Opto-Elektronische Prozessor-Speicher Koppelung in Multiprozessoren

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Informatik

der Universität Karlsruhe (Technische Hochschule)

angenommene

D i s s e r t a t i o n

von

Pawel Lukowicz

Tag der mündlichen Prüfung:	29. Juni 1999
Erster Gutachter:	Prof. Dr. Walter F. Tichy
Zweiter Gutachter:	Prof. Dr. Jürgen Jahns

Danksagung

An dieser Stelle möchte ich vor allem meiner Frau Astrid für die geduldige und liebevolle Unterstützung während der schwierigen Zeit danken. Besonderes wertvoll war ihre engagierte Mithilfe bei der Korrektur der Ausarbeitung, die mir einige zeitaufwendige und von mir verhaßte Korrekturiterationen erspart hat.

Eine große Hilfe war auch mein langjähriger Freund, Zimmerkollege und Promotions-Leidensgenosse Ernst A. Heinz. Unsere kontroversen Diskussion während der langen Promotionstage (und Nächte) haben den Anstoß für viele interessanten Ideen gegeben und manch einen 'Knick in der Logik' entlarvt.

Besonderes herzlich möchte ich bei meinen beiden Referenten Prof. Walter F. Tichy und Prof. Jürgen Jahns für die tolle Zusammenarbeit bedanken. Walter hat an seinem Lehrstuhl eine Arbeitsatmosphere geschaffen, die zur Beschäftigung mit neuen interessanten Ideen animiert und einem den Spaß an der Forschung vermittelt. Dies liegt vor allem an der Freiheit bei der Auswahl von Forschungsthemen, seiner Unterstützung bei den exotischten Vorhaben, und seiner jeder Zeit offenen Tür.

Zu der Entstehung der Arbeit haben neben meiner oben genannten Referenten viele Leute beigetragen, den ich an dieser Stelle für Ihre Hilfe danken möchte. Dazu gehören die (ex) Studenten Thomas Joachim, Thorsten, Bernd Köcke, Mathias Hahn und Oliver Schiff die als HiWis, Diplmanden und Studienarbeiter an der Arbeit beteiligt waren. Hinzu kommen alle meine Kollegen am Lehrstuhl, insbesondere Thomas Warschko, Joachim Blum, Lutz Prechelt und Michael Philippsen, die mir mit (nicht nur) fachlichen Ratschlägen zur Seite standen und stets für ein angenehmes Arbeitsklima gesorgt haben. Last but not least ist hier unsere Sekretärin Frau Ruth Ghafari zu nennen, die viel zu oft mein organisatorisches Chaos 'ausbaden' mußte.

Abschließend möchte ich noch meinen Eltern danken, die mir mein Studium ermöglicht haben.

Karlsruhe Juni 1999

Die vorliegende Arbeit wurde im Rahmen eines von der Deutschen Forschungsgemeinschaft geförderten Projektes durchgeführt.

Inhaltsverzeichnis

Kapitel 1

Überblick

Die vorliegende Arbeit beschäftigt sich mit der Frage, wie neue technologische Entwicklungen im Bereich der optischen Nachrichtenübertragung zur Verbesserung der Prozessor-Speicher Koppelung in Multiprozessoren genutzt werden können. Sie zeigt, daß daß die Argumente für die Beschränkung des sog. SMP Speichermodells auf Rechner mit wenigen Prozessoren und gegen seine Anwendung bei der Vernetztung von Arbeitsplatzrechnern nicht für opto-elektronische Prozessor-Speicher Koppelung gelten. Sie stellt eine Grundlagenuntersuchung dar, die den Weg für eine praktische Nutzung der neuen Technologie in der parallelen Rechnerarchitektur ebnet. Die Erkenntnisse der Arbeit haben zur Einrichtung eines DFG-Forschungsprojekt geführt, das die Implementierung von Rechnern mit opto-elektronischer Prozessor-Speicher Koppelung zum Ziel hat.

Im Nachfolgenden wird zunächst die Motivation für die Arbeit erläutert. Daraufhin werden die Ziele dargelegt und die Beiträge zusammengefaßt. Abschließend wir der Aufbau der Ausarbeitung beschrieben.

1.1 Motivation

Die Arbeit ist durch die Unzulänglichkeiten der Speicherarchitektur heutiger Parallelrechner und die Fortschritte der opto-elektronischen Netzwerktechnologie motiviert.

1.1.1 Symmetrische Multiprozessoren

Die zwei größten Probleme der Parallelverarbeitung sind die mangelhafte Programmierbarkeit und die Diskrepanz zwischen der theoretisch möglichen und der in der Praxis erreichbaren Leistung der heutigen Rechner. Die meisten Parallelrechner erreichen nur für wenige Anwendungen eine gute Effizienz und erfordern eine komplexe, hardwarenahe Programmierung. Eine Schlüsselrolle spielt dabei die Kommunikation zwischen den Prozessoren. Zugriffe auf Daten, die von mehreren Prozessoren gleichzeitig genutzt werden, sind in der Regel wesentlich langsamer als normale Speicherzugriffe und erfordern seitens des Programmierers eine besondere Behandlung. Eine Ausnahme bilden in dieser Hinsicht die sog. symmetrischen Multiprozessoren (SMPs). Sie zeichnen sich durch eine für alle Prozessoren einheitliche (symmetrische) Sicht des Speichers aus. Dabei werden für alle Speicherzugriffe eine einheitliche Speicherzugriffszeit und die volle Cache-Kohärenz garantiert. Die Kommunikation erfolgt über den Speicher. Es wird also nicht zwischen Zugriffen auf gemeinsam und privat genutzte Daten unterschieden. Dies läßt ein einfaches Programmiermodell und eine hohe Effizienz für ein breites Spektrum von Anwendungen zu.

Leider stellt die SMP-Architektur sehr hohe Anforderungen an die Leistung des Verbindungsnetzwerks und der Netzwerkschnittstellen. Das Hauptproblem besteht darin, daß für eine effiziente Realisierung der Cache-Kohärenz ein leistungsfähiger Rundruf-Mechanismus benötigt wird. Dabei steigt die benötigte Rundruf-Bandbreite linear mit der Anzahl der Prozessoren. Ein solcher Rundruf-Mechanismus kann mit der heutigen Technologie nur für wenige Prozessoren und räumlich kleine Systeme implementiert werden. Dies hat zwei Gründe. Zum einen wird es auf Grund fundamentaler Eigenschaften elektrischer Leitungen mit steigendem Fanout und Leitungslänge immer schwieriger, eine hohe Bandbreite bei geringer Latenz zu erreichen. Zum anderen setzt ein Rundruf voraus, daß alle Prozessoren jede Nachricht empfangen und untersuchen. Dadurch werden die Netzwerkschnittstellen der Prozessoren mit zunehmender Bandbreite des Rundrufkanals überfordert. Dabei spielen folgende Faktoren eine

entscheidende Rolle: die geringe Ein/Ausgabe Bandbreite von VLSI-Bausteinen (das Pinout-Problem) und die beschränkte Geschwindigkeit der Cache-Operationen (das Cache-Update Preassure Problem):

Pinout Problem: Die Rechnerkapazität heutiger VLSI-Bausteine ist mehr als ausreichend, um auch bei massiv parallelen SMPs die Anforderungen an die Netzwerkschnittstellen zu erfüllen. Leider liegt ihre Ein/Ausgabe-Bandbreite um mehrere Größenordnungen unter der Rechenkapazität. Damit stellt der Datenfluß von der Platine zum Chip einen Flaschenhals dar, der die Leistung konventioneller Netzwerkschnittstellen bestimmt. Dieser Flaschenhals resultiert aus den schlechten elektrischen Signalübertragungseigenschaften der Pins und Bondingdrähte, über die die VLSI-Chips an die Platine angeschlossen sind. Hinzu kommt, daß die Anschlüsse viel Chipfläche benötigen. Ihre Anzahl ist heute auf 500 bis 1000 pro Chip beschränkt.

Cache-Update-Preassure Problem: Um die Cache-Kohärenz zu gewährleisten muß jeder Prozessor die über den Brodcast-Kanal verschickten Daten mit seinem Cache-Inhalt vergleichen und bei Bedarf den Cache-Inhalt modifizieren. Somit ist die Bandbreite des Rundrufkanals durch die Zugriffsgeschwindigkeit des Cache beschränkt.

Aus obigen Überlegungen folgt, daß das SMP-Konzept nicht zur Koppelung von Arbeitsplatzrechnern verwendet werden kann. Außerdem sind zur Zeit effiziente SMPs auf Rechnern mit 2 bis 16 Prozessoren beschränkt. Bei größeren Prozessorzahlen ist es nicht mehr möglich, eine mit sequentiellen Rechnern vergleichbare Speicherlatenz zu garantieren. Der bisher größte SMP ist der Ultra Enterprise 10000 Server von SUN mit 64 Prozessoren. Dabei kann die große Prozessorzahl allerdings nur auf Kosten einer Hauptspeicherlatenz von bis zu 600ns (verglichen mit ca. 150Ns bei sequentiellen Rechnern) und eines sehr großen Implementierungsaufwandes erreicht werden.

1.1.2 Fortschritte in der Optischen Nachrichtentechnik

Es ist seit geraumer Zeit bekannt, daß die optische und opto-elektronische Nachrichtenübertragung der elektronischen überlegen ist. Trotzdem spielt sie bislang im Bereich der Rechnerarchitektur noch keine Rolle. Der Grund hierfür liegt darin, daß komplexe, leistungsfähige optische Systeme bisher nicht robust und kompakt genug für die praktische Anwendung gebaut werden konnten. Lediglich sequentielle Lichtleiterfaser-Systeme mit einer Datenrate von bis zu 2.4 Gbit/s konnte bisher in der Praxis realisiert werden. In der Telekomunikation und im Bereich der LAN-Netzwerke sind solche Systeme elektrischen Alternativen überlegen, da elektrische Datenübertragung mit einer so hohen Frequenz nur über wenige Meter möglich ist. Für die Anwendung in der Rechnerarchitektur wäre dagegen eine um einen Faktor 10 bis 100 höhere Bandbreite notwendig. Sie kann mit einfachen Fasersystemen aus zwei Gründen nicht realisiert werden. Einerseits ist es nicht praktikabel, solche Datenraten sequentiell durch eine Leitung zu schicken. Dies würde die elektronischen Treiber überfordern. Andererseits war es bisher nicht möglich, viele einzelne Fasern zu einem parallelen Übertragungssystem zu kombinieren. Dazu waren bisher die Sender und Treiber zu aufwendig. Ein weiteres Problem sequentieller optischer Systeme ist ihre hohe Latenz. Sie resultiert aus der Notwendigkeit, Daten in einem speziellen Format zu senden, um das Fehlen von Kontrol- und Taktleitungen zu kompensieren.

Die Arbeit ist durch den rasanten Fortschritt motiviert, der zur Zeit im Bereich optischer und opto-elektronischer Systemtechnik stattfindet. Er ist mit der Entwicklung in den Anfangsjahren der Computertechnik vergleichbar. Opto-elektronische Systeme, die bisher die Fläche eines ganzen Labortisches füllten, können nun in Form weniger Zentimeter oder gar Millimeter großer integrierter Module hergestellt werden. Solche Module können Tausende optischer Ein/Ausgabekanäle und komplexe Netzwerktopologien beinhalten. Vom Standpunkt der Rechnerarchitektur gesehen besonders bemerkenswert sind zwei Entwicklungen: die Fortschritte bei der Integration optischer Ein/Ausgabekanäle auf konventionellen elektronischen Bausteinen, sowie neuartige massiv parallele Übertragungssysteme. Sie ermöglichen massiv parallele optische Verbindungen, die direkt zwischen VLSI -Bausteinen verlaufen.

Wie in 4.16 genau erläutert ([86]) zeichnen sich solche Verbindungen dadurch aus, daß:

- die Bandbreite der einzelnen Kanäle im Bereich von 2 bis 4 Bit pro Prozessortaktzyklus liegt und mit der Leistung der VLSI-Technologie skaliert,
- die optischen Anschlüsse nur wenig Chipfläche benötigen, so daß bis zu 64000 auf einem Chip untergebracht werden können, ohne die Leistungsfähigkeit der Logik zu beeinträchtigen,
- die Latenz der Kanäle im Bereich einiger weniger Gatterschaltzeiten plus der Signallaufzeit liegt,
- die Daten aller Kanäle parallel von der Logik eines VLSI-Bausteins verarbeitet werden und
- die obigen Leistungsparameter nicht durch die Leitungslänge und nur unwesentlich durch den Fanout beeinflußt werden.

Massiv parallele optische Verbindungen zwischen VLSI-Bausteinen sind damit in der Lage, Rundruf-Kanäle mit einer hohen Bandbreite, hohen Fanout und einer großen räumlichen Ausdehnung zu realisieren. Dies erlaubt die Koppelung von Arbeitsplatzrechnern mit Hilfe des SMP-Speichermodells. Gleichzeitig wird das Pinout-Problem durch die hohe Bandbreite beseitigt, die die direkt auf den VLSI-Bausteinen integrierten optischen Ein/Ausgabe Kanäle realisieren. Wie später in der Arbeit gezeigt wird, erlaubt die hohe Anzahl dieser Ein/Ausgabe Kanäle zusammen mit einem geeigneten Speicherzugriffsprotokoll eine Parallelisierung der Cache-Zugriffe. Auf diese Weise kann das Cache-Update-Preassure Problem gelöst werden. Damit werden die Voraussetzung für die Realisierung von SMPs mit hoher Prozessorzahl geschaffen.

1.2 Ziele

Die Arbeit stellt eine Grundlagenuntersuchung dar. Sie hat sich zum Ziel gesetzt, den Weg für die Implementierung opto-elektronischer SMPs zu ebnen. Dabei soll zum einen gezeigt werden, daß die zu erwartenden Vorteile solcher Architekturen den hohen Aufwand für Ihre Entwicklung rechtfertigen. Zum andern sollen die vielversprechendsten Implementierungsansätze aufgezeigt werden. Hierbei sollen sowohl Fragen der Technologie als auch die der Rechnerarchitektur berücksichtigt werden. Im einzelnen sollen in der Arbeit folgende Fragen beantwortet werden:

1. Welche Technologie und welche Komponenten erscheinen für die Anwendung in der Prozessor-Speicher Koppelung am vielversprechendsten ?
2. Bei welchen Komponenten ist noch im Hinblick auf eine solche Anwendung weitere Forschung und Entwicklung besonders notwendig ?
3. Wie können die Vorteile der neuen Technologie in einer SMP-Architektur am besten genutzt werden ?
4. Welche Leistung ist von den Architekturen unter realistischen Annahmen bezüglich der Leistungsfähigkeit der opto-elektronischen Komponenten zu erwarten ?

1.3 Ansätze

Wie oben dargelegt, scheitert die elektronische Implementierung von SMPs mit großer räumlicher Ausdehnung und großer Prozessorzahl an zwei Dingen: der Schwierigkeit leistungsfähige Rundrufkanäle mit einem hohen Fanout und großer Leitungslänge zu realisieren, und der unzureichenden Leistung elektronischer Netzwerkschnittstellen. In der Arbeit wird gezeigt, wie die obigen Probleme durch den Einsatz von VLSI-Bausteinen mit integrierten optischen Ein/Ausgabefenstern und massiv parallelen optischen Verbindungen gelöst werden können. Die Grundidee ist in Abbildung 1.2 dargestellt. Sie kann wie folgt zusammengefaßt werden:

- Das elektronische Verbindungsnetzwerk wird auf einen VLSI-Chip 'geschrumpft', der mit optischen Ein/Ausgabefenstern versehen ist. Durch die geringe Leitungslänge und günstige elektrische Leitungseigenschaften innerhalb des Chips kann so eine hohe Bandbreite bei niedriger Latenz und hohem Fanout erreicht werden.

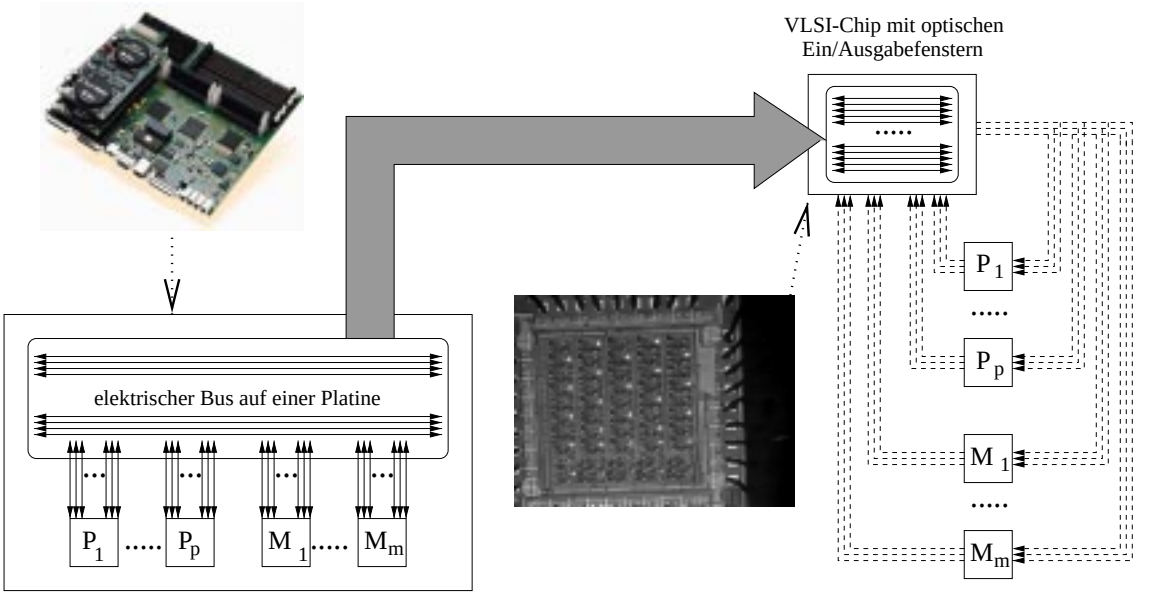


Abbildung 1.1: Die Grundidee der in der Arbeit betrachteten Systeme am Beispiel eines Bus-basierten Rechners.

- Der auf dem Chip befindliche Bus wird mit Hilfe einer unidirektionalen optischen Leitung (B-Kanal genannt) verlängert. Über diese Leitung kann der Chip Daten an alle Prozessoren und Speicherbänke einen Rundruf schicken. Da der Rundruf optisch stattfindet, kann weitgehend unabhängig von der Anzahl der Prozessoren und der Ausdehnung des Systems auch hier eine hohe Bandbreite und geringe Latenz garantiert werden.

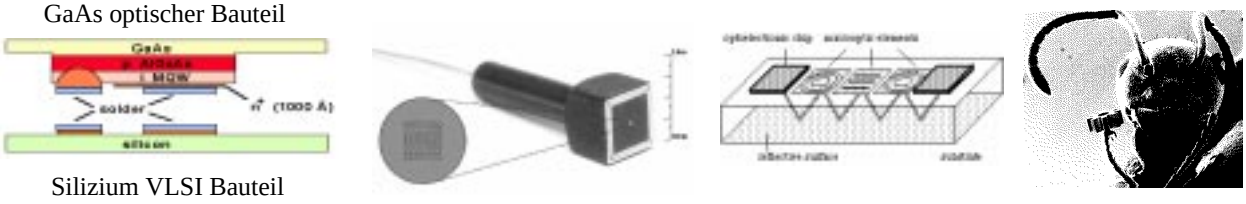
- Die Prozessoren und Speicherbänke werden über optische Punkt-zu-Punkt Kanäle (P- und M-Kanäle genannt) an die optischen Ein/Ausgabefenster des Chips angeschlossen. Alle Nachrichten, die sie an das System verschicken wollen, gehen zuerst über diese Kanäle an den Chip. Dieser speichert sie zwischen, führt im Fall von Konflikten eine Arbitrierung durch und leitet sie dann weiter. Dank der Vorteile der optischen Datenübertragung und der direkten Anbindung an den VLSI-Chip haben die P-und M-Kanäle unabhängig von der Systemgröße eine geringe Latenz.

- Die Prozessoren werden mit einer opto-elektronischen Netzwerkschnittstelle versehen um das Pinout-Problem zu lösen.

An dieser Stelle mag es zunächst unverständlich erscheinen, warum eine opto-elektronische anstelle einer rein optischen Architektur verwendet wird. Hierfür gibt es zwei Gründe:

1. Für einen Rundruf wird ein Transmitter mit hoher Bandbreite und einer zur Prozessorzahl proportionalen Ausgangsleistung benötigt. Für hohe Prozessorzahlen sind solche Transmitter aufwendig und teuer. Da in einem rein optischen Bussystem jeder Prozessor und jede Speicherbank auf dem Rundrufkanal senden können muß, steigt die Anzahl solche Sender im System linear mit der Anzahl der Prozessoren. In dem hier vorgeschlagenen System wird der Rundruf hingegen nur von dem opto-elektronischen Chip durchgeführt. Die Prozessoren und Speicherbänke schicken ihre Daten über Punkt-zu-Punkt Kanäle, so daß sie mit verhältnismäßig einfachen Sendern auskommen. Somit wird unabhängig von der Anzahl der Prozessoren nur ein solcher Transmitter benötigt.

2. Auf dem VLSI-Chip befindet sich zu jedem Zeitpunkt die Information über alle im System vorhandenen Anfragen. Dies ermöglicht eine effiziente und intelligente Arbitrierung. Außerdem können auf dem Chip verschiedene Optimierungen (z.B. eine effiziente Synchronisation) durchgeführt werden.



a) b) c) d)

Abbildung 1.2: Die in der Arbeit ausgewählten Technologien. Abbildung a) stellt das Flip-Chip Bonden eines GaAs optischen Bauteils auf einem Silizium-VLSI-Schaltkreis dar. In Abbildung b) ist ein Stecker für ein paralleles Faser-Übertragungssystem zu sehen. Das Prinzip der planar integrierten Freiraumoptik ist in Bild c) dargestellt. Die Möglichkeiten der LIGA-Technologie verdeutlicht das in Bild d) gezeigte mikroskopische Zahnrad.

1.3.1 Technologie

Für die Implementierung des oben beschriebenen Architekturkonzeptes werden drei Dinge benötigt:

1. eine Möglichkeit, komplexe VLSI-Schaltkreise mit vielen optischen Ein/Ausgabefenster zu kombinieren,
2. ein optisches System, daß sehr viele hochfrequente Kanäle flexibel und robust über Entfernungen bis zu 10m überträgt,
3. eine Möglichkeit, das optische System robust und kompakt an die optischen Ein/Ausgabefenster des VLSI-Chips anzubinden.

Für die erste Aufgabe wird das Flip-Chip Bonden von optischen Bauelementen auf konventionellen CMOS VLSI-Chips (siehe z.B. [16]) vorgeschlagen. Diese Technologie erlaubt die billige und einfache Integration einer breiten Palette von optischen Sendern und Empfängern. Dabei garantiert sie eine geringe Latenz und schränkt die Funktionalität der CMOS-Schaltkreise in keiner Weise ein. Für die Datenübertragung wurden massiv parallele Lichtleiter-Fasersysteme ausgewählt. Solche Systeme arbeiten mit ein- und zweidimensionalen Bündeln von 8 bis maximal 512 Fasern, wobei in jeder Faser bis zu 32 Signale mit Hilfe unterschiedlicher Wellenlängen übertragen werden können ([85]). Sie bieten die benötigte hohe Anzahl von Kanälen und sind gleichzeitig robust und ähnlich einfach in der Handhabung wie konventionelle elektrische Kabel. Hinzu kommt, daß sie mit einer Vielzahl an kommerziell erhältlichen Faserkomponenten kompatibel sind. Für die Anbindung der Faser-Übertragungsysteme an die optischen Fenster des VLSI-Chips wird eine Kombination aus planar integrierter Freiraumoptik ([75]) und LIGA-Mikromechanik ([35]) vorgeschlagen. Die planare Freiraumoptik integriert komplexe optische Systeme im Inneren einer kleinen Glasplatte, auf deren Ober- und Unterseite die benötigten optischen Bauelemente eingeätzt werden. Sie erlaubt komplexe Abbildungen der Faserkanäle auf die Ein/Ausgabefenster der VLSI-Chips in einem stabilen, kompakten Aufbau. Für die Befestigung der planaren Optik an den Fasern und dem Chip werden mikromechanische Komponenten benötigt, die am besten mit dem LIGA-Verfahren realisiert werden können. Diese Technologie ermöglicht die Herstellung komplizierter Strukturen mit einer Genauigkeit von weniger als $1\mu m$. Da sie auf Lithographie und plastischer Abformung basiert, ist sie gleichzeitig für die billige Massenproduktion gut geeignet.

1.3.2 Architekturen

Basierend auf der oben beschriebenen Grundidee, werden in der Arbeit drei Architekturvarianten untersucht: ein einfacher opto-elektronischer Bus, eine opto-elektronisches Bus/Crossbar Kombination, und ein Netzwerk aus mehreren opto-elektronischen Bussen. Sie werden nachfolgend als PHOTOBUS, PHOTOBAR, und PHOTON bezeichnet. Die drei Architekturen sind so konzipiert, daß sie aufeinander aufbauen und dabei immer stärker die Vorteile optischer und opto-elektronischer Netzwerkkomponenten nutzen. Sie geben so Meilensteine für eine stufenweise Einführung der neuen Technologie in die Rechnerarchiktur vor.

PHOTOBUS

Auf dem opto-elektronischen VLSI-Chip wird ein konventioneller paketorientierter Bus realisiert. Wenn ein Prozessor oder eine Speicherbank eine Nachricht auf dem Bus übertragen möchte, dann muß sie diese zunächst über

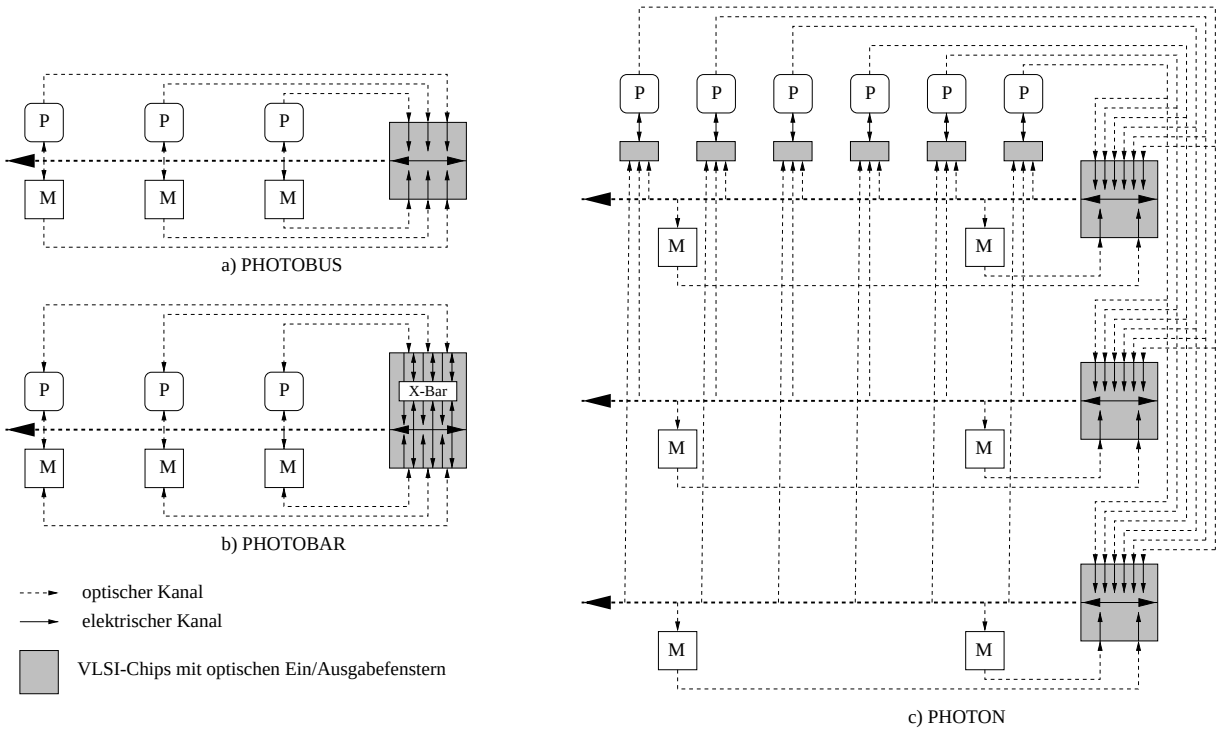


Abbildung 1.3: Die Abbildung zeigt die drei in der Arbeit untersuchten Architekturen: die PHOTOBUS-Architektur (a), die PHOTOBAR-Architektur (b) und die PHOTON-Architektur (c)

ihren P- bzw. M-Kanal an den Bus-Chip schicken. Der Chip speichert die Nachricht zwischen und schickt sie so bald wie möglich über den B-Kanal an alle Systemkomponenten.

Die obige Architektur hat gegenüber konventionellen elektronischen SMPs vor allem den Vorteil, daß sie sich für die Implementierung eines SMP Speichermodells bei Arbeitsplatzrechner-Clustern eignet. Da der Rundruf optisch stattfindet, spielt es für die Bandbreite und Latenz keine Rolle, ob sich die Prozessoren auf einer Platine oder in verschiedenen in einem Raum verteilten Rechnern befinden. Hinzu kommt, daß der optische Rundrufkanal eine höhere Bandbreite als ein elektrischer erreichen kann und somit mehr Prozessoren zuläßt.

Die PHOTOBUS-Architektur basiert weitgehend auf kommerziell bzw. als Laborprototypen erhältlichen Komponenten. Lediglich die Anbindung der Faser an den opto-elektronischen Bausteine muß in Form einer Sonderanfertigung realisiert werden. Die PHOTOBUS-Architektur damit die Möglichkeit, verhältnismäßig schnell einen Prototyp zu bauen, der die Eignung der Opto-Elektronik für den Einsatz in der Praxis unter Beweis stellt.

PHOTOBAR

Diese Architektur ist durch die Tatsache motiviert, daß nur ein Teil der Nachrichten in einem SMP eines Rundrufs bedarf. Um dies zu nutzen, sind die optischen Kanäle von den Prozessoren und Speicherbänken zum opto-elektronischen Chip bidirektional. Außerdem besitzt der Chip neben einem Bus auch noch ein Crossbar-Netzwerk. Nachrichten, die keines Rundrufs bedürfen, werden über das Crossbar und die bidirektionalen Kanäle geroutet. Dadurch werden der Rundrufkanal entlastet und der Systemdurchsatz gesteigert.

Die PHOTOBAR-Architektur ist ähnlich wie die PHOTOBUS-Architektur für die Koppelung von Arbeitsplatzrechnern geeignet. Gleichzeitig ist sie Dank der Entlastung des B-Kanals wesentlich besser skalierbar. Wie in Kapitel 8 gezeigt wird, kann sie bis zu 128 Prozessoren unterstützen. So kann das SMP-Konzept den Bereich großer Server und kleiner Supercomputer erweitern. Der Preis hierfür ist die Abkehr von sofort kommerziell bzw. als Prototypen erhältlichen Komponenten.

PHOTON

Für SMPs mit Hunderten von Prozessoren wird eine Architektur vorgeschlagen, die aus mehreren opto-elektronischen Bus-Chips besteht. Jeder Bus-Chip ist für die Zugriffe auf eine Untermenge von Speicherbänken zuständig. Er

kann über optischen Eingabekanäle Daten von dieser Untermenge von Speicherbänken sowie allen Prozessoren gleichzeitig empfangen und speichern. Außerdem kann er über einen optischen Rundrufkanal Daten an diese Speicherbänke und die Prozessoren verschicken. Damit die Prozessoren Daten an alle Bus-Chips schicken können, sind ihre optischen Ausgabekanäle als Rundrufkanäle ausgelegt. Außerdem besitzt jeder Prozessor einen opto-elektronischen VLSI Chip als Schnittstelle zu den Rundrufkanälen der Bus-Chips. Dadurch wird sichergestellt, daß die hohe Bandbreite der vielen Rundrufkanäle von den Prozessoren verarbeitet werden können.

Die PHOTON-Architektur erweitert das Einsatzgebiet der SMPs in den Bereich großer, massiv paralleler Supercomputer. Um dies zu erreichen, setzt sie auf neue, zum Teil noch im Entwicklungsstadium befindliche Bauteile. Sie stellt damit die angestrebte Endstufe der Entwicklung opto-elektronischer Multiprozessoren.

1.4 Beiträge

Das wichtigste Ergebnis der Arbeit besteht in der folgenden Feststellung:

Die Argumente für die Beschränkung des SMP Speichermodells auf rechner mit wenigen Prozessoren und gegen seine Anwendung bei der Vernetzung von Arbeitsplatzrechnern gelten nicht für Rechner mit opto-elektronische Prozessor-Speicher Koppelung.

Es wird gezeigt, daß unter realistischen Annahmen bezüglich der Leistungsfähigkeit der Opto-Elektronik, das SMPs-Speichermodell sowohl im Bereich der Arbeitsplatzrechner-Cluster als auch in Superrechnern mit Hunderten von Prozessoren anwendbar ist. Damit wird eine Annahme in Frage gestellt, die die Architektur der Parallelrechner in den letzten Jahren geprägt hat.

Die einzelnen Beiträge der Arbeit können in drei Bereiche eingeteilt werden: Technologiestudie, Architekturentwurf und Leistungsbewertung.

1.4.1 Technologie

In der Arbeit wird ein Konzept für die Realisierung opto-elektronischer Verbindungen entwickelt, die auf die Anwendung in der Prozessor-Speicher Koppelung in SMPs optimiert sind. Die Beiträge können wie folgt zusammengefaßt werden:

1. Unter der Vielzahl neuer optischer und opto-elektronischer Komponenten werden diejenigen identifiziert, die für den Einsatz in der Prozessor-Speicher Koppelung am besten geeignet sind. Dabei werden vor allem zwei Aspekte beachtet: die Reife der Technologien und ihre gegenseitige Kompatibilität.

2. Auf der Basis der in der Literatur veröffentlichten Daten wird abgeschätzt, welche Leistung von einem auf obigen Komponenten basierenden Netzwerk zu erwarten ist.

3. In Zusammenarbeit mit Wissenschaftlern aus den Gebieten der Optik, Opto-Elektronik und Mikrosystemtechnik wird ein Konzept für ein modulares opto-elektronisches Baukastensystem entwickelt.

4. Die Realisierbarkeit des obigen Systems wird durch die Implementierung von einfachen Prototypen für einige kritische Teile des Systems untermauert. Diese Prototypen wurden am Institut für Mikrotechnik in Mainz und bei den Bell-Labs in USA nach den im Rahmen der Arbeit erarbeiteten Entwürfen gefertigt.

1.4.2 Architekturentwurf

Der Beitrag der Arbeit im Bereich der Rechnerarchitektur besteht im Entwurf der drei im vorigen Abschnitt beschriebenen opto-elektronischen SMP-Speicherarchitekturen. Dabei wird für wie folgt vorgegangen:

1. Für jede Architektur wird gezeigt, welche Bedingungen erfüllt werden müssen, um eine korrekte Ausführung des Berkeley-Cache-Kohärenz Prototkolls zu garantieren.

2. Es wird der Aufbau der Komponenten beschrieben, die zur Implementierung obiger Protokolle notwendig sind. Die Funktion der Komponenten wird mit Hilfe endlicher Automaten und Blockdiagrammen spezifiziert.

3. Es wird am Beispiel der Synchronisationsoperationen LOCK und UNLOCK gezeigt, wie komplexe Operationen auf dem Bus bzw. Schalterchip implementiert werden können.

4. Es wird am Beispiel der PHOTON-Architektur gezeigt, wie das Problem der cache-Update-Pressure mit Hilfe der opto-elektronischen VLSI-Bausteine und einer geeigneten Netzwerktopologie gelöst werden kann.

1.4.3 Leistungsbewertung

Für die einzelnen Architekturen wird ermittelt, wie ihre Effizienz von den Technologieparametern abhängt. Dabei wird gezeigt, daß unter realistischen Annahmen bezüglich der Technologieparameter die Realisierung effizienter SMPs mit Hunderten von Prozessoren möglich ist. Auch die Machbarkeit von Arbeitsplatzrechner-Clustern mit einem SMP-basierenden Speichermodell wird untermauert. Im einzelnen werden im Bereich der Leistungsbewertung folgende Beiträge geleistet:

1. Es wird ein einfaches schlangentheoretisches Modell der entworfenen Architekturen erstellt.

2. Es wird eine Simulationsumgebung geschaffen, in der die verschiedenen Architekturen unter verschiedenen Annahmen über die Leistungsfähigkeit der Technologie bewertet werden können.

3. Die Leistung der entworfenen Architekturen werden an Hand ausgewählter SPLASH-2 Benchmarks in dem Simulator untersucht. Durch einen Vergleich der Ergebnisse mit der Theorie kann die Korrektheit der Simulation untermauert werden.

1.5 Aufbau der Ausarbeitung

Die Ausarbeitung besteht aus zwei Teilen. Der erste Teil beschäftigt sich mit den Grundlagen und den Hintergründen. Im zweiten Teil werden die Ergebnisse der Arbeit beschrieben.

Grundlagen

Da die Arbeit einen interdisziplinären Charachter hat, fällt dieser Teil recht ausführlich aus. Es sorgt dafür, daß sowohl der aus der Informatik kommender Leser die mit den Grundlagen der Opto-Elektronik vertraut gemacht wird, als auch der in der Opto-Elektronik beheimatete Laser in die Problematik paralleler Rechnerarchitektur eingeführt wird.

Kapitel 2: Hier werden die wichtigsten Aspekte paralleler Rechnerarchitektur zusammengefaßt. In diesem Zusammenhang wird begründet, warum die symmetrischen Multiprozessoren eine besonderes attraktive Architekturvariante darstellen.

Kapitel 3: In diesem Kapitel wird die Speicherarchitektur von SMPs erläutert. Dabei wird gezeigt, wo die Probleme bei der Implementierung großer leistungsfähiger SMPs liegen. Abschließend werden Forschungsarbeiten angesprochen, die sich mit Speichersystemen von massiv parallelen Rechnern und Arbeitzplatzrechner-Clustern beschäftigen.

Kapitel 4: Kapitel 4 beschäftigt sich mit der opto-elektronischen Netzwerktechnik und ihrer Anwendung in der Rechnerarchitektur. Es faßt die fundamentalen Vorteile der Optik zusammen und gibt einen Überblick über den Stand der Technik. Dabei wird deutlich, warum neue technologische Entwicklungen eine Anwendung in SMPs praktikabel erscheinen lassen. Abschließend werden verwandte Arbeiten diskutiert, die sich mit opto-elektronischen Parallelrechnern beschäftigen.

Ergebnisse

Kapitel 5: Dieses Kapitel stellt die in der Arbeit verfolgten Ansätze und die gewählte Vorgehensweise vor. Dabei werden unter anderem die Grundidee der theoretischen Modellierung und der Aufbau der Simulationsumgebung beschrieben.

Kapitel 6: Das Kapitel behandelt die Ergebnisse der Technologiestudie. Es beschreibt die gewählten Technologien und ermittelt ihre Leistungsparameter. Darüberhinaus werden das Konzept des opto-elektronischen Baukastensystems und die implementierten Komponenten-Prototypen vorgestellt.

Kapitel 7: Kapitel 7 beschäftigt sich mit der PHOTOBUS-Architektur. Als erstes werden die Bedingungen erläutert, die zur korrekten Realisierung des Berkley-Protokolls berücksichtigt werden müssen. Daraufhin wird der Aufbau eines Systems beschrieben, das diese Bedingungen erfüllt. Basierend auf dieser Beschreibung werden die Leistungsparameter des Systems ermittelt. Diese Parameter werden als Grundlage der nachfolgend beschriebenen theoretischen Modellierung und Simulation benutzt. Abschließend wird die Realisierung der LOCK und UNLOCK Operationen auf dem Bus-Chip beschrieben.

Kapitel 8: Kapitel 8 stellt die PHOTOBAR-Architektur vor. Die Vorgehensweise ist dabei mit der von Kapitel 7 identisch.

Kapitel 9: Diese Kapitel ist massiv parallelen SMP Architekturen auf der Basis des PHOTON-Konzeptes gewidmet. Auch hier wird wie im Kapitel 7 vorgegangen.

Kapitel 10: Kapitel 10 faßt die Ergebnisse zusammen. Außerdem werden einige Ideen für weiterführende Forschungsarbeiten vorgestellt.

Teil I

Grundlagen

Kapitel 2

Grundlagen: Parallele Speicherarchitekturen

Das Ziel dieses Kapitels besteht darin, die vorliegende Arbeit in den größeren Zusammenhang paralleler Rechnerarchitektur einzuordnen. Dabei wird vor allem die Bedeutung der Speicherarchitektur für die Effizienz und die Programmierbarkeit eines Parallelrechners hervorgehoben. So wird auch deutlich, warum die symmetrischen Multiprozessoren mit ihrer einheitliche Speichersicht eine so wichtige Rechnerklasse sind.

Abschnitt 2.1 beschäftigt sich mit den Grundkonzepten paralleler Rechnerarchitekturen und Programmiermodelle. Daraufhin werden in den Abschnitten 2.2 und 2.3 die beiden wichtigsten Aspekte der parallelen Rechnerarchitektur genauer erläutert: der Kontrollfluß und die Speicherarchitektur. Als nächstes wird im Abschnitt 2.4 gezeigt, wodurch sich gute Programmiermodelle für Parallelrechner auszeichnen. Abschnitt 2.5 widmet sich der Frage, welche Voraussetzungen eine Rechnerarchitektur erfüllen muß, damit gut Programmiermodelle effizient implementiert werden können. Dabei wird die Schlüsselrolle des CC-UMA Speichermodels verdeutlicht, daß den symmetrischen Multiprozessoren zugrunde liegt.

2.1 Grundüberlegungen

Ein Parallelrechner ist eine Ansammlung von Prozessoren, die gemeinsam an der Lösung eines Problems arbeiten. Dabei soll die Lösungszeit gegenüber einem sequentiellen Rechner möglichst stark verkürzt werden. Gleichzeitig möchte man möglichst wenig Arbeit in die Parallelisierung des Problems stecken. Die Leistungsfähigkeit eines Parallelrechners wird demnach nach zwei Gesichtspunkten beurteilt: seiner Effizienz und der Programmierbarkeit. Unter der Effizienz versteht man dabei den Geschwindigkeitsgewinn, den man durch die Parallelisierung von Anwendungen erzielen kann. Die Programmierbarkeit bestimmt den Arbeitsaufwand, der zum Erreichen dieses Geschwindigkeitsgewinns notwendig ist. Von ihr hängt es ab, wie schwierig es ist, den parallelen Algorithmus zu implementieren, wie leicht das resultierende Programm zu warten ist, und wie portabel es ist.

Die Parallelisierung einer Anwendung ähnelt dem Versuch, eine komplexe Aufgabe auf ein Team von Mitarbeitern möglichst effizient zu verteilen. Dazu muß man zunächst einen Plan (Algorithmus) formulieren, wie die Aufgabe sinnvoll aufgeteilt werden kann. In einem idealen Team würden die Mitglieder in der Lage sein, den Plan durchzulesen, sich selbst die notwendigen Unterlagen zu suchen und ihn durchzuführen. In Wirklichkeit klappt dies jedoch selten. Das Team würde die Aufgabe entweder gar nicht oder nur ineffizient bewältigen. In der Regel ist es daher notwendig, jedem seine Rolle in dem Team zu erklären und ihm die benötigten Unterlagen zur Verfügung zu stellen. Außerdem muß man dafür sorgen, daß die Teammitglieder miteinander reden (kommunizieren) und ihre Vorgehensweisen aufeinander abstimmen (ihre Arbeit synchronisieren). Je besser ein Team ist, um so weniger braucht man seinen Mitgliedern zu erklären, damit sie ihre Aufgabe schnell und gut bewältigen.

Analog wäre ein ideales paralleles System ein solches, das lediglich eine abstrakte Beschreibung des parallelen Algorithmus benötigt. In Wirklichkeit gibt es einen solchen idealen Parallelrechner leider nicht. Für eine effiziente Implementierung muß sich der Programmierer um eine Vielzahl technischer Details kümmern. Dabei müssen die Aufgaben auf die Prozessoren verteilt und die benötigten Daten den entsprechenden Prozessoren zur Verfügung gestellt werden. Gleichzeitig muß man sicherstellen, daß die Prozessoren miteinander kommunizieren und sich miteinander synchronisieren. Dabei gilt wie bei einem menschlichen Team, daß ein System um so besser ist, je

weniger man sich explizit mit den Details des parallelen Programmablaufs beschäftigen muß, um eine effiziente Ausführung zu gewährleisten.

2.1.1 Das PRAM-Modell

Ein idealer Parallelrechner, der einfach zu programmieren und dabei für den jeweiligen Algorithmus optimale parallele Effizienz garantiert, wird durch das PRAM-Modell (Parallel Random Access Machine) beschrieben [47]. Es ist eine Erweiterung des elementaren RAM (Random Access Machine) Maschinenmodells, das eine Idealisierung eines sequentiellen Rechners darstellt. Ein PRAM-Rechner besteht aus P Prozessoren, die an einen gemeinsamen Speicher angeschlossen sind. Das Modell geht davon aus, daß

1. die Prozessoren auf den Speicher in einer uniformen, von dem Datum und dem Zugriffsmuster unabhängigen Zeit zugreifen können,
2. alle Prozessoren das Programm synchron Zeile für Zeile abarbeiten und
3. für die Programmausführung beliebig viele Prozessoren zur Verfügung stehen.

Zur Implementierung eines parallelen Algorithmus in PRAM-Modell müssen für jede Programmanweisung lediglich spezifiziert werden, welche Prozessoren die Anweisung ausführen müssen, und auf welche Daten welcher Prozessor zugreifen soll.

2.1.2 Reale Parallelrechner

Ein realer Parallelrechner ist eine Ansammlung von Prozessoren, Speicherbänken und Ein/Ausgabe-Komponenten, die mit Hilfe eines Verbindungsnetzwerks zu einer Einheit zusammengefügt wurden. Die Unterschiede zwischen der Architektur realer Parallelrechner und der PRAM können in zwei Bereiche eingeteilt werden: den Kontrollfluß und der Speicherarchitektur.

Kontrollfluß: Im allgemeinen arbeiten die Prozessoren eines Parallelrechners weder synchron, noch führen sie Anweisung für Anweisung das gleiche Programm aus.

Speicherarchitektur: Es ist in der Praxis nicht möglich, ein System zu realisieren, bei dem alle Prozessoren in Einheitszeit und ohne Konflikte auf einen gemeinsamen Speicher zugreifen können.

Hinzu kommt, daß bei realen Parallelrechnern nicht immer genug Prozessoren zur Verfügung stehen, um den maximalen Parallelitätsgrad einer jeden Anwendung zu nutzten.

2.1.3 Programmiermodell und Rechnerarchitektur

Bei der Betrachtung eines Parallelrechners muß man zwischen zwei Ebenen unterscheiden: dem Programmiermodell und der Rechnerarchitektur. Das Programmiermodell stellt die logische Sicht des Rechners dar. Es gibt die Konstrukte an, mit deren Hilfe die Verteilung der Berechnung und der Daten sowie die Kommunikation und die Synchronisation stattfinden. Damit ist es für die Programmierbarkeit ausschlaggebend. Unter der Architektur versteht man den physikalischen Aufbau des Rechners. Sie stellt die Mechanismen zur Verfügung, auf die die Konstrukte des Programmiermodells abgebildet werden. Dabei geht es vor allem darum, wie die Prozessoren miteinander und mit dem Speicher verbunden sind. Zusammen mit der für die Implementierung verwendeten Technologie bestimmt die Architektur die fundamentalen Grenzen der Effizienz eines Parallelrechners.

Die Effizienz eines Programms auf einer bestimmten Rechnerarchitektur hängt in der Regel sehr stark davon ab, wie gut das Programm an die Eigenheiten der Architektur angepaßt ist. Ein gute Effizienz kann nur ein Programm erreichen, das die Stärken und Schwächen einer Architektur auf Hardwareebene optimal berücksichtigt. Auf der anderen Seite setzt ein gutes Programmiermodell voraus, daß der Programmierer von den Details der Rechnerarchitektur möglichst weit abstrahieren kann. Damit ist es Aufgabe des Übersetzers und des Betriebssystems, aus der abstrakten Algorithmenbeschreibung einen an die Architektur angepaßten Code zu generieren. Die Leistungsfähigkeit eines Parallelrechners hängt somit in der Praxis vor allem von der Wechselwirkung zwischen dem Programmiermodell und der Rechnerarchitektur ab.

Eine gute Parallelrechnerarchitektur zeichnet sich dadurch aus, daß sie auch bei der Verwendung eines abstrakten, einfachen Programmiermodells eine hohe Effizienz erreicht.

Wie in den nachfolgenden Abschnitten gezeigt wird, erfüllen von den heutigen Parallelrechnerarchitekturen die symmetrischen Multiprozessoren die obige Forderungen am besten. Darin liegt die Motivation für die vorliegende Arbeit.

2.2 Der Kontrollfluß in realen Parallelrechnern

Die bekannteste Klassifikation der Parallelrechner in Bezug auf den Kontrollfluß wurde von Flynn in [45] vorgenommen. Bei der Flynn'schen Klassifikation kommt es darauf an, ob die Daten- und Befehlsströme der Prozessoren gemeinsam oder getrennt sind. In diesem Sinne kann man zwischen vier Konzepten unterscheiden: gemeinsamer Daten- und Befehlsstrom (SISD für single instruction single data), getrennter Datenstrom und gemeinsamer Befehlsstrom (SIMD für single instruction multiple data), getrennter Datenstrom und getrennter Befehlsstrom (MIMD für multiple instruction multiple data), sowie gemeinsamer Datenstrom und getrennter Befehlsstrom (MISD für multiple instruction single data). Von diesen vier Konzepten sind nur MIMD und SIMD für die parallele Rechnerarchitektur von Bedeutung. Das SISD-Konzept enspricht einen sequentiellen Rechner während das MISD in Vektorrechnern angewendet wurde.

2.2.1 SIMD-Architekturen

Bei einem SIMD-Rechner wird das Programm ähnlich wie bei einer PRAM abgearbeitet. Alle Prozessoren bearbeiten die Anweisungen synchron im Gleichschritt. Sie führen also zu jedem Zeitpunkt alle jeweils die gleiche Anweisung aus. Damit sie nicht alle exakt das Gleiche tun, kann jeder Prozessor auf seine eigenen Daten zugreifen. Außerdem besteht die Möglichkeit, daß bei bestimmten Anweisungen bestimmte Prozessoren aussetzen, also nichts tun. So können Verzweigungsanweisungen implementiert werden, bei denen unterschiedliche Prozessoren unterschiedliche Zweige wählen.

Das SIMD-Modell hat zwei Vorteile. Zum einen wird keine zusätzliche Synchronisation benötigt. Dies vereinfacht die Programmierung und vermeidet Verzögerungen durch explizite Synchronisationsanweisungen. Zum anderen werden durch den gemeinsame Befehlsstrom die Kosten für die Implementierung des Rechners gesenkt. Die für das Verwalten der Instruktionen verantwortliche Logik wird nämlich nur einmal, zentral benötigt. Die einzelnen Prozessoren werden dadurch einfacher. Dies macht das Konzept besonders attraktiv für massiv parallele Maschinen.

Ein großer Nachteil der SIMD-Architekturen besteht darin, daß sie bei Verzweigungsanweisungen sehr ineffizient sind. Dies liegt daran, daß die Prozessoren in unterschiedlichen Zweigen nicht parallel arbeiten können. Statt dessen werden die Zweige sequentiell abgearbeitet, wobei nur die Prozessoren aktiv sind, die den jeweiligen Zweig genommen haben. Die anderen setzten aus und warten, bis ihr Zweig an der Reihe ist.

2.2.2 MIMD-Architekturen

Bei einem MIMD-Rechner hat jeder Prozessor sein eigenes Programm und seine eigenen Daten. Die Prozessoren arbeiten voneinander unabhängig und asynchron. Dadurch wird zum einen eine effizientere Ausführung von Verzweigungen möglich. Außerdem kann der Rechner vielseitiger genutzt werden. So ist es möglich, eine Maschine in mehrere Partitionen zu unterteilen, die jeweils an einem anderen Programm arbeiten. Hinzu kommt, daß es sich bei den einzelnen Prozessoren um kommerzielle Bausteine handeln kann, die serienmäßig in sequentiellen Rechner eingesetzt werden. Damit ist das MIMD-Konzept auch für die Koppelung von Arbeitsplatzrechnern zu Clustern geeignet.

Der Nachteil von MIMD-Architekturen besteht in den Kosten für die Synchronisation. Sie kann auf zwei Arten erfolgen: mit Hilfe eines speziellen Synchronisationsnetzwerks oder wie in 2.4.3 beschrieben über Speicheroperationen.

2.3 Parallele Speicherarchitekturen

Bei der Implementierung des Speichersystems eines Parallelrechners müssen in der Praxis verschiedene physikalische und technologische Randbedingungen berücksichtigt werden:

1. Die Prozessoren und Speicherbausteine haben eine nicht vernachlässigbare Ausdehnung. Somit ist eine für alle Prozessoren und Speicherstellen einheitliche Speicherzugriffszeit aufgrund der endlichen Signalausbreitungsgeschwindigkeit prinzipiell nicht möglich.

2. Für die Verbindung zwischen den Prozessoren und dem Speicher wird ein Netzwerk benötigt. Die Latenz der heutigen Netzwerke liegt um mindestens eine bis zwei Größenordnungen über der Prozessortaktzeit. Dies macht sich selbst bei sequentiellen Rechnern bemerkbar, bei denen die Prozessor-Speicher-Koppelung in der Regel mit Hilfe eines Busses mit 50 bis 100 ns Latenz erfolgt (bei Prozessor-Befehlszyklen von 2-4 ns). Mit steigender Anzahl von Prozessoren und Speicherbauteilen verschlechtert sich die Latenz rapide und liegt bei größeren Maschinen im μs Bereich. Die Gründe hierfür sind in den technologischen und zunehmend auch fundamentalen, physikalischen Schranken der elektronischen Verbindungstechnologie zu finden, die in Kapitel 4 genauer erläutert werden.

3. Da bei einer PRAM die Speicherlatenz nicht von dem Zugriffsmuster der Prozessoren abhängt, werden zu ihrer Implementierung Speicherbausteine benötigt, die einen echt parallelen Zugriff für alle Prozessoren ermöglichen. Dies ist jedoch höchstens für sehr wenige Prozessoren praktikabel, da bei der heutigen Speichertechnologie die Fläche eines Speicherbausteins quadratisch mit der Anzahl paralleler Zugriffsports ansteigt. Wie in Kapitel 4 genauer erläutert, hängt dies mit fundamentalen Schranken für die Komplexität planarer VLSI-Verbindungen mit einer festen Anzahl an Leitungsebenen zusammen.

4. Aufgrund der geringen Zugriffsgeschwindigkeit des DRAM Hauptspeichers und der großen Speicherbus-Latenzen wird in sequentiellen Rechnern eine hierarchische Speicherarchitektur verwendet. Dabei werden kleine, schnelle Caches benutzt, um einen effizienten Zugriff auf oft genutzte Daten zu ermöglichen.

Die obigen Randbedingungen machen eine perfekte Implementierung des PRAM Speichermodells unmöglich. Statt dessen werden in der Praxis verschieden gute Näherungen entwickelt. Wie unten genauer beschrieben, unterscheidet sich die Architektur paralleler Speichersysteme von der PRAM in vier Bereichen: der physikalischen Struktur des Speichers, der Struktur des Adreßraums (logischen Struktur des Speichers), der Verteilung der Zugriffslatenzen, und der Semantik von parallelen Speicherzugriffen. Letztere wird durch die Cache-Kohärenzund das Speicherkonsistenz-Modell bestimmt.

2.3.1 Physikalische Struktur des Speichers

Das PRAM-Modell geht davon aus, daß die Prozessoren alle direkt an eine kompakte Speichereinheit angeschlossen sind. In Wirklichkeit besteht ein Parallelrechner aber aus einer Menge von Prozessoren, jeder mit seinem eigenen Cache-System, und einer Menge von Speicherbänken. Sie sind miteinander durch ein Netzwerk verbunden, das jedem Prozessor den Zugriff auf jede Speicherbank ermöglicht. Bei der Verbindung wird zwischen zwei Grundarchitekturen unterschieden: einem *gemeinsamen Speicher* (GS) und einem *verteilten Speicher* (VS).

Gemeinsamer Speicher

Beim gemeinsamen Speicher (Abbildung 2.2a) befinden sich die Prozessoren mit ihren Cache-Subsystemen auf der einen, und die Speicherbänke auf der anderen Seite des Netzwerks. Dabei werden die Speicherbänke in Anlehnung an das PRAM-Modell zu einer Einheit zusammengefügt. Dies erlaubt eine einfache und effiziente Implementierung eines PRAM-ähnlichen Programmiermodells. Das Problem der GS-Architektur besteht in der Schwierigkeit, sie für größere Prozessorzahlen zu realisieren. Mit steigender Prozessorzahl steigen auch die Netzwerklatenz und damit die Speicherlatenz. Die einheitliche Sicht des Speichers bedeutet dann eine einheitlich ineffiziente Sicht. Gleichzeitig steigen die Kosten für die Implementierung des Systems überproportional an. Das Ergebnis ist dann ein sehr ineffizientes, sehr teures System. Um dies zu vermeiden, beschränkt man sich heute bei der Implementierung von GS-Rechnern im wesentlichen auf Rechner mit wenigen Prozessoren (in der Regel 4 bis 16, maximal 64).

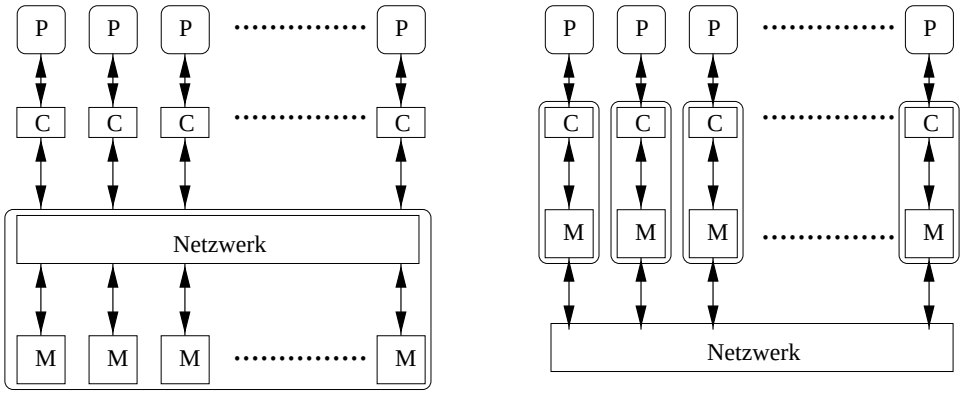


Abbildung 2.1: Der Unterschied zwischen gemeinsamen und verteilten Speicher. In der Abbildung stehen P für einen Prozessor, C für einen Cache und M für eine Speicherbank.

Verteilter Speicher

Die Probleme bei der Realisierung des GS für große Prozessorzahlen haben zur Entwicklung des VS-Konzeptes geführt. Dabei werden die Speicherbänke auf die einzelnen Prozessoren verteilt und jedem Prozessor ein eigener *lokaler Speicher* zugeordnet. Anders als beim GS verbindet das Netzwerk nun nicht die Prozessoren mit den Speicherbänken, sondern die Prozessoren bzw. deren Speichersubsysteme untereinander. Die Koppelung zwischen dem Prozessor und den Speichereinheiten seines lokalen Speichers ist weitgehend mit der Prozessor-Speicher Verbindung in einem sequentiellen Rechner identisch. Dementsprechend effizient sind auch die Zugriffe auf die lokalen Daten. Zugriffe auf die Daten in den Speichern anderer Prozessoren (*globale Zugriffe*) können hingegen in der Regel wesentlich langsamer sein.

Im Vergleich zum GS besteht der Vorteil des VS darin, daß es wesentlich geringere Anforderungen an das Verbindungsnetzwerk stellt. Dies liegt daran, daß das Netzwerk nur noch einen Teil der Speicherzugriffe, nämlich die globalen Zugriffe bewältigen muß. Dadurch wird das Netzwerk zum einem weniger belastet und kommt mit einer geringeren Bandbreite aus. Zum anderen wirkt sich eine hohe Netzwerklatenz weniger stark auf die Programmeffizienz aus, da sie nur eine Untermenge aller Speicherzugriffe verlangsamt. Beides zusammen läßt die Realisierung von Parallelrechnern auch dann zu, wenn eine zu hohe Netzwerklatenz oder zu hohe Implementierungskosten die Verwirklichung eines gemeinsamen Speichers unmöglich machen. Gleichzeitig nimmt man allerdings eine inhomogene Speicherstruktur und geringe Effizienz für Anwendungen mit einem hohen Anteil globaler Zugriffe in Kauf. In diesem Sinne ist der VS ein Notkompromiß.

2.3.2 Struktur des Adreßraums

Aus der Sicht des Programmierers ist neben dem physikalischen Aufbau des Speichersystems auch seine logische Struktur wichtig, die durch den Aufbau des Adreßraums gegeben ist. Bei der PRAM sind beide trivialer Weise identisch. In der Praxis kann der Adreßraum dagegen unabhängig von dem physikalischen Aufbau des Speichers gemeinsam oder getrennt sein.

Gemeinsamer Adreßraum

Bei einem gemeinsamen Adreßraum erfolgt die Adresszuteilung wie bei einer PRAM. Jede Speicherstelle hat eine eindeutige, globale Adresse, über die jeder Prozessor auf sie zugreifen kann. Der Speicher erscheint allen Prozessoren als eine logische Einheit. Der gemeinsame Adreßraum stellt so die offensichtliche logische Speicherstruktur für Rechner mit physikalisch gemeinsamem Speicher. Er kann aber auch auf physikalisch verteiltem Speicher implementiert werden. Man spricht in diesem Fall vom *virtuellen gemeinsamen Speicher*. Dabei sind die Netzwerkschnittstellen in das Speichersubsystem der Prozessoren integriert. Sie sorgen dafür, daß Zugriffe auf globale Daten über das Netzwerk zu dem richtigen Prozessor geschickt werden.

Bei Rechnern mit virtuellen gemeinsamem Speicher stellt die Zuordnung der Adressen zu den Speichersubsystemen ein weiteres Unterscheidungskriterium dar. In den meisten Fällen erfolgt sie nach einem festen statischen

Muster. Adressen zwischen `0` und `x` liegen im lokalen Speicher des ersten Prozessors, die zwischen `x+1` und `2x` im lokalen Speicher des zweiten Prozessors usw. Es gibt aber auch Ansätze eine dynamische, bedarfsorientierte Zuordnung vorzunehmen. Dabei wird eine Adresse immer dem lokalen Speicher des Prozessors zugewiesen, der auf das Datum zugreift. Der gesamte Speicher funktioniert wie ein großer, paralleler Cache. Solche Speichersysteme werden daher *COMA* für 'cache only memory' genannt ([2, 72, 129]).

Getrennter Adreßraum

Bei getrenntem Adreßraum beziehen sich die Adressen jeweils auf die lokalen Speicher der Prozessoren. Speicherstellen müssen also durch die Angabe einer lokalen Adresse und einer Prozessornummer identifiziert werden. Getrennter Adreßraum stellt das logische Gegenstück zum VS dar. Die klare Trennung zwischen lokalen und globalen ermöglicht die Optimierung der Programme im Hinblick auf die unterschiedliche Latenz verschiedener Speicherzugriffe. Im Gegenzug wird die Programmierung allerdings erheblich erschwert und die Portabilität der Programme stark eingeschränkt.

2.3.3 Speicherlatenz

Das Problem der Speicherlatenz wurde bereits im Zusammenhang mit der physikalischen Speicherstruktur angesprochen. Es stellt den wichtigsten Unterschied zwischen der PRAM und den praktischen Rechnerarchitekturen dar. Bei der PRAM kann ein Speicherzugriff immer in einem Taktzyklus erfolgen. Dies ist in der Realität nicht mal bei sequentiellen Rechnern der Fall. Je nachdem, ob sich ein Datum im Hauptspeicher oder im Cache befindet, kann die Zugriffszeit zwischen einem und mehreren hundert Prozessorzyklen betragen. Bei Parallelrechnern kommen zu solchen Speicherhierarchie-bedingten Unterschieden in der Speicherlatenz weitere, durch die Parallelität verursachte Verzögerungen hinzu. Dies liegt vor allem daran, daß mit steigender Prozessor- und Speicherbankzahl:

1. die Netzwerklatenz und Signallaufzeit zwischen den Prozessoren und den Speicherbänken zunehmen,
2. es immer öfter zu Zugriffskonflikten auf Speicherbänken bzw. Netzwerkkomponenten und damit zu Sequentialisierungsverzögerungen kommt.

Je nach Größe solcher zusätzlichen Verzögerungen unterscheidet man in der Praxis zwei Arten paralleler Speicherarchitekturen: Speicherarchitekturen mit uniformer Zugriffszeit (UMA für uniform memory access) und nichtuniformer Zugriffszeit (NUMA für non uniform memory access).

Uniforme Zugