Technische Universität München
Lehrstuhl für Integrierte Schaltungen

Ein rekursives Verfahren zur Abbildung und zum Scheduling von Prozess-Graphen mit Kontrollabhängigkeiten

Thomas Wild

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzende: Univ.-Prof. Dr. rer. nat. Doris Schmitt-Landsiedel

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Ingolf Ruge, em.
2. Univ.-Prof. Dr.-Ing. Georg Färber

Die Dissertation wurde am 24.03.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.07.2003 angenommen.

Vorwort

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit am Lehrstuhl für Integrierte Schaltungen der Technischen Universität München entstanden. Darin sind neben aktuellen Fragen zu VLSI Architekturen insbesondere auch die in verschiedenen Projekten gewonnenen Erfahrungen aus dem Umfeld des Anwendungsbereichs Networking eingeflossen.

Mein erster und besonderer Dank gilt Herrn Prof. Dr.-Ing. I. Ruge, der mir die Durchführung dieser Arbeit ermöglicht, mich stets ermutigt und mir die entsprechende Geduld entgegengebracht hat. Die langjährige und vertrauensvolle Zusammenarbeit mit ihm hat mir auf den unterschiedlichsten Ebenen interessante und wichtige Erfahrungen und Kontakte ermöglicht.

Recht herzlich möchte ich mich auch bei Herrn Prof. Dr.-Ing. G. Färber, dem Inhaber des Lehrstuhls für Realzeit-Computersysteme der TUM, für die Übernahme des Zweitberichts und das damit bekundete Interesse an der Arbeit bedanken.

Ein besonderer Dank geht auch an Dr. Rudi Knorr, Dr. Walter Stechele und Dr. Helmut Steckenbiller für die stets hilfreichen Diskussionen und die vielen guten Tipps, die zum Gelingen der Arbeit beigetragen haben.

Für die enge und effektive Zusammenarbeit sowie die vielfältigen Diskussionen und den regen Austausch über den großen Teich hinweg möchte ich mich bei Winthir Brunnbauer, unserem Mann im Silicon Valley, recht herzlich bedanken. Auch der intensive und fachlich anregende Austausch mit den weiteren Kollegen der Networking-Gruppe, Jürgen Foag und Nuria Pazos, waren sehr hilfreich für mich.

Darüber hinaus möchte ich mich bei Dr. Werner Bachhuber, Winthir Brunnbauer, Dr. Hubert Mooshofer und Werner Mühleisen bedanken, die die Mühen des Korrekturlesens auf sich genommen und mich dabei noch mit vielen guten Anregungen versorgt haben.

Der Admin-Gruppe des Lehrstuhls gilt mein besonderer Dank für die gute Zusammenarbeit und den zuverlässigen Betrieb der Rechner und des Rechnernetzes.

Ein herzliches Dankeschön möchte ich auch allen wissenschaftlichen und nicht-wissenschaftlichen Mitarbeitern des Lehrstuhls, aktuellen wie auch ehemaligen, für die stets angenehme Zusammenarbeit und das gute Arbeitsklima aussprechen. Besonders möchte ich mich bei Verena Draga und Dr. Walter Stechele für die in jeder Beziehung hervorragende Kooperation in administrativen und organisatorischen Dingen sowie in Angelegenheiten der Lehre bedanken.

München, im März 2003

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der verwendeten Abkürzungen

ADSL	Asymmetric Digital Subscriber Line
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction Set Processor
ATM	Asynchroner Transfer Modus
CAM	Content Adressable Memory
CDFG	Control Data Flow Graph
CFSM	CoDesign Finite State Machine
CP	Critical Path
CPG	Conditional Process Graph
CPU	Central Processing Unit
DFG	Data Flow Graph
DSP	Digitaler Signalprozessor
ECP	Extended Critical Path
FAST	Fast Assignment using Search Technique
FBM	Funktions-basiertes Modell
FIFO	First In First Out Speicher
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
HW	Hardware
IC	Integrated Circuit
ICMP	Internet Control Message Protocol
ID	Identifikator
IP	Intellectual Property
IP	Internet Protocol
MoC	Model of Computation
MPLS	Multiprotocol Label Switching
OBM	Objekt-basiertes Modell
QoS	Quality of Sevice
RFC	Recursive Flow Classification
RFC	Request For Comments
RTL	Register Transfer Ebene
SA	Simulated Annealing
SDL	Specification and Description Language
SoC	System on Chip
SPADE	System Level Performance Analysis and Design Space Exploration
SPI	System Property Intervals

SW	Software
TASK	Topological Assignment and Scheduling Kernel
TDT	Timed Decision Table
TGFF	Task Graphs for Free (Werkzeug zur Erzeugung künstlicher Graphen)
TOS	Type Of Service
TS	Tabu Suche
UML	Unified Modeling Language
VHDL	Very High Speed Integrated Circuits HDL
VLSI	Very Large Scale Integration
VSIA	Virtual Socket Interface Alliance
WWW	World Wide Web

Kurzfassung

Integrierte digitale Systeme werden mit der fortschreitenden Strukturverkleinerung in der Mikroelektronik zunehmend komplexer und leistungsfähiger. Den damit verbundenen neuen Implementierungsmöglichkeiten stehen heute jedoch nicht in ausreichendem Maß die entsprechend leistungsfähigen Entwurfsmethoden für die Realisierung derartiger Systeme gegenüber. Deshalb laufen intensive Anstrengungen, die Entwurfsproduktivität durch geeignete Maßnahmen zu erhöhen. Ziel ist es dabei insbesondere, komplexe Systeme mit Hilfe des Einsatzes vorgefertigter Module schneller und effizienter zu entwickeln und den Einstiegspunkt für den Einsatz automatisierter Entwurfwerkzeuge auf zunehmend höhere Abstraktionsebenen zu verlagern. Werkzeuge für diese Aufgabe sind jedoch bisher nicht kommerziell verfügbar.

Komplexe Bausteine, oft als System on Chip (SoC) bezeichnet, erlauben die Implementierung komplexer Systeme, die bisher verteilt auf einer Platine realisiert wurden, auf einem gemeinsamen Substrat. Durch den Einsatz unterschiedlicher Module als Bestandteil von SoCs ergeben sich neue Freiheitsgrade beim Entwurf. Der Einsatz eingebetteter Prozessoren neben dedizierter Hardware ermöglicht die Anpassung derartiger Bausteine durch Änderungen des Software-Anteils.

Aufgrund dieser unterschiedlichen Implementierungsvarianten ergeben sich eine Vielzahl von Möglichkeiten, wie eine vorgegebene Funktion auf eine geeignete Architektur umgesetzt werden kann. Um beim Entwurf die unter vorgegebenen Randbedingungen am besten geeignete Variante auszuwählen, sind vor der eigentlichen Implementierungsphase ausgiebige Untersuchungen erforderlich. Eine hierbei besonders wichtige Frage ist die Aufteilung der gesamten Funktionalität in die als HW bzw. SW zu implementierenden Teile.

Die vorliegende Arbeit beschäftigt sich mit dieser Fragestellung insbesondere im Zusammenhang mit Anwendungen aus dem Bereich der Datenkommunikation, die kontrolldominiert sind. Bisher sind sehr wenige Ansätze bekannt, die diese Eigenschaft bei der Zuordnung der Teilfunktionen auf entsprechende Bausteinarchitekturen unterstützen.

Daneben konzentriert sich die Arbeit insbesondere auf den Aspekt der internen Kommunikation in komplexen SoC Architekturen, der zunehmend als ein für deren Leistungsfähigkeit entscheidender Faktor erkannt wird. Aus diesem Grund wird die Optimierung von Kommunikationsarchitekturen für derartige Systeme vermehrt Gegenstand der Forschung. Andererseits kommt im Hinblick auf die möglichen System-Komplexitäten dem effizienten Entwurf geeigneter Kommunikationsarchitekturen zunehmend große Bedeutung zu. Die hier entwickelte Methode zu Partitionierung von HW/SW Systemen trägt dem Kommunikationsaspekt durch intensivere Berücksichtigung der Transferzeiten besonders Rechnung.

Die Methode geht von einem die zu implementierende Funktionalität beschreibenden Prozeß-Graphen und einer vorgegebenen Zielarchitektur aus. Sie beruht auf einem rekursiven Ansatz, bei dem die Performance, d.h. die Verarbeitungsdauer der Lösung im ungünstigsten Fall, iterativ durch lokale Suche optimiert wird. Die einzelnen Schritte bestehen hierbei aus der Abbildung der Teilprozesse auf die vorhandenen Verarbeitungsressourcen, der Festlegung der Verarbeitungsreihenfolge (Scheduling) sowie der eigentlichen Optimie-

rung. Die Unterstützung der Kontrollabhängigkeiten erfolgt über das Prinzip der gemeinsamen Ressourcennutzung für Prozesse, die sich gegenseitig ausschließen. Für die vorliegenden Anforderungen waren die bekannten Verfahren zur Detektion der gegenseitigen Exklusivität zweier Prozesse nicht ausreichend, weshalb hierfür eine neue Methode entwickelt wurde, die beliebige konditionale Übergänge im Prozeß-Graphen erlaubt.
Als Suchverfahren wurden sowohl Simulated Annealing als auch Tabu Suche betrachtet, wobei der letztgenannte Ansatz aufgrund besserer Performance bei kürzeren Bearbeitungszeiten weiterverfolgt wurde. Lokale Suche basiert grundsätzlich auf der Definition einer geeigneten Nachbarschaft, innerhalb derer - ausgehend von der jeweils aktuellen Lösung - iterativ nach Verbesserungen in Richtung des Optimums gesucht wird. In der vorliegenden Arbeit wurden hierfür unterschiedliche Nachbarschaftsdefinitionen untersucht. Insbesondere wurde eine neue Nachbarschaft definiert, die auf dem kritischen Pfad des Graphen basiert und um Prozesse, die zusätzliche Verzögerungen erzeugen können, erweitert ist. Auf diese Weise werden dynamische Effekte insbesondere auf der Kommunikationsarchitektur und deren Einfluß in die Entscheidung über die Zuordnung der Prozesse auf die Ressourcen der Architektur einbezogen. Alternativ zur Optimierung mit Tabu Suche wurde mit FAST ein statistisches Suchverfahren betrachtet, das ebenfalls im Umfeld des kritischen Pfades arbeitet.
Alle Verfahren wurden anhand künstlich erzeugter Graphen sowie einer konkreten Anwendung aus dem Bereich der Datenkommunikation (Diffserv Funktionalität auf einer Router Schnittstellenkarte) untersucht und für unterschiedliche Varianten mit der konventionellen Methodik verglichen. Mit Hilfe von Experimenten mit unterschiedlichen Konfigurationen wurde der Einfluß der Eigenschaften des Graphen bzw. der Zielarchitektur auf die Algorithmen untersucht. Dabei ergaben sich durchgehend bessere Performancewerte für das neue rekursive Verfahren. Alle in der Arbeit betrachteten Nachbarschaftsdefinitionen für die Tabu Suche als auch das FAST Sucherverfahren führten zu besseren Resultaten als ein aus der Literatur bekanntes konstruktives List Scheduling Verfahren. Insbesondere die erweiterte Nachbarschaft des kritischen Pfades führt zu guten Ergebnissen. Bei der Diffserv-Anwendung lagen die Vorteile dieses Verfahrens bei 15% gegenüber dem konstruktiven Algorithmus.
Als Nachteil konnten generell die sehr hohen Laufzeiten des rekursiven Verfahrens festgestellt werden, der diesen Ansatz nur für Problemstellungen mit Graphgrößen von mehreren Hundert Knoten einsetzbar macht. Wie mit dem Diffserv-Anwendungsbeispiel gezeigt, können jedoch auch reale Anwendungen mit der Methode behandelt werden.

1 Einleitung

1.1 Überblick und Zielsetzung

Die kontinuierlichen Fortschritte der Mikroelektronik erlauben die Implementierung zunehmend komplexer Schaltungen auf einem VLSI Baustein. Damit wird die Realisierung kompletter Systeme auf einem gemeinsamen Substrat möglich, wodurch die von der Komponentenebene bekannten Vorteile der Integration auch auf der Systemebene zum Tragen kommen. Mit Hilfe dieser als System on Chip (SoC) bezeichneten Bausteine kann die erhöhte Leistungsfähigkeit der Mikroelektronik ausgenutzt und für die Umsetzung bisher nicht möglicher oder ganz neuer Anwendungen eingesetzt werden.
Mit den enormen Fortschritten bei der Produktion mikroelektronischer Bausteine gehen jedoch eine Vielzahl von ungelösten Problemen in Zusammenhang mit deren Entwurf einher. Die Produktivität der Designer kann mit den bisher beim Chipdesign verwendeten Methoden der Zunahme der möglichen Chipkomplexitäten nicht folgen. Die Überbrückung dieser Produktivitätslücke ist deshalb das vordringliche Ziel der Forschung in diesem Bereich. Um eine effiziente Nutzung der infolge steigender Integrationsdichten neuen Möglichkeiten unter den zunehmend härter werdenden Anforderungen des Marktes zu erreichen, müssen neue Entwurfsmethoden entwickelt werden, die die Beherrschung der Komplexitäten ermöglicht.
Um diese Anforderung zu erfüllen, werden hauptsächlich zwei grundsätzliche Ansätze verfolgt. Zum einen soll durch Wiederverwertung bereits vorgefertigter Module der Designaufwand reduziert werden. Dieser Ansatz führt über die damit verbundene Erweiterung der Freiheitsgrade zu einer erheblichen Ausweitung der Implementierungsvarianten und damit zu einem wesentlich vergrößerten Lösungsraum. Gleichzeitig besteht jedoch bei derartigen, integrierten Lösungen die im Vergleich zu verteilten Realisierungen verschärfte Notwendigkeit, weitere Entwurfszyklen für die Beseitigung von Designfehlern oder die nachträgliche Änderung der Spezifikation zu vermeiden.
Zum anderen und in Ergänzung hierzu wird angestrebt, den Einstiegspunkt bei der Definition entsprechender Lösungen auf zunehmend höhere Abstraktionsebenen zu verschieben sowie durch Einsatz geeigneter Werkzeuge den Entwurf zu unterstützen und mit entsprechenden Synthese- und Verifikationsverfahren zu automatisieren.
Der Auswahl geeigneter Implementierungsvarianten vor der eigentlichen Realisierungsphase kommt deshalb entscheidende Bedeutung zu. Um aus dem großen Lösungsraum die unter den jeweiligen Randbedingungen am besten geeignete Variante auszuwählen, sind auf hoher Abstraktionsebene ausführliche Untersuchungen zu unterschiedlichen Architekturalternativen, auf denen die benötigte Funktionalität implementiert werden kann, erforderlich. Die vorliegende Arbeit beschäftigt sich mit der in diesem Zusammenhang entscheidenden Frage der Unterteilung der Gesamtfunktionalität in die als HW oder als SW zu implementierenden Teilfunktionen, auf die die jeweilige Realisierung folgt.
Als Anwendungsgebiet für die im Rahmen dieser Arbeit untersuchten Methoden dient der Bereich Datenkommunikation. Dieses Themenfeld stand in den letzten Jahren insbesondere im Mittelpunkt des Interesses. Dies nicht nur aus Gründen der wirtschaftlichen Relevanz

sondern vielmehr auch aufgrund der Tatsache, daß von VLSI Architekturen in diesem Gebiet in zunehmenden Maß besondere Anforderungen nach hoher Verarbeitungsleistung und gleichzeitig großer Flexibilität zu erfüllen sind. Eine besondere Eigenschaft dieses Gebietes ist von besonderer Bedeutung für die hier durchgeführten Untersuchungen: Da die Verarbeitung von Datenpaketen eine stark kontrolldominierte Aufgabe darstellt, müssen Verfahren zur Unterstützung des Entwurfs derartiger Lösungen diese Eigenschaft in geeigneter Weise unterstützen. Bisher wurden in der Literatur nur vereinzelt Verfahren vorgeschlagen, die diese Anforderung erfüllen. Die Berücksichtigung dieser Eigenschaft stellt deshalb eine spezielle Randbedingung für diese Arbeit dar.
Daneben hat auch der Aspekt der systeminternen Kommunikation als weiterer Motivationspunkt diese Arbeit entscheidend mitbeeinflußt. Die interne Kommunikation wird zunehmend als ein Faktor erkannt, von dem die Leistungsfähigkeit einer Lösung in besonderem Maß abhängt. Dies gilt insbesondere für VLSI Architekturen für den genannten Anwendungsbereich, bei denen im Zusammenhang mit der Bearbeitung der Pakete aufgrund der verteilten Verarbeitung und der Notwendigkeit von Tabellenzugriffen ein besonders hoher Kommunikationsbedarf besteht. In den letzten Jahren wurden deshalb eine zunehmende Anzahl von Arbeiten zum Thema Kommunikation veröffentlicht. Meist handelt es sich hierbei um Ansätze, die entweder in Bezug auf den Entwurfsablauf dort ansetzen, wo die entscheidenden Fragen der Festlegung der Architektur bereits getroffen wurden und deren Verfeinerung in Richtung Realisierung verbessern, oder die neue Chipinterne Kommunikationsstrukturen zum Gegenstand haben
Im Gegensatz hierzu wird in der vorliegenden Arbeit der Aspekt der Kommunikation bereits bei der Festlegung der Architektur, d.h. bei der Aufteilung der Anwendung und der Zuordnung der Teilfunktionen auf HW- und SW-Ressourcen berücksichtigt. Aufgrund der Tatsache, daß diese Festlegung über die Notwendigkeit der Kommunikation über die interne Kommunikationsarchitektur entscheidet, wird eine frühzeitige und die relevanten Effekte berücksichtigende Einbeziehung dieses Aspekts als besonders wichtig erachtet.

Zielsetzung der vorliegenden Arbeit ist deshalb die Entwicklung eines Verfahrens, das die Definition geeigneter Architekturen im Rahmen der Explorationsphase unterstützt und dabei die für den Bereich der Datenkommunikation bzw. Paketverarbeitung entscheidende Eigenschaft besitzt, Kontrollabhängigkeiten in der Anwendung zu erfassen. Die Hauptaufgabe bei der Architekturexploration besteht in der Aufteilung der zu implementierenden Funktion in die als HW oder SW zu realisierenden Teile und deren Optimierung. Bisher gibt es nur sehr wenige Ansätze, die diese Anforderungen erfüllen. Ein weiteres Ziel besteht deshalb darin, bessere Lösungen als diese bekannten Verfahren zu generieren.

1.2 Vorgehensweise und Aufbau der Arbeit

Die Vorgehensweise zur Durchführung dieser Arbeit besteht aus drei Phasen: Ausgangspunkt ist die Untersuchung der Hauptmotivationspunkte Unterstützung von Kontrollabhängigkeiten und Beeinflussung der erreichbaren Performance durch die systeminterne Kommunikation. Daraus wird zunächst ein Rahmen für das rekursive Verfahren definiert, der die zu erfüllenden Teilaufgaben Abbildung der Funktionen und deren Optimierung so-

wie die Festlegung der Verarbeitungsreihenfolge im System (Scheduling) umfaßt, und unterschiedliche Optionen der Suche definiert. Nach der Festlegung des Verfahrens werden Experimente anhand einer Vielzahl künstlich generierter Anwendungsfälle mit unterschiedlichen Eigenschaften sowie einer konkreten Anwendung aus dem Bereich der Paketverarbeitung durchgeführt.

Diese Vorgehensweise spiegelt sich entsprechend im Aufbau der Arbeit wieder. Zunächst wird im 2. Abschnitt der Stand der Technik im Bereich des Entwurfs integrierter Schaltungen und SoCs bzw. eingebetteter Systeme dargestellt. Dieses Kapitel bringt daneben einen kurzen Überblick über die Trends des hier im Mittelpunkt des Interesses stehenden Anwendungsbereichs Datenkommunikation. Das 3. Kapitel beinhaltet mit der ersten der o.g. Phasen die für die Arbeit maßgeblichen Einflußfaktoren sowie die für die Durchführung der Arbeit getroffenen Randbedingungen. Den Kern der Arbeit stellt das Kapitel 4 dar, in dem das neue rekursive Verfahren zur Abbildung und zum Scheduling von kontrolldominierten Anwendungen beschrieben wird. Für dieses, auf lokaler Suche basierende Verfahren werden darin auch unterschiedliche Varianten vorgestellt, bevor im 5. Abschnitt Untersuchungen zu dessen Verifikation und zum Vergleich mit alternativen Ansätzen durchgeführt werden. Das 6. Kapitel schließt die Arbeit mit einer Bewertung und einem Ausblick auf weitere Optimierungsmöglichkeiten ab.

2 Stand der Technik

In diesem Kapitel wird der gegenwärtige Stand der Technik im Bereich Entwurfsverfahren und VLSI Architekturen dargestellt. Hierbei werden zunächst die allgemeinen Trends beschrieben, bevor im weiteren insbesondere auf den Bereich des Entwurfs von komplexen HW/SW-Bausteinen und die damit verbundenen Teilaufgaben näher eingegangen wird. Der Entwurf von gemischten HW/SW-Systemen ist grundsätzlich kein neues Thema. Derartige Systeme sind meist als verteilt implementierte Lösungen unter dem Begriff eingebettete Systeme (embedded Systems) schon längere Zeit Gegenstand der Forschung. Mit den neuen Möglichkeiten der Mikroelektronik ergeben sich jedoch aufgrund der Komplexitätssteigerung von einzelnen Bausteinen neue Anforderungen, die über die bisherigen Entwurfsmethoden hinausgehende Ansätze erforderlich machen.
Da für die vorliegende Arbeit das Anwendungsgebiet Datenkommunikation und dafür geeignete Architekturen im Mittelpunkt des Interesses stehen, werden nachfolgend die Trends in diesem Gebiet und die sich daraus ergebenden Anforderungen dargestellt.
Aus dem dargelegten Stand der Technik wird im Kapitel 3 die Motivation für die Erweiterung der bisherigen Entwurfsmethoden abgeleitet und die für die vorliegende Arbeit angenommenen Randbedingungen dargestellt.

2.1 VLSI Entwurf

2.1.1 Entwicklung der Mikroelektronik

Die rasanten Fortschritte im Bereich der Mikroelektronik erlauben die Realisierung von Bausteinen mit stetig zunehmender Komplexität. Die regelmäßig weiterentwickelte Prozeßtechnik bei der Produktion von integrierten Schaltkreisen ermöglicht die Herstellung immer kleinerer minimalen Strukturgrößen und bildet damit die Voraussetzung für zunehmend leistungsfähigere Bausteine. Diese Verbesserungen ermöglichen insbesondere eine steigende Anzahl von Transistoren, die auf einem Chip integriert werden können.
Die maximale Anzahl der auf einem Chip integrierbaren Transistoren als Maßzahl für die Mächtigkeit der Mikroelektronik verdoppelt sich gemäß dem Gesetz von Gordon Moore alle 18 Monate ([83]). Diese empirische Gesetzmäßigkeit wurde ursprünglich im Jahr 1965 aufgestellt und hat bisher grundsätzlich seine Gültigkeit behalten. Bild 2-1 zeigt das exponentielle Wachstum der Chipkomplexität anhand der Größen der bisherigen bzw. prognostizierten Generationen von Speicherbausteinen sowie der bisherigen Intel-Prozessoren.
Mit der Zunahme an Komplexität geht anderseits auch eine Erhöhung der Performance durch Erhöhung der maximalen Taktraten einher. Durch Reduktion der Versorgungsspannung sind diese Verbesserungen möglich, ohne daß die in den Bausteinen entstehende Verlustleistung übermäßig ansteigt. Auch andere Parameter, die die Einsetzbarkeit komplexer Bausteine stark beeinflussen, wie neue Gehäusetechniken, die eine immer größere Anzahl von Pins oder die Abfuhr immer höherer Verlustleistung ermöglichen, sind hier zu nennen.
Ein Ende der zunehmenden Strukturverkleinerungen ist trotz zunehmender Bedenken bisher nicht in Sicht, auch wenn sich im Bereich unter 0,25 µm minimaler Strukturgröße prinzipielle Probleme bei der Prozessierung der Bausteine ergeben, die heute noch nicht

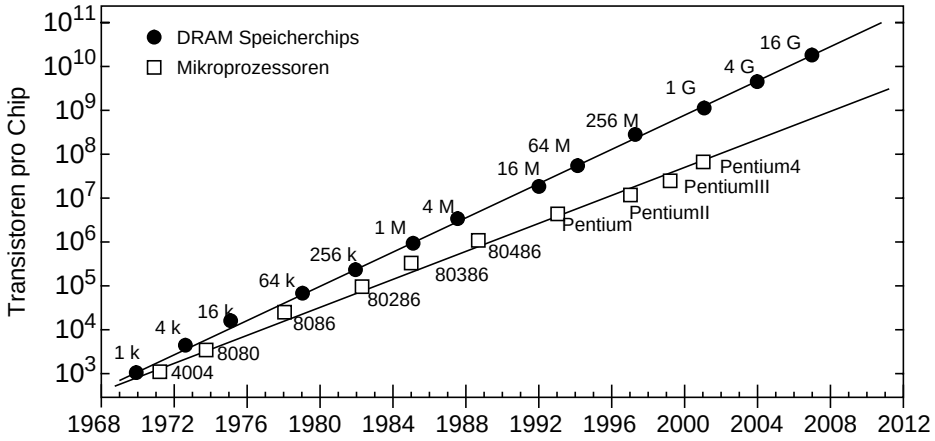


Bild 2-1: Das Gesetz von Gordon Moore für Speicher und Prozessoren

abschließend gelöst sind. Aus der Erfahrung der bisherigen Entwicklung hat sich jedoch gezeigt, daß für die jeweils anstehenden Problemstellungen immer eine neue Lösung gefunden wurde.

Die durch steigende Chip-Komplexitäten zunehmende Integration ermöglicht die weitere Miniaturisierung der Systeme, wodurch auch neue Anwendungsgebiete für die Mikroelektronik erschlossen werden. Wo früher zur Realisierung bestimmter Funktionen eine Platine mit mehreren Bausteinen benötigt wurde, kann zunehmend ein einzelner Baustein eingesetzt werden, der die komplette Funktion auf einem gemeinsamen Substrat realisiert. Neben der Verkleinerung des Systems kann aufgrund des Wegfalls von Chipgrenzen eine Erhöhung der Zuverlässigkeit und vor allen auch eine Erhöhung der Performance erreicht werden. Durch die Integration von Funktionen auf einem gemeinsamen Substrat entfallen die bei einer Implementierung auf einer Platine notwendigen Verbindungen über das Board. Aufgrund der geringeren kapazitiven Last von chipinternen Verdrahtungen sind damit weniger und kleinere Treiberbausteine erforderlich, was zu den genannten Verbesserungen beiträgt.

Neben den durch die bessere Prozeßtechnik möglichen größeren Komplexitäten profitiert die Mikroelektronik jedoch auch von der Umsetzung neuer Konzepte im Bereich der Prozessoren, die neben den Speichern als weiterer Gradmesser für den Fortschritt betrachtet werden. Durch neue Architekturkonzepte wird die Rechenleistung von Prozessoren stetig verbessert, wodurch sich zunehmend neue Anwendungsgebiete eröffnen. Prozessoren sind heute Bestandteil in vielen Produkten, die auf diese Weise an Flexibilität und Bedienbarkeit hinzugewinnen.

Der Zunahme der Leistungsfähigkeit mikroelektronischer Systeme stehen aber andererseits die gestiegenen Anforderungen aus den verschiedensten Anwendungsgebieten gegenüber, die die neuen Möglichkeiten auch ausnutzen. So sind Anwendungen aus dem Bereich Datenkommunikation und Internet, Multimedia und Bildverarbeitung oder auch Mobilfunk mit immer höheren Datenraten und komfortableren Eigenschaften nur aufgrund neuester Technologien möglich. Grundsätzlich gehen die Anforderungen zum einen hin zur Integra-

tion von mehr Funktionalität in einem Chip und zu größerer Verarbeitungsleistung, andererseits zu geringeren Verlustleistungen. Neueste Technologien ermöglichen die Erreichung beider Zielsetzungen und erlauben auf diese Weise neue Anwendungen, die aufgrund ihrer Anforderungen nach Größe, Portabilität oder Laufzeit bisher nicht oder nicht ausreichend realisierbar waren.
Von den neuen Möglichkeiten der Mikroelektronik profitiert aber nicht nur Bestehendes, sondern es erschließen sich immer mehr neue Anwendungsfelder für die Mikroelektronik. Das Spektrum reicht hierbei von Konsumgütern bis hin zur Industrieelektronik und zum Anlagenbau. Beispiele sind tragbare Geräte wie Mobilfunkgeräte, PDAs, MP3-Player oder auch Vermittlungsanlagen, bei denen sich der Platzbedarf durch Intergration von zunehmend mehr Teilnehmeranschlüssen auf einer Linecard immer weiter reduzieren läßt. Im Automobilbereich tragen entsprechende Systeme zur Erhöhung der Verkehrssicherheit bei.

Zum Aufbau komplexer Systeme mit hohen Anforderungen an die Verarbeitungsleistungen waren lange Zeit spezifische, für den jeweiligen Anwendungsfall entworfene Lösungen erste Wahl. Diese kundenspezifischen Bausteine (ASIC) sind jedoch zeitaufwendig zu entwerfen und müssen voll- oder semikundenspezifisch produziert werden. Daneben kommen jedoch zunehmend alternative Lösungskonzepte zum Einsatz. Wenn z.B. besondere Anforderungen an die Flexibilität der Systeme zu erfüllen sind, werden oft SW-programmierbare Lösungen mit digitalen Signalprozessoren (DSP) oder für einen breiteren Einsatzbereich entwickelte, programmierbare Spezialprozessoren (ASIP, Application Specific Instruction Set Processor) eingesetzt. Daneben kommen zunehmend feldprogrammierbare Bausteine (FPGA) zum Einsatz, die immer höhere Anforderungen an implementierbarer Komplexität und erreichbarer Verarbeitungsgeschwindigkeit erfüllen können und gleichzeitig den Entwicklungsprozeß entscheidend vereinfachen. Für Anwendungsbereiche mit höheren Stückzahlen werden andererseits von Anbietern Standardbausteine auf den Markt gebracht, die durch Rekonfigurierbarkeit bzw. Programmierbarkeit eine Vielzahl von Anwendungsfällen unterstützen ([13]). Durch standardisierte Schnittstellen und mit Hilfe von vorgefertigten Anwendungsbibliotheken sowie entsprechenden Entwurfswerkzeugen lassen sich somit vom Anwender schnell und flexibel Lösungen für die benötigte Systemumgebung generieren.
Mit den genannten Fortschritten in der Technologie werden völlig neue Bausteine möglich, die ganze Systeme auf einem gemeinsamen Silizium-Substrat beinhalten. Für derartige Bausteine hat sich der Begriff System-on-Chip (SoC) eingebürgert, der diese Möglichkeiten beschreibt. Der Begriff System-on-Chip ist jedoch nicht exakt definiert und wird oft in verschiedener Hinsicht gebraucht. Unterschiedliche Definitionen beinhalten z.B. eine bestimmte Mindestanzahl von Transistoren, die Integration von mindestens einem eingebetteten Prozessor bzw. anderer von Drittanbietern bereitgestellter Teilkomponenten. Vielfach werden darunter auch Bausteine verstanden, die komplette, bisher in mehreren Chips verteilt auf einer Platine realisierte Systeme beinhalten. Allen Definitionen ist die große Komplexität einer derartigen Lösung gemeinsam, die mit der Wiederverwertung von Komponenten, entweder selbst entwickelt oder von extern zugekauft, und der Implementierung von Teilen der Lösung in SW verbunden ist. Der Begriff SoC soll im weiteren in diesem Sinne verstanden werden.

Komplexe SoC Bausteine stellen ganz neue Anforderungen an die Entwurfsmethodik. Die Beherrschung der Komplexität bei der Spezifikation und der Verifikation derartiger Lösungen ist eine der größten Herausforderungen. Bereits heute liegt der Anteil der Verifikation am gesamten Entwurfsaufwand bei ca. 70%. Daneben werden heute jedoch auch aufgrund wirtschaftlicher Faktoren verschärfte Anforderungen an den Entwurfsprozeß gestellt. Infolge des zunehmend größer werdenden Marktdrucks besteht die Notwendigkeit, die Entwurfszyklen immer mehr zu verkürzen. Die Zeit zwischen der Definition eines Produkts und dessen tatsächlicher Verfügbarkeit, oft als "Time-to-Market" bezeichnet, ist wesentlich entscheidend für den Markterfolg eines Design-Projekts. Waren vor einigen Jahren für ASIC Designs noch Zyklen von 24 Monaten üblich, geht man heute von 12 Monaten aus ([95]). Unter diesen verschärften Randbedingungen ist es zwingend erforderlich, daß zeitaufwendige Neu-Entwürfe infolge von Fehlern und falschen Design-Entscheidungen vermieden werden. Neben der Entwurfsdauer ist auch im Hinblick auf die Kosten für die Produktion, insbesondere die Erstellung des Maskensatzes für einen kundenspezifisch entwickelten Baustein, der Erfolg der ersten Implementierung von größter Bedeutung.
Betrachtet man unter diesen Randbedingungen die Weiterentwicklung der technischen Möglichkeiten im Vergleich zur Produktivität von Designteams, so ist festzustellen, daß diese der Zunahme der möglichen Komplexität von Chips nicht folgen konnte. Vielmehr tut sich zwischen der möglichen Chipkomplexität und der Produktivität der Designer eine sich verbreiternde Lücke auf ([95]).

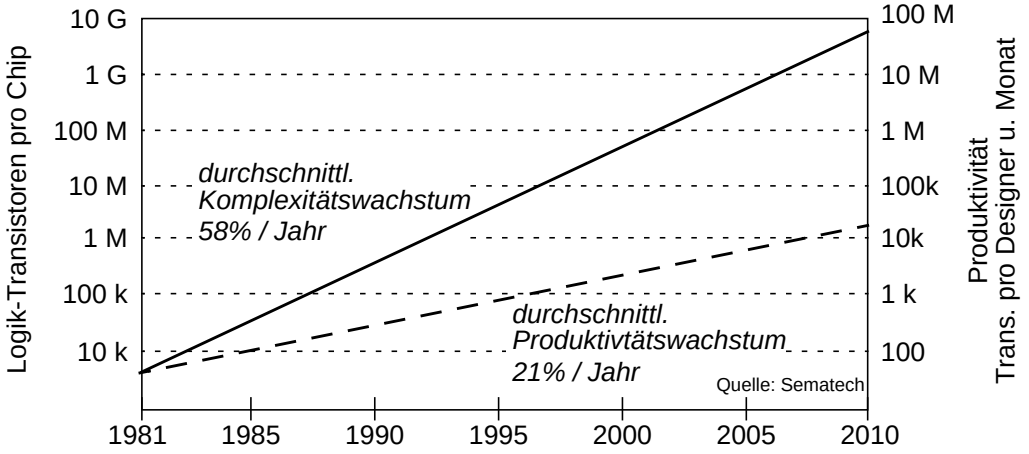


Bild 2-2: Entwurfsproduktivitäts-Lücke

Um den neuen Anforderungen in Bezug auf Komplexität und Entwurfszeit Rechnung zu tragen, werden deshalb intensive Anstrengungen unternommen, die Produktivität beim Entwurf zu erhöhen. Im Zusammenhang mit der Wiederverwertung von Blöcken ist aufgrund der wesentlich höheren Anzahl von Realisierungsalternativen ein strukturiertes Vorgehen notwendig. Das bisher meist auf der Erfahrung der Designer basierende ad-hoc Vorgehen zur Festlegung der Systemarchitektur ist nicht mehr anwendbar, da der Entwurfsraum wesentlich größer ist. Deshalb wird ein möglichst durchgängiger Ablauf für den

Entwurf benötigt, der ausgehend von der Spezifikation der gesamten Funktion bis hin zu deren Implementierung in jeder Phase die Verifikation zuläßt, um die Umsetzung schnell und möglichst ohne zusätzliche Re-Design Schleifen durchführen zu können. Ziel ist es, Entwurfsmethodiken zu finden, die auf festgelegten Regeln basieren und die Korrektheit des Ergebnisses entweder durch Konstruktion oder durch mächtige Synthese- und Verifikationswerkzeuge garantieren ([13]).

2.1.2 Etablierter Ablauf für den Entwurf integrierter Schaltungen

Betrachtet man die Trends bei der Vorgehensweise zur Entwicklung von integrierten Schaltungen in der Vergangenheit, so ist festzustellen, daß sich mit zunehmender Komplexität der Bausteine der Ansatzpunkt beim Chipentwurf zunehmend auf höhere Abstraktionsebenen verschoben hat. Der Grund hierfür liegt darin, daß die Menge an Information, die einen konkreten Baustein auf den unterschiedlichen Entwurfsebenen beschreibt - im Bild 2-3 kurz als Komplexität bezeichnet - mit zunehmender Abstraktion auf den höheren Ebenen abnimmt. Auf diese Weise blieb der Entwurf zunehmend größerer Bausteine für den Entwickler handhabbar.

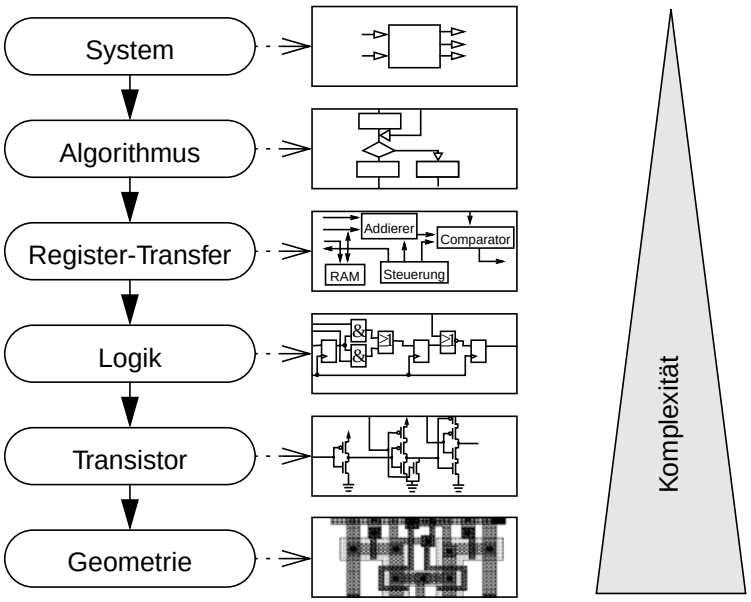


Bild 2-3: Ebenen des IC-Entwurfs

Dieses vereinfachte Modell demonstriert, daß größere Chipkomplexitäten nur dadurch beim Entwurf gemeistert werden konnten, daß der Einstiegspunkt auf immer höhere Abstraktionsebenen verschoben wurde. Dieser Trend setzt jedoch voraus, daß mit Hilfe geeigneter Werkzeuge die Übergänge zwischen den darunterliegenden Ebenen durch entsprechende Synthese- und Verifikationsmethoden automatisiert oder zumindest teilau-

tomatisiert durchgeführt werden konnten. Heute sind derartige Tools von der Industrie verfügbar, die den Entwurf auf der Register-Transferebene (RTL) und auch auf der algorithmischen Ebene unterstützen.
Der in der industriellen Praxis etablierte Ablauf beim Chipdesign beginnt heute mit der Modellierung von Schaltungen auf der Register Transfer Ebene (RTL) mittels einer Hardware Beschreibungssprache (HDL) wie z.B. VHDL oder Verilog. Anhand des HDL-Modells kann die zu entwerfende Funktion in ihrer Funktion verifiziert werden. Nach Simulation und Verifikation der Funktion auf RTL Ebene erfolgt die Synthese und Optimierung mit Hilfe von Synthesewerkzeugen, die die RTL Schaltung auf die Gatterebene abbilden und dabei mit einer Logik-Bibliothek eines Halbleiterherstellers verknüpfen. In einer derartigen Technologiebibliothek sind Modelle von Logikgattern enthalten, die mit ihren Eigenschaften bezüglich Flächenbedarf, Zeitverhalten und Verlustleistung abstrakt charakterisiert sind. Auf diese Weise ist es möglich, durch Synthesewerkzeuge die Funktion auf eine Gatterschaltung abzubilden und mit entsprechenden Tools zu simulieren bzw. zu analysieren, ohne daß Details über die dahinterliegende Transistorschaltung bekannt sein müssen, und ohne daß eine Simulation auf der tieferen Ebene notwendig ist. (Diese Abstraktion erlaubt es damit, größere Schaltungen zu beherrschen als dies auf Transistorebene möglich wäre.)
Das beschriebene Vorgehen beruht jedoch darauf, daß alle auf der Logikebene relevanten Parameter im Modell korrekt erfaßt sind. Als Ergebnis dieses Schrittes resultiert eine Gatternetzliste, die mit Hilfe der Synthesewerkzeuge nach bestimmten Kriterien wie Fläche, Laufzeit des kritischen Pfades oder auch Verlustleistung optimiert werden kann. Bei der Synthese werden neben den Eigenschaften der einzelnen Gatter durch Einbeziehung gewisser Modelle auch Schätzungen der Laufzeiten auf den Verbindungsleitungen berücksichtigt. In der zugehörigen Gattersimulation können damit sowohl die Funktion als auch eine Schätzung das Zeitverhaltens der Schaltung überprüft werden.
Im Plazierungs- und Verdrahtungsschritt (Layout) werden aus der Gatternetzliste die für die Produktion benötigten Daten generiert. Mit diesem Schritt sind die Lage der einzelnen Gatter auf dem Substrat festgelegt und damit auch die Verbindungsleitungen. Die relevanten Parameter für die Verbindungsleitungen können daraus extrahiert und in der sog. Back-Annotation in die Gatter-Netzliste miteinbezogen werden, wodurch ein realistischeres Modell generiert werden kann. Eine Gattersimulation unter Berücksichtigung des Zeitverhaltens auch auf den Verbindungsleitungen erlaubt es, das Verhalten auf dem tatsächlich produzierten Chip sehr realistisch zu erfassen. Bild 2-4 zeigt die einzelnen Schritte des HDL-basierten Entwurfs und die zugehörigen Simulationsläufe.
Aufgrund der wesentlich längeren Dauer von Simulationen auf der Gatterebene sind diese nur in eingeschränktem Umfang möglich. Eine vollständige Simulation ist in den meisten Fällen ausgeschlossen, und es ist nur eine teilweise Simulation der Betriebsfälle möglich. Deshalb werden weitere Verifikationsmethoden angewendet. Zum einen können durch statische Timing-Analyse die maximale Betriebsfrequenz der Schaltung und die dabei begrenzenden kritischen Pfade untersucht werden. Andererseits ist durch formale Verifikation eine Überprüfung auf funktionale Identität von Modellen auf unterschiedlichen Abstraktionsebenen möglich.
Bei der Abbildung auf die Gatterebene und die Optimierung müssen neben der Verfügbar-

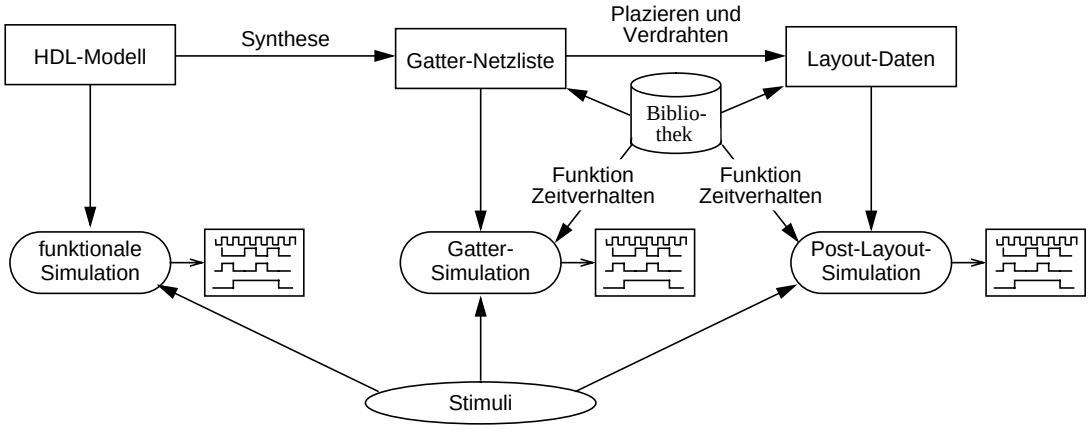


Bild 2-4: Übersicht zum HW Design-Ablauf ([117])

keit einer korrekt charakterisierten Bibliothek vom Synthesewerkzeug gewisse Annahmen in Bezug auf die tatsächlich auf dem Chip entstehenden Verzögerungen aufgrund der Laufzeiten zwischen den Gattern getroffen werden. Für die zunehmend kleineren Strukturgrößen werden diese Modelle zunehmend ungenauer, da die Leitungslaufzeiten im Vergleich zu den Verzögerungen in den einzelnen Gattern immer mehr dominieren. Deshalb ist es für Designs mit Technologien unterhalb 0,25 µm zunehmend erforderlich, daß in den Syntheseschritt Informationen über die Plazierung der einzelnen Schaltungsteile miteinbezogen werden, um ausreichend zuverlässige Optimierungsergebnisse zu erhalten. Ebenso wird durch kapazitive Kopplung zwischen Leiterbahnen die Signalintegrität zu einem Problem. Derartige Trends führen dazu, daß mit der weiteren Strukturverkleinerung auch in Bezug auf die tieferen Entwurfsebenen große Herausforderungen für neue Methoden und Werkzeuge gemeistert werden müssen. In der vorliegenden Arbeit sollen derartige Aspekte jedoch nicht näher betrachtet werden. Hier stehen vielmehr die höheren Abstraktionsebenen im Vordergrund.

Neben der etablierten Vorgehensweise, Hardwaremodelle ausgehend von Registertransferebene durch Logiksynthese in eine Gatternetzliste umzusetzen, sind jedoch auch bereits Ansätze und zugehörige Werkzeuge vorhanden, die auf der nächsthöheren Abstraktionsebene ansetzen. Zum einen ist hier die sog. High-Level Synthese zu nennen, die aus algorithmischen Beschreibungen Gatter-Netzlisten erzeugen kann. Der Behavioral Compiler von Synopsys ([103]) ist hierfür ein Beispiel, der aus einem in einer Hardwarebeschreibungssprache (HDL) codierten Verhaltensmodell den Datenpfad und die zugehörige Steuerung generieren kann. Ein ähnliches Werkzeug ist der Protocol Compiler von Synopsys, der speziell auf den Bereich der Bearbeitung von Datenverarbeitungsprotokollen ausgerichtet ist. Andere Werkzeuge wie CoWare N2C ([20]) ermöglichen die Synthese von Schnittstellen zwischen unterschiedlichen Hardware-Blöcken der Architektur und auch die Erzeugung der zugehörigen SW Treiber.

Mit Tools wie COSSAP (heute Cocentric, [105]) von Synopsys oder Incisive-SPW von Cadence ([12]) können Systeme aus bestimmten Anwendungsgebieten, bei denen der Schwerpunkt im Bereich der Signalverarbeitung liegt, auf der algorithmischen Ebene mo-

delliert und bezüglich des Einflusses unterschiedlicher Parameter auf die resultierenden Systemeigenschaften untersucht werden. Dabei kommen Bibliotheken von abstrakten Funktionsblöcken zum Einsatz, aus denen komplette Systeme aufgebaut und simuliert werden können. Diese Werkzeugen erlauben auch die Generierung von Programmcode für DSPs oder auch von synthetisierbaren HDL Beschreibungen. Derartige Code-Generatoren sind auch in anderen Werkzeugen zur abstrakten Untersuchung von Algorithmen wie z.B. MATLAB ([80]) oder von Zustandsautomaten für Steuerungen wie z.B. Statemate ([52]) enthalten.
Diese Ansätze sind jedoch im Gegensatz zur oben beschriebenen RTL-Logiksynthese meist sehr stark auf bestimmte Anwendungsgebiete fokussiert und liefern entsprechende Teillösungen bzw. Lösungen für spezifische Probleme beim IC Design. Sie setzten zwar auf höheren Entwurfsebenen an als die fest etablierten Werkzeuge des RTL-basierten HW-Entwurfs. Sie erlauben aber keine Erfassung aller Aspekte eines komplexen SoC, das unterschiedliche vorgefertigte Module integriert und insbesondere auch eine gemischte Implementierung der Funktion in HW und SW erlaubt.
Deshalb ist ein weiterer Schritt in Richtung höherer Abstraktion beim Entwurf erforderlich. Hierfür muß der heute vorherrschende Designstil zu einem strukturierten Vorgehen erweitert werden, bei dem ausgehend von einer Spezifikation des kompletten Systems auf abstrakter Ebene die Freiheitsgrade schrittweise reduziert werden, bis die etablierten Methoden der Umsetzung von Funktionalitäten in HW und SW angewendet werden können. Dieser Aspekt soll im weiteren anhand des Begriffs des System-Level Entwurfs näher beleuchtet werden.

2.1.3 System-Level Entwurf

2.1.3.1 Überblick

Unter dem Begriff System-Level Entwurf versteht man ein Vorgehen, bei dem ausgehend von der Spezifikation ein Modell des Systems generiert wird, das alle Eigenschaften des Systems vollständig erfaßt. Dieses Modell ist auf einem im Vergleich zu den oben genannten Methoden und Werkzeugen höheren Abstraktionsniveau angesiedelt und beinhaltet alle möglichen Implementierungsvarianten des Systems. Durch schrittweise Verfeinerungen wird der Entwurfsraum eingeengt und die Anzahl der möglichen Lösungen reduziert, um schließlich die gewünschte Lösung unter Einhaltung der Vorgaben zu erreichen. Ist nach dieser systematischen Verfeinerung die Architektur des Systems festgelegt, insbesondere die Aufteilung in die in HW bzw. SW zu realisierenden Teile, schließt sich daran der gemeinsame Entwurf an.
Die im Zuge der Verfeinerung durchgeführte Abwägung unterschiedlicher Varianten auf den unterschiedlichen Abstraktionsebenen wird als Exploration bezeichnet. Durch die dabei getroffenen Festlegungen wird die Abstraktionsebene jeweils reduziert und die Genauigkeit erhöht. Die Übergänge auf die tiefere Abstraktionsebene sind jeweils mit einer Transformation des Modells verbunden, die den Detaillierungsgrad erhöht. Ziel ist es, die Richtigkeit des dabei generierten Modells nach dem Prinzip "correct-by-construction" ([28], [93]) zu gewährleisten. Mit der Reduktion des Abstraktionsniveaus geht jedoch ein erhöhter Aufwand in Bezug auf das Datenvolumen des Modells oder die erforderliche Si-

mulationszeit einher. Aus diesem Grund wird angestrebt, die tatsächliche Implementierung auf den niedrigeren Abstraktionsebenen durch automatisierte Synthese zu erzeugen, während eine intensive Validierung auf den höheren Abstraktionsebenen mittels Simulation oder formale Verifikation möglich ist.
In dem aus [77] bzw. [125] entnommenen Bild 2-5 ist dieser Zusammenhang als Pyramide dargestellt, deren Grundlinie die Menge aller möglichen Realisierungen des Systems darstellt. Durch die sukzessive Verfeinerung der Modelle und der damit verbundenen Reduktion der Abstraktionsebene wird der von dem jeweiligen Punkt noch erreichbare Umfang der Realisierungen eingeschränkt, bis auf den untersten Ebenen die tatsächliche Implementierung erreicht ist.

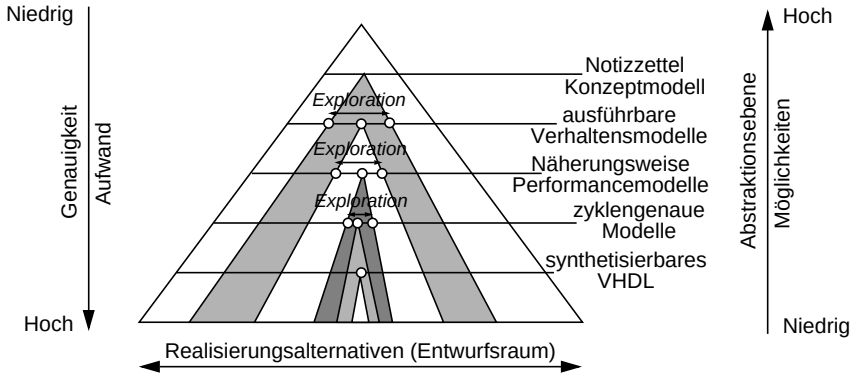


Bild 2-5: Die Abstraktions-Pyramide (nach [125]*)*

Dieses Vorgehen ist aus dem Entwurf von eingebetteten Systemen, die ähnliche Problemlösungen benötigen wie System-on-Chip Bausteine, bekannt ([28]).
Beim strukturierten System-Level Entwurfsvorgehen wird Schritt für Schritt das Abstraktionsniveau reduziert, während das bis jetzt meist verbreitete ad-hoc Vorgehen durch direkten Übergang von der Konzeptionsebene auf ein zyklengenaues oder synthetisierbares Modell gekennzeichnet ist, ohne die Möglichkeiten des Entwurfsraums durch Explorationsschritte auf den dazwischenliegenden Abstraktionsebenen zu untersuchen. Dabei werden Festlegungen für die Implementierung statt basierend auf einer systematischen Untersuchung von Lösungsalternativen auf den einzelnen Abstraktionsebenen häufig aufgrund der Erfahrung oder Einschätzung von Systemarchitekten oder Designern getroffen, was zu suboptimalen Lösungen führen kann.
Neben der strukturierten Vorgehensweise zur schrittweisen Verfeinerung und Reduktion des Abstraktionsniveaus stellt die Wiederverwertung von vorgefertigten Funktionsblöcken ein weiteres Hauptprinzip des System-Level Design dar. Dieses bereits beim reinen ASIC Entwurf angestrebte Vorgehen ist beim SoC Entwurf jedoch in wesentlich breiterem Kontext von Bedeutung ([55]). Derartige, auch als IP Module (Intellectual Property), virtuelle Komponenten oder kurz als "Cores" bezeichnete Blöcke stellen die Basis für den erfolgreichen Entwurf großer System-on-Chip Bausteine dar. IP Module dienen zur Implementierung bestimmter Teilfunktionen des Systems und beinhalten spezifisches Know-how des

jeweiligen Anbieters. IP Module können als Analogon zu den beim Platinen-Entwurf verwendeten Standardbauelementen betrachtet werden. Grob lassen sich eingebettete Prozessoren wie z.B. ARM, MIPS oder PowerPC Cores, vom Nutzer konfigurierbare Prozessoren wie z.B. Xtensa von Tensilica ([109]) oder ARCtangent von ARC ([1]), sowie feste Hardware Beschleuniger Blöcke unterscheiden. Durch Einsatz entsprechender Blöcke und einer geeigneten Kommunikationsarchitektur lassen sich auf diese Art und Weise gemischte HW-/SW-Lösungen aufbauen, die in ihren Eigenschaften den Anforderungen hinsichtlich Performance und Flexibilität angepaßt sind.
IP Module können in unterschiedlicher Form verfügbar sein: Sog. "Soft Cores" werden in Form von synthetisierbaren RTL Code geliefert, wobei der Nutzer für alle weiteren Realisierungsschritte zuständig ist. "Firm Cores" werden als Netzlisten zur Verfügung gestellt, die der Anwender in das eigene Design integriert und von ihm nicht mehr modifiziert werden können. "Hard Cores" wie z.B. Prozessoren oder auch analoge Komponenten werden als Black Box bereitgestellt, bei der der Nutzer keinen Zugriff auf die interne Struktur oder die Implementierung der Funktion erhält, und die auch nicht änderbar ist. Im Zusammenhang mit IP Modulen ist auch der Schutz des geistigen Eigentums von besonderem Interesse, weshalb dieser Aspekt auch bei der Standardisierung Beachtung findet.
Neben der Verfügbarkeit entsprechender Module hängt die Integrierbarkeit in konkrete Designprojekte insbesondere von der Kompatibilität mit der Umgebung ab, in der sie eingesetzt werden. Deshalb wurden intensive Bemühungen zur Definition geeigneter Standards unternommen, die einen breiten Einsatz derartiger Module ermöglichen sollen. Dies betrifft neben der physikalischen Integration in den SoC Baustein auch die Unterstützung der eingesetzten Entwurfswerkzeuge und der verwendeten übrigen Software-Umgebung. Als treibende Faktoren für die Definition von Standards gelten die Notwendigkeit gemeinsamer Kommunikationsprinzipien, gemeinsame Designformate sowie die Bewertung und Sicherstellung der Qualität von Entwürfen ([75]).
Hier sind insbesondere die Standardisierungsbemühungen bei der VSIA (Virtual Socket Interface Alliance) zu nennen, die die Austauschbarkeit von IP Modulen zum Ziel haben. Zu diesem Zweck wurden von der VSIA bereits u.a. der System-Level Interface (SLIF) Standard, der die Spezifikation der Kommunikation von IP Modulen auf unterschiedlichen Abstraktionsebenen und die Trennung des Kommunikations- und Verarbeitungsaspekts beschreibt, der Virtual Component Interface Standard (OCB), der allgemein die Schnittstellen von Cores und deren Verbindung über On-Chip Busse spezifiziert, sowie der Virtual Component Attributes (VCA) Standard, der die Charakterisierung von IP Modulen zum Inhalt hat, festgelegt.
Daneben gibt es weitere Aktivitäten aus dem Bereich der Industrie und von Universitäten, die ähnliche Ziele verfolgen aber auch den konkreten Austausch bzw. die Vermarktung von virtuellen Komponenten betreiben.

2.1.3.2 Begriffe

Im Zusammenhang mit dem System-Level Entwurf werden vielfach Begriffe für Vorgehensweisen oder Prinzipien verwendet, die gewisse Teilaspekte beschreiben und die hier zunächst erläutert werden sollen. Teilweise handelt es sich um Konzepte, die bereits disku-

tiert wurden, bevor der Entwurf komplexer SoCs als Forschungsthema aktuell wurde.
Der Begriff des **HW/SW Co-Design** ist ein Beispiel dafür. Im Zusammenhang mit eingebetteten Systemen, die neben einem Prozessor auch spezifische Hardware besitzen, war die Zielsetzung dieser Vorgehensweise, derartige Systeme in einem gemeinsamen Entwicklungsprozeß zu entwerfen und dabei durch möglichst optimale Abstimmung der entsprechenden Komponenten untereinander eine optimierte Lösung zu generieren. Teilaspekte sind zum einen die Unterteilung der zu implementierenden Funktionalität in HW und SW als auch der eng verzahnte Entwurf und die Simulation der in HW bzw. SW implementierten Teile eines Systems. Die im Zusammenhang mit dem HW/SW Co-Design behandelten Probleme sind ein Teil des System-Level Design Ansatzes, der aufgrund der Berücksichtigung von wiederverwertbaren Blöcken und allgemeiner betrachteten Architekturen noch weiter gefaßt ist.
Unter **Plattform-basiertem Design** werden unterschiedliche Vorgehensweisen verstanden. Zum einen wird eine Plattform als eine Basisarchitektur betrachtet, die als Grundlage für unterschiedliche Varianten oder auch unterschiedliche Generationen von Systemen dient. Durch geeignete Anpassung der Plattform, z.B. durch Erweiterung oder Austausch einzelner HW- oder auch SW-Teile des Systems können sehr schnell und auf relativ einfache Art und Weise spezifische Lösungen generiert werden. Grundlage hierfür ist eine Plattform, die neben der Grundarchitektur auch die zugehörigen Werzeuge zur SW-Entwicklung für den Anwender der Plattform beinhaltet. Plattform-basiertes Design wird oft im Sinne der Wiederverwertung einer entsprechenden Plattform-Architektur verstanden. Diese Definition entspricht weniger einem strukturierten Konzept zur Definition und zum Entwurf derartiger Grundarchitekturen, die auf die entsprechende unterschiedliche Nutzungsmöglichkeit ausgerichtet sein müssen. Hier steht vielmehr die Zielsetzung im Vordergrund, eine derartige Plattform zur effizienten Bereitstellung neuer sowie erweiterter oder an spezifische Bedürfnisse angepaßter Lösungen zu nutzen.
Andererseits werden jedoch auch grundlegende Vorgehensweisen und Randbedingungen für die Generierung derartiger Plattformen als Basis für die Wiederverwertung beschrieben ([15]). In [13] wird der Ansatz des Plattform-basierten Designs verallgemeinert. Eine Plattform wird dabei als Abstraktionsebene betrachtet, die Details aus den darunterliegenden Ebenen und den damit verbundenen konkreten Realisierungsmöglichkeiten verbirgt. Plattform-basiertes Design wird folglich als strukturierte Methode definiert, bei der im Sinne eines top-down Vorgehens unter Berücksichtigung der spezifizierten Anforderungen schrittweise von einer Plattform jeweils auf die darunterliegende nächstniedrige Plattform übergegangen wird. Andererseits wird ähnlich einem bottom-up Vorgehen eine Bibliothek mit Verarbeitungs- und Kommunikationskomponenten aufgebaut, die mit Performance-Informationen charakterisiert und zur Implementierung verwendet werden können. Diese Bibliothek definiert dann den Bereich, innerhalb dessen die Exploration auf der jeweiligen Abstraktionsebene stattfinden kann. Durch Kombination der beiden Ansätze im Sinne eines "meet-in-the-middle" ([13]) ist auf den jeweiligen Abstraktionsebenen eine durchgängig konsistente Modellierung möglich.
Dabei müssen von den oberen Abstraktionsebenen jeweils die Vorgaben nach unten durchgereicht werden, während performance-relevante oder andere für die Optimierung wichtige Daten in die abstrakteren Modell miteinbezogen werden müssen. Ziel ist es mit dieser Vor-

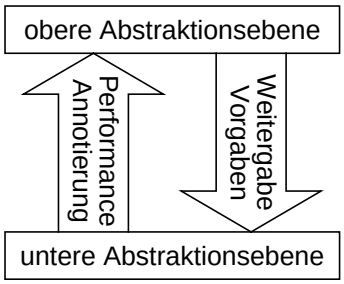


Bild 2-6: Interaktion der Abstraktionsebenen

gehesweise, große Schleifen über Abstraktionsebenen hinweg zu vermeiden. Wie die Exploration auf den jeweiligen Ebenen durchgeführt wird, ist jedoch damit nicht festgelegt. Randbedingungen sind dabei jedoch zum einen die Vorhersagbarkeit, die die Optimierung auf abstrakteren Ebenen erlaubt, und zum anderen die Verifizierbarkeit, die auf allen Ebenen die Überprüfung auf Korrektheit ermöglicht. Die im Zusammenhang mit dieser Definition des Plattform-basierten Designs beschriebenen Ansätze sind konsistent mit den oben erläuterten Prinzipien des System-Level Designs und konkretisieren diese.
Beim Entwurf von SoCs wird auch oft von **Interface- oder Kommunikations-basiertem Entwurf** ([66], [92], [97]) gesprochen. Damit soll ausgedrückt werden, daß dem Aspekt der Schnittstellen zwischen einzelnen Funktionsblöcken des Systems besondere Beachtung geschenkt werden muß. Als Grundannahme für Interface-basiertes Design wird von einer "Orthogonalisierung" der Aspekte Verarbeitung und Kommunikation ([66]) innerhalb eines Systems ausgegangen. Konkret bedeutet dies, daß beide Aspekte unabhängig von einander im Designprozeß behandelt werden müssen, um die gesamte Komplexität des Designablaufs besser beherrschbar zu machen. Diese Trennung stellt auch die Voraussetzung für die Wiederverwendung von vorgefertigten Modulen in SoCs dar. Bei diesem Vorgehen wird davon ausgegangen, daß beide Domänen jeweils auf unterschiedlichen Abstraktionsebenen erfaßt werden können und durch die getrennte Behandlung eine bessere Exploration des Design-Raumes möglich ist.

2.1.3.3 Prinzipieller Ablauf

Der Ansatz des System-Level Entwurfs basiert auf den oben beschriebenen Prinzipien der Wiederverwertung von Blöcken sowie einer strukturierten Vorgehensweise zur kontinuierlichen Reduktion des Abstraktionsniveaus. Der prinzipielle Ablauf, der - abgesehen von spezifischen Details - den in der Literatur veröffentlichten Ansätzen (z.B. [34], [14]) zugrundeliegt, soll im folgenden dargestellt werden. Es handelt sich um kein reines top-down Vorgehen, da durch Wiederverwertung und Schätzungen von Performancedaten auch Informationen aus niederen Abstraktionsebenen berücksichtigt werden. Das verwendete Hauptprinzip liegt jedoch in der sukzessiven Eingrenzung des Design-Raums bis zur Festlegung der Realisierung. Folgende Phasen können dabei unterschieden werden ([14]).

- System-Spezifikation
- Architektur-Exploration
- Architektur-Implementierung

Bild 2-7 zeigt den prinzipiellen Ablauf (nach [34] und [14]), dessen einzelnen Schritte nachfolgend erläutert werden.

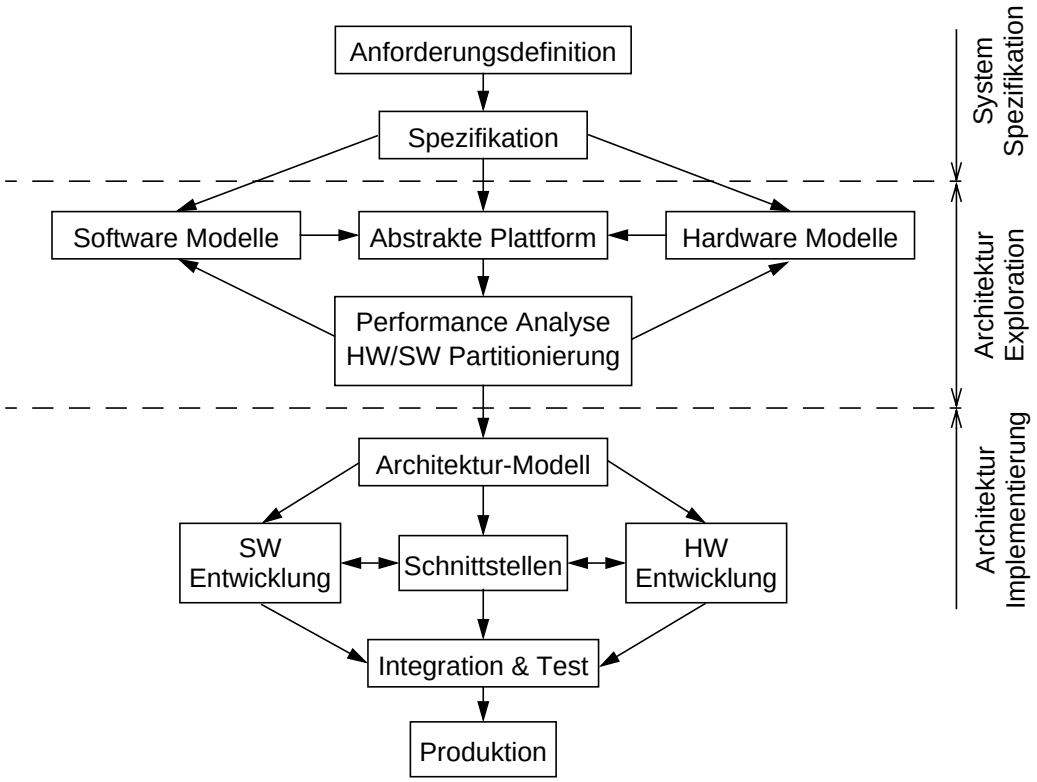


Bild 2-7: System-Level Entwurf

Dieses Schema entspricht auch der in [93] getroffenen Unterteilung der Design Strategie in eine funktionale, eine Abbildungs- und eine architekturale Ebene. Dahinter liegt das Prinzip der Trennung von Funktion und Architektur eines Systems, wobei der für die Realisierung notwendige Schritt des Übergangs zwischen diesen Bereichen mit der Abbildung der Funktion auf die Architektur im Rahmen der Exploration erfolgt. Dies entspricht den im Zusammenhang mit Bild 2-5 erläuterten Transformationen zwischen den Abstraktionsebenen, die dort noch feiner unterteilt sind.
Nachfolgend sollen die einzelnen in Bild 2-7 gezeigten Phasen erläutert werden.

System-Spezifikation

Ausgangspunkt ist eine meist in textueller Form vorliegende Anforderungsdefinition, die als Basis für die Erstellung eines geeigneten Modells dient, das die Funktionalität des Sy-

stems vollständig erfaßt. Dieses Modell wird insbesondere für die funktionale Verifikation und die Überprüfung auf Einhaltung der Anforderungen auf dieser Ebene verwendet. Im Rahmen einer Exploration auf dieser Ebene können auch alternative Algorithmen und deren Eignung untersucht werden. Darüber hinaus wird das funktionale Modell oft auch als Referenz verwendet, mit der die Verfeinerungen auf tieferen Abstraktionsebenen verglichen und auf diese Weise auf Konformität mit der ursprünglichen Spezifikation überprüft werden können. Dies kann entweder durch Nutzung des funktionalen Modells in einer gemeinsamen Simulationsumgebung oder durch Bereitstellung geeigneter Test-Muster erfolgen. Daneben kann dieses abstrakte Modell auch als abstrakter Prototyp dienen, mit dem es für den Auftraggeber möglich ist, das zu entwerfende System in seinen Eigenschaften zu untersuchen und ggf. in der geplanten Umgebung zu testen. Um funktionale Modelle in der beschriebenen Weise einsetzen zu können, ist es erforderlich, daß sie nicht nur die benötigte Funktionalität erfassen sondern auch simuliert sowie automatisiert ausgewertet und visualisiert werden können. Derartige Modelle werden auch als ausführbare Spezifikation bezeichnet.

Die in [56] getroffene Unterscheidung zwischen anwendungsorientierten und implemeniterungsorientierten Systemmodellen konkretisiert diese Darstellung. Ein anwendungsspezifisches Modell zielt auf Untersuchungen ausschließlich der System-Funktionalität ab und soll durch exakte und vollständige Beschreibung der Funktionalität gewährleisten, daß tatsächlich das gewünschte System erfaßt ist, d.h. daß das richtige System realisiert wird. Demgegenüber beschreiben implementierungsorientierte Modelle die Art und Weise, wie ein System aufgebaut ist, und bilden die Basis für die spätere Implementierungsphase. Diese beiden Modelltypen werden in [56] auch als Funktions-basiert (FBM) bzw. als Objektbasiert (OBM) bezeichnet. Objektbasierte Modelle können sowohl Eigenschaften eines Architekturmodells beschreiben (Ressourcen als Objekte), aber auch ein bereits verfeinertes funktionales Modell (Prozesse als Objekte) darstellen. Beide Modelltypen werden als notwendig erachtet, wobei der Übergang vom FBM zum OBM als der entscheidende Schritt betrachtet wird, der nicht automatisierbar ist, da eine prinzipielle Diskontinuität zwischen beiden Modellen besteht. An dieser Stelle soll darauf hingewiesen werden, daß das objektbasierte Modell im Sinne von [56] zu einem höheren Abstraktionsniveau gehört als das in Bild 2-7 gezeigte Architekturmodell, das die Architektur bereits komplett beschreibt. Informationen zu Objekten des Designs sind vielmehr je nach verwendeter Modellierungsart bereits in der abstrakten Plattform bzw. zum Teil in der (ausführbaren) Spezifikation enthalten, da insbesondere in der Explorationsphase intensiv auf implementierungsrelevante Informationen gemäß des beschriebenen bottom-up Ansatzes Bezug genommen werden muß, um die geeignete Architekturwahl treffen zu können.

Der Erstellung eines Systemmodells liegen immer gewisse Annahmen zugrunde, wie die einzelnen Teile des Modells abgearbeitet werden (einschließlich deren Aktivierung und der Erfassung des zeitlichen Verhaltens), wie die gleichzeitige Ausführung parallel laufender Modell-Teile bewerkstelligt wird, und wie die Kommunikation zwischen den einzelnen Teilen des Modells erfolgt. Für die Beschreibung dieses Zusammenhangs wird der Begriff "Model of Computation" (MoC) verwendet. Mit Hilfe eines derartigen Modells kann die Funktionalität eines Systems eindeutig erfaßt werden. Es dient der Verifikation gegenüber den Vorgaben und erlaubt die automatische Synthese von Teilen der Spezifikation auf die

Ressourcen einer Architektur oder die manuelle Verfeinerung und deren Überprüfung ([96]). Übersichten über die für die Modellierung von Systemen auf abstrakter Ebene bisher meist verwendeten MoCs wie Zustandsautomaten, Discrete-Event Modellen, Prozeß-Netzwerken und Kontroll- bzw. Datenflußgraphen sind in [23] oder [96] enthalten, während [28] eine ausführliche, mehr formale Darstellung zum Thema MoC beinhaltet.
Die Auswahl eines geeigneten Model of Computation für die Erstellung des abstrakten Systemmodells ist zudem abhängig vom Anwendungsgebiet und von den Eigenschaften der zu modellierenden Funktion. Nicht alle Modelle sind auf der obersten Abstraktionsebene relevant, sondern kommen ggf. im Laufe der Verfeinerungsschritte zum Einsatz. Die Verwendung eines bestimmten MoC zur Modellierung des Systems stellt bereits eine Einengung des Lösungsraums dar. Deshalb ist die Wahl eines geeigneten MoC für das betrachtete System eine entscheidende Voraussetzung.
In Kapitel 4.2.1 ist eine Übersicht der Modellierungsansätze mit Graphen enthalten, die für die in dieser Arbeit entwickelte Methode relevant sind. Hier soll deshalb nur kurz auf sprachorientierte Ansätze zur Modellierung auf der Systemebene eingegangen werden. Die Zielsetzung ist dabei vielfach, eine Durchgängigkeit bei der Modellierung auf unterschiedlichen Abstraktionsebenen zu erreichen und damit unnötige und fehlerträchtige (manuelle) Umsetzungen auf andere Modelle bzw. Sprachen zu vermeiden.
Da ein erheblicher Teil eines System on Chip aus SW besteht, die auf einem eingebetteten Prozessor läuft, sind C oder C++ als etablierte Programmiersprachen, die die Erstellung von kompakten und abstrakten Systembeschreibungen erlauben, auch Kandidaten für die Modellierung des kompletten Systems. Da jedoch zur Systemmodellierung insbesondere die Nachbildung von Eigenschaften wie Parallelität, Reaktivität und Zeitverhalten erforderlich sind, sind für den Einsatz dieser Sprachen Ergänzungen erforderlich.
SystemC ([108]) als eine auf C++ basierende Sprache stellt hierfür eine entsprechende Klassenbibliothek und einen Simulationskernel zur Verfügung. Mit SystemC ist die Modellierung von Systemen auf unterschiedlichen Abstraktionsebenen möglich. Die Spannbreite reicht von rein funktionalen Modellen ohne bzw. mit Zeitinformation bis hin zur Register Transferebene, auf der entweder buszyklus- oder taktgenau modelliert werden kann. Durch Unterstützung von unterschiedlichen Kommunikationsmodellen, die vom Benutzer im Zuge der Verfeinerung auch erweiterbar sind, ist eine flexible Modellierung von Systemen möglich.
Zur Erstellung von SystemC Modellen können die üblichen Werkzeugen zur C/C++ Programmierung und zur Anzeige von Signalverläufen eingesetzt werden. Zunehmend werden auch verschiedene kommerzielle Werkzeuge mit Unterstützung von SystemC verfügbar. Neben Simulationstools aus verschiedenen Anwendungsgebieten, die auch SystemC als Eingabe erlauben, sind Designumgebungen zur Unterstützung des System-Level Designs mit SystemC wie CoWare N2C ([20]) und zur Synthese von Gatternetzlisten oder synthetisierbaren HDL Modellen aus SystemC Beschreibungen wie der CoCentric SystemC Compiler von Synopsys ([104]) auf dem Markt.
Ein alternativer Ansatz zur Definition einer Beschreibungssprache für die Systemebene ist SpecC ([100]). SpecC basiert auf ANSI-C und beinhaltet Erweiterungen, die zur Modellierung von Systemen benötigt werden. Dies umfaßt die Unterstützung von Parallelität im System, Hierarchie, Kommunikation und Synchronisation sowie die Erfassung des

Zeitverhaltens. Ähnlich wie SystemC deckt SpecC vier Abstraktionsebenen ab: Spezifikation (funktional ohne Zeitinformation), Architektur (geschätzte Zeitinformation, strukturell), Kommunikations-Modell (mit Zeitinformation, Bus-funktional), Implementierungsmodell (zyklengenau, Register-Transfer Ebene).
Neben diesen auf universellen Einsatz abzielenden Sprachen, die z.T. durch starke unterstützung aus der Industrie an Gewicht gewonnen haben, werden auch Ansätze vorgeschlagen, die ihre Wurzeln in bestimmten Teilgebieten des Systementwurfs haben. Ein Beispiel hierfür ist der Einsatz der aus dem Bereich des SW-Entwurfs bekannten Sprache UML ([87]). UML erlaubt aufgrund seiner Objektorientierung die strukturierte Erfassung von verschiedenen Aspekten des Systems auf unterschiedlichen Abstraktionsebenen einschließlich der einzuhaltenden Vorgaben. Aus dem Bereich der Hardwareentwicklung werden u.a. Erweiterungen von bekannten Hardwarebeschreibungssprachen in Richtung System Level Entwurf wie Superlog ([102]) bzw. SystemVerilog ([107]) propagiert.
Zusätzlich zu diesen neueren Entwicklungen sind einige bereits länger bekannte oder auf ein bestimmtes Anwendungsgebiet zugeschnittene Sprachen bzw. auf die Beschreibung von Zustandsübergängen ausgerichtete Modellierungsanätze wie Esterel ([9]) oder SpecCharts ([84]) im Einsatz. In [98] ist der Einsatz von SDL als Sprache für das System-Level Design beschrieben. SDL wird dort zur System- und zur Implementierungsspezifikation verwendet bevor zur eigentlichen Implementierung die Umsetzung in konventionelle Entwurfssprachen erfolgt.
Der Entwurf von Systemen, bei denen einzelne Teile eingesetzt werden, die mit unterschiedlichen Modellierungsansätzen beschrieben sind, stellt eine besondere Herausforderung dar. Neben der Heterogenität der Beschreibung ist die Verbindung unterschiedlicher MoCs problematisch. Zur Lösung dieser Problematik sind grundsätzlich zwei Ansätze denkbar: Der Einsatz einer universellen Beschreibungssprache, oder die Integration verschiedener Modellierungsansätze in eine gemeinsame Repräsentation, die als Ausgangspunkt für die Analyse und Synthese von Systemen dient.
In [18] ist das SPI (System Property Intervals) Modell und eine zugehörige Workbench vorgeschlagen, das es ermöglicht, gemischt reaktiv/transformative Systeme, die in unterschiedlichen Beschreibungen vorliegen, in ein gemeinsames Modell, dem SPI Graphen, umzusetzen, mit dem die weiteren Entwurfsschritte insbesondere die Cosynthese durchgeführt werden können. Das SPI Modell stellt eine Abstraktion der eigentlichen Funktionalität dar und erfaßt das Zeitverhalten, die Aktivierung und die Kommunikation der einzelnen Komponenten. Die Eigenschaften werden dabei als Wertintervalle repräsentiert, wodurch die Berücksichtigung von Unsicherheiten bei der Abschätzung möglich ist. Auf diese Weise werden unterschiedliche MoCs durch Umsetzung auf eine gemeinsame homogene Systembeschreibung aneinander angepaßt, und eine Optimierung über die von einzelnen Modellen und den zugehörigen Sprachen definierten Grenzen hinweg ist möglich. In einer derartigen Umgebung können außerdem anwendungsspezifische Vorgehensweisen und Werkzeuge weiterhin für die einzelnen Teilsysteme weiterverwendet werden.
Mit diesem Ansatz ist eine gemeinsame Erfassung des Systems und dessen Analyse in einer heterogenen Umgebung möglich. Eine Methode zur Erzeugung einer optimierten Systemarchitektur ist mit dem SPI Ansatz jedoch nicht festgelegt. Aufbauend auf den SPI Graphen ist jedoch die Anwendung von bekannten Syntheseansätzen oder die manuelle Festlegung

der Architektur möglich.

Architektur-Exploration

Nachdem auf der funktionalen Ebene ein abstraktes Modell des zu implementierenden Systems erstellt und verifiziert wurde, ist die Abbildung auf eine geeignete Architektur, auf der die Funktion ausgeführt wird, erforderlich. Das Ergebnis dieses Schritts ist das in Bild 2-7 gezeigte Architekturmodell, das als Ausgangspunkt für die weitere Implementierung dient.

Unter Architektur sollen im folgenden die Ressourcen zur Verarbeitung der Teilaufgaben sowie die zu ihrer Verbindung verfügbaren Kommunikationsressourcen und deren Struktur verstanden werden. Zusätzlich zu dieser, die HW der Architektur definierenden Parameter ist insbesondere auch die Abbildung (Mapping) der einzelnen Teilfunktionen auf die einzelnen Ressourcen Bestandteil der Architekturfestlegung.

Die Aufgabe der Architektur-Exploration besteht darin, unter möglichen Alternativen diejenige Architektur mit dem zugehörigen Mapping auszuwählen, die die gegebenen Randbedingungen am besten einhält. Randbedingungen sind hier Designvorgaben, die in der Anforderungsdefinition festgelegt sind. Beispiele sind die Verarbeitungsdauer, der Flächenbedarf, die Verlustleistung oder einfach auch die Kosten für Produktion einschließlich Lizenzen für IP Blöcke etc.

Man unterscheidet prinzipiell zwischen harten und weichen Randbedingungen beim Entwurf von Systemen. Unter einer harten Vorgabe ist zu verstehen, daß für die gewählte Lösung das jeweilige Kriterium erfüllt sein muß, damit sie gültig ist. Beispiele hierfür sind Verarbeitungsdauern, um zu gewährleisten, daß die notwendige Leistungsfähigkeit auch tatsächlich erreicht wird, oder auch bestimmte maximale Flächen, damit die Größe der Implementierung eingehalten werden kann. Kriterien können auch als weiche Anforderungen gegeben sein, für die es keine scharfe Grenze gibt, die erlaubte von ungültigen Lösungen abgrenzt. Es besteht vielmehr die Zielsetzung, das jeweilige Kriterium unter den gegebenen Randbedingungen möglichst zu optimieren.

Daneben können auch andere Vorgaben zu berücksichtigen sein, wie z.B. der Einsatz bestimmter vorgefertigter IP Module, insbesondere bestimmte Prozessorarchitekturen aufgrund von bereits erbrachten Vorleistungen oder der Erfahrung der Designer. Derartige Randbedingungen können den Lösungsraum bei der Architekturexploration erheblich einschränken. Bei der Optimierung sind oft Ziele anzustreben, die nicht miteinander vergleichbar oder auch gegenläufig sind, wie z.B. Flächenbedarf und Geschwindigkeit oder Kosten und Performance. Im Hinblick auf die vielen Freiheitsgrade bei der Definition einer Systemarchitektur führt dies zu einer weiteren Erhöhung der Komplexität bei der Lösung des Optimierungsproblems ([29]).

Die Architekturexploration setzt sich insgesamt aus den folgenden Teilaufgaben zusammen:

a) Allokation von Ressourcen
b) Unterteilung der Funktion in Teilfunktionen
c) Abbildung der Teilfunktionen auf die Ressourcen der Architektur
d) Erstellung der zeitlichen Verarbeitungsreihenfolge innerhalb der Architektur (Scheduling)

e) Untersuchung der Performance der Architektur
f) Änderungen an Abbildung, Architektur oder Anwendung

Unter Allokation der Ressourcen ist die Festlegung der verwendeten Blöcke als Bestandteil der Architektur zu verstehen. Dies umfaßt sowohl Verarbeitungs- als auch Kommunikationsressourcen sowie deren Verknüpfung. Im Rahmen der Exploration wird zur Erreichung der Optimierungsziele eine Anpassung der Architektur durch Ergänzung, Austausch oder andere Verknüpfung der Ressourcen vorgenommen.
Die Schritte b), c) und d) werden oft als HW/SW-Partitionierung bezeichnet. Dabei wird von einer vorgegebenen Zielarchitektur ausgegangen, auf die die Funktion unter Einhaltung der Randbedingungen optimiert abgebildet werden soll. Auf die Partitionierung wird nachfolgend in Kapitel 2.2 eingegangen.
In Bild 2-8 ist der prinzipielle Ablauf der Exploration anhand des Y-Diagramms nach Kienhuis ([61], nicht zu verwechseln mit dem Y-Diagramm von Gajski) dargestellt. In diesem Schema wird die Untersuchung von programmierbaren Architekturen anhand eines in Y-Form angeordneten Modells dargestellt.

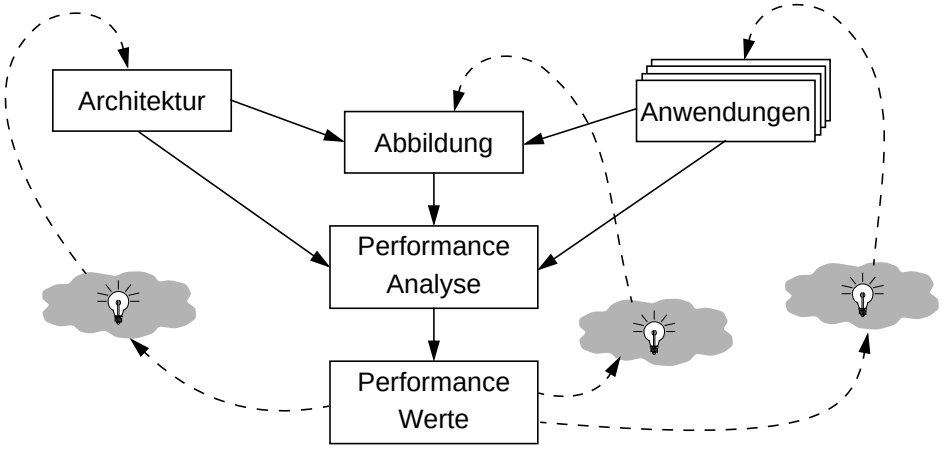


Bild 2-8: Die Untersuchung von programmierbaren Architekturen mit dem Y-Diagramm

Im Bild 2-8 wird angenommen, daß eine Architektur gefunden werden muß, die zur Implementierung verschiedener Anwendungen geeignet ist, was eine verschärfte Randbedingung bei der Findung der Lösung darstellt. Die Vorgehensweise für den einfacheren Fall des Entwurfs eines SoC für eine einzelne Anwendung, der im Rahmen dieser Arbeit betrachtet wird, ist damit jedoch ebenfalls abgedeckt. Das Vorgehen umfaßt die Abbildung eines geeigneten funktionalen Modells der Anwendung auf eine bestimmte Architektur und die Analyse der dabei erreichten Performance, meist anhand einer Simulation. Dabei werden geeignete Parameter extrahiert, die Rückschlüsse auf die Engpässe im System erlauben. Daraus ergeben sich Hinweise für Änderungen, mit denen die Lösung den zu erfüllenden Designkriterien angenähert werden kann.
Die gezeigten Rückkopplungen stellen die entscheidenden Schritte der Exploration dar.

Anhand der extrahierten Performancewerte werden dabei Entscheidungen bezüglich der Änderungen der Abbildung und insbesondere der Architektur getroffen mit dem Ziel, die Lösung derart zu verändern, daß die vorgegebenen Randbedingungen eingehalten werden.

Die im Bild 2-8 enthaltene Rückkopplung zur Anwendung erlaubt zum einen die gleichzeitige Untersuchung mehrerer Anwendungen, ermöglicht jedoch auch die Anpassung des funktionalen Modells, die bei Änderungen an der Architektur ggf. erforderlich ist. Dies entspricht der oben gemachten Feststellung, daß bereits das funktionale Modell objektbasiert gemäß [56] mit spezifischen für die Abbildung auf die Architektur relevanten Eigenschaften sein kann. Die Erstellung eines für diese Untersuchungen geeigneten Modells der Anwendung, ist insbesondere mit einer Unterteilung der gesamten Funktion in einzelne Teile verbunden, was einem Übergang zu einem objektbasierten Modell entspricht.

Nachfolgend werden einige Werkzeuge von Universitäten oder aus der Industrie vorgestellt, die die Architekturexploration unterstützen. Verfahren, die diese Aufgabe automatisieren, sind bisher nicht verfügbar.
Die in Berkeley entwickelte Designumgebung POLIS ([4]) zielt auf eine einheitliche Repräsentation und eine automatische Synthese von gemischten HW/SW-Systemen für kontrollintensive reaktive Realzeitsysteme ab. POLIS basiert auf der Modellierung des Systems in Form eines Netzwerks von speziellen Zustandsautomaten (CFSM, CoDesign Finte State Machine), kann jedoch unterschiedliche beschriebene Spezifikationen einlesen und in ein entsprechende CFSM Modell umsetzen. Daneben beinhaltet POLIS eine Anbindung von Werkeugen zur formalen Verifikation und kann zur Co-Simulation auf unterschiedliche Simulatoren aufsetzen. Weiterhin ermöglicht POLIS die Synthese von VHDL Code oder einer SW Struktur mit Prozeduren für einzelne CFSMs und einem einfachen RTOS. Die Partitionierung des Systems in seine als HW bzw. SW zu implementierenden Teile ist vom Designer jedoch manuell vorzunehmen. Aufgrund der Repräsentation mit Zustandsautomaten ist POLIS in seiner Anwendung als allgemeines Explorationswerkzeug eingeschränkt.
Ptolemy ([23]) ist eine Entwurfsumgebung, die unterschiedliche Models of Computation in einem gemeinsamen Systemmodell integriert und so eine Verifikation und Validierung auf abstrakter Ebene ermöglicht. Durch Integration unterschiedlicher Werkzeuge und Anbindung von Modellen auf verschiedenen Abstraktionsniveaus ist eine übergreifende Simulation eines kompletten Systems möglich. Daneben kann durch entsprechende Code-Generatoren auch SW bzw. HW synthetisiert werden.
Für Ptolemy existieren eine Vielzahl von für bestimmte Anwendungsfelder spezifischen und auf speziellen MoCs beruhenden Modulen, die kombiniert und für den SoC Entwurf verwendet werden können. Da die Umgebung offen ist, können beliebige Ergänzungen vorgenommen werden.

In der gängigen Industriepraxis wurde bisher meist keine umfangreiche Exploration der Architekturalternativen vorgenommen, sondern aufgrund der Erfahrung der Systemarchitekten die Architektur und die Zuordnung der Teilfunktionen festgelegt. Um mit Untersuchungen auf der Architekturebene vor der eigentlichen Implementierungsphase die

erreichbare Performance überprüfen zu können, werden auf dieser Ebene zunehmend Werkzeuge eingesetzt, die eine strukturiertere Vorgehensweise für System-Level Design unterstützen.
Virtual Component Codesign (VCC) von Cadence ([11]) ist eine auf POLIS basierende Designumgebung für den System-Level Entwurf, die die gemeinsame Entwicklung von HW und SW sowie die Integration von IP Modulen ermöglicht. Durch die Abwägung von HW/SW Trade-offs und die Abschätzung der System-Performance können SoC Architektur-Alternativen früh im Entwurfsablauf untersucht werden. VCC erlaubt die Spezifikation des Systems auf unterschiedliche Art und Weise und ermöglicht die Einbeziehung von verschieden modellierten Systemteilen (u.a. C/C++, SDL, Cadence SPW, Matlab, ...). Neben der eigentlichen Architekturexploration werden auch die einzelnen Schritte zur Verfeinerung des Architekturmodells bis hin zum Export für die Entwicklung von HW und SW unterstützt. Bild 2-9 zeigt den Ablauf des Entwurfsprozesses mit VCC.

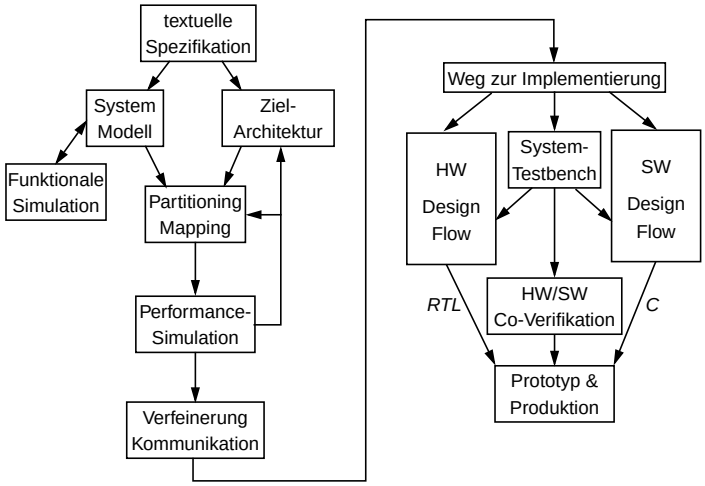


Bild 2-9: Übersicht zum Design Ablauf mit Cadence VCC

VCC basiert auf der Trennung von Funktion und Architektur eines Systems und ermöglicht damit die Untersuchung unterschiedlicher Alternativen und die Festlegung basierend auf den Performance Untersuchungen, die Informationen zur Auslastung von Ressourcen oder anderen Engpässen im System liefern. Die Abbildung der Funktion bei der Exploration ist jedoch nicht automatisiert sondern beruht auf den Entscheidungen des Entwicklers. Weiterhin implementiert es das Prinzip der Trennung von Verarbeitung und Kommunikation und erlaubt damit eine unabhängige Verfeinerung der beiden Teilaspekte. Als Input für die Implementierung des Systems ermöglicht VCC die automatische Generierung von Kommunikationsschnittstellen und auch der Testumgebung für die nachfolgenden Schritte.
Ähnliche Ansätze werden auch von alternativen Werkzeugen unterstützt, mit denen eine abstrakte Architektur und die zugehörigen Kommunikationsmechanismen in einem strukturalen Modell nachgebildet werden können. Mit Hilfe von Simulationen ist es damit möglich, unterschiedliche Varianten auf ihre Leistungsfähigkeit hin zu untersuchen.

Entsprechende Werkzeuge sind z.B. System Architect ([106]) von Summit Design oder MLDesigner ([82]) von ML Design Technologies. Mit Hilfe dieser Werkzeuge, können unter Nutzung von Bibliotheken mit vorgefertigten Modulen und auf unterschiedlichem Detaillierungsgrad Simulationsumgebungen aufgebaut werden, anhand derer auch das dynamische Verhalten des Systems erfaßt werden kann. Aus den Simulationen können Performance-relevante Daten wie die Auslastung von Ressourcen, Werten zu Durchsatz oder Verzögerung extrahiert und visualisiert werden. Diese Information dient dem Entwickler als Input für Änderungen sowohl in Bezug auf die Abbildung der Teilfunktionen als auch hinsichtlich der eingesetzten Ressourcen und deren Parameter. Da die Untersuchungen simulativ durchgeführt werden, können auch reale Stimuli verwendet werden, die eine bessere Anpassung an die tatsächliche Systemumgebung erlauben als vereinfachte Quellenmodelle.
Den beschriebenen Werkzeugen ist gemeinsam, daß sie von einer festen Aufteilung der Funktionalität auf die Ressourcen der Architektur ausgehen. Die Frage der Exploration von unterschiedlichen Alternativen, wird von diesen Werzeugen nicht durchgeführt, sondern sie obliegt dem Entwickler durch manuelle Anpassungen der jeweiligen Architektur. Sollen mit diesen Werkzeugen alternative Architekturen verglichen werden, bei denen auch die Struktur des Modells verändert werden muß, ist die Modifikation am Architekturmodell mit erheblichem Modellierungsaufwand verbunden, wodurch die Anzahl der Alternativen, die untersucht werden können, sehr begrenzt ist. Aus diesem Grund sind derartige Werkzeuge in der frühen Phase der Architekturfindung, in der noch viele Entscheidungen offen sind, weniger geeignet.
Die bisher von der Industrie bereitgestellten Werkzeuge und die damit verbundenen Entwicklungsprozesse können die gestellten Anforderungen nach einer schnellen und flexiblen Untersuchung von Architekturalternativen bisher nicht vollständig erfüllen, da sie meist nur Teilprobleme lösen und keine übergreifende Durchgängigkeit über die einzelnen Schritte hinaus ermöglichen.
Parallel zur vorliegenden Arbeit wird eine Dissertation angefertigt, die im Rahmen der Architekturexploration eine schnell an geänderte Vorgaben in Bezug auf Abbildung und Architektur anpaßbare Performance-Abschätzung von VLSI Architekturen erlaubt. Der Ansatz basiert auf einer funktionalen Modellierung des Systems unter Einbeziehung von Architekturinformationen auf der Basis von SystemC. Ergebnisse zu dieser Arbeit sind in [89] veröffentlicht.

Im Gegensatz zu den beschriebenen Werkzeugen wird bei der Architektursynthese nicht von einer unveränderlichen Architektur ausgegangen, hier wird vielmehr die Architektur, die Zuordnung der Teilfunktionen auf die Ressourcen und die Abarbeitungsreihenfolge gemeinsam konstruiert. Es ergibt sich damit keine rekursive Anpassung der Architektur, wie sie in Bild 2-8 gezeigt ist.
Zur Architektursynthese sind bisher keine kommerziell verfügbaren Werzeuge verfügbar. Jedoch wurden bereits verschiedene Ansätze aus dem Universitätsumfeld vorgeschlagen ([24], [21]).

Architektur-Implementierung

Die von der Architekturexploration festgelegte Definition der Architektur wird für die Erstellung des strukturalen Architekturmodells verwendet. Die vielfach durchgeführte Abstraktion muß bei der Erstellung des Architekturmodells wieder rückgängig gemacht werden, da für die folgenden Implementierungsschritte die komplette Funktionalität im Modell enthalten sein muß.

Mit dem Modell sind auch die Festlegungen getroffen, welche Teile des Systems in HW bzw. in SW zu implementieren sind. Von diesem Schritt an können die bekannten Verfahren zur Erzeugung von HW und SW mit den entsprechenden Werkzeugen eingesetzt werden. Die oben beschriebenen Designwerkzeuge besitzen dabei bereits vielfach entsprechende Codegeneratoren, deren Ergebnisse entsprechend mit den speziellen Implementierungstools weiterverarbeitet werden kann. Sollen in einem SoC unterschiedliche Prozessoren eingesetzt werden können, sind hier insbesondere auch retargierbare Compiler erforderlich.

2.2 Partitionierung für HW/SW-Bausteine

Wie oben beschrieben, werden im Rahmen der Architektur-Exploration mögliche Architektur-Alternativen für das System untersucht und die unter Einhaltung der Randbedingungen beste ausgewählt. Neben der Allokation geeigneter Ressourcen ist dabei die Abbildung der Funktion auf diese Ressourcen durchzuführen. Eine Beschreibung der für die Exploration relevanten Ansätze ist in Kapitel 2.1.3 enthalten. Im Rahmen der Architekturexploration kann es erforderlich sein, diesen Schritt mit jeweils modifizierten Architekturvorgaben mehrfach zu durchlaufen, bis die in der Anforderungsspezifikation beschriebenen Vorgaben eingehalten sind. Nachfolgend soll der Schritt der Partitionierung als Bestandteil der Exploration näher betrachtet werden.

Die Partitionierung geht von einer festen Architektur aus, für die die optimale Abbildung der zu implementierenden Funktion gesucht wird. Hierfür ist es notwendig, daß die gesamte Funktionalität in einzelne Teilfunktionen unterteilt ist, die die Objekte der weiteren Verarbeitungsschritte bilden.

2.2.1 Unterteilung einer gesamten Funktion in einzelne Teilfunktionen

Ein entscheidender Faktor, der die Partitionierungsaufgabe beeinflußt, ist die Granularität, mit der die gesamte zu implementierende Funktion in N einzelne Teilfunktionen unterteilt wird, für die jeweils zu entscheiden ist, auf welcher Verarbeitungsressource sie implementiert werden sollen. Wenn in der Architektur M unterschiedliche Ressourcen vorhanden sind, ergibt sich dabei eine maximale Anzahl vom M^N unterschiedlichen Lösungen. Diese Zahl erhöht sich noch erheblich, falls es pro Ressource mehrere, bzgl. Komplexität oder Kosten unterschiedliche Implementierungsvarianten gibt. Sind z.B. pro Ressource 2 Alternativen vorhanden, ergibt sich die maximale Anzahl vom Möglichkeiten zu $2^N \cdot M^N$. Sind für bestimmte Teilfunktionen durch anderweitige Randbedingungen schon Festlegungen getroffen, reduziert sich der Exponent N entsprechend. Diese Maximalzahl gilt nur, wenn

die vorhandenen Ressourcen jeweils alle *N* Teilfunktionen implementieren können. Im betrachteten Fall des Entwurfs von SoC Systemen können für bestimmte spezifische Funktionen einbezogene Beschleunigermodule jedoch nur einige sehr wenige Teilfunktionen realisieren, weshalb diese Zahl nur von theoretischer Bedeutung ist. Sie zeigt jedoch die Komplexität der Aufgabe, die den Einsatz geeigneter Optimierungsverfahren notwendig macht, die mit geeigneten Mitteln aus der großen Anzahl von Möglichkeiten zielgerichtet eine unter den vorgegebenen Randbedingungen möglichst optimale Lösung finden können. Die einzelnen Teile, in die eine gesamte Funktion unterteilt ist, sollen nachfolgend äquivalent als Teilfunktionen, Prozesse oder auch Tasks bezeichnet werden. Wenn speziell Bezug auf eine Notation als Graph genommen wird, stellen diese Teilfunktionen die Knoten des Graphen dar.
Wird eine sehr feine Unterteilung der Funktion mit großem *N* vorgenommen, führt dies zu einer großen Anzahl von Objekten, die gemäß obigem Zusammenhang in den weiteren Schritten zu einer erheblichen Komplexität führen können. Es ermöglicht jedoch eine sehr angepaßte Lösung für die jeweilige Architektur zu finden. Im Gegensatz dazu kann eine zu grob gewählte Unterteilung der Funktionalität den Aufwand reduzieren aber möglicherweise eine zu wenig angepaßte Lösung generieren. Außerdem ist es möglich, daß in den Schritten nach der Partitionierung durch einen zu geringen Detaillierungsgrad die Vorgänge in der Architektur nicht ausreichend erfaßt sind, so daß nachträglich Optimierungen auf anderer Ebene durchgeführt werden müssen, die bei der Partitionierung einfacher möglich wären. In der Literatur wurden auch Verfahren vorgeschlagen, bei denen die Granularität dynamisch modifiziert wird, um eine schnellere Konvergenz und eine bessere Anpaßbarkeit in Bezug auf die Bewertung der Güte einer Lösung zu erreichen ([49]), bzw. bei denen die Funktionalität mit hierarchischen Graphen modelliert wird ([21]). Mit der Unterteilung in einzelne Teilunktionen kann das Problem der Partitionierung generell als Graph-basiertes Problem dargestellt werden. Nachfolgend soll deshalb die unterteilte Funktion immer als Graph und die Teilfunktionen als Prozesse bzw. Knoten verstanden werden.

2.2.2 Abschätzung der Performance Daten

Um die verschiedenen alternativen Abbildungen auf die Ressourcen der Architektur bewerten zu können, ist für jede Teilfunktion eine Schätzung zur Implementierung auf den jeweils möglichen Ressourcen erforderlich. Diese Schätzung muß einerseits möglichst schnell verfügbar sein und ohne großen Aufwand erstellt werden können, d.h. insbesondere ohne die Implementierung tatsächlich durchzuführen, andererseits muß sie ihr jedoch möglichst nahe kommen, um die relevanten Parameter wie Verarbeitungsdauer bzw. Verzögerung, Leistungsverbrauch oder Flächenbedarf, die bei der Optimierung mit einbezogen werden müssen, zuverlässig zu liefern.
Bei der Abschätzung ist zu berücksichtigen, daß für die dabei betrachteten Ressourcen unterschiedliche Realisierungsvarianten möglich sind, die sich in Bezug auf einzelne Parameter wie z.B. Verarbeitungsdauer und Aufwand stark unterscheiden. Entweder müssen die Anforderungen bei der Abschätzung bereits bekannt sein, um die gewünschte Implementierungsalternative auszuwählen oder die verschiedenen Möglichkeiten werden im Rahmen der Architekturexploration evaluiert. Einzelne Verfahren zur Partitionierung wie z.B. [58]

können deshalb auch zwischen unterschiedlichen Implementierungsvarianten auf einer Ressource auswählen.
Die verschiedenen in der Literatur veröffentlichten Partitionierungsverfahren gehen jeweils davon aus, daß Implementierungsmetriken vorhanden sind und entweder durch geeignete Verfahren abgeschätzt bzw. der Spezifikation von IP Modulen entnommen werden können. Methoden zur Abschätzung der relevanten Metriken auf hoher Ebene, die mögliche Implementierungen der einzelnen Prozesse auf den jeweiligen Ressourcen charakterisieren, sind z.B. in [50] und [110] für HW sowie in [6] und [74] für SW beschrieben.

2.2.3 Abbildung der einzelnen Teilfunktionen auf die Architektur

Unter dieser Teilaufgabe ist die Zuordnung der einzelnen Teilfunktionen auf die vorhandenen Ressourcen in einer Architektur zu verstehen. In der Literatur wird dieser Schritt meist mit dem Begriff Mapping bezeichnet.
Ausgangspunkte für die Abbildung sind die funktionale Spezifikation in Form einer Definition von Prozessen, die über entsprechende Schätzverfahren für die einzelnen Ressourcen der Architektur charakterisiert sind, sowie die Zielarchitektur. Konkret bedeutet die Abbildung der Prozesse auf die verfügbaren Ressourcen, daß sie mit den für die gewählte Implementierung geltenden Parametern annotiert werden. Damit ist die Voraussetzung geschaffen, die sich ergebende Lösung insgesamt zu bewerten.
Die Entscheidung, auf welche Ressource ein Prozeß abgebildet werden soll, hängt vom verwendeten Optimierungsalgorithmus ab. Darauf wird nachfolgend eingegangen.

2.2.4 Erstellen einer Verarbeitungsreihenfolge der Teilaufgaben (Scheduling)

Den letzten Teil der Partitionierungsaufgabe stellt die Festlegung der Verarbeitungsreihenfolge dar, mit der die Aktivierung der einzelnen Prozesse auf den jeweiligen Verarbeitungsressourcen der Architektur bzw. die Steuerung der Verarbeitungsabfolge insgesamt im System erfolgt. Diese Aufgabe wird mit dem Begriff Scheduling beschrieben. Unter Scheduling wird teilweise auch die gesamte Partitionierungsaufgabe für Mulitporzessorarchitekturen, insbesondere auch die Abbildung auf die Ressourcen bezeichnet. In der vorliegenden Arbeit soll der Begriff jedoch ausschließlich im Sinne der Festlegung der zeitlichen Verarbeitungsreihenfolge verwendet werden.
Scheduling ist eine der entscheidenden Aufgaben in einer Umgebung mit parallel arbeitenden Verarbeitungseinheiten ([30], [68]). Man differenziert prinzipiell zwischen statischem und dynamischem Scheduling, das bei der Erstellung des Systems bzw. während der Laufzeit festgelegt wird. Da nach der Abbildung der Prozesse das vorliegende Scheduling Problem vollständig charakterisiert ist, d.h. da die Information über die Ausführungszeiten, die Abhängigkeiten und die Kommunikation bzw. Synchronisation der Prozesse verfügbar ist, kann die Verarbeitungsreihenfolge für den hier betrachteten Anwendungsfall statisch festgelegt werden.
Daneben sind präemptive und nicht-präemptive Verfahren zu unterscheiden, bei denen laufende Prozesse durch Tasks mit höherer Priorität unterbrochen werden können, bzw. bei denen ein laufender Prozeß immer beendet werden muß, bevor ein Taskwechsel stattfinden

kann. Die präemptive Schedulingstrategie erfordert jedoch zusätzlichen Aufwand für die Verwaltung und die Sicherung des Zustands des unterbrochenen Prozesses, weshalb sie für die hier relevanten SoC Architekturen nicht weiter betrachtet wird.
Daneben ist zu berücksichtigen, welche Architektur angenommen wird, und welche Fähigkeiten die vorhandenen Ressourcen besitzen. Hier ist zum einen die Topologie des Verbindungsnetzwerks zwischen den Ressourcen von besonderer Bedeutung und auch die Möglichkeit inwieweit Ressourcen mehrere Prozesse parallel ausführen können. Dies ist insbesondere für Beschleunigerblöcke von Bedeutung, die bzgl. der vorhandenen Bearbeitungspfade nicht eingeschränkt sind. Andererseits können damit Konfliktsituationen innerhalb der Architektur nicht erfaßt werden, wenn diese Annahme getroffen wird. Im Hinblick auf die im Rahmen dieser Arbeit betrachteten Anwendung und die Zielsetzung, auch Ressourcenkonflikte insbesondere auf der Kommunikationsarchitektur zu erfassen, wird angenommen, daß auf jeder Ressource jeweils nur eine Task gleichzeitig ausgeführt werden kann.
Die sog. List-Scheduling Technik ([68]) ist das meist angewendete Verfahren, auf dem auch viele spezielle weiterentwickelte Verfahren basieren. Dabei wird davon ausgegangen, daß allen Prozessen ein Prioritätswert zugewiesen wird. Die Auswahl der Prozesse erfolgt mit Hilfe einer Liste, die alle lauffähigen Prozesse beinhaltet, und von der der Prozeß mit der jeweils höchsten Priorität entnommen und der Verarbeitungsressource zugewiesen wird, die die frühestmögliche Bearbeitung erlaubt. Im Fall, daß die Abbildung bereits festgelegt ist, erfolgt die Zuordnung auf die dabei festgelegte Ressource.
Für die Bestimmung der Prioritäten existieren unterschiedliche Ansätze. Als Komponenten zur Berechnung von Prioritäten werden neben anderen Aspekten der sog. t-Level bzw. der b-Level ([68]) verwendet. Der b-Level oder auch Bottom-Level eines Knoten gibt die Länge des längsten Pfads zu einem Endknoten, der t-Level oder Top-Level die maximale Pfadlänge von einem Anfangsknoten zum betrachteten Knoten an.
Zunehmend werden auch Verfahren vorgeschlagen, bei denen die Priorität der noch nicht festgelegten Prozesse im Laufe des Scheduling dynamisch angepaßt wird, um auf diese Weise eine angepaßtere Entscheidung unter Einbeziehung der bisher getroffenen Festlegung und ihrer Auswirkungen auf den Schedule zu berücksichtigen.

Die Partitionierungsaufgabe läßt sich damit grundsätzlich durch das in Bild 2-10 gezeigte Schema beschreiben.

2.2.5 Optimierung

Die Optimierung der Abbildung und des Scheduling der Teilfunktionen auf die Zielarchitektur stellt ein kombinatorisches Optimierungsproblem dar. Dabei muß entsprechend den Optimierungszielen der Architekturexploration eine Lösung gefunden werden, die für die gegenwärtig geltende Architektur, d.h. für die vorgegebenen Verarbeitungs- und Kommunikationsressourcen, die harten Randbedingungen einhält und gleichzeitig die übrigen Parameter optimiert.
Die optimierte Abbildung und des Scheduling eines Prozeßgraphen ist ein NP komplettes Problem ([30], [68], [39]). Es gibt nur einige wenige einfache Fälle, bei denen es möglich

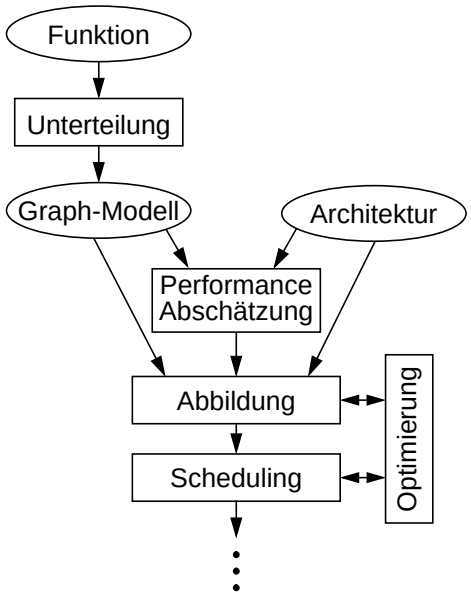


Bild 2-10: Ablauf der Partitionierung

ist, daß Algorithmen optimale Lösungen in polynomialer Zeit finden ([30]). Daher wurden für praktisch relevante Probleme viele heuristische Ansätze veröffentlicht, die brauchbare Lösungen in überschaubarer Zeit generieren können.
Gemäß [28] lassen sich Partitionierungsansätze grob in "greedy" Heuristiken, Clustering Methoden, iterative Verbesserungsansätze sowie mathematische Programmiermethoden unterteilen. Diese Verfahren unterscheiden sich auf vielfältige Art: im Optimierungsziel, im erfaßten Anwendungsgebiet und den dafür unterstützten Spezifikationsmodellen, in der Granularität des Graphen, der unterstützten Zielarchitektur oder anderen Randbedingungen. Aus diesem Grund ist ein direkter Vergleich sehr schwierig. Nachfolgend sollen einige Beispielverfahren dieser Lösungsansätze kurz vorgestellt werden.
Exakte Lösungen sind aufgrund der Komplexität des Problems nur in wenigen Fällen, d.h. bei sehr geringer Anzahl von Prozessen machbar. Dies trifft zum einen auf einfache Suchverfahren zu, die wie Backtracking den Lösungsraum vollständig absuchen, oder davon abgeleitete Methoden, die wie Branch-and-Bound Verfahren den Suchraum auf bestimmte Art und Weise begrenzen. Ähnliches gilt auch für mathematische Optimierungsverfahren wie Integer Linear Programming ([58]). In [86] ist ein auf Integer Linear Programming beruhendes Verfahren zur HW/SW-Partitionierung für Multiprozessorsysteme vorgestellt, das zur Reduktion des Berechnungsaufwands in zwei Phasen arbeitet: In der ersten Phase wird die Abbildung der Knoten auf die Architektur unter Zuhilfenahme einer Schätzung des Schedules berechnet, während in der zweiten Phase dafür der korrekte Zeitablauf generiert wird. Da in dieser Untersuchung nur sehr kleine Systeme mit max. 29 Knoten betrachtet werden, und kein Vergleich mit anderen Verfahren durchgeführt wird, ist eine Beurteilung des Aufwandes nicht möglich.
In [7] wird ein feingranulares, auf Clustering basierendes Verfahren für die Spezifikations-

sprache Unity vorgestellt, bei dem die als Elemente bezeichneten Partitionierungseinheiten nach unterschiedlichen Kriterien klassifiziert und zusammengefaßt werden mit dem Ziel der Minimierung der Schnitte im Clustering Baum. In einem zweiten Schritt werden Verbesserungen durch Pipelining und Ressourcen Sharing angestrebt. Als Architektur wird eine einfache aus einem Prozessor mit mehreren dedizierten Beschleunigern bestehende Anordnung zugrundegelegt, die über einen gemeinsamen Bus verbunden und über einen gemeinsamen Speicher kommunizieren.
Die folgenden Methoden arbeiten rekursiv, d.h.es wird iterativ eine Verbesserung der Lösung angestrebt, bis das Optimierungsziel erreicht ist. In [35] wird für das COSYMA Design System ein Verfahren zur Partitionierung für eine einfache Prozessor-Coprozessor Architektur beschrieben. Darin werden unter Nutzung von Simulated Annealing ausgehend von einer reinen SW-Implementierung sukzessive Teilfunktionen auf den Coprozessor verschoben, bis die Performance Kriterien erfüllt sind. Den umgekehrten Ansatz hierzu verwendet [44]. Hier werden solange Prozesse aus der ursprünglich reinen HW Implementierung in SW verlagert, bis die Anforderungen verletzt werden.
In [2] werden unterschiedliche heuristische Suchverfahren zur Partitionierung von Realzeitsystemen mit dem Ziel der Einhaltung von Deadlines bei Minimierung der Kosten untersucht. Die betrachteten Methoden sind neben Simulated Annealing und Tabu Suche auch genetische Algorithmen. Die Untersuchungen werden für komplexere Multiprozessor-Architekturen mit einem Bus durchgeführt und beinhalten auch Anpassungen der Architektur. Die Ergebnisse zeigen, daß genetische Algorithmen stets schlechtere Lösungen als die beiden anderen Verfahren liefern. Bezüglich der Qualität der Ergebnisse liegen gemäß [2] Simulated Annealing und Tabu Suche auf ähnlichem Niveau, wobei jedoch das letztere wesentlich schneller gute Lösungen und bei beschränkter Anzahl von Iterationen wesentlich bessere Lösungen liefert.
Weitere Untersuchungen zu Simulated Annealing und zur Tabu Suche sind in [33] enthalten. Auch hier soll die Performance unter Begrenzung des Implementierungsaufwands für HW und SW maximiert werden, jedoch wird nur eine einfache Architektur mit einem Prozessor und einem Beschleuniger betrachtet. Besonderes Augenmerk wird auf die Minimierung der Kommunikation zwischen HW und SW geachtet und eine möglichst hohe Parallelität angestrebt. Das Ergebnis bestätigt die in [2] getroffenen Aussagen hinsichtlich Güte der Ergebnisse und Laufzeit der Verfahren.
Zum Scheduling von Multiprozessorlösungen wird in [70] der FAST (Fast Assignment using Search Technique) Algorithmus vorgeschlagen, der in einer späteren Veröffentlichung ([69]) verbessert wurde, die auch eine parallele Implementierung enthält. Er basiert auf einer Suche mit Hilfe zweier verschachtelter Schleifen: In der inneren Schleife werden alternative, sich in der Zuordnung eines Prozesses unterscheidende Abbildungen des Graphen untersucht. Falls keine Verbesserung erreicht wird, wird die Änderung rückgängig gemacht. Diese Schleife wird solange ausgeführt, bis sich in einer bestimmten maximalen Anzahl keine Verbesserung mehr eingestellt hat. Daraufhin wird in der äußeren Schleife ein Prozeß des kritischen Pfades auf eine alternative Ressource geändert und die innere Schleife erneut ausgeführt. Die Auswahl von zu ändernden Prozessen bzw. Ressourcen erfolgt dabei jeweils zufallsgesteuert. Um die Veränderung durch die jeweiligen alternativen Abbildungen feststellen zu können, wird für jede Abbildung jeweils ein Schedule generiert.

Die jeweils gefundene beste Lösung wird gemerkt. Durch vorgegebene Grenzen für die Anzahl der Schleifendurchläufe wird die probabilistische Suche bei FAST zeitlich begrenzt.
Ein weiteres auf lokaler Suche basierende Methode ist in [122] beschrieben. In dem als TASK (Topological Assignment and Scheduling Kernel) bezeichneten Verfahren wird eine durch ein beliebiges Verfahren generierte Anfangslösung, d.h. Abbildung und Scheduling der Prozesse auf einer Architektur, durch ein lokales topologisches Suchverfahren rekursiv verbessert. Hierbei werden sukzessiv für alle Knoten alternative Zuordnungen auf Verbesserungen der Schedule Dauer überprüft. Bei einer Verbesserung wird der zugehörige Prozeß auf eine neue Ressource verlagert, und der Schedule entsprechend angepaßt. Die Vorgehensweise entspricht einer Kompaktierung des Schedule und damit Verkürzung der Ablaufzeit. Dieses Verfahren ist sehr effizient, da keine komplett neuen Schedules generiert werden müssen, wenn Prozesse alternativ abgebildet werden, sondern nur die betroffenen Zeitbereiche bei den jeweiligen Ressourcen angepaßt werden müssen. Der Algorithmus unterstützt jedoch nicht die Auflösung von Ressourcenkonflikten auf der Kommunikationsarchitektur und geht von homogenen, d.h. identischen Verarbeitungsressourcen aus. Durch das einfache Verschieben von Prozessen ohne Neu-Scheduling ist zudem eine Erweiterung zur Unterstützung von Kontrollabhängigkeiten im Graphen nicht möglich.
In [17] ist ein iteratives Verfahren beschrieben, das für eine Prozessor-Koprozessor Architektur eine gemeinsame Mapping- und Scheduling-Entscheidung trifft und dabei die Ausführungsdauer unter Einhaltung einer Flächenvorgabe minimiert. Der Algorithmus basiert auf einer wechselweisen Verschiebung von Tasks von SW nach HW und umgekehrt, bis keine Beschleunigung mehr erreicht wird. Die noch nicht endgültig festgelegten Teilaufgaben werden jeweils für alle ihre Implementierungsvarianten überprüft und dabei ein Scheduling durchgeführt. Die Methode ist auf grobgranular spezifizierte Datenfluß-dominierte Anwendungen ohne Kontrollabhängigkeiten ausgerichtet und ist nur für sehr kleine Graphen geeignet.
Unter der Bezeichnung "greedy" Heuristiken versteht man Verfahren, die schrittweise die Lösung generieren und dabei unter bestimmten Annahmen für die Bewertung der im jeweiligen Schritt möglichen Abbildungs- und Schedulingalternativen die jeweils lokal beste Entscheidung treffen. Die nachfolgenden Algorithmen sind Beispiele für diese Klasse von Verfahren. In [123] ist ein konstruktiver Algorithmus zur Abbildung und zum Scheduling eines Prozeß-Graphen beschrieben, der auch Kontrollabhängigkeiten beinhalten kann. Dieses Verfahren basiert auf einer sukzessiven Abarbeitung der einzelnen Knoten des Graphen mit Hilfe eines List Schedule. Für alle lauffähigen Knoten werden die mit Hilfe eines als "Dynamic Urgency" bezeichneten Maßzahl unterschiedliche Varianten der Abbildung und des Scheduling verglichen und die höchstpriore Möglichkeit ausgewählt.
Einen ähnlichen Ansatz verfolgt [58] mit dem GCLP (Global Criticality / Local Phase) Ansatz. Hier wird die Entscheidung ebenfalls anhand eines globalen Maßes getroffen, wobei jedoch die Minimierung des HW-Aufwands bei gleichzeitiger Einhaltung von Deadlines möglich ist. Bei jedem Abbildungsschritt wird dynamisch entschieden, welches Optimierungsziel die höhere Priorität besitzt. Im Gegensatz zu [123] werden jedoch keine bedingten Übergänge unterstützt, jedoch ist in [58] eine Erweiterung enthalten, die unter

verschiedenen Implementierungsvarianten bei Implementierung auf einer Ressource auswählen und so zusätzlich die benötigte Fläche optimieren kann.
Ein weiteres Verfahren, das Kontrollabhängigkeiten berücksichtigt, ist in [31] beschrieben. Die Methode basiert auf einer Überlagerung einzelner Teilgraphen, die bestimmten Kombinationen von Bedingungen entsprechen. Der Algorithmus liefert die Schedule Dauer für den ungünstigsten Fall und optimiert dabei das Scheduling und die Parameter des Kommunikationsprotokolls. Dabei wird jedoch von einer festen Abbildung der Tasks auf die Verarbeitungsressourcen ausgegangen.

Nachfolgend sollen zunächst die Anforderungen aus dem betrachteten Anwendungsgebiet näher betrachtet werden, bevor im 3. Kapitel die konkrete Motivation für die vorliegende Arbeit dargestellt wird.

2.3 Networking

Die im Rahmen dieser Arbeit entwickelte Methode zur Partitionierung von SoC Architekturen wurde insbesondere im Hinblick auf ihre Anwendung im Bereich der Bearbeitung von Datenkommunikationsprotokollen entwickelt. Da dieses Anwendungsgebiet kontrolldominiert ist, kommt der Unterstützung von Kontrollabhängigkeiten besondere Bedeutung zu. Zudem müssen derartige Systeme eine besonders hohe Leistungsfähigkeit in Bezug auf die systeminterne Kommunikation besitzen. Um die Anforderungen, die sich daraus ergeben, herleiten zu können, soll deshalb nachfolgend zunächst ein grober Überblick über den Stand und die Trends in diesem Gebiet gegeben werden.

2.3.1 Bedeutung des Internet

Der Bereich Datenkommunikationsnetzwerke ist in den letzten Jahren stark von den Entwicklungen im Zusammenhang mit dem Internet geprägt worden. Das zunächst mit dem Hintergrund der Ausfallsicherheit unter militärischen Aspekten entwickelte und zwischenzeitlich insbesondere als Netz zur Verbindung der Forschungseinrichtungen dienende Internet hat sich zum weltumspannende Datennetz und als Plattform für die unterschiedlichsten Datendienste weiterentwickelt. Die Zunahme der Bedeutung des Internet läßt sich an seinem eindrucksvollen Wachstum ablesen. In Bild 2-11 ist die Zunahme der am Internet angeschlossenen Rechner gemäß den Untersuchungen des Internet Software Consortiums ([54]) dargestellt.

Neben den ursprünglichen Datendiensten wie ftp, telnet oder mail werden in den letzten Jahren vermehrt attraktive, um multimediale Elemente erweiterte Dienste über das Internet abgewickelt. Diese Entwicklung hat insbesondere das World Wide Web (WWW) als einfache Möglichkeit der Bereitstellung und des Abrufens unterschiedlichster, oftmals multimedial angereicherter, Informationen aus dem Netz ermöglicht. Neben Graphiken und Bilddaten werden zunehmend auch Audio- und Videosequenzen über das Internet übertragen. Die damit verbundenen umfangreichen Datenmengen bedingen, daß zur Sicherstellung einer ausreichend kurzen Reaktionszeit beim Abruf ausreichend Kapazitäten im Netz,

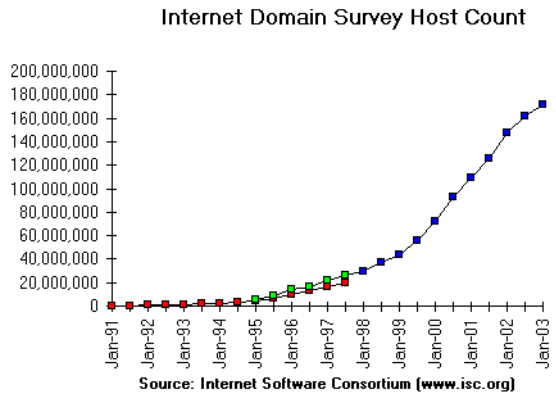


Bild 2-11: Wachstum des Internet

insbesondere beim Teilnehmeranschluß, zur Verfügung stehen. Zunächst wurden derartige Daten heruntergeladen, zwischengespeichert und mit den zugehörigen Anwendungsprogrammen offline wiedergegeben. Zunehmend werden jedoch Inhalte über das Internet im sog. Streaming Modus angeboten, bei dem die Daten im Endgerät nur für kurze Zeit zum Ausgleich von schwankenden Laufzeiten der einzelnen Datenpakete zwischengespeichert, ansonsten aber in Echtzeit dargestellt und danach verworfen werden. Auch werden zunehmend Anwendungen relevant, bei denen die Teilnehmer in Echtzeit über das Netz interaktiv kommunizieren. Derartige isochrone Anwendungen stellen jedoch wesentlich höhere Anforderungen an das Netz. Heutige Lösungen leiden jedoch noch unter der grundlegenden Eigenschaft des Internet, daß keine garantierte Qualität hinsichtlich Laufzeit bzw. Laufzeitschwankungen und Paketverlust sondern nur "best-effort" Verkehr möglich ist. Aus diesem Grund ist eine der wichtigsten Anforderungen an die Weiterentwicklung des Internet, Dienstgüten bereitzustellen, die unabhängig von der Netzlast und anderer Faktoren eine garantierte Ende-zu-Ende Übertragung der Daten in Echtzeit ermöglicht.
Aufgrund des enormen Wachstums des Internet ergibt sich eine zunehmende Dominanz der Datennetze gegenüber dem konventionellen Telefonnetz. Aus Gründen der Wirtschaftlichkeit und der durch Vereinheitlichung möglichen Vereinfachung des Managements der Netze wird die Konvergenz von Sprach- und Datennetzen angestrebt. In einer derartigen konvergierten Netzlandschaft wird erwartet, daß über ein gemeinsames, um die Unterstützung von Dienstgüten erweitertes Datennetz neben den bekannten Diensten auch interaktive Echtzeitdatendienste und insbesondere auch der Telefondienst mit der aus dem bisherigen Telefonnetz bekannten Qualität abgewickelt werden. Als Voraussetzung hierzu muß das heutige Internet durch geeignete Techniken erweitert werden, um die mit diesem Trend verbundenen Anforderungen erfüllen zu können. Da aus heutiger Sicht noch nicht endgültig geklärt ist, welche Techniken übergreifend eingesetzt werden, und auch die Standardisierung noch nicht abgeschlossen ist, müssen Systeme der Datenkommunikation flexibel genug sein, um entsprechende Änderungen auch in bereits installiertem Equipment vornehmen zu können.
Der Erfolg des WWW beim Teilnehmer hat zu einer ungeahnten Zunahme des Verkehrs-

volumens im Internet geführt. Die Nutzerzahlen sowie die Anzahl der am Internet angeschlossenen Rechner hat in den vergangenen Jahren exponentiell zugenommen. Selbst wenn dieser Trend in Zukunft abgeschwächt werden sollte, wird die gegenwärtig stattfindende Einführung breitbandiger Anschlußtechniken wie ADSL für die breite Masse der Heimnutzer und die damit möglichen datenintensiveren Anwendungen zu einer weiteren Zunahme der transportierten Datenmenge führen. Die Anforderungen, die das eingesetzte Equipment in Bezug auf den Datendurchsatz erfüllen muß, wird deshalb auch weiterhin steigen.
Zusammenfassend läßt sich feststellen, daß die beiden Haupttrends in der Weiterentwicklung der Datennetze, die sich aus der geschilderten Änderung des Nutzerverhalten bzw. in Bezug auf die über das Internet abgewickelten Anwendungen ergeben, in der Erhöhung des Durchsatzes der Systeme sowie in der Bereitstellung von Dienstgüten im Netz liegen.
Nachfolgend wird kurz drauf eingegangen, mit welchen Techniken, diese Anforderungen angegangen werden.

2.3.2 Protokolle der Datenkommunikation

Ausgangspunkt für die Definition des Internet Protokolls (IP) war die Zielsetzung, auch dann noch Ende-zu-Ende Kommunikation zu ermöglichen, wenn Teile des Netzes ausgefallen sind. Aus diesem Grund wurde IP als verbindungsloses Protokoll implementiert, bei dem keine zentralen Instanzen für die Verwaltung von Verbindungen bzw. generell für das Management des Netzes vorhanden sind. Dies bedeutet, daß bei der Weiterleitung der Pakete variabler Länge in den beteiligten Knoten (Routern) für jedes Paket von neuem entschieden werden muß, auf welchem Ausgangsport des Routers das Paket ausgegeben werden muß, damit es letztlich bei der gewünschten Zieladresse ankommt. Durch diese Vorgehensweise ist es in Verbindung mit Routingprotokollen möglich, Änderungen an der Netzkonnektivität bzw. Topologie quasi automatisch zu erfassen und in die Datenbasis, anhand derer die Weiterleitung der Pakete in den Routern erfolgt, einzubauen und für die Weiterleitung der Pakete zu berücksichtigen.
Aufgrund der Begrenzungen im von IPv4 unterstützen Adreßraum von 32 Bit breiten Adressen und in Verbindung mit der bisherigen Praxis bei der Vergabe von Internetadressen sowie zur besseren Unterstützung der schnellen Weiterleitung wurde die Weiterentwicklung zu IPv6 standardisiert. Neben der Vergrößerung der Adressen auf 128 Bit wurde hierbei eine wesentliche Vereinfachung des Paketheaders vorgenommen, wodurch der Verarbeitungsaufwand in den Routern begrenzt und so die Erreichung höherer Durchsätze unterstützt werden soll. Weiterhin wurden im Header durch Definition eines Flow-Labels die Unterstützung der Identifizierung zusammengehörender Datenströme verbessert.
Eine ausführliche Darstellung des Internetprotokolls ist in [19] enthalten.
Als alternativer Ansatz zur Implementierung eines Breitbandnetzes wurde Ende der 80-er Jahre der Asynchrone Transfer Modus (ATM) für die logische Weiterentwicklung des Telefonnetzes zum Breitband ISDN entwickelt und von der ITU-T standardisiert. Diese Technik wurde von den Equipmentherstellern und Betreibern der öffentlichen Telekommunikationsnetze entwickelt und später auch von den Computernetzwerk-Herstellern übernommen. In diesem Bereich konnte sich ATM jedoch aufgrund seiner relati-

ven Komplexität gegen das sich stetig weiterentwickelnde und einfach handzuhabende Ethernet und aufgrund des Bestands der auf IP basierenden Anwendungen nicht durchsetzen. Insgesamt ist die Bedeutung von ATM rückläufig, auch wenn es heute im Kernnetz und insbesondere beim Netzzugang über ADSL breit eingesetzt wird. Die im Zusammenhang mit ATM entwickelten Konzepte finden sich jedoch in den Ansätzen zur Weiterentwicklung des Internet Protokolls zur Unterstützung von Dienstqualitäten wieder.
ATM arbeitet verbindungsorientiert und transportiert die Daten in Zellen fester Länge. Beim Aufbau einer Verbindung muß dabei der Weg durch das Netz festgelegt und in den beteiligten ATM Switchen die zugehörigen Ressourcen reserviert werden. Im Gegensatz zum Internet Protokoll erlaubt ATM unterschiedliche Dienstklassen, insbesondere auch garantierte Qualität für Dienste mit konstanter bzw. variabler Datenrate.
ATM kann auch für den Transport von IP genutzt werden und dabei als Übertragungsplattform mit QoS Möglichkeiten diese für IP nutzbar machen. Dieser Ansatz führt jedoch zu teilweise erheblichem Aufwand zur Segmentierung und Reassemblierung der Datenpakete im Netz und ist deshalb nur in Teilbereichen oder als Zwischenlösung relevant. Weitere Ansätze zur Bereitstellung von QoS im Internet liegen in der Einführung des Integrated Services (Intserv) bzw. Differentiated Services (Diffserv) Konzepts sowie in der Nutzung von Multiprotocol Label Switching (MPLS).
Unter Intserv wird ein Ansatz verstanden, bei dem in den beteiligten Routern mit Hilfe eines Reservierungsprotokolls die benötigten Ressourcen für jeden einzelnen Datenfluß (Flow) bzw. jede einzelne Verbindung reserviert werden. Dieses Vorgehen ist relativ aufwendig, da regelmäßige Auffrisch-Mitteilungen für die Ressourcenreservierung verschickt werden müssen und jeder Router den Status jedes einzelnen Flusses, der über ihn läuft, speichern muß. Dies führt zu Problem in der Skalierbarkeit dieses Ansatzes.
Diffserv dagegen ist ein Konzept, bei dem keine absoluten Dienstegarantieen gegeben werden sondern mit Hilfe von Diensteklassen (Class of Service) nur relative Qualitätsgarantien unterstützt werden. Dabei wird davon ausgegangen, daß Pakete anhand bestimmter Kriterien klassifiziert und im Netz entsprechend der jeweiligen Diensteklasse mit der zugehörigen Priorität behandelt werden. Gleichartig klassifizierte Flüsse (auch als Mikroflüsse bezeichnet) werden dabei aggregiert und im Netz gemeinsam behandelt. Auf diese Weise entfällt die aufwendige individuelle Behandlung der einzelnen Mikroflüsse, was zu einer besseren Skalierbarkeit dieses Konzepts führt.
MPLS stellt einen Ansatz dar, bei dem den Paketen, die in eine MPLS-Domäne eintreten eine Kennung (Label) angefügt wird, das innerhalb der Domäne zur Weiterleitung benutzt wird. Dadurch ist der aufwendige Nexthop-Lookup in den Routern durch eine einfachere Switching-Entscheidung ersetzt. Dieses verbindungsorientierte Konzept setzt voraus, daß innerhalb der MPLS-Domäne entsprechende Pfade zu den Routern, die den Zieladressen am nächsten sind, bereits etabliert sind, auf denen die Pakete dann weitergeleitet werden. MPLS ist nicht primär eine QoS Methode sondern ermöglicht über die Verbindung des von ATM bekannten Weiterleitungskonzepts mit dem Routing Konzept des Internetprotokolls eine effiziente Übermittlungstechnik für das Kernnetz, das sowohl auf IP als auch auf ATM basieren kann. MPLS unterstützt jedoch die Umsetzung von Diffserv. Damit kann eine Verteilung der unterschiedlich priorisierten Verkehrsströme auf unterschiedliche Pfade durch das Netz vorgenommen und so eine angepaßte Verkehrslenkung (Traffic Enginee-

ring) der verschiedenen Verkehrsklassen umgesetzt werden.
Die genannten Verfahren beruhen jeweils darauf, daß in den Routern, die die genannten Konzepte unterstützen, jeweils geeignete Verfahren zur Überwachung und Begrenzung der Verkehrsströme sowie zum Scheduling und zum Management der Warteschlangen implementiert sind, die die das Netz vor Überlastung schützen, unterschiedlich priorisierte Pakete in geeigneter Weise behandeln und insbesondere Fairness garantieren.
Die Implementierung von Mechanismen zur Bereitstellung von Dienstgüten im Internet ist ein Themengebiet, das noch nicht abschließend geklärt ist. Auch wenn bereits unterschiedliche Konzepte in aktuellen Produkten der Netzwerkhersteller umgesetzt und in verschiedenen Netzen zum Einsatz kommen, sind viele Fragestellungen noch ungeklärt. Für die nahe bis mittelfristige Zukunft sind deshalb noch Anpassungen in den Konzepten zu erwarten. Eine Übersicht über den aktuellen Stand zu den QoS Ansätzen für das Internet ist in [94] enthalten

2.3.3 Zukünftige Netzstruktur

Aus den oben beschriebenen Anforderungen und den Ansätzen zu deren Umsetzung mit geeigneten Protokollen ergeben sich zwei grundsätzliche Trends in Bezug auf die Struktur des zukünftigen Internets:

- Der Kern des Netzes muß zur Bewältigung der Datenvolumina zunehmend höhere Übermittlungsleistungen bereitstellen.
- Am Netzzugang besteht aufgrund der neuen Konzepte und aufgrund unterschiedlicher dort genutzter Zugangstechniken ein erhöhter Verarbeitungsaufwand.

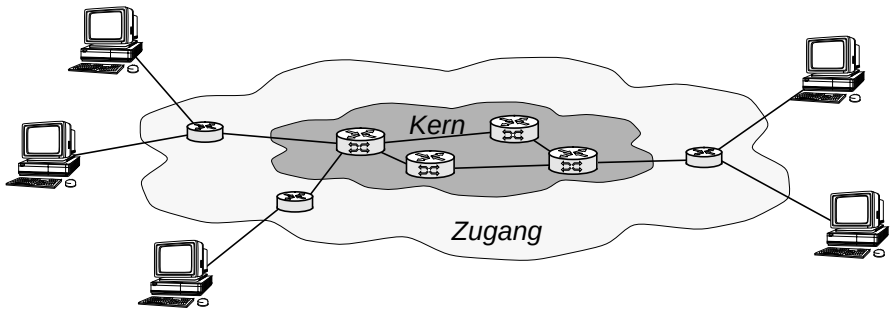


Bild 2-12: Zukünftige Struktur des Internet mit QoS Unterstützung

Diese Unterteilung der Netzstruktur in einen Zugangsbereich (Edge) und den Netzkern (Core) spiegelt sich auch in den jeweils dafür eingesetzten Routern wieder, die die jeweiligen Anforderungen erfüllen müssen. Insbesondere im Kernnetz, jedoch auch zumindest für die Verbindung des Edge-Bereichs mit dem Kernnetz, müssen Router Schnittstellen mit Datenraten von 10 Gbits/s bis 40 Gbit/s besitzen. Die von Core-Routern geforderten aggregierten Durchsätze liegen in der Größenordnung von Tbit/s. Während im Kernnetz die Be-

tonung auf hoher Übermittlungsleistung liegt, die - wie oben beschrieben - durch reduzierte Verarbeitungstiefe erleichtert wird, sind am Netzzugang im entsprechenden Equipment als Ausgleich dafür entsprechend komplexere Verarbeitungsschritte erforderlich, was jedoch bei geringeren Anschlußgeschwindigkeiten leichter möglich ist. Die Anschlußdatenraten liegen hier in der Größenordnung von Gbit/s und die aggregierten Durchsätze bei bis zu 100 Gbit/s. Zu der von Edge-Routern zu unterstützenden erhöhten Komplexität der Verarbeitung trägt auch bei, daß eine Vielzahl unterschiedlicher Anschlußtechniken, nicht nur bezüglich der Übertragungstechnik sondern insbesondere auch bezüglich der auf den OSI Schichten 2 und 3 in diesem Bereich verwendeten Protokollszenarien, unterstützt werden muß. Dient der Router auch für die Aggregation vieler niederratiger Anschlüsse kommt der Port-Dichte, d.h. der Anzahl der unterstützen Schnittstellen pro Router Linecard, der Skalierbarkeit und auch der Anzahl der unterstützten Teilnehmer besondere Bedeutung zu. Aufgrund der Vielzahl von Protokollszenarien und der noch nicht endgültig festgelegten Konzepte für die QoS-Unterstützung sind für Edge-Router besondere Anforderungen hinsichtlich Anpaßbarkeit an neue Anforderungen zu erfüllen.
Im nächsten Kapitel soll deshalb zunächst auf die wichtigste Voraussetzung für die Implementierung entsprechender Systeme und die dabei eingesetzten Komponenten eingegangen werden.

2.4 Systeme und Bausteine für die Datenkommunikation

Wie oben dargestellt, spielen die Internet Router, die die o.g. erweiterten Funktionalitäten bei gleichzeitig erhöhten Anforderungen an den Durchsatz implementieren müssen, eine zunehmend entscheidende Rolle im zukünftigen Internet. Die neuen Anforderungen spiegeln sich deshalb auch in der Weiterentwicklung der Router-Konzepte wieder.
Ein Router ist üblicherweise aus folgenden grundlegenden Bestandteilen aufgebaut ([3]): mehrere Netzwerkschnittstellen, eine oder mehrere Prozessierungseinheiten, Zwischenpuffer und eine Einheit zur Verbindung der internen Komponenten. Grundsätzlich lassen sich zwei Verarbeitungszweige in einem Router unterschieden, die unterschiedliche Randbedingungen erfüllen müssen. Zum einen betrifft dies die Weiterleitung der ankommenden Pakete und die zugehörige Verarbeitung der betreffenden Kommunikationsprotokolle. Die damit zusammenhängenden Teilaufgaben werden auch oft als "Forwarding" bzw. aufgrund der dabei zu erfüllenden hohen zeitlichen Anforderungen als "Fast Path" bezeichnet. Darunter fallen z.B. die Verifikation von Paketen, Extraktion von bestimmten Feldern wie Ziel- oder Senderadressen, Lookups in Tabellen, Statistik-Funktionen etc., die für jedes weiterzuleitende Paket schritthaltend mit der eingangsseitigen Paketrate ausgeführt werden müssen. Andererseits sind verschiedene weitere Teilfunktionen im Router zu implementieren, für die geringere zeitliche Anforderungen erfüllt werden müssen. Hierunter fallen insbesondere Control- bzw. Managementaufgaben oder Funktionen, die sehr komplexe Bearbeitungsschritte beinhalten. Derartige Funktionen gehen auch oft mit der Verwaltung von Zuständen einzelner Verbindungen wie z.B. die Nutzerauthentifizierung, die Zuweisung von Adressen o.ä. einher. Die wichtigste Aufgabe dieser als "Control Plane" oder auch "Slow Path" bezeichneten Funktionalitäten besteht in der Realisierung der Routing-Protokolle, mit deren Hilfe die Routen innerhalb des Netzwerks berechnet werden, sowie die

Verwaltung der zugehörigen Routing Tabelle. Bild 2-13 zeigt die generische Architektur eines Routers.

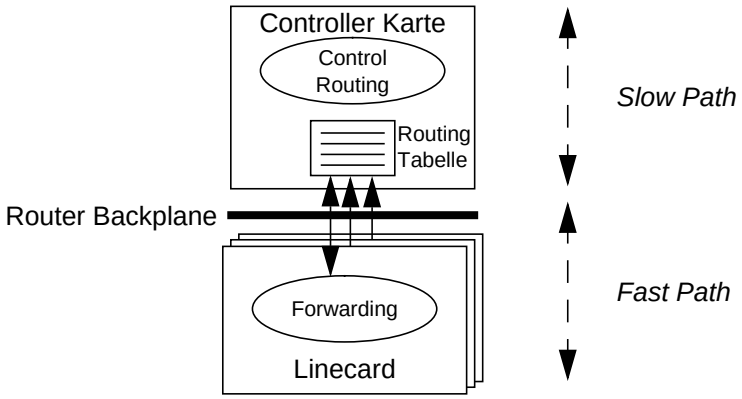


Bild 2-13: Prinzipelle Router-Architektur [3]

IP Router wurden zunächst als reine Softwarelösung implementiert, bei der die komplette Funktion des Weiterleitens der Datenpakete auf einem Standardprozessor ausgeführt wurde. Ein derartiger SW Router wurde meist auf einem Rechner mit mehreren Netzwerkkarten realisiert, bei dem alle ankommenden Pakete von der CPU gesteuert von den Netzwerkkarten über den Systembus in den Hauptspeicher des Rechners transferiert und zwischengespeichert wurden. Nach der Routing-Entscheidung der CPU wurden die Pakete aus dem Speicher wieder ausgelesen, in der CPU modifiziert und über das entsprechende Netzinterface ausgegeben. Mit höher werdenden Anforderungen an die Geschwindigkeiten der Anschlußleitungen bzw. zunehmenden Durchsätzen wurden zunächst mehrere Prozessoren zentral eingesetzt und schließlich dazu übergegangen, mehr Verarbeitungsfunktionen auf die Schnittstellenkarten zu verlagern. Mit diesem Schritt wurde auch die Unterteilung in die oben beschriebenen Teile des Slow und des Fast Paths vorgenommen, die auch heutigen Routern zugrundeliegt. Zunächst wurden in derartigen dezentralen Architekturen CPUs eingesetzt und im weiteren auch spezifische ASICs hierfür verwendet. Aktuelle Hochgeschwindigkeitsrouter sind in einer komplett dezentralen Architektur implementiert, bei der intelligente Linecards über ein Verbindungsnetzwerk verbunden wurden.

Je mehr unterschiedliche Anforderungen insbesondere hinsichtlich der gleichzeitig zu verarbeitenden Protokolle oder hinsichtlich eines Einsatzes in unterschiedlichem Equipment zu erfüllen sind, kommt neben der hohen Verarbeitungsgeschwindigkeit der Flexibilität entscheidende Bedeutung zu. Deshalb wird zunehmend die Verwendung komplexer und relativ aufwendig zu implementierender Spezial-ASICs durch einen neuen auf Standardbauteilen basierenden Ansatz abgelöst.

Um die Anforderungen hinsichtlich hoher Verarbeitungsgeschwindigkeit bei gleichzeitiger Flexibilität zu erfüllen, wurden in den letzten Jahren sog. Netzwerkprozessoren als eine neue Klasse von Bausteinen zur Bearbeitung von Kommunikationsprotokollen auf den

Markt gebracht. Damit soll es möglich werden, die hohen Anforderungen an Durchsatz zu erfüllen und gleichzeitig eine schnelle Entwicklung von entsprechenden Lösungen zu erlauben. Der Anwender eines Netzwerkprozessors erstellt damit seine Lösung durch die Programmierung von SW, die auf einem Standardbaustein abläuft, der in einer möglichst einfachen Systemarchitektur eingesetzt ist. Dieses Vorgehen zielt auf eine Steigerung der Effizienz solcher Lösungen bei gleichzeitiger Reduktion des wirtschaftlichen Risikos derartiger Implementierungen ab. Sollten Fehler im Entwurf enthalten sein, können diese durch eine einfache SW-Änderung korrigiert werden, anstatt einen langwierigen und kostspieligen Neuentwurf durchführen zu müssen, was bei einer dedizierten HW-Lösung erforderlich wäre. Ein weiterer Vorteil durch den Einsatz von Netzwerkprozessoren soll sich aus der Möglichkeit ergeben durch entsprechende SW-Updates die Funktion des Equipments nachträglich zu ändern, d.h. entweder an geänderte Standards anzupassen oder neue Funktionalitäten einzuspielen, ohne die HW ersetzen zu müssen.
Auf dem Markt sind mittlerweile eine Reihe von Netzwerkprozessoren verfügbar, die sich in ihrer Architektur und dem darauf abgebildeten Verarbeitungskonzept, den Schnittstellen, dem Programmiermodell und insbesondere auch in ihrer Verarbeitungsgeschwindigkeit und damit zusammengehörend dem Einsatzgebiet unterscheiden. Beispiele sind der Intel IXP1200 bzw. IXP2400 ([53]), der IBM PowerNP ([51]) oder der EZChip NP 1 ([36]).
Netzwerkprozessoren bestehen meist aus einer größeren Anzahl parallel arbeitender, mit einem spezifischen Befehlssatz ausgestatteter Prozessoren, die die Teilaufgaben des Fast Path in SW realisieren. Oft besitzen Netzwerkprozessoren für bestimmte aufwendige Teilfunktionen bestimmte Beschleunigermodule wie z.B. zur Durchführung von Lookups oder zur Verwaltung von Warteschlangen etc. Für den Slow Path, die Initialisierung und für Überwachungsaufgaben sind zudem oft eingebettete Standardprozessoren integriert.
Netzwerkprozessoren liegen teilweise sehr unterschiedliche Architekturkonzepte zugrunde, die von Parallelverarbeitung unter Nutzung von HW Multithreading bis zu Pipelining-Ansätzen, die sich an den Verarbeitungsschritten der Paketverarbeitung orientieren, reichen. Sie besitzen eine Vielzahl von Schnittstellen, die die Verbindung zum Netz bzw. zum systeminternen Verbindungsnetzwerk bereitstellen, insbesondere aber auch eine breitbandige Andbindung von externem Speicher für Paketdaten bzw. sonstige Steuerinformationen oder die benötigten Lookup Tabellen erlauben. Daneben sind meist Interfaces für die Kommunikation mit einem externen Host-Prozessor vorhanden.

Der Ansatz der Systemrealisierungen mit Netzwerkprozessoren hat sich jedoch noch nicht endgültig etabliert, insbesondere aus Kostengründen bei großen Stückzahlen ist weiterhin mit einen Bedarf an maßgeschneiderten Lösungen für Bausteine der Datenkommunikation zu rechnen. Dabei müssen für den jeweiligen Einsatzfall unter Berücksichtigung der Möglichkeiten des VLSI Entwurfs geeignete Lösungen entwickelt werden. Die Untersuchungen in der vorliegenden Arbeit wurden deshalb mit Blick auf den effizienten Entwurf von entsprechend maßgeschneiderten Bausteinen aus dem Bereich der Datenkommunikation durchgeführt.

3 Motivation

Das in dieser Arbeit entwickelte Verfahren, das im Rahmen der Exploration von Systemarchitekturen verwendet werden kann, wurde speziell im Hinblick auf seinen Einsatz in Zusammenhang mit Anwendungen aus dem Bereich der Datenkommunikation insbesondere zur Bearbeitung von Datenkommunikationsprotokollen entwickelt. Die hierfür relevanten Anforderungen sollen in diesem Kapitel zunächst kurz dargestellt werden. Generell läßt sich feststellen, daß die Verarbeitung von Datenkommunikationsprotokollen durch folgende besonderen Anforderungen gekennzeichnet ist:

- Kontrolldominiertheit
- Hoher interner Kommunikationsbedarf

Zunächst soll auf diese beiden Aspekte näher eingegangen werden, bevor zum Schluß dieses Kapitels die für die Untersuchung getroffenen Annahmen und Randbedingungen näher erläutert werden.

3.1 Kontrolldominiertheit der Paketverarbeitung

Den in der Datenkommunikation eingesetzten Mechanismen liegt das Prinzip zugrunde, die gesamte für den Datenaustausch über ein Netzwerk erforderliche Funktionalität in einzelne Schichten zu unterteilen, die jeweils spezifische Teilfunktionen implementieren. Diese Teilfunktionen sind in sich modular und gekapselt, d.h. sie werden in der Regel unabhängig voneinander ausgeführt.
Das bekannteste dieser Konzepte ist das von der ISO standardisierte Modell ([124]) mit 7 Schichten, bei dem die gesamte Funktion in die Anwendungs-, Präsentations-, Sitzungs-, Transport-, Netzwerk-, Sicherungs- und die physikalische Schicht unterteilt ist. Im gegensatz hierzu besteht das Internet Kommunikationsmodell aus lediglich 4 Schichten, dem Anwendungs-, dem Transport-, dem Netzwerk- und dem Medium Access Layer. Beiden Fällen liegt jedoch das Prinzip zugrunde, daß sich eine Ebene zur Realisierung der eigenen Funktionalitäten der Dienste der darunterliegenden Schicht bedienen kann und der darüberliegenden Schicht jeweils seinen eigenen Dienst zur Verfügung stellt. Durch definierte Schnittstellen zwischen den Ebenen können diese relativ unabhängig voneinander entwikkelt werden. Dies führt zu einer modularen Implementierung von Kommunikationsmechanismen, bei der einzelne Schichten auch einfach ausgetauscht werden können (z.B. der Ersatz einer Fast Ethernet durch eine Gigabit Ethernet Karte oder der Austausch eines Transportprotokolls durch ein speziell für bestimmte Randbedingungen geeignetes), ohne daß andere Funktionalitäten geändert werden müssen.
Für die Implementierung der einzelnen Schichten bedeutet dies, daß den Daten in der Quelle ausgehend von der Anwendungsschicht mit jeder darunterliegenden Schicht ein zusätzlicher Protokollkopf (Header) angefügt wird. Aus den von der darüberliegenden Schicht gelieferten Daten (Service Data Unit, SDU) wird mit dem Header die neue Protocol Data Unit (PDU) gebildet, die wiederum die SDU für die darunterliegende Schicht darstellt. Bei diesem Vorgang muß im jeweiligen Header eine Information beigefügt werden, über die

bei der empfangsseitigen Bearbeitung erkannt werden kann, von welcher Art die enthaltenen Protokolldaten sind. Ein Beispiel hierzu ist die Information im IP Header über das im Paket enthaltene Transportprotokoll.
Wird der im Paket enthaltene Protokollstapel entweder in den Netzknoten zur Bearbeitung der für die Weiterleitung erforderlichen Teilfunktionen in Teilen bzw. beim Empfänger zur Rekonstruktion der ursprünglichen Nachricht komplett abgearbeitet, werden diese Informationen genutzt, um die zu den jeweiligen Protokollen gehörenden Verarbeitungsschritte aufzurufen. In der Senke wird umgekehrt zur Quelle zudem bei der Übergabe der Daten an die höhere Schicht der Header der niederen Ebene entfernt, bis die ursprünglichen Daten auf der Anwendungsschicht vorliegen. Bild 3-1 stellt das Schichtenkonzept schematisch dar.

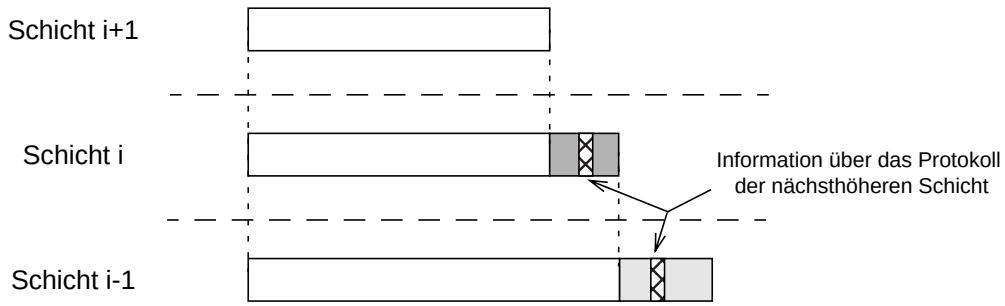


Bild 3-1: Schichtenkonzept der Datenkommunikation

Die Felder, die zur Unterscheidung der einzelnen Protokolle verwendet werden, sind standardisiert und erlauben die Unterscheidung einer Vielzahl unterschiedlicher Protokollstapel. Diese Heterogenität spiegelt entsprechend die unterschiedliche Nutzung der Kommunikationsnetze wieder, in denen auf der gleichen Plattform mit den gleichen Systemen verschiedenartige Protokollszenarien implementiert werden.
Neben diesen, der Unterscheidung der Protokolle dienenden Feldern führen eine Vielzahl weiterer Funktionalitäten zu einer differenzierten Behandlung der einzelnen Datenpakete. In Abhängigkeit bestimmter Verarbeitungsschritte, wie z.B. der Ergebnisse von Tabellen-Lookups, der Entscheidung über die Korrektheit und Integrität von Paketen oder der Ergebnisse von Überwachungsmechanismen müssen Datenpakete sehr unterschiedlich gehandhabt werden. Die Verarbeitung der Datenkommunikationsprotokolle ist deshalb ein Beispiel eines Anwendungsgebiets, das stark kontrolldominiert ist.

Für die Bearbeitung in den Netzknoten bedeutet dies, daß jedes ankommende Paket je nach beinhalteter Funktionalität und abhängig von gewissen systeminternen Zuständen unterschiedlich behandelt werden muß. Die gesamte Funktion, die alle Möglichkeiten beinhalten muß, besitzt deshalb eine sehr große Zahl von Fallunterscheidungen. Für die Implementierung dieser Funktionen ist, wie in Kapitel 2.4 dargestellt, sowohl der Fast Path als auch der Slow Path zuständig. Eine große Anzahl dieser Fallunterscheidungen ist im Slow Path zu implementieren, der üblicherweise in SW realisiert ist und weniger kritische

Zeitanforderungen erfüllen muß. Daneben sind jedoch auch im Fast Path für die direkt zur Weiterleitung der Pakete notwendigen Verarbeitungsschritte insbesondere aufgrund der am Netzzugang zunehmend komplexer werdenden Funktionalitäten erhöhte Anforderungen in dieser Beziehung zu erfüllen.
Eine Grundvoraussetzung, die Methoden zur Untersuchung von Systemarchitekturen für die Paketverarbeitung im Fast Path erfüllen müssen, ist folglich die Beherrschung der Kontrolldominiertheit in diesem Gebiet. Dieser Aspekt ist in letzter Zeit zunehmend zum Gegenstand der Forschungsarbeiten geworden. Bislang liegen jedoch, wie in Kapitel 2 erläutert, lediglich einige konstruktiv arbeitende Methoden vor.

3.2 Kommunikation innerhalb des Bausteins

Im oben geschilderten Vorgehen bei der Festlegung der Architektur auf der Systemebene wird dem Aspekt der systeminternen Kommunikation als einem für die erreichbare Performance entscheidenden Faktor zunehmend größere Bedeutung zugewiesen. Aus diesem Grund wurden die Forschungsaktivitäten zu diesem Themenkomplex in den letzten Jahren intensiviert.
Im Zusammenhang mit Anwendungen für die Protokollverarbeitung besitzt die interne Kommunikation besonderes Gewicht. In zugehörigen Systemarchitekturen müssen zwischen den einzelnen Verarbeitungs- und Speicherblöcken je nach gewählter Art der Implementierung intensiv Daten transferiert werden. Dies trifft sowohl für den Fall einer mehr softwarebasierten Implementierung als auch beim Einsatz von HW-Beschleunigermodulen zu, an die zur Nutzung ihrer Funktionalität die erforderlichen Daten übertragen werden müssen. Neben der Speicherung der Paketdaten selbst müssen insbesondere zur Durchführung der unterschiedlichsten Lookup-Aufgaben und auch zur Durchführung von Überwachungs- und Statistikfunktionen Daten vom Speicher gelesen und zum Teil auch wieder zurückgeschrieben werden. Beim Einsatz von Beschleunigermodulen müssen die erforderlichen Teile des Headers und die Ergebnisse bzw. sonstige Statusinformationen zwischen den beteiligten Architekturblöcken ausgetauscht werden. Da die Paketverarbeitung schritthaltend mit der ankommenden Paketrate erfolgen soll, ist eine möglichst hochperformante Implementierung erforderlich.

Nachfolgend soll deshalb der Aspekt der systeminternen Kommunikation näher beleuchtet werden. Der Kommunikationsaspekt kann während zweier Phasen des Entwurfsprozesses berücksichtigt werden:

- Berücksichtigung der Kommunikation bei der Abbildung der Funktion auf die Architektur
- Verfeinerung der Kommunikationsarchitektur nach der Abbildung

In [66] wird die Trennung zwischen den Aspekten Verarbeitung und Kommunikation beim Entwurfsprozeß innerhalb eines Systems gefordert. Die Motivation für den darin beschriebenen Ansatz der "Orthogonalization of Concerns" liegt in der Begrenzung der Komplexität und der Beschleunigung des Designablaufs, um eine effiziente Untersuchung von Designalternativen zu erlauben. Eine ähnliche, bereits seit langem praktizierte Trennung

gilt für die Bereiche Funktion und Architektur, wodurch die Korrektheit der zu realisierenden Funktionalität und die Art und Weise ihrer Implementierung unabhängig voneinander und ohne gegenseitige Beeinflussung untersucht werden können.
Die Abbildung der Funktion auf die Architektur ist jedoch der Designschritt, bei dem diese Trennung aufgehoben wird, die im weiteren auch nicht mehr wiederhergestellt wird. Im Gegensatz dazu kann die Trennung zwischen Verarbeitung und Kommunikation auch in den nach diesem Schritt folgenden Designphasen größtenteils aufrechterhalten werden. Die Verfeinerung der Kommunikationsarchitektur ist hierfür ein Beispiel, bei dem ohne Kenntnis der Implementierung der den einzelnen Architekturblöcken zugeordneten Teilfunktionen die systeminterne Kommunikation und ihre Parameter konkret festgelegt werden können.
Ähnlich wie bezüglich Funktion und Architektur ist beim Mapping der Teilfunktionen jedoch die entsprechende Trennung von Kommunikation und Verarbeitung nicht möglich, da bei dieser Entscheidung sich die Notwendigkeit der Kommunikation über die systeminterne Kommunikationsarchitektur erst ergibt. Werden zwei auf funktionaler Ebene miteinander kommunizierende Prozesse auf die gleiche Verarbeitungsressource abgebildet, entsteht in der Regel kein Kommunikationsbedarf (abgesehen von der Situation, bei der über gemeinsamen Speicher Daten ausgetauscht werden), während bei einer Abbildung auf unterschiedliche Ressourcen Datentransfers direkt (Message Passing) oder indirekt (gemeinsamer Speicher) Daten über die systeminternen Kommunikationsressourcen ausgetauscht werden müssen. Nachfolgend wird gezeigt, daß der infolge der Entscheidung über die Zuordnung der Teilfunktionen entstehende Kommunikationsbedarf großen Einfluß auf die resultierende Lösung haben kann, und deshalb eine gemeinsame Betrachtung dieses Aspekts notwendig ist.
Die hier dargestellte Sichtweise stellt jedoch keinen Widerspruch zu den in [66] vorgestellten Prinzipien dar. Die beiden orthogonal zu handhabenden Paare Funktion und Architektur bzw. Verarbeitung und Kommunikation sind in Bezug auf den Designablauf jeweils in ähnlicher Weise weitestgehend getrennt, andererseits aber teilweise auch gemeinsam zu betrachten, wenn beide Aspekte gemeinsam betroffen sind.

3.2.1 Verfeinerung der Kommunikation

Viele der in Kapitel 2 beschriebenen Ansätze für den Systementwurf gehen von einer bereits festgelegten Abbildung der einzelnen Teilfunktionen auf die Ressourcen der Architektur aus. Mit dieser Zuordnung der Prozesse auf die Architekturblöcke ist auch der systeminterne Kommunikationsbedarf definiert. Die Optimierungen beziehen sich deshalb nur auf die zeitlichen Abläufe der Transaktionen und die Festlegung der Parameter, die die Kommunikationsarchitektur besitzt. Darunter fallen insbesondere die physikalischen Parameter des Kommunikationskanals (Wortbreite und Taktrate), das verwendete Protokoll und die den einzelnen Architekturblöcken zugeordneten Kommunikationseigenschaften oder auch die Struktur der Kommunikationsarchitektur.
In [92] und [97] werden Methoden vorgestellt, wie die Konkretisierung der Kommunikationsarchitektur durchgeführt und die entsprechende Verfeinerung formalisiert werden kann. [97] beschreibt dabei einen Ansatz zur Definition eines On-Chip Netzwerks, bei dem die

Prinzipien der Datenkommunikation auf diesen Bereich übertragen werden. Dabei wird zur Systematisierung des Kommunikationsproblems auch für die Kommunikation innerhalb des Bausteins ein Schichtenkonzept analog zum OSI-Modell vorgeschlagen, mit dem Fragen wie die Auswahl physikalischer Eigenschaften der Kommunikationsarchitektur oder der Zugriff auf das Medium erfaßt werden. Dies beinhaltet auch die Arbitrierung sowie Blockierungen, die bei mehrfach genutzten Kommunikationsinfrastrukturen von besonderer Bedeutung sind. Dazu gehört zudem die Wahl geeigneter Übertragungsprotokolle, die indirekte Kommunikation, d.h. mehrere nacheinander durchlaufene Schaltungsteile innerhalb des Chips, erlauben. Auch im Designablauf von Cadence VCC ([11]) folgt der Aufteilung der Funktionalitäten die Verfeinerung der Kommunikation, bevor zur Implementierung übergegangen wird.
Die Arbeiten von Lahiri et al. ([71], [72]) beinhalten Untersuchungen zur Performance-Analyse und zur Optimierung von Kommunikationsarchitekturen. In [71] wird die aus einer Co-Simulation einer Bus-basierten SoC Architektur extrahierte Information über die Aktivitäten im System, insbesondere der Kommunikationsarchitektur, als Ausgangspunkt für deren Verfeinerung verwendet. Derartige Informationen über den systeminternen Abläufe werden oft als Traces bezeichnet. Das dort vorgestellte Verfahren ermöglicht unter Einbeziehung weiterer Details der Kommunikationsmedien eine schnelle und genaue Performance-Untersuchung des Systems, ohne erneut eine komplette Simulation des Systems durchführen zu müssen, und kann damit den Entwickler bei der Exploration unterschiedlicher Alternativen unterstützen. In der weiterführenden Veröffentlichung [72] wird die aus der Kommunikationsanalyse gewonnene Information verwendet, um die Transaktionen automatisch und optimiert auf eine vorgegebene und aus beliebigen dedizierten bzw. gemeinsam genutzten Kommunikationskanälen abzubilden. Neben der Zuordnung der Transfers wird dabei für jeden Kanal auch das zugehörige Protokoll generiert.
In [78] wird eine Methode vorgestellt, die ebenfalls derartige Traces verwendet, um eine Exploration der Kommunikationsarchitektur vorzunehmen. Sie basiert auf der in [77] beschriebenen SPADE Methode, bei der eine scharfe Trennung zwischen dem Modell der Anwendungs- und der architekturalen Ebene angenommen wird. Mit Hilfe der in [78] beschriebenen Transformationsmethoden kann ein von einer Applikation generierter Trace von der Anwendungs- auf die Architekturebene umgesetzt und mit den relevanten Details ergänzt werden. Dabei wird die Abfolge der einzelnen Teilschritte abhängig von der Architektur optimiert. Das Verfahren kann zum einen für die schnelle Untersuchung der Eignung unterschiedlicher Varianten von Kommunikationsarchitekturen eingesetzt oder als Bestandteil eines entsprechenden Explorationssystems verwendet werden.
Daneben werden in [73] sog. “Kommunikationsarchitektur-Tuner” vorgeschlagen, die intelligente Schnittstellen zum gemeinsam genutzten Bus darstellen. Durch dynamische Modifikation der Parameter des Kommunikationsprotokolls, abhängig vom jeweiligen Zustand des Systems, soll die Kommunikationsarchitektur besser ausgenutzt und an die jeweiligen Erfordernisse der einzelnen Architekturblöcke angepaßt werden können. Auch mit diesem Ansatz - in diesem Fall durch Maßnahmen, die zur Laufzeit wirksam sind - wird eine Verbesserung des Verhaltens der Kommunikationsarchitektur nach der Abbildung der Funktionen auf die Architektur angestrebt, um eine höhere Performance zu erzielen.
Ein Verfahren zur Optimierung des Scheduling sowohl der Verarbeitungsprozesse als auch

insbesondere der Transaktionen auf der Kommunikationsarchitektur wird in [31] vorgestellt. Der Algorithmus unterstützt Kontrollabhängigkeiten im Prozeß-Graphen und zielt darauf ab, die Schedule Dauer der Anwendung für den ungünstigsten Fall der möglichen Bedingungskombinationen zu minimieren. Zu diesem Zweck wird jeweils mit Hilfe eines angepaßten List Scheduling Verfahrens unter Berücksichtigung der einzelnen Kombinationen eine Schedule Tabelle generiert, die ein im System dezentral realisierter Scheduler nutzt, um zur Laufzeit abhängig von der tatsächlich vorliegenden Bedingung dynamisch über die Aktivierung der einzelnen Prozesse bzw. Transfers zu entscheiden. Die systeminterne Kommunikation wird bei diesem Algorithmus insbesondere auch in Bezug auf gemeinsam genutzte Ressourcen und die Parameter der zugehörigen Protokolle berücksichtigt. Dabei wird jedoch von einer bereits erfolgten Zuordnung der Prozesse auf die Architektur ausgegangen, die nicht mehr verändert wird.

Die beschriebenen Ansätze zielen jeweils darauf ab, durch eine geeignete Wahl von Parametern und die Festlegung von Architektureigenschaften eine möglichst an die zu erfüllenden Anforderungen angepaßte Lösung zu generieren, ohne jedoch den entscheidenden Schritt der Partitionierung der Funktionalität und ihren Einfluß auf die Lösung zu berücksichtigen.

3.2.2 Der Kommunikationsaspekt bei der Partitionierung

Bei der Partitionierung bzw. insbesondere der Abbildung der Funktion muß abgewogen werden, ob Vorteile durch alternative Zuordnungen von Teilfunktionen auf Ressourcen die dabei auftretenden Nachteile überkompensieren können. So ist beispielsweise zu entscheiden, ob die Implementierung einer bestimmten Teilaufgabe in einem speziellen, für diese Teilaufgabe optimierten und mit geringerer Ausführungsdauer verbundenen Beschleuniger-Modul den zusätzlichen Aufwand an HW oder auch den Zeitverlust durch die dabei erforderlichen Datentransfers im Vergleich zu einer SW-Implementierung und dem damit verbundenen Umfang an Maschinencode aufwiegt. Hierbei ist zu beachten, daß bei einer Implementierung mit einem dedizierten Beschleuniger-Modul der Gewinn bei der reinen Ausführungszeit schon durch den zusätzlichen Kommunikationsaufwand reduziert wird. Ziel ist es generell, eine Aufteilung zu finden, die im Hinblick auf bestimmte Kriterien möglichst optimal ist, zumindest aber vorgegebene Designziele erreicht. Darunter fallen insbesondere Performance im Sinne von Durchsatz bzw. Latenz, Flächenverbrauch, Leistungsaufnahme etc.
In Bezug auf die Kommunikation bedeutet dies, daß Performance insbesondere durch zusätzliche interne Transfers verlorengehen kann, wenn nicht durch beschleunigte Verarbeitung der transferierten Daten dieser Effekt überkompensiert werden kann. Hierbei ist jedoch eine genaue Betrachtung der Abläufe auf dem Kommunikationsmedium erforderlich. Das nachfolgende Beispiel in Bild 3-2 demonstriert für einen gemeinsamen, die einzelnen Verarbeitungseinheiten verbindenden Bus, daß dynamische Effekte, die erst mit der Erstellung des Schedules ersichtlich sind, berücksichtigt werden müssen.
Es bestehe die Aufgabe, einen Graphen mit 4 Knoten auf eine busbasierte Architektur abzubilden, die aus drei Verarbeitungsblöcken *A*, *B* und *C* aufgebaut werden kann. Für die

Teilfunktion 2 bestehe die Alternative, sie auf dem Prozessor *A* oder auf dem zur Beschleunigung zusätzlich einzusetzenden HW Block *B* zu realisieren. Die anderen Teilfunktionen seien nur auf dem Prozessor *A* bzw. auf den Beschleuniger *C* implementierbar. Im Bild sind rechts oben die Ausführungszeiten bzw. die reinen Transferzeiten zwischen den einzelnen Teilfunktionen angegeben, falls ein Transfer auf dem Bus notwendig ist.

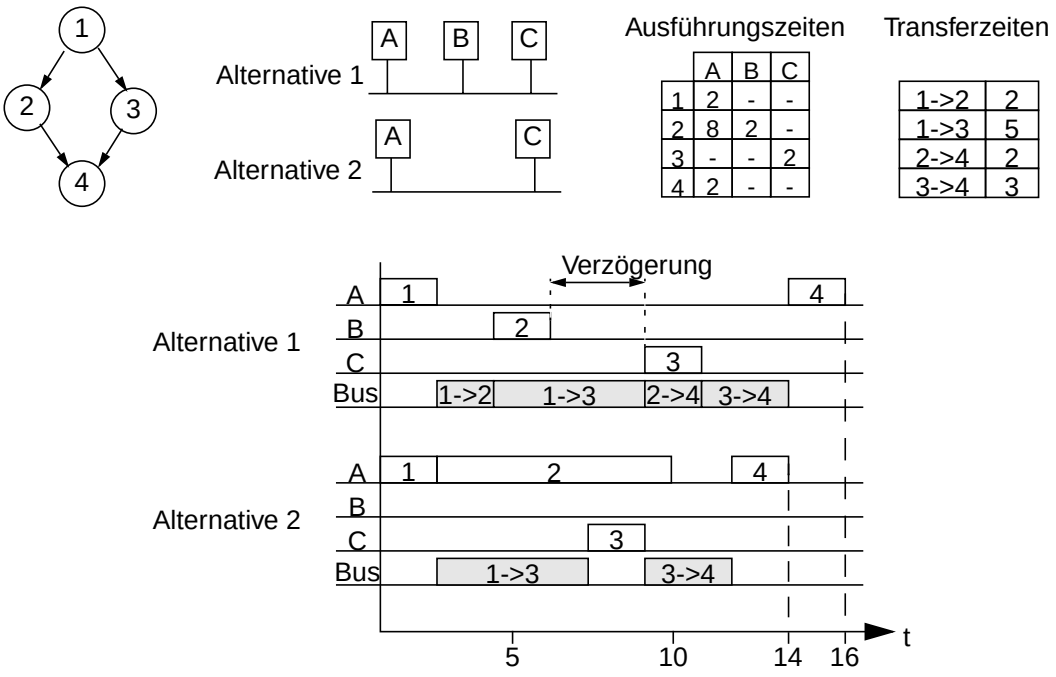


Bild 3-2: Auswirkung dynamischer Effekte auf die erreichbare Performance

Die beiden gezeigten Alternativen führen zu unterschiedlichen Schedule Dauern der Anwendung. Obwohl für die Variante mit dem Beschleuniger *B* nominell eine höhere Performance zu erwarten wäre, da trotz der beiden Transfers die Summe aus Verarbeitungs- und Transferzeiten kleiner als die Verarbeitungszeit der Teilfunktion 2 auf dem Prozessor *A* ist, trifft dies tatsächlich nicht zu. (Anmerkung: Auch eine Änderung der Reihenfolge, d.h. Vorziehen von Prozeß 3 und der zugehörigen Kommunikation, führt zum gleichen Resultat.)

Die Ursache hierfür liegt in der Überlastung des Busses während der Transfers zu den Beschleuniger bzw. zum Prozessor zurück. Als Folge treten Verzögerungen auf, die sich wie eine verlängerte Bearbeitungszeit auswirken. Um derartige Effekte zu vermeiden, sollte den Vorgängen auf der Kommunikationsarchitektur deshalb bereits in der Phase der Zuordnung der Teilfunktionen auf die Verarbeitungsressourcen besondere Aufmerksamkeit gewidmet werden. In den bisherigen Verfahren zur Abbildung der Funktionen auf die Architektur wurde der Kommunikationsaspekt jedoch nicht bzw. nur sehr rudimentär in Bezug auf auftretende dynamische Effekte berücksichtigt.

Im folgenden soll deshalb ein kurzer Überblick auf die Einbeziehung der Kommunikation bei der Abbildung der Teilfunktionen gegeben werden. Durch entsprechende Clustering-

Verfahren wurden zum einen Teilfunktionen, die intensiven Datenaustausch durchführen, zusammengefaßt und gemeinsam auf eine Verarbeitungseinheit abgebildet, um einen Datentransfer zwischen verschiedenen Blöcken zu vermeiden ([30], [7]). Diesem Ansatz liegt die Annahme zugrunde, daß der lokale Datenaustausch zwischen Teilfunktionen, die entweder in HW oder SW implementiert sind, über lokale Register effizient durchgeführt werden kann. Dieses Vorgehen entspricht im wesentlichen der Vermeidung von Kommunikation über die Kommunikationsressourcen innerhalb des Bausteins.
Eine weitere Vorgehensweise zur Berücksichtigung der On-Chip Kommunikation ist daneben, die Zusammenfassung von Architekturblöcken mit intensiven Datenaustausch und deren Verbindung über eigene Busse, die wiederum über Brücken miteinander verbunden sind. Auf diese Weise soll gewährleistet werden, daß die Kommunikationsaktivitäten auf getrennten Bussen sich nicht gegenseitig überlagern und so insgesamt eine höhere Performance erreicht werden kann. Diese Vorgehensweise führt jedoch, wie in [15] gezeigt, nicht unbedingt zu einem optimalen Ergebnis.
In [63] wurde eine detaillierte Analyse für einen Kommunikationskanal zwischen einem Prozessor und einem HW Beschleunigermodul dargestellt. Im Mittelpunkt standen dabei der Zugriff von HW- bzw. SW Modulen auf einen Bus sowie die Eigenschaften des Busses. Die bei diesen Arbeiten verfolgte Zielsetzung war eine Abschätzung des infolge von interner Kommunikation erforderlichen Aufwands an Code bzw. Chipfläche für Treiber, um dies im Designsystem LYCOS (siehe [79]) zu berücksichtigen. Bei dieser Vorgehensweise wurde ebenfalls die Generierung der Kommunikationsarchitektur in die Schritte der Architektur-Exploration und der System Partitionierung miteinbezogen. Mit Hilfe des Kommunikationsmodells sollte eine schnelle Abschätzung des Kommunikationsaufwandes zur Einbeziehung in die Partitionierungsentscheidung verfolgt werden. Die Motivation hierbei lag jedoch in der Berücksichtigung des Implementierungsaufwands für die interne Kommunikation und nicht in der Optimierung der Performance.
Bei der Partitionierung für einfache Prozessor-Koprozessor Architekturen in [48] und [33] wurde die Kommunikation jeweils als eine Teilkomponente der Kostenfunktionen berücksichtigt. Dabei wurden jedoch - auch aufgrund der eingeschränkten Architektur - lediglich einfache Schätzungen über die für den Datenaustausch benötigten Transferzeiten miteinbezogen. Da kein gemeinsames Kommunikationsmedium angenommen wurde, ist das oben gezeigte Problem dort nicht relevant.
Zum Scheduling für Multiprozessorarchitekturen wurden verschiedene Verfahren vorgeschlagen, die die Berücksichtigung des Kommunikationsaspekts beinhalten. Hierbei handelt es sich jedoch meist um sehr homogene, in Bezug auf die Ressourcen insbesondere der Kommunikation kaum eingeschränkte Architekturen, bei denen volle Konnektivität zwischen identischen Prozessoren angenommen wurde ([91], [122]). Die vorliegende Problematik für eine kontrolldominierte Anwendung auf einem System-on-Chip kann damit nicht abgedeckt werden.
Das in [123] beschriebene Verfahren von Xie und Wolf berücksichtigt die Kommunikation bei der Abbildung und dem Scheduling nur eingeschränkt. So werden Transfers nicht mit der tatsächlichen Übertragungszeit sondern nur mit einer symbolischen Standard-Verzögerung berücksichtigt, und zudem werden Transfers zu HW-Beschleunigern immer bevorzugt. Aufgrund der Auswahl des jeweils abzubildenden Prozesses anhand des

Maximalwerts einer dynamisch bestimmten Priorität werden Festlegungen getroffen, die insbesondere Überlastzustände von Ressourcen aufgrund vorher getroffener Entscheidungen nicht vermeiden lassen. Für das oben gezeigte Beispiel, in dem aufgrund der Überlastung des gemeinsam genutzten Busses Verzögerungen entstehen können, würde auch die Anwendung des Verfahrens von Xie und Wolf, das bedingte Verzweigungen erfassen kann und damit für die Anwendung im Bereich Paketverarbeitung geeignet ist, zu dieser suboptimalen Lösung führen.
Eine intensivere Betrachtung der Kommunikation während der Festlegung der Abbildung der einzelnen Teilfunktionen auf die Architektur ist deshalb von besonderer Bedeutung. Beim Verfahren von Xie/Wolf wird die Lösung sukzessive konstruiert, und dabei die Zuordnung der einzelnen Teilfunktionen nur unter Kenntnis eines Teils des Schedules durchgeführt, ohne daß diese Entscheidungen nachfolgend revidiert werden können. Deshalb wird in der vorliegenden Arbeit ein Verfahren angestrebt, das sich iterativ einer möglichst optimalen Lösung annähert. Ein iteratives Verfahren, das gleichzeitig auch bedingte Übergänge im Graphen unterstützt ist bisher nicht bekannt.

Daraus ergibt sich als Motivation für die Arbeit die nachfolgend beschriebene Zielsetzung. Danach werden die für die Untersuchung getroffenen Randbedingungen beschrieben.

3.3 Zielsetzung

Für das oben geschilderte Anwendungsgebiet Paketverarbeitung soll ein Verfahren zur Abbildung der Teilfunktionen auf eine Systemarchitektur entwickelt werden, das zum einen die notwendige Voraussetzung der Unterstützung von Kontrollabhängigkeiten erfüllt und zum anderen den Kommunikationsaspekt Rechnung trägt, um Überlastungen auf gemeinsam genutzten Ressourcen, die zu Performanceengpässen führen können, zu vermeiden. Da Verfahren, die die Lösung schrittweise konstruieren, die oben beschriebenen Nachteile besitzen, soll eine Methode angewendet werden, die die Lösung rekursiv findet und iterativ entsprechende Engpässe beseitigt. Hierzu wird ein auf dem Prinzip der lokalen Suche basierendes Verfahren vorgeschlagen. Von dieser Klasse von Algorithmen ist bisher keiner bekannt, der Funktionalitäten mit bedingten Übergängen unterstützt.
Parallel zur vorliegenden Dissertation wird eine Arbeit zur Verbesserung konstruktiver Verfahren angefertigt, die auf einer erweiterten Berücksichtigung von internen Datentransfers beruht. Teilergebnisse hierzu sind in [119] enthalten.
Zunächst sollen nachfolgend die Einordnung in einen möglichen Ablauf zum Entwurf auf der Systemebene und die für die Definition des Verfahrens getroffenen Randbedingungen erläutert werden, bevor im Kapitel 4 das Verfahren selbst dargestellt wird.

3.3.1 Einordnung in einen Ablauf zum Entwurf auf der Systemebene

Das nachfolgende Bild 3-3 zeigt, wie sich die vorgeschlagene Methode in die auf der linken Seite angedeuteten Abfolge zum Systementwurf einordnet. Zur Festlegung der Architektur wird eine Abstraktion des die Funktion voll erfassenden und als ausführbare Spezifikation vorliegenden Systemmodells zu einem abstrakten Graphenmodell vorgenommen. Dies

führt dazu, daß nicht mehr die komplette Funktionalität im Modell sondern lediglich die Struktur und die Interaktion der einzelnen Teilfunktionen enthalten sind.
Für diesen Vorgang der Abstraktion zu einem Graphenmodell sind Werkzeuge vorhanden, die aus einer funktionalen Beschreibung einen Prozeßgraphen extrahieren können. In [113] ist ein derartiges Verfahren beschrieben, das aus einem in C beschriebenen Systemmodell einen Graphen generiert, der für die Untersuchungen im Rahmen der Architekturexploration eingesetzt werden kann. Für diese Umsetzung in eine entsprechend abstrakte Form ist auch der Einsatz der SPI ([18]) Umgebung denkbar (vgl Kapitel 2.1.3.3). Durch diese Vorgehensweise ist für ein verifiziertes Systemmodell auch die Korrektheit des Graphen gewährleistet.

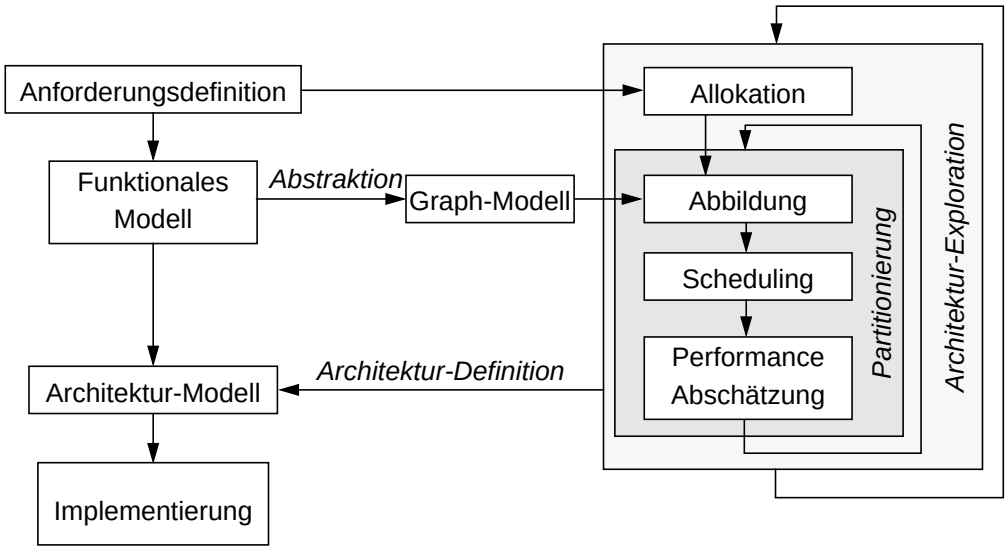


Bild 3-3: Einordnung des Verfahrens in den Entwurf auf der Systemebene

Das Verfahren zur Festlegung der Abbildung und des Scheduling wird im Rahmen der Architekturexploration eingesetzt, die für die zu implementierende Funktion eine geeignete Architektur auf abstrakter Ebene festlegt und die entsprechende Vorgaben zur Erstellung des konkreten architekturalen Modells liefert. Dieses architekturale Modell dient als Basis für die weiteren Schritte hin zur Implementierung mit den zugehörigen Konkretisierungen und der weiteren Einengung des Implementierungsraums.

3.3.2 Randbedingungen

Für die Definition des rekursiven Verfahrens zur Abbildung und zum Scheduling wurden folgende Annahmen getroffen:

a) Optimierung nach Performance
   Als Optimierungsziel wird ausschließlich die Performance des Systems betrachtet. Das Verfahren soll unter den oben genannten Vorgaben ausschließlich die Ablaufzeit des Systems (Schedule Dauer) auf einer zugrundgelegten Zielarchitektur

minimieren. Die Architekturexploration mit der Auswahl geeigneter Architekturen und deren Bewertung sind nicht Gegenstand der vorliegenden Arbeit.

b) Scheduling
   Es wird angenommen, daß die Abfolge der einzelnen Prozesse auf den Ressourcen der Architektur statisch, d.h. zur Entwurfszeit, festgelegt und nicht verändert wird. Die mit dem Verfahren erzeugten Schedules decken den ungünstigsten Fall ab, der sich aufgrund unterschiedlicher Kombinationen von Bedingungswerten ergeben kann. Durch das statische Scheduling wird zum einen der Aufwand bei der Implementierung begrenzt, zum anderen ist gewährleistet, daß sich die Implementierung deterministisch verhält und der Worst Case eingehalten wird.

c) Zugrundeliegende Architektur
   Als Zielarchitektur wird dem Verfahren eine Architektur zugrundegelegt, die aus parallel arbeitenden Verarbeitungseinheiten besteht, die mit einem gemeinsam genutzten Bus verbunden sind. Aufgrund der Anwendung für den Bereich von System-on-Chip wird angenommen, daß die Ressourcen unterschiedliche Eigenschaften besitzen können, wodurch insbesondere die Unterstützung von HW-Beschleunigern mit einem beschränkten Funktionsumfang möglich wird. Diese und auch die Beschränkung auf einen gemeinsamen Bus unterscheidet die betrachtete Klasse von Architekturen von Multiprozessorarchitekturen, für die oft volle Konnektivität von identischen Prozessorelementen angenommen wird.
   Andererseits soll die Architektur nicht voll reprogrammierbar im Sinne einer allgemein einsetzbaren (Netzwerkprozessor-)Architektur sein, sondern eine für eine einzelne Anwendung erstellte Plattform, die durch den Einsatz von eingebetteten Prozessoren zur Verarbeitung von Aufgaben des Fast Path in Kommunikationsanwendungen in ihrer Funktionalität geändert werden kann.

d) Zu implementierende Funktion
   Als Anwendungsgebiet wird von der Verarbeitung von Datenpaketen ausgegangen. Die Kontrollabhängigkeit resultiert aus der Abfolge der im Header des Pakets beinhalteten Kommunikationsprotokolle. Als Funktion wird deshalb die Bearbeitung ausschließlich des Headers des Pakets betrachtet.
   Für die Untersuchung wird weiterhin angenommen, daß nur eine Instanz des Graphen aktiv ist, d.h. es erfolgt nur die Verarbeitung eines Pakets, es können also keine Überlagerungen von unterschiedlichen Verarbeitungsinstanzen auftreten.

Die getroffenen Einschränkungen erlauben es, das Verfahren auf die oben genannten Hauptpunkte der Unterstützung von Kontrollabhängigkeiten und der besonderen Berücksichtigung der Kommunikation zu konzentrieren.
Im nachfolgenden Kapitel wird auf die Definition des entsprechenden Verfahrens näher eingegangen.

4 Die rekursive Abbildungs- und Scheduling-Methode

4.1 Überblick

In diesem Kapitel wird das neue rekursive, auf lokaler Suche basierende Verfahren zum Abbilden und Scheduling eines Prozeß-Graphen mit Kontrollabhängigkeiten beschrieben. Ausgehend von einer Erläuterung der dabei angenommenen Randbedingungen wird insbesondere darauf eingegangen, auf welche Weise bedingte Knoten behandelt werden, und wie die Suche durch eine geeignete Wahl der Nachbarschaft in Richtung der besten Lösung gelenkt werden kann.
Das Verfahren führt die Optimierung ausschließlich im Hinblick auf die Performance der Lösung, d.h. einer möglichst kurzen Ausführungsdauer, durch. Es basiert auf dem Prinzip der lokalen Suche in einer um die aktuelle Lösung definierten Nachbarschaft. Diese Nachbarschaft steht in einer bestimmten Beziehung zur jeweils aktuellen Lösung und erlaubt es, den Lösungsraum effizient und mit Zielrichtung auf das absolute Minimum der Bewertungsgröße abzusuchen. Der Definition einer geeigneten Nachbarschaft, die zum einen verbesserte Lösungen liefert und zum anderen jedoch auch den Aufwand und Zeitbedarf der Lösungsfindung begrenzt, ist bei derartigen Verfahren besondere Beachtung zu schenken. Auf diesen Aspekt wird weiter unten in diesem Kapitel und nachfolgend bei der Diskussion der mit dem Verfahren bei unterschiedlichen Nachbarschaftsdefinitionen erreichbaren Ergebnisse näher eingegangen.
In diesem Verfahren findet die Berücksichtigung von Resssourcenkonflikten besondere Beachtung. Insbesondere die im Kapitel 3.2.2 beschriebenen Effekte im Zusammenhang mit der systeminternen Kommunikation stellen einen Schwerpunkt dar. Diese Informationen über dynamische Effekte auf gemeinsam genutzten Ressourcen fließen in die Definition der Nachbarschaft in geeigneter Weise ein, was eine Verbesserung der Lösungsgüte ermöglicht.

4.2 Ausgangspunkte

Vor der Beschreibung des Verfahrens sollen zunächst die dabei gewählten Randbedingungen dargestellt werden. Die Entwicklung des Verfahrens wurde insbesondere mit Blick auf Anwendungen für die Datenkommunikation durchgeführt. In diesem Bereich müssen als Grundvoraussetzung einerseits Kontrollabhängigkeiten, andererseits aber zur Erreichung eines hohen Durchsatzes vor allem eine möglichst große Verarbeitungsleistung bereitgestellt werden. Auch wenn andere in der Literatur veröffentlichte Methoden ([22], [24]) die gleichzeitige Optimierung mehrerer Parameter wie Performance der Lösung, Fläche bzw. Produktkosten oder zunehmend auch die Verlustleistung einbeziehen, beschränkt sich das vorgestellte Verfahren deshalb auf die Optimierung der Performance, d.h. der Suche nach einer Lösung mit möglichst kurzer Ausführungsdauer, nachfolgend auch als Schedule Dauer bezeichnet.

Das Verfahren generiert eine Lösung, die den Worst Case erfaßt, der infolge der Kontrollabhängigkeiten entstehen kann. Die sich für die resultierende Abbildung der Funktion auf die Architektur ergebende Schedule Dauer entspricht damit derjenigen Kombination von Bedingungen, zu deren Abarbeitung die meiste Zeit benötigt wird. Mit dem Ergebnis des Algorithmus ist deshalb gewährleistet, daß die zu implementierende Funktion in jedem Fall maximal in der angegebenen Zeit abgearbeitet ist. Dies wird nachfolgend im Zusammenhang mit dem Scheduling näher erläutert.

4.2.1 Modellierung der Funktion

Ausgangspunkt für das Verfahren ist die Verfügbarkeit der auf die Zielarchitektur abzubildenden Funktion in Form eines Graphenmodells, das deren Eigenschaften in einem auf der Systemebene ausreichenden Detaillierungsgrad erfaßt. D.h. die in einem ggf. als ausführbare Spezifikation vorhandenen Systemmodell enthaltenen Detailinformationen wie z.B. zu konkreten Algorithmen und Datenstrukturen werden für die Untersuchungen zum Mapping und Scheduling nicht benötigt und können auf dieser Ebene deshalb vernachlässigt werden.
Nach dieser Abstraktion werden lediglich Informationen zur Unterteilung der gesamten Funktionalität in einzelne Teilfunktionen oder Prozesse (nachfolgend auch als Task bezeichnet) sowie ihre Abfolge und der zugehörige Datenaustausch untereinander benötigt. Für die Abbildung und das Scheduling sind selbst keine funktional ausführbaren Modelle notwendig, die die Teilfunktionen beschreiben, sondern lediglich ein abstraktes funktionales Modell und eine Bibliothek architekturbezogener Daten wie Ausführungszeiten, Flächen- bzw. Speicherbedarf oder auch Verlustleistung, die als Basis für die Bewertung von Realisierungsalternativen dienen. Diese Informationen werden durch Performanceabschätzung auf der Ebene der einzelnen Prozesse geliefert, die nicht Gegenstand der vorliegenden Arbeit ist. Die jeweils angewandten Methoden, die teilweise detaillierte Informationen bzw. ausführbare Modelle der Teilfunktionen benötigen, sind in Kapitel 2.2.2 aufgezeigt. Die Genauigkeit der mit diesen Verfahren gewonnenen Daten hat jedoch entscheidenden Einfluß auf die Untersuchungen zum Mapping und Scheduling. Da das hier entwickelte Verfahren ausschließlich hinsichtlich der erreichbaren Performance im Sinne von Ausführungszeit der Funktion auf der betrachteten Architektur optimiert, wird davon ausgegangen, daß die Worst Case Verarbeitungsdauern der einzelnen Prozesse auf den jeweiligen Ressourcen vorhanden sind. Weitere ressourcenbezogene Informationen sind nicht erforderlich.
Als weitere Voraussetzung für die Abbildung und das Scheduling müssen innerhalb des funktionalen Modells die für das gewählte Anwendungsgebiet charakteristischen Kontrollabhängigkeiten erfaßbar sein. Zunächst wird deshalb nachfolgend die Festlegung des als Grundlage des Verfahrens genutzten abstrakten Notationsmodells beschrieben, das die genannte Eigenschaft besitzt.
In [38] ist eine ausführliche Übersicht von Methoden zur Modellierung eines Systems angegeben. Die dort vorgenommene Einteilung differenziert in zustands-, aktivitäts-, struktur- oder datenorientierte bzw. heterogene Ansätze. Zur Nachbildung des Systems auf funktionaler Ebene sind jedoch strukturale Modelle und datenbezogene Methoden nicht ge-

eignet, da hierbei die Funktion und insbesondere die Kontrollstrukturen nur implizit im Modell enthalten sind. Über die Modellierung als Zustandsmodell können die Kontrollabhängigkeiten des Systems zwar sehr gut nachgebildet werden, es besitzt jedoch den Nachteil, daß hierbei der Datenfluß nur schwer erfaßt werden kann, und derartige Modelle mit zunehmender Größe sehr schnell unübersichtlich werden.
Der in vielen Anwendungsbereichen eingesetzte Ansatz der Repräsentation in Form eines Datenflußgraphen (DFG, Data Flow Graph) kommt jedoch auch mangels Berücksichtigung von Kontrollstrukturen nicht in Frage. Diese Spezifikationsmethode ist jedoch wegen ihrer Anschaulichkeit in der Abbildung der Funktion auf einzelne miteinander verbundene Teilfunktionen oder Prozesse und wegen der expliziten Erfassung des Datenaustauschs sehr interessant. Ähnliche Nachteile, jedoch im reziproken Sinn besitzen reine Kontrollflußgraphen.
Der beschriebenen Anforderung kommt die Modellierung der Funktionalität in Form eines CDFG (Control/Data Flow Graph) noch am nächsten. Ein CDFG erweitert die rein datenflußorientierte Modellierung eines DFG (Data Flow Graph) um die Berücksichtigung von Kontrollstrukturen wie Fallunterscheidungen oder Schleifenkonstrukte. Die Modellierung von Schleifen würde dabei im Normalfall zu zyklischen Graphen führen.
Im betrachteten Anwendungsgebiet der Paketverarbeitung treten auf der hier relevanten Abstraktionsebene üblicherweise keine Schleifen mit einer unbestimmten Anzahl von Durchläufen auf. In manchen Teilfunktionen sind jedoch vereinzelt Verarbeitungsschritte enthalten, die wiederholt ausgeführt werden müssen, wobei sich erst zur Laufzeit entscheidet, in welcher konkreten Anzahl dies der Fall ist. Es existiert jedoch immer eine obere Schranke für Zahl der Durchläufe. Da die Zielsetzung des Verfahrens in der Optimierung der Performance für den ungünstigsten Fall liegt, kann die Problematik von Schleifen durch Aufrollen unter Berücksichtigung der maximalen Anzahl der Durchläufe gelöst werden. (Ein Beispiel für eine derartige Funktion ist der im Verifikationsbeispiel in Kapitel 5 verwendete Next-Hop Lookup Algorithmus.) Es wird deshalb angenommen, daß die Notation der Funktion in Form eines nicht-zyklischen Graphen erfolgt.
Obwohl durch die Darstellung des Systemmodells in Form eines CDFG die beiden Aspekte Datenfluß und Kontrollabhängigkeiten berücksichtigt werden können, beinhalten derartige Modelle mehr Information als für die hier geplante Verwendung nötig ist. Die Information des in einzelnen Knoten auszuführenden Codes ist auf der hier relevanten Abstraktionsebene nicht erforderlich. Ziel ist vielmehr die Verwendung einer einfachen, um alle für die Partitionierungsschritte irrelevanten Details reduzierte Datenstruktur, die modular erweiterbar und leicht annotierbar ist. Die Informationen die an die Elemente des Modells annotiert werden müssen, sind zum einen zur Funktion gehörende Informationen wie ausgetauschte Datenmengen und insbesondere die Abfolge der Teilfunktionen, daneben aber auch die architekturspezifischen Verzögerungen und Ausführungsdauern.
Als einfache und effiziente Notation, die diese Anforderungen auf der benötigten Abstraktionsebene erfüllt, wird für diesen Zweck die in [32] beschriebene graphische Repräsentation in Form eines CPG (Conditional Process Graph) verwendet. Ein CPG ist ein gerichteter, azyklischer, polarer Graph *G(V, E)*, bei dem die Knoten $P_i \in V$ einzelnen Teilfunktionen entsprechen, die in ihrer Gesamtheit die zu implementierende Funktion nach-

bilden. Die Teilfunktionen werden nachfolgend auch als Tasks oder Prozesse bezeichnet. Polarität in diesem Zusammenhang bedeutet, daß der Graph jeweils einen Start- und Endknoten besitzt. Diese Eigenschaft entspricht den Randbedingungen betrachteten Anwendungsgebiet Networking, wo der Beginn des Graphen der Ankunft eines zu bearbeitenden Pakets entspricht, während der Endknoten das Speichern des Pakets in einer Warteschlange oder seine Weiterleitung darstellt. Im übrigen ist es durch Einfügen von Knoten am Anfang bzw. am Ende leicht möglich, diese Eigenschaft auch für andere Graphen zu erzwingen, ohne daß damit Einschränkungen verbunden sind.

Die Knoten des Graphen sind über gerichtete Kanten $e_{ij} \in E$ verbunden. Eine Kante e_{ij} verbindet den Knoten P_i mit dem Knoten P_j und ist mit einer Datenmenge annotiert, die zur Abarbeitung des Prozesses *j* vom Prozeß *i* bereitgestellt werden muß. Konkret bedeutet dies, daß nach dem Ende der Bearbeitung des Prozesses *i* die entsprechende Datenmenge zum Prozeß *j* transportiert sein muß, bevor dieser mit der Verarbeitung beginnen kann. Die Kanten dienen damit auch der Festlegung der Abhängigkeiten der Teilfunktionen untereinander.

Neben der transferierten Datenmenge kann einer Kante auch eine Bedingung C_k zugeordnet sein, die erfüllt sein muß, damit der zugehörige Übergang auch tatsächlich ausgeführt wird. Alle derartigen Kanten ec_{ij}, die mit einer Bedingung versehen sind, werden in der Menge $E_c \subseteq E$ zusammengefaßt. Unbedingte Kanten es_{ij} seien in der Menge E_s enthalten, so daß gilt $E_c \cap E_s = \emptyset, E_c \cup E_s = E$.

Unter einer Bedingung soll im Weiteren die Aufspaltung der weiteren Verarbeitung anhand eines gemeinsamen Kriteriums verstanden werden, analog zur case-Anweisung oder dem if-else-if Konstrukt in C-Programmen. Eine Bedingung bezieht sich üblicherweise auf das Ergebnis einer Berechnung im Prozeß P_i oder dem Inhalt von Daten, die in P_i zugreifbar sind. Ein bedingter Übergang ec_{ij} ist also immer im Zusammenhang mit dem zugehörigen Knoten P_i zu sehen, von dem er ausgeht.

Zur besseren Umsetzbarkeit des CPG Modells im Rechnerprogramm für das Verfahren wird folgende Vereinfachung angenommen: pro Knoten P_i kann nur eine Bedingung C_k auftreten, d.h. der Ablauf nach Ende der Abarbeitung von P_i wird nur von einer auf einem gemeinsamen Kriterium basierenden Entscheidung beeinflußt. Sind für bestimmte Aufgaben mehrere Bedingungen abzuarbeiten, ist eine Unterteilung in mehrere einzelne Prozesse erforderlich, denen jeweils eine Bedingung zugordnet wird. Bei der Generierung des Graphen-Modells ist auf diese Randbedingung zu achten. Diese Einschränkung ist jedoch nicht prinzipieller Natur, sondern stellt lediglich eine Erleichterung für die Notation dar.

Weiterhin soll gelten, daß nur für einen der zu diesem Kriterium gehörenden Zweige der Wert der Bedingung zutrifft, so daß sich die zum Kriterium C_k gehörenden Pfade ec_{ij} gegenseitig ausschließen. D.h. von allen bedingten Kanten, die von einem Knoten abgehen, kann immer nur eine tatsächlich auch aktiv werden. Unabhängig von bedingten Übergängen können von einem Knoten aber beliebig viele unbedingte Kanten abgehen, von denen stets alle aktiv werden.

Mit Hilfe der bedingten Übergänge ist es möglich, die im Hinblick auf das Anwendungs-

gebiet erforderlichen Kontrollabhängigkeiten effizient zu modellieren. Bild 4-1 zeigt ein Beispiel eines einfachen CPG.

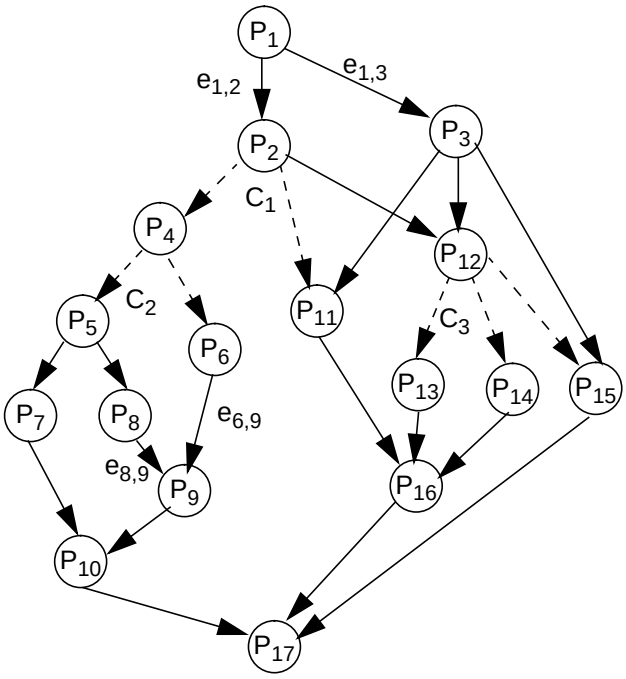


Bild 4-1: Beispiel eines CPG

Umgekehrt zu Knoten, bei denen sich der Kontrollfluß aufgrund mehrerer abgehender Kanten aufspaltet, vereinigen sich unterschiedliche Verarbeitungspfade an Knoten P_i, die mehr als eine ankommende Kante besitzen. In einem konventionellen Datenflußgraphen müssen alle an einem Knoten P_i ankommenden Kanten aktiv geworden sein, damit P_i auch selbst aktiviert werden kann. In einem CPG ist jedoch aufgrund der möglichen bedingten Verzweigungen zu berücksichtigen, daß als Voraussetzung für die Bearbeitung von P_i nur die zur jeweiligen Bedingungskombination gehörenden Kanten zutreffen müssen bevor P_i abgearbeitet werden kann. Im Beispielgraphen in Bild 4-1 bedeutet dies, daß der Knoten P_9 nur auf $e_{6,9}$, nicht aber auf $e_{8,9}$ warten muß, wenn aufgrund von C_2 die Abarbeitung nach P_4 mit P_6 fortgesetzt wurde. (Siehe hierzu auch Kapitel 4.5.2.)

Erzeugung des funktionalen Modells

Die Verfügbarkeit der Funktion in Form eines CPG ist eine Voraussetzung, auf die das rekursive Verfahren zum Mapping und Scheduling der Funktion auf eine Zielarchitektur aufsetzt. Bei der Generierung des Modells sind einige Randbedingungen zu beachten, um die Erfassung der Effekte zu ermöglichen, die bei den Untersuchungen auf der jeweiligen Kommunikationsarchitektur auftreten können.

Die Festlegung der Granularität ist hierbei ein entscheidender Faktor. Zusätzlich zu den in Kapitel 2.2.1 dargestellten grundsätzlichen Überlegungen hinsichtlich der Wahl geeigneter

Prozeßgrößen sind, wie nachfolgend beschrieben, gewisse Eigenschaften der Architektur und ihrer Verwendung von Bedeutung. Grundsätzlich sollte die funktionale Modellierung im Prinzip unabhängig von architekturalen Festlegungen sein, jedoch ist eine vollständig unabhängige Betrachtung beider Aspekte nicht möglich. Diese Problematik wird deshalb zunächst näher betrachtet.
Da eine Zielsetzung des Verfahrens die verstärkte Erfassung der systeminternen Kommunikation ist, muß dem Zusammenhang zwischen Kommunikationsarchitektur und den Kanten des Graphen besondere Aufmerksamkeit gewidmet werden. Sollen alle Transfers erfaßbar sein, muß bei der Erstellung des Graphenmodells dafür gesorgt werden, daß zugehörige Kanten im Graph vorhanden sind, was zusätzliche Vorgaben zur Unterteilung der Gesamtfunktion in einzelne Prozesse liefert.
Bei der Erstellung des Modells müssen deshalb weitere über funktionale Aspekte hinausgehende Kriterien beachtet werden. Dies betrifft insbesondere die Berücksichtigung des Speichermodells, das für die Architektur und die darauf abzubildende Anwendung geplant ist. D.h. der Ort und die Art und Weise wie Daten gespeichert werden, die die einzelnen Teilfunktionen während der Verarbeitung benötigen, muß bei der Modellbildung beachtet werden, da die Datenhaltung intensiven Einfluß auf Kommunikationsbedarf hat.
Grundsätzlich können folgende Arten der Datenhaltung bzw. des Datenaustauschs unterschieden werden

- rein lokale Daten
- Message Passing
- gemeinsamer Speicher

Der erstgenannte Fall beschreibt Daten, die lokal in der Verarbeitungseinheit, auf die ein Prozeß abgebildet wird, gehalten und ausschließlich dort verwendet werden. Dieser Fall besitzt keine Relevanz in Bezug auf die Kommunikation auf der Systemebene.
Besteht die Notwendigkeit der Aufteilung der Funktionalität in mehrere Prozesse, die mit den gleichen Daten arbeiten müssen, die jedoch ebenfalls lokal in der entsprechenden Verarbeitungsressourcen gehalten werden sollen, so müssen die Daten über Message Passing ausgetauscht werden. Dies bedeutet daß zwischen den zugehörigen Knoten im Graphen entsprechende Kanten vorhanden sein müssen.
Benutzen derartige Prozesse jedoch Daten, die in einem gemeinsamen Speicher außerhalb der Verarbeitungsressourcen liegen sollen, so ist dies bei der Erstellung des Modells zu berücksichtigen. Das folgende Beispiel in Bild 4-2 für einen Lesezugriff soll dies verdeutlichen.
Soll in einem Prozeß P_s auf einen Wert x, der im gemeinsamen Speicher liegt, lesend zugegriffen werden, muß an dieser Stelle eine Unterteilung von P_s, hier in P_i und P_k, vorgenommen werden. Der Lesezugriff selbst ist als eigener Prozeß P_j zu modellieren, der zum einen über eine Kante mit P_i verbunden ist, die der Übertragung der Adresse an den Speicher entspricht. Der Transfer der gelesenen Daten wird mit einer Kante zwischen P_i und P_k nachgebildet. Zusätzlich zu diesen Übergängen muß der in P_k für die weitere Bearbeitung der restlichen Teilfunktion von P_s benötigte Kontext durch eine entsprechende Kante von

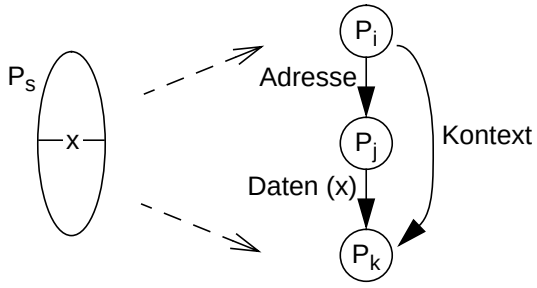


Bild 4-2: Modellierung eines Lesezugriffs auf den gemeinsamen Speicher

P_i übertragen werden. In Bezug auf die Zielarchitektur ist bei dieser Vorgehensweise zu berücksichtigen, daß der Prozeß P_j nur auf die entsprechende Speicherressource abgebildet werden kann, in der x gespeichert wird. Eine entsprechende Festlegung der Prozesse ist auch für den Fall erforderlich, daß ein Wert in den gemeinsamen Speicher geschrieben werden muß.
Das gezeigte Prinzip stellt neben den rein auf funktionalen Zusammenhang bezogenen Aspekten der Unterteilung der Funktion einen gegenläufigen Trend zur ansonsten auf Begrenzung der Prozeßanzahl ausgerichteten Vorgehensweise dar. Um die Aktivitäten auf der Kommunikationsarchitektur und ihren Einfluß auf die erreichbare Performance erfassen zu können, ist dieser Faktor jedoch in Kauf zu nehmen. Bei der automatischen Erzeugung des Graphenmodells aus der ausführbaren funktionalen Spezifikation, wie in Bild 3-3 gezeigt, müssen diese Informationen zur Wahl der geeigneten Prozeßgrenzen Berücksichtigung finden. Wird der Graph manuell wie bei dem im Kapitel 5.3 beschriebenen Verifikationsbeispiel erzeugt, ist dies bei der Wahl der Architektur und den darauf notwendigen Datentransfers, die die Anwendung benötigt, zu beachten.
Weiterhin wird davon ausgegangen, daß die einmal getroffene Entscheidung über die Unterteilung in einzelne Prozesse, die als Design-Objekte in der weiteren Vorgehensweise dienen, nicht durch Clustering oder Hierarchiebildung verändert wird (feste Granularität), so daß einer geeigneten Festlegung der Prozesse große Bedeutung zukommt. Soll das Abbildungsproblem in mehreren Durchläufen mit unterschiedlichen Vorgaben in Bezug auf die Speicherarchitektur untersucht werden, muß deshalb ggf. eine Anpassung der Prozeßgrenzen vorgenommen werden, falls Änderungen an der Speicherarchitektur vorgenommen werden.

Kommunikationsknoten

Sind die Prozesse P_i und P_j auf unterschiedlichen Verarbeitungsressourcen implementiert, muß der Transfer über die Kommunikationsarchitektur des Systems abgewickelt werden. Wie nachfolgend beschrieben, wird zu diesem Zweck ein Kommunikationsknoten im Graphen eingeführt. Sind jedoch beide Prozesse P_i und P_j auf derselben Ressource realisiert, ist zwar ebenfalls der Transfer notwendig, er erfolgt aber ressourcen-intern z.B. über Register, ohne daß die Kommunikationsinfrastruktur beteiligt ist. Wie bei den in der Literatur veröffentlichten Methoden üblich, bleibt auch in der vorliegenden Arbeit diese Art der lo-

kalen Kommunikation unberücksichtigt.
Durch die Einführung von Kommunikationsknoten können die beiden Aspekte Kommunikation und Verarbeitung auf die gleiche Weise behandelt werden. Dabei entsteht ein zweistufiges Abbildungsproblem: Die Zuordnung der Verarbeitungsknoten auf entsprechende Ressourcen bedingt das Auftreten von Kommunikationsknoten, die wiederum selbst auf die Kommunikationsarchitektur abgebildet werden müssen. Durch dieses Vorgehen ist eine Erweiterung der in dieser Arbeit betrachteten, auf einen gemeinsamen Bus beschränkte Kommunikationsarchitektur auf beliebige interne Verbindungsstrukturen möglich.
Generell ist bei dieser Vorgehensweise zu beachten, daß die beiden Knotentypen jeweils nur auf die zugehörigen, kompatiblen Ressourcen abgebildet werden können. Im Falle der Unterstützung einer erweiterten Kommunikationsarchitektur ist zudem die Konnektivität der jeweiligen Kommunikationsressourcen zu berücksichtigen. Nur Ressourcen, die die benötigte Verbindung herstellen können, kommen für die Abbildung des jeweiligen Transfers in Betracht. Dies entspricht den Einschränkungen durch eine Bibliothek von Verarbeitungsressourcen, bei der einzelne Ressourcen nicht alle Prozesse realisieren können.
Beim Scheduling der Knoten muß nicht zwischen Kommunikationsknoten und Verarbeitungsknoten unterschieden werden. Beide Knotenarten können nach der Festlegung der Abbildung gleich behandelt werden, auch im Hinblick auf die unten beschriebene Unterstützung von Kontrollabhängigkeiten.

4.2.2 Zielarchitektur

Das Verfahren zum rekursiven Abbilden und Scheduling ist gemäß dem im Kapitel 3.3.1 beschriebenen System-Entwurfsablauf in den Rahmen der Architekturexploration einzuordnen. Es geht von einer Zielarchitektur aus, die neben der als Graph beschriebenen Funktion als weiterer Input vorgegeben sein muß. Für diese Architektur wird eine Implementierung der Funktion mit möglichst geringer Gesamtlaufzeit gesucht. Unter einer Zielarchitektur soll im folgenden eine Architektur verstanden werden, die über eine Bibliothek von potentiell darin eingesetzten Ressourcen definiert ist, die in einer bestimmten Struktur miteinander verbunden sind. Unter Struktur sollen die Konnektivität der einzelnen Ressourcen und die dabei verwendeten Kommunikationsmedien verstanden werden.
Bei der Optimierung wird die vorgegebene Architektur nicht verändert. Die Struktur bleibt dabei fest, und ein Hinzufügen neuer Ressourcen ist nicht möglich. Derartige Anpassungen müssen in der übergeordneten Schleife vorgenommen werden, in der die Exploration des möglichen Lösungsraums für Architektur-Alternativen bewerkstelligt wird. Andererseits ist es jedoch möglich, daß einzelne in der Zielarchitektur vorhandene Ressourcen nicht genutzt werden, d.h. keine Teilfunktion darauf abgebildet wird. Eine derartige Änderung entspricht damit faktisch der Entfernung der jeweiligen Ressourcen.
Mit diesem Vorgehen wird keine Architektur unter Einhaltung von entsprechenden Anforderungen synthetisiert, sondern vielmehr die vorgegebene Zielarchitektur unter Verwendung der darin vorgegebenen Ressourcen für die zu implementierende, mit Kontrollabhängigkeiten versehene Funktion möglichst optimal in Bezug auf die Performance genutzt.

Für die Zielarchitektur wird angenommen, daß die darin benutzten Ressourcen parallel arbeiten und über die Kommunikationsarchitektur Daten austauschen, die auch zur Synchronisation dienen. Der Ablauf werde durch eine geeignete Steuerung kontrolliert, die das festgelegte Scheduling garantiert.

Struktur und Kommunikationsressourcen

Für die Struktur der Zielarchitektur wird angenommen, daß die vorhandenen Verarbeitungsressourcen über ein gemeinsam genutztes Kommunikationsmedium miteinander verbunden sind. Dieses ermöglicht den Datenaustausch zwischen den angeschlossenen Ressourcen und führt zu Zugriffskonflikten, die mittels Arbiter gelöst werden müssen. Konkret wird als Kommunikationsmedium ein gemeinsamer Bus, der durch seine Wortbreite und den Bustakt charakterisiert ist, angenommen. Als weitere Detaillierung kann leicht die Einführung von weiteren Parametern, wie die maximale pro Transaktion übertragbare Datenmenge oder Prioritäten-Informationen in das Verfahren berücksichtigt werden. Durch Unterteilung der Datenmenge in mehrere Transfers der maximalen Größe und der Einführung der entsprechenden Anzahl von Kommunikationsknoten können derartige Parameter leicht mit in das Verfahren einbezogen werden. Dies gilt auch für die Einführung einer ggf. vorhandenen Arbitrierungsverzögerung, die als zusätzlicher Kommunikationsknoten mit fester Dauer berücksichtigbar ist.
Komplexere Kommunikationsarchitekturen, bei denen die Verarbeitungsressourcen auf mehreren Kommunikationskanälen miteinander kommunizieren können, sind in dem Verfahren bisher nicht berücksichtigt. Durch eine Erweiterung im Sinne des oben beschriebenen zweistufigen Mapping-Problems ist dies jedoch ohne größere Probleme möglich. Durch die Vorgehensweise mit der Einführung von Kommunikationsknoten in den Graphen ist die Methodik hierfür vorbereitet.
Für die Kommunikationsarchitektur wird angenommen, daß Transfers gepuffert von den Ressourcen vorgenommen werden. D.h. zu transferierende Daten werden entweder in der sendenden oder empfangenden Verarbeitungseinheit solange zwischengespeichert, bis sie tatsächlich über den Bus transferiert oder von den eigentlichen Verarbeitungsressourcen genutzt werden. Die Frage der Dimensionierung dieser Puffer wird in dieser Arbeit nicht näher betrachtet, es wird die Verfügbarkeit von unbegrenztem Puffer an den Bus-Schnittstellen angenommen. Zur Bestimmung der erforderlichen Speichergrößen wären Simulationen der Systemarchitektur mit realen Stimuli bzw. Analysen des Worst Case unter Berücksichtigung der intern sich aufbauenden Warteschlangen erforderlich. Für die hier angestellten Untersuchungen in Bezug auf die erreichbare Performance sind diese Fragen jedoch nur von untergeordneter Bedeutung. Für die Verarbeitungsressourcen bedeutet dies außerdem, daß sie unabhängig von den Transfers arbeiten, d.h. eine Verzögerung aufgrund eines Zugriffskonflikts auf dem gemeinsamen Bus führt zu keiner lokalen Verzögerung der Verarbeitung. Eine derartige Entkopplung der Kommunikation von der Verarbeitung wird üblicherweise in den Arbeiten in diesem Themengebiet angenommen (z.B. [31] oder [123]).
Zu den Kommunikationsressourcen ist neben dem eigentlichen Übertragungsmedium, das eine direkte Kommunikation zwischen einzelnen Verarbeitungsressourcen im Sinne des Message Passing Konzepts ermöglicht, auch globaler Speicher zu zählen. In der Zielarchi-

tektur wird angenommen, daß Speicherblöcke zur Speicherung von Paketen sowie insbesondere von gemeinsamen Daten vorhanden sind, die zur Bearbeitung der einzelnen Teilfunktionen benötigt werden. Transfers von und zu diesem letztgenannten Speicher werden wie oben beschrieben innerhalb des Graphenmodells erfaßt.
Außerdem gehören zu dieser Kategorie von Kommunikationsressourcen auch die externen Schnittstellen der Zielarchitektur zur Peripherie. In bezug auf das Anwendungsgebiet Datenkommunikation sind hierzu z.B. die Schnittstellen der Linecard bzw. zum internen Switching Netzwerk zu rechnen. Für die weiteren Betrachtungen zum Mapping und Scheduling sind diese Schnittstellen jedoch nicht relevant.
Bild 4-3 zeigt die den Untersuchungen zugrundeliegende Architektur mit dem gemeinsamen Bus.

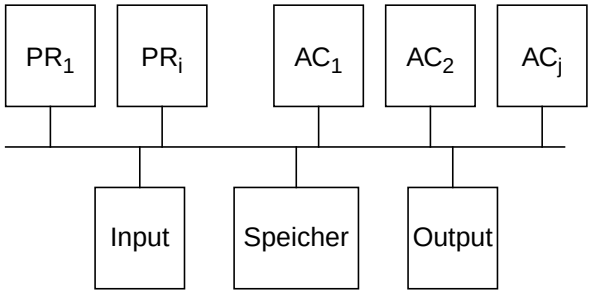


Bild 4-3: Zielarchitektur

Verarbeitungsressourcen

Neben den Kommunikationsressourcen wird von zwei Arten von Verarbeitungsressourcen ausgegangen, die in der Zielarchitektur eingesetzt werden können.

- flexibel programmierbare Prozessoren PR_i
- Beschleunigermodule AC_j

Beide Typen lassen sich über ihre Einsetzbarkeit zur Implementierung von Teilfunktionen innerhalb der SoC Lösung sowie der im Mittel damit erreichbaren Performance charakterisieren. Mit Hilfe eingebetteter Prozessoren ist es möglich, die erforderliche Flexibilität hinsichtlich der Funktionalität zu erreichen und somit nachträgliche Änderungen an der Lösung zu ermöglichen. Prozessoren erlauben im Prinzip die Implementierung aller Teilfunktionen, erreichen jedoch dabei geringere Performance, d.h. benötigen eine längere Ausführungsdauer als entsprechende Beschleuniger. Für die vorliegende Untersuchung wird von generischen Prozessoren ohne Berücksichtigung von Echtzeit-Betriebssystemen ausgegangen. Als Vorbild für diesen Ressourcen-Typus dienen die im Intel IXP1200 eingesetzten Microengines, die die flexible Implementierung von Paketverarbeitungsaufgaben des Fast Path ermöglichen und eine in HW realisierte Prozeßverwaltung besitzen. Verzö-

gerungen, die hierbei auftreten, seien in den jeweiligen Schätzwerten für die Ausführungsdauer der Prozesse enthalten.
Als zweite Klasse von Ressourcen sollen Beschleunigermodule berücksichtigt werden, die zur Implementierung spezieller Teilfunktionen vorgesehen sind, die auf Prozessoren nur mit zu geringer Performance realisierbar sind und die Leistungsfähigkeit des gesamten Systems begrenzen würden. Derartige Blöcke können entweder als sog. IP Module (Intellectual Property) von Drittanbietern mit fest vorgegebenem Funktionsumfang zugekauft oder als selbst entworfener HW Block entweder spezifisch für des konkrete Design-Projekt entwickelt oder im Sinne der Wiederverwertung aus vorangegangenen Vorhaben eingesetzt werden. Im Fall der spezifischen Entwicklung bestehen viele Freiheitsgrade der Definition des damit abgebildeten Funktionsumfangs, der verwendeten Algorithmen und auch der Einbindung in die Zielarchitektur, während bei bereits fertigen Modulen Anpassungen nur in sehr geringem Umfang möglich sind. Sowohl eingebettete Prozessoren als auch Beschleuniger sind als vorgefertigte Module verfügbar.

Ressourcenbibliothek

Die Ressourcenbibliothek faßt die in der Zielarchitektur verfügbaren Kommunikations- und Verarbeitungsressourcen mit ihren jeweiligen Parametern zusammen und dient neben der als Graph repräsentierten Funktion als Input für das rekursive Verfahren. Im Hinblick auf das Optimierungsziel einer möglichst geringen Verarbeitungszeit der Anwendung auf der Zielarchitektur ist für jeden Prozeß die Kenntnis erforderlich, auf welchen der vorhandenen Ressourcen er implementiert werden kann, und welche Ausführungsdauer sich jeweils im Worst Case auf den möglichen Ressourcen ergibt. Hier bestehen für die im Rahmen dieser Arbeit betrachteten SoC Architekturen Unterschiede zu Verfahren aus dem Bereich Multiprozessor-Architekturen. Dort wird üblicherweise von einer homogenen Architektur ausgegangen, bei der meist alle vorhandenen Ressourcen alle Prozesse des Graphen ausführen können. Im vorliegenden Fall gelten derartige Annahmen nicht. So können IP Module, die für eine bestimmte Funktionalität entworfen sind, ausschließlich diese Funktion realisieren und keinerlei andere Prozesse ausführen. Diese Heterogenität in Bezug auf den Funktionsumfang der verfügbaren Ressourcen muß in der Ressourcen Bibliothek erfaßt werden.
Die Ressourcenbibliothek besteht im Prinzip aus einer Matrix, in der die Liste der Prozesse und die Liste der verfügbaren Ressourcen zeilen- bzw. spaltenweise angegeben sind. Neben den die einzelnen Prozesse auf der jeweiligen Ressource kennzeichnenden Parametern ist die Angabe der Implementierbarkeit, d.h. der Zulässigkeit der jeweiligen Prozeß-Ressourcen Kombination erforderlich.
In dieser Ressourcen Bibliothek können auch indirekt andere Vorgaben, wie z.B. bestimmte Präferenzen der Implementierung definiert werden. Sollen z.B. bestimmte Alternativen per se ausgeschlossen sein, d.h. eine Implementierung eines bestimmten Prozesses als SW nicht erlaubt sein, kann dies ebenfalls auf diesem Wege gesteuert werden.
Zur Erstellung der Ressourcen Bibliothek ist es erforderlich, daß nach der Unterteilung der Gesamtfunktion in einzelne Prozesse und der Festlegung der in der Zielarchitektur vorhandenen Ressourcen eine Abschätzung dieser Parameter durchgeführt und die erhaltenen Daten für das Mapping und Scheduling bereitgestellt werden. Dieser Schritt wird in der

Literatur meist als Estimation bezeichnet. Konkret bedeutet dies, daß mit möglichst geringem Aufwand, und ohne daß die jeweiligen Funktionen tatsächlich komplett implementiert werden, ein möglichst genauer Schätzwert über die zu erwartende Verarbeitungszeit oder anderer charakteristischer Parameter bereitgestellt wird (s. Kapitel 2.2.2). Die notwendigen Informationen zur Charakterisierung von HW IP Modulen können dem zugehörigen Datenblatt entnommen werden.
Die Angaben in der Ressourcenbibliothek für das vorliegende Verfahren entsprechen der längsten Ausführungsdauer des jeweiligen Prozesses auf der entsprechenden Ressource, d.h. dem ungünstigsten Fall. Aus Vereinfachungsgründen wird diese ungünstigste Ausführungszeit im folgenden meist verkürzend nur als Ausführungsdauer bezeichnet.
Zur Untersuchung von alternativen Lösungen im Rahmen der Architekturexploration kann der Designer auch die Wirksamkeit spezifischer Beschleuniger untersuchen, indem bestimmte Ressourcen in die Ressourcen Bibliothek aufgenommen oder gezielt daraus entfernt werden. Dies entspricht mehreren Untersuchungen mit unterschiedlicher Zielarchitektur.

4.3 Unterstützung von Kontrollabhängigkeiten

Ein wesentlicher Neuigkeitsaspekt der vorliegenden Arbeit ist die Unterstützung von bedingten Übergängen in einem rekursiven Mapping- und Schedulingverfahren, wodurch Kontrollabhängigkeiten erfaßt werden können. Dies ist eine notwendige Voraussetzung, um einen derartigen Algorithmus in Systemen einsetzen zu können, bei denen der Kontrollfluß in der Anwendung stark von den zu bearbeitenden Daten abhängt.
Die Unterstützung von Kontrollabhängigkeiten im Rahmen des vorliegenden Verfahrens wird innerhalb des Scheduling-Schrittes umgesetzt, der in Kapitel 4.5.2 erläutert ist. In diesem Abschnitt sollen die hierzu verwendeten Prinzipien und ihre Umsetzung dargestellt werden.
Ein Graph mit Kontrollabhängkeiten beinhaltet infolge der bedingten Verzweigungen eine Vielzahl von möglichen Varianten, wie er für einen konkreten Fall, d.h. für eine konkrete Kombination von Bedingungswerten, tatsächlich ausgeführt werden kann. Jede dieser unterschiedlichen Möglichkeiten entspricht einem Graphen ohne Kontrollabhängigkeiten. Ziel des Verfahrens ist die Erzeugung eines für alle möglichen Kombinationen von Bedingungen gültigen Schedule und gleichzeitig die Optimierung der für die ungünstigste Bedingungskombination benötigte Ablaufzeit der Anwendung auf der Zielarchitektur.

4.3.1 Gegenseitige Exklusivität von Prozessen

Zur Berücksichtigung von Kontrollabhängigkeiten nutzt das Verfahren das Prinzip des gegenseitigen Ausschlusses und der gemeinsamen Ressourcen-Nutzung (Ressourcen-Sharing). Darunter ist zu verstehen, daß zwei oder mehrere Prozesse zeitlich überlappend auf derselben Ressource abgebildet werden können, wenn gewährleistet ist, daß in keinem Fall mehr als einer dieser überlappenden Prozesse tatsächlich ausgeführt werden muß. Für einen auf die Zielarchitektur abgebildeten CPG muß deshalb garantiert sein, daß es keine Bedingungskombination gibt, in der Zugriffskonflikte auf den Ressourcen zwischen zeitlich

überlappend zugeordneten Prozessen entstehen können.
Das nachfolgende Beispiel demonstriert das Prinzip für 2 gegenseitig sich über eine Bedingung C_1 ausschließende Äste in einem Graphen, der auf eine Architektur mit 2 Ressourcen abgebildet ist, und dessen Schedule erzeugt werden muß.

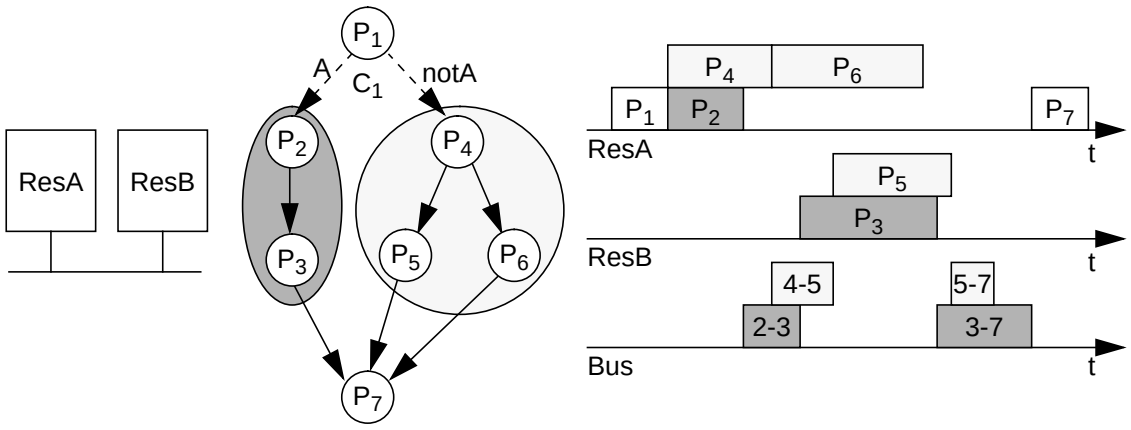
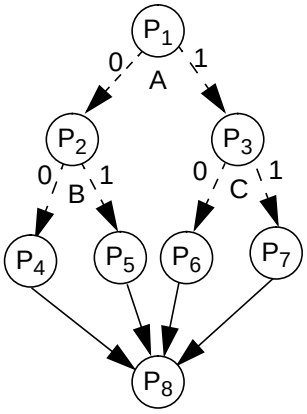


Bild 4-4: Beispiel zur Behandlung von Bedingungen

Die beiden grau schraffierten Bereiche des Graphen seien aufgrund der Bedingung C_1, die entweder wahr oder falsch sein kann, zueinander exklusiv. Da die Prozesse P_2 und P_3 nie ausgeführt werden müssen, wenn $C_1 = notA$ ist und deshalb P_4-P_6 aktiv werden bzw. umgekehrt, können die Schedules der beiden Äste auf den jeweiligen Ressourcen überlagert werden. D.h. Prozesse werden in überlappenden Zeitbereichen auf die vorgegebenen Ressourcen angeordnet. Dies gilt sowohl für Verarbeitungsprozesse als auch für Transfers auf der Kommunikationsarchitektur. Da P_7 erst ausgeführt werden kann, wenn beide gegenseitig exklusiven Zweige abgearbeitet sind, liefert diese Vorgehensweise außerdem die Worst Case Verarbeitungsdauer für beide möglichen Fälle.
Das Exklusivitäts-Prinzip ist eine bekannte und in vielen anderen Bereichen angewandte Methode, die die Zuordnung einer Ressource an mehrere Teilfunktionen in zeitlich überlappenden Zeitbereichen erlaubt. Die Voraussetzung hierfür ist jedoch, daß alle gleichzeitig allozierten Prozesse gegenseitig exklusiv sind, d.h. daß die Bedingungen, die mit den einzelnen Prozessen einhergehen, sich gegenseitig ausschließen und in keinem Fall gleichzeitig zutreffen.
Um die Exklusivität sicherzustellen, ist es erforderlich, beim Abbilden und Scheduling erkennen zu können, ob zwei beliebige Knoten des Graphen, die auf die gleiche Ressource, sei es eine Verarbeitungs- oder Kommunikationsressource, abgebildet werden, sich gegenseitig ausschließen. Zunächst wird deshalb die Erkennung der Exklusivität im Graphen näher betrachtet, bevor in Kapitel 4.5.2 im Rahmen des Scheduling-Schrittes erläutert wird, wie dies im vorliegenden Verfahren eingesetzt wird.
Verfahren zur Erkennung gegenseitig sich ausschließender Knoten hängen auch vom zugrundeliegenden Graphen ab. Wie nachfolgend erläutert wird, müssen entsprechende Me-

thoden insbesondere seine Struktur und die darin enthaltenen bedingten Überänge berücksichtigen. In der Literatur sind derartige Verfahren aus unterschiedlichen Anwendungsbereichen ([76], [115], [57] und [123]) beschrieben. In [123] wird ein konstruktiver Mapping- und Scheduling-Algorithmus mit Unterstützung von Kontrollabhängigkeiten beschrieben, der die gegenseitige Exklusivität beim Scheduling gemäß Bild 4-4 unterstützt. Die dabei verwendete Methode basiert auf der Annotierung der Knoten mit Informationen, in welchem bedingten Zweig des Graphen sich der jeweilige Knoten befindet. Die annotierte Information besteht dabei aus einer Hierarchie von Marken (Label), die die einzelnen durchlaufenen Bedingungen identifiziert, und den jeweiligen Bedingungswerten. Das folgende Bild 4-5 zeigt einen entsprechend annotierten Graphen.



Knoten	Ebene	Marken	Bedingung
P_1	0	n.v.	n.v.
P_2	1	A	A0
P_3	1	A	A1
P_4	2	A B	A0 B0
P_5	2	A B	A0 B1
P_6	2	A C	A1 C0
P_7	2	A C	A1 C1
P_8	0	n.v.	n.v.

Bild 4-5: Annotation des Graphen gemäß [123]

Anhand des schrittweisen Vergleichs der jeweils annotierten Daten kann die gegenseitige Exklusivität zweier Knoten im Graphen detektiert werden.
Bei dieser Methode ist eine Zusammenfassung von Zweigen, die zu unterschiedlichen Bedingungen gehören nicht möglich, da Übergänge zwischen Knoten mit unterschiedlichen Marken oder mit einer um mehr als eins unterschiedlichen Hierarchieebene nicht von diesem Schema erfaßt werden können. Derartige Situationen können jedoch durch Zusammefassung gemeinsamer Prozesse auftreten, die unter ganz verschiedenen Situationen aktiviert werden müssen.
Wären in obigen Beispiel z.B. die Prozesse P_5 und P_6 identisch, so könnten sie bei diesem Verfahren nicht in einem Prozeß zusammengefaßt werden, da sie unterschiedliche Marken auf der zweiten Ebene besitzen. Bestehen derartige gemeinsamen Zweige aus einer Vielzahl von Knoten, führt diese Einschränkung zu erheblichem Mehraufwand in der Modellierung des Graphen. Die bei diesem Verfahren zugrundegelegten Graphen mit streng gekapselten, hierarchisch angeordneten und mit gemeinsamen Bedingungen versehenen Bereichen stellen für den Anwendungsbereich der Paketverarbeitung eine zu starke Einschränkung dar, weshalb dieses Methode hier nicht angewendet werden kann.
Im Bereich der Verhaltens-Synthese ist die Erkennung von gegenseitig exklusiven Anweisungen ebenfalls ein sehr intensiv untersuchtes Thema. Ziel ist es hierbei, Verarbeitungs-

ressourcen mehrfach auszunutzen, um den Hardwareaufwand zu begrenzen und dabei gleichzeitig die Anzahl der notwendigen Verarbeitungsschritte nicht zu erhöhen. Dies entspricht grundsätzlich der vorliegenden Problematik, die Schedule Dauer bei vorgegebenen Ressourcen der Zielarchitektur zu minimieren.
Der Ausgangspunkt für die Verhaltens-Synthese ist ein in Form einer Beschreibungs- oder Programmiersprache vorliegendes Modell. Aufgrund des erlaubten Sprachumfangs oder auch des vom Designer gewählten Modellierungsstils ergeben sich jedoch Einschränkungen in Bezug auf die Struktur des Graphen, in den dieses Modell überführt werden kann.
Ein Beispiel für eine derartiges Verfahren ist das in [76] beschriebene Verfahren, das auf sog. "Timed Decision Tables" (TDT) beruht.
TDTs sind Tabellen, in denen systematisch Bedingungskombinationen auf die zugehörigen Aktionen bzw. Operationen abgebildet werden. Sie werden benutzt, um das in VHDL oder HardwareC beschriebene Modell als wiederholt aktivierte Prozesse oder als Prozeduren nachzubilden. Hierarchiebildung ist zur Reduktion des Komplexität möglich. Durch geeignete Transformationen können die TDTs so umgeformt werden, daß struktural und auf das Verhalten bezogene, gegenseitig exklusive Operationen in unterschiedlichen Spalten der Tabellen auftreten und auf diese Art und Weise leicht erkennbar sind. Das bei diesem Verfahren angewandte Prinzip stellt im Grunde die für die jeweilige Operation notwendigen Bedingungen auf und erlaubt durch geeignete Anordnung die Identifikation von gegenseitig exklusiven Anweisungen. Diese Methode basiert auf einem in einer Beschreibungssprache verfügbaren Modell und arbeitet lediglich auf der Ebene von Basic Blocks.
Diese Einschränkungen gelten auch für verschiedene weitere Verfahren aus dem Bereich Logiksynthese oder Verhaltens-Synthese. [57] enthält einen Vergleich von Methoden, die in diesem Bereich zum Einsatz kommen. Die in den genannten Referenzen beschriebenen Ansätze sind jedoch stark in Bezug auf die im Graph möglichen Übergänge eingeschränkt. Dies ergibt sich zum einen aus dem Anwendungsgebiet und der Art und Weise wie das für Exklusivitäts-Detektion benutzte Modell generiert wird. Auch die Unterstützung von mehr als zwei konditionalen Verzweigungen pro Knoten bzw. von zusätzlichen unkonditionalen Kanten ist nur teilweise möglich. Für die Erkennung gegenseitiger Exklusivität im Bereich Verhaltens-Synthese stellt dies kein Problem dar. Jedoch ist im Fall eines High-Level Systemmodells keinerlei Einschränkung möglich, da alle Übergänge, ob mit Bedingung versehen oder nicht, auftreten können.
Durch Betrachtung auf Basic Block Ebene alleine ist dies nicht möglich, andererseits kommen derartige beliebigen Übergänge im üblichen sprachorientierten Modellierungsstil für die Logik- oder Verhaltens-Synthese nicht vor bzw. sind nicht erlaubt. Bei der Bildung eines abstrakten Systemmodells für das betrachtete Anwendungsgebiet sind derartige Sprünge von und zu bedingten Ästen durch Abfragen bestimmter Situationen, die mit den gleichen notwendigen Aktionen verbunden sind, oder durch Zusammenfassungen von unterschiedlichen Fällen leicht möglich. Ein Beispiel hierfür ist die Notwendigkeit der Versendung von ICMP Paketen, die aufgrund unterschiedlichster Situationen erforderlich werden kann. Für den vorliegenden Fall der Untersuchung der Schedules für unterschiedliche Abbildungen auf der Systemebene sollen im Graphen deshalb keine Einschränkungen in Bezug auf die möglichen Übergänge innerhalb eines mit Kontrollabhängigkeiten versehenen Graphen gemacht werden.

Das folgende Beispiel in Bild 4-6, das einen Ausschnitt aus einem Graphen mit bedingten Übergängen zeigt, demonstriert, daß bei Vorhandensein beliebiger Übergänge in bedingte Bereiche hinein ($e_{x,16}$) oder aus solchen Bereichen heraus ($e_{12,y}$) den Umständen, unter denen die einzelnen Knoten aktiviert werden können, bei der Feststellung der gegenseitigen Exklusivität genaue Beachtung geschenkt werden muß.

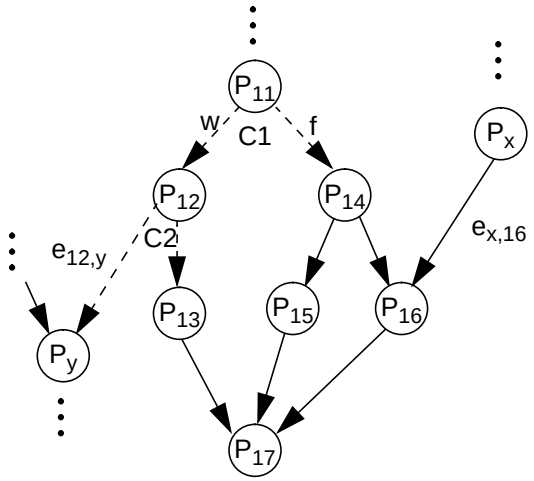


Bild 4-6: Übergänge von und zu bedingten Zweigen

Aufgrund des Übergangs von P_x nach P_{16} kann keine eindeutige Aussage über die Exklusivität zwischen P_{16} und z.B. P_{12} getroffen werden. Hierzu ist genaue Kenntnis erforderlich, unter welchen Bedingungen P_x als ein Vorgänger von P_{16} ausgeführt wird. Ebenso sind Übergänge nach P_y aus Bereichen des Graphen im Vorfeld von P_{11} denkbar, die dazu führen können, daß P_y zu den Knoten P_{13} bis P_{17} nicht gegenseitig exklusiv ist, obwohl der erste Anschein dies erwarten lassen könnte.
Aus diesem Grund wurde für die Anwendung im Zusammenhang mit der rekursiven Mapping- und Partitionierungsmethode ein allgemeines Verfahren entwickelt, das unabhängig von der Struktur des Graphen und ohne Einschränkung für die darin erlaubten Übergänge die gegenseitige Exklusivität zweier Knoten erkennen kann. Diese Methode knüpft an das in [115] beschriebene, auf sog. "Condition Vectors" basierende Verfahren an, das auch über die Grenzen von Basic Blocks hinaus optimieren kann. Aufgrund der Vergabe nur eines einzelnen Bedingungsvektors pro Knoten ist jedoch eine Unterscheidung von verschiedenen Situationen, die einen Knoten aktivieren können, nicht möglich.
Da ein Knoten auf unterschiedlichen Pfaden, d.h. mit unterschiedlichen und unabhängigen Kombinationen von Bedingungen erreicht werden kann, wird dieser Ansatz hier auf Bedingungspfade (Condition Pfade) erweitert. In jedem Knoten wird eine Liste aller voneinander unabhängigen Kombinationen von Bedingungen abgelegt, die eine Aktivierung des Knotens hervorrufen können. Zur Überprüfung des gegenseitigen Ausschlusses zweier Knoten muß paarweise geprüft werden, ob es eine gemeinsame Bedingungs-Kombination gibt, die die Aktivierung beider Knoten zur Folge hat. Sollte mindestens eine derartige Kombination

existieren, sind die beiden Knoten nicht gegenseitig exklusiv.
Die Umsetzung dieses Ansatzes erfolgt in zwei Stufen. Zunächst wird die notwendige Datenbasis über die Condition Pfade einmalig beim Initialisieren des Graphen generiert, die Überprüfung erfolgt dann jeweils für die fraglichen Knoten während des Scheduling Schrittes.

4.3.2 Annotation des Graphen zur Vorverarbeitung

Zur Implementierung wird jeder verzweigenden Bedingung im Graphen eine eindeutige Nummer C_i mit $i = [1,n_{cond}]$ zugeordnet, wobei n_{cond} die Anzahl der im Graphen vorkommenden Knoten mit abgehenden, bedingten Übergängen ist. Ein mit einer Bedingung versehener Knoten soll im weiteren als bedingter Knoten bezeichnet werden. Jede von einem bedingtem Knoten abgehende, konditionale Kante wird ebenfalls eindeutig durch die Zuordnung einer Bedingungsnummer *CN* identifiziert. Diese Bedingungsnummer ist ein "one-hot" codierter Binärwert, der die unterschiedlichen Alternativen kennzeichnet. Das nachfolgende Bild 4-7 zeigt einen Ausschnitt eines Graphen, in dem 2 bedingte Knoten mit 3 bedingten bzw. 2 bedingten und einem unbedingten Übergang dargestellt sind.

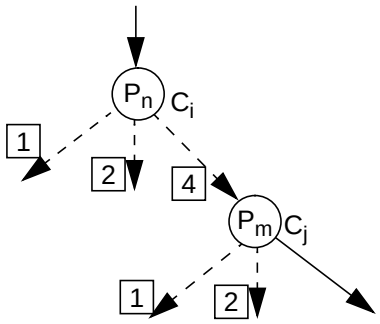


Bild 4-7: Numerierung von Bedingungsknoten und -werten

Ein Condition Pfad *CondP* beschreibt eine Kombination von Bedingungen, über die der zugehörige Knoten des Graphen erreicht werden kann. Die Anzahl der Komponenten eines Condition Pfades $CondP = (CV_1, CV_2, \ldots, CV_{n_{cond}})$ entspricht der Gesamtzahl n_{cond} der Bedingungen, die im Graphen enthalten sind. Jeder Eintrag CV_i in einem Condition Pfad entspricht der Bedingungsnummer *CN* der *i*-ten Bedingung, die bei der Abarbeitung des Graphen durchlaufen werden muß, um zum betreffenden Knoten zu gelangen. Infolge der nachfolgend beschriebenen Zusammenfassung von Bedingungspfaden bei der Annotation des Graphen können mehrere Bedingungsnummern überlagert werden, wodurch Werte mit mehreren Bits entstehen. Jedes in einem CV_i gesetzte Bit entspricht damit einer mit einer Bedingung versehenen Kante, die vom zur Bedingung C_i gehörenden Knoten P_n abgeht.
Ist eine Bedingung C_k für einen Condition Pfad nicht relevant, d.h. wird der betreffende Knoten unabhängig von C_k erreicht, entspricht dies einem "don’t care", das nachfolgend als

"-" gekennzeichnet ist. Konkret bedeutet dies, daß im zugehörige Eintrag CV_k alle Bits gesetzt werden. Dies ergibt sich aus der nachfolgend beschriebenen Art und Weise, wie die gegenseitige Exklusivität überprüft wird. "Don't-Cares" in CV_k treten auf, wenn der zu der Bedingung *k* gehörende Knoten nicht durchlaufen oder über eine zugehörige unbedingte Kante verlassen wird.
Jedem Knoten können mehrere Condition Pfade zugeordnet sein, abhängig davon auf wievielen unabhängigen Wegen der Knoten erreichbar ist. Wird ein Knoten ohne jegliche Bedingung immer aktiviert, sind alle CV_i auf "don't care" gesetzt. Ein derartiger Knoten ist mit keinem anderen Knoten gegenseitig exklusiv. In Bild 4-8 ist ein Beispiel eines mit den Bedingungsinformationenen versehenen Graphen gezeigt.

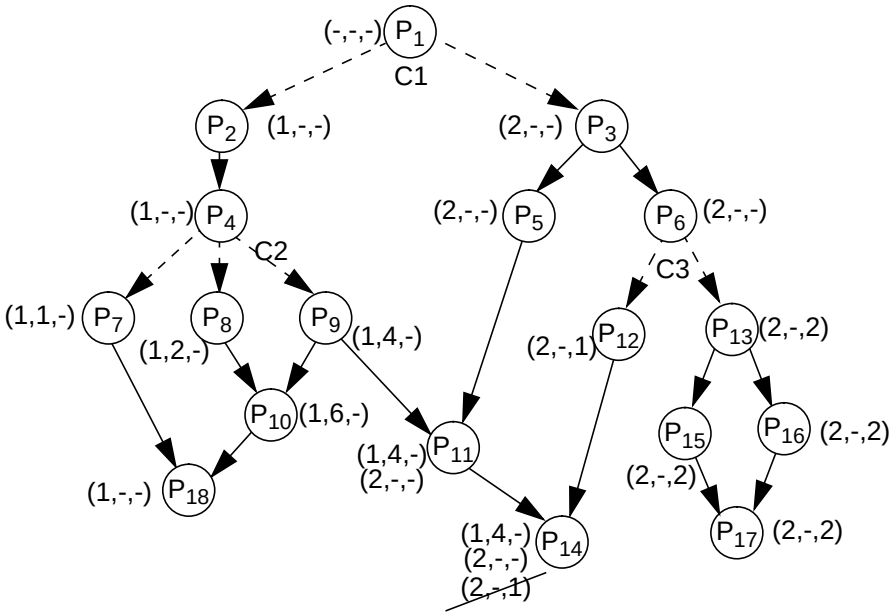


Bild 4-8: Annotierter CPG mit Condition Pfaden

Bei der Erstellung der Pfad-Informationen für die einzelnen Knoten des Graphen sind folgende Prinzipien zu beachten:

- Ein unbedingter Übergang e_{ij} transportiert die komplette Pfad-Information des Ausgangsknotens *i* zum Zielknoten *j*.
- Ein bedingter Übergang e_{kl}, der mit der CV_m assoziiert ist, transferiert die Pfad-Information des Knotens *k* nach *l*, wobei jedoch bei allen transferierten Pfaden an der Stelle *m* der Werte auf CN_m gesetzt wird.

Zur Abarbeitung müssen jedoch folgende weiteren Gesichtspunkte berücksichtigt werden, um sowohl die Datenmenge der pro Knoten zu speichernden Pfad-Information als auch den Aufwand bei der Überprüfung der Exklusivität zu begrenzen:

- Ist mindestens ein Vorgänger *i* eines Knotens *j* ohne Pfad-Information, d.h. immer

auszuführen, und ist die zugehörige Kanten e_{ij} ohne Bedingung, so ist auch der Knoten *j* immer auszuführen, und es muß keine Pfad-Information gespeichert werden. Dies gilt auch, falls Pfad-Information bei anderen Vorgängern vorhanden sind.

- Zur Reduktion des Speicherbedarfs werden doppelt vorhandene Pfade, die sich parallel im Graphen ausbreiten können, bei zusammenlaufenden Knoten eliminiert, d.h. nur einmal gespeichert. Ein Beispiel hierfür ist Knoten P_{17} im obigen Graphen.
- Pfad-Konvergenz: Unterscheiden sich zwei oder mehr Pfade nur in einem gemeinsamen Bedingungswert CV_x, und sind sie ansonsten identisch, ist eine Zusammenfassung der Pfade durch komponentenweise Oder-Verknüpfung möglich. Dies entspricht dem Zusammenlaufen von mehreren alternativen Zweigen der Bedingung *x* (P_{10} im obigen Beispiel) oder der Situation, wenn die Bedingung *x* durch einen alternativen Pfad irrelevant ist (P_{14}). Wenn alle möglichen Bedingungswerte in dem Bitmuster vorkommen, d.h. wenn die Anzahl der gesetzten Bits der Anzahl der zur Bedingung x gehörenden exklusiven Kanten entspricht, kann an dieser Stelle ein "don't-Care" gesetzt werden (P_{18} in Bild 4-8). Nachfolgende Knoten können aufgrund der zugehörigen Bedingung x nicht mehr zu anderen Knoten exklusiv sein. D.h. für nachfolgende Knoten hat die Bedingung *x* ihre Wirkung als Kriterium für den gegenseitigen Ausschluß mit anderen Knoten verloren.

Die oben genannten Prinzipien spiegeln sich im zugehörigen Algorithmus wieder, der nachfolgend in Bild 4-9 in Pseudo-Code Notation angegeben ist. Dabei wird davon ausgegangen, daß der Graph beginnend vom Startknoten abgearbeitet wird. Aufgrund der Übernahme der Pfad-Information vom Vorgängerknoten, entweder unverändert oder mit einer Bedingung modifiziert, müssen alle Vorgänger bearbeitet sein, bevor ein Knoten mit den Pfad-Informationen annotiert werden kann.

4.3.3 Überprüfung des gegenseitigen Ausschlusses

Die Datenstrukturen zur Beschreibung der Bedingungs-Pfade wurden derart gewählt, daß die Feststellung der gegenseitigen Exklusivität zweier Knoten sehr einfach durchgeführt werden kann. Hierzu müssen paarweise alle zu beiden Knoten gehörenden Condition Pfade komponentenweise, d.h. für jeden CV_i mit $i = [1,n_{cond}]$, miteinander UND-verknüpft werden. Bei dieser Verknüpfung kann leicht festgestellt werden, ob das jeweils betrachtete Paar von Condition Pfaden gemeinsame Bedingungen besitzt, die es erlauben, daß die beiden Knoten aktiviert werden. Resultiert für wenigstens ein *i* ein Wert mit ausschließlich Bitwerten von 0, bedeutet dies, daß beide Pfade keine Gemeinsamkeiten besitzen, die beide Knoten aktivieren. Das betrachtete Pfad-Paar ist damit kein Hindernis für Exklusivität, die anhand der weiteren Pfad-Paare festgestellt werden muß. Resultieren bei der Verknüpfung jedoch für alle Komponenten Werte, bei denen mindestens ein Bit gesetzt ist, kann aufgrund der gemeinsamen Pfadinformation sofort auf die Nicht-Exklusivität geschlossen werden.

```
set node = start node
while(not all nodes are processed)
do
     for all successors(node)
     do
          if(not all predecessors are processed)
               continue;
          for all predecessors
          do
               if(transition is conditional)
                    copy cond_paths(predecessor node) to node
                    set condition_Val to all copied paths
               else
                    if(predecessor is all don't-care)
                         set node to all dont't-care
                         break
                    else
                         copy cond_paths(predecessor node) to node
          done
     done
     remove duplicate cond_paths
     join cond_paths
     mark node processed
     select next node
done
```

Bild 4-9: Algorithmus zur Annotation des Graphen mit Condition Pfaden

Dies soll nachfolgend anhand von zwei Beispielen aus dem obigen Graphen demonstriert werden. In Bild 4-10 soll die gegenseitige Exklusivität der Prozesse P_{10} und P_{14} bzw. P_7 und P_{14} aus dem obigen Beispiel überprüft werden.

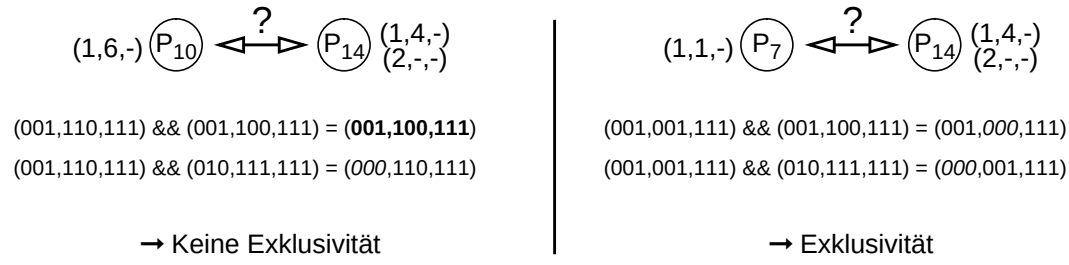


Bild 4-10: Beispiele zu Überprüfung der gegenseitigen Exklusivität

Am obigen Beispiel ist auch die Wirkung der "don't care" Werte ersichtlich, die bei diesem Verfahren den jeweils anderen Bedingungswert dominieren lassen. Sind in einem Knoten keine Condition Pfade vorhanden, d.h. handelt es sich um einen unbedingt ausgeführten Knoten, ist dieser mit jedem anderen Knoten nicht exklusiv. Diese Eigenschaft wird vor der o.g. Prozedur abgeprüft und führt unmittelbar zu einem negativen Ergebnis.

Da das vorgestellte Verfahren rekursiv immer neue Zuordnungen von Knoten auf die einzelnen Ressourcen überprüft, die Verarbeitungsknoten jedoch bzgl. ihrer Exklusivitäts-Eigenschaften unverändert bleiben, wird das Ergebnis einer Exklusivitätsprüfung zweier Knoten in einem Exklusivitäts-Cache gespeichert. Falls zu einem späteren Zeitpunkt in der gleichen oder einer zukünftigen Iteration die beiden selben Knoten erneut auf Exklusivität überprüft werden müssen, ist eine erneute Durchführung der geschilderten Prozedur nicht mehr erforderlich, sondern ein Zugriff auf das gespeicherte Exklusivitätsinformation erlaubt eine sehr schnelle Überprüfung. Da Kommunikationsknoten abhängig von der Abbildung der Verarbeitungsknoten im Graphen vorkommen, wird eine derartige Speicherung für sie nicht durchgeführt.

```
if(node_a is uncond || node_b is uncond)
     return non_exclusive
if(exclusivity_chache(node_a,node_b)==hit)
     return exclusivity(node_a,node_b)
else
     for all cond_path_a
     do
          for all cond_path_b
          do
               set exclusive=no
               for all cond
               do
                    if(cond_path_a(cond) && cond_path_b(cond)==0
                         set exclusive=yes
                         break
               done
               if(exclusive==no)
                    store exclusivity(node_a,node_b)
                    return non_exclusive
          done
     done
     store exclusivity(node_a,node_b)
     return exclusive
end if
```

Bild 4-11: Algorithmus zur Überprüfung der gegenseitigen Exklusivität zweier Knoten

4.4 Definition der Nachbarschaft für die lokale Suche

Lokale Suche basiert auf der Definition einer geeigneten Nachbarschaft zur aktuellen Lösung. Diese Nachbarschaft wird durchsucht und daraus die aktuelle Lösung für den nächsten Iterationsschritt ausgewählt. Ziel ist dabei, durch geeignete Wahl die zu optimierende Größe, hier die Schedule Dauer, zu minimieren.
Die Nachbarschaft *N(s)* der aktuellen Lösung s ist demnach eine Teilmenge des gesamten

Lösungsraums L, d.h. $N(s) \subseteq L$. Durch diese Einschränkung kann eine komplette Suche im gesamten Lösungsraums, die aus Aufwandsgründen nicht möglich ist, durch eine iterative Nachbarschaftssuche ersetzt werden. Durch eine geeignete Wahl der Nachbarschaft und des Optimierungsverfahrens soll eine dem globalen Optimum möglichst nahe kommende Lösung gefunden werden. Die Definition der Nachbarschaft für das Abbildungs- und Schedulingverfahren wird nachfolgend beschrieben, auf das Optimierungsverfahren wird in Kapitel 4.5.3 eingegangen.

Für die Definition einer für die lokale Suche günstigen Nachbarschaft sind unterschiedliche Anforderungen zu beachten:

- *N(s)* sollte nicht zu groß sein, um das iterative Durchsuchen möglichst effizient durchführen zu können.
- *N(s)* sollte möglichst zielgerichtet definiert sein, um Lösungen zu finden, die die Suche stark in Richtung zum globalen Minimum der zu optimierenden Größe lenken

Eine Lösung $l \in L$ ist dabei eine erlaubte Abbildung der Prozesse des Graphen auf die Ressourcen in der Architektur $V \rightarrow Res_i$.

Eine Nachbarschaft ist dadurch gekennzeichnet, daß alle Lösungen aus *N* in einer bestimmten Beziehung zur Lösung s stehen. Bei einer Nachbarschaftsdefinition für das vorliegende Abbildungs-Problem handelt es sich um alternative Zuordnungen der Prozesse auf die Ressourcen, die sich gemäß bestimmter begrenzender Kriterien von der aktuellen Lösung unterscheiden. Als Unterscheidungskriterien kommen in Betracht

a) die Auswahl der Prozesse, die alternativ abgebildet werden können, und
b) die Anzahl der anders zugeordneten Prozesse.

Als sehr einfache Variante einer Nachbarschaftsdefinition ist beispielsweise die Menge aller Lösungen, die sich bzgl. ihrer Abbildung in der Zuordnung genau eines Prozesses von der aktuellen Lösung unterscheiden. Bei dieser Definition wäre keine Einschränkung gemäß a) enthalten, was zu einer relativ undifferenzierten Suche führen würde, da die Suche nicht in eine bestimmte Richtung gelenkt würde. Eine Nachbarschaftsdefinition, die durch gezielte Auswahl von Knoten, bei denen eine Änderung ihrer Zuordnung auf die Ressourcen potentiell zu einer Verbesserung führt, beruht auf dem nachfolgend beschriebenen kritischen Pfad.

4.4.1 Kritischer Pfad

Der kritischer Pfad stellt den längsten von unterschiedlichen Wegen durch den Graphen dar, der damit die gesamte Laufzeit des Graphen auf der betrachteten Architektur bestimmt. Aufgrund dieser Tatsache stellt der kritische Pfad das Umfeld dar, in dem eine Änderung der Abbildung der zugehörigen Knoten auf alternative Ressourcen Potential zur Beschleunigung besitzt. Die Knoten des kritischen Pfads werden deshalb von verschiedenen anderen Mapping- und Scheduling-Ansätzen ebenfalls zur Optimierung der Zuordnung auf die Res-

sourcen verwendet ([58], [70], [122]).
Ausgehend von dem auf die Verarbeitungsressourcen abgebildeten, mit Kommunikationsknoten versehenen und um die jeweiligen Verarbeitungs- bzw. Transferzeiten annotierten Graphen kann der kritischer Pfad bestimmt werden. Grundsätzlich ist es möglich, daß ein Graph mehrere kritischen Pfade mit der gleichen maximalen Länge besitzt. Dies stellt jedoch für die nachfolgend beschriebene Vorgehensweise keine Problem dar. Wird ein kritischer Pfad durch Änderungen in der Zuordnung eines seiner Knoten beschleunigt, reduziert sich die Zahl der gleich langen kritischen Pfade. Falls jedoch durch alternatives Mapping eine Verlangsamung resultiert, entsteht dadurch ebenfalls ein eindeutiger kritischer Pfad, der in den weiteren Iterationen wieder die Nachbarschaft für die lokale Suche darstellt.
In Bild 4-12 ist das Beispiel eines kritischen Pfads als Ausschnitt eines komplexeren Graphen angegeben.

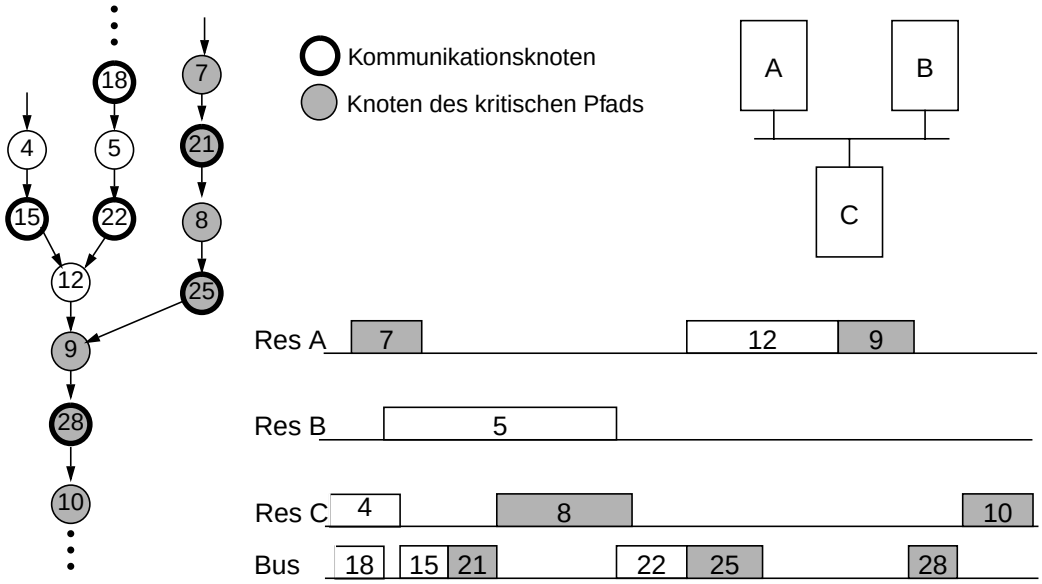


Bild 4-12: Beispiel eines kritischen Pfads für eine Architektur mit 3 Ressourcen

In diesem Beispiel sind neben den Verarbeitungsknoten auch die eingefügten Kommunikationsknoten, die ebenfalls zum kritischen Pfad gehören können, eingezeichnet. Durch geeignetes Mapping ist es möglich, die Verarbeitungsdauer von Prozessen durch Zuweisen auf schnellere Ressourcen zu reduzieren und damit die gesamte Laufzeit der Anwendung zu reduzieren, falls die jeweils notwendige Kommunikation im System einen möglichen Gewinn nicht überkompensiert. Alternativ sind auch Lösungen denkbar, bei denen Kommunikationsknoten im kritischen Pfad vermieden werden können, wenn die miteinander kommunizierenden Prozesse auf dieselbe Ressource abgebildet werden, wodurch ebenfalls eine Verkürzung der Ausführungszeit erreicht werden kann. Dieses setzt voraus, daß die alternative Abbildung der Verarbeitungsknoten nicht zu erhöhten Bearbeitungszeiten führt, die den Vorteil der Vermeidung von Kommunikation zunichte macht. Mit Hilfe von lokaler Suche werden im Umfeld des kritischen Pfads alternative Zuordnungen derartiger Knoten

aus dem kritischen Pfad vorgenommen, um diese Effekte auszunutzen.

4.4.2 Erweiterter Kritischer Pfad

Wenn man den kritischen Pfad mit den Knoten $P_{CP,i}$ in einem Graphen genauer untersucht, erkennt man jedoch, daß für seine Dauer *D(CP)* in Abhängigkeit von der Verarbeitungs- bzw. Transferzeit $D(P_{CP,i})$ der einzelnen Knoten gilt

$$D(CP) \geq \sum_i D(P_{CP,i}) . \tag{4.1}$$

Durch Ressourcen, die von mehreren Prozessen oder Transfers genutzt werden und die gleichzeitig lauffähig wären, kommen jedoch im kritischen Pfad Verzögerungen einzelner $P_{CP,i}$ zustande, die eine weitere Verzögerung der Anwendung hervorrufen. Diese zusätzlichen Verzögerungen werden nachfolgend als Wait-Zeiten $T_{Wait,i}$ bezeichnet. Daraus ergibt sich

$$D(CP) = \sum_i D(P_{CP,i}) + \sum_i T_{Wait,i} \tag{4.2}$$

Eine Verkürzung der Schedule Dauer ist damit nicht nur durch die oben genannten alternativen Zuordnungen möglich, mit denen die Dauer einzelner Knoten reduziert oder im Fall von Kommunikationsknoten auch ganz eliminiert werden können, sondern auch durch Beseitigung des verzögernden Einflusses anderer Knoten auf Knoten des kritischen Pfads. Im Bild 4-13 sind die Wait-Zeiten aus obigem Beispiel gekennzeichnet.

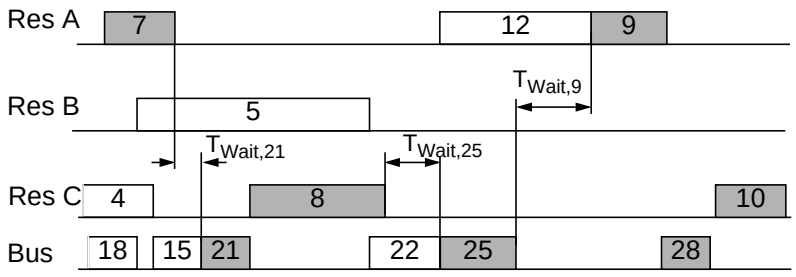


Bild 4-13: Schedule mit Wait-Zeiten

In Bild 4-13 erkennt man, daß der kritische Pfad ggf. auch durch Vermeidung der Transfers 15 und 22 verkürzt werden könnte, da dann die nachfolgenden Transfers 21 und 25 früher stattfinden könnten. Dies würde bedeuten, daß man durch ein Mapping der Knoten 4 und 12 oder 5 und 12 auf eine gemeinsame Ressource möglicherweise Verbesserungen erreichen kann, obwohl diese Knoten nicht direkt zum kritischen Pfad gehören. Ähnliche Verzögerungen können jedoch auch auf Verarbeitungsressourcen auftreten. So könnte eine alternative Abbildung für Prozeß 12 die Verzögerung von Prozeß 9 vermeiden.

Eine weitere Möglichkeit besteht darin, Vorgänger von derartig verzögernd wirkenden Knoten ebenfalls alternativ abzubilden, was ebenfalls die Wait-Zeit vermeiden oder zumindest reduzieren kann. Dies trifft z.B. für Prozeß 8 in obigem Beispiel zu.
Aus diesem kleinen Beispiel ergibt sich damit eine Liste von Optionen, aus denen die Nachbarschaft um weitere Knoten erweitert werden kann, um zielgerichtet Knoten zu definieren, die erhöhtes Potential besitzen, daß ein alternatives Zuordnen auf die Ressourcen zu einer Verkürzung der Ausführungszeit führt.
Als Kandidaten für eine erweiterte, auf den Kritischen Pfad bezogene Nachbarschaft kommen deshalb folgende Prozesse in Frage:

a) Verarbeitungsknoten des kritischen Pfads
b) Kommunikationsknoten des kritischen Pfads
c) Transfers, die direkt vor Kommunikationsknoten liegen, die eine Wartezeit besitzen
d) Vorgängerknoten von Prozessen des Typs c)
e) Prozesse, die Verarbeitungsknoten des kritischen Pfads mit Wartezeit vorangehen

Die Erweiterung der Nachbarschaft, die mit diesen zusätzlichen Optionen verbunden ist, führt jedoch zu einer Vergrößerung des Suchaufwands. Auf diesen Aspekt wird in im weiteren noch eingegangen. Die durch den erweiterten kritischen Pfad gebildete Nachbarschaft wird nachfolgend mit "ECP", die ausschließlich Knoten des kritischen Pfads umfassende Variante wird mit "CP" referenziert.

4.5 Der Ablauf des rekursiven Verfahrens

Nachfolgend werden die einzelnen Teilschritte des Verfahrens beschrieben. Bild 4-14 gibt einen Überblick über das Verfahren. Ausgangspunkte sind die Festlegung der Zielarchitektur, das funktionale Modell sowie die Abschätzung der Implementierungsparameter der einzelnen Teilfunktionen auf den jeweils möglichen Verarbeitungsressourcen der Zielarchitektur. Die danach folgenden Schritte werden rekursiv in einer Schleife durchlaufen, bis die Abbruchbedingung erfüllt ist.
Neben dem genannten Input ist eine initiale Abbildung der Prozesse auf die Ressourcen erforderlich, das den Startpunkt für die Iterationen bildet. Als Alternativen sind hier zum einen eine zufällige Wahl oder eine deterministische Vorgabe möglich. Da die Optimierung ausschließlich nach Schedule Dauer vorgenommen wird, wird für dieses Anfangs-Mapping die Auswahl der jeweils für die einzelnen Prozesse schnellsten Ressourcen angenommen. Untersuchungen haben gezeigt, daß diese Vorgehensweise vorteilhaft sowohl in Bezug auf die erreichbare Lösung als auch auf die Laufzeit ist. In Kapitel 4.5.3 und insbesondere bei den durchgeführten Experimenten wird dieser Punkt noch näher beleuchtet.
Ausgehend von der initialen Abbildung wird für die jeweilige Zuordnung der Prozesse ein Schedule erzeugt und als Vorbereitung für die Festlegung der Nachbarschaft analysiert. Die Dauer des Schedules bildet die zu Optimierende Größe. In der Optimierungsschleife wird zum einen die jeweils gültige aktuelle Lösung festgelegt, über die die Nachbarschaft

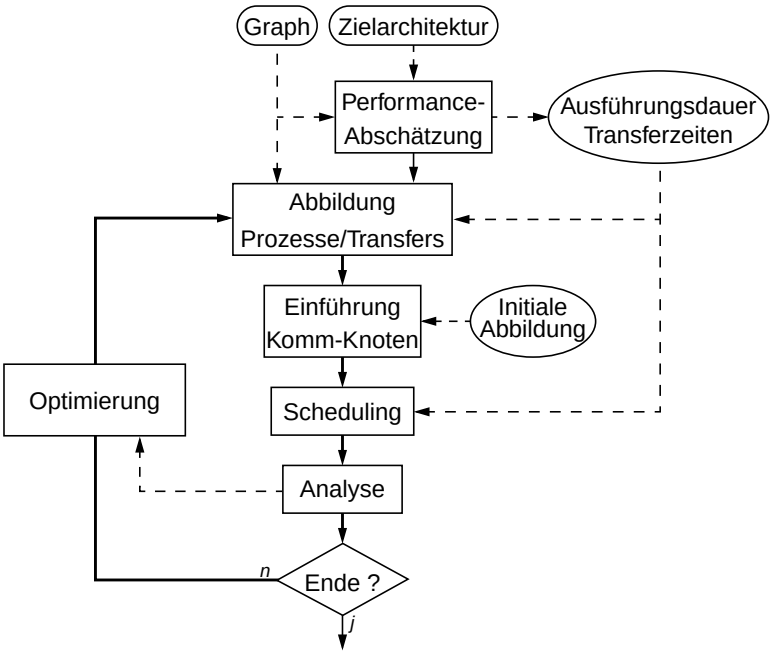


Bild 4-14: Ablauf des rekursiven Abbildungs - und Scheduling-Verfahrens

definiert wird, und zum anderen auch die Nachbarschaftssuche durchgeführt. Das Abbruchkriterium hängt vom jeweils verwendeten Suchalgorithmus ab. Darauf wird nachfolgend bei der Beschreibung der betrachteten Verfahren eingegangen.

4.5.1 Abbildung der Prozesse auf die Ressourcen

Zunächst werden die einzelnen Prozesse den Ressourcen entsprechend der initialen Abbildung, bzw. in den weiteren Iterationen gemäß dem durch den Optimierungsalgorithmus vorgegebenen Mapping zugeordnet, und die zugehörigen Verarbeitungsdauern annotiert. Danach werden abhängig von der Abbildung der Prozesse auf die Verarbeitungsressourcen Kommunikationsknoten eingefügt, die als eigenständige Prozesse verwaltet werden, und die nur auf Kommunikationsressourcen - hier auf den gemeinsamen Bus - abgebildet werden können. Die Dauer der Kommunikationsknoten entspricht der reinen Übertragungszeit über das Medium und ist durch die Taktrate und die Wortbreite des Busses sowie durch die zu transferierende Datenmenge bestimmt. Neben der reinen Übertragungszeit kann als Erweiterung noch ein möglicher Arbitrierungsoverhead als fester Verzögerungsanteil mit einbezogen werden. Auch eine Begrenzung der maximal in einem Burst übertragenen Datenmenge ist hier möglich. Das würde eine Unterteilung eines Kommunikationsknotens in mehrere einzelne, bei denen jeweils die maximale Datenmenge übertragen wird, und den letzten Transfer mit dem verbleibenden Daten bedeuten.
Alle Kommunikationsknoten haben jeweils nur einen Vorgänger und einen Nachfolger und besitzen die gleichen Eigenschaften bzgl. Condition Pfaden wie ihre Vorgänger-Knoten. Auf diese Weise ist es möglich, konkurrierende Transfers auf der Kommunikationsressour-

ce nach dem gleichen Schema wie Verarbeitungs-Prozesse zu behandeln und die gegenseitige Exklusivität auch dieser Knoten, wie in Kapitel 4.3 beschrieben, auszunutzen.
Das Bild 4-15 verdeutlicht die Abbildung eines Graphen auf eine Architektur mit einem Prozessor PR_1 und zwei Beschleunigern AC_1 und AC_2, die über einen gemeinsamen Bus kommunizieren.

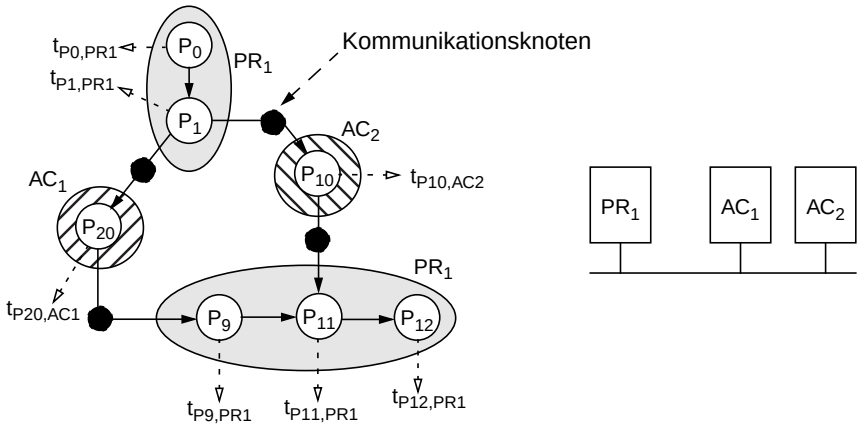


Bild 4-15: Abbildung eines CPG auf eine busbasierte Architektur

4.5.2 Scheduling

Als Voraussetzung für die Bewertung der Güte einer Abbildung muß ein Scheduling der Prozesse auf der vorgegebenen Architektur durchgeführt werden. Dieses Scheduling liefert als Ergebnis die Dauer der Anwendung auf der Architektur für die jeweilige Zuordnung der Prozesse auf die enthaltenen Ressourcen.
Im Schedulingschritt ist die Behandlung der bedingten Verzweigungen des Graphen enthalten. Wie in Kapitel 4.3 dargestellt, beruht dies auf der Eigenschaft des gegenseitigen Exklusivität zweier Prozesse. Dort ist auch das verallgemeinerte Verfahren zur Detektion dieser Eigenschaft zwischen zwei Prozessen, die lauffähig sind und der gleichen Ressource zugeordnet wurden, erläutert.
Nachfolgend wird dargestellt, wie diese Eigenschaft beim Scheduling berücksichtigt wird. Grundsätzlich beinhaltet der Graph alle Möglichkeiten der Verarbeitung, die durch die unterschiedlichen Kombinationen der Bedingungen gegeben sind. Mit dem hier verwendeten Schedulingverfahren wird gewährlistet, daß die resultierende gesamte Ausführungsdauer für den Graphen der Worst Case Situation entspricht. Das folgende kleine Beispiel in Bild 4-16 soll dies demonstrieren.
Am Knoten 1 finde eine über die Bedingung *Cond* kontrollierte Aufspaltung in zwei sich gegenseitig ausschließende Pfade statt. Falls die Bedingung den Wert *C* annimmt, soll die Verarbeitung mit Knoten 2 fortfahren, andernfalls mit dem Knoten 3. Im Prozeß 4 treffen beide Pfade wieder zusammen. Das zweite Fragment aus einem Graphen zeigt eine Situation, bei der sich die Verarbeitung bei Knoten 11 ohne Bedingung in die beiden parallelen Prozesse 12 und 13 aufspaltet und bei Knoten 14 wieder zusammentrifft. Im letzten Fall ist

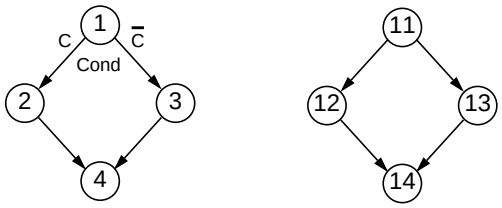


Bild 4-16: Graphen zur Demonstration des Worst Case Schedule

eindeutig, daß Knoten 14 erst dann lauffähig ist, wenn beide Vorgängerknoten 12 und 13 abgeschlossen sind. Betrachtet man den linken Teilgraphen, so hängt der Zeitpunkt, ab dem der Knoten 4 lauffähig ist, davon ab, welchen Wert die Bedingung *Cond* angenommen hat, und wann der zugehörige Prozeß des gewählten Pfads abgeschlossen ist. Um die Worst Case Situation zu erfassen, sind jedoch immer beide Fälle zu berücksichtigen. Im linken wie im rechten Beispiel können jeweils beide Äste nach Knoten 1 bzw. 11 zeitlich angeordnet werden. Die Lauffähigkeit der Knoten 4 bzw. 14 ist dann gegeben, wenn die jeweiligen Vorgänger komplett abgeschlossen sind. Konkret bedeutet dies, daß es in Bezug auf das Scheduling bei bedingten Verzweigungen nicht relevant ist, welche bedingte Kante tatsächlich aktiviert wird, denn alle möglichen Fälle müssen im Schedule enthalten sein. Dies unterscheidet die Erstellung eines Schedule für alle möglichen Bedingungen von einer Simulation, die einen konkreten Fall, d.h. eine bestimmte Kombination von Bedingungen, erfaßt, und bei der im linken Graphenfragment die Verarbeitung mit Knoten 4 fortfahren würde, unabhängig vom nicht ausgewählten Ast.
Bedingte Verzweigungen werden jedoch bei der Erstellung des Schedule derart berücksichtigt, daß bei Zuordnung der Knoten auf die gleiche Ressource eine zeitlich überlappende Ausführung festgelegt werden kann, wenn Knoten sich gegenseitig ausschließen. Für das obige Beispiel würde das jeweils den folgenden Schedule (links für den Graphen mit der Bedingung) bedeuten, wenn man annimmt, daß alle Knoten auf die gleiche Ressource abgebildet worden sind.

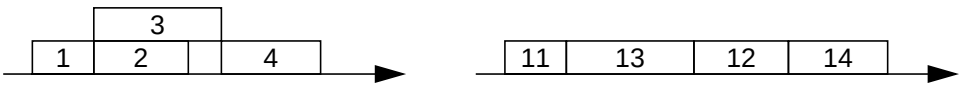


Bild 4-17: Beispiel-Schedule zur Demonstration des Worst Case

Besitzen die beiden bedingten Pfade zum Knoten, an dem sie zusammenlaufen, unterschiedliche Laufzeit, muß jedoch gewährleistet sein, daß auch der ungünstigste Fall möglich ist. Deshalb darf Prozeß 4 erst beginnen, wenn der länger dauernde der beiden Prozesse abgeschlossen ist. Die Lauffähigkeit von Prozeß 4 ist deshalb erst dann gegeben, wenn beide Vorgänger "virtuell" beendet sind. Die zeitliche Lücke im Fall *Cond=C* ist deshalb notwendig um auch den Worst Case *Cond=*$\overline{C}$ abzudecken. Im rechten Beispiel ist die rein sequentielle Abarbeitung notwendig.

Da die Prozesse bereits über den vorangegangenen Abbildungs-Schritt den einzelnen Ressourcen zugeordnet wurden, ist im Scheduling lediglich die Abfolge der Prozesse innerhalb der Architektur festzulegen. Zur Durchführung dieser Aufgabe wird ein einfacher List Scheduling Ansatz verwendet, der nachfolgend erläutert wird.
Ausgangspunkt ist dabei zunächst der um die Kommunikationsknoten erweiterte Graph. Für alle Knoten dieses Graphen werden jeweils Prioritäten vergeben. Als Priorität wird die maximalen Pfadlänge vom jeweiligen Knoten zum Endknoten verwendet. Unter Pfadlänge ist in diesem Zusammenhang die maximale Zeit zu verstehen, die nach dem Knoten folgende Prozesse oder Transfers benötigen, bis die Verarbeitung am Ende angelangt ist. Dies ist eine typische auf den kritischen Pfad bezogene Prioritätsdefinition ([60]). Diese maximale Pfadlänge zum Endknoten wird in der Literatur oft auch als b-Level angegeben ([68]).
Die bereits oben angesprochenen Wartezeiten, die aufgrund von Überlagerungseffekten entstehen, sind dabei nicht berücksichtigt, deshalb stellen diese Prioritäten optimistische Schätzungen für die Laufzeiten bis zum Ende dar. Diese Prioritäten sind statisch, d.h. sie bleiben während des Scheduling-Vorgangs fest, und dienen dazu, die Reihenfolge, in der die Knoten bei Ressourcenkonflikten zugeordnet werden aufzulösen. Höhere Priorität bedeutet dabei eine längere, a priori zu erwartende Laufzeit bis zum Ende, weshalb die zugehörigen Prozesse präferiert werden mit dem Ziel, die gesamte Dauer zu minimieren.
Für jede Ressource, einschließlich der Kommunikationsressourcen, werden jeweils zwei Listen verwaltet. Darin sind zum einen alle Prozesse enthalten, die lauffähig sind, aber deren zeitliche Lage noch nicht festgelegt wurde, und zum anderen alle Prozesse, die zum aktuellen Zeitpunkt auf der Ressource laufen. Die zweite Liste ist deswegen erforderlich, da aufgrund gegenseitiger Exklusivität von Prozessen die Ressource mit mehreren derartigen Prozessen belegt sein kann, und für jeden potentiell neu hinzukommenden Prozeß die Exklusivität mit allen bereits laufenden Prozessen überprüft werden muß.
Das Scheduling basiert auf einem Ereignis-basierten Vorgehen. D.h. sobald auf einer Ressource die Bearbeitung eines laufenden Prozesses abgeschlossen wird, werden die davon abhängenden Prozesse daraufhin überprüft, ob neue Prozesse lauffähig geworden sind. Anschließend wird für alle Ressourcen überprüft, ob weitere Prozesse festgelegt werden können. Auch wenn eine Ressource belegt ist, kann ggf. ein zu den aktuell laufenden Prozessen sich ausschließender Prozeß trotzdem auf der Ressource zum betrachteten Zeitpunkt angeordnet werden. Die Überprüfung wird jeweils beginnend mit dem höchstprioren Prozeß aus der Liste der lauffähigen Prozesse begonnen und paarweise mit den bereits auf der Ressource laufenden Prozessen durchgeführt. Ist gegenseitige Exklusivität mit allen laufenden Prozessen gegeben, kann der neue Prozeß hinzugefügt werden.
Für alle Ressourcen wird jeweils eine Liste der als nächstes endenden Prozesse verwaltet, aus der der nächste Zeitpunkt für ein Ereignis entnommen wird. Die Referenz bildet eine gemeinsame Zeitbasis, anhand derer die Zeit bis zum nächsten Ereignis auf einer der Ressourcen inkrementiert wird.
Nachfolgend ist in Bild 4-18 der verwendete Scheduling Algorithmus in Pseudo-Code Notation angegeben:

```
while not all nodes scheduled
do
     update ready_list
     for all resources
     do
          for all nodes m from ready_list highest prio first
          do
               no_sched = 0;
               for all nodes n from run_list
               do
                    if(n and m are not exclusive)
                         no_sched = 1;
                         break;
               done
               if(no_sched == 1)
                    continue;
               else
                    schedule node m
                    update run_list
          done
     done
     jump to time of next ending nodes
     remove all completed nodes from run_list
done
```

Bild 4-18: Scheduling Algorithmus

4.5.3 Optimierung

Nachfolgend wird die Auswahl des Optimierungsverfahrens beschrieben, das in die rekursive Schleife eingebunden wird. Das Verfahren soll einen Parameter, die Schedule Dauer, die sich aus dem Scheduling Schritt ergibt, minimieren. Zum Einsatz soll ein allgemeines, auch in anderen Bereichen eingesetztes Verfahren zur Optimierung mit Hilfe lokaler Suche kommen. Damit die Suche nicht in lokalen Minima endet, müssen auch Verschlechterungen im Vergleich zu in vorangegangenen Iterationen gefundenen Lösungen akzeptiert werden. Ein einfaches Hill-Climbing ist deshalb per se ausgeschlossen.
Als Kandidaten wurden daher zwei Methoden in Betracht gezogen, die die Lösung für den nächsten Iterationsschritt aus der Nachbarschaft entweder durch statistische Auswahl oder durch eine deterministische Auswahl bestimmen. In der näheren Betrachtung waren Simulated Annealing und Tabu Suche, die die genannten Eigenschaften besitzen. In [2], [33] und [35] wurden diese Verfahren für das Abbildungs- und Scheduling-Problem ohne Kontrollabhängigkeiten in einfacheren Architekturen bzw. mit Einschränkungen in Bezug auf die Betrachtung der systeminternen Kommunikation zum Einsatz gebracht. Nachfolgend werden beide Verfahren kurz beschrieben. Beide Algorithmen wurden implementiert und in Tests mit künstlichen Graphen verglichen. Die dabei erhaltenen Daten wurden neben den Ergebnissen der zitierten Literatur als zusätzliche Informationsquelle zur Auswahl des für das vorliegende Verfahren besser geeigneten Algorithmus verwendet.

Tabu Suche

Bei der Tabu Suche ([41], [42]), im weiteren mit TS abgekürzt, wird die Nachbarschaft der aktuellen Lösung vollständig untersucht und die bzgl. der Kostenfunktion beste Lösung als neue aktuelle Lösung ausgewählt. Dies gilt unabhängig davon, ob diese gewählte Lösung schlechter als die aktuelle oder auch die bisher beste gefundene Lösung ist. Um in keiner endlosen Schleife mit Pendeln zwischen zwei Lösungen zu verharren, werden alle gewählten Lösungen in eine Tabu Liste aufgenommen, aus der sich der Name des Verfahrens herleitet. Bevor eine Lösung evaluiert wird, wird überprüft, ob diese Lösung in der Tabu Liste enthalten ist. Falls dies zutrifft, wird diese Lösung bei der Überprüfung der Nachbarschaft nicht weiter betrachtet. Dieses Vorgehen ermöglicht es, lokale Minima zu verlassen.
Die Tabu Liste arbeitet als FIFO Speicher mit einer bestimmten Größe S_{FIFO}. Erreicht der Füllstand den Wert S_{FIFO}, werden die ältesten Einträge von den neuesten überschrieben. Dies führt dazu, daß in der Suche Schleifen entstehen können, die der Größe des Tabu FIFO entsprechen, falls die überschriebene Lösung sich noch in der Nachbarschaft der aktuellen Lösung befindet. Deshalb kommt dem Abbruchkriterium zusätzliche Bedeutung zu. Falls für eine vorzugebende Anzahl von Iterationen I_{max} keine Verbesserung der bisher besten gefundenen Lösung erreicht wird, wird die Suche beendet. Konkret bedeutet dies, daß man noch I_{max} Iterationen ausführt, falls man eine bereits in einem früheren Schritt als aktuelle Lösung gewählte Abbildung erneut auswählt. Ein weiteres Kriterium für den Abbruch der Suche stellt die Situation dar, wenn in der Nachbarschaft z.B. nur noch aus Möglichkeiten, die im Tabu FIFO enthalten sind besteht, oder für Knoten in der Nachbarschaft aufgrund Einschränkungen bei den Ressourcen keine alternativen Möglichkeiten der Abbildung existieren.

In Bild 4-19 ist der verwendete Algorithmus in Pseudo-Code Notation wiedergegeben ([41]). (Das zweite Abbruchkriterium wurde dabei aus Gründen der Übersichtlichkeit weggelassen.)

Um eine effiziente Überprüfung auf Gleichheit einer Abbildung durchführen zu können und dabei einen komponentenweisen Vergleich der Zuordnungs-Information aller Knoten zu vermeiden, wird ein Hash-Wert über die Abbildungsinformation aller Knoten gebildet und zusätzlich zu jeder Abbildung abgespeichert. Dies ermöglicht es, daß nur bei Identität des Hash-Werts ein komponentenweiser Vergleich notwendig ist, im Normalfall jedoch nur die Hash-Werte verglichen werden müssen.

Simulated Annealing

Simulated Annealing ([41], [62]), nachfolgend auch als SA bezeichnet, ist eine Methode, die aus dem Bereich der Materialwissenschaften entlehnt ist und den Abkühlprozeß z.B. eines Metalls nachbildet, bei dem durch kontrollierte Temperaturverminderung und Übergang auf einen minimalen energetischen Zustandes günstige Eigenschaften erreicht werden. Konkret bedeutet dies für eine Anwendung in Bereich HW/SW-Partitionierung, daß aus der Nachbarschaft der aktuellen Lösung zufällig eine mögliche alternative Lösung

```
empty TabuFIFO
s = initial_mapping;
best_mapping = initial_mapping;
best_dur = infinity;
no_improvment = 0;
do
     min_dur = infinity;
     for all mappings m of N(s)
     do
          if m in TabuFIFO
               continue;
          if(dur(m) < min_dur)
               min_mapping = m;
               min_dur=dur(m);
     done
     store min_mapping in TabuFIFO
     s = min_mapping;
     if(min_dur < best_dur)
          best_mapping = min_mapping;
          best_dur = min_dur;
          no_improvement = 0;
     else
          no_improvement++;
while(no_improvement < iter_max)
return best_mapping
```

Bild 4-19: Tabu Suche Algorithmus

ausgewählt wird. Führt diese Lösung zu einer Verbesserung im Vergleich zur aktuellen Lösung, so wird diese akzeptiert und stellt dann die neue aktuelle Lösung für den nächsten Iterationsschritt dar. Führt die aus der Nachbarschaft gewählte Lösung zu keiner Verbesserung, wird eine gleichverteilte Zufallszahl $x \in [0, 1)$ generiert. Gilt $x < e^{\frac{-\Delta C}{T}}$, so wird die Lösung trotz der Verschlechterung akzeptiert. In der voranstehenden Relation bedeuten ΔC die Differenz zwischen den Kosten (im vorliegenden Fall der Schedule Dauer) der gewählten Lösung zu denen der aktuellen Lösung und T die aktuelle Temperatur des Abkühlvorgangs. Kleine Verschlechterungen der Kosten werden damit mit größerer Wahrscheinlichkeit akzeptiert als größere. Mit abnehmender Temperatur des Abkühlvorgangs verkleinert sich die Akzeptanzwahrscheinlichkeit einer Lösung, die gegenüber der aktuellen Lösung eine Verschlechterung darstellt, ebenfalls. Auf diese Weise wird zum einen gewährleistet, daß lokale Minima der Kostenfunktion verlassen werden können, zum anderen ist damit eine Konvergenz der Lösung erreichbar, wenn ein geeignetes Abbruchkriterium damit verbunden wird.
Der betrachtete Simulated Annealing Algorithmus ist im Bild 4-20 in Pseudo-Code Notation ([41]) angegeben.
Als Kriterium für die Erreichung des thermischen Gleichgewichts innerhalb der Abkühlschleife wurde eine feste Anzahl von Versuchen angenommen. Als Abkühl-Schedule wurde eine geometrische Reduktion der Temperatur verwendet.

```
s = inital_mapping;
T = Tstart;
load Tmin
min_dur = infinity;
do
     do
          select random mapping rm from neighborhood N(s)
          delta = dur(rm)-dur(s);
          if(delta <= 0)
               s = rm;
               if(dur(rm) < min_dur)
                    min_mapping = rm;
                    min_dur=dur(rm);
          else
               if(exp(-delta/T) > random(1))
                    s = rm;
          end if
     while(!thermal_equilibrium)
     decrement T
while(T>Tmin)
return min_mapping
```

Bild 4-20: Simulated Annealing Algorithmus

4.5.4 Festlegung des Optimierungsverfahrens

Auswahl des Optimierungsverfahrens

Für die beiden Algorithmen wurden anhand von künstlichen Graphen (die Erzeugung dieser Graphen ist in Kapitel 5.2.1 erläutert) Experimente durchgeführt und die erzielten Schedule Dauern und die dafür erforderlichen Laufzeiten der Algorithmen untersucht. Beide Verfahren besitzen eine sehr breite Palette an Parametern zur Anpassung, die für die Entscheidung nur in eingeschränktem Maße untersucht werden konnten. Insbesondere wurde bei Simulated Annealing lediglich ein geometrischer Abkühl-Schedule verwendet und bei der Tabu Suche ein verhältnismäßig kleines Tabu FIFO.

