kapitel wird die Softwareimplementierung des Task-Schedulers, genannt CoreManager, auf der Synopsys CoMET-Plattform eingeführt

und analysiert. Hierbei werden mehrere Anpassungen vorgenommen, um sowohl die Flexibilität als auch die Leistungsfähigkeit zu erhöhen. Das sechste Kapitel beschäftigt sich mit fehlerhaften Systemen. Hierbei wird ein Ansatz auf Systemebene vorgestellt, welcher zur Laufzeit Fehler erkennen und korrigieren kann. Fehlerhafte Module werden isoliert. Im siebten Kapitel wird eine Adaption des CoreManager-Konzeptes an ein batteriebetriebenes System vorgenommen. Dafür wird unter anderem ein Batteriemodell in das System eingeführt. Dies erlaubt eine gezielte Änderung des Schedulings und dessen Anpassung an den aktuellen Zustand der Batterie. Im folgenden Kapitel wird die Tensilica XTSC Umgebung verwendet um den Befehlssatz eines Prozessors für das Task-Management anzupassen und hierbei zu beschleunigen. Im darauf folgenden Kapitel wird die Chip-Implementierung des heterogenen Tomahawk2 MPSoCs vorgestellt. Der Fokus liegt hierbei auf dem CoreManager. Für diesen werden Messwerte dargelegt und mit anderen Ansätzen verglichen. In Kapitel zehn wird eine analytische Betrachtung des Systems und eine Modellierung des CoreManagers vorgestellt. Das abschließende Kapitel fasst die Arbeit zusammen und zeigt offene Punkte für weitere Forschungsmöglichkeiten auf.

Relevante eigene Publikationen

- ***Leistungsverbrauch von Makro- und Femtobasisstationen***

  Arnold, Oliver ; Richter, Fred ; Fettweis, Gerhard ; Blume, Oliver: Power consumption modeling of different base station types in heterogeneous cellular networks. In: *IEEE Future Network and Mobile Summit*, 2010

- ***Parallelitätsprofile von Applikationen und deren Anwendung***

  Ristau, Bastian ; Limberg, Torsten ; Arnold, Oliver ; Fettweis, Gerhard: Dimensioning heterogeneous MPSoCs via parallelism analysis. In *Design, Automation & Test in Europe Conference & Exhibition, DATE'09.*, 2009

- ***CoreManager und TaskC Programmiermodell***

  Arnold, Oliver ; Matus, Emil ; Noethen, Benedikt ; Winter, Markus ; Limberg, Torsten ; Fettweis, Gerhard: Tomahawk: Parallelism and heterogeneity in communications signal processing MPSoCs. In: *ACM Transactions on Embedded Computing Systems (TECS) - Special Issue on Design Challenges for Many-Core Processors*, 2014