Benedikt Nöthen

Untersuchungen von Kommunikationsmechanismen in heterogenen Mehrprozessorsystemen

Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 80

Benedikt Nöthen

Untersuchungen von Kommunikationsmechanismen in heterogenen Mehrprozessorsystemen

VOGT

Dresden 2015

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

Bibliographic Information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2015

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Untersuchungen von Kommunikationsmechanismen in heterogenen Mehrprozessorsystemen“ von Benedikt Nöthen überein.

Gesetzt vom Autor

ISBN 978-3-938860-98-4

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

Untersuchungen von Kommunikationsmechanismen in heterogenen Mehrprozessorsystemen

Benedikt Nöthen

von der Fakultät Elektrotechnik und Informationssystemtechnik

der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Jun.-Prof. Dr.-Ing. Peter Birkholz
Gutachter:	Prof. Dr.-Ing. Dr.h.c. Gerhard P. Fettweis
	Prof. Dr.-Ing. Ulrich Rückert

Tag der Einreichung:	02.07.2015
Tag der Verteidigung:	08.10.2015

Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Vodafone Stiftungslehrstuhl der Technischen Universität Dresden. In diesem Zusammenhang gilt mein ganz besonderer Dank dem Leiter des Lehrstuhls, Herrn Prof. Dr.-Ing. Dr. h.c. Gerhard Fettweis. Seine Anregungen und Förderungen meiner Arbeit sowie die außergewöhnlich guten Arbeitsbedingungen haben das Anfertigen dieser Dissertation ermöglicht.

Zudem danke ich ganz herzlich Herrn Prof. Dr.-Ing Ulrich Rückert von der Universität Bielefeld für das große Interesse an meiner Arbeit und das Anfertigen des Zweitgutachtens.

Weiterer Dank gilt meinem Gruppenleiter Emil Matúš sowie meinen Freunden, Kollegen und Projektpartnern für die fruchtbaren Diskussionen: Björn Almeroth, Friedrich Pauls, Steffen Kunze, Tobias Seifert, Sebastian Haas, Erik Fischer, Esther Pérez, Oliver Arnold, Marcus Völp, Nils Asmussen, Annett Ungetühm, um nur einige zu nennen.

Nicht zuletzt möchte ich auch meiner Familie danken, die mir all die Jahre einen großartigen Rückhalt gegeben hat. Besonderer Dank gilt meinen Eltern, die mir das Studium ermöglicht haben und stets ihre Wünsche diesem Ziel untergeordnet haben. Darüber hinaus möchte ich meiner Freundin Maria für ihre Geduld und Ausdauer mit mir danken.

Dresden, Oktober 2015 Benedikt Nöthen

Kurzfassung

Die ständig steigenden Anforderungen von Mobilfunksystemen in Bezug auf Energieeffizienz, Datendurchsatz und Latenzzeiten sowie eine komplexere Signalverarbeitung erfordern zunehmend leistungsfähige Mehrprozessorsysteme. Dazu werden heute heterogene Mehrprozessorsysteme eingesetzt, die besonders für zeitkritische und rechenintensive Berechnungsaufgaben eigens zugeschnittene, programmierbare Prozessoren besitzen. Darüber hinaus ist in Verbindung mit intelligenten DMA-Controllern eine weitere Verbesserung des Datendurchsatzes und der Energie-Effizienz möglich.

In dieser Arbeit wird ein Mehrprozessorsystem untersucht, das aus heterogenen Verarbeitungseinheiten und einem der Aufgabenverwaltung gewidmeten Prozessor (Task Scheduler) besteht. In den betrachteten Systemen bildet dieser Scheduler einen die Performance limitierenden Engpass. Aus diesem Grund werden in dieser Arbeit Konzepte zur Steigerung der Performance in heterogenen Mehrprozessorsystemen untersucht. Dabei werden Controller in die Verarbeitungseinheiten integriert, die eine lokale Taskverwaltung und Konfiguration der heterogenen Prozessoren ermöglichen und dadurch den zentralen Scheduler entlasten.

Für die zukünftig notwendige Bereitstellung von Rechenkapazitäten für Funktionen des Cloud-Computings in Mobilfunk-Basisstationen - zum Beispiel für Datenbankanwendungen - werden zusätzliche Verfahren der Vorverarbeitung von Daten im Controller untersucht. Dafür werden beispielsweise Funktionen zur Selektion aus Listen, zur Datenmodifikation auf Bit-Ebene und zur Verteilung zwischen verschiedenen Verarbeitungseinheiten in einem intelligenten DMA-Controller integriert. Auf diese Weise wird die Verarbeitungseinheit von wiederkehrenden, einfachen Aufgaben entlastet und in der Folge die Performance des Gesamtsystems wesentlich gesteigert. Eine weitere Performancesteigerung kann durch Vorverarbeitung von dynamischen Datenstrukturen, zum Beispiel von verketteten Listen, die ebenfalls häufig in Datenbanksystemen zum Einsatz kommen, erreicht werden. In diesem Fall erfolgt eine beschleunigte Verarbeitung dieser Datenstrukturen direkt am Speicher mit dem intelligenten DMA-Controller. Darüber hinaus wird eine zusätzliche Erweiterung untersucht, um neben Datenflussanwendungen auch das häufig eingesetzte Client-Server-Modell auf heterogenen Mehrprozessorsystemen abzubilden. Auch hier wird eine verbesserte Performance des Mehrprozessorsystems nachgewiesen.

Abstract

Modern mobile communication standards are characterized by continuously increasing signal processing complexity and strict energy-efficiency, latency and performance requirements. To fulfill these conflicting requirements, heterogeneous multiprocessor system on chips are used, which embed specialized and programmable cores for time critical and complicated computation tasks. An integration of these heterogeneous processing elements in combination with intelligent DMA-Controllers can further increase the system throughput as well as energy efficiency of the multiprocessor.

In this work, a multiprocessor system on chip containing several heterogeneous processing elements controlled by a dedicated scheduler for task dispatching is investigated. In these systems, the central scheduler is a bottleneck that limits the performance due to its managing tasks. Hence, this work analyzes concepts to increase the system performance in heterogeneous multiprocessors. As a suitable solution to reduce the managing overhead at the central scheduler, a controller is integrated into the heterogeneous processing elements in order to allow local task management and configuration. Subsequently, the managing overhead at the central scheduler is reduced, resulting in increased system performance.

Furthermore, mechanisms for data preprocessing in upcoming cloud computing workload hosted in basestations, e.g. database processing, are studied. Therefore, functionality for data selection, data modification on bit-level and data distribution to processing elements are integrated in an intelligent DMA-Controller. Thus, recurring workloads at the processing elements are reduced and the system performance is enhanced. To further increase the system performance an extension of the intelligent DMA-Controller for processing of dynamic datastructures, which are often used in database systems, is investigated. This allows preprocessing of datasets directly at the memory by the intelligent DMA-Controller. Additionally, another mechanism is analyzed in order to also accelerate applications based on the frequently used client-server model. The integration of these mechanisms into the DMA-Controller again leads to significant performance increase of the multiprocessor system.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADPLL	All-Digital Phase-Locked Loop
AES	Advanced Encryption Standard
AM	Antwort-Mechanismus
App	Applikationsprozessor
ASIP	Application Specific Instruction Processor
AT	Produkt aus Fläche und Zeit
ATE	Produkt aus Fläche, Zeit und Energie
CCC	Configuration und Communications Controller
Cell BE	Cell Broadband Engine
cfaed	Exzellenzcluster Center for Advancing Electronics Dresden
CM	CoreManager
CM-SO	CoreManager-Spinoff
CPI	Clock per Instruction
CPU	Central Processing Unit
CSV	Client-Slot Verhältnis
DCM	dezentraler Konfigurationsspeicher
DDF	dynamischer Datenflussgraph
DDR2	Double Data Rate 2
DK	Datenkriterium
DMA	Direct Memory Access Controller
DPAA	Data Path Acceleration Architecture
DSP	Digitaler Signalprozessor
Duo-PE	Integration von zwei heterogenen Prozessoren in einer PE
DVFS	Dynamic Voltage and Frequency Scaling
EDMA	Enhanced Direct Memory Access
EIB	Element Interconnect Bus
ESL	Electronic System Level
etc	et cetera
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIFO	First-In-First-Out
FK	Filterkriterium
FPGA	Field-Programmable-Gate-Array
GHz	Gigahertz
GPIO	General Purpose Input/Output
HD	High Definition

HDL	Hardware Description Language
I2C	Inter-Integrated Circuit Bus
iDMA	intelligenter DMA-Controller
IOMMU	Input/Output Memory Management Unit
KPN	Kahn Prozess Netzwerk
L2-Cache	Level-2 Cache
LDPC	Low Density Parity Check
LTE	Long Term Evolution
MFC	Memory Flow Controller
MHz	Megahertz
MMC	Microprogrammable Memory Controller
MMU	Memory Management Unit
MPSoC	Multiprocessor System-on-Chip
MUX	Multiplexer
NoC	Network on Chip
PCI	Peripheral Component Interconnect
PE	Processing Element (Verarbeitungseinheit)
PG	Performancegewinn
PPE	Power-Processing-Element
R	Router eines Network on Chips
RAM	Random Access Memory
RISC	Reduced-Instruction Set Computer
RPC	Remote Procedure Call
RTL	Register-Transfer-Level
SC-FDMA	Single Carrier Frequency Division Multiple Access
SDF	synchrone Datenflussgraphen
SDR	Software Defined Radio
SDSP	Scalar Digital Signal Processor
SIMD	Single Instruction Multiple Data
SME	Smart Memory Engine
SPE	Synergistic Processing Element
TIE	Tensilica Instruktionssatz-Erweiterungen
TLM	Transaction Level Modeling
TSD	Turbo Sphere Detector
TSMC	Taiwan Semiconductor Manufacturing Company
TU	Technische Universität
u.a.	unter anderem
UMC	United Microelectronics Corporation
UMTS	Universal Mobile Telecommunications System
VDSP	Vektor-DSP
VK	Verteilkriterium

VPS	Virtuelles Prototyping und Simulation
WCDMA	Wideband Code Division Multiple Access
WiMAX	Worldwide Interoperability for Microwave Access
z.B.	zum Beispiel

Symbolverzeichnis

A	Chipfläche
D	Abstand zwischen Datenobjekten innerhalb des Speichers
E	Energie
H	Anzahl Hops zwischen zwei Modulen
K	Anzahl an DMA-Kanälen
N	Anzahl an Datenobjekten einer Liste
N_{BEP}	Anzahl der Listenobjekte am Break-Even-Point
N_{Bits}	Anzahl selektierter Datenbits
N_{Client}	Anzahl an Clients im Client-Server Modell
N_S	Anzahl der Slots des iDMA-Controllers
P	Verlustleistung
PG	Performancegewinn
R	Verhältnis Anzahl Nutz-Datenobjekte zu Gesamt-Datenobjekte
T	Berechnungszeit für einen Algorithmus
T_{BTrf}	Takte zum Transfer einer Taskbeschreibung vom CM zu einer PE
T_{Base}	Takte für die Überprüfung aller Datenbits
T_{BitMod}	Takte für das Zusammensetzen selektierter Datenbits
T_{Bit}	Takte für die Verarbeitung eines selektierten Bits
T_{CM}	Takte am CoreManager zum Zuweisen eines Tasks an eine PE
T_{Calc}	Takte für eine Berechnung auf selektierten Daten
T_{Ctrl}	Takte der Verzögerung durch den Controller
T_{Distr}	Takte für das Verteilen von Datenobjekten
T_{Irq}	Takte für die Ausführung der Interrupt-Routine
T_{Laden}	Takte für das
T_{Loop}	Takte zur Iteration über selektierte Listenobjekte
T_{Mem}	Takte der Speicherlatenz für das Ablegen eines Datenobjektes
T_{NoCHop}	Zeit für den Transfer eines Pakets zwischen zwei Routern im NoC
T_{NoC}	Takte für das Entgegennehmen eines NoC-Pakets

$T_{Operation}$ . . . . Takte für die auszuführende Operation auf selektierten Daten

T_{Prog} . . . . . Takte für die Programmierung des Datentransfers am DMA-Controller

T_{Req} . . . . . Takte für das Generieren einer Leseanfrage

T_{Rsp} . . . . . Takte für das Generieren einer Antwort

T_{Sig} . . . . . Takte zur Signalisierung des Taskendes am CoreManager

$T_{Speichern}$ . . . . Takte für das Speichern des Endergebnisses

T_{Such} . . . . . Takte für die Selektion eines Listenobjektes

T_{Task} . . . . . Takte für die Ausführung eines Tasks

$T_{TrfObjekt}$ . . . . Takte für das Laden eines Datenobjektes

T_{Trf} . . . . . Takte für das Transferieren eines Listenobjektes

T_V . . . . . Gesamtanzahl von Takten für die Verarbeitung

T_{ZwMem} . . . . . Zusätzliche Verzögerungstakte beim Speicherzugriff

X . . . . . Grad der Selektion von Datenobjekten einer Liste

Y . . . . . Anzahl der Empfänger

Z . . . . . Anzahl zusätzlicher Verzögerungen T_{ZwMem} beim Speicherzugriff

$\overline{H}$ . . . . . Mittlere Entfernung zwischen zwei Modulen

$\overline{T_{Vergl}}$ . . . . . Mittelwert der Takte für Vergleich der Verteilbedingungen

f . . . . . Taktfrequenz

m . . . . . Anzahl von Datenobjekten, die das Filterkriterium erfüllen

1 Einleitung

Mit der rapiden Weiterentwicklung von Mobilfunkstandards und der wachsenden Nachfrage von Multi-Standard-Endgeräten wird in Mobilfunknetzen zunehmend mehr Flexibilität bei gleichzeitig steigenden Anforderungen in Bezug auf Energieeffizienz, Datendurchsatz und Latenz benötigt.

Die notwendige Flexibilität wird entsprechend dem Stand der Technik mit Software Defined Radios (SDR) realisiert. Das Konzept der SDR basiert auf der Nutzung von leistungsfähiger Standard-Hardware und der Realisierung von Funktionen in Sendern und Empfängern mit Software. Dies ermöglicht die Unterstützung von unterschiedlichen Mobilfunkstandards durch Softwareanpassungen. Der Paradigmenwechsel von Hardware dominierten Mobilfunksystemen mit dedizierten Verarbeitungseinheiten für verschiedene Aufgaben oder Berechnungen hin zu Software dominierten Systemen, wurde schon vor Jahren von Joseph Mitola in [Mit93] vorgestellt und wird heute in vielen Bereichen eingesetzt.

Eine fundamentale Herausforderung für die Realisierung zukünftiger SDR ist im Speziellen die Verarbeitung verschiedener hoch performanter Mobilfunkstandards zukünftiger Generationen. Zusätzlich müssen die Randbedingungen Größe, Gewicht und Leistungsverbrauch minimiert sowie die Kosten niedrig gehalten werden. Zum Beispiel werden für ein LTE Endgerät der Mobilfunkgeneration 3.9 Datenraten von bis zu 100Mb/s bei einer Verlustleistung von weniger als einem Watt gefordert [JLJ+10b]. Eine weitere drastische Verschärfung hinsichtlich Datenraten, Verlustleistung und Kommunikationslatenzen ist in den zukünftigen Standards, zum Beispiel für die Mobilfunkstandards der vierten Generation (LTE-Advanced) und der aktuell im Standardisierungsprozess befindlichen fünften Generation, erkennbar.

Dabei werden besonders zeitkritische und rechenintensive Aufgaben, zum Beispiel die Kanaldetektion oder die Fehlerschutzkodierung, von speziell zugeschnittenen, programmierbaren Prozessoren ausgeführt, um den Rechen- sowie Leistungsanforderungen gerecht zu werden. Aus diesem Grund integrieren aktuelle Chiplösungen für Mobilfunkanwendungen eine Vielzahl von spezialisierten, programmierbaren Verarbeitungseinheiten, Digitale Signalprozessoren (DSP) und allgemeine Reduced-Instruction Set Computer (RISC) Prozessoren. Die Kombination dieser verschiedenen Prozessoren auf einem Chip lässt sich unter dem Oberbegriff heterogene Mehrprozessorsysteme (*heterogeneous Multiprocessor System-on-Chip*, heterogeneous MPSoC) zusammenfassen. Mit der steigenden Anzahl integrierter Prozessoren können parallele Verarbeitungen auf verschiedenen Ebenen ausgenutzt werden. Durch die Nutzung verschiedener Recheneinheiten eines Prozessors innerhalb eines MPSoCs wird eine parallele Abarbeitung

von Programmbefehlen möglich, häufig auch als Befehlsebenenparallelität bezeichnet. Zusätzlich wird eine Parallelität auf Datenebene erreicht, indem mehrere Eingangsdaten von verschiedenen Prozessoren eines MPSoCs parallel bearbeitet werden. Zentrale Fragestellungen dabei sind jedoch:
Wie erfolgt die Kommunikation und Synchronisation zwischen parallel arbeitenden Verarbeitungseinheiten? Wie erfolgt eine energieeffiziente und gleichzeitig performante Ausführung der Anwendung? Wie erfolgt eine effiziente Programmierung einer heterogenen Hardwareplattform? Eine weitere zentrale Fragestellung ist die Isolation parallel ausgeführter Anwendungen, damit diese sich nicht beeinflussen. Durch die immer weiter steigende Integration von Prozessoren [Int11] steigt auch die Möglichkeit der parallelen Ausführung von Anwendungen aus verschiedenen Anwendungsbereichen. Es ist für die Entwicklung der fünften Mobilfunkgeneration zu erwarten, dass für die Anwendung der Steuerung von Maschinen über das Mobilfunknetz striktere Latenzanforderungen gefordert werden und die Ausführung der Steuerungssoftware selbst auf der Basisstation erfolgen muss [Fet14, BSDL14]. Weiterhin ist zu erwarten, dass Funktionen des Cloud-Computings, wie zum Beispiel das Bereitstellen von Rechenkapazität und Datenspeicher, dynamisch in Mobilfunk-Basisstationen zur Verfügung gestellt werden. Die angeführten Fragestellungen geben einen Eindruck über die Komplexität zukünftiger Systeme und können beliebig erweitert werden. In zahlreichen Forschungsprojekten werden dabei weitaus mehr Fragestellungen untersucht. Innerhalb des Exzellenzclusters "'Center for Advancing Electronics Dresden"' (*cfaed*) an der Technischen Universität (TU) Dresden wird ein Forschungsprojekt zur Integration und Nutzung wild heterogener Systeme durchgeführt. Wild heterogene Systeme bestehen aus unterschiedlichsten Materialien und Charakteristika in Mehrprozessorsystemen.

1.1 Ziele dieser Arbeit

Diese Arbeit befasst sich mit dem Entwurf und der Analyse neuartiger Kommunikationsmechanismen für den direkten Datenaustausch zwischen Verarbeitungseinheiten innerhalb von heterogenen Mehrprozessorsystemen. Um die Leistungsfähigkeit der verschiedenen Kommunikationsmechanismen zu untersuchen, wurde eine Entwicklungsplattform konzipiert, die Systemsimulationen von komplexen heterogenen Mehrprozessorsystemen ermöglicht.

Ein weiterer Schwerpunkt dieser Arbeit bildete die Entwicklung eines Konzeptes zur Entlastung von Verwaltungsaufgaben des zentralen Schedulers (CoreManagers) in Mehrprozessorsystemen. Hierfür wurde der CoreManager-Spinoff entwickelt, der erstmalig in jeder Verarbeitungseinheit innerhalb des heterogenen Mehrprozessorsystems Tomahawk2 des Lehrstuhls für Mobile Nachrichtensysteme der TU-Dresden als Hardwareeinheit integriert wurde. Die Vorteile des CoreManager-Spinoffs wurden analysiert

und untersucht sowie gegenüber anderen Konzepten vergleichend dargestellt.

Es zeichnet sich ab, dass mit weiter steigenden Verarbeitungsleistungen von SDRs auch in den Basisstationen Rechenkapazität und Speicher für kundenspezifische Anwendungen zur Verfügung gestellt werden können. Es ist davon auszugehen, dass auch die Datenhaltung für das Bereitstellen von Anwendungen innerhalb der Basisstation komplexer wird und zunehmend Datenbanken zur Anwendung kommen. Aufbauend auf diesen Anforderungen werden in zunehmendem Maße heterogene Mehrprozessorsysteme eingesetzt. Um die Verarbeitungsleistung in diesen MPSoCs zu steigern, wurde der klassische Direct Memory Access Controller (DMA) zum intelligenten DMA-Controller (iDMA) als Hardwareeinheit weiterentwickelt. Der iDMA-Controller filtert und verteilt Daten nach bestimmten Kriterien an Verarbeitungseinheiten innerhalb eines heterogenen Mehrprozessorsystems durch eine direkte Kommunikation der iDMA-Controller von anderen Verarbeitungseinheiten. Die Prozessoren werden durch die Vorverarbeitung der Datenströme im iDMA-Controller von untergeordneten Aufgaben entlastet. Mit diesem Ansatz kann die Performance gesteigert werden und zum Beispiel Datenbankanwendungen aufgrund des effektiveren Datenaustausches effizienter realisiert werden. In diesem Zusammenhang wurde der Leistungsverbrauch sowie der Flächenbedarf für den iDMA-Controller untersucht.

In Datenbanksystemen werden in der Regel dynamische Datenstrukturen zur Datenhaltung eingesetzt. Dynamische Datenstrukturen werden immer dann eingesetzt, wenn die Menge der zu verwaltenden Daten nicht von vornherein feststeht. Hierfür wird das Konzept des iDMAs funktionell erweitert und untersucht. In den Untersuchungen wurde festgestellt, dass die Ablage der dynamischen Datenstrukturen in externen Speichern Optimierungspotential beinhaltet. Im Rahmen dieser Arbeit wurde die Speicherstrategie dynamischer Datenstrukturen mit dem Ziel optimiert, die Performance zu steigern.

Während bei den vorhergehenden Kommunikationsmechanismen des iDMAs nur ein unidirektionaler Datenaustausch innerhalb von MPSoCs im Fokus stand, werden zukünftig auch bidirektionale Kommunikationsmechanismen benötigt, wie zum Beispiel Remote Procedure Call (RPC). RPC ist ein Client-Server Kommunikationsmodell, in dem eine Verarbeitungseinheit eine Anfrage an eine andere Verarbeitungseinheit sendet und auf die Antwort wartet. Es ist zweckmäßig den iDMA mit Funktionen zur Unterstützung der bidirektionalen Kommunikation von Verarbeitungseinheiten zu erweitern. Um eine beliebige Kommunikation zwischen Verarbeitungseinheiten zu unterbinden, erfolgt eine selektive Freigabe für den Datenaustausch durch den CoreManager. Um den CoreManager von häufig wiederkehrenden Freigaben zu entlasten, wird der iDMA-Controller zur Unterstützung einer bidirektionalen Kommunikation erweitert. Für diesen Fall muss die Isolation von parallel ausgeführten Anwendungen berücksichtigt werden.

In den nachfolgenden Abschnitten folgt ein Überblick über die Gliederung dieser Ar-

beit. Anschließend wird ein punktueller Überblick über aktuell existierende MPSoCs vermittelt. Darauf aufbauend werden Grundlagen zur Modellierung einer Applikation gegeben, die für das weitere Verständnis der Arbeit erforderlich sind.

1.2 Gliederung der Arbeit

Diese Arbeit ist wie folgt untergliedert: Einleitend werden derzeit existierende heterogene Mehrprozessorsysteme und Veröffentlichungen zur Steigerung der Systemperformance durch Vorverarbeitung von Daten während dem Datentransfer vorgestellt.

Darauf aufbauend wird in Kapitel 3 die Methodik zur Analyse und Entwicklung von neuen Konzepten der Kommunikation für heterogene Mehrprozessorsysteme auf Basis des vom Autor entwickelten virtuellen Prototypen beschrieben.

Kapitel 4 beschreibt den entwickelten Mechanismus zur lokalen Steuerung von heterogenen Verarbeitungseinheiten innerhalb eines heterogenen Mehrprozessorsystems. Darüber hinaus wird in diesem Kapitel die Erweiterung des Konzeptes des Lehrstuhls für Mobile Nachrichtensysteme der TU-Dresden zur Effizienzsteigergung von heterogene Mehrprozessorsystemen aus dem Bereich der digitalen Signalverarbeitung beschrieben und eine entsprechende Realisierung dieses Konzeptes im Tomahawk2 Forschungschip erläutert.

Für die Unterstützung anderer Anwendungsgebiete neben der digitalen Signalverarbeitung, z.B. Datenbanken, wird in Kapitel 5 ein Mechanismus vorgestellt, der die Daten direkt am internen bzw. externen Speicher nach bestimmten Kriterien selektiert und auf die Verarbeitungseinheiten verteilt. Gleichzeitig wird auf Basis der entwickelten Hardware ein ereignisbasiertes Ausführungsmodell vorgestellt und verglichen.

In Kapitel 6 werden Erweiterungen der Datenselektion zur Unterstützung von dynamischen Datenstrukturen untersucht, z.B. die Unterstützung von verketteten Listen. Eine weitere Erweiterung wird in Abschnitt 6.2 beschrieben und untersucht. Dabei werden neue Mechanismen innerhalb der Modulschnittstelle realisiert, um heterogene Prozessoren ohne spezielle Betriebssystemfunktionalität innerhalb eines Mehrprozessorsystems zu integrieren und diese isoliert von anderen Anwendungen betreiben zu können.

Abschließend werden in Kapitel 7 die Ergebnisse der Arbeit zusammenfassend präsentiert und ein Ausblick auf weitere Forschungsaspekte gegeben.

2 Stand der Technik

2.1 Heterogene Mehrprozessorsysteme

2.1.1 Konzept

Heterogene Mehrprozessorsysteme sind in mobilen Endgeräten, Multimedia-Einheiten und Anwendungen der Signalverarbeitung integriert, um mit standardisierter Hardware die steigenden Performanceanforderungen von Anwendungen mit limitierten Energievorgaben zu ermöglichen. Diese heterogenen Mehrprozessorsysteme bestehen aus vielen unterschiedlichen spezialisierten Prozessoren, die über ein Verbindungsnetzwerk miteinander verbunden und auf einem Chip integriert sind.

Der nachfolgende Abschnitt gibt einen kurzen beispielhaften Überblick über existierende heterogene Mehrprozessorsysteme.

2.1.2 Heterogene Mehrprozessorsysteme für mobile Endgeräte

In mobilen Endgeräten, wie zum Beispiel Smartphones und Tablets, wird u.a. das von der Firma ARM Ltd. entwickelte Big.LITTLE Konzept integriert. Das Big.LITTLE Konzept sieht eine Kombination von leistungsstarken ARM-Prozessoren und energieeffizienten ARM-Prozessoren vor. Für berechnungsintensive Anwendungen wird dabei der leistungsstarke Prozessor verwendet, während für weniger berechnungsintensive Anwendungen oder Standby der energiesparende Prozessor zum Einsatz kommt. Für die Software agieren die beiden Prozessortypen als ein Prozessor. Ein Beispiel dafür ist der [SSK+13], der zwei Cluster mit jeweils vier ARMv7 Prozessoren integriert. Ein Cluster besteht aus vier ARM Cortex-A15 Prozessoren, die mit einer maximalen Taktfrequenz von $1,8$ GHz für performancekritische Anwendungen genutzt werden. Der zweite Cluster besteht aus kleineren ARM Cortex-A7 Prozessoren, die mit einer maximalen Frequenz von $1,2$ GHz betrieben werden können und mit einem kleineren lokalen Speicher ausgestattet sind.

2.1.3 Heterogene Mehrprozessorsysteme für Multimedia-Anwendungen

Im Bereich der Multimedia-Anwendungen werden heterogene Prozessoren, zum Beispiel die Cell Broadband Engine (Cell BE) [KDH+05], eingesetzt. Die Cell Broadband

Engine besitzt ein Ausführungskonzept, das der später vorgestellten Tomahawk Plattform ähnlich ist.

Die Cell BE, die von Sony, Toshiba und IBM [KDH+05] gemeinsam entwickelt wurde, integriert neun heterogene Verarbeitungseinheiten. Acht sogenannte Synergistic Processing Elements (SPEs) sind für berechnungsintensive Aufgaben vorgesehen. Die Aufgabenverteilung auf die SPEs erfolgt durch das Power-Processing-Element (PPE) , das auf einem PowerPC Prozessor basiert. Das PPE besitzt einen 32kByte Instruktions- und Datenspeicher und einen 512kByte großen L2 Cache. Die SPEs hingegen besitzen einen 256kByte lokalen Speicher mit einem Memory Flow Controller (MFC), der den Datenaustausch mit dem Hauptspeicher ermöglicht. Die Recheneinheiten sind mit einem Element Interconnect Bus (EIB) verbunden, der aus vier 16Byte Kanälen besteht.

Bei der Cell-Plattform arbeiten die SPEs nicht direkt auf dem Hauptspeicher. Stattdessen wird ein explizites Speichermanagement unter Nutzung des MFCs verwendet. Hierbei muss die ausgeführte Software auf dem SPE selbst explizit Befehle zum Transfer von Daten in den lokalen Datenspeicher initiieren.

Für die Kommunikation mit anderen Einheiten werden bei der Cell-BE effektive Adressen verwendet. Die Umsetzung dieser effektiven Adressen in die entsprechende systemweite reale Adresse erfolgt mit einer Speicher-Management Einheit (Memory Management Unit (MMU)).

2.1.4 Heterogene Mehrprozessorsysteme für die Signalverarbeitung

Im Bereich der Signalverarbeitung wurde in [GIM+06] die SDR Plattform Sandblaster SB3011 vorgestellt, die die Standards UMTS, WCDMA und WiMAX unterstützt. Er integriert vier digitale Signalprozessoren (DSPs) und einen ARM9-Prozessor. Jeder DSP besitzt einen 32kByte Instruktions-Cache und einen 64kByte Datenspeicher. Für den Datenaustausch unter den DSPs steht ein gemeinsam genutzter Speicher von 1MB zur Verfügung. Jeder DSP unterstützt die Ausführung von bis zu 8 unabhängigen Threads. Die Ausführung wird mit einem Token Triggered Threading bestimmt. Sobald ein langsamer Hauptspeicherzugriff erfolgt, wird mit dem Token Triggered Threading der aktuelle Thread angehalten und zu einem anderen Thread gewechselt.

Eine weitere heterogene Mehrprozessorplattform für die Signalverarbeitung ist Infineon MuSIC, die in [Ram07] vorgestellt wurde. Die Infineon MuSIC Plattform integriert vier Single-Instruction-Multiple-Data (SIMD) DSPs und dedizierte Prozessoren für Filterung und Kanalkodierung. Jeder DSP besitzt vier Verarbeitungseinheiten, die Hardware Multitasking unterstützen. Über das Multitasking ist es möglich, zwischen mehreren Tasks umzuschalten, um Durchsatzeinbußen durch Anhalten der Pipeline innerhalb eines Tasks zu reduzieren. Als Datenspeicher stehen jedem DSP bis zu 40kByte Speicher zur Verfügung. Zusätzlich können mit dem Direct Memory Access Controller

(DMA) parallel zur Ausführung Daten aus dem gemeinsam genutzten on-Chip Speicher von 512kByte in den lokalen Speicher transferiert werden. Die Kontrolle über das System besitzt ein ARM926-Prozessor. Der ARM-Prozessor führt Kontrollprogramme aus und sendet den Verarbeitungseinheiten Adressen und Instruktionen des nächsten auszuführenden Tasks.

In [CBL+10] wurde das heterogene Mehrprozessorsystem MAGALI vom Institut CEA-LETI vorgestellt. Der Forschungschip integriert 23 heterogene Verarbeitungseinheiten in einem regulären $3x5$ Gitter-Netzwerk. Als Verarbeitungseinheiten sind fünf DSPs, vier spezielle Hardware-Akzeleratoren für FFTs und vier Data- und Konfigurations-Speichermodule integriert. Das asynchrone paketvermittelnde Netzwerk ermöglicht einen maximalen Datendurchsatz von bis zu 17GBit/s. Die Steuerung des Systems erfolgt mit einem dedizierten ARM-Prozessor. Innerhalb der Verarbeitungseinheiten sind für die Kontrolle und einheitliche Kommunikation untereinander ein sogenannter Configuration und Communications Controller (CCC) integriert. Dieser Controller ist für das Laden der Eingangs- und Ausgangsdaten und das Setzen der entsprechenden Frequenzen der Verarbeitungseinheiten verantwortlich. Wird eine Funktion, wie zum Beispiel eine FFT, auf einem Prozessor ausgeführt, wird zu Beginn eine Anfrage an den lokalen CCC gesendet. Dieser lädt daraufhin die entsprechenden Daten von einem der zentralen Konfigurationsspeicher (DCM) auf den Chip. Der DCM dient als gemeinsam zwischen den Verarbeitungseinheiten nutzbarer on-Chip Speicher. Dabei unterstützt der DCM das Ablegen der Daten nach dem First-In-First-Out (FIFO) Prinzip. Der integrierte Speichercontroller im DCM ist programmierbar und ermöglicht das Abbilden von bis zu vier FIFOs in den lokalen Speicher [MBCD09]. Für die Kommunikation zwischen Modulen wird ein kreditbasierter Flusskontrollmechanismus eingesetzt. Der Datenfluss wird zur Kompilierzeit definiert, wobei die Datenmenge zur Laufzeit bestimmt werden kann.

Eine weitere heterogene Mehrprozessorplattform ist STHORM, vormals als "Platform-2012" [BFFM12] bezeichnet, von den Firmen STMicroelectronics und CEA Leti. Die in [BFFM12] vorgestellte Plattform stellt ein Grundgerüst für heterogene Mehrkernprozessoren dar. Die Plattform kann als Co-Prozessor über High-Speed Links mit einem Host-Prozessor verbunden werden oder als alleinstehender Mehrkernprozessor verwendet werden. Die Berechnungsaufgaben werden dabei mit dem Fabriccontroller auf die verfügbaren Cluster verteilt. Die Cluster sowie der Fabriccontroller sind mit einem globalen asynchronen Netzwerk miteinander verbunden. Berechnungsaufgaben nimmt ein ClusterController vom Fabriccontroller entgegen. Der ClusterController ist für die Aufgabenverteilung und das Energiemanagement der bis zu 16 Verarbeitungseinheiten innerhalb des Clusters zuständig. Der ClusterController basiert auf einem STxP70 Prozessor der Firma STMicroelectronics und besitzt Zugriff auf einen 16kByte Instruktionscache, sowie 32kByte Datencache. Die Verarbeitungseinheiten greifen über ein internes Verbindungsnetzwerk innerhalb des Clusters auf bis zu 32 gemeinsam genutzte Speicherbänke zu.

Eine weitere heterogene Mehrprozessorarchitektur ist die KeyStone II Architektur von Texas Instruments, welche im Bereich der Signalverarbeitung, Cloud-Computing etc. eingesetzt werden kann. Die Plattform unterstützt verschiedene Konfigurationen und integriert bis zu acht DSPs als auch bis zu vier ARM Cortex-A15 Prozessoren [Tex13]. Darüber hinaus sind in dem *TCI6636K2H* vier Viterbi-Decoder und FFT-Co-Prozessoren integriert und über das TeraNet mit den Prozessoren verbunden. Die Speicherhierarchie der KeyStone II Architektur besteht aus drei Ebenen. Für jede dieser Ebenen stehen spezielle DMA-Controller zur Verfügung.

Die DSPs und ARM-Prozessoren besitzen einen 32kByte Programm- und Datencache. Zusätzlich verfügen die DSPs jeweils über einen 1024kByte L2 Cache. Für den Datentransfer zwischen dem L2 und L1 Cache eines DSPs steht ein interner DMA-Controller zur Verfügung, der bis zu 256Bits bei der halben Prozessortaktfrequenz transferieren kann. Für den Datenaustausch zwischen externen Speichern sowie dem ARM-Prozessorsystem steht ein erweiterter DMA (EDMA) zur Verfügung, der über das Verbindungsnetzwerk mit den Prozessoren und der Peripherie verbunden ist. Der EDMA stellt bis zu 64 DMA Konfigurationskanäle bereit, die über ein Ereignis oder manuell aktiviert werden können. Die Konfiguration erfolgt dabei über einen Parameter RAM (PaRAM) und ermöglicht das Verketten von Transferanweisungen durch Angabe des nachfolgenden Transfers. Neben ein- bzw. zweidimensionalen Datentransfers ermöglicht der EDMA auch dreidimensionale Transfers.

Die Kontrolle über die Verarbeitungseinheiten der KeyStone II Architektur erfolgt mit dem Multicore Navigator. Dieser besteht aus Hardware-Einheiten, um einen effizienten Datenaustausch sowie Synchronisation zwischen den Verarbeitungseinheiten zu ermöglichen. In der KeyStone II Architektur besitzt der Multicore Navigator bis zu 16.000 sogenannte Queues für die Verwaltung der Berechnungsaufgaben.

2.1.5 Der Tomahawk MPSoC

Als Lösung für die steigenden Anforderungen an den Datendurchsatz und die Energieeffizienz von Mobilfunkstandards, als auch an die Unterstützung mehrerer Mobilfunkstandards wurde in [LWB+08] die Tomahawk SDR Plattform vorgestellt. Das Basiskonzept der Tomahawk-Architektur wurde am Lehrstuhl für Mobile Nachrichtensysteme der TU-Dresden entwickelt. Die Tomahawk-Architektur ermöglicht aufgrund der strikten Unterteilung von Verarbeitungs- und Kontroll-Ebenen eine einfache Programmierung sowie durch das dynamische Task-Scheduling eine dynamische und energieeffiziente Anpassung an Änderungen innerhalb der Anwendung.

Der nachfolgende Abschnitt erläutert das Tomahawk-Architektur-Konzept und anschließend die Beschreibung einer Anwendung mit dem TaskC Programmiermodell des Lehrstuhls für Mobile Nachrichtensysteme der TU-Dresden.

2.1.5.1 Tomahawk-Architektur-Konzept

Die zugrundeliegende Idee der Architektur des Tomahawk Mehrprozessorsystems basiert auf [Sei06] und der logischen Unterteilung des MPSoCs in die Kontroll- (Control-Plane Subsystem) und Verarbeitungsebene (Data-Plane Subsystem). Die Blockstruktur ist in Abbildung 2.1 dargestellt.

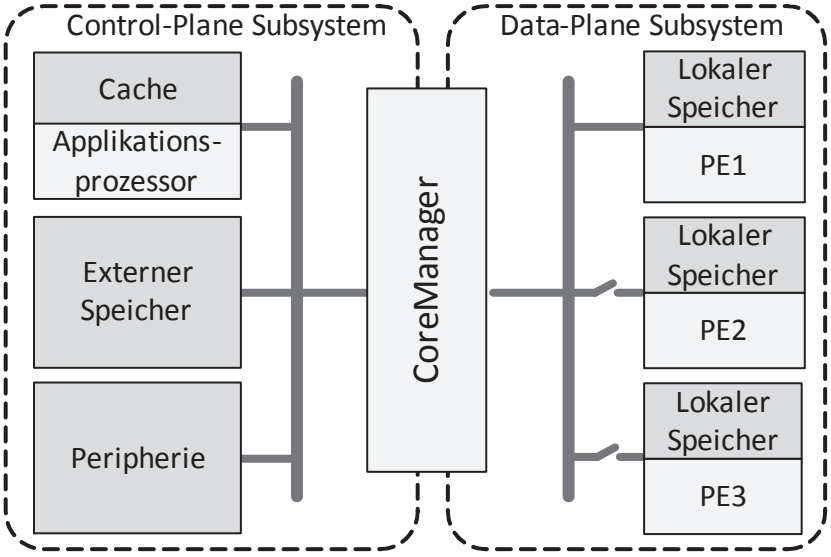


Abbildung 2.1: Tomahawk Architektur-Konzept

Die Kontrollebene enthält einen Applikationsprozessor, den dazugehörigen externen Speicher, sowie die Eingabe- und Ausgabe-Einheiten (*Peripherie*). Die Verarbeitungsebene setzt sich aus mehreren heterogenen Verarbeitungseinheiten (Processing Element, *PE*) zusammen, die über ein Verbindungsnetzwerk miteinander verbunden sind. Das Bindeglied dieser beiden Ebenen stellt der zentrale Scheduler dar, der im Tomahawk-Architektur Konzept als *CoreManager* bezeichnet wird [LWB+08] [Arn13]. Der CoreManager steuert das Daten-, Aufgaben- und Leistungsmanagement des gesamten Mehrprozessorsystems.

Der Applikationsprozessor innerhalb der Kontrollebene führt das Betriebssystem und den Kontrollcode der Applikation aus. Im Tomahawk-Konzept werden berechnungsintensive Aufgaben einer Applikation nicht auf dem Applikationsprozessor ausgeführt, sondern mit dem CoreManager - für den Programmierer transparent - den verfügbaren heterogenen Verarbeitungseinheiten (PE) zugewiesen, zum Beispiel DSPs oder ASIPs. Der Programmierer wird dadurch von Verwaltungsaufgaben entlastet. Um dieses Ziel zu erreichen, muss die Applikation in sinnvolle Module (Tasks) unterteilt werden. Ein Task ist atomar und verfügt während der Ausführung über keinerlei Kommunikation zu anderen Einheiten, ähnlich der Aktoren in synchronen Datenflussgraphen (SDF) [LM87] und dynamischen Datenflussgraphen (DDF) [LNZ14]. Tasks werden mit Funktionsaufrufen aus dem Kontrollpfad der Applikation aufgerufen und an den CoreManager gesendet. Der CoreManager nimmt diese Beschreibung des auszuführen-

den Tasks entgegen und führt eine Datenabhängigkeitsanalyse der Tasks durch, um die Ausführungsreihenfolge zu ermitteln. Besitzt ein Task keine Datenabhängigkeiten zu vorhergehenden Tasks, kann dieser einer entsprechenden Verarbeitungseinheit zugewiesen und ausgeführt werden. Bestehen Abhängigkeiten, muss auf die Beendigung der abhängigen Tasks gewartet werden. Für die Ausführung eines Tasks werden die benötigten Daten mit einem DMA-Controller aus dem Hauptspeicher in den lokalen Speicher der Verarbeitungseinheit geladen und die Ausführung gestartet. Die Allokation des lokalen Speicherbereichs für die Eingangs- sowie Ausgangsdaten erfolgt durch den CoreManager. Der Prozessor der Verarbeitungseinheit greift bei der Ausführung mit physischen Adressen auf den lokalen Speicher zu und kann nur auf diesen zugreifen. Der Prozessor ist der einzige Nutzer seines hardwaremäßig zugeordneten lokalen Speichers und hat keinerlei Kommunikationsmöglichkeiten mit anderen Einheiten. Nach der Ausführung werden die Daten wieder zurück in den Hauptspeicher geschrieben und stehen dem Applikationsprozessor wieder zur Verfügung.

Das Tomahawk-Konzept ist analog zu Reduced Instruction Set Computer (RISC) Superskalar-Prozessoren mit dynamischen Scheduling [HP11] aufgebaut, bei denen zum Erreichen von wenigen Takten pro Instruktion (Clock per Instruction, CPI) die auszuführenden Befehle zur Laufzeit den entsprechenden funktionalen Einheiten des Prozessors zugewiesen werden. Vor der Ausführung eines Befehls wird ebenfalls eine Datenabhängigkeitsanalyse durchgeführt, sodass die Ausführung der Befehle in einer anderen Reihenfolge erfolgen kann, als im Instruktionsspeicher angegeben. Im Vergleich zu RISC Superscalar-Prozessoren sind die Tasks und die Verarbeitungseinheiten in der Tomahawk-Architektur mächtiger und unterstützen komplexere Verarbeitungsalgorithmen.

2.1.5.2 Programmierung des Tomahawk MPSoC

Die Programmierung von Anwendungen für die Tomahawk-Architektur erfolgt mit dem TaskC Programmiermodell, basierend auf [Sei06] und den Erweiterungen aus [AF10]. Das TaskC Programmiermodell basiert auf der Programmiersprache C und ist um spezielle Funktionen zum parallelen Abarbeiten von Tasks erweitert worden. Listing 2.1 zeigt ein Beispiel für den Kontrollcode des Applikationsprozessors.

```
/* Initialisierung */
task(taskInit, IN(ptr0, size0), OUT(dataSet, size1), OUT(header, size2));
/* Partitionierung der Arbeit */
while(size1 != 0){
    task(taskWork, IN(ptr3, size3), OUT(ptr4, size4));
     if (size1 < header){
        ...
    }else{
        ...
    }
}
```

Listing 2.1: Beispiel eines TaskC Kontroll-Codes für den Applikationsprozessor des Tomahawk MPSoCs

Mit dem *task()* Aufruf können die definierten Tasks auf die vom CoreManager verwalteten Verarbeitungseinheiten ausgelagert werden. Der Aufruf sendet alle innerhalb der Klammern angegebenen Parameter, wie zum Beispiel den Namen des auszuführenden Tasks und die benötigten Datenbereiche für die Eingangs- und Ergebnisdaten an den CoreManager. Diese Datenbereiche werden durch die Startadresse, die Größe des Datenbereichs und die Zugriffsart beschrieben. Die Zugriffsarten sind Daten lesen, Daten schreiben und Daten modifizieren. Hierfür stehen die Makros *IN(), OUT(), INOUT()* zur Verfügung. Der CoreManager kann mit diesen Informationen die Abhängigkeiten der Tasks ermitteln. Die Ausführung der Tasks erfolgt parallel zum Kontrollfluss auf dem Applikationsprozessor. Die Beschreibung eines Tasks erfolgt separat und ist in Listing 2.2 exemplarisch dargestellt.

```
#pragma TASK_BEGIN task_test
#pragma TASK_TYPE pe_typ
void task_test (void *ptr0, void *ptr1, ... void *ptrn){
        ...
        /* TASK IMPLEMENTIERUNG FÜR TASK-TYP */
}
#pragma TASK_END
```

Listing 2.2: Beispiel eines Tasks in TaskC

Jeder Task ist durch sogenannte Pragmas *TASK_BEGIN* und *TASK_END* umschlossen, die durch einen Präprozessor vor der Kompilierung ausgewertet werden. Das Pragma *TASK_BEGIN* definiert gleichzeitig den Namen des Tasks, der bei einem Taskaufruf am Applikationsprozessor angegeben wird. Das Pragma *TASK_TYPE* definiert den Prozessor, auf dem der Task später ausgeführt werden kann. Als Parameter der in der Programmiersprache C definierten Funktion *task_test* können nur Zeiger verwendet werden, die später bei der Ausführung auf eine Kopie der Datenbereiche im lokalen Speicher der Verarbeitungseinheit zeigen. Der Task selbst darf auch nur innerhalb der

Datenbereiche arbeiten, da sonst ein Speicherkonflikt mit dem CoreManager auftreten kann, der die Speicherverwaltung der Verarbeitungseinheiten koordiniert.

2.1.5.3 Realisierung des Tomahawk1 MPSoCs

In [LWB+08] wurde die erste Realisierung des Tomahawk-Architektur-Konzeptes mit dem Tomahawk MPSoC der TU-Dresden vorgestellt. Die erste Realisierung des Tomahawk-Konzeptes wird in dieser Arbeit zur sicheren Unterscheidung als Tomahawk1 bezeichnet. Der Tomahawk1 MPSoC wurde 2007 in 130nm United Microelectronics Corporation (UMC) Technologie gefertigt und kann als alleinstehendes Mehrprozessorsystem verwendet werden. Das heterogene Mehrprozessorsystem integriert 13 ASIPs und DSPs.

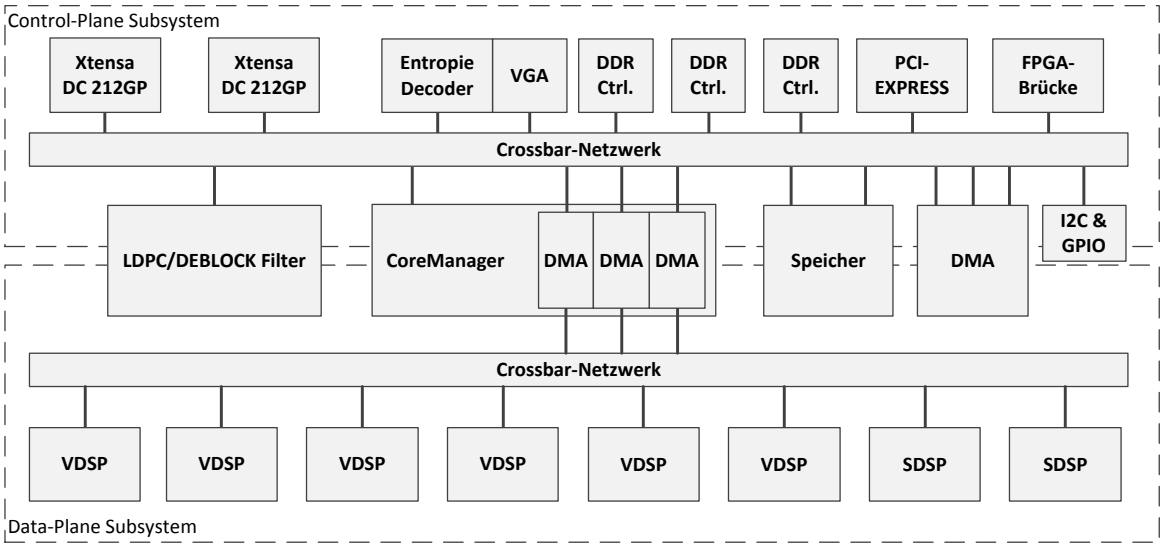


Abbildung 2.2: Blockschaltbild des Tomahawk1 MPSoC

Aus dem Blockschaltbild, Abbildung 2.2, ist die Trennung von Kontroll- und Verarbeitungsebene durch die zwei separaten Crossbar Verbindungsnetzwerke ersichtlich. Innerhalb der Kontrollebene sind zwei DC212GP Prozessoren der Firma Tensilica, der CoreManager, drei unabhängige Schnittstellen über DDR-Controller (DDR-Ctrl.) zum externen DDR-SDRAM Speicher und weitere Peripherie-Module (I2C, GPIO) integriert. Für die Kommunikation mit einem Host-Computer steht eine FPGA-Brücke oder eine PCI-Express Schnittstelle zur Verfügung. Die Verarbeitungsebene integriert sechs Vektor DSPs (VDSP) und zwei Scalar Floating Point DSPs (SDSP), die über den CoreManager gesteuert werden. Die beiden Verbindungsnetzwerke sind über die drei verfügbaren DMA-Controller am CoreManager miteinander verbunden. Die Steuerung der Prozessoren in der Verarbeitungseinheit (VDSP) erfolgt über den CoreManager mit einem DMA-Controller. Der CoreManager weist explizite Datentransfers vom externen Speicher in den lokalen Speicher der Verarbeitungseinheit an und startet nach Beendigung des Datentransfers die Ausführung auf der Verarbeitungseinheit. In [LWB+08] konnte die Leistungsfähigkeit des CoreManager Ansatzes gezeigt werden, der für das

Scheduling eines Tasks auf eine Verarbeitungseinheit zwischen 60 und 100 Takten benötigt. Dabei konnte bei Taskausführungszeiten im Mittel von 5.000 Takten und sechs Verarbeitungseinheiten eine nahezu lineare Performancesteigerung erreicht werden.

2.2 Datenaustausch in Mehrprozessorsystemen

2.2.1 Konzept

In Mehrprozessorsystemen erfolgt der Datenaustausch zwischen den Verarbeitungseinheiten und der Peripherie in der Regel mit direkten Speicherzugriffsmechanismen über Direct Memory Access Controller (DMA). Mit der steigenden Anzahl integrierter Verarbeitungseinheiten in heterogenen Mehrprozessorsystemen bildet der Datenaustausch im zunehmenden Maße einen Engpass. Bei zentraler Steuerung der DMA-Controller über zentrale Scheduler werden auszuführende Datentransfers in Warteschlangen der DMA-Controller abgelegt und nacheinander ausgeführt. Die sequentielle Abarbeitung der Datentransfers kann zu Wartezeiten an den parallel arbeitenden Verarbeitungseinheiten (PEs) führen. Das führt zu Einbußen in der Performance des Gesamtsystems. Zur Steigerung der Systemperformance sind andere Konzepte zum Datenaustausch notwendig. Dabei sind folgende Ansatzpunkte denkbar:

- Datenaustausch mit Steuerung der Verarbeitungseinheiten
- Datenaustausch mit Vorverarbeitung
    - Reduzierung der zu transferierenden Daten durch Vorverarbeitung
    - Entlastung der PEs durch Selektion
    - Entlastung der PEs durch Modifikation
    - Entlastung der PEs durch Verteilung der Daten
    - Ausnutzung von a-priori-Wissen bei Datenstrukturen
- Datenaustausch mit Überprüfung der Berechtigung an der Verarbeitungseinheit zur Entlastung des zentralen Schedulers

In den folgenden Abschnitten wird der Stand der Technik zum Datenaustausch in heterogenen Mehrprozessorsystem und die daraus abgeleiteten Methoden zur Performancesteigerung kurz beschrieben.

2.2.2 Datenaustausch mit Steuerung der Verarbeitungseinheiten

Die Veröffentlichungen [CBL+10] und [CLP+09] beschreiben die MAGALI-Plattform, die einen lokalen Controller für die Verwaltung des Prozessors integriert. Der Controller, auch Configurations and Communications Controller (CCC) genannt, unterstützt das An- sowie Abschalten des Taktes des Prozessors, sobald die Daten eines Tasks vollständig vorhanden sind oder keine Tasks mehr ausgeführt werden sollen. Der CCC ist

zusätzlich für die Steuerung des Prozessors und der Netzwerkschnittstelle verantwortlich. Eine Konfiguration enthält Anweisungen in Form von Mikro-Instruktionen. Die Reihenfolge der auszuführenden Konfigurationen werden beim Start einer Anwendung vom Scheduler oder Host-PC gesetzt. Der CCC kann mehrere Konfigurations-Slots besitzen und unterstützt das Laden von Konfigurationen parallel zur Ausführung. Nach dem Empfang eines Start-Signals prüft der CCC, ob die aktuell angeforderte Konfiguration geladen wurde. Ist die Konfiguration noch nicht verfügbar, werden die entsprechenden Daten aus seinem zugewiesenen on-Chip Konfigurationsspeicher (DCM) geladen. Anschließend wird die Netzwerkschnittstelle für das Entgegennehmen oder Weiterleiten von Daten konfiguriert und ein credit-basierter Flusskontrollmechanismus genutzt. Das Ende des Tasks signalisiert der Prozessor mit einem End-of-Computation Flag. Sind die angegebenen Konfigurationen abgearbeitet worden, kann der CCC einen Interrupt an den Host senden, um weitere Tasks zur Ausführung zu erhalten.

Der CCC entspricht den in dieser Arbeit verfolgten Ansätzen, die in den Kapiteln 4 und 5 beschrieben werden. In dieser Arbeit wird eine dynamische Konfiguration der Controller verfolgt, um flexibel auf unterschiedliche Anforderungen reagieren zu können. Zusätzlich wird direkter Datenaustausch mit allen Verarbeitungseinheiten des MPSoCs sowie externen Speichern ermöglicht.

Die Platform2012/STHORM [BFFM12] der Firmen STMicroelectronics und Cea Leti folgt einem dezentralen Ansatz, der bis zu 16 Prozessoren einem Cluster Controller zuweist. Der Cluster Controller basiert auf einem *STxP70* Prozessor der Firma STMicroelectronics und steuert das Starten der Prozessoren und das Scheduling innerhalb des Clusters. Im Gegensatz zu den Verarbeitungseinheiten innerhalb seines Clusters besitzt der Cluster Controller einen Vollzugriff auf die integrierte Peripherie des Clusters, zum Beispiel auf die zwei integrierten DMA-Controller. Die Prozessoren können Datentransferanfragen an den Cluster Controller richten, der diese überprüft und nach der Speicherallokation im Cluster einen der beiden verfügbaren DMA Kanäle programmiert. Der Datenaustausch innerhalb eines Clusters erfolgt über die gemeinsam genutzten Speicherbänke, die über ein spezielles Netzwerk mit den Prozessoren verbunden sind.

In [DJM11] wird eine ähnliche Trennung vorgestellt, indem neben dem zentralen Scheduler (Task Maestro) lokale Task Controller in jeder Verarbeitungseinheit integriert sind. Der Task Controller besitzt einen Task Buffer von zwei Taskbeschreibungen (Taskdescriptor), lädt die Eingangsdaten und ist für das Zurückschreiben der Daten verantwortlich. Der zentrale Scheduler, Task Maestro, ist verantwortlich für die Auflösung von Taskabhängigkeiten und Scheduling. Bis zu zwei Taskbeschreibungen können auf einer Verarbeitungseinheit gespeichert werden. Dieser Ansatz wurde in die Software für die Cell Architektur implementiert [MJ10], um die Systemperformance zu untersuchen. Die Cell-BE Architektur unterstützt ebenfalls durch die Integration der DMAs innerhalb der SPEs ein vom zentralen Scheduler losgelöstes Laden und

Schreiben der Daten. Der DMA kann über das Netzwerk oder durch den Prozessor innerhalb der SPE konfiguriert werden. Bis zu 16 Transfers können parallel behandelt werden. Die SPEs können durch DMA Anfragen an den PPE senden, um einen neuen Task zur Ausführung zu erhalten. Daraufhin werden die Taskbeschreibung und somit die benötigten Daten geladen. Zusätzlich ermöglicht der integrierte DMA-Controller innerhalb der SPEs das Verteilen von Daten.

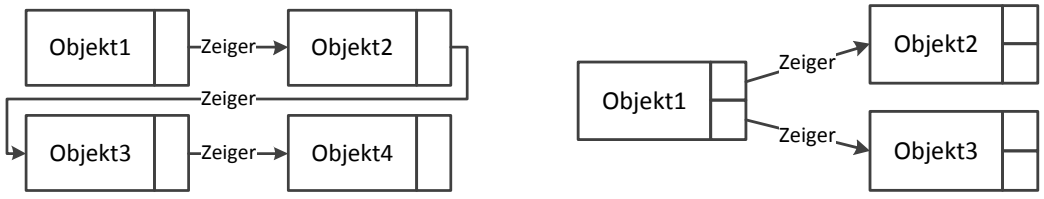
Der lokale Task-Controller entspricht dabei dem in Abschnitt 4 beschriebenen Konzept der lokalen Taskverwaltung an den Verarbeitungseinheiten. Zusätzlich zu den Datentransfers wird in dieser Arbeit die Steuerung von heterogenen Prozessoren innerhalb der Verarbeitungseinheit berücksichtigt.

2.2.3 Datenaustausch mit Vorverarbeitung

In zahlreichen Anwendungsbereichen wird eine Vorverarbeitung verwendet. In Datenbank-, Multimedia- und Signalverarbeitungssystemen werden Hardwarebeschleuniger eingesetzt, um durch Vorverarbeitung die Systemperformance zu steigern.

2.2.3.1 Beschleunigung des Datenzugriffes in Datenbanksystemen

In Datenbanksystemen werden je nach Anwendungsfall feste und dynamische Datenstrukturen zur Speicherung und Organisation von Daten verwendet. Ein Beispiel für eine feste Datenstruktur ist der eindimensionale Vektor oder die zweidimensionale Matrix, die eine feste Anzahl von Datenobjekten hintereinander im Speicher ablegen. Bei dynamischen Datenstrukturen kann die Organisation der Datenobjekte im Speicher beliebig erfolgen, indem jedes Datenobjekt einen oder mehrere Verweise oder Zeiger auf nachfolgende Datenobjekte enthält. Durch die Zeiger sind die Objekte miteinander verkettet. Eine spezielle Verkettung der Datenobjekte ist die hierarchische Anordnung der Datenobjekte, die eine schnelle Suche nach Datenobjekten ermöglicht. Diese Anordnung wird auch als Baum bezeichnet.



(a) Beispiel verketteter Objekte

(b) Beispiel einer hierarchischen Datenstruktur

Abbildung 2.3: Beispiele von dynamischen Datenstrukturen

Beschleunigung des Datenzugriffes in Datenbanksystemen mit statischen Datenstrukturen

Im Bereich der Datenbanksysteme wurden bis in die frühen neunziger Jahre zahlreiche Hardwarebeschleuniger vorgestellt. Als Beispiel sind [LHMH91] und [ISH+91] zu nennen, die spezielle Co-Prozessoren zur Beschleunigung von Grundoperationen von Datenbanken beschreiben. [LHMH91] erläutert einen programmierbaren Filterprozessor für die Mustererkennung innerhalb von Zeichenketten (Strings). In [ISH+91] wird ein Co-Prozessor für die Beschleunigung der Selektion und Sortierung vorgestellt. In beiden Ansätzen werden die Co-Prozessoren zur Filterung relevanter Daten verwendet, die aufgrund der spezialisierten Hardware eine schnellere Bearbeitung ermöglichen als eine Softwarelösung. Dabei werden die Co-Prozessoren entsprechend der Datenbankanfrage konfiguriert. Die Ergebnisse werden anschließend über den gemeinsam genutzten Hauptspeicher der Central Processing Unit (CPU) für die weitere Verarbeitung der Datenbankanfrage zur Verfügung gestellt.

Auf einem vergleichbaren Konzept basiert das von IBM entwickelte Netezza System [Fra14] für die Beschleunigung der Verarbeitung der Selektion mit anschließender Kompression in Serverfarmen. Dabei ermöglicht ein DMA-Controller innerhalb eines Speichersystems neben der Selektion auch die Modifikation von Daten durch eine Kompression der resultierenden Ergebnisse. Die häufig in einem großen Speichersystem (Storage System) abgelegten Daten können mit Kompressionsverfahren in den Hauptspeicher abgelegt werden. Weiterhin wird aufgrund der kürzeren Zugriffszeit auf den Hauptspeicher aufgrund der Kompression die Datenbankanfrage beschleunigt. Das Konzept nutzt einen Field Programmable Gate Array (FPGA) und ermöglicht eine weitere Anpassung der Selektions- und Kompressionshardware durch das Umprogrammieren des FPGAs in Abhängigkeit der Datenbankanfragen.

Die Bausteine zum Datenaustausch in Datenbanksystemen verwenden überwiegend Methoden der Selektion während des Datentransfers zur Performancesteigerung.

Beschleunigung des Datenzugriffes in Datenbanksystemen mit dynamischen Datenstrukturen

Veröffentlichungen zum Thema "'Beschleunigung des Datenzugriffes in dynamischen Datenstrukturen"' können in Software- und Hardwareansätze unterteilt werden.

Software-Ansätze zur Beschleunigung der Verarbeitung

Bei diesen Ansätzen wird im Wesentlichen die Datenstruktur an die Struktur des Speichers angepasst. Eine softwareseitige Lösung zur Performancesteigerung wurde in [RS99] veröffentlicht. Das beschriebene Konzept sieht die Nutzung weiterer Zeiger innerhalb eines Objektes vor, um schneller das gesuchte Element zu ermitteln. In der Literatur wird dieses Konzept auch als Jump-Pointer bezeichnet. Abbildung 2.4 zeigt das Jump-Pointer Konzept am Beispiel einer Liste.

Bei der Verwendung des Jump-Pointer Ansatzes muss allerdings der aufgespannte Pfad der Zeiger bekannt sein, um ein effizientes Vorladen zu ermöglichen [HG09] [RS99].

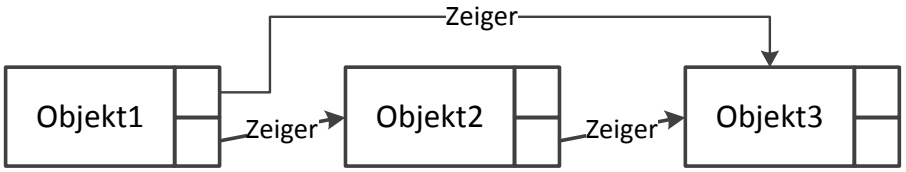


Abbildung 2.4: Anwendung des Jump-Pointer Konzeptes bei einer Liste

Mit diesem Ansatz kann bei Listen eine Performancesteigerung erreicht werden. Bei hierarchischen Datenstrukturen führt das Hinzufügen weiterer Zeiger zu einem deutlich höheren Speicherplatzbedarf, sodass am Prozessor weniger Objekte innerhalb des lokalen Speichers abgelegt werden können. Die Reduktion des verfügbaren lokalen Speicherplatzes am Prozessor führt zu häufigerem Nachladen und somit zu einem Performanceverlust.

Die beiden folgenden Veröffentlichungen [RR00] und [CHL00] beschreiben eine weitere softwareseitige Lösungsmöglichkeit. Diese Veröffentlichungen basieren auf einer Anpassung der Datenstruktur an die zugrundeliegende Hardware, insbesondere an die Speicherhierarchie. Grundgedanke dieser Veröffentlichungen ist die Anpassung der Datenstruktur unter Berücksichtigung des blockweisen Datenaustausches zwischen den Ebenen der Speicherhierarchie des Prozessorsystems.

In [RR00] wird dazu eine Anpassung eines B+ Baums[1] vorgestellt, die mehrere Objekte einer Ebene des Baumes zu einem neuen Datenobjekt zusammenfasst. Das Zusammenfassen von Objekten einer Ebene ermöglicht schnelles Hinzufügen und Löschen von Objekten. Bei der Suche nach Objekten kann jedoch die räumliche Lokalität des Caches bei einem Zugriff auf das nachfolgende Objekt nicht ausgenutzt werden. Dafür wurden in [CHL00] Zugriffsmuster identifiziert, um die Definition der Datenstruktur mit Programmprofilen zu optimieren. Für die Beschleunigung der Suche innerhalb eines binären Baumes[2] wurde daraufhin in [KCS+10] eine hierarchische Unterteilung unter Ausnutzung der Speicherhierarchie vorgestellt. Zusätzlich wurde für eine parallele Abarbeitung mehrerer Einträge durch den Prozessor mit Single-Instruction-Multiple-Data (SIMD) Befehlen der binäre Baum strukturell erweitert. Abbildung 2.5 stellt die vorgestellte hierarchische Unterteilung grafisch dar. Der kleinste Teilbaum wurde dabei so definiert, dass dieser innerhalb der SIMD-Register des Prozessors geladen werden kann. Mehrere dieser Teilbäume werden daraufhin in eine Cache-Zeile abgelegt. Die Teilbäume innerhalb einer Cache-Zeile bilden wiederum einen größeren Teilbaum im Hauptspeicher. Durch diese Anpassung kann ein Performancegewinn bei der Suche nach Objekten um bis zu 70% für balancierte binäre Bäume erreicht werden.

[1] Ein B+ Baum ist dadurch gekennzeichnet, dass die Daten innerhalb der Blatt-Knoten des Baumes abgespeichert sind.

[2] Ein binärer Baum, auch binärer Suchbaum genannt, bezeichnet eine dynamische Datenstruktur bei denen die Objekte nach Ihren Werten hierarchisch angeordnet sind.

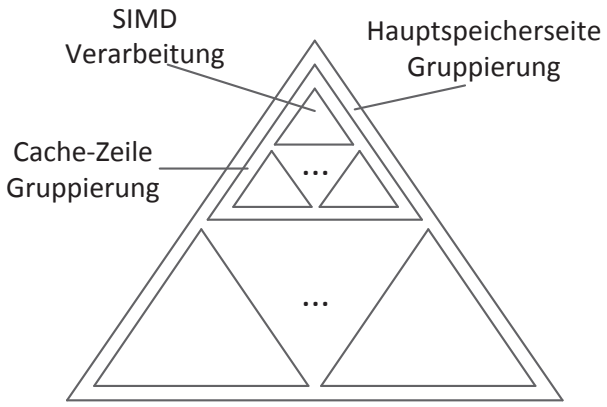


Abbildung 2.5: Hierarchische Unterteilung des binären Baumes nach [KCS+10]

Hardware-Ansätze zur Beschleunigung der Verarbeitung
Bei diesem Ansatz werden spezielle Co-Prozessoren zur Verarbeitung dynamischer Datenstrukturen eingesetzt. Für die hardwareseitige Unterstützung dynamischer Datenstrukturen wurde in [KCKY01] ein Konzept beschrieben, das einen Co-Prozessor für die Verarbeitung von verketteten Listen integriert. Der Prozessor konfiguriert hierfür den Co-Prozessor mit einer Beschreibung der Datenstruktur und der Startadresse im Hauptspeicher. Mit dieser Information kann der Co-Prozessor die Position der Zeiger innerhalb der Datenobjekte ableiten und Objekte vor dem Zugriff durch den Prozessor aus dem Speicher laden (Prefetching). Die vorgeladenen Objekte werden in einen Puffer abgelegt, der über die Software vom Prozessor aus erreichbar ist. Die Verarbeitung der Suchanfragen und Überprüfung der Datenobjekte erfolgt per Software auf dem Prozessor. Der Co-Prozessor unterstützt ein internes Scheduling für die parallele Verarbeitung von unabhängigen Datenstrukturen. Der Vorteil dieses Ansatzes ist eine Vorverarbeitung der Objekte durch den Co-Prozessor. Dadurch wird die Verarbeitungseinheit entlastet und eine Performancesteigerung erreicht.

Aufgrund der Auswertung der Suchbedingung in Software durch den Prozessor, ist in [KCKY01] nur die Unterstützung von verketteten Datenstrukturen vorgesehen, bei denen der Verweis auf das Nachfolgeobjekt eindeutig ist. In dieser Arbeit erfolgt die Auswertung der Suchbedingung für ein Datenobjekt in Hardware parallel zum Datentransfer. Dieses ermöglicht eine Unterstützung von hierarchischen Datenstrukturen und eine weitere Performancesteigerung, da der Prozessor für die Suche eines Datenobjektes nicht benötigt wird.

2.2.3.2 Beschleunigung des Datenzugriffes in Multimediasystemen

Optimierungen von Speicherzugriffen durch zusätzliche Unterstützung innerhalb des Speichercontrollers [BI12] oder durch DMA-Controller ist besonders in Multimedia Domänen verbreitet, um die steigenden Anforderungen der Datenverarbeitung in Videostandards von HD und UltraHD zu unterstützen. Hierzu wurden zum Beispiel in [Nan14] und [WWZW14] für den Datentransfer spezielle Befehle vorgesehen, um das

Datenformat für die Verarbeitungseinheiten anzupassen.

In [VCGT+12] wird eine konzeptionelle Erweiterung eines DMA-Controllers beschrieben, um die transferierten Daten während des Transfers zu modifizieren. Das Konzept ist dabei die Anpassung der Daten an das für die Verarbeitung auf einem Prozessor benötigte Datenformat. Dabei wird ein *Realignment Network*, bestehend aus einem Barrelshifter, verwendet. Der Barrelshifter ermöglicht eine beliebige Verschiebung der Datenbits der transferierten Daten. Mit einem *Merge Network* können Daten an den Bereichsgrenzen zu einem neuen Datum zusammengesetzt werden.

Die Bausteine zum Datenaustausch in Multimediasystemen basieren im Wesentlichen auf einer Performancesteigerung durch Datenmodifikation während des Datentransfers.

2.2.3.3 Beschleunigung des Datenzugriffes in Signalverarbeitungssystemen

Für den Bereich der digitalen Signalverarbeitung wurde in [HSB+10] ein erweiterter DMA Controller, Namens *SmartDMA* vorgestellt, der Kryptographieverfahren, zum Beispiel *Advanced Encryption Standard* (AES) und *SNOW 3G* unterstützt. Durch die Integration wird ein Umweg über ein dediziertes Modul verhindert und aufgrund der reduzierten Datentransfers eine Energieeinsparung von zwei Drittel im Vergleich zu einer konventionellen Lösung ermittelt. Dieser Baustein ist der Performancesteigerung durch Modifikation zuzuordnen.

In [CBL+10] wird ein heterogener Basisbandprozessor mit drei integrierten Speichermodulen (MMC) für die Unterstützung von Datenflussanwendungen beschrieben. Das Speichermodul integriert einen Controller und vier unabhängige Speichermodule mit jeweils 32kB Speicherkapazität. Der Controller ist programmierbar und unterstützt Funktionen wie Synchronisation und Pufferung [MBCD09]. Für die Pufferung stehen vier FIFO-Puffer zur Verfügung, die in die lokalen Speicher abgebildet werden. Ein lesender Zugriff auf einen FIFO-Puffer erfolgt durch programmierbare Lese-Prozesse. Ein Lese-Prozess ermöglicht das Berechnen der Leseadressen, das Laden der Daten aus dem lokalen Speicher und das anschließende Versenden der Daten per Software. Ein schreibender Zugriff ist nicht programmierbar. Für die Kontrolle und Kommunikation steht der Configuration Communication Controller (CCC) zur Verfügung. Der CCC stellt neben dem Powermanagement einen Flusskontrollmechanismus zwischen Sender und Empfänger auf Basis von Credits zur Verfügung.

In [JLJ+10b] wird das Modul aus [MBCD09] innerhalb einer Verarbeitungseinheit integriert und als sogenannte *Smart Memory Engine* (SME) bezeichnet, ähnlich wie der iDMA-Controller in dieser Arbeit. Mit dem Konzept der *Smart Memory Engine* wird direkte Kommunikation zwischen Verarbeitungseinheiten ermöglicht. Eine Filterung ist durch die Mikroprogrammierung prinzipiell möglich, jedoch wurde dieses aufgrund der Anwendung im Bereich der digitalen Signalverarbeitung nicht weiter untersucht.

Ein Vergleich mit dem MMC aus [MBCD09] erfolgt in Abschnitt 5.6.2.2.

Beide vorgenannten Controller ermöglichen eine Modifikation und Datenflussunterstützung zur Steigerung der Systemperformance.

Kommerzielle Systeme, zum Beispiel der *Dual Data Movement Accelerator* von TI [Tex07] ermöglicht die Abbildung eines Ring-Puffers innerhalb des lokalen Speichers. Die Größe eines Datenelementes ist dabei variabel zwischen 8, 16 und 32 Bit einstellbar. Als Transfer-Modi stehen neben $1D$, $2D$ und $3D$ Datentransfers zur Verfügung. Zusätzlich werden schreibende und lesende Datentransfers in die FIFO-Puffer unterstützt. Für die Event gesteuerte Verarbeitung unterstützt die Einheit Interrupts, die bei einem bestimmten Füllstand des FIFOs ausgelöst werden (sogenannte Watermarks).

Dieser Controller ermöglicht eine Datenverteilung und eine Datenflussunterstützung zur Steigerung der Systemperformance.

2.2.3.4 Beschleunigung des Datenzugriffes in sonstigen Systemen

Eine weitere kommerzielle Lösung wurde in [Fre12] beschrieben. Speziell für MPSoCs für die Paketverarbeitung in Netzwerken wurde die Data Path Acceleration Architetcure (DPAA) von der Firma Freescale entwickelt. Der DPAA ist ein Co-Prozessor und kann innerhalb eines MPSoCs integriert werden, um eingehende Paketströme aus Netzwerken mit entsprechenden Kriterien innerhalb von Warteschlangen abzulegen. Die integrierten Prozessoren können anschließend auf die Warteschlangen zugreifen und werden von der Filterung entlastet. Der DPAA kann zusätzlich mit Akzeleratoren aus dem Bereich der Kryptographie erweitert werden. Anders als der Ansatz der Firma Freescale wird in dieser Arbeit die Integration innerhalb der Verarbeitungseinheit verfolgt, die eine beliebige Verschaltung der Verarbeitungseinheiten zu logischen Pipelines und auch eine Datenanpassung (Datenmodifikation) für den Datenaustausch im heterogenen MPSoC ermöglicht. Der in dieser Arbeit beschriebene Baustein ermöglicht die Selektion, Verteilung von Daten und Datenflussunterstützung zur Steigerung der Systemperformance.

In [Bax02] wird eine DMA Lösung patentiert, die logisch arithmetische Operationen auf dem Datenstrom durchführen kann, zum Beispiel logische Verknüpfungen und arithmetische Operationen. Das Patent ist der Performancesteigerung durch Modifikation der Daten zuzuordnen.

Der kommerzielle DMA-Controller EDMA von Texas Instruments [Tex11] steigert die Performance durch Datenflussunterstützung. Die CoreLink DMA-330 [ARM12] und PrimeCell μDMA-Controller von ARM ermöglichen eine Performancesteigerung durch Datenverteilung. Grundfunktionalitäten der CoreLink DMA-330 und PrimeCell μDMA sind neben $1D$ und $2D$ Datentransfers auch das Verteilen von Daten in vordefinierten Bereichen.

2.2.4 Überprüfung der Berechtigung an der Verarbeitungseinheit

Mit der Übernahme von zusätzlichen Verarbeitungsaufgaben innerhalb der Basisstation sind Mechanismen zur Isolation von Anwendungen unabdingbar. Die Mechanismen zur Isolation stellen dabei sicher, dass eine fehlerhafte Anwendung andere laufende Anwendungen auf einem MPSoC nicht beeinflusst. Isolation wird in der Praxis häufig durch die Nutzung von virtuellen Adressräumen gelöst [Fot61]. Dabei wird jeder Anwendung ein eigener virtueller Adressraum zugeordnet und durch eine Speicherverwaltungseinheit (Memory Management Unit, MMU) innerhalb des Prozessors in phyisische Adressen umgewandelt. Die Umwandlung erfolgt dabei transparent für die ausgeführte Anwendung auf dem Prozessor. Um sicherzustellen, dass der Prozessor nur auf freigegebene Adressbereiche zugreift, erfolgt durch die MMU eine Überprüfung der Zugriffsberechtigung.

Um auch die vom Prozessor gesteuerten und zunehmend in den Verarbeitungseinheiten von MPSoCs integrierten DMA-Controller vor Missbrauch durch eine Anwendung zu schützen, wurde in [SG12] eine Lösung zur Isolation von Anwendungen auf Basis von Virtualisierungskonzepten vorgestellt. Bei diesem Ansatz werden alle Konfigurationen durch die MMU an einen Hypervisor weitergereicht, der auf dem selben Prozessor ausgeführt wird. Dieser Hypervisor verhindert die direkte Konfiguration des DMA-Controllers. Der Hypervisor stellt somit einer Anwendung ein virtuelles Modell des DMA-Controllers zur Verfügung. Durch den Hypervisor können die Konfigurationen überprüft werden und erst dann an den DMA-Controller weitergereicht werden. Voraussetzungen bei diesem Ansatz sind, dass alle Verarbeitungseinheiten eine MMU besitzen, der Prozessor die Trennung zwischen User- und privilegiertem Mode unterstützt und auf jeder Verarbeitungseinheit ein sogenannter Hypervisor ausgeführt werden muss. Diese Voraussetzungen sind besonders in spezialisierten heterogenen Mehrprozessorsystemen aufgrund der softwareseitigen Überprüfung der DMA-Konfiguration durch den Hypervisor nicht ohne Performanceverlust einsetzbar.

Ein anderer Ansatz zur Isolation von Anwendungen wurde in [Tel90] beschrieben, der die Grundlage von heutigen Input/Output MMUs (IOMMU) [ByMK+06] bildet. Dabei wird eine MMU dem DMA-Controller zur Verfügung gestellt, die vor einem Datentransfer die Adressumwandlung sowie die Zugriffsberechtigungen überprüft. Damit ist die Systemstabilität durch das Verhindern von schreibendem Zugriff auf nicht freigegebene Datenbereiche gewährleistet. Das Konzept wurde in [FAH+06] [KDH+05] erfolgreich eingesetzt und ermöglicht das dynamische Referenzieren von Speicherbereichen für den Datenaustausch zwischen anderen parallel ausgeführten Anwendungen auf unterschiedlichen Verarbeitungseinheiten des MPSoCs [Gsc08]. Die Verwaltung der Einträge erfolgt durch den PPE Prozessor, der fehlende Einträge nachladen kann.

Ein anderer Ansatz zur Isolation von Anwendungen in heterogenen Mehrprozessorsytemen wurde in [FPLS07] beschrieben. Dieser Ansatz eignet sich besonders für spezialisierte heterogene Mehrprozessorsysteme mit Akzeleratoren, die mit einem limitier-

ten Energiebudget auskommen und nicht zwingend mit einem DMA-Controller und MMU ausgestattet sein müssen. In [FPLS07] wird dabei ein Mechanismus innerhalb der Netzwerkschnittstelle der Verarbeitungseinheiten beschrieben.

Sendet der Prozessor der Verabeitungseinheit eine Anfrage an die Netzwerkschnittstelle, wird anhand der angelegten Adresse die entsprechende Routing-Information mit einer Tabelle ermittelt. Die Tabelle enthält Informationen über die Speicherbereiche, Zugriffsarten sowie die Modul-ID. Wird ein Zugriff außerhalb des erlaubten Speicherbereichs festgestellt, wird die Anfrage verworfen. Eine Erweiterung wird in [FPL+08] vorgestellt, die einen ähnlichen Mechanismus zur Verwaltung der Tabelleneinträge nutzt. Ein dediziertes Modul, in [FPL+08] auch als Network Security Manager genannt, ist für die Verwaltung und Konfiguration der Netzwerkschnittstellen verantwortlich. Mit dieser Einheit können zur Laufzeit Zugriffsberechtigungen auf Speicherbereiche aktualisiert werden. Werden neue Berechtigungen benötigt, können diese beim Network Security Manager angefragt werden, der daraufhin bei erfolgreicher Überprüfung eine neue Konfiguration sendet.

In [FPLS07] und [FPL+08] ist nur ein An- bzw. Abschalten des Netzwerkverkehrs vorgesehen. Eine Limitierung der Datenrate für die Kommunikation zwischen Verarbeitungseinheiten ist im Gegensatz zu dieser Arbeit nicht vorgesehen.

2.2.5 Zusammenfassung und Bewertung zum Datenaustausch in MPSoCs

In Tabelle 2.1 sind die verschiedenen Bausteine und Methoden zur Beschleunigung des Datentransfers vergleichend dargestellt. Dieser Vergleich basiert auf den in diesem Kapitel aufgeführten Veröffentlichungen und Datenblättern und erhebt keinen Anspruch auf Vollständigkeit. In den Bereichen Signalverarbeitung und Multimedia werden in den zitierten Veröffentlichungen modifizierte DMA-Controller zur Verbesserung der Systemperformance beschrieben. Publikationen aus dem Bereich der Datenbanken beschreiben vorwiegend Co-Prozessoren oder Software-Lösungen zur Steigerung der Verarbeitungsleistung.

Das in [JLJ+10b] beschriebene Konzept integriert zusätzliche Funktionalitäten in Verarbeitungseinheiten und kommt dem in dieser Arbeit beschriebenen Ansatz nahe. Der Unterschied besteht in der Datenflussunterstützung und der Datenmodifikation. Der DPAA aus [Fre12] nutzt drei Methoden (Selektion, Modifikation und Datenflussunterstützung) auf der Basis eines Co-Prozessors. In dieser Arbeit werden Bausteine beschrieben und untersucht, die alle relevanten Mechanismen in einem erweiterten DMA-Controller integrieren. In den ausgewerteten Dokumenten sind keine Beschreibungen zu finden, die zur Performancesteigerung in MPSoCs alle wesentlichen Methoden in einem Baustein, zum Beispiel DMA-Controller, Co-Prozessor oder Controller, verwenden. In dieser Arbeit werden die wesentlichen Verfahren zur Performancestei-

gerung, die Steuerung der Verarbeitungseinheiten, die Selektion, die Modifikation und die Verteilung von Daten, die Verbesserung der Kommunikation durch Datenflussunterstützung und die Isolation von Anwendungen, in einem Baustein integriert. Die nachfolgenden Kapitel erläutern die Methoden zur Performancesteigerung im Detail.

Anwendung	Realisierung	Performance-Steigerung durch	Steuerung der PE	Selektion von Daten	Modifikation von Daten	Verteilung von Daten	Datenfluss-unterstützung	Isolation von Anwendungen
Datenbank	[LHMH91]	Co-Prozessor	-	X	-	-	-	-
	[ISH+91]	Co-Prozessor	-	X	-	-	-	-
	IBM Netezza [Fra14]	FPGA	-	X	X	-	-	-
	[KCKY01]	Co-Prozessor	-	X[3]	-	-	-	-
	[RR00]	Software	-	X	-	-	-	-
	[CHL00]	Software	-	X	-	-	-	-
	[KCS+10]	Software	-	X	-	-	-	-
Multimedia	[BI12]	DMA-Controller	-	-	-	-	-	-
	[Nan14]	DMA-Controller	-	-	X	-	-	-
	[WWZW14]	DMA-Controller	-	-	X	-	-	-
	DMA++ [VCGT+12]	DMA-Controller	-	-	X	-	-	-
	Cell BE [FAH+06]	DMA + MMU	-	-	-	X[4]	-	X
Signalverarbeitung	SmartDma [HSB+10]	DMA-Controller	-	-	X	-	-	-
	MMC [MBCD09]	Co-Prozessor	-	-	X	-	X	-
	SME [JLJ+10b]	DMA-Controller	-	-	X	-	X	-
	CCC [CBL+10]	Controller[5]	X	-	-	-	X	X
	STHORM [BFFM12]	Co-Prozessor	X	-	-	-	-	X
	Task Contr. [DJM11]	Controller[5]	X	-	-	-	-	X
	TI DMAX [Tex07]	DMA-Controller	-	-	-	X	X	-
Sonstige	[Bax02]	DMA-Controller	-	-	X	-	-	-
	TI EDMA3 [Tex11]	DMA-Controller	-	-	-	-	X	-
	DMA330 [ARM12]	DMA-Controller	-	-	-	X[4]	-	-
	DPAA [Fre12]	Co-Prozessor	-	X	-	X	X	-
	[SG12]	DMA + MMU	-	-	-	-	-	X
	[FPLS07] [FPL+08]	Controller[5]	-	-	-	-	-	X

Tabelle 2.1: Gegenüberstellung relevanter Methoden zum Datenaustausch zur Steigerung der Verarbeitungsperformance

[3]Eine Auswertung der Suchbedingung eines Datenobjektes erfolgt in Software auf dem Prozessor.
[4]Verteilung von vordefinierten Datenbereichen
[5]Integrierter Controller in Verarbeitungseinheit

3 Virtuelles Prototyping und Simulation

Die ständig steigende Integration von Verarbeitungseinheiten und die Interaktion dieser Komponenten untereinander innerhalb eines Mehrkernprozessorsystems führen zu einer exponentiell steigenden Systemkomplexität. Diese stellt zunehmend eine Herausforderung bei der Entwicklung von eingebetteten Systemen dar, da das Zusammenspiel zwischen Software und Hardware erst im nahezu letzten Schritt getestet werden kann.

Für die Untersuchung von heterogenen Mehrprozessorsystemen sind Simulations- und Testplattformen zur Untersuchung neuer Kommunikationsmechanismen oder Prozessorstrukturen unabdingbar. Virtuelles Prototyping und Simulation (VPS) ist heute fester Bestandteil bei der Entwicklung von neuen Produkten. Durch VPS können frühzeitig Varianten innerhalb des Entwicklungsprozesses untersucht und bewertet werden, ohne dass ein realer Prototyp erstellt werden muss. Mit einem virtuellen Prototypen können erste Aussagen zur Skalierbarkeit und Performance eines neuen Systems getroffen werden.

Ziel ist dabei die Nutzung von Modellen mit geeigneten Abstraktionsebenen, um das Verständnis eines Systems zu verbessern und somit eine kosteneffektive Implementierung der Funktionalität zu ermöglichen [BM09]. Dabei kann das Abstraktionslevel zur Beschreibung des Modells beliebig sein und während des Entwicklungsprozesses zum Beispiel von einem funktionalen Modell zum zyklenakkuraten Modell präzisiert werden.

Für die vereinfachte Beschreibung von virtuellen Prototypen wurde Mitte der 90er Jahre das aktorbasierte Beschreibungsmodell, SystemC, entwickelt [Acc14]. SystemC ist eine Erweiterung von C++, sodass Komponenten mit aktuellen Programmierkonstrukten beschrieben werden und in das zu analysierende System integriert werden können. Der Vorteil durch SystemC ist die einfache Beschreibung und Kommunikation paralleler Funktionseinheiten und deren Kommunikation mit Hilfe von Transaktionen (Transaction Level Modeling, TLM) . Die unterstütze Abstraktion erstreckt sich von zeitlosen Modellen (*untimed Model*) für die Systemspezifikation, bis zum *Transfer Level Model*, das einem zyklenakkuraten Modell entspricht und als Referenzmodell zur Beschreibung auf Register-Transfer-Level (RTL) mittels Hardware Beschreibungssprache (Hardware Description Language; HDL) dienen kann.

Dieses Kapitel beschreibt die Simulationsmethodik mit dem vom Autor entwickelten virtuellen Prototypen eines heterogenen Mehrprozessorsystems. An diesem Prototypen wurden Kommunikationskonzepte für Mehrprozessorsysteme entwickelt, untersucht und optimiert.