Die Ergebnisse von Untersuchungen an Graphen mit 20, 50, 100 und 200 Knoten ohne bedingte Kanten sind in der folgenden Tabelle zusammengefaßt ([118]). Hierfür wurden jeweils 60 verschiedene Exemplare pro Graphgröße erzeugt und die Algorithmen für verschiedene Nachbarschaftsdefinitionen darauf angewendet.

Neben den auf den kritischen Pfad bezogenen Nachbarschaften CP und ECP wurde zum Vergleich eine weniger gerichtet wirkende Nachbarschaft untersucht, bei der sich die Abbildung genau eines Knotens von der aktuellen Abbildung unterscheidet. Diese ist nachfolgend als "One" gekennzeichnet. Auf diese Weise kann der Lösungsraum nahtloser aber weniger zielgerichtet durchsucht werden.

Für die Schedule Dauer wurde jeweils der Mittelwert aus den Teilergebnissen gebildet und relativ zu TS-ECP angegeben. Da es sich bei Simulated Annealing um ein statistisches Ver-

fahren handelt, wurde jeder Graph bei diesem Verfahren in 5 Läufen mit unterschiedlichen Startwerten für den Zufallsgenerator durchgeführt. Von diesen Ergebnissen wurde jeweils das beste ausgewählt, als Laufzeit wurde die Summe der einzelnen Läufe verwendet. Aufgrund der übergroßen Laufzeiten für Simulated Annealing wurden keine Graphen mit 200 Knoten simuliert.

	Graph Größe	SA-One	SA-ECP	TS-One	TS-CP	TS-ECP
Schedule Dauer rel. zu TS-ECP	20	+4,36%	+1,64%	-0,27%	+2,48%	-
	50	+7,40%	+0,45%	-0,47%	+4,53%	-
	100	+7,99%	+1,55%	-0,41%	+4,54%	-
	200	n/a	n/a	-0,50%	+6,16%	-
Laufzeit *)	20	3,1	192,4	0,11	0,05	0,5
	50	38,2	4399,3	1,36	0,36	0,61
	100	249,7	9415,9	14,59	2,14	6,95
	200	n/a	n/a	197,81	14,71	111,83
*) Werte in s; AMD ATHLON XP 1700+						

Tabelle 4-1: Vergleich der Optimierungsalgorithmen

Die Ergebnisse in Tabelle 4-1 zeigen durchgehend Vorteile für die Tabu Suche im Vergleich zu Simulated Annealing. Insbesondere hinsichtlich der für die Bearbeitung der Graphen benötigten Rechenzeit ist TS der Optimierung mit Simulated Annealing weit überlegen.
Aufgrund dieser Ergebnisse, die für Simulated Annealing sowohl geringere Performance als auch sehr hohe Laufzeiten im Vergleich zu Tabu Search ergaben, wurde Simulated Annealing für die Verwendung als Optimierungsverfahren in der vorliegenden Arbeit nicht weiter betrachtet. Die sehr starken Unterschiede bei der Schedule Dauer und die extremen Nachteile bei der Laufzeit, die durch weitere Optimierungen insbesondere in Bezug auf den Abkühl-Schedule sicherlich hätten reduziert werden können, wurden jedoch als zu groß eingeschätzt, als daß zur Tabu Suche konkurrenzfähige Werte erreichbar gewesen wären. Diese Vorgehensweise wurde als gerechtfertigt angesehen, da die Literatur, in der beide Verfahren für Anwendungen ohne Kontrollabhängigkeiten untersucht wurden ([2], [33] und [35]), ebenfalls die Tabu Suche als der besser geeignete Algorithmus beurteilt wurde.

Auswahl der Parameter für die Tabu Suche

Nachfolgend wird die Festlegung der Parameter für die Tabu Suche beschrieben. Für die zugehörigen Untersuchungen, wurden dabei ebenfalls künstliche Graphen verwendet, deren Erzeugung in Kapitel 5.2.1 beschrieben ist.
Das hier verwendete Tabu Suche Verfahren besitzt neben der Definition der Nachbarschaft im wesentlichen zwei Parameter, die den Verlauf der Suche bestimmen. Zum einen ist die Größe S_{FIFO} des Tabu FIFO entscheidend für die Vermeidung von Schleifen in der Suche, die nicht mehr verlassen werden können. Der andere Parameter I_{max} begrenzt die Dauer der Suche durch eine maximale Anzahl von Nachbarschaftssuchen. Wird innerhalb dieses Li-

mits keine Verbesserung des bisher gefundenen besten Ergebnisses erzielt, wird die Suche abgebrochen.
Die Untersuchungen dieser Parameter wurden an künstlichen Graphen mit 100 Knoten mit einem Anteil von 10% bedingter Knoten vorgenommen. Die mittlere, zwischen den Knoten transferierte Datenmenge betrug 50 Bit. Die Zielarchitektur bestand jeweils aus 2 Prozessoren und 5 Beschleunigerblöcken. Auf den Beschleunigern waren jeweils 25%, auf den Prozessoren 100% der Prozesse ausführbar. Für die Verarbeitungsdauer der Prozesse wurde angenommen, daß sie im Mittel auf den Beschleunigern mit Faktor 5 schneller ausgeführt werden.
In den nachfolgenden Diagrammen sind für die drei Nachbarschaftsdefinitionen One, CP und ECP die Schedule Dauer jeweils absolut mit der Anzahl von abstrakten Zeiteinheiten t_a eingegeben. Diese Zeiten ergaben sich aus der Abarbeitung der Graphen unter Einbeziehung der Ressourcen-Informationen, die beide vom Werkzeug zur Graph-Generierung ([25]) erzeugt wurden (s. hierzu Kapitel 5.2.1). Die Angabe erfolgt hier insbesondere in absoluten Zahlen, um den Unterschied zwischen den nachfolgend untersuchten Varianten der initialen Abbildung, für die jeweils derselbe Satz von Graphen verwendet wurde, zu verdeutlichen.
Zuerst wurde von einer initialen Abbildung ausgegangen, bei der alle Knoten jeweils der Ressource mit der für sie kürzesten Ausführungsdauer zugeordnet waren.

Zunächst wurden Untersuchungen zum Einfluß von I_{max} durchgeführt, deren Ergebnisse in Bild 4-21 angegeben sind. Die Größe des Tabu FIFO betrug hierbei 100 Einträge.

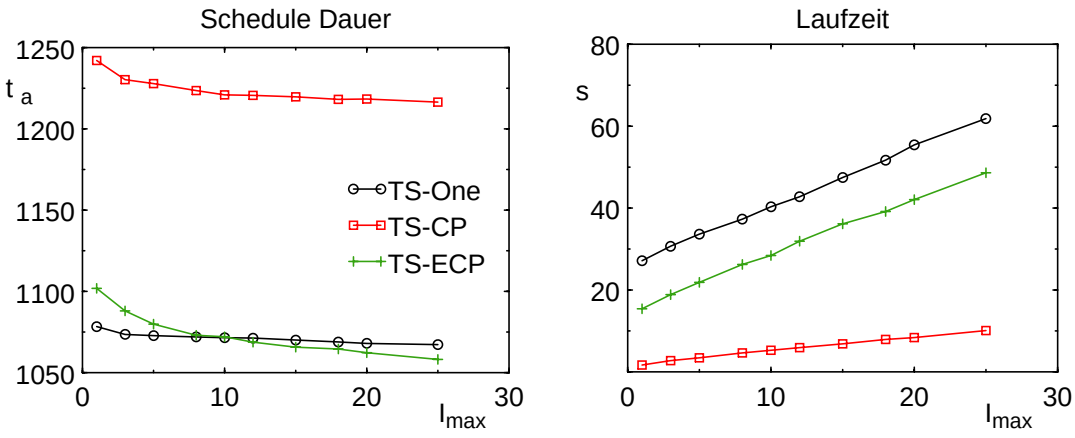


Bild 4-21: Einfluß der maximalen Anzahl von Iterationen (min. Initialmapping) bei der Tabu Suche

Alle Verfahren, besonders TS-ECP, finden mit steigendem I_{max} zunehmend bessere Lösungen. Dieser Trend ist jedoch nur für kleine Werte von I_{max} bis zu 7 und für die am kritischen Pfad orientierten Verfahren ausgeprägt, danach fallen die Kurven der Schedule Dauer sehr langsam ab. TS-One besitzt durchgehend einen relativ flachen Verlauf. Diese kontinuierliche Abnahme ergibt sich aufgrund der größeren Anzahl von Nachbarschaftssuchen wäh-

rend derer weitere Bereiche des Lösungsraums abgedeckt werden, die noch bessere Lösungen beinhalten. Die starke Abflachung ist in erster Linie von der benutzten initialen Abbildung verursacht. Da bereits von der für alle Verarbeitungsknoten jeweils schnellsten Ressource ausgegangen wird, sind nur noch beschränkte Verbesserungen möglich. Der anfänglich stärkere Rückgang für die am kritischen Pfad orientierten Verfahren wird von der Erreichbarkeit von für die Performance besonders relevanten Lösungsalternativen verursacht, die auf die Relevanz des kritischen Pfads für die Lösung hindeuten. Durch die Suche im erweiterten kritischen Pfad bzw. bei ungerichteter Suche werden im Vergleich zu der ausschließlich auf den kritischen Pfad basierenden Variante wesentlich bessere Lösungen generiert.
Die Werte für die benötigten Laufzeit zeigen die erwartete lineare Abhängigkeit von I_{max}. Die unterschiedliche Steigung korrespondiert mit der mittleren Größe der in einer Iteration zu betrachtenden Nachbarschaft, die im Gegensatz zu ECP und One bei der konventionellen kritischen Pfad Nachbarschaft wesentlich kleiner ist.
Als einen Kompromiß zwischen benötigter Laufzeit und den mit zunehmender maximaler Iterationszahl I_{max} erreichbaren Verbesserungen, wurde der Wert von I_{max} für die Tabu suche auf 10 festgelegt.
In einem darauffolgenden Schritt wurde der Einfluß der Größe S_{FIFO} des Tabu FIFO für die unterschiedlichen Nachbarschaftsdefinitionen untersucht.

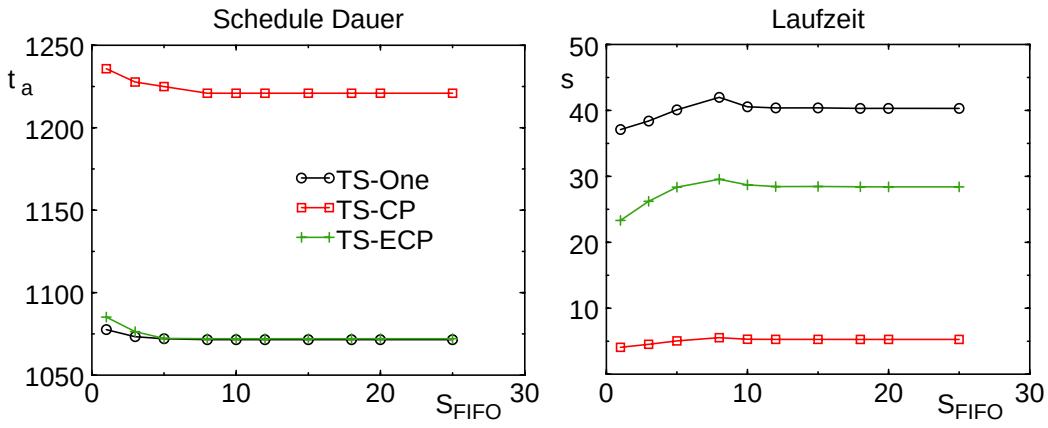


Bild 4-22: Einfluß der Tabu FIFO-Größe (min. Initialmapping)

Die Ergebnisse der Untersuchungen zeigen die sehr geringe Abhängigkeit sowohl der Schedule Dauer als auch der Laufzeit von diesem Parameter. Aus diesem Grund wurde für das weitere Vorgehen ein Wert von 500 Einträgen festgelegt.

Neben den konkreten Parametern der Tabu Suche wurde abschließend zur Festlegung der Optimierungsmethode der Einfluß der initialen Abbildung auf die von den Verfahren gelieferten Ergebnisse untersucht. Im Vergleich zur bisher eingesetzten Anfangs-Zuordnung der Knoten auf die Ressourcen mit der kürzesten Verarbeitungszeit wurde deshalb zum Test für die Fähigkeit, sich der minimalen Lösung anzunähern, die Vorgabe einer wesent-

lich ungünstigeren Ausgangsposition überprüft. Hierbei kann auch der Einfluß von I_{max} besser demonstriert werden. Zu diesem Zweck wurden Tests mit einer initialen Zuordnung der Knoten auf die für sie ungünstigste Ressource, d.h. mit der längsten Ausführungszeit bei sonst gleichen Randbedingungen, d.h. insbesondere mit den identischen Graphen durchgeführt. In Bild 4-23 sind die Ergebnisse enthalten.

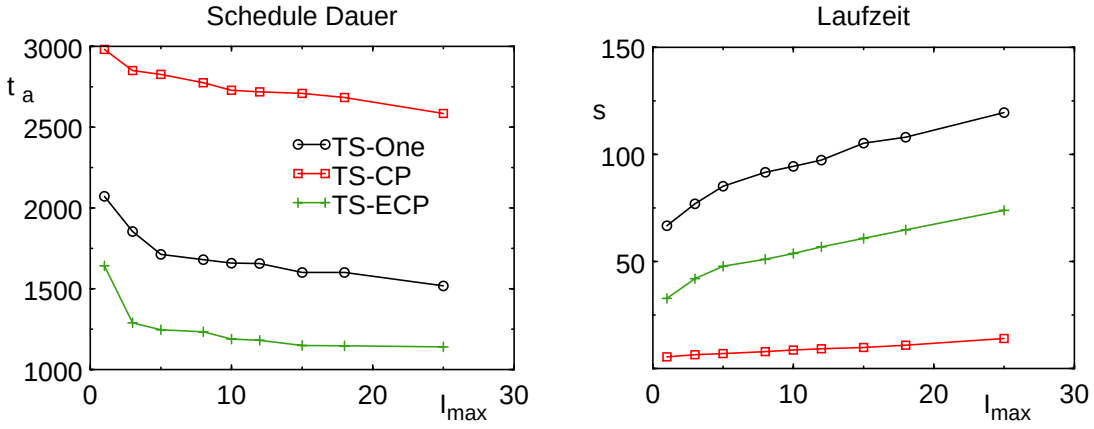


Bild 4-23: Einfluß der maximalen Anzahl von Iterationen (max. Initialmapping) bei der Tabu Suche

Im Vergleich zu den Ergebnissen für die minimale initiale Abbildung zeigen die Kurven der Schedule Dauer ähnliches Verhalten, jedoch tritt bei TS-CP und TS-One ein stärkerer Rückgang auf. Zwei wesentliche Unterschiede sind jedoch festzustellen: Alle Verfahren mit Ausnahme von TS-ECP liefern Ergebnisse mit erheblich höherer Schedule Dauer als bei minimaler Anfangs-Abbildung. Daneben führt bei der ungünstigen Anfangssituation TS-One zu Lösungen mit wesentlich schlechteren Ausführzeiten als TS-ECP. Die Erweiterung des kritischen Pfads um die Verzögerung verursachenden Prozesse ermöglicht damit eine wesentlich zielgerichtetere und damit wirkungsvollere Suche als andere Nachbarschaftsdefinitionen. Die Laufzeit der Suche nimmt wie im vorherigen Fall mit zunehmender Tabu FIFO- Größe ebenfalls linear zu, bewegt sich aber auf wesentlich höherem Niveau.
Neben den Untersuchungen zur maximalen Anzahl von ohne Verbesserung durchsuchten Nachbarschaften (I_{max}) wurden wie oben der Einfluß der Größe des Tabu FIFO näher betrachtet. Die Ergebnisse sind dabei im Verhalten identisch mit denen bei minimalem Anfangsmapping und sind deshalb hier nicht abgebildet.

Aufgrund der hier erhaltenen Ergebnisse wird für die weiteren Arbeiten, die auf die Optimierung der Schedule Dauer abzielen, für die Tabu Suche von einer initialen Abbildung der Prozesse auf die Ressource ausgegangen, die jeweils die minimale Ausführdauer besitzt. Die Parameter der Tabu Suche werden dabei entsprechend den oben angegebenen Werten festgesetzt.

4.6 Diskussion und Einsatzbereich des Verfahrens

In diesem Kapitel wurde der Ablauf und die einzelnen Teilschritte des rekursiven Abbildungs- und Schedulingverfahren beschrieben und die Festlegung von einzelnen Parametern durchgeführt. Aufgrund des ins Auge gefaßten Anwendungsgebiets Paketverarbeitung kommt der Unterstützung von bedingten Verzweigungen im Graphen besondere Bedeutung zu. Diese wird im Scheduling-Schritt durch Berücksichtigung der gegenseitigen Exklusivität zweier Knoten im Graphen bewerkstelligt. Mit Hilfe eines verallgemeinerten Verfahrens zur Erkennung dieser Eigenschaft ist die hier entwickelte Methode für beliebige Graphen mit Kontrollabhängigkeiten ohne Beschränkung von bedingten Übergängen auf die Ebene von Basic Blocks geeignet.
Die lokale Suche beschränkt sich hierbei auf die Abbildung, das Scheduling wird in Abhängigkeit von der jeweiligen Abbildung der Knoten auf die Ressourcen durchgeführt. Eine Integration des Scheduling in die Nachbarschaftssuche wie in [122], die möglicherweise zu einer Reduktion der erforderlichen Iterationen führen könnte, ist wegen der Berücksichtigung der Kontrollabhängigkeiten hier nicht möglich.
Das Verfahren optimiert ausschließlich die Schedule Dauer, wobei jedoch unter Berücksichtigung der im Graphen enthaltenen, sich gegenseitig ausschließenden Pfade der ungünstigste Fall erfaßt wird. D.h. die gelieferte Information zur Zuordnung des Graphen auf die Zielarchitektur und der Schedule gilt für die Kombination von Bedingungen, die die ungünstigste Ausführungszeit liefern.
Das Verfahren ist dabei auf azyklische Graphen, d.h. Graphen ohne Schleifen, beschränkt. Besitzt eine Anwendung Schleifen, müssen diese bei der Generierung des Graphen aufgerollt werden. Im Sinne der Untersuchung des ungünstigsten Falles, ist dies möglich, wenn die Anzahl der Durchläufe nach oben begrenzt ist. Um die Effekte auf der Kommunikationsarchitektur erfassen zu können, muß bei der Generierung des Graphen-Modells die verwendete Speicherarchitektur berücksichtigt werden. Dies führt dazu, daß das funktionale Modell nicht vollständig von der Architektur unabhängig ist.
Bei der Berücksichtigung der Kommunikation wird von einem vereinfachten Kommunikationsprotokoll auf dem gemeinsamen Bus ausgegangen. Durch Erweiterungen ist es möglich, auch detailliertere Informationen wie Arbitrierungsverzögerungen oder maximale Burstgrößen zu berücksichtigen.
Die Methode unterstützt in der vorliegenden Version nur eine aktive Instanz eines Graphen, d.h. auf der Architektur kann nur die Verarbeitung eines Pakets erfaßt werden. Die Untersuchung von Architekturen zur parallelen und voneinander unabhängigen Prozessierung mehrerer Graphen und die Überlagerung der zugehörigen Prozesse auf der Architektur ist in der vorliegenden Fassung nicht möglich.
Das nachfolgende Kapitel beinhaltet die Untersuchung des Verfahrens unter verschiedenen Parametern, die zum einen den abzubildenden Graphen und zum andern die Zielarchitektur charakterisieren.

5 Verifikation

5.1 Vorgehensweise

5.1.1 Übersicht

Zur Verifikation der Methode werden zunächst künstlich erzeugte Graphen verwendet, die mit einem geeigneten Werkzeug reproduzierbar generiert werden und als Benchmark dienen. Sie werden für die Untersuchung der Leistungsfähigkeit des Ansatzes in Abhängigkeit verschiedener Parameter eingesetzt, die eine Abbildungs- und Scheduling-Aufgabe charakterisieren. Danach wird ein konkretes Anwendungsbeispiel aus dem Bereich der Datenkommunikation untersucht, um die Eignung des Verfahrens unter realen Verhältnissen verifizieren zu können. In beiden Fällen wird das neue rekursive Verfahren angewandt und mit konventionellen Methoden in Bezug auf die dabei für eine bestimmte Architektur erreichbare Performance verglichen. Die Vergleiche finden unter den gleichen Randbedingungen, d.h. für dieselbe Funktionalität (bzw. die jeweils identischen Graphen), auf derselben Architektur mit den identischen Ressourcen statt.
Hierbei ist anzumerken, daß ein Vergleich nur für die unterschiedlichen Suchmethoden möglich ist, die einen Teil des im Rahmen des in dieser Arbeit vorgestellten rekursiven Abbildungs- und Scheduling-Verfahrens darstellen. Eine alternative rekursive Methode, die ebenso Kontrollabhängigkeiten wie die vorgestellte interstützt, ist nicht bekannt. Die im Kapitel 4 vorgestellten Suchmethoden werden in den Rahmen des vorliegenden Verfahrens eingebaut und miteinander verglichen. Als zusätzliches Verfahren, das bedingte Übergänge in Graphen ermöglicht, wird die in [123] veröffentliche Methode, die jedoch auf einem konstruktiven Ansatz beruht, ebenso in den Vergleich mit einbezogen, um den Bezug zum Stand der Technik herzustellen. In den nachfolgenden Vergleichsangaben werden die damit erreichbaren Ergebnisse unter der Bezeichnung "Xie/Wolf" referenziert.

5.1.2 Vergleichsverfahren

Nachfolgend wird eine kurze Übersicht über die in den folgenden Kapiteln referenzierten Vergleichsverfahren gegeben. Wie bereits erläutert, gibt es bisher kein rekursives Verfahren zum Mapping und Scheduling von Graphen, das eine Unterstützung von Kontrollabhängigkeiten bietet. Insofern ist ein direkter Vergleich mit alternativen Ansätzen nur in sehr eingeschränktem Maße möglich.

FAST Suchverfahren

Um jedoch einen alternativen Ansatz zu der auf Tabu Suche basierenden neuen Methode als Referenz verwenden zu können und insbesondere die Vorteile der erweiterten Nachbarschaft des kritischen Pfads zu demonstrieren, wurde der in [70] beschriebene und in [69] aktualisierte Ansatz der statistischen und nicht vollständigen Suche im Rahmen des in Bild 4-14 dargestellten Ablaufs eingesetzt. Der Vergleich mit diesem im weiteren als "FAST" referenzierten Verfahren ist damit nur in Bezug auf das verwendete Suchverfahren aussa-

gekräftig.
Das in FAST verwendete Suchverfahren ist nachfolgend im Bild 5-1 als Pseudo-Code wiedergegeben.

```
read in initial mapping;
best_sched_dur = infinity;
min_sched_dur = infinity;
searchcount = 0;
do{
     searchstep = 0; counter = 0;
     do{
          pick node n randomly
          pick resource r randomly
          move n to r
          calculate schedule
          if(min_sched_dur < sched_dur)
               move n to previous resource
               counter++;
          else
               min_mapping = curr_mapping;
               min_sched_dur = sched_dur;
               counter = 0;
     } while(searchstep++ < MAXSTEP && counter < MARGIN);
     if(best_sched_dur > min_sched_dur){
          best_mapping = min_mapping;
          best_sched_dur = min_sched_dur;
     }
     pick random node from critical path
     remap node to random resource
}while(searchcount++ < MAXCOUNT);
```

Bild 5-1: Der FAST Suchalgorithmus

Die Suche bei FAST erfolgt in zwei verschachtelten Schleifen, mit denen das Umfeld des kritischen Pfades abgesucht wird. Die Anzahl der Durchläufe ist zum einen durch eine feste Vorgabe der Parameter MAXCOUNT = 64 und MAXSTEP = 8 nach oben begrenzt. Zum anderen wird die Suche in der inneren Schleife durch den Parameter MARGIN = 2 vorzeitig beendet, wenn keine bessere Lösung gefunden wird. Die Werte für diese Parameter wurden in [69] auf die oben genannten Werte festgelegt.

Konstruktives Verfahren nach Xie/Wolf

Als alternative Methode zum rekursiven Scheduling und Mapping eines Prozeß-Graphen wird in den nachfolgenden Experimenten ein aktueller Ansatz, der die Lösung durch einmaliges Abarbeiten des Graphen konstruktiv generiert, als weitere Vergleichsmethode aufgelistet. Es handelt sich dabei um den Ansatz von Xie und Wolf, der in [123] veröffentlicht wurde. Dieser Ansatz unterstützt ebenfalls Kontrollabhängigkeiten, jedoch sind hierbei Einschränkungen in Bezug auf die im Graphen erlaubten bedingten Übergänge gegeben,

die im Kapitel 4.3 bereits erläutert wurden. Um dieselben Graphen wie für die rekursive Methode verwenden zu können, wurde deshalb das originale Verfahren um die verallgemeinerten Funktionen zur Detektion der gegenseitigen Exklusivität erweitert, wie sie in der rekursiven Vorgehensweise zum Einsatz kommen.
Im ursprünglichen Algorithmus von Xie und Wolf wurde die Abbildung von Prozessen auf HW-Ressourcen nicht von diesem Algorithmus festgelegt, sondern als bereits vorgegeben betrachtet. Die Transfers zu HW Ressourcen wurden gegenüber Transfers zwischen Prozessoren bevorzugt. Diese Annahme wurde im hier verwendeten Vergleichsalgorithmus beseitigt und durch eine gleichartige Behandlung aller Ressourcen und Prozesse bzw. Transfers ersetzt. Daneben wurden die dort benutzten einheitlichen, einer Zeiteinheit entsprechenden Transfers durch die den tatsächlichen Datenmengen entsprechenden Transferzeiten ersetzt, sowie beim Scheduling die Suche nach freien Zeitbereichen optimiert.
Das in den genannten Punkten erweiterte Verfahren wird nachfolgend trotz der Modifikationen als "Xie/Wolf" referenziert, da am zugrundeliegenden Ansatz keine Änderung vorgenommen wurde. Es soll jedoch betont werden, daß die Ergebnisse nicht mit dem originalen Xie/Wolf Algorithmus sondern mit einer verallgemeinerten und an die betrachtete Problemstellung angepaßten Fassung erzielt wurden. Der referenzierte Algorithmus ist nachfolgend in Pseudo-Code Notation angegeben.

```
for each task task_i
     calculate static_urgency(task_i)
calculate ready list
repeat{
     for each ready task task_i and each CPU pe_j
          calculate dynamic_urgency(task_i, pe_j)
     pick task_i on pe_j with maximium dynamic_urgency
     schedule task_i on pe_j
     update ready task list
} until all tasks are scheduled
```

Bild 5-2: Der Algorithmus von Xie/Wolf

Das Verfahren basiert auf dem List Scheduling Prinzip und nutzt eine statische (static urgency) und eine dynamisch angepaßte (dynamic urgency) Prioritätsdefinition zur Auswahl derjenigen Prozeß-Ressourcen-Kombination, die die frühestmögliche Beendigung des jeweiligen Prozesses ermöglicht. Dies entspricht einer oft in der Literatur als "Greedy Heuristic" bezeichneten Vorgehensweise.

Nachfolgend wird auf die Vorgehensweise zur Durchführung von Experimenten mit den verschiedenen Algorithmen beschrieben, bevor in Kapitel 5.2.4 die Ergebnisse der Untersuchungen dargestellt werden.

5.2 Synthetische Graphen und Architekturdaten

Um die Leistungsfähigkeit des Verfahrens zu untersuchen, ist es erforderlich, eine größere Anzahl von Abbildungs- und Scheduling-Problemen zur Verfügung zu haben und diese mit dem neuen Verfahren sowie mit entsprechenden Vergleichsmethoden zu lösen. Eine manuelle Erzeugung einer ausreichenden Anzahl von Graphen, die statistisch abgesicherte Aussagen über die Qualität der von den unterschiedlichen Methoden gelieferten Ergebnisse erlauben, ist nicht möglich. Hinzu kommt, daß Graphen durch eine Große Anzahl von Parametern charakterisiert werden können, die durch eine begrenzte Anzahl von manuell generierten, mit realen Anwendungen hinterlegten Graphen nicht ausreichend abgedeckt werden können. Somit wären nur sehr eingeschränkte Aussagen über die Leistungsfähigkeit des neuen Verfahrens möglich.
Aus diesem Grund ist eine Möglichkeit zur Erzeugung von künstlichen Graphen erforderlich, die es erlaubt, schnell und insbesondere reproduzierbar eine große Anzahl von Graphen zu Testzwecken zu erzeugen. Eine zusätzliche Anforderung besteht darin, daß die dabei generierten Graphen in ihren Eigenschaften ausreichend differenziert werden können, um die Untersuchungen abhängig von diesen Eigenschaften durchführen zu können. Damit ist eine Charakterisierung der Graphen in verschiedene Klassen möglich. Aus den Tests mit diesen Graphen lassen sich die Ergebnisse besser eingrenzen und spezifischere Aussagen zu den betrachteten Verfahren und ihrer Abhängigkeit von den zugehörigen Parametern treffen.
Neben dem Graphen als Input für die Mapping- und Scheduling-Aufgabe sind zusätzlich auch die Eigenschaften der Architektur relevant, auf die der Graph abgebildet werden soll. Auch hierfür gilt, daß deren Eigenschaften parametrisiert und für die Tests mit unterschiedlichen Werten hinterlegt werden können, um die Güte der Methoden unter diesem Gesichtspunkt beurteilen zu können.
Nachfolgend werden die verwendeten Parameter zur Charakterisierung von Graphen und Zielarchitekturen erläutert, und die synthetische Generierung des von den Verfahren benötigten Inputs beschrieben.

5.2.1 Erzeugung der synthetischen Testdaten

Zunächst sollen die Parameter synthetischer Graphen dargestellt werden, die als Ersatz für reale Anwendungen in größerer Anzahl schnell generiert und als Benchmark zur Bewertung des Verfahrens verwendet werden können. Als Voraussetzung zur künstlichen Erzeugung geeigneter Graphen sollen deshalb zunächst grundlegende Eigenschaften von Graphen mit Kontrollabhängigkeiten beschrieben werden, bevor nachfolgend auf deren synthetische Generierung eingegangen wird. Da die Graphen konkreter Anwendungsbeispiele wegen ihrer Heterogenität und insbesondere auch der möglichen unterschiedlichen Abstraktion nicht mit einer begrenzten Anzahl von Parametern einfach beschrieben werden können, ist die Verwendung derartiger Graphen lediglich als ein erstes Hilfsmittel zur Bewertung zu sehen. Die damit mögliche Durchführung vieler Tests mit unterschiedlichen Werten für die einzelnen Parameter gibt jedoch starke Hinweise über die Güte der Verfahren, wenn sich daraus konkrete Trends erkennen lassen. Darauf wird im weiteren bei der

Durchführung der Tests hingewiesen. Andererseits ist eine vollständige Überprüfung aller Parameterkombinationen aus Aufwandsgründen nicht möglich. Deshalb schließt sich an die Untersuchungen von künstlichen Graphen die Betrachtung einer konkreten Anwendung an, die zur Beurteilung der entsprechenden Trends dient.

Eigenschaften der Graphen

Zur Charakterisierung von künstlichen Graphen kommen in erster Linie folgende Eigenschaften in Frage:

- Anzahl der Knoten
- Konnektivität der Knoten, Struktur des Graphen
- Datenmenge, die zwischen Knoten transferiert wird
- Anteil der bedingten Übergänge

Diese Eigenschaften sind dahingehend zu unterscheiden, inwieweit sie durch geeignete Parameter beschrieben werden können, die zur Reproduktion unterschiedlicher Exemplare aus der gleichen Klasse von Graphen verwendet werden können. Die Struktur des Graphen ist hierbei eine Eigenschaft, die einen Graphen entscheidend bestimmt. Jedoch ist es sehr schwierig, dies in Form von numerischen Parametern zu schreiben. Als mögliche Kandidaten hierfür können folgende Punkte in Erwägung gezogen werden:

- Anzahl der Startknoten
- Anzahl der Endknoten
- mittlere Anzahl von abgehenden Übergängen pro Knoten
- mittlere Anzahl von ankommenden Übergängen pro Knoten

Im Hinblick auf das Anwendungsgebiet Paketverarbeitung wird angenommen, daß es sich um polare Graphen handelt, die nur jeweils einen Start- und Endknoten besitzen. Diese Eigenschaft ist zum einen leicht durch Einführung eines zusätzlichen Knotens am Anfang bzw. am Ende des Graphen erreichbar. Zum anderen ist diese Annahme auch in Bezug auf das betrachtete Anwendungsgebiet realistisch. Dabei beginnt die Bearbeitung eines Pakets mit dessen Ankunft bzw. mit der Bereitstellung der Daten des Paketheaders und endet mit dem Speichern der Information, wie das Paket behandelt werden muß, oder mit der Weiterleitung des Pakets selbst. Diese Ereignisse zu Beginn und am Ende der Verarbeitung können jeweils als ein Knoten betrachtet werden.
Die Mittelung der Anzahl der Übergänge, die an einem Knoten ankommen bzw. davon abgehen, führt zu Graphen, die tendenziell homogen sind. D.h. eine Struktur, die in verschiedenen Bereichen des Graphen eine unterschiedliche Konnektivität der Knoten besitzt, ist damit nicht machbar. Für den gedachten Verwendungszweck der künstlichen Graphen zum Vergleich der Verfahren ist dies jedoch keine entscheidende Einschränkung. Im Zusammenhang mit dem verwendeten Werkzeug zur Erzeugung der Graphen ([25]) wird die Verwendung derartiger Graphen zu Vergleichszwecken dargelegt.

Mit Hilfe dieser Parameter lassen sich Klassen von Testgraphen definieren, für die jeweils eine beliebige Anzahl von Graphen erzeugt werden können.

Eigenschaften der Architektur

Neben der in Form eines Graphen gegebenen Funktion ist die Architektur der zweite wichtige Input für das Verfahren. Die Architektur kann durch folgende Parameter beschrieben werden.

- Struktur der Architektur
- Kommunikationsarchitektur
- Anzahl der Blöcke/Blocktypen in der Architektur (Ressourcen)
- Verarbeitungszeiten der Teilfunktionen auf den einzelnen Ressourcen
- Implementierbarkeit von Prozessen auf den Ressourcen

Für die Durchführung der Tests mit künstlichen Graphen ist die Generierung der zugehörigen Informationen für die Architektur, d.h. insbesondere die Ausführungsdauer der einzelnen Prozesse (Knoten) auf den Ressourcen der Architektur erforderlich. Da dies eng mit den im Graphen vorhandenen Knoten zusammenhängt, ist eine gemeinsame Generierung mit dem Graphen sinnvoll.
Wie oben bereits erläutert, wurde das Verfahren unter der Randbedingung einer Zielarchitektur mit einem gemeinsamen Bus entwickelt, so daß für die weiteren Untersuchungen von den genannten Eigenschaften lediglich die Anzahl der unterschiedlichen Ressourcen und die zu den jeweiligen Prozessen gehörenden Ausführungszeiten der einzelnen Ressourcen verbleiben. Zu der letztgenannten Eigenschaft gehört auch der Anteil von Prozessen, der auf dem jeweiligen Ressourcen-Typ überhaupt realisiert werden kann. Bei Prozessoren ist dies üblicherweise jede Aufgabe (mit entsprechenden Performance-Nachteilen), während Beschleuniger üblicherweise sehr wenige Funktionen implementieren können.

Erzeugung von synthetischen Graph- und Architekturdaten

Zur Generierung von Graphen, die als Vergleichsmaßstab für unterschiedliche Verfahren aus dem Bereich der Mapping und Scheduling dienen, wurde das an der Princeton University entwickelte Werkzeug "Task Graphs for Free" (TGFF, [25], erhältlich unter [26]) verwendet. TGFF erfüllt die vorstehend genannten Anforderungen und erlaubt die konfigurierbare und reproduzierbare Erzeugung von pseudo-zufälligen Prozeß-Graphen sowie von Daten, die die Architektur und die darin eingesetzten Blöcke charakterisieren.
Durch entsprechende Parameter, die über eine Steuerdatei dem Werkzeug mitgeteilt werden, kann die Wirkungsweise von TGFF beeinflußt werden. Durch die Vergabe von Seed-Werten für die in TGFF verwendeten Zufallsgeneratoren lassen sich die Eingangsdaten zur Wiederholung der Testläufe reproduzieren bzw. unterschiedliche Exemplare mit gleichen Eigenschaften erzeugen. Der Seed-Wert beeinflußt sowohl die Struktur als auch die sonstigen den Graphen beschreibenden Daten. TGFF besitzt eine breite Palette von Optionen, die über das im Rahmen der vorliegenden Arbeit Notwendige hinausgeht. Andererseits wurden

einige Erweiterungen vorgenommen, die erst die Implementierung einiger der oben genannten Eigenschaften erlauben.
Für die Untersuchungen der Abbildungs- und Partitionierungsverfahren waren im wesentlichen die flexible Erzeugung eines Graphen und der zugehörigen Architekturparameter (Ressourcen Library) erforderlich. Da andererseits einige besondere Anforderungen nicht direkt in TGFF erfüllbar waren, wurde dies in separaten Programmen bzw. durch kombinierte Auswertung unterschiedlicher von TGFF gelieferter Parameter bewerkstelligt. Dies wird nach der kurzen Beschreibung der Verwendung von TGFF erläutert.
Hauptpunkt bei der Generierung der Graphen ist in TGFF die Erzeugung der Graph-Struktur, die sich über die Anzahl der Knoten sowie Angaben zu deren Konnektivität ergibt. Ausgehend von einem Knoten werden rekursiv weitere Knoten ein- oder ausgangsseitig hinzugefügt bis die gewünschte Knotenzahl erreicht ist. Z.B. wird für den ausgangsseitigen Fall zufällig ein Knoten aus der Menge von Knoten gewählt, bei denen die Differenz x zwischen der erlaubten und der tatsächlichen Anzahl von abgehenden Übergängen maximal ist. An den gewählten Knoten wird dann ausgangsseitig eine zufällige Zahl y von Knoten (y aus dem Bereich $[0;x]$) angefügt. Entsprechendes gilt auch für die eingangsseitige Erweiterung des Graphen. Auf diese Art und Weise wird die Struktur des Graphen bestehend aus Knoten und Kanten bestimmt. Ergänzend hierzu können in Form von sog. Tabellen beliebige zusätzliche, auch durch mehrere Parameterwerte beschriebene pseudo-zufällige Attribute generiert werden. Die Festlegung der Art und Struktur der benötigten Daten und insbesondere deren Interpretation bzw. Verwendung bleibt dem Anwender überlassen.
TGFF ermöglicht die Festlegung von Parametern des Graphen und der Architektur zum Teil über fest vorgegebene Werte aber auch durch statistische Größen. Statistische Parameter sind gleichverteilt und werden u.a. durch Angabe des Mittelwertes und der halben Intervallbreite spezifiziert. Die im Rahmen der Untersuchungen relevanten, mit geeigneten Attributen modellierbaren Parameter sind in erster Linie die zu den Übergänge gehörenden Datenmengen sowie die Machbarkeit und Ausführungsdauern der Prozesse auf unterschiedlichen Ressourcen. Weitere Informationen zu Eigenschaften und Benutzung von TGFF sind in der zugehörigen Dokumentation und in [25] enthalten.
Für die konkrete Verwendung von TGFF zur Generierung der Testdaten für die Graphen werden folgende Parameter verwendet:

- Anzahl der Knoten (n)
- Anzahl der maximal an einem Knoten ankommenden (id) bzw. davon abgehenden Kanten (od)
- Datenmenge, die zwischen den Knoten transferiert wird

Zur Anzahl der tatsächlich erzeugten Knoten im Graphen ist anzumerken, daß sie gemäß obiger Beschreibung nicht exakt bei n liegt sondern aufgrund der oben beschriebenen Methode der Graph-Erzeugung im Bereich $[n;n+od-1]$. Da die Werte für od relativ gering gewählt werden (5 für kleine, 10 für große Graphen ab 100 Knoten), ist die Abweichung vernachlässigbar. Wenn nachfolgend bestimmte Graphgrößen benannt werden, ist diese geringe Abweichung zu berücksichtigen und lediglich eine Größenklasse gemeint.

Als Einstellmöglichkeiten für die Eigenschaften der Architektur werden mit Hilfe von TGFF folgende Parameter direkt benutzt:

- Anzahl der Ressourcen pro Ressourcenklasse
- Art der Ressourcen (Ressourcenklasse)
- die jeweiligen Worst Case Ausführungszeiten der einzelnen Prozesse auf diesen Ressourcen

TGFF fehlen in der verwendeten Version einige der benötigten Eigenschaften, die eine Erweiterung für den hier betrachteten Anwendungsfall notwendig machen. So können nur unbedingte aber keine bedingten Übergänge zwischen einzelnen Prozessen generiert werden. Deshalb wurde aus dem von TGFF gelieferten Graphen ein vorgebbarer Prozentsatz von Knoten mit bedingten Kanten versehen. Von den Kanten, die vom jeweiligen Knoten abgehen, wurde ein ebenfalls einstellbarer Anteil in bedingte, gegenseitig exklusive Kanten umgeändert. Die Auswahl der Knoten und der davon abgehenden Kanten wurde über einen Zufallsgenerator vorgenommen.
Weiterhin kann TGFF unterschiedliche Verarbeitungsressourcen nicht ausreichend in Bezug auf die darauf implementierbaren Prozesse differenzieren. Für die im Rahmen dieser Arbeit betrachteten SoC Architekturen sind jedoch neben eingebetteten Prozessoren, die im Prinzip alle Funktionalitäten mehr oder weniger effizient implementieren können, auch Beschleunigerblöcke notwendig, die einen sehr eingeschränkten Funktionalitätsumfang besitzen. Deshalb wurde die Erzeugung der Ausführungszeiten um eine Option erweitert, die bei der Umwandlung in das für das Mapping- und Schedulingwerkzeug geeignete Format die Implementierbarkeit der einzelnen Prozesse auf der jeweiligen Ressource extrahierbar macht. Durch Vorgabe geeigneter Werte kann die Machbarkeit der Prozesse des Graphen für die jeweiligen Ressourcenklassen auf jeweils vorgebbare Anteile eingestellt werden.

5.2.2 Randbedingungen der Simulationen

Die nachfolgend wiedergegebenen Ergebnisse wurden durch Simulation von jeweils 60 Graphen mit den angegeben Eigenschaften und Bildung des Mittelwerts gewonnen. Da es sich bei der Tabu Suche um ein deterministisches Verfahren handelt, wurde jeder dieser 60 Graphen nur einmal bearbeitet und das Ergebnis und die notwendige Laufzeit registriert.
Für die Vergleichsverfahren, bei denen die Suche statistisch abläuft, ist eine genauere Betrachtung erforderlich. Da der Zufallsgenarator über die erzeugte Serie von Zufallszahlen den Ablauf der Suche direkt beeinflußt, ist eine mehrfache Abarbeitung eines Graphen mit unterschiedlichen Seed-Werten notwendig. Die Anzahl der Suchläufe mit verschiedenen Folgen von Zufallszahlen, aus denen derjenige mit der kürzesten Schedule Dauer die Lösung liefert, soll nachfolgend als n_{run} bezeichnet werden. Andererseits steigt mit n_{run} der insgesamt notwendige Berechnungsaufwand und die effektiv notwendige Laufzeit des Verfahrens an. Es ist also eine Abwägung zwischen der Güte des Ergebnisses und der hier-

für erforderlichen Laufzeit erforderlich. Aus diesem Grund soll vorab eine Untersuchung von n_{run} durchgeführt werden.

Nachfolgend wird das FAST Verfahren für Graphgrößen von 50, 100 und 200 Knoten untersucht, wobei eine Architektur bestehend aus 2 Prozessoren und 5 ASIC Beschleunigern angenommen wird, die im Mittel die Prozesse um den Faktor 5 gegenüber den Prozessoren beschleunigen und die 25% der Prozesse ausführen können. Um den Einfluß des Anteils bedingter Knoten ebenfalls zu erfassen, wurden für die genannten Größen der Graphen jeweils Anteile von 0%, 10%, 20% und 30% betrachtet.

(Neben den gezeigten Ergebnissen der Untersuchung mit den oben genannten Parametern wurden weitere Testläufe mit anderen Parameterwerten durchgeführt, die die gezeigten Ergebnisse grundsätzlich bestätigt haben, hier jedoch nicht angegeben sind.)

Für jede Graphgröße und jeden Anteil bedingter Knoten wurden 60 unterschiedliche Graphen generiert, für die in jeweils n_{run} Läufen mit unterschiedlichem Startwert für den Zufallsgenerator die Abbildung und das Scheduling mit Hilfe von FAST erzeugt wurde. Von diesen n_{run} Läufen für den einzelnen Graphen wurde als Endergebnis jeweils der Minimalwert der Schedule Dauer und als Laufzeit die Summe der n_{run} Läufe verwendet. Das Ergebnis für einen bestimmten Wert von n_{run} für die jeweilige Graphgröße bzw. den Anteil bedingter Knoten wurde jeweils aus der Mittelung der 60 verschiedenen Graphen gebildet. Da die bei dieser Vorgehensweise resultierenden Ergebnisse unabhängig vom Anteil bedingter Knoten waren, wurden alle Experimente mit der gleichen Graphgröße, unabhängig vom Anteil bedingter Knoten, zusamemengefaßt. Das linke Diagramm im Bild 5-3 zeigt die relative Verbesserung der weiteren Läufe im Vergleich zum Ergebnis des ersten Suchlaufs in Abhängigkeit von n_{run} für unterschiedliche Graphgrößen. Rechts ist die schrittweise Verbesserung gezeigt, die der Differenz zwischen nebeneinanderliegenden Verbesserungswerten entspricht.

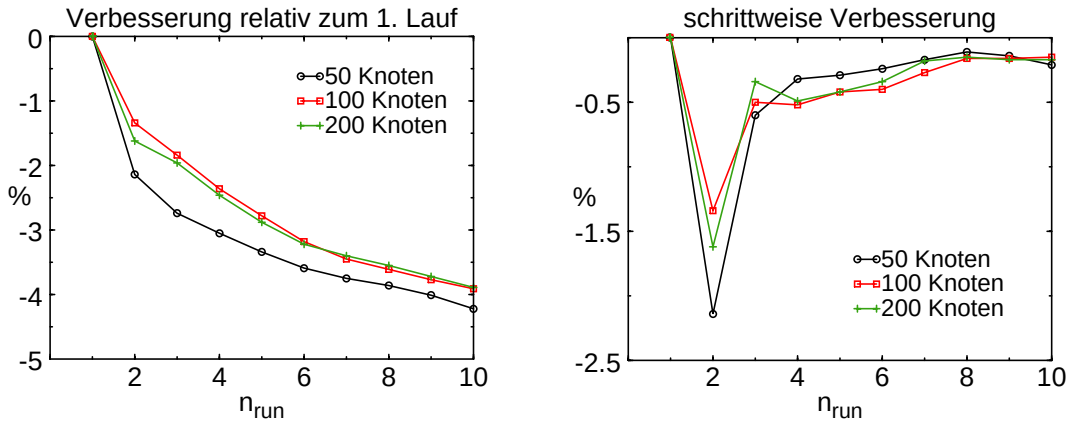


Bild 5-3: Verbesserung des Ergebnisses bei FAST abhängig von der Anzahl der Suchläufe

Zunächst ist festzustellen, daß die Güte des Ergebnisses stark von der Anzahl der durchgeführten Läufe abhängt, jedoch die mit weiteren Läufen gewonnene Verbesserung zuneh-

mend geringer wird. Die betrachteten Graphgrößen unterscheiden sich dabei nur unwesentlich in ihrem Verhalten.
Da FAST die Knoten aus der Nachbarschaft und auch in der übergeordneten Schleife aus dem kritischen Pfad zufällig wählt, wird eine relativ lückenhafte Suche ausgeführt, bei der die Sequenz der Zufallszahlen großen Einfluß besitzt. Durch mehrfaches Durchführen der Suche mit unterschiedlichen Sequenzen wird eine bessere Abdeckung des Lösungsraums erreicht, wodurch bessere Lösungen gefunden werden. Dieser Effekt nimmt mit zunehmender Anzahl von Läufen ab.
Diesen Verbesserungen muß die Zunahme des Rechenaufwandes und der damit verbundenen Laufzeit gegenübergestellt werden. Die jeweils notwendige Laufzeit für die mehrfachen Simulationen steigt linear mit zunehmendem n_{run}. Berechnet man daraus die auf den Berechnungsaufwand bezogene Verbesserung (aufwandsbez. Verbesserung = Verbesserung / n_{run}), ergibt sich die in Bild 5-4 gezeigte Abhängigkeit, wo zum Vergleich auch die mittlere schrittweise Verbesserung eingetragen ist. (Beide Kurven stellen dabei die Mittelwerte über die drei unterschiedlichen Graphgrößen dar.)

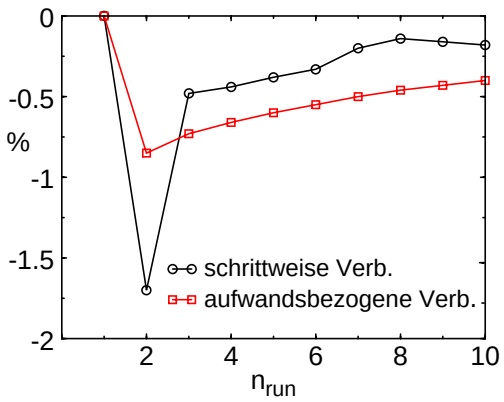


Bild 5-4: Verbesserung bei FAST durch mehrfache Läufe unter Berücksichtigung des Aufwands

Aufgrund des starken Rückgangs der einzelnen Verbesserung bei Erhöhung der durchgeführten Versuche geht der auf den linear zunehmenden Berechnungsaufwand bezogene Verbesserungseffekt ebenfalls stark zurück. Die erreichten Vorteile stammen vor allem aus den ersten Versuchen, die noch zu größeren Rückgängen in der Schedule Dauer geführt haben. Der zunehmend geringer werdende Gewinn an Schedule Dauer kann den Aufwand jedoch nicht mehr aufwiegen, so daß eine Begrenzung von n_{run} erforderlich ist.
In den weiteren Untersuchungen wird deshalb als Kompromiß zwischen der Verbesserung des Ergebnisses und der linearen Zunahme der dazu erforderlichen Laufzeit jeder Graph von FAST mit jeweils 5 unterschiedlichen Läufen untersucht, wobei der jeweils beste ausgewählt wird. Als die dafür notwendige Laufzeit wird dabei jeweils die Summe aller 5 Läu-

fe angegeben.

5.2.3 Architekturannahmen

Wie weiter oben bereits erläutert, wird von einer Architektur mit einem gemeinsam genutzten Bus ausgegangen, über den die gesamte Kommunikation abgewickelt wird. An diesem Kommunikationsmedium sind jeweils eine beliebige Anzahl von Prozessoren und Beschleunigerblöcken angebunden. Die Prozessoren sind mittels TGFF so modelliert, daß sie alle Prozesse abarbeiten können, jedoch mit geringerer Performance als die Beschleuniger. Diese besitzen hingegen einen geringeren Umfang der unterstützten Funktionalität, sind aber im Mittel um einen vorgebbaren Faktor schneller als die Prozessoren.

Der gemeinsame Bus wird durch seine Wortbreite W und die zugehörige Taktrate f_B modelliert. Zusammen mit den Datenmengen D, die den einzelnen Transfers zugeordnet sind, ergibt sich daraus jeweils die zugehörige Dauer $T_{trans,p}$ für die Übertragung über den Bus ohne Berücksichtigung einer ggf. vorhandenen Wartezeit.

$$T_{trans,\,p} = \left\lceil \frac{D}{W} \right\rceil \cdot \frac{W}{f_B} \tag{5.1}$$

Dabei bedeutet $\lceil x \rceil$ die nächstgrößere ganze Zahl y mit $y \geq x$.
Durch Einbeziehung eines bestimmten Offsets T_{arb} kann ein ggf. erforderlicher Arbitrierungsoverhead für jeden Transfer berücksichtigt werden.

$$T_{trans,\,a} = T_{trans,\,p} + T_{arb} = \left\lceil \frac{D}{W} \right\rceil \cdot \frac{W}{f_B} + T_{arb} \tag{5.2}$$

Als weiteres Charakteristikum ist prinzipiell auch die Berücksichtigung einer maximalen Anzahl s_{Burst} von Worten möglich, die in einem Burst auf dem Bus übertragen werden können. Hierbei muß ein Kommunikationsknoten in $n = \left\lceil \frac{D}{s_{Burst} \cdot W} \right\rceil$ einzelne Kommunikationsknoten unterteilt werden, wovon n-1 dieser Knoten ohne Berücksichtigung der Arbitrierung eine Dauer von

$$T_{trans,\,b} = s_{Burst} \cdot \frac{W}{f_B} \tag{5.3}$$

und der letzte eine Dauer von

$$T_{trans,\,r} = \left\langle \left\lceil \frac{D}{W} \right\rceil - \langle n-1 \rangle \right\rangle \cdot \frac{W}{f_B} \tag{5.4}$$

besitzen.

Beide Faktoren, Arbitrierungsverzögerungen und maximale Burst-Größe, sind in der bisherigen Fassung noch nicht enthalten, können durch entsprechende Erweiterungen aber

leicht in die Methode eingebaut werden. Für die Untersuchungen auf Systemebene, für die die vorgestellte Methode gedacht ist, wird dieser Detaillierungsgrad jedoch als nicht erforderlich betrachtet.

5.2.4 Ergebnisse

Nachfolgend werden die Ergebnisse von Experimenten mit dem neuen rekursiven Verfahren vorgestellt. Zum einen wird dabei Tabu Suche in einer breit definierten Nachbarschaft (One) sowie in Nachbarschaften eingesetzt, die über den konventionellen kritischen Pfad (CP) bzw. über den, um Verzögerungen erzeugende Prozesse erweiteren, kritischen Pfad (ECP) definiert sind. Zum anderen wird als alternatives Suchverfahren FAST berücksichtigt. Diesen Verfahren wird zum Vergleich eine aktuelle konstruktive Methode (Xie/Wolf) gegenübergestellt.
Die unterschiedlichen Verfahren bzw. ihre Varianten werden dabei auf künstliche, mit TGFF erzeugte Graphen unter Verwendung der zugehörigen Ressourcenbibliothek angewandt. Die in Kapitel 5.2.1 beschriebenen Parameter werden variiert und ihr Einfluß auf die Ergebnisse untersucht. Da die Modelle zu viele Freiheitsgrade besitzen, um deren Einfluß in allen Kombinationen einerseits vollständig zu simulieren und andererseits auch in geeigneter Weise darzustellen, werden sie jeweils nur in einem eingeschränkten Wertebereich untersucht. Um die Einflüsse einzelner Faktoren festzustellen, werden deshalb jeweils nur einzelne Parameter variiert, während die übrigen auf bestimmte Werte festgesetzt werden.
Alle nachfolgenden Untersuchungen wurden, sofern keine anderen Angaben beim jeweiligen Test angegeben sind, anhand einer Konfiguration mit folgenden Randbedingungen durchgeführt, die im weiteren als Standardproblem referenziert wird:

a) Graph-Größe: 100 Knoten
b) Anteil bedingter Knoten: 10%
c) Mittlere transferierte Datenmenge: 50 Bits
d) Ressourcen: 2 Prozessoren, 5 Beschleuniger (ASIC Blöcke)
e) Beschleunigungsfaktor durch ASIC-Blöcke: 5
f) Anteil der auf Beschleunigern implementierbaren Prozesse: 25%

In den nachfolgenden Graphiken sind keine absoluten Angaben zur Schedule Dauer angegeben, da diese aufgrund der synthetischen Erzeugung keine direkte Aussagekraft besitzen. Im Hinblick auf die direkte Vergleichbarkeit werden deshalb jeweils relative Werte zu FAST als dem rekursiven Vergleichsverfahren angegeben. Aus Gründen der Übersichtlichkeit werden in den Graphiken lediglich die sich aus den Ergebnissen der jeweils untersuchten 60 Graphen resultierenden Mittelwerte dargestellt. Im Anhang sind zu den Mittelwerten auch die jeweiligen Werte für die Standardabweichung in tabellarischer Form aufgelistet.
Die Laufzeiten werden hingegen als absolute Werte wiedergegeben, um einen realen Eindruck über den Berechnungsaufwand zu vermitteln. Diese Zeitangaben beziehen sich auf einen Linux PC mit einem AMD Athlon XP 1700+. Als Compiler wurde dabei *gcc* mit der Optimierungsoption *O3* verwendet.

5.2.4.1 Eigenschaften der Graphen

Größe der Graphen

Zunächst wird der Einfluß der Größe des abzubildenden Graphen betrachtet. Wie sich in Kapitel 4.5.3 bei der Auswahl des Optimierungsalgorithmus bereits gezeigt hat, stellt die Graph-Größe den wichtigsten Parameter bei der Untersuchung dar. Im folgenden sind die Ergebnisse für das Standardproblem mit zunehmender Graphgröße angegeben.

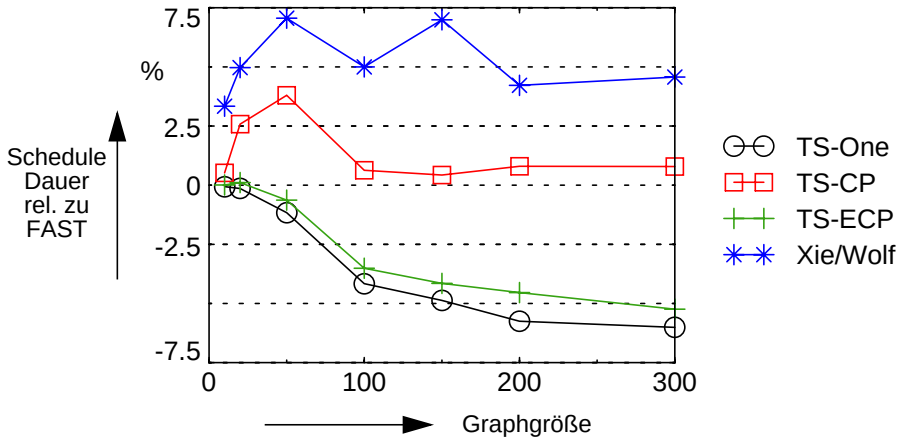


Bild 5-5: Schedule Dauer in Abhängigkeit von der Graphgröße

Bei FAST wird die Suche über die maximal vorgegebene Anzahl der Durchläufe der beiden verschachtelten Suchschleifen nach oben fest begrenzt, was die erreichbare Performance mit zunehmender Graphgröße limitiert. Aufgrund der Beschränkung der Suche wird von FAST ein zunehmend geringerer Anteil des Lösungsraums untersucht. Dies wirkt sich bei FAST insbesondere in der äußeren Schleife aus, bei der Knoten des kritischen Pfades alternativ abgebildet werden. Durch die zufällige Auswahl einer maximalen Anzahl von alternativen Abbildungen wird ein zunehmend geringer werdender Anteil des kritischen Pfades überprüft.

Da bei der Tabu Suche eine derartige Begrenzung nicht vorgenommen wird, sondern eine vorgegebene Anzahl von vollständigen Nachbarschaftssuchen ohne Verbesserung des Ergebnisses als Abbruchkriterium dient, hat dies eine bessere Abdeckung des Lösungsraums zur Folge.

Im Fall der Suche in der CP-Nachbarschaft führt die Vergrößerung des Graphen zur geringsten, in etwa logarithmischen Zunahme der Nachbarschaftsgröße, wodurch sich ein ähnlicher Effekt wie bei FAST ergibt. Da bei TS-CP ausschließlich Knoten des kritischen Pfades alternativ zugeordnet werden, und von der initialen Lösung mit jeweils minimaler Ausführungszeit der einzelnen Knoten ausgegangen wird, kann der damit verbundene Bereich der Lösungsalternativen nicht mehr verlassen werden, wodurch sich die relativ ungünstige Leistungsfähigkeit erklärt. Dies ist auch in der Untersuchung in Kapitel 4.5.4 zur Fähigkeit der Annäherung an die optimale Lösung zu erkennen, bei denen die CP Variante die schlechtesten Ergebnisse aller 3 Varianten der Tabu-Suche liefert.

Die breit gefaßte One-Nachbarschaft besitzt eine linear ansteigende Nachbarschaftsgröße. Bei ECP vergrößert sie sich mehr als linear mit zunehmender Graphgröße, liegt jedoch im betrachteten Bereich bei maximal 60% (bei 300 Knoten) des Wertes von One. Diese Zunahme bei der Nachbarschaftsgröße führt zusätzlich auch zu einer Erhöhung der Iterationszahl und ermöglicht damit, den vom aktuellen kritischen Pfad bestimmten Lösungsraum zu verlassen und durch Zuordnung von nicht direkt im kritischen Pfad liegenden Knoten Verbesserungen zu erreichen.
Anmerkung: Die Nachbarschaft bei One bzw. im Umfeld des kritischen Pfades unterscheiden sich darin, daß bei One nur Verarbeitungsknoten alternativ abgebildet werden, während bei den auf den kritischen Pfad bezogenen Varianten zur Vermeidung von Kommunikationsknoten auch die beiden kommunizierenden Verarbeitungsknoten gleichzeitig anderweitig zugeordnet werden können. Für One besteht deshalb ein direkter Zusammenhang zwischen Größe der Nachbarschaft und Größe des Graphen, während dies für (E)CP nicht gilt. Hier hängt die Nachbarschaftsgröße von der Länge des kritischen Pfads des Graphen bzw. bei ECP zusätzlich vom gewählten Schedule und den dabei auftretenden Zugriffskonflikten ab.

Der Rückstand des konstruktiven Verfahrens von Xie/Wolf zu den beiden Varianten mit Tabu Suche liegt in der Tatsache begründet, daß zur gleichzeitigen Entscheidung über Abbildung und Scheduling der Knoten nur eine Schätzung über die nachfolgenden Abläufe vorgenommen wird, die im folgenden nicht mehr revidiert werden können. Demgegenüber können rekursive Verfahren den gesamten Schedule bei den Iterationen mit berücksichtigen und die Entscheidung über die Zuordnung schrittweise verbessern.
Die relative Performance der Verfahren gegenüber FAST korrespondiert auch mit der von den Verfahren generierten Auslastung des Busses. D.h. die beiden performanteren TS-Varianten lasten den Bus prozentual höher aus als FAST. Mit zunehmender Graphgröße wächst der Unterschied von 0,5% auf ca. 3% an.

Die oben dargestellten Zusammenhänge in Bezug auf die Sucherverfahren führen zu entsprechenden Konsequenzen bei der Laufzeit der einzelnen Algorithmen.

Anzahl Knoten	Laufzeit [s]				
	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
10	0,02	0,01	0,01	0,18	0,00
20	0,11	0,05	0,04	0,18	0,00
50	1,40	0,36	0,58	1,71	0,01
100	13,43	2,17	5,84	6,66	0,04
150	58,65	6,16	25,63	17,17	0,13
200	187,71	14,37	81,87	37,26	0,31
300	1198,45	55,44	810,97	128,80	1,20

Tabelle 5-1: Laufzeit der Verfahren abhängig von der Graphgröße

Die Laufzeiten hängen neben der Graphgröße für die rekursiven Verfahren insbesondere

auch von der Anzahl der untersuchten alternativen Abbildungen ab. Die oben dargestellte Entwicklung der Nachbarschaftsgrößen korrespondiert deshalb mit den in Tabelle 5-1 aufgelisteten Werten. Der Vorteil von ECP gegenüber One in Bezug auf die geringere Laufzeit bei ähnlichen Performance-Werten nimmt jedoch ab, da die Zahl der untersuchten Abbildungen bei ECP infolge der stärker steigenden Nachbarschaftsgröße wesentlich mehr zunimmt. Die kürzeren Laufzeiten von CP im Vergleich zu FAST bei der Laufzeit ergeben sich aus der geringen Anzahl von untersuchten Alternativen.
Die Einsetzbarkeit der rekursiven, auf lokaler Suche basierten Methoden ist aufgrund der sich ergebenden Laufzeiten stark durch die Größe des Graphen beschränkt. Die Ergebnisse aus Tabelle 5-1 bestätigen die bei der Auswahl des Optimierungsverfahrens gemessenen Laufzeiten. Die Vorteile bei der Performance im Vergleich zum eingeschränkten Suchverfahren FAST und insbesondere zum konstruktiven Verfahren Wolf, das aufgrund der wesentlich geringeren Laufzeit auch große Graphen bearbeiten kann, werden hier stark relativiert.

Anteil bedingter Knoten

Da die Zielsetzung des Verfahrens in der Unterstützung von Anwendungen mit Kontrollabhängigkeiten liegt, soll den bedingten Übergängen im Graphen besondere Aufmerksamkeit gewidmet werden. Deshalb wird nachfolgend der Anteil der in einem Graphen beinhalteten bedingten Kanten näher untersucht. Da die Besonderheit im Zusammenhang mit dieser Größe in der gegenseitigen Exklusivität von Übergängen liegt, wird deshalb statt dessen der Anteil von Knoten mit Bedingungen an der Gesamtzahl von Knoten als Parameter betrachtet. Dies hängt mit der Erzeugbarkeit entsprechender Graphen zusammen, da pro Knoten zumindest zwei derartige Kanten abgehen müssen. Durch eine zufällige Auswahl von beliebigen Kanten des Graphen, die dann mit Bedingungen versehen werden, könnte dies nicht garantiert werden. Deshalb werden zur Erzeugung von Graphen mit bedingten Übergängen Knoten zufällig selektiert, bei denen dann mindestens zwei abgehende Kanten mit Bedingungen versehen werden.

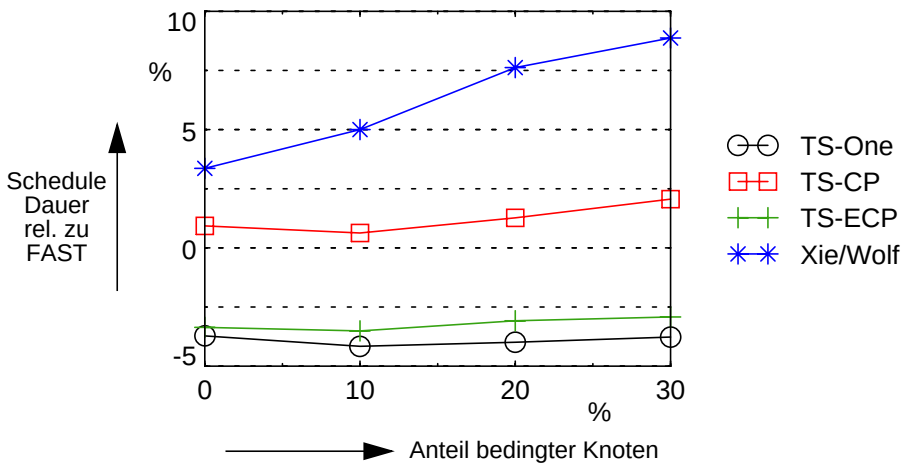


Bild 5-6: Schedule Dauer in Abhängigkeit vom Anteil bedingter Knoten

Die Performance der auf Tabu Suche basierenden Verfahren ist im Vergleich zu FAST relativ unabhängig vom Anteil der Knoten mit Bedingungen. Da die Berücksichtigung der Bedingungen über das Scheduling und nicht durch das Suchverfahren selbst erfolgt, bestimmt hier in erster Linie die gemeinsame Scheduling Methode über die Güte des Ergebnisses. Für das konstruktive Verfahren nach Xie/Wolf, bei dem die Abbildung und das Scheduling gleichzeitig festgelegt werden, und bei dem die Priorität zur Auswahl der besten Knoten-Ressourcen-Kombination über eine dynamisch modifizierte Abschätzung der zum Endknoten erforderlichen Verarbeitungszeit bestimmt wird, ergeben sich bei höherem Anteil bedingter Knoten zunehmend schlechtere Ergebnisse im Vergleich zu FAST. Der Grund hierfür liegt darin, daß bei größer werdenden Anteil von gegenseitig exklusiven Prozessen Xie/Wolf die Möglichkeiten der gemeinsamen Ressourcennutzung wegen der frühzeitigen Festlegungen nicht mehr so gut ausnützen kann wie die rekursiven Verfahren.

Anteil bed. Knoten	Laufzeit [s]				
	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
0%	14,59	2,14	6,95	6,79	0,04
10%	13,43	2,17	5,84	6,66	0,04
20%	17,02	2,77	6,06	8,89	0,06
30%	77,04	11,95	21,21	42,78	0,80

Tabelle 5-2: Laufzeit in Abhängigkeit vom Anteil bedingter Knoten

Die Laufzeiten für die verschiedenen Verfahren bis 20% sind nahezu unabhängig vom Anteil der bedingten Knoten. Die bei allen Verfahren eintretende starke Zunahme bei 30% hängt von einzelnen Graphen ab, die zu erheblich über dem übrigen Durchschnitt liegenden Laufzeiten führen. Die Ursache hierfür liegt in der Struktur der jeweiligen Graphen begründet, die bei der synthetischen Generierung entsteht. Aufgrund der sich ergebenden Übergänge innerhalb des Graphen steigt die Anzahl der bei den Knoten zu speichernden Pfadinformationen derart an, daß bei der paarweisen Überprüfung aller möglichen Pfadkombinationen erheblicher Zeitaufwand notwendig ist. Die zur Speicherung des annotierten Graphen erforderliche Datenmenge ist für diese Graphen ebenfalls wesentlich größer als die der übrigen Graphen.
Neben dem Anteil von Knoten mit Bedingungen ist der Anteil der von einem bedingten Knoten abgehenden Kanten, die tatsächlich mit der Bedingung des Knoten verknüpft sind, ein zusätzlicher Parameter. In weiteren, hier nicht explizit angegebenen Untersuchungen hat sich gezeigt, daß die Performance der Verfahren und deren Laufzeit unabhängig von diesem Parameter ist. Deshalb wird hier darauf nicht näher eingegangen.

Transferierte Datenmenge pro Kante

Als weitere den Graphen beschreibende Größe wird die mittlere zwischen den Knoten zu transferierende Datenmenge betrachtet. Dieser Parameter hat insofern besondere Bedeutung als damit der Aspekt der Kommunikation über das gemeinsam genutzte Kommunikationsmedium, d.h. den Bus, erfaßt werden kann.

Dieser Parameter hängt auch eng mit den Eigenschaften des Busses zusammen. Da in dem Verfahren bisher lediglich die Breite und die Geschwindigkeit des Busses berücksichtigt werden, kann die Veränderung der Datenmenge ebenfalls äquivalent zu Änderungen der Buseigenschaften betrachtet werden. Eine Erhöhung der Datenmenge korrespondiert deshalb mit einer Reduktion der Busgeschwindigkeit oder einer Reduktion der Breite des Busses. Deshalb sollen im weiteren auf nähere Untersuchungen zu den Bus-Parametern verzichtet werden, da deren Einfluß auf die Leistungsfähigkeit bereits mit diesen Simulationen in der genannten Weise abgedeckt ist.

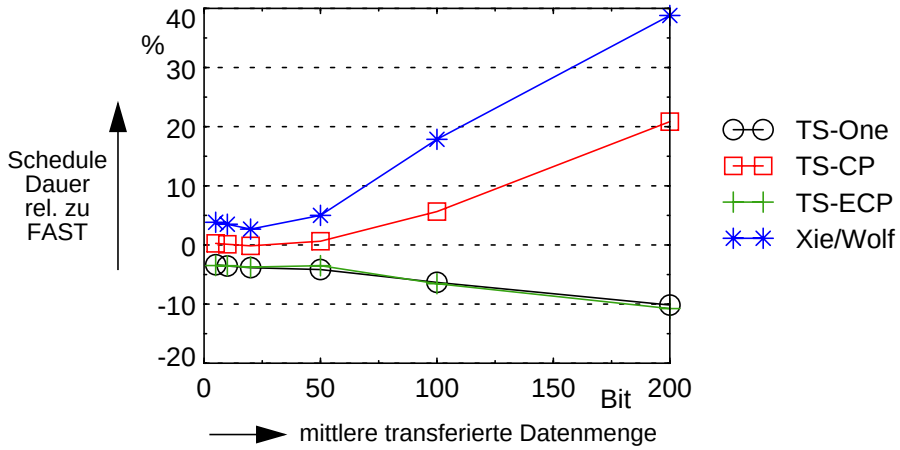


Bild 5-7: Schedule Dauer in Abhängigkeit der transferierten Datenmenge

Mit zunehmender transferierter Datenmenge nehmen die Vorteile von TS-One und TS-ECP kontinuierlich zu, während TS-CP und insbesondere Xie/Wolf teilweise erheblich schlechtere Lösungen generieren als FAST. Bei TS-ECP steigt aufgrund längerer Transfers und damit längerer Busbelegung die Menge der verzögerten Prozesse, was zu einer zunehmenden Nachbarschaftsgröße führt, die jedoch die konstant bleibende Größe der Nachbarschaft von One nur bis zu ca. 75% erreicht. In der bei TS-ECP sich vergrößernden Nachbarschaft werden bei dem gewählten Abbruchkriterium häufig Verbesserungen gefunden, so daß bei TS eine höhere Anzahl von Iterationen durchgeführt wird. Obwohl die Größe der Nachbarschaft von TS-One konstant bleibt, steigt dabei die Zahl der Durchläufe sogar stärker als bei TS-ECP. Dies läßt darauf schließen, daß bei zunehmender Transferdauer der Lösungsraum zunehmend "weniger flach" ist, d.h. daß sich durch entsprechende Überlagerungseffekte auf dem Bus durch Alternativabbildungen Verbesserungen erreichen lassen, die mit FAST bzw. TS-CP nicht oder nicht mit ausreichender Dichte abgesucht werden. Dies führt zu einer zunehmend lückenhaften Abdeckung des Lösungsraums und im Vergleich zu TS-One und TS-ECP zu schlechteren Lösungen.
Bei CP und noch verstärkt bei Xie/Wolf ist festzustellen, daß die Prozessoren mit zunehmender Transfergröße immer geringer ausgelastet werden und deshalb infolge der Transfers insgesamt eine zunehmend geringere Performance erreicht wird. Im Falle von Xie/

Wolf kann dieser Trend durch die Strategie der Auswahl der Prozeß-Ressourcen-Paars mit frühester Endzeit erklärt werden, wodurch Beschleuniger zu Lasten der Prozessoren bevorzugt und Engpässe auf dem Bus verursacht werden.
Bei FAST und den anderen auf Tabu Suche basierenden Verfahren sind die Zahlen der Auslastung der Prozessoren unabhängig von der transferierten Datenmenge. Hier wird also eine gute Abwägung zwischen Implementierung auf dem Prozessor und der mit einem Transfer verbundene Auslagerung in einem Beschleuniger getroffen.
Diese Interpretation wird auch dadurch gestützt, daß die Anzahl der Iterationen bei One am meisten zunimmt, obwohl die Nachbarschaftsgröße konstant bleibt.
Aufgrund der bei TS-One und TS-ECP insgesamt wesentlich höheren Anzahl von überprüften Abbildungsalternativen ergibt sich deshalb jeweils eine wesentlich bessere Performance als für FAST.

mittlere Transfermenge [Bit]	Laufzeit [s]				
	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
5	7,56	1,47	1,63	6,03	0,20
10	7,90	1,47	1,81	6,06	0,19
20	8,82	1,61	2,01	6,05	0,20
50	13,46	2,21	5,87	7,15	0,21
100	25,47	3,28	19,20	8,96	0,26
200	42,43	4,32	27,95	10,28	0,23

Tabelle 5-3: Laufzeit in Abhängigkeit der transferierten Datenmenge

Der Anstieg der Laufzeit bei den rekursiven Verfahren geht mit der Erhöhung der Transfermenge einher, die mit einer Zunahme der betrachteten Abbildungsvarianten verbunden ist. Die Laufzeit bei Xie/Wolf ist nahezu unabhängig von der transferierten Datenmenge, da sie keinen direkten Einfluß auf das Auswahlverfahren der einzelnen Prozeß-Ressourcen-Kombinationen besitzt.

5.2.4.2 Eigenschaften der Architektur

Nachfolgend sollen die Eigenschaften der zugrundeliegenden Architektur und der darin vorhandenen Ressourcen betrachtet werden. Da die gesamte Struktur der Architektur und insbesondere die Kommunikationsarchitektur mit dem gemeinsamen Bus als Randbedingung vorgegeben ist, sind hier insbesondere die Anzahl und die Eigenschaften der Verarbeitungsressourcen von Interesse. Die Frage, wie die betrachteten Algorithmen eine vorgegebene Architektur mit einer unterschiedlichen Anzahl, Art und Performance von Verarbeitungsressourcen ausnutzen können, soll deshalb im folgenden betrachtet werden. Als Randbedingungen werden hierzu jeweils zwei Klassen von Ressourcen angenommen, die in unterschiedlicher Anzahl in der Architektur eingesetzt werden können. Zum einen wird ein generell einsetzbarer eingebetteter Prozessor angenommen, zum anderen spezifische HW-Beschleuniger, die z.B. als ASIC-Module implementiert sein können. Die Unterscheidung dieser beiden Ressourcentypen erfolgt anhand der mittleren Verarbeitungsdauer für die Prozesse und des Anteils der auf der jeweiligen Klasse implementierbaren Prozesse.

Für die Untersuchungen können beide Typen von Ressourcen unterschiedlich konfiguriert und in ihren Eigenschaften relativ zueinander verändert werden. Dies betrifft zum einen die mittlere Verarbeitungsdauer der auf den Ressourcen lauffähigen Prozesse. Als weiteres Charakteristikum für diese beiden Ressourcen wird berücksichtigt, welcher Anteil der Prozesse des Graphen auf den Beschleunigern implementiert werden kann. Für die in der Architektur vorhandenen Prozessoren wird jeweils angenommen, daß alle Prozesse darauf ausführbar sind, jedoch mit geringerer Performance als auf den Beschleunigern.

Anzahl der Ressourcen in der Architektur

Zunächst wird die in der Architektur vorhandene Anzahl von Prozessoren und Beschleunigern betrachtet. Die nachfolgenden Ergebnisse wurden bei der Untersuchung von Graphen mit 100 Prozessen bei einem Anteil von 10% bedingter Knoten gewonnen. Dabei wurde von einer mittleren Beschleunigung der auf den ASIC Modulen implementierbaren Prozesse um den Faktor 5 im Vergleich zu den Prozessoren ausgegangen. Der Anteil der beschleunigbaren Prozesse wurde mit 25% angenommen.

Im nachfolgenden Bild 5-8 sind jeweils die Simulationsergebnisse für die verschiedenen Verfahren im Vergleich zu FAST als Funktion der Anzahl von Prozessoren und Beschleunigermodulen angegeben.

Auch hier zeigen TS-One und TS-ECP ähnliches Verhalten. Mit zunehmender Anzahl von Beschleunigermodulen und insbesondere von Prozessoren nimmt der Vorteil gegenüber FAST zu. Die von beiden Verfahren erzielte Verbesserung liegt zum einen an der mit steigender Ressourcenzahl zunehmenden Anzahl von untersuchten Alternativabbildungen, die sich jeweils aus der Zunahme der Nachbarschaftsgröße ergeben. Zum anderen ist festzustellen, daß bei FAST die Menge der untersuchten Alternativen insbesondere mit steigender Anzahl von Prozessoren abnimmt und damit der größer werdende Lösungsraum zunehmend lückenhafter abgesucht wird. Dieser Rückgang wird durch das Abbruchkriterium des FAST Algorithmus verursacht, der zum einen eine Begrenzung der Anzahl der maximal betrachteten Lösungen in der inneren Suchschleife beinhaltet, gleichzeitig aber auch einen vorzeitigen Abbruch vorsieht, wenn in einer sehr geringen Anzahl von aufeinanderfolgenden Versuchen keine Verbesserung erreicht wird. Mit steigender Anzahl von Ressourcen wird dieser vorzeitige Ausstieg vermehrt aktiviert. Bei Architekturen mit einem Prozessor wird für One bzw. bei Architekturen mit einem oder zwei Prozessoren wird bei ECP jeweils eine geringere Anzahl von Alternativen untersucht als bei FAST. Trotzdem erreichen in diesen Fällen beide Verfahren gleiche oder bessere Performance.
Bei TS-CP sind keine einheitlichen Trends festzustellen. Bessere Performance als FAST erhält man lediglich bei hoher Anzahl von Prozessoren und gleichzeitig wenig Beschleunigern. Wird bei TS-CP bei einer großen Anzahl von Prozessoren die Beschleunigerzahl erhöht, so nimmt auch hier die Menge der untersuchten Lösungsvarianten ab.
Da FAST und TS-CP ausschließlich im engeren Umfeld des kritischen Pfades suchen, wobei TS-CP die zugehörige Nachbarschaft vollständig absucht, können von diesem Teil des Lösungsraums offenbar keine Bereiche mit potentiellen Verbesserungen erreicht werden.

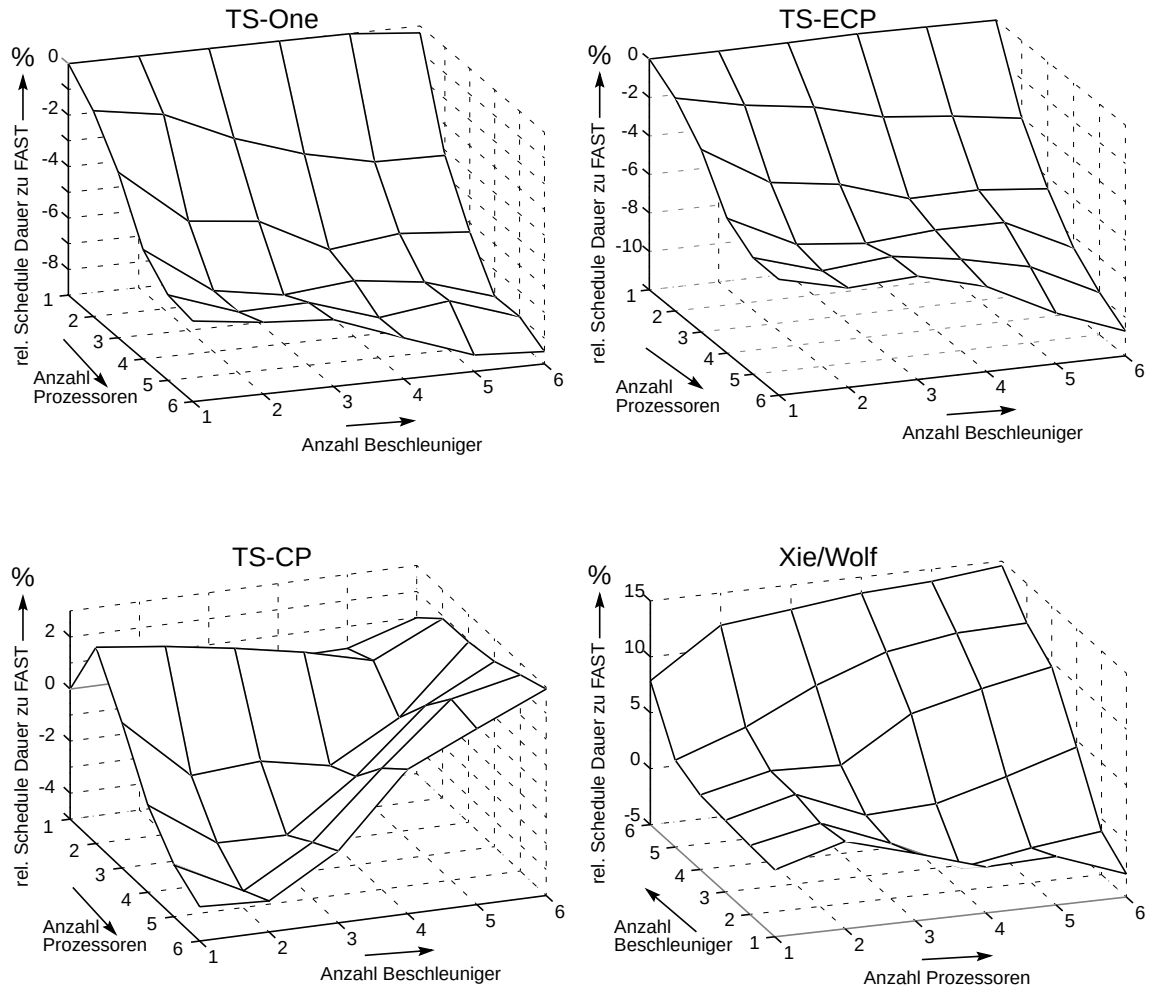


Bild 5-8: Schedule Dauer in Abhängigkeit der Ressourcenanzahl

Der Grund hierfür liegt darin, daß durch Auslagerung der beschleunigbaren Prozesse auf entsprechende Beschleuniger sich der kritische Pfad nicht in der Art ändert, daß andere, potentiell bessere Lösungen gefunden werden können, mit der Folge daß keine weiteren Verbesserungen der Schedule Dauer erreichbar sind. Eine größere Anzahl von Beschleunigern wird bei diesen Verfahren auch stärker genutzt, was jedoch durch eine im Vergleich zu TS-One und TS-ECP etwas erhöhte Buslast erkauft wird, die durch größere Verzögerungen auf dem Bus den Gewinn durch die Beschleunigung wieder zunichte macht.
Diese enge Beschränkung der Suche ist bei TS-ECP und TS-One nicht gegeben. Durch die jeweils breiter gefaßte Nachbarschaft und insbesondere bei ECP durch die Berücksichtigung von speziellen, die Performance beschränkenden Prozessen sind weitere Verbesserungen möglich. Durch die größere Anzahl der betrachteten Lösungsalternativen ergibt sich für die TS-Varianten auch eine im Vergleich zu FAST bessere Abdeckung des Lösungsraums.
Das konstruktive Verfahren von Xie/Wolf liefert mit steigender Anzahl von Beschleunigern zunehmend schlechtere Performance als FAST. Dieses Verhalten tritt mit mehr Pro-

zessoren zunehmend schneller auf, während bei geringer Anzahl von Beschleunigern mit steigender Prozessorzahl noch Verbesserungen erreicht werden können. Die genannten Effekte werden durch die Auswahlstrategie der am frühesten endenden Prozeß-Ressourcen-Zuordnung verursacht, die die notwendigen Transfers, insbesondere nach Beendigung von Prozessen auf Beschleunigern nicht ausreichend berücksichtigt. Insgesamt ist festzustellen, daß für dieses Verfahren ein wesentlich schlechteres Leistungsniveau erreicht wird als bei den rekursiven Verfahren.

Anzahl		Laufzeit [s]				
Prozessoren	Beschleuniger	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
1	1	0,47	0,03	0,03	5,05	0,01
	2	1,00	0,09	0,11	6,23	0,01
	3	1,88	0,21	0,29	6,94	0,01
	4	3,05	0,37	0,61	7,44	0,02
	5	4,58	0,63	1,14	8,32	0,02
	6	8,20	0,98	2,97	9,53	0,03
2	1	4,62	0,69	2,27	4,88	0,02
	2	5,32	0,84	2,23	5,16	0,02
	3	7,02	1,12	2,62	5,46	0,02
	4	9,77	1,59	3,64	6,05	0,03
	5	13,43	2,17	5,84	6,67	0,04
	6	21,44	2,92	13,10	7,52	0,05
3	1	11,86	2,15	5,35	4,59	0,02
	2	13,23	2,51	5,84	4,76	0,03
	3	15,93	3,03	6,67	5,11	0,03
	4	21,53	3,72	10,68	5,68	0,05
	5	27,42	4,33	17,30	6,19	0,06
	6	37,15	5,09	24,24	6,87	0,08
4	1	23,47	4,35	9,97	4,63	0,03
	2	24,17	5,43	10,19	4,79	0,04
	3	28,08	5,59	12,56	5,13	0,05
	4	35,31	6,15	18,80	5,62	0,07
	5	44,35	6,64	24,77	6,03	0,08
	6	57,66	7,69	34,12	6,73	0,09
5	1	35,07	7,34	14,75	4,71	0,04
	2	35,37	7,68	13,81	4,86	0,05
	3	39,49	7,99	21,32	5,22	0,07
	4	52,64	8,24	24,40	5,65	0,08
	5	54,44	7,75	33,18	6,12	0,09
	6	71,27	9,07	44,74	6,74	0,11
6	1	45,58	10,72	17,32	4,79	0,05
	2	48,01	10,13	19,75	4,99	0,07
	3	51,78	10,13	19,75	4,99	0,07
	4	64,31	9,76	30,56	5,78	0,10
	5	71,49	9,75	38,92	6,20	0,11
	6	84,35	9,87	53,33	6,86	0,13

Tabelle 5-4: Laufzeit in Abhängigkeit der Ressourcenanzahl

Die Laufzeiten steigen für alle Verfahren mit der Anzahl der Prozessoren bzw. der Beschleuniger an. Eine allen betrachteten Verfahren gemeinsame Ursache hierfür liegt in der Zunahme des Aufwandes beim Scheduling, die sich aus der erhöhten Anzahl von möglichen Implementierung der einzelnen Prozesse ergibt. Dieser generelle Zuwachs ist mit zunehmender Prozessorzahl größer als bei der Erhöhung der Beschleunigeranzahl, die nur einen begrenzten Teil der Prozesse implementieren können.
Bei den auf Tabu Suche basierenden Ansätzen verursacht in erster Linie die Vergrößerung der jeweiligen Nachbarschaften und der damit verbundenen höheren Anzahl von untersuchten Abbildungsalternativen diese Zunahme. Bei CP ist der leichte Rückgang der Laufzeit bei 6 Prozessoren durch die sehr stark fallende Anzahl von untersuchten Alternativen verursacht, die den oben genannten Scheduling-Aufwand überkompensiert.
Bei FAST steigt die benötigte Laufzeit mit zunehmender Beschleunigerzahl mäßig an, während sie bei Erhöhung der Prozessorzahl nahezu gleichbleibt. Die oben erwähnte Abnahme der untersuchten Alternativen wird durch den steigenden grundsätzlichen Berechnungsaufwand jedoch überkompensiert.

Mittlere Performance der Ressourcen

Als weitere Größe zur Beschreibung der Architektur wird die mittlere Bearbeitungszeit der Blöcke für die einzelnen Prozesse untersucht. Dabei wird angenommen, daß in der Architektur 2 Prozessoren und 5 ASIC Blöcke vorhanden sind, und alle Prozesse auf den Prozessoren, jedoch nur jeweils 25% auf einem Beschleuniger lauffähig sind. Um die Anzahl der Simulationen zu begrenzen, wird lediglich die Beschleunigung B der ASIC Blöcke untersucht, die durch das Verhältnis der mittleren Verarbeitungszeit von Prozessoren zu ASICS über $B = D_{proz}/D_{ASIC}$ gegeben ist. Dabei wird die mittlere Verarbeitungsdauer der Prozessoren konstant gehalten und lediglich die der ASIC Blöcke entsprechend reduziert.

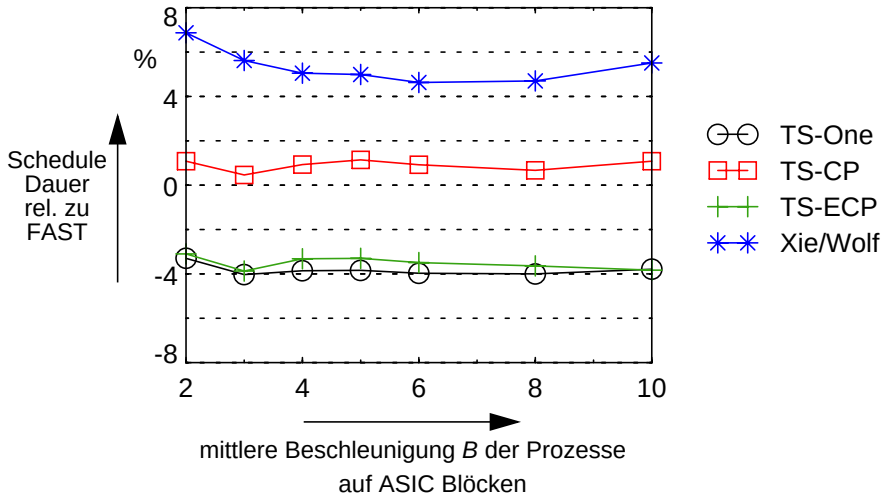


Bild 5-9: Schedule Dauer in Abhängigkeit der Beschleunigung B der ASIC Blöcke

Man erkennt in Bild 5-9, daß alle rekursiven, auf Tabu Suche basierenden Verfahren unabhängig von der Definition der Nachbarschaft nahezu gleiches Verhalten mit zunehmender mittleren Beschleunigung durch die ASIC Blöcke zeigen. Auch in diesem Fall resultiert eine wesentlich bessere Performance der ECP- und One-Nachbarschaft.
Die Nachbarschaftsgrößen bleiben mit Ausnahme von TS-ECP im wesentlichen konstant, auch die untersuchten Abbildungsalternativen sind nahezu unabhängig von der Beschleunigung *B* der ASIC Blöcke. Mit zunehmender Performance der Beschleuniger ergibt sich bei den ansonsten gleichbleibenden Transport- und Verarbeitungszeiten auf den Prozessoren eine größere Anzahl von Situationen, in denen Wartezeiten entstehen. Dies hat bei TS-ECP zur Folge, daß die verursachenden Prozesse in die untersuchte Nachbarschaft aufgenommen werden. Die damit verbundene Zunahme der mittleren Nachbarschaftsgröße führt auch zu dem in der Tabelle 5-5 ersichtlichen Anstieg der Laufzeit.

mittlere Beschleunigung	Laufzeit [s]				
	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
2	12,05	2,33	4,39	6,20	0,28
3	12,32	2,13	5,28	6,55	0,27
4	12,77	2,06	5,68	6,76	0,20
5	13,47	2,23	6,00	6,87	0,20
6	13,05	2,14	6,49	7,11	0,20
8	13,36	2,21	6,54	7,21	0,20
10	13,99	2,17	6,94	7,35	0,20

Tabelle 5-5: Laufzeit in Abhängigkeit von der mittleren Beschleunigung

Anteil von beschleunigbaren Prozessen

Nachfolgend wird betrachtet, wie sich der Anteil der auf den jeweiligen Ressourcen implementierbaren Prozesse auf die Leistungsfähigkeit der Algorithmen auswirkt. Zu diesem Zweck wurden die Architekturinformationen variiert, die vorgeben, welcher Anteil der im Graphen enthaltenen Prozesse auf den Beschleunigern implementiert werden kann. Die Beschleunigung wurde dabei auf den Faktor 5 eingestellt. Die Graphgröße beträgt 100 Knoten mit 10% bedingten Knoten. Die Ergebnisse sind im Bild 5-10 bzw. in der Tabelle 5-6 angegeben.

Die Abhängigkeit vom Anteil der beschleunigbaren Prozesse wirkt sich ähnlich wie die Zunahme der Beschleunigeranzahl in der Architektur aus. Hier können ähnliche Trends in der relativen Performance bzw. der benötigten Laufzeit beobachtet werden. Bei den rekursiven auf Tabu Suche basierenden Verfahren ist ein starkes Anwachsen der Nachbarschaftsgröße zu beobachten was zu einer erheblichen Zunahme der Anzahl untersuchter Abbildungen führt

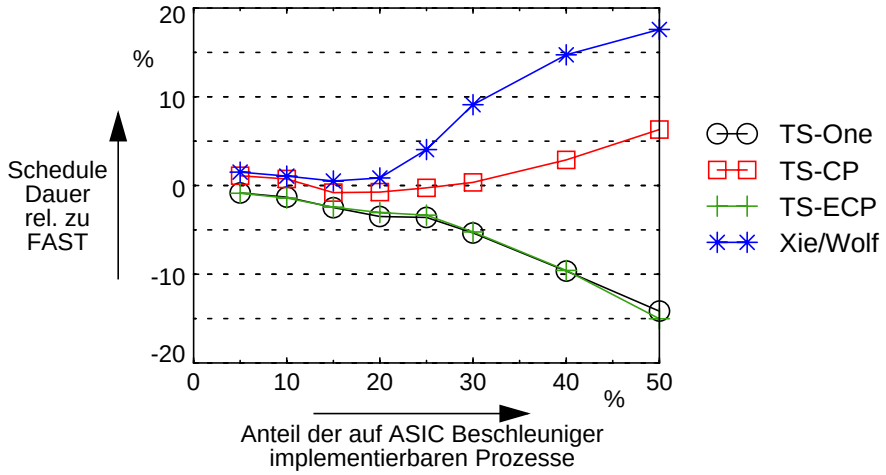


Bild 5-10: Schedule Dauer in Abhängigkeit vom Anteil der auf einem Beschleuniger implementierbaren Prozesse

.

Anteil beschleunigbarer Prozesse	Laufzeit [s]				
	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
5%	5,66	1,07	3,43	4,72	0,19
10%	6,95	1,39	3,55	5,14	0,17
15%	8,70	1,82	3,73	5,48	0,19
20%	11,02	2,38	4,35	5,80	0,19
25%	14,92	2,77	8,54	6,53	0,27
30%	22,17	3,78	13,83	6,87	0,23
40%	41,75	5,49	29,44	7,42	0,30
50%	59,77	7,51	44,35	7,42	0,25

Tabelle 5-6: Laufzeit in Abhängigkeit der beschleunigbaren Prozesse

Einfluß der initialen Abbildung der Prozesse

In den voranstehenden Untersuchungen wurde bei den rekursiven Verfahren jeweils von einer initialen Zuordnung ausgegangen, bei der alle Prozesse auf die für sie jeweils schnellste Ressource abgebildet wurden (schnellstes Initialmapping). Diese Wahl wurde zum einen unter dem Hintergrund der in Kapitel 4.5.4 gezeigten Untersuchungen, insbesondere aber auch im Hinblick auf die Zielsetzung der Optimierung ausschließlich der Performance getroffen.

Als denkbare Alternative kommt jedoch auch eine zufällige Wahl aus den erlaubten Res-

sourcen in Betracht. Voruntersuchungen haben gezeigt, daß die zufällige Auswahl eines Mappings als Startwert zu sehr unterschiedlichen Lösungen führt. Insbesondere wird nur in wenigen Fällen die Güte der Ergebnisse erreicht, die sich bei der Verwendung der minmalen Variante ergibt. Um den Einfluß der zufällig gewählten Anfangslösung zu reduzieren, müßten mehrere Läufe mit unterschiedlichen Startwerten durchgeführt werden, was den Berechnungsaufwand zusätzlich erhöhen würde. Deshalb wurde diese Variante nicht weiter betrachtet.
Um jedoch grundsätzlich die Eignung der Verfahren bewerten zu können, sich der optimalen Lösung anzunähern, wurde als sehr weit von der optimalen Lösung entfernt liegendes Initial-Mapping die Auswahl der jeweils langsamsten Ressource (langsamstes Initialmapping) untersucht. In Bild 5-11 sind für diesen Fall links die Performancewerte relativ zu FAST angegeben, während im rechten Diagramm das Verhältnis der bei den Simulationen mit diesen beiden Anfangslösungen resultierenden mittleren Schedule Dauern angegeben ist.

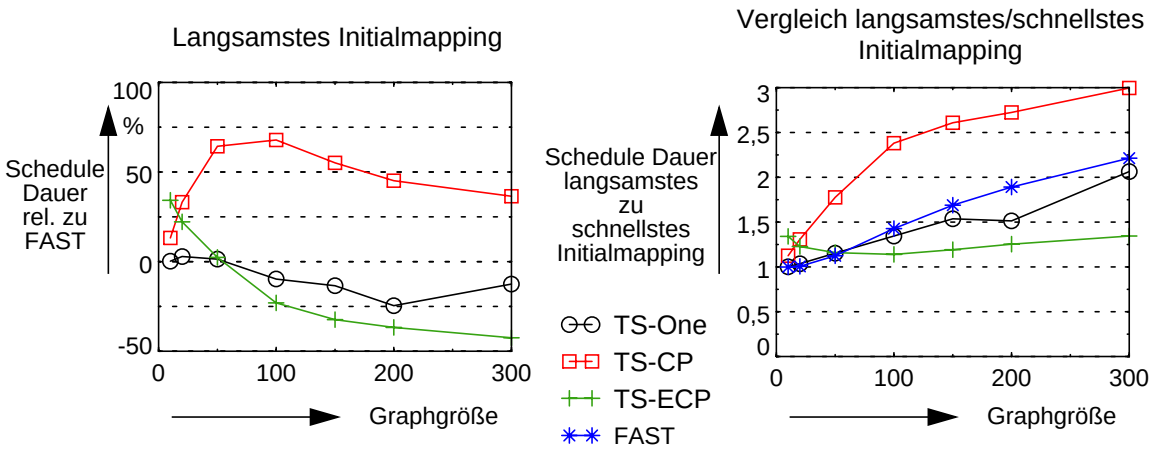


Bild 5-11: Verhalten der Verfahren bei ungünstiger initialer Abbildung

Die Ergebnisse zeigen, daß die Wahl der Anfangsabbildung erheblichen Einfluß auf die letztlich resultierenden Lösungen besitzt. Bei kleinen Graphgrößen liefert FAST die besten Lösungen, jedoch mit zunehmender Graphgröße gewinnen TS-ECP und TS-One erhebliche Vorteile.
Das schlechte Verhalten von TS-CP stützt auch die bereits in den voranstehenden Untersuchungen getroffene Feststellung, daß das alleinige Absuchen des kritischen Pfades nicht ausreicht, um den Lösungsraum effektiv zu untersuchen.
Im rechten Diagramm erkennt man, daß alle Verfahren mit steigender Graphgröße zunehmend schlechtere Ergebnisse liefern, wenn die Prozesse anfänglich auf die jeweils langsamste Ressource zugeordnet werden. Die Kurve für TS-ECP besitzt hierbei jedoch die geringste Steigung, was darauf hinweist, daß dieses Verfahren noch die beste Fähigkeit besitzt, sich an die optimalen Lösung anzupassen.

Im Bild 5-12 ist für die rekursiven Verfahren jeweils das Verhältnis der Laufzeit bei langsamster zu schnellster initialer Zuordnung angegeben. Bei TS-One und TS-ECP erkennt man ausgeprägte Maxima bei Graphen mit ca. 150-200 Knoten, während FAST und TS-CP nach anfänglich leichtem Anstieg schon frühzeitig einen stetig fallenden Verlauf zeigen.
Die zu größeren Graphen hin auftretende Abnahme aller Kurven wird durch den zunehmend früheren Abbruch der Suche verursacht, da weitere Verbesserungen nicht zu erreichen sind. Infolge der bei TS-ECP und TS-One insgesamt größeren Nachbarschaften, führt dies zu ausgeprägteren Maxima bei größeren Graphgrößen. Allen hier betrachteten Suchverfahren ist jedoch gemein, daß bei ungünstiger Anfangsabbildung mit zunehmender Graphgröße die Situation eintritt, daß die Suche nicht mehr in den Bereich mit aussichtsreichen Verbesserungen gelangt, so daß die Suche im Vergleich zu kleineren Graphen zunehmend früher abgebrochen wird. Im Fall von FAST und TS-CP sind damit sogar für den ungünstigen Startpunkt kürzere Laufzeiten als für den günstigen zu beobachten.
Der bei TS-ECP und geringer bei TS-One auftretende Vorteil hinsichtlich der erreichbaren Performance schlägt sich hier in erheblich höheren Laufzeiten nieder, da jeweils eine wesentlich größere Anzahl von Iterationen durchlaufen werden muß, um sich von der ungünstigen Anfangslösung in Richtung Optimum zu bewegen.

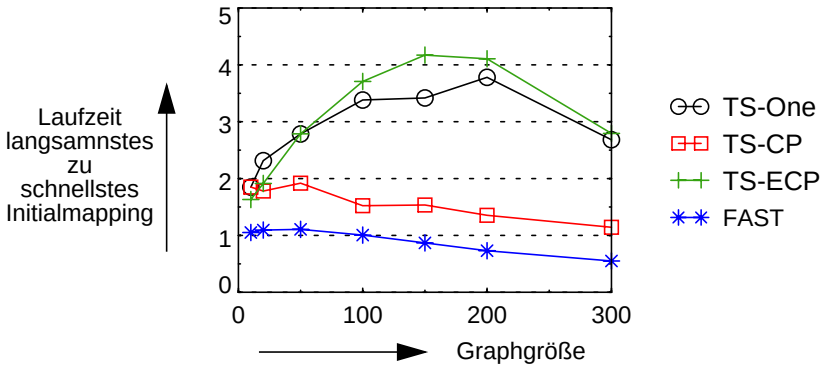


Bild 5-12: Einfluß der Initialen Abbildung auf die Laufzeit

Die weitaus besseren Ergebnisse bei einer anfänglichen Wahl der Ressourcen mit minimaler Prozeßausführungszeit sowohl im Hinblick auf die Performance der jeweils resultierenden Lösung als auch auf die benötigte Laufzeit bestätigen damit die oben getroffene Wahl.

5.2.4.3 Zusammenfassung der Ergebnisse der Experimente mit synthetischen Graphen

Aus den Untersuchungen anhand der Experimente mit synthetischen Graphen können folgende Feststellungen getroffen werden:

- Das rekursive Verfahren erzielt unabhängig von der verwendeten Nachbarschafts-

variante in allen untersuchten Fällen bessere Ergebnisse als der konstruktive Algorithmus von Xie/Wolf, bei dem einmal getroffene Festlegungen nicht revidiert werden können.

- Die benötigte Laufzeiten für die rekursiven Verfahren liegen jedoch um durchschnittlich 2 Größenordnungen höher als bei Xie/Wolf, so daß ihre Einsetzbarkeit auf kleine bis mittelgroße Graphen beschränkt ist.
- Die rekursiven Verfahren können jedoch anhand von Parametern in Bezug auf ihre Suchdauer angepaßt werden, bei dem konstruktiven Algorithmus besteht diese Möglichkeit nicht.
- Bei den rekursiven Verfahren erzielen TS-One und TS-ECP die besten Ergebnisse bei der Performance der resultierenden Lösung. Dabei benötigt TS-ECP infolge der zielgerichteteren Suche wesentlich kürzere Laufzeiten als TS-One.
- Die Verwendung von FAST als Suchmethode führt infolge der eingeschränkten Suche zu wesentlich schlechteren Ergebnissen in Bezug auf die damit erreichbare Performance als die Varianten der Tabu Suche, die nicht auf den kritischen Pfad im engeren Sinn beschränkt sind.
- Die Suche in einer ausschließlich auf den eigentlichen kritischen Pfad begrenzten Nachbarschaft, wie sie von FAST und TS-CP durchgeführt wird, beschränkt den möglichen Lösungsbereich und führt zu signifikant schlechteren Lösungen als die erweiterten Nachbarschaftsvarianten. Obwohl TS-CP in vielen Fällen ähnlich gute Ergebnisse wie FAST liefert, ist die entsprechende Laufzeit geringer.
- Die Laufzeiten von FAST liegen meist in der Größenordnung von TS-ECP mit Ausnahme der Fälle, in denen sich aufgrund der Eigenschaften des Graphen bzw. der Architektur eine starke Zunahme der Größe der ECP-Nachbarschaft ergibt.
- Die Laufzeit für FAST ist insbesondere von der Anzahl der aufgrund der statistischen Suchmethode mehrfach durchgeführten Läufe abhängig, über die die Güte des Ergebnisses direkt beeinflußt werden kann. Demgegenüber ergibt sich die Laufzeit der auf dem Prinzip der Tabu Suche basierenden, die Nachbarschaft vollständig absuchenden Verfahren neben dem Abbruchkriterium v.a. durch die Größe der Nachbarschaft. Dies führt unter bestimmten Randbedingungen, wie z.B. bei einer hohen Anzahl beschleunigbarer Prozesse, zu wesentlich längeren Laufzeiten als bei FAST.
- Die Qualität der von rekursiven Verfahren gelieferten Ergebnisse ist sehr stark von der Wahl des Anfangs-Abbildung abhängig. TS-ECP besitzt dabei die beste Fähigkeit, sich auch bei ungünstiger Anfangsbedingung der optimalen Lösung anzunähern.

5.3 Ein Anwendungsbeispiel aus der Datenkommunikation

Die synthetischen Graphen spiegeln aufgrund der Art und Weise, wie sie von TGFF erzeugt werden, nur in eingeschränktem Maß reale Anwendungsfälle wieder. Da bei der Erzeugung zufällige Verbindungen der Knoten hergestellt werden, entstehen unstrukturierte Graphen, die sehr irregulär aufgebaut sind.
Derartige Graphen besitzen aufgrund dieser unregelmäßigen Verknüpfungen bei großer Knotenzahl und einem hohen Anteil bedingter Knoten sehr viele Condition Pfade, die beim Scheduling zu einem hohen Aufwand bei der Überprüfung der gegenseitigen Exklusivität führen. Für reale Anwendungen hingegen sind wesentlich strukturiertere Graphen zu erwarten, bei denen größere Bereiche existieren, die in sich geschlossen sind und bei denen nur in beschränkter Zahl (bedingte) Übergänge zwischen derartig abgeschlossenen Bereichen vorkommen. Die in den oben bei der Untersuchung von synthetischen Graphen mit hohem Anteil von bedingten Übergängen auftretenden hohen Laufzeiten sind deshalb bei zu realen Anwendungen gehörenden Graphen nicht zu erwarten.
Nach den künstlichen Graphen, die als Ersatz für reale Anwendungen in großer Menge zum Test des Verfahrens verwendet wurden, die diese aber nicht vollständig nachbilden können, soll nun ein Anwendungsbeispiel aus dem Bereich Networking zu weiteren Tests mit den betrachteten Verfahren herangezogen werden. Hierzu wird zunächst die Anwendung und die dazu betrachtete Systemumgebung erläutert, die hinter dem daraus entwickelten Prozeß-Graphen steht.

5.3.1 Anwendungsbeispiel Diffserv

Als Anwendungsbeispiel zur Verifikation des Verfahrens soll die empfangsseitige Paketverarbeitung auf einer Router Linecard dienen. Dabei wird die Bearbeitung der Header nach der Speicherung der Pakete im Datenspeicher bis zum Ablegen der erforderlichen Information über die weitere Behandlung in einer Warteschlange vor dem Weiterleiten über das Switching Netzwerk betrachtet.
Als Systemumgebung wurde dabei eine auf Ethernet basierte Schnittstelle angenommen. Die betrachtete Anwendung umfaßt zum einen Ethernet-Switching sowie alle zur Verarbeitung von IP Paketen notwendigen Aufgaben. Neben der grundsätzlichen Routing-Funktion werden außerdem noch Diffserv-Funktionalitäten (s. Kapitel 2.3.2) wie sie gemäß [10] und [85] ausgeführt werden müssen, mit einbezogen. Dies betrifft insbesondere die Klassifikation der Pakete, die Überwachung und Begrenzung des Datenflusses, in der Literatur als Metering und Marking (Policing) bezeichnet, sowie zusätzliche Statistik-Funktionen (Accounting).
Nur die Funktionen, die zum sogenannten "Fast Path" ([3]) gehören, d.h. die zeitkritischen Funktionen, die auf ein Paket anzuwenden sind, das sofort weitergeleitet werden muß, finden Berücksichtigung. Die Bearbeitung von Management- und Kontrollfunktionen, die üblicherweise weniger zeitkritisch ist und oft als "Slow Path" bezeichnet wird, bleibt unberücksichtigt. Diese Aufgaben werden in einem Router ohnehin nicht direkt auf der Linecard mit den dort vorhandenen Ressourcen bearbeitet.
Der grobe Ablauf, der die genannten Funktionen beinhaltet, ist in Bild 5-13 zusammenge-

faßt. Dabei werden nur die wichtigsten Aufgaben gekennzeichnet, die notwendigen Details für die einzelnen Protokolle sind nicht in Bild 5-13 aber im zugehörigen Graphen enthalten. Nachfolgend werden ausgewählte Funktionalitäten noch genauer dargestellt.

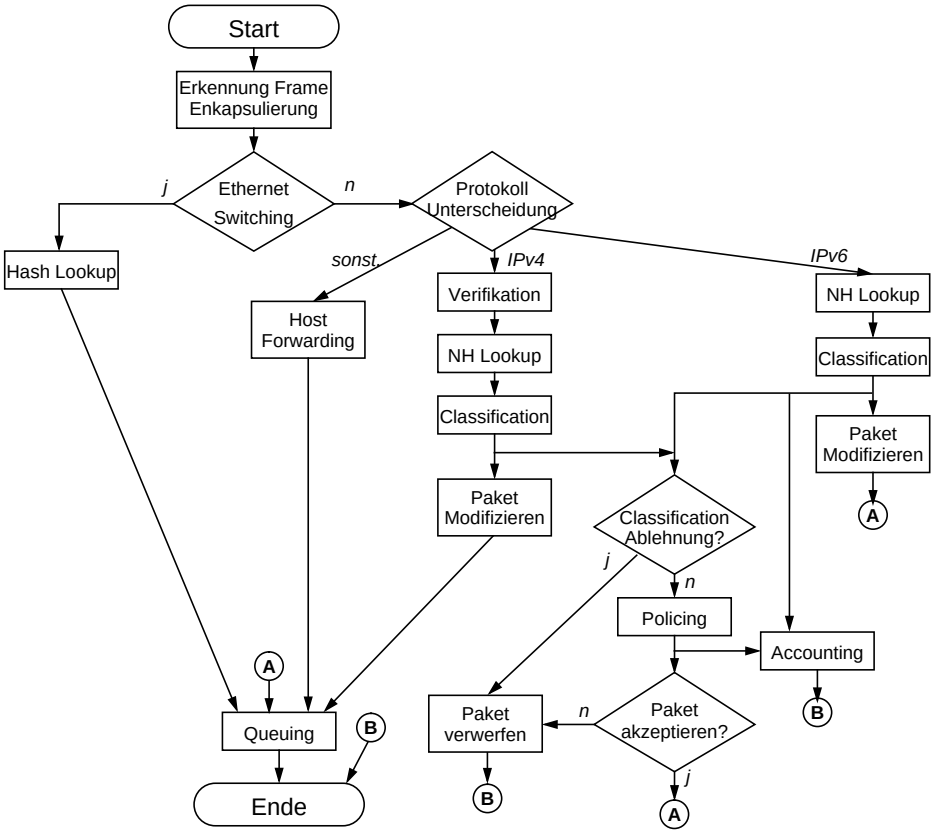


Bild 5-13: Ablaufdiagramm des Diffserv Beispiels

Von den gezeigten Funktionen sind einige in Bezug auf die insgesamt erreichbare Performance besonders relevant. Für diese Funktionen soll deshalb zunächst ein kurzer Überblick über die im Graphen berücksichtigten Algorithmen gegeben werden. Dies betrifft insbesondere die Routing-Entscheidung mit dem Nexthop (NH) Lookup, d.h. die Frage, an welchen Router ein Paket als nächstes auf dem Weg zum Ziel weitergeleitet werden muß. Daneben besitzt die Paket-Klassifikation, die über die Priorität, mit der das Paket behandelt werden muß, und die über die Zuordnung zu bestimmten Verkehrsströmen entscheidet, besondere Bedeutung.

Routing

IP Pakete müssen auf ihrem Weg durch das Internet in den Routern schrittweise weitergeleitet werden, bis sie ihr Ziel erreicht haben. In jedem Router ist deshalb für jedes Paket die Routing-Entscheidung zu treffen, die besagt, zu welchem Router das Paket als nächstes ge-

schickt werden muß, um näher in Richtung Ziel zu kommen. Konkret bedeutet dies festzulegen, auf welchem Interface des Routers das jeweilige Paket weitergeleitet werden muß.
Unter Nexthop Lookup ist zu verstehen, daß in einer Routing-Tabelle, der Eintrag für den nächsten Router und das zugehörige Interface gefunden werden muß. Dieser Lookup sucht den Eintrag, dessen Präfix die längste Übereinstimmung mit der IP Zieladresse des Pakets besitzt. Hinter dieser Anforderung steckt das Konzept des CIDR (Classless Interdomain Routing). Diese Konzept löste das auf unterschiedlichen Netzklassen basierte Routing ab, um die Anzahl der Einträge in Routing Tabellen zu begrenzen. Bei CIDR können mehrere Zielnetze mit zusammenhängendem Adreßbereich in einem gemeinsamen Tabelleneintrag erfaßt werden. Tabelleneinträge bestehen dabei aus einem Präfix und der Interface-Nummer, die mit dem darüber erreichbaren Router korrespondiert. Unter Präfix ist dabei ein Teil einer IP Adresse mit einer bestimmten signifikanten Länge zu verstehen. Meist wird ein Präfix in folgender Notation angegeben 129.187.155/24. Dies bedeutet, daß von der IP Nummer die 24 ersten Bits signifikant sind.
Hintergrund dieser Vorgehensweise ist die Annahme, daß je spezifischer ein Routing-Eintrag zur gewünschten Zieladresse paßt, d.h. je länger ein mit dem entsprechenden Teil der Zieladresse übereinstimmendes Präfix ist, desto näher führt der jeweilige Eintrag der Routing Tabelle das Paket zum Ziel.
Der Nexthop Lookup stellt den komplexesten Teil der Bearbeitung von IP Paketen in Routern dar und begrenzt damit den erreichbaren Durchsatz. Aus diesem Grund ist dieses Thema schon lange Zeit Gegenstand intensiver Forschung, zu dem sehr viele Arbeiten, sei es als dedizierte HW oder SW-Lösung veröffentlicht wurden. Die Vorschläge unterscheiden sich insbesondere in der benötigten Zeit für einen Lookup, ihrem Speicherbedarf, der erforderlichen Vorverarbeitung oder auch in Bezug auf die Möglichkeit Änderungen an der Datenbasis vornehmen zu können. An dieser Stelle sei lediglich auf [45] verwiesen, wo ein breiter Überblick über die aktuellen Verfahren gegeben wird.
Im Rahmen des Verifikationsbeispiels wird hier als Algorithmus für den Nexthop-Lookup das in [116] beschriebene Verfahren verwendet, das auf einer binären Suche in Hash Tabellen beruht, die nach Präfix-Längen sortiert sind. Für einen NH Lookup für IPv4 führt dieses Verfahren zu maximal 5, für IPv6 zu maximal 7 Speicherzugriffen. Da hier der Worst Case betrachtet werden soll, wird angenommen, daß jeder Lookup jeweils diese maximale Anzahl von Zugriffen benötigt.
Für die Routingtabelle wird davon ausgegangen, daß sie von einer Verarbeitungseinheit, die den langsamen Pfad bearbeitet, bereitgestellt und im gemeinsamen Datenspeicher abgelegt worden ist. Die Berechnung der zugehörigen Routing-Protokolle, die für die Routingtabelle notwendig ist, wird im Graphen selbst nicht berücksichtigt. Es ist jedoch eine Schnittestelle im Graphen vorhanden, über die derartige Pakete an diese Verarbeitungseinheit weiterleitet werden können, wenn erkannt wurde, daß es sich um entsprechende Pakete handelt.

Paket-Klassifikation

Elementarer Bestandteil des Diffserv Konzepts ist die Klassifizierung ankommender Pakete in einzelne Verkehrsflüsse (Flows). Diese Flows entsprechen unterschiedlichen Dienste-

Klassen, über die die weitere Behandlung der Pakete definiert ist ([10]). Mit Hilfe dieser für Pakete einer Klasse gleichen Behandlung, die auch als "Per Hop Behavior" bezeichnet wird, können bestimmte Dienstgütemerkmale wie z.B. Paketverlustwahrscheinlichkeit, Verzögerung oder Variation der Verzögerung (Jitter) relativ zu den verschiedenen Dienste-Klassen garantiert werden.
Die Klassifizierung wird anhand bestimmter Felder des Paketheaders vorgenommen und liefert als Ergebnis einen Klassen Identifikator (Class ID), der das jeweilige Paket in eine der möglichen Diensteklassen einordnet. Dies wird oft auch als Mehrfeld-Klassifizierung (Multifield-, MF Classification) bezeichnet. Für die angekommenen Pakete wird in den Klassifikationsregeln der erste Eintrag gesucht, der für die Felder im Paketheader zutrifft. Die zugehörige Class-ID wird damit dem Paket zugeordnet. Als Felder des Paketheaders, die für die Klassifizierung herangezogen werden, werden meist Quell- und Ziel-Internetadresse, das Type-Of-Service Feld (TOS), das Protokoll-Feld und aus der Transportschicht die Quell- und Ziel-Portnummern verwendet. Dazu kommt meist die Nummer der physikalischen Schnittstelle, auf der das Paket angekommen ist. Diese Wahl wird auch für den nachfolgend betrachteten Algorithmus zugrundegelegt.
Alternativ ist auch anstelle der MF Klassifikation, insbesondere innerhalb einer Diffserv Domäne, die Verwendung nur des TOS Felds bei IPv4 oder des Traffic Class Felds bei IPv6 direkt als Class ID denkbar. Dies setzt jedoch voraus, daß am Netzzugang bereits eine MF Klassifizierung stattgefunden hat. Für den hier betrachteten Anwendungsfall wird von MF Klassifikation ausgegangen.
Für die Multifeld Klassifikation wurden eine Vielzahl unterschiedliche Verfahren vorgeschlagen. Neben HW-basierten Lösungen z.B. mit ternären CAM (Content Adressable Memory) sind in den letzten Jahren diverse Algorithmen für eine SW Implementierung vorgeschlagen worden. Diese Algorithmen können in elementare Suchverfahren, geometrische oder heuristische Methoden unterschieden werden. Neben der Dauer für den Klassifikationsvorgang und den erforderlichen Speicher sind die Anzahl der unterstützen Felder, die Größe des möglichen Regelsatzes und auch dessen dynamische Erweiterbarkeit als Bewertungskriterien zu nennen. Eine Übersicht von aktuellen Vorschlägen zu MF Klassifikationsalgorithmen ist in [47] enthalten.
Für das Anwendungsbeispiel wird der RFC (Recursive Flow Classification) Algorithmus verwendet, der in [46] beschrieben ist. Bei diesem Verfahren wird durch eine komplexe Vorverarbeitung eine Datenstruktur im Speicher aufgebaut, die nach stufenweisen Speicherzugriffen und der Linearkombination der jeweiligen Ergebnisse im letzten Lookup das Ergebnis der Klassifikation liefert. Die Adressen werden dabei entweder aus einem Teil der Datenmenge, die als Kriterien der Klassifikation dienen (nachfolgend als Chunks bezeichnet), oder aus den jeweiligen Linearkombinationen der zusammengefaßten Ergebnisse der einzelnen Speicherzugriffe gebildet. Bei der Wahl dieser Parameter für den RFC Algorithmus ist ein Kompromiß zu finden zwischen der erforderlichen Anzahl von Speicherzugriffen und der für die Klassifikation benötigte Speichergröße.
Das nachfolgende Bild 5-14 zeigt ein mögliches Schema für IPv4, bei dem für den RFC Algorithmus 3 Phasen und 8 Chunks verwendet werden, und das der gewählten Anwendung zugrundeliegt.
Für den Fall IPv6 mit 128 Bit breiten Adressen wird ein entsprechend vergrößertes Schema

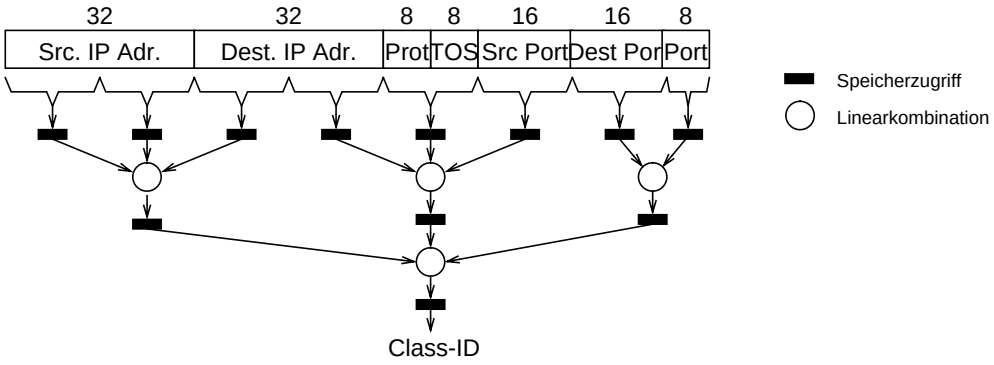


Bild 5-14: Verwendetes Chunk-Phasen Modell für den RFC Algorithmus (IPv4)

mit insgesamt 18 Anfangs-Chunks und insgesamt 4 Phasen gewählt, wobei insgesamt 27 Speicherzugriffe erforderlich sind.

Policing

Für das Policing wird angenommen, daß für jeden Nutzer (IP Adresse) unter Einschluß des Ergebnisses der Klassifikation ein eindeutiger Fluß-Identifikator berechnet wird, für den entsprechende Policing Parameter hinterlegt sind, und der zur Adressierung verwendet wird. Unter Policing Parametern sind Angaben zu verstehen, die einen Leaky Bucket mit einer durchschnittlichen Datenrate bei einer bestimmten erlaubten Bursthaftigkeit definieren. Diese Parameter werden pro Datenfluß überwacht. Falls ein Paket den erlaubten Datenfluß überschreitet, wird es verworfen.
Zu diesem Zweck müssen bei Ankunft eines Pakets nach der Berechnung der aus Class-ID und Source IP Adresse gebildeten Flow ID die zugehörigen Parameter und der letzte Füllstand des entsprechenden Leaky Bucket aus dem Speicher geholt, und der aktuell gültige Leaky Bucket Wert berechnet werden. Danach wird die angekommene Datenmenge aufaddiert und der aktuelle Füllstand gegen den erlaubten Maximalwert verglichen. Ist dieser überschritten, wird entweder das Paket sofort verworfen oder bzgl. der Class ID in seiner Priorität reduziert.
Es wird angenommen, daß die regelmäßige Aktualisierung der Zähler, die zur Vermeidung von Überläufen bei großen Paketabständen erforderlich ist, von einem übergeordneten Controller neben den sonstigen Managementaufgaben durchgeführt wird. Diese Aufgabe ist deshalb nicht im Graphen enthalten.

Für die vorstehend beschriebenen Funktionalitäten wurde der zugehörige Conditional Process Graph (CPG) entwickelt. Dabei ergab sich unter Berücksichtigung der in Kapitel 4.2.1 beschriebenen Randbedingungen zur Erstellung von Graphen eine Größe von insgesamt 149 Knoten. Von diesen besitzen insgesamt 18 Knoten bedingte Übergänge, was einem Anteil von 12% entspricht. Die im Mittel übertragene Datenmenge ist 56 Bit, der Anteil der beschleunigbaren Prozesse beträgt 17%, die mittlere Performance der Beschleuniger liegt

um den Faktor 9 höher als die der Prozessoren.

5.3.2 Architektur

Als Basis für die Untersuchung wird eine busbasierte Architektur mit folgenden abstrakten Ressourcen verwendet.

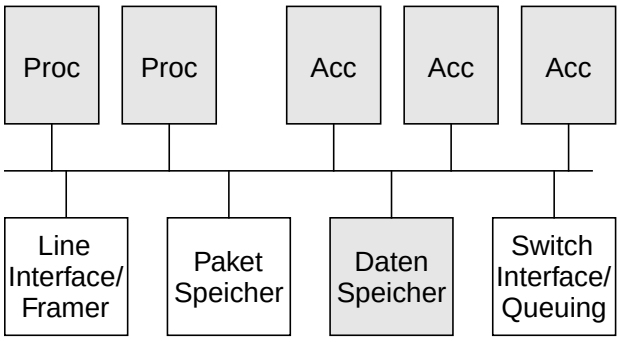


Bild 5-15: Zielarchitektur für das Diffserv Beispiel

Neben einem oder zwei eingebetteten Prozessoren sind mehrere Beschleuniger vorhanden, in denen Nexthop Lookups, Classification, Policing und allgemeine Tabellenlookups schnell durchgeführt werden können. Im System befinden sich neben diesen Verarbeitungsressourcen entsprechende Schnittstellen, die zum einen für den Empfang der Pakete vom Framer und zum anderen zur Weiterleitung an das Switching Netzwerk zuständig sind, und in dem auch das Queuemanagement durchgeführt wird. Daneben sind noch zwei unabhängige Speicher im System, die zum einen zum Ablegen der kompletten Paketdaten, zum anderen zur Speicherung von Lookup-Tabellen und aller anderen Daten, wie Paketdeskriptoren, Accounting-Daten und Policing-Parameter, dienen.
Die in Bild 5-15 grau hinterlegten Blöcke der Architektur werden bei der Untersuchung betrachtet. Die übrigen Module sind nur für den Empfang der Pakete und deren Speicherung sowie für das Queueing und Weiterleiten relevant, was im beschriebenen Beispiel nicht enthalten ist.
In den Untersuchungen an synthetischen Graphen hatte sich gezeigt, daß Eigenschaften des Graphen wesentlich höheren Einfluß auf die Leistungsfähigkeit der Verfahren besitzen als die Parameter der Architektur. Insbesondere das Verhältnis der Performance-Werte der Ressourcen haben nur geringen Einfluß auf die von den jeweiligen Verfahren gelieferte relative Performance (s. Bild 5-9). Deshalb wurden die für die Untersuchung des Diffserv-Beispiels verwendeten Daten für die Prozessoren bzw. HW-Beschleuniger in ihrer Größenordnung unter Berücksichtigung der jeweils durchzuführenden Teilfunktionen nur grob abgeschätzt.
Als beschleunigbare Funktionen wurden die unterschiedlichen Lookups (NH Lookup und Hash Lookup), die Classification sowie das Policing angenommen, da für diese Funktionalitäten auch entsprechende IP Module von kommerziellen Anbietern verfügbar sind. Damit

können abhängig von der Abbildungsentscheidung der Algorithmen in der Architektur neben den Prozessoren bis zu 4 Beschleuniger zum Einsatz kommen.

5.3.3 Ergebnisse

Die Anwendung der verschiedenen Algorithmen zur Abbildung und zum Scheduling der Diffserv Funktionalität auf die Zielarchitektur mit einem oder zwei Prozessoren liefert folgende Ergebnisse.

	Prozessoren	TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
Schedule Dauer *)	1	-1.00%	-0,00%	-1,92%	-	+15,88%
Laufzeit **)		7,14 s	0,37 s	0,83 s	4,31 s	0,02 s
Schedule Dauer *)	2	-1.01%	-2.66%	-3.58%	-	+13,3%
Laufzeit **)		5,29 s	2,61 s	3,38 s	3,33 s	0,02 s
*) relativ zu FAST, **) AMD Athlon XP 1700+						

Tabelle 5-7: Ergebnisse des Diffserv Beispiels

Dem vorliegenden Anwendungsbeispiel mit 2 Prozessoren entspricht als Referenz in etwa das oben definierte Standardproblem, jedoch mit einer Graphgröße von 150 statt 100 Knoten. Zur Einordnung der erhaltenen Ergebnisse ist jedoch zu berücksichtigen, daß es sich beim Diffserv-Beispiel um einem einzelnen Graphen handelt, während die Resultate aus Kapitel 5.2 aus der Mittelung der Ergebnisse für jeweils 60 unterschiedliche synthetische Graphen gewonnen wurden. In der nachfolgenden Tabelle sind deshalb für die synthetische Referenzkonfiguration neben den zugehörigen Mittelwerten m_i, auch die Standardabweichungen σ_i der jeweiligen Schedule Dauer relativ zu FAST aufgelistet.

		TS-One	TS-CP	TS-ECP	FAST	Xie/Wolf
Schedule Dauer *)	m_i [%]	-4,87	0,43	-4,15	-	6,99
	σ_i [%]	3,16	3,78	3,10	-	7,95
Laufzeit [s] **)		58,65	6,16	25,63	17,17	0,13
*) relativ zu FAST, **) AMD Athlon XP 1700+						

Tabelle 5-8: Ergebnisse für das Standardproblem bei synthetischen Graphen mit 150 Knoten

Die Ergebnisse für den Graphen der Diffserv-Anwendung entsprechen im Trend den aus den Untersuchungen mit den synthetischen Graphen resultierenden Werten. Dabei weichen die Performancewerte aber teilweise erheblich von den Mittelwerten der Referenz ab. Diese Beobachtung relativiert sich jedoch bei Berücksichtigung der jeweiligen Werte für σ. Deshalb sollen die bei der Untersuchung der entsprechenden künstlichen Graphen gewonnenen Einzelergebnisse dem Resultat des Diffserv-Beispiels gegenübergestellt werden. Im Bild 5-16 sind die zu den verschiedenen Verfahren gehörenden Ergebnisse verglichen. Für

die einzelnen Verfahren sind oben jeweils die Einzelergebnisse der 60 synthetischen Graphen aufgetragen, während darunter als einzelnes, fett gedrucktes Symbol das entsprechende Ergebnis des Diffserv-Beispiels eingezeichnet ist.

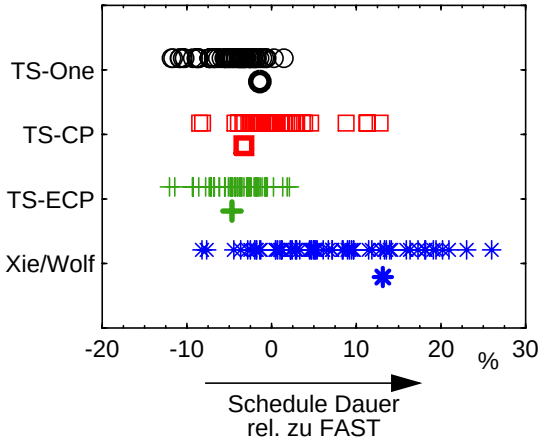


Bild 5-16: Vergleich der Ergebnisse des Diffserv Beispiels mit den Einzelergebnissen bei synthetischen Graphen

Die Ergebnisse für das Diffserv-Beispiel liegen trotz der teilweise erheblichen Abweichung von den Mittelwerten im Streubereich der für die Tests mit synthetischen Graphen erhaltenen Einzelwerte. Daraus wird geschlossen, daß die Ergebnisse für das konkrete Anwendungsbeispiel in Bezug auf die sich ergebende Performance der Verfahren konform mit den Experimenten sind, die mit den synthetischen Graphen durchgeführt wurden.
Die Laufzeiten aller Verfahren liegen jedoch erheblich unter den Werten, die sich bei entsprechenden künstlich erzeugten Graphen ergeben. Dies ist in erster Linie auf die Struktur des Graphen zurückzuführen, der, wie oben erläutert, wesentlich regulärer und damit erheblich unkritischer bei der Überprüfung der gegenseitigen Exklusivität ist als die synthetisch erzeugten Beispiele.

5.4 Zusammenfassung der Ergebnisse des rekursiven Verfahrens

Die in diesem Kapitel beschriebenen Experimente sowohl mit synthetisch erzeugten Graphen als auch mit der Diffserv Anwendung zeigen die Einsetzbarkeit des vorgeschlagenen rekursiven Verfahrens für die Untersuchung von Graphen mit Kontrollabhängigkeiten. Im Rahmen der Rekursionsschleife können verschiedene Verfahren zur Suche im Lösungsraum eingesetzt werden, die unterschiedliches Verhalten in Bezug auf die Lösungsgüte und die benötigte Laufzeit aufweisen.
Unter den rekursiven Varianten liefert das Verfahren mit der Tabu Suche in der erweiterten Nachbarschaft des kritischen Pfades ähnliche Ergebnisse wie die ungerichtete Suche, die

jedoch wesentlich höhere Laufzeiten verursacht. Die Einbeziehung der Prozesse in die Nachbarschaft, die zusätzliche Verzögerungen von im kritischen Pfad liegenden Knoten verursacht, führt zu wesentlichen Verbesserungen. Insbesondere diese Suchmethode besitzt im Vergleich zu allen anderen hier betrachteten rekursiven Methoden die besten Eigenschaften, sich von ungünstigen Anfangslösungen dem Optimum anzunähern. Die ausschließlich auf den konventionellen kritischen Pfad beschränkte Suche schneidet durchweg schlechter ab. Der Einsatz von FAST als limitierte Suchmethode in diesem Umfeld leidet wie auch TS-CP unter der damit zusammenhängenden Einengung des Suchbereichs, die die Lösungsfindung eingrenzt und deshalb zu geringerer Performance als TS-ECP und TS-One führt.
Im Vergleich zu dem konstruktiven Verfahren von Xie und Wolf führen jedoch alle Varianten des rekursiven Verfahrens zu besseren Ergebnissen. Ein erheblicher Nachteil des rekursiven Verfahrens besteht aber in der sehr viel höheren Laufzeit, die seinen Einsatz auf Anwendungen mit kleiner und mittlerer Graphgröße beschränkt.

6 Zusammenfassung, Bewertung und Ausblick

6.1 Zusammenfassung und Bewertung

In der vorliegenden Arbeit wurde ein rekursives Verfahren zur Abbildung und zum Scheduling von Prozeß-Graphen entwickelt, das die Unterstützung von Kontrollabhängigkeiten beinhaltet. Diese Eigenschaft wird mit Hilfe des Prinzips der gemeinsamen Ressourcennutzung realisiert, das es erlaubt, gegenseitig exklusive Prozesse zeitlich überlappend derselben Ressource zuzuordnen. Durch diesen Ansatz werden Lösungen erzeugt, die die ungünstigste Situation erfassen. Damit ist gewährleistet, daß bei keiner möglichen Kombination von Bedingungen eine Schedule Dauer resultiert, die länger als die durch das Verfahren gelieferte ist. Zentraler Punkt bei dieser Vorgehensweise ist die Feststellung der gegenseitigen Exklusivität zweier Prozesse. Um die im betrachteten Anwendungsgebiet möglichen beliebigen und nicht auf Basic Blocks begrenzten bedingten Übergänge zu erlauben, wurde hierzu ein neues, auf Bedingungspfaden beruhendes Verfahren entwickelt.
Im Rahmen der rekursiven Vorgehensweise, die sich in die Schritte Abbildung, Scheduling und Optimierung untergliedert, können unterschiedliche Optimierungsansätze zum Einsatz kommen. Als Input des Verfahrens müssen zum einen die zu realisierende Funktion in Form eines Graphen und zum anderen die entsprechenden Informationen zur Architektur, auf die die Funktion abgebildet werden soll, vorhanden sein. Dies betrifft insbesondere die in der Architektur vorhandenen Ressourcen sowie die Schätzungen der Ausführungsdauer der einzelnen Prozesse auf den jeweils möglichen Verarbeitungsressourcen der Architektur. Als Ergebnis liefert das Verfahren die Zuordnung der einzelnen Prozesse auf die Ressourcen sowie die zeitliche Abfolge der Abarbeitung der Prozesse und der Transaktionen auf der internen Kommunikationsinfrastruktur. Das vorliegende Verfahren ist hierbei auf Architekturen mit einem von allen Verarbeitungsressourcen gemeinsam genutzten Bus beschränkt.
Als Optimierungsverfahren wurden zum einen Simulated Annealing als auch Tabu Suche betrachtet. Da sich bei ersten Untersuchungen Simulated Annealing als zu wenig performant bei gleichzeitig wesentlich höheren Laufzeiten erwies, und auch in verschiedenen Untersuchungen in der Literatur zu konventionellen Graphen Tabu Suche favorisiert wurde, wurde im weiteren nur noch dieser Ansatz verfolgt.
Für die auf Tabu Suche basierende Optimierung wurden unterschiedliche Nachbarschaftdefinitionen untersucht. Neben einer ungerichteten und sehr breit suchenden Variante wurden zwei Nachbarschaftsdefinitionen betrachtet, die am kritischen Pfad der Anwendung orientiert sind. Zum einen handelt es sich um eine Nachbarschaft, die ausschließlich durch die Prozesse des kritischen Pfads gebildet wird. Um die Einflüsse der systeminternen Kommunikation bei der Abbildung erfassen zu können, wurde zum anderen eine neue, ebenfalls am kritischen Pfad orientierte, jedoch um zusätzliche, Verzögerungen erzeugende Prozesse erweiterte Nachbarschaft vorgeschlagen. Mit dieser Nachbarschaft können dynamische Effekte auf der Kommunikationsarchitektur bei der Abbildungsentscheidung mit berücksichtigt werden. Neben der Tabu Suche wurde mit FAST als Alternative ein statistisches Sucherverfahren als Optimierungsmethode betrachtet.

Zur Untersuchung der Leistungsfähigkeit dieser Varianten der rekursiven Abbildungs- und Scheduling-Methode wurden Experimente sowohl mit synthetisch erzeugten Graphen als auch mit einer Anwendung aus dem Bereich Datenkommunikation durchgeführt. Da aus der Literatur bisher keine rekursive Methode mit Unterstützung von Kontrollabhängigkeiten bekannt ist, konnte kein Vergleich mit einem entsprechenden, etablierten Verfahren durchgeführt werden. Zur Einordnung der Ergebnisse wurde jedoch eine Gegenüberstellung mit dem konstruktiven Ansatz von Xie und Wolf vorgenommen.
Die Ergebnisse der Experimente zeigen für alle Varianten des rekursiven Ansatzes durchgehend bessere Lösungen als das konstruktive Verfahren. Lediglich für einige wenige Parameterkonstellationen bei synthetischen Graphen ergeben sich leichte Vorteile für das konstruktive Verfahren gegenüber dem FAST Suchansatz. Unter den rekursiven Suchvarianten besitzen die breit angelegte Nachbarschaft und die erweiterte, am kritischen Pfad orientierte Version die beste Performance, wobei letztere dies jedoch mit wesentlich kürzeren Laufzeiten erreicht. Im Vergleich der beiden am reinen kritischen Pfad orientierten Suchmethoden liefert überwiegend FAST das bessere Ergebnis, wobei das auf Tabu Suche basierende Verfahren für die hier betrachteten Graphgrößen meist kürzere Ausführungsdauern besitzt. Generell kann festgestellt werden, daß insbesondere die Variante mit Tabu Suche in der erweiterten Umgebung des kritischen Pfades zu erheblich besseren Lösungen führt. Bei der Diffserv-Anwendung werden dabei Lösungen mit um ca. 15% kürzeren Schedule Dauern generiert als mit dem Verfahren von Xie und Wolf.
Die Vorteile resultieren zum einen aus der prinzipiellen Möglichkeit durch rekursive Änderungen schrittweise Verbesserungen zu erreichen, wobei jeweils der gesamte Schedule und die Auswirkung alternativer Abbildungen berücksichtigt werden können. Demgegenüber kann im konstruktiven Ansatz nur ein eingeschränkter Teil des Schedule in die Entscheidung miteinbezogen werden, die auch nicht mehr revidierbar ist. Außerdem wird mit den Erweiterungen des kritischen Pfades, die insbesondere Ressourcenkonflikte v.a. auf der internen Kommunikationsarchitektur erfassen, weitere Prozesse bei der Suche berücksichtigt, die hohen Einfluß auf die Performance der Lösung besitzen.
Ein erheblicher Nachteil der rekursiven Methode im Vergleich zu Xie/Wolf besteht jedoch darin, daß der Algorithmus wesentlich höhere Laufzeiten benötigt. Dies liegt zum einen am wiederholten Überprüfen alternativer Lösungen, als auch an der durch die Erweiterungen vergrößerten Nachbarschaft. Das rekursive Verfahren ist deshalb nur für kleine und mittelgroße Graphen in der Größenordnung von einigen Hundert Knoten im Rahmen der Architekturexploration geeignet, bei der in mehreren Durchläufen Untersuchungen verschiedener Zielarchitekturen durchgeführt werden müssen. Im Zusammenhang mit der Untersuchung der Diffserv-Anwendung hat sich jedoch gezeigt, daß reale Anwendungen durchaus in dieser Größenordnung liegen.

6.2 Ausblick

In der vorliegenden Arbeit wurde ein neues Verfahren zur Partitionierung von Systemen entwickelt, das Kontrollabhängigkeiten in der Anwendung unterstützt. Mit der erweiterten Nachbarschaft des kritischen Pfads, wodurch mehr Informationen über die Engpässe im System in die Entscheidung über die Partitionierung einfließen, wurde ein Ansatz vorge-

stellt, der auch der zunehmenden Relevanz der Kommunikation Rechnung trägt.
Das Verfahren optimiert dabei ausschließlich die von der Lösung benötigte Schedule Dauer. Für eine Ausweitung des Verfahrens um andere Kriterien wie Flächenbedarf, Kosten oder Verlustleistung der jeweiligen Ressourcen müssen jedoch komplexere Optimierungsansätze eingesetzt werden. Diese Aussage gilt insbesondere auch in Bezug auf eine zusätzliche Erweiterung des Verfahrens in Richtung Architektursynthese, für die eine mehrdimensionale Optimierung ebenfalls eine Voraussetzung darstellt.
Die Berücksichtigung einer komplexeren Kommunikationsarchitektur als des jetzt unterstützen gemeinsamen Busses kann entweder auf der Ebene der Zielarchitektur durch Vorgaben bzgl. der Zuordnung bestimmter Transfers zu einzelnen Kommunikationsressourcen oder allgemeiner über ein zweistufiges Abbildungsverfahren erfolgen. Im letztgenannten Fall würden nach der Zuordnung der Prozesse, die über die Notwendigkeit von Transfers bestimmen, diese Transfers in einem zweiten Schritt auf jeweils mögliche Kommunikationsressourcen abgebildet werden. Dieses Vorgehen vergrößert die Anzahl der Abbildungsmöglichkeiten ggf. erheblich und würde die Problematik der Laufzeit des Algorithmus verschärfen. Die erstgenannte Variante ist mit sehr geringem Aufwand möglich, verlagert die Frage der Zuordnung aber auf die Ebene der Architekturdefinition.
Sollen mit der vorliegenden Methode Architekturen untersucht werden, die die zeitlich überlappende Bearbeitung mehrerer Datenpakete ermöglichen, bedeutet dies die Abbildung einer entsprechenden Anzahl unabhängiger Instanzen des Prozeßgraphen, deren Kommunikationsmuster sich ebenfalls auf der Kommunikationsinfrastruktur überlagern. Ein erster Ansatz hierfür kann die Replikation von mehreren Graphen sein, die mit den jeweiligen Zwischenankunftszeiten entsprechenden Verzögerungen versehen einen neuen komplexen Graphen darstellen. Zur korrekten Berücksichtigung der gleichartigen Zuordnung korrespondierender Prozesse aus den parallelen Graph-Instanzen, entweder unabhängig auf parallel laufende oder konkurrierend auf gemeinsam genutzte Ressourcen, sind jedoch Erweiterungen erforderlich, die bisher von der bestehenden Methode nicht unterstützt sind.

Die Bereitstellung von Methoden und Werkzeugen zur Untersuchung und Generierung geeigneter VLSI Architekturen ist eine entscheidende Aufgabe, um die von der Produktionstechnik bereitgestellten Möglichkeiten der Integration besser nutzen zu können. Mit der vorliegenden Arbeit wurde hierzu ein Beitrag geliefert, der die Definition einer im Hinblick auf die Verarbeitungsdauer optimierte Architektur für Anwendungen mit Kontrollabhängigkeiten unterstützt. Für die Zukunft bleiben jedoch noch viele Aufgaben zu lösen, um den Einstieg in den rechnergestützten Designprozeß auf höherem Abstraktionsniveau als bisher üblich durch die Bereitstellung entsprechender Explorations- und Synthesewerkzeuge zu ermöglichen.

Literaturverzeichnis

[1] "ARCtangent", Datenblatt, ARC International plc, The Waterfront, Elstree Road, Elstree, Herts, WD6 3BS, UK, http://www.arc.com/products/arctangent.htm

[2] J. Axelsson, "Three Search Strategies for Architecture Synthesis and Partitioning of Real-Time Systems", IDA Technical Report, LiTH-IDA-R-96-32, 1996

[3] J. Aweya, "On the Design of IP Routers Part1: Router Architectures", Journal of Systems Architecture, Vol. 46, No. 6, S. 483-511, 2000

[4] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A, Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, B. Tabbara, "Hardware-Software Co-Design of Embedded Systems: The Polis Approach", Kluwer Academic Press , Juni 1997

[5] Balboni, W. Fornaciari, and D. Scuito, “Partitioning and Exploration Strategies in the TOSCA Design Flow”, Proc. 8th. Int. Symp. System Synthesis, S. 10-15, 1995

[6] J.R. Bammi, E. Harcourt, W. Kruijtzer, L. Lavagno, M.T. Lazarescu, "Software Performance Estimation Strategies in a System-Level Design Tool", Proc. Int. Workshop on Hardware/Software Codesign 2000, S. 82-87, 2000

[7] E. Barros, W. Rosenstiel, X. Xiong, "A Method for Partitioning UNITY Language in Hardware and Software", Proc. EURODAC ’94, S. 220-225, 1994

[8] S. Bakshi and D. D. Gajski, "Partitioning and Pipelining for Performance-Constrained Hardware/Software Systems", IEEE Trans. VLSI Systems, Vol. 7, No. 4, S. 419-431, 1999

[9] G. Berry, "The Esterel v5 Language Primer, Version 5.21 release 2.0", ftp://ftp-sop.inria.fr/meije/esterel/papers/primer.pdf, April 1999

[10] S. Blake, et al., "An Architecture for Differentiated Services", RFC2475, Dezember 1998

[11] "Cadence Virtual Component Co-Design (VCC)", Datenblatt, Cadence Design Systems Inc., 2655 Seely Avenue, San Jose, CA 95134, USA, http://www.cadence.com

[12] "Incisive-SPW", Datenblatt, Cadence Design Systems Inc., 2655 Seely Avenue, San Jose, CA 95134, USA, http://www.cadence.com/products/incisive_spw.html

[13] L.P. Carloni, F. De Bernardinis, A.L. Sangiovanni-Vincentelli, M. Sgroi, "The Art and Science of Integrated Systems Design", Proc. 28th European Solid-State Circuits Conference, Sept. 2002

[14] W.O. Cesário, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, A.A. Jerraya, L. Gauthier, M. Diaz-Nava, "Multiprocessor SoC Platforms: A Component-Based Design Approach", IEEE Design & Test of Computers, Vol. 19, No. 6, S. 52-63, November-Dezember 2002

[15] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, L. Todd, “Surviving the SOC Revolution: A Guide to Platform-Based Design”, Kluwer, 1999

[16] J.-M. Chang and M. Pedram, “Codex-dp: Co-Design of Communicationg Systems Using Dynamic Progamming”, IEEE Trans. Computer-Aided Design, Vol. 19, No. 7, S. 732-744, 2000

[17] K.S. Chatha, R. Vemuri, "An Iterative Algorithm for Hardware-Software Partitioning, Hardware Design Space Exploration and Scheduling", Design Automation for Embedded Systems, Vol. 5, S. 281-293, 2000

[18] F. Cieslok, R. Ernst, M. Jersak, K. Richter, K. Strehl, J. Teich, L. Thiele, F. Wolf, D. Ziegenbein, "Embedded System Design using the SPI Workbench", Proc. FDL'00, Forum on Design Languages 2000, September 2000

[19] D.E. Comer, "Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture", Prentice Hall, Englewood Cliffs, New Jersey, 1995

[20] "CoWare N2C", Datenblatt, CoWare Inc., 2121 North First Street, San Jose, CA 95131, USA, http://www.coware.com/cowareN2C.html

[21] B.P. Dave, N.K. Jha, "COHRA: Hardware-Software Co-Synthesis of Hierarchical Distributed Embedded System Architectures", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 17, No. 10, S. 900-919, Oktober 1998

[22] B.P. Dave, G. Lakshminarayana, N.K. Jha, "COSYN: Hardware-Software Co-Synthesis of Embedded Systems", Proc. DAC 1997, S. 703-708, 1997

[23] J. Davis, C. Hylands, J. Janneck, E.A. Lee, J. Liu, S. Neuendorffer, S. Sachs, M. Stewart, K. Vissers, P. Whitaker, Y. Xiong, "Overview of the Ptolemy Project", Technischer Bericht UCB/ERL M01/11, University of California, Berkeley, März 2001

[24] R.P. Dick, N.K. Jha, "MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Distributed Embedded Systems", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 17, No. 10, S. 920-935, Oktober 1998

[25] R. P. Dick, D. L. Rhodes, W. Wolf, "TGFF: Task Graphs For Free", Proc. Int. Workshop Hardware-Software Codesign, 1998.

[26] Quelle zum Download von TGFF: http://www.ee.princeton.edu/~cad/projects.html

[27] M. Drinic, D. Kirovski, S. Meguerdichian, M. Potkonjak, "Latency Guided On-Chip Bus Network Design", ICCAD'00, S. 420-423, 2000

[28] S. Edwards, L. Lavagno, E.A. Lee, A. Sangiovanni-Vincentelli, "Design of Embedded Systems: Formal Models, Validation, and Synthesis", Proc. of the IEEE, Vol. 85, Nr. 3, S. 366-390, März 1997

[29] M. Eisenring, L. Thiele, E. Zitzler, "Conflicting Criteria in Embedded System Design", IEEE Design & Test of Computers, Vol. 17, No. 2, S. 51-59, April-Juni 2000

[30] H. El-Rewini, T.G. Lewis, H.H. Ali, "Task Scheduling in Parallel and Distributed Systems", Prentice Hall, 1994

[31] P. Eles, A. Doboli, P. Pop, Z. Peng, "Scheduling with Bus Access Optimization for Distributed Embedded Systems", IEEE Trans. VLSI Systems, Vol. 8, No. 5, S. 472-491, Oktober 2000

[32] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli, P. Pop, "Scheduling of Conditional Process Graphs for the Synthesis of Embedded Systems", Proc. DATE '98

[33] P. Eles, Z. Peng, K. Kuchcinski, A. Doboli, "System Level Hardware/Software Partitioning based on Simulated Annealing and Tabu Search", Journal on Design

Automation for Embedded Systems, Vol. 2, S. 5-32, 1997

[34] R. Ernst, "Codesign of Embedded Systems: Status and Trends", IEEE Design & Test of Computers, Vol. 15, No. 2, S. 45-54, April-Juni 1998

[35] R. Ernst, J. Henkel, T. Benner, "Hardware-Software Cosynthsis for Microcontrollers", IEEE design & Test of Computers, Vol. 10, No. 4, S. 64-75, 1993

[36] "NP-1 Family", EZchip Technologies Ltd., 1 Hatamar Street, PO Box 527, Yokneam 20692, Israel, http://www.ezchip.com/html/pr_np-1.html

[37] D.D. Gajski, N.D. Dutt, A.C-H. Wu, S.Y-L. Lin., "High-Level Synthesis: Introduction to Chip and System Design", Kluwer Academic Publishers, 1992

[38] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, "Specification and Design of Embedded Systems", Prentice Hall, 1994

[39] M.R. Garey, D.S. Johnson, "Computers and Intractability - A Guide to the Theory of NP-Completeness", W.H. Freeman and Co., 1979

[40] M. Gasteier, M. Glesner, "Bus-Based Communication Synthesis in System Level", ACM Transactions on Design Automation of Electronic Systems, Vol. 4, No.1, S.1-11, 1999

[41] S.H. Gerez, "Algorithms for VLSI Design Automation", J. Wiley, 1999

[42] F. Glover, E. Taillard, D. de Werra, "A User's Guide to Tabu Search", Annals of Op. Research, Vol. 21, S. 3-28, 1993

[43] G. G. Gogniat, M. Augustin, L. Blanco, and A. Pegatoquet, "Communication synthesis and HW/SW integration for Embedded System Design", Proc. Int. Workshop Hardware-Software Codesign, 1998

[44] R. Gupta, G. De Micheli, "Hardware-Software Cosynthesis for digital Systems", IEEE Design & Test of Computers, Vol. 10, No. 3, S. 29-41, 1993

[45] P. Gupta, "Algorithms for Routing Lookups and Packet Classification", Dissertation, Stanford University, Dezember 2000

[46] P. Gupta, N. McKeown, "Packet Classification on Multiple Fields", Proc. ACM SIGCOMM 1999, S. 147-160, September 1999

[47] P. Gupta, N. McKeown "Algorithms for Packet Classification", IEEE Network, Vol. 15, No. 2, S. 24-32, März/April 2001

[48] J. Henkel, "Automatisierte Hardware/Software Partitionierung im Entwurf integrierter Echtzeitsysteme", Dissertation, Technische Universität Braunschweig, 1996

[49] J. Henkel, R. Ernst, "A Hardware/Software Partitioner using a dynamically determined Granularity", Proc. Design Automation Conference, S. 691-696, 1997

[50] J. Henkel, R. Ernst, "High-Level Estimation Techniques for Usage in Hardware/ Software Co-Design", Proc. IEEE/ACM Asia and South Pacific Design Automation Conference, S. 353-360, 1998

[51] "PowerNP Network Processors", Übersicht, International Business Machines Corporation, New Orchard Road, Armonk, NY 10504, USA, http://www-3.ibm.com/chips/products/wired/products/network_processors.html

[52] "Statemate MAGNUM", Datenblatt, I-Logix Inc., 3 Riverside Drive, Andover,

MA 01810, USA, http://www.ilogix.com/products/magnum/index.cfm

[53] "Intel Network Processors", Übersicht, Intel Corporation, 2200 Mission College Blvd., Santa Clara, California 95052-8119, USA, http://www.intel.com/design/network/products/npfamily/index.htm

[54] Internet Software Consortium, "Internet Domain Survey", Internet Software Consortium, 950 Charter Street, Redwood City, CA 94063, USA, http://www.isc.org/ds/

[55] M.F. Jacome, H.P. Peixoto, "A Survey of Digital Design Reuse", IEEE Design & Test of Computers, Vol. 18, No. 3, S. 98-107, May-June 2001

[56] A. Jantsch, I. Sander, "On the Roles of Functions and Objects in System Specification", Proc. CODES 2000, S. 8-12, 2000

[57] H.-P. Juan, V. Chaiyakul, D. Gajski, "Condition Graphs for High-Quality Behavioral Synthesis", UC Irvine, Technical Report ICS-TR-94-32, August 1994

[58] A. Kalavade, E.A. Lee, "The Extended Partitioning Problem: Hardware/Software Mapping, Scheduling, and Implementation-bin Selection", Design Automation for Embedded Systems, Vol. 2, No. 2, S. 125-163, 1997

[59] F. Karim, A. Nguyen, A. Dey, R. Rao, "On-chip Communication Architecture for OC-768 Network Processors", Proc. DAC 2001, S. 678-683, 2001

[60] H. Kasahara, S. Narita, "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing", IEEE Trans. on Computers, Vol. 33, No. 11, S. 1023-1029, 1984

[61] B. Kienhuis, E. Deprettere, K. Vissers, P. van der Wolf, "An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures", Proc. ASAP ’97, 1997

[62] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vechi, "Optimization by Simulated Anealing", Science, Vol. 220, No. 4598, S. 671-680, 1983

[63] P.V. Knudsen, J. Madsen, "Communication Estimation for Hardware/Software Codesign", Proc. Int. Workshop Hardware-Software Codesign, 1998

[64] P.V. Knudsen, J. Madsen, "Integrating communication protocol selection with hardware/software codesign", IEEE Trans. Computer-Aided Design, Vol. 18, No. 8, S. 1077-1095, 1999

[65] P.V. Knudsen, J. Madsen, "PACE: A Dynamic Programming Algorithm for Hardware/Software Partitioning", Proc. 4th. Int. Workshop CODES, S. 85-92, 1996

[66] K. Kreutzer, S. Malik, A. R. Newton, J. M. Rabaey, A. Sangiovanni-Vincentelli. "System-Level Design: Orthogonalization of Concerns and Platform-Based Design", IEEE Trans. Computer-Aided Design, Vol. 19, No. 12, S. 1523-1543, 2000

[67] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, A. Hemani, "A network on chip architecture and design methodology", Proceedings of IEEE Computer Society Annual Symposium on VLSI, April 2002

[68] Y-K. Kwok, I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors", ACM Computing Surveys, Vol. 31, No. 4, 1999

[69] Y-K. Kwok, I. Ahmad, "FASTEST: A Practical Low Complexity Algorithm for

Compile-Time Assgnment of Parallel Programs to Multiprocessors", IEEE Trans. on Parallel and Distributed Systems, Vol. 10, No. 2, Februar, 1999

[70] Y-K. Kwok, I. Ahmad, J. Gu, "FAST: A Low-ComplexityAlgorithm for Efficient Scheduling of DAGs of Parallel Processors", Proc. Int. Conference on Parallel Processing, Vol. II, S. 150-157, 1996

[71] K. Lahiri, A. Raghunathan, S. Dey, "Fast Performance Analysis of Bus-Based System-On-Chip Communication", Proc. Int. Conf. CAD, S. 566-573, 1999

[72] K. Lahiri, A. Raghunathan, S. Dey, "Efficient Exploration of the SoC Communication Architecture Design Space", Int. Conf. on CAD, 2000

[73] K. Lahiri, A. Raghunathan, G. Lakshminarayana, and S. Dey, "Communication Architecture Tuners: A Methodology for the Design of High-Performance Communication Architectures for System-on-Chips", Proc. 37th DAC, S. 513-518, 2000

[74] M. Lajolo, M. Lazarescu, A. Sangiovanni-Vincentelli, "A Compilation-based Software Estimation Scheme for Hardware/Software Co-simulation", Proc. Int. Workshop on Hardware/Software Codesign. 1999, S. 85-89, 1999

[75] C.K. Lennard, P. Schaumont, G. de Jong, "Standards for System-Level Design: Practical Reality or Solution in Search of a Question?", Proc. DATE 2000, S. 576-585, 2000

[76] J. Li, R.K. Gupta, "An Algorithm to Determine Mutually Exclusive Operations in Behavioral Descriptions", Proc. DATE 1998

[77] P. Lieverse, P. van der Wolf, E. Deprettere, K. Vissers, "A Methodology for Architecture Exploration of heterogeneous Signal Processing Systems", Proc. IEEE Workshop on Signal Processing Systems (SiPS), S. 181-190, 1999

[78] P. Lieverse, P. van der Wolf, E. Deprettere, "A Trace Transformation Technique for Communication Refinement", Proc. Int. Workshop Hardware-Software Codesign, S. 134-139, 2001

[79] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen, A. Haxthausen, "LYCOS: The Lyngby Co-Synthesis System", Design Automation for Embedded Systems, Vol. 2 No. 2, S. 195-235, 1997

[80] "Matlab", Datenblatt, The MathWorks Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA, http://www.mathworks.com/products/matlab/

[81] K. McClogrie, F. Kastenholz, "The Interfaces Group MIB using SMIv2"; RFC2233, November 1997

[82] "MLDesigner", Datenblatt, MLDesign Technologies, Inc., 2230 St. Francis Drive, Palo Alto, CA 94303, USA, http://www.mldesigner.com/mldesigner.html

[83] G. Moore, "Cramming more components onto integrated circuits", Electronics, Vol. 38, No. 8, April 1965

[84] S. Narayan, F. Vahid, D.D. Gajski, "System Specification with the SpecCharts Language", IEEE Design & Test of Computers, Vol. 9, No. 4, S. 6-13, Oktober/ Dezember 1992

[85] K. Nichols et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC2474, Dezember 1998

[86] R. Nieman, P. Marwedel, "An Algorithm for Hardware/Software Partitioning Using Mixed Integer Linear Programming." Design Automation for Embedded Systems, Vol. 2, No. 2, S. 165-193, 1997

[87] Object Management Group, http://www.omg.org/uml/

[88] M. O'Nils, "Communication in Hardware/Software Embedded Systems", Technical Report TRITA-ESD-1997-08, Electronic Systems Design Laboratory, Royal Institute of Technology, Sweden, 1997

[89] N. Pazos, W. Brunnbauer, J. Foag, T. Wild, "System-based Performance Estimation of Multi-processing, Multi-threading SoC Networking Architectures", Forum on Design Languages (FDL'02), Marseille, September 2002

[90] G. Quan, X. Hu, G. Greenwood, "Preference-Driven Hierarchical Hardware/Software Partitioning", Proc. ICCD 1999

[91] A. Radulescu, C. Nicolescu, A.J.C. van Gemund, P.P. Jonker, "CPR: Mixed Task and Data Parallel Scheduling for Distributed Systems", Proc. 15th Int. Parallel and Distributed Processing Symposium 2001, San Francisco, 2001

[92] J. A. Rowson, A. Sangivanni-Vincentelli, "Interface-Based Design", Proc. DAC 1997, S. 178-183, 1997

[93] A. Sangivanni-Vincentelli, M. Sgroi, L. Lavagno, "Formal Models for Communication Based Design", Proceedings of the 11-th International Conference on Concurrency Theory, Concur '00, August 2000

[94] R. Schott, "Quality of Service in IP- und MPLS-Netzen", Telekommunikation Aktuell, Heft 4/5, April/Mai 2002

[95] Semiconductor Industry Association, "International Technology Roadmap for Semconductors, 1999 Edition, Design", Austin, TX, International SEMATECH, 1999

[96] M. Sgroi, L. Lavagno, A. Sangiovanni-Vincentelli, "Formal Model for Embedded System Design", Design & Test of Computers, Vol. 17, No. 2, S. 14-27, April-Juni 2000

[97] M. Sgroi, M. Sheets, A. Mihal, K. Kruetzer, S. Malik, J. Rabaey, A. Sangivanni-Vincentelli, "Addressing the System-on-a-Cip Interconnect Woes Through Communication-Based Design", Proc. DAC 2001, S. 667-672, 2001

[98] F. Slomka, M. Dorfel, R. Munzenberger, R. Hofmann, "Hardware/Software Codesign and Rapid Prototyping for Embedded Systems", IEEE Design & Test of Computers, Vol. 17, No. 2, S. 28-38, April-Juni 2000

[99] Homepage von Sonics Inc.: http://www.sonicsinc.com/

[100] SpecC Homepage: http://www.ics.uci.edu/~specc/

[101] J. Staunstrup, W. Wolf (Hrsg.), "Hardware/Software Co-Design - Principles and Practice", Kluwer, 1997

[102] Superlog Homepage: http://www.superlog.org/

[103] "Behavioral Compiler" Datenblatt, Synopsys Inc., 700 East Middlefield Road, Mountain View, CA 94043, USA, http://www.synopsys.com/products/beh_syn/beh_syn.html

[104] "CoCentric SystemC Compiler", Datenblatt, Synopsys Inc., 700 East Middlefield

Road, Mountain View, CA 94043, USA, http://www.synopsys.com/products/cocentric_systemC/cocentric_systemC_ds.html

[105] "CoCentric System Studio", Datenblatt, Synopsys Inc., 700 East Middlefield Road, Mountain View, CA 94043, USA, http://www.synopsys.com/products/cocentric_studio/cocentric_studio_ds.pdf

[106] "System Architect", Datenblatt, Summit Design Inc., 35 Corporate Drive, Burlington, MA 01803, USA, http://www.sd.com

[107] "SystemVerilog 3.0, Accelleras Extensions to Verilog", Referenz-Manual, Accellera, 1370 Trancas Street, #163, Napa, CA 94558, http://www.accellera.org

[108] SystemC Homepage: http://www.systemc.org/

[109] "Xtensa", Datenblatt, Tensilica Inc., 3255-6 Scott Boulevard, Santa Clara, CA 95054-3013, USA, http://www.tensilica.com/html/products.html

[110] F. Vahid, D.D. Gajski, "Incremental Hardware Estimation during Hardware/Software Functional Partitioning", IEEE Trans. on VLSI Systems, Vol. 3, No. 3, S. 459-464, 1995

[111] F. Vahid, J. Gong, and D. D. Gajski, "A Binary-Constraint Search Algorithm for Minimizing Hardware during Hardware/Software Partitioning", Proc. Euro-DAC ’94, S. 214-219, 1994

[112] F. Vahid, "Techniques for Minimizing and Balancing I/O During Functional Partitioning", IEEE Trans. Computer-Aided Design, Vol. 18, No. 1, S. 69-75, 1999.

[113] K.S. Vallerio, N.K. Jha, "Task graph extraction for embedded system synthsis", Int. Conf. on VLSI Design, Jan. 2003

[114] VSI Alliance, "Virtual Component Interface Standard Version 2 (OCB 2 2.0)" , April 2001.

[115] K. Wakabayashi, H. Tanaka, "Global Scheduling Independent of Control Dependencies Based on Condition Vectors", Proc. 29th DAC, 1992

[116] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, "Scalable High Speed IP Routing Lookups", Proc. ACM SIGCOMM 1997, S. 25-37, 1997

[117] T. Wild, "Entwurf und Test Integrierter Schaltungen", in V. Jung, H.-J. Warnecke (Hrsg), "Handbuch für die Telekommunikation", 2. Auflage, Springer Verlag, 2002, S. 2-24 - 2-29

[118] T. Wild, W. Brunnbauer, J. Foag, N. Pazos, "Mapping and Scheduling for Architecture Exploration of Networking SoCs", Proc. 16th Int. Conf. on VLSI Design, New Delhi, Januar 2003

[119] T. Wild, J. Foag, N. Pazos, W. Brunnbauer, "Integrating On-Chip Communications in HW/SW-Partitioning of Networking Systems-on-Chip", Proc. Int. Workshop on IP based Design, S. 23-28, Grenoble, Dezember 2001

[120] W. H.Wolf, "Hardware-software co-design of embedded systems", Proc. of the IEEE, 82(7), S. 967–989, 1994

[121] W. Wolf, "CAD Techniques for Embedded Systems-on-Silicon", Proc. Int. Conf. on Computer Design 1999, S. 24-31, 1999

[122] M.Y Wu, W. Shu, J.Gu, "Efficient Local Search for DAG Scheduling", IEEE Trans. Parallel and Distributed Systems, Vol. 12, No. 6, S. 617-627, 2001

[123] Y. Xie, W. Wolf, "Allocation and Scheduling of Conditional Task Graph in Hardware/Software Co-Synthesis", Proc. DATE 2001, S. 620, März 2001

[124] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", IEEE Trans. on Communication COM-28, No 4, S. 425-432, April 1980

[125] V.D. Zivkovic, P. Lieverse, "An Overview of Methodologies and Tools in the Field of System-Level Design", in: F. Deprettere, J. Teich, S. Vassiliadis (Eds.), "Embedded Processor Design Challenges, Systems, Architectures, Modeling, and Simulation - SAMOS", S. 74-88, 2002

Anhang

A Ergebnisse der Experimente mit künstlichen Graphen

Nachfolgend sind die Ergebnisse der Experimente mit den synthetisch erzeugten Graphen tabellarisch aufgeführt. Neben den Mittelwerten der mit den einzelnen Verfahren aus jeweils n=60 durchgeführten Experimenten erzielten relativen Schedule Dauern gegenüber FAST werden zusätzlich auch die zugehörigen Standardabweichungen angegeben, um die Streuung der Einzelwerte einschätzen zu können.
Für alle nachfolgenden Mittelwerte m_i der relativen Schedule Dauern $s_{i,k}$ gilt

$$m_i = \frac{1}{n} \cdot \sum_k s_{i,k}, \tag{A.1}$$

Die Standardabweichung der einzelnen Werte berechnet sich folgendermaßen:

$$\sigma_i = \sqrt{\frac{1}{n-1} \cdot \sum_k (s_{i,k} - m_i)^2}, \tag{A.2}$$

In den Nachfolgenden Tabellen sind für die unterschiedlichen Experimente die Werte m_i und σ_i der relativen Schedule Dauer der einzelnen Verfahren gegenüber FAST, d.h. für $i \in \{TS-One, TS-CP, TS-ECP, XieWolf\}$ angegeben.

Alle Angaben erfolgen in %.

Größe der Graphen

Anzahl Knoten	TS-One		TS-CP		TS-ECP		Xie/Wolf	
	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
10	-0,06	0,58	0.52	2,73	0,01	0,66	3,34	4,69
20	-0,14	1,40	2,58	5,26	0,11	1,54	4,97	5,86
50	-1,16	2,07	3,80	5,32	-0,63	2,56	7,06	7,25
100	-4,16	3,43	0,63	4,31	-3,51	3,15	5,00	6,92
150	-4,87	3,16	0,43	3,78	-4,15	3,10	6,99	7,95
200	-5,75	3,55	0,80	4,81	-4,54	3,13	4,22	6,24
300	-6,01	3,35	0,79	5,13	-5,24	3,43	4,57	6,91

Tabelle A-1: Abhängigkeit von der Graphgröße

Anteil bedingter Knoten

Anteil bed. Knoten	TS-One		TS-CP		TS-ECP		Xie/Wolf	
	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
0%	-3,72	2,70	0,93	3,79	-3,36	2,67	3.36	4,95
10%	-4,16	3,43	0,63	4,31	-3,51	3,15	5,00	6,92
20%	-3,99	3,29	1,27	5,25	-3,08	3,70	7,62	8,74
30%	-3,77	3,20	2,06	4,79	-2,92	3,49	8,87	10,12

Tabelle A-2: Abhängigkeit vom Anteil bedingter Knoten

Transferierte Datenmenge pro Kante

mittlere Transfermenge [Bit]	TS-One		TS-CP		TS-ECP		Xie/Wolf	
	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
5	-3,36	3,48	0,27	5,14	-3,48	3,70	3,83	7,65
10	-3,56	3,52	0,12	5,09	-3,51	3,84	3,50	7,40
20	-3,85	3,49	-0,15	4,02	-3,74	3,56	2,68	6,54
50	-4,16	3,43	0,63	4,31	-3,51	3,15	5,00	6,92
100	-6,33	2,74	5,64	3,34	-6,55	2,66	17,86	6,98
200	-10,16	3,21	20,85	9,27	-10,76	3,25	38,79	9,43

Tabelle A-3: Abhängigkeit von der im Mittel transferierten Datenmenge

Anzahl der Ressourcen in der Architektur

Anzahl		TS-One		TS-CP		TS-ECP		Xie/Wolf	
Prozessoren	Beschleuniger	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
1	1	0,00	0,00	0,00	0,01	0,00	0,01	0,99	1,10
	2	0,00	0,00	0,00	0,01	0,00	0,00	1,16	1,37
	3	0,00	0,00	0,04	0,14	0,03	0,12	1,40	1,81
	4	0,00	0,10	0,12	0,34	0,08	0,29	1,67	2,87
	5	-0,01	0,15	0,16	0,32	0,03	0,25	2,77	3,37
	6	-0,27	1,15	0,99	1,48	-0,03	1,02	7,85	7,68
2	1	-1,00	1,21	2,54	3,46	-0,93	1,19	2,82	2,31
	2	-1,44	1,87	2,22	4,40	-1,71	1,94	2,44	2,65
	3	-2,66	2,46	1,80	4,58	-2,22	2,56	3,01	4,15
	4	-3,56	3,51	1,30	4,83	-3,24	2,99	3,11	4,52
	5	-4,16	3,43	0,63	4,31	-3,51	3,15	5,00	6,92
	6	-4,20	3,42	1,89	3,42	-4,03	3,47	12,10	9,26
3	1	-2,56	1,83	0,58	3,88	-2,50	1,95	1,03	2,79
	2	-4,76	2,76	-1,80	2,58	-4,63	2,68	-0,06	3,51
	3	-5,06	2,84	-1,60	3,53	-5,14	3,34	0,44	4,87
	4	-6,45	3,26	-2,12	3,50	-6,29	3,18	2,90	6,46
	5	-6,12	2,74	-0,63	4,20	-6,25	2,71	8,02	6,50
	6	-6,35	2,63	1,94	3,47	-6,60	2,69	12,81	7,48
4	1	-4,74	2,62	-1,65	2,82	-4,99	2,59	-0,88	3,48
	2	-6,63	2,73	-3,45	3,30	-6,75	2,65	-2,95	4,41
	3	-7,09	2,80	-3,50	4,68	-7,11	3,00	0,76	6,74
	4	-6,83	3,31	-1,60	3,82	-6,83	3,28	6,79	6,20
	5	-7,18	2,47	0,77	3,38	-6,84	2,98	10,31	5,46
	6	-8,03	3,26	2,12	3,01	-8,58	2,99	13,56	6,82
5	1	-5,68	2,80	-2,99	2,23	-5,94	2,37	-0,66	9,20
	2	-6,63	2,99	-4,37	2,45	-7,03	2,74	-1,90	5,27
	3	-6,56	3,09	-2,84	3,16	-6,69	2,94	2,47	7,47
	4	-7,44	2,70	-0,35	3,28	-7,25	2,24	8,32	5,56
	5	-7,08	3,26	1,96	2,86	-8,05	2,93	11,29	5,41
	6	-7,99	2,90	2,54	2,85	-9,80	3,15	13,90	6,58

Tabelle A-4: Abhängigkeit von der Anzahl der Ressourcen

Anzahl		TS-One		TS-CP		TS-ECP		Xie/Wolf	
Prozessoren	Beschleuniger	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
6	1	-5,87	2,94	-3,68	2,98	-5,97	2,68	-2,93	5,88
	2	-6,21	3,05	-3,81	2,81	-6,83	2,63	-1,13	5,85
	3	-6,40	3,33	-3,24	3,24	-6,63	2,76	4,39	5,46
	4	-7,40	3,14	0,55	3,14	-7,58	2,63	9,56	5,51
	5	-8,36	3,26	1,76	3,36	-6,37	3,58	11,47	5,54
	6	-8,52	3,26	2,93	2,68	-10,71	3,67	14,59	6,41

Tabelle A-4: Abhängigkeit von der Anzahl der Ressourcen

Mittlere Performance der Ressourcen

mittlere Beschleunigung	TS-One		TS-CP		TS-ECP		Xie/Wolf	
	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
2	-3,30	3,21	1,08	4,65	-3,10	3,07	6,87	8,66
3	-4,03	3,18	0,46	4,20	-3,87	3,14	5,62	6,34
4	-3,86	3,30	0,93	4,56	-3,32	3,12	5,05	6,53
5	-3,84	2,84	1,14	4,48	-3,30	2,79	4,99	6,67
6	-3,97	3,43	0,92	4,36	-3,49	3,26	4,63	7,02
8	-4,00	3,06	0,67	4,57	-3,64	3,18	4,70	6,77
10	-3,79	2,84	1,08	4,60	-3,83	3,12	5,51	6,97

Tabelle A-5: Abhängigkeit von der mittleren Beschleunigung

Anteil von beschleunigbaren Prozessen

Anteil beschleunigbarer Prozesse	TS-One		TS-CP		TS-ECP		Xie/Wolf	
	m_i	σ_i	m_i	σ_i	m_i	σ_i	m_i	σ_i
5%	-0,83	1,00	1,10	1,38	-0,87	1,17	1,51	1,67
10%	-1,31	1,80	0,74	1,30	-1,41	1,65	1,06	2,10
15%	-2,48	2,34	-0,80	2,36	-2,44	2,13	0,51	2,82
20%	-3,49	2,94	-0,74	2,68	-3,05	2,64	0,85	4,62
25%	-3,60	3,14	-0,26	3,34	-3,34	3,16	4,05	7,59
30%	-5,34	2,86	0,35	3,61	-5,23	2,96	9,11	7,24
40%	-9,63	3,26	2,89	3,36	-9,57	3,22	14,73	5,95
50%	-14,16	4,29	6,31	4,24	-15,02	4,51	17,59	7,34

Tabelle A-6: Abhängigkeit vom Anteil beschleunigbarer Prozesse