Rainer Schaffer

Parallelisierung von Algorithmen zur Nutzung auf Architekturen mit Teilwortparallelität

Beiträge aus der Informationstechnik

Rainer Schaffer

Parallelisierung von Algorithmen zur Nutzung auf Architekturen mit Teilwortparallelität

VOGT

Dresden 2010

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 Die Deutsche Bibliothek
Die 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., 2010

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Parallelisierung von Algorithmen zur Nutzung auf Architekturen mit Teilwortparallelität“ von Rainer Schaffer überein.

Gesetzt vom Autor

ISBN 978-3-938860-37-3

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

Parallelisierung von Algorithmen zur Nutzung auf Architekturen mit Teilwortparallelität

Dipl.-Ing. Rainer Schaffer

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. W.-J. Fischer		
Gutachter:	Prof. Dr.-Ing. habil. R. Merker	Tag der Einreichung:	28.09.2009
	Prof. Dr. Ir. F. Catthoor	Tag der Verteidigung:	09.03.2010

Inhaltsverzeichnis

Zeichen, Bezeichnungen und Abkürzungen

Zeichen, Bezeichnungen

Bezeichnungen

e	Datenabhängigkeit
s	Zuweisung s
s_t	Teilzuweisung t der Zuweisung s
$f : \mathcal{X} \to \mathcal{Y}$	Abbildung f von $\mathcal{X}$ nach $\mathcal{Y}$
$y_s[.]$, $y_r[.]$, $y_a[.]$	Variablen eines Algorithmus

Skalare

m	Skaler (allgemein)

Vektoren

$\mathbf{m}$	Vektor (allgemein)
$\mathbf{0}$	Nullvektor $\mathbf{m}$ mit $\forall j : \ m_j = 0$
$\mathbf{1}$	Einsvektor $\mathbf{m}$ mit $\forall j : \ m_j = 1$
$\mathbf{i}$	Iterationsvektor
$\mathbf{i}^\kappa$	partitionierter Iterationsvektor
$\mathbf{d}$	Abhängigkeitsvektor (allgemein)
$\mathbf{d}(\mathbf{i}, \mathbf{i}')$	Abhängigkeitsvektor zwischen Iteration $\mathbf{i}$ und $\mathbf{i}'$
$\mathbf{d}(e)$	Abhängigkeitsvektor der Datenabhängigkeit e
$\mathbf{d}^\kappa$	partitionierter Abhängigkeitsvektor
$\mathbf{d}^\Theta$	innerer Teilvektor
$\widehat{\mathbf{d}}^\Theta$	äußerer Teilvektor
$\boldsymbol{\alpha}, \widehat{\boldsymbol{\alpha}} \in \mathbb{Z}^{1\times n}$	Abfolgevektor
$\boldsymbol{\iota}, \widehat{\boldsymbol{\iota}} \in \mathbb{Z}^{1\times n}$	Leerlaufverzögerung
$\boldsymbol{\tau}, \widehat{\boldsymbol{\tau}} \in \mathbb{Z}^{1\times n}$	Ablaufvektor

Matrizen

$\mathbf{M}$	Matrix (allgemein)
$\mathbf{E}$	Einheitsmatrix
$\mathbf{0}$	Nullmatrix
$\mathbf{\Gamma}$	Allokationsmatrix
$\mathbf{\Gamma}^M$	Datenausrichtungsmatrix
$\mathbf{\Theta}$	Partitionsmatrix (allgemein)
$\mathbf{\Theta}^P$	Partitionsmatrix (lokal parallel)
$\mathbf{\Theta}^S$	Partitionsmatrix (lokal sequenziell)

Mengen

$\mathcal{M}$	Menge (allgemein)
$\mathcal{E}$	Menge der Datenabhängigkeiten e
$\mathcal{D}^\kappa(e)$	Menge der partitionierten Abhängigkeitsvektoren $\mathbf{d}^\kappa$ resultierend aus der Datenabhängigkeit e
$\mathcal{H}$	konvexes Polyeder
$\mathcal{H}^\square$	Hyperquader
$\mathcal{I}$	Iterationsraum
$\mathcal{I}^\square$	Hyperquader, der den Iterationsraum $\mathcal{I}$ umschließt
$\mathcal{I}^\kappa$	partitionierter Iterationsraum
$\mathcal{K}$, $\widehat{\mathcal{K}}$	Partitionsraum
$\mathcal{K}^{\sigma(\mathbf{d})}$	Quellenteilraum
$\mathcal{K}^{\delta(\mathbf{d})}$	Senkenteilraum
$\mathcal{L}_s$	Indexraum der linksseitigen Variable $y_s[l(\mathbf{i})]$
$\mathcal{P}$	Rechenfeld
$\mathcal{R}^{k_{s'}}_{s's_t}$	Indexraum der rechtsseitigen Variable $y_{s'}[r^{k_{s'}}_{s's_t}(\mathbf{i})]$
$\mathcal{R}^{k_r}_{r\,s_t}$	Indexraum der Eingabevariable $y_r[r^{k_r}_{r\,s_t}]$
$\mathcal{S}$	Menge der Zuweisungen s eines Algorithmus
$\mathcal{Y}_L$	Menge aller Variablen $y_a[.]$, die auf der linken Seite einer Zuweisung $s \in \mathcal{S}$ eines Algorithmus auftreten.
$\mathcal{Y}_R$	Menge aller Variablen $y_a[.]$, die auf der rechten Seite einer Zuweisung $s \in \mathcal{S}$ eines Algorithmus auftreten.
$\mathcal{Y} = \mathcal{Y}_L \cup \mathcal{Y}_R$	Menge aller Variablen $y_a[.]$ eines Algorithmus
$\mathcal{Y}_E$	Menge aller Eingabevariablen $y_a[.]$ eines Algorithmus.
$\mathcal{Y}_A$	Menge aller Ausgabevariablen $y_a[.]$ eines Algorithmus.
$\mathcal{Y}_G = \mathcal{Y}_E \cup \mathcal{Y}_A$	Menge aller globalen Variablen $y_a[.]$ eines Algorithmus.
$\mathcal{Y}_I$	Menge aller internen Variablen $y_a[.]$ eines Algorithmus.
$\mathcal{C}^m_{\lvert\mathcal{M}\rvert}$	Menge der Kombination von m Elementen aus der Menge $\mathcal{M}$
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen

$\mathbb{Q}$	Menge der rationalen Zahlen
$\mathbb{Q}_+$	Menge der positiven rationalen Zahlen ohne Null
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{N}_+$	Menge der natürlichen Zahlen ohne Null

Funktionen

$f(\mathbf{m})$	Allgemeine Funktion angewendet auf Vektor $\mathbf{m}$
$\mathcal{M}_2 = \text{conv}(\mathcal{M}_1)$	Konvexe Hülle von $\mathcal{M}_1$
$m = \text{Rg}(\mathbf{M})$	Rang der Matrix $\mathbf{M}$
$\mathbf{m} = \text{diag}(\mathbf{M})$	Vektor der Hauptdiagonalen der Matrix $\mathbf{M}$
$\mathbf{M} = \text{diag}(\mathbf{m})$	Matrix mit Vektor $\mathbf{m}$ als Hauptdiagonale
$n = \dim(\mathcal{M})$	Dimension von $\mathcal{M}$
$f(\mathcal{M}_1)$	Menge der Elemente der Funktion $f(\mathbf{m})$ für alle $\mathbf{m} \in \mathcal{M}$
$n = \text{sign}(m)$	Vorzeichen eines Skalars m ($n \in \{-1, 0, 1\}$)
$s = \sigma(e)$	Quelle der Datenabhängigkeit e (Zuweisung)
$s_t = \delta(e)$	Ziel der Datenabhängigkeit e (Teilzuweisung)
$\text{ggT}(m_1, m2)$	größter gemeinsamer Teiler von m_1 und m_2
$\rho_s(\mathbf{i})$	Reindizierungsfunktion für Zuweisung s
$\gamma(\mathbf{i}^\kappa)$	Allokationsfunktion
$t_s(\mathbf{i})$	Ablaufplanfunktion
$\vartheta(\mathbf{i}^\kappa)$	Partitionierung
$\forall m$	Für alle m
$\exists m$	Es existiert mindestens ein m
$\exists! m$	Es existiert genau ein m
$\|m\|$	Betrag des Skalars m
$\|\mathcal{M}\|$	Anzahl der Elemente der Menge $\mathcal{M}$
$\mathcal{A} \oplus \mathcal{B}$	direkte Summe $\{\mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}$

Abkürzungen

BPA	Baum der partitionierten Abhängigkeitsvektoren
DSP	Digitaler Signalprozessor
DvB	Datenwort voller Breite
FE	Funktionseinheit
GLP	Ganzzahlig lineare Programmierung
KEA	Kantenerkennungsalgorithmus
LPGS	Lokal parallele, global sequenzielle Partitionierung
LSGP	Lokal sequenzielle, global parallele Partitionierung
MMC	Mehrstufige modifizierte Copartitionierung
PE	Prozessorelement
RAG	Reduzierter Abhängigkeitsgraph
STAF	Short Term Analysis Filtering

SPRF	Schwach programmoerbares Rechenfeld
RF	Rechenfeld
TW	Teilwort
TWP	Teilwortparallelität
UItA	Uniform-iterativer Algorithmus
const	konstant

Kapitel 1

Einleitung

Der technologische Fortschritt gestattet die Implementierung zunehmend komplexerer Prozessorarchitekturen auf einem Schaltkreis. Dabei werden sowohl immer umfangreichere Speicherarchitekturen auf den Chips implementiert als auch zusätzliche Funktionseinheiten eingebaut bzw. deren Funktionalität erweitert. Ein weiterer Schritt in dieser Entwicklung ist der Entwurf sogenannter „Systeme auf einem Schaltkreis“ (engl.: System on the Chip. kurz: SOC). Damit können nun Mehrkernsysteme (engl.: Multicore Systems) entwickelt werden, welche mehrere Prozesssorkerne besitzen. Diese Prozesssorkerne können gleiche oder unterschiedliche Aufgaben parallel abarbeiten.

Aus diesem Grund bieten heutige Prozessoren vielfältige Möglichkeiten zur Parallelverarbeitung. Abhängig von der Art der ausführbaren Operationen (gleiche oder unterschiedliche Funktionen) und deren Ansteuerungen (Einzelansteuerung, gemeinsamer Steuerbefehl oder Verwendung von VLIW-Befehlen (engl.: very large instruction word)) werden verschiedene Ansätze der Parallelverarbeitung unterschieden.

Das Ziel dieser Arbeit ist die systematische Abbildung von Algorithmen auf Rechenfelder (kurz: RF, engl.: processor arrays) mit Teilwortparallelität (kurz: TWP, engl.: sub-word parallelism). Die Abbildung 1.1 zeigt eine solche Architektur mit den drei Ebenen der Parallelverarbeitung:

(1) Parallelverarbeitung im Rechenfeld aufgrund mehrerer Prozessorelemente

(2) Parallelverarbeitung im Prozessorelement aufgrund mehrerer Funktionseinheiten

(3) Parallelverarbeitung in der Funktionseinheit aufgrund teilwortparalleler Operationen

Rechenfelder sind regulär angeordnete Prozessorelemente, die jeweils nur mit ihren Nachbarn verbunden sind. Der Aufbau (Anzahl und Art der Funktionseinheiten und Größe des lokalen Speichers) und die Funktion (ausführbare Befehle) der Prozessorelemente sind weitestgehend identisch.

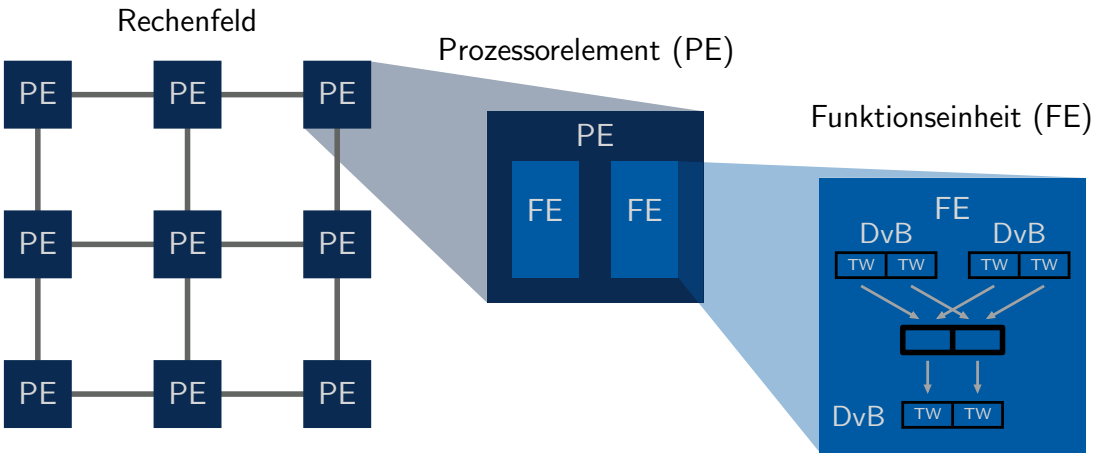

Abbildung 1.1: Rechenfeld mit Teilwortparallelität

Die Teilwortparallelität (kurz: TWP, engl.: sub-word parallelism) bezeichnet die Parallelverarbeitung innerhalb einer Funktionseinheit. Hierbei wird der Datenpfad der Funktionseinheit in mehrere schmale Datenpfade aufgeteilt, um Daten mit einer niedrigen Wortbreite auf einer Funktionseinheit mit hoher Datenpfadbreite parallel verarbeiten zu können. Die Daten niedriger Wortbreite bezeichnen wir als Teilwörter (kurz: TWs). Die Teilwörter, welche gleichzeitig verarbeitet werden sollen, werden in Datenwörtern voller Breite (kurz: DvBs) zusammengefasst. Die Berechnungen in den Funktionseinheiten werden auf Basis dieser DvBs durchgeführt. Die Operationen, welche auf die Teilwörter in einem Takt angewendet werden, sind meist identisch und hängen vom jeweiligen Befehlssatz der Funktionseinheit ab. Der Grad der Parallelität kann mit jedem Befehl geändert werden.

Da die Daten als Teilwörter in den DvBs zusammengefasst sind, müssen diese DvBs zwischen den Berechnungen meist umorganisiert werden. Dieses Umorganisieren wird auch als „Packen der Daten“ bezeichnet. Hierfür stehen spezielle Packbefehle zur Verfügung. Der Aufwand für das Packen der Daten sollte minimiert werden, denn die zusätzlichen Packoperationen können den Geschwindigkeitsgewinn aufgrund der parallelen Berechnung teilweise oder vollständig wieder zunichte machen.

Die teilwortparallele Verarbeitung von Daten wird auch in einer Vielzahl weiterer Architekturen genutzt. Als Beispiele seien hier, der Core2-Prozessor von Intel [Int07a], der Cell-Prozessor von IBM [Int07b] und der TMS320C64x-Prozessor von Texas Instruments [Ins01] erwähnt. Bei diesen Architekturen wurde die Hardware für die teilwortparallele Verarbeitung entsprechend erweitert. In [BDT06] wird eine Softwarerealisierung für die Teilwortparallelität vorgestellt, welche teilwortparallele Verarbeitung mit beliebigen Teilwortbreiten in fast jeder Funktionseinheit ermöglicht.

In dieser Arbeit wird eine systematische Abbildung von Algorithmen auf Rechenfelder mit Teilwortparallelität präsentiert. Die Abbildung basiert auf den Abbildungsmethoden für Rechenfelder, insbesondere auf der mehrstufigen modifizierten Copartitionierung [Sie03]. In dieser Arbeit wird die mehrstufige modifizierte Copartitionierung erweitert

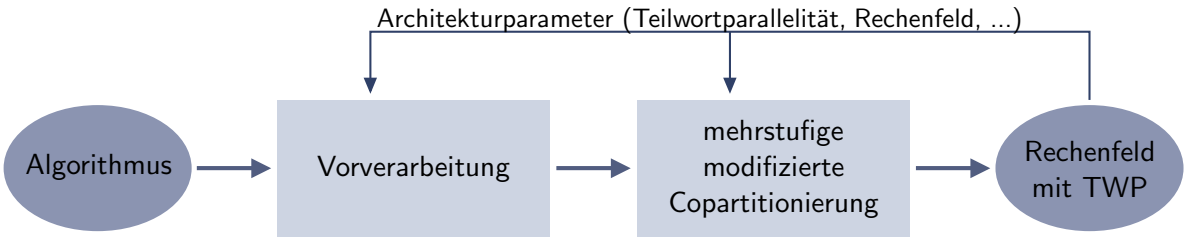

Abbildung 1.2: Systematische Abbildung eines Algorithmus

und systematisiert, um eine mehrstufige Parallelität zu realisieren und eine Anpassungen an lokale und globale Speicher zu ermöglichen. Des Weiteren wird in dieser Arbeit der Abbildungsschritt *Vorverarbeitung* entwickelt, welcher der mehrstufigen modifizierten Copartitionierung vorangestellt wird, um eine effiziente Abbildung des Algorithmus zu erzielen.

Abbildung 1.2 illustriert den Abbildungsprozess bestehend aus den zwei Schritten, der *Vorverarbeitung* und der *mehrstufigen modifizierten Copartitionierung*. Mit der Vorverarbeitung wird der Algorithmus so umformuliert, sodass die nachfolgende mehrstufige modifizierten Copartitionierung zu effizienten Abbildungen auf das Rechenfeld mit Teilwortparallelität führt. Die *Vorverarbeitung* ist zielarchitekturunabhängig. Auch wenn einzelne Parameter der Zielarchitektur mit in die *Vorverarbeitung* einbezogen werden, findet bei der *Vorverarbeitung* noch keine Anpassung an diese Parameter der Zielarchitektur statt. Erst bei der mehrstufigen modifizierten Copartitionierung wird der Algorithmus stufenweise an die Parameter der Zielarchitektur angepasst. Hierbei dient die erste Copartitionierungsstufe der Anpassung an die Teilwortparallelität. Die zweite Copartitionierungsstufe wird sowohl für die Ausnutzung der lokalen Speicher in den Prozessorelementen verwendet als auch für die Abbildung des Algorithmus auf das Rechenfeld genutzt. Nachfolgende Copartitionierungsstufen werden für die Anpassung des Algorithmus an den globalen Speicher verwendet.

Die Arbeit ist wie folgt gegliedert:

Zuerst wird in Kapitel 2 die Zielarchitektur ausführlich vorgestellt. Für die Funktionseinheiten mit Teilwortparallelität wurde eine XML-basierte Beschreibung entwickelt, aus der die Eigenschaften der Funktionseinheiten ermittelt werden können. Ein weiterer wichtiger Aspekt ist die Beschreibung der Speicherzugriffe, welche bei der Teilwortparallelität eine besondere Bedeutung haben, da die Daten auf Basis von DvBs oder auf Basis von Teilwörtern gelesen und geschrieben werden können.

Im nachfolgenden Kapitel 3 wird die Algorithmenklasse der *uniform, iterativen Algorithmen (kurz: UItA)* eingeführt. Diese Algorithmenklasse ist der Ausgangspunkt für die Abbildungsstrategie. Ein Algorithmus kann sehr unterschiedlich formuliert sein, wobei alle Formulierungen einen UItA darstellen. Diese verschiedenen Formulierungen können jedoch aufgrund ihrer Eigenschaften zu sehr unterschiedlichen Abbildungen führen. Aus diesem Grund wird eine Algorithmenkonvention präsentiert, welche der eigentliche Ausgangspunkt für die Abbildung des Algorithmus auf die Zielarchitektur ist. Man beachte, dass jeder Algorithmus der Algorithmenklasse so umformuliert werden kann, dass er die

Algorithmenkonvention erfüllt.

Für die Abbildung eines Algorithmus auf die Zielarchitektur wird der Algorithmus mehrfach transformiert. Die Grundtransformationen, das sind die Reindizierung und die mehrstufige Partitionierung, werden im Kapitel 4 vorgestellt. Bei der Vorstellung der mehrstufige Partitionierung wird im Abschnitt 4.2.2 eine neuartige Methode zur Partitionierung der Datenabhängigkeiten des Algorithmus präsentiert.

Auf Basis der mehrstufigen Partitionierung wird in Kapitel 5 die mehrstufige modifizierte Copartitionierung (kurz: MMC) vorgestellt. Die mehrstufige modifizierte Copartitionierung kombiniert die Grundversionen der Partitionierung, die lokal sequenzielle, global parallele (kurz: LSGP) und die lokal parallele, global sequenzielle (kurz: LPGS) Partitionierung miteinander. Weiterhin beinhaltet die MMC die Ablaufplanung und die Bindung. Mit dem Ablaufplan wird exakt festgelegt, zu welchem Zeitpunkt welche Zuweisung des Algoritmus in welcher Funktionseinheit von welchem Prozessorelement abgearbeitet wird. Die Zuordnung der Funktionseinheiten zu den Zuweisungen des Algorithmus wird als Bindung bezeichnet. In dieser Arbeit wird der Ablaufplan für eine allgemeinere Beschreibung durch die Leerlaufverzögerung $\boldsymbol{\iota}$ erweitert. Die Leerlaufverzögerung ermöglicht das Einfügen von Leertakten in den Ablaufplan. Eine Leerlaufverzögerung $\boldsymbol{\iota} \neq \mathbf{0}$ verringert die Effizienz des Algorithmus auf dem Rechenfeld, ist aber für die konfliktfreie Abbildung einzelner Algorithmen notwendig.

Die Ausnutzung der Teilwortparallelität im Abbildungsprozess mit dem Ziel der Beschleunigung der Abarbeitung des Algorithmus ist ein Schwerpunkt dieser Arbeit. Aus diesem Grund wird im Kapitel 6 die Transformation eines Ausgangsalgorithmus, der seine Operationen auf „einzelnen“ Daten (kurz: Einzeldatenoperationen) ausführt, in einen Algorithmus mit teilwortparallelen Operationen ausführlich abgeleitet. Diese Transformation basiert auf der ersten Stufe der mehrstufigen modifizierten Copartitionierung, wobei für diese Stufe die notwendigen Restriktionen angegeben werden. Bei der Transformation des Algorithmus werden die notwendigen Packoperationen für die teilwortparallele Version des Algorithmus systematisch ermittelt. Weiterhin wird die Transformation der Ein- und Ausgabezuweisungen des Ausgangsalgorithmus ausführlich vorgestellt. Diese Transformation hängt von der Art der Speicherung der Daten im Hauptspeicher sowie deren Lese- und Schreibmöglichkeiten ab. Beide Eigenschaften haben einen entscheidenden Einfluss auf die Beschleunigung des Algorithmus.

Die gesamte Abbildungsstrategie zur Abbildung eines Algorithmus auf ein Rechenfeld mit Teilwortparallelität wird nun in Kapitel 7 unter Verwendung der zuvor eingeführten Methoden ausführlich vorgestellt. Die Abbildungsstrategie gliedert sich, wie in Abbildung 1.2 dargestellt, in zwei Schritte, die Vorverarbeitung und die mehrstufige modifizierte Copartitionierung.

Die Vorverarbeitung (Abschnitt 7.1) wird in dieser Arbeit neu entwickelt. Sie unterteilt sich in zwei Teilschritte, die *Reindizierung* und die *Anpassung an die Algorithmenkonvention*. Ziel der Vorverarbeitung ist ein im Sinne der Abbildung auf das Rechenfeld gut formulierter Algorithmus, denn solch ein Algorithmus erlaubt eine schnelle und effiziente Abbildung auf das Rechenfeld mittels der mehrstufigen modifizierten Copartitionierung.

Für die Reindizierung wurde eine Optimierungsaufgabe basierend auf ganzzahliger linearer Optimierung entwickelt. Diese optimiert stufenweise mehrere Parameter des Algorithmus bezüglich der effizienten Abbildung des Algorithmus auf ein Rechenfeld bei der nachfolgenden mehrstufigen modifizierten Copartitionierung. Dabei fließen in die Optimierung teilweise schon Architekturparameter mit ein. Im Abschnitt „Anpassung an die Algorithmenkonvention“ wird der Algorithmus so umformuliert, dass er die Vorgaben der Algorithmenkonvention von Kapitel 3 erfüllt.

Der zweite Schritt der Abbildungsstrategie, die Anwendung der mehrstufigen modifizierten Copartitionierung, wird in Abschnitt 7.2 vorgestellt. Die mehrstufige modifizierte Copartitionierung wird in ihre einzelnen Stufen und in die abschließende Ablaufplanung unterteilt. Mit jeder Stufe der mehrstufigen modifizierten Copartitionierung erfolgt eine Anpassung des Algorithmus an Parameter der Zielarchitektur. So dient die erste Stufe der Ausnutzung der Teilwortparallelität und die zweite Stufe sowohl der Realisierung der Parallelverarbeitung im Rechenfeld als auch der Ausnutzung der lokalen Register. Weitere Copartitionierungsstufen können zur Anpassung an die Speicherarchitektur verwendet werden. Für die Anpassung an die Speicher wird eine Methode zur Bestimmung der Datenmengen vorgestellt, welche zum einen zwischen Prozessorelementen des Rechenfeldes und zum anderen zwischen Rechenfeld und Speicherarchitektur transferiert werden müssen. Diese Methode basiert auf der neuen Methode zur Partitionierung von Datenabhängigkeiten, die im Abschnitt 4.2.2 vorgestellt wird. Die abschließende Ablaufplanung ordnet den Zuweisungen des Algorithmus die Funktionseinheiten der Prozessorelemente zu und legt den exakten Ablaufplan fest.

In Kapitel 8 wird die in der Arbeit präsentierte Abbildungsstrategie auf drei praxisrelevante Algorithmen angewendet. Zwei Algorithmen werden auf ein Prozessorelement mit Teilwortparallelität abgebildet. Das dritte Beispiel zeigt die Abbildung des Kantenerkennungsalgorithmus auf eine Rechenfeld mit Teilwortparallelität. Alle Realisierungen wurden bezüglich ihres Geschwindigkeitsgewinns bewertet.

Abschließend erfolgt eine Zusammenfassung und es wird ein Ausblick auf offene Zielstellungen gegeben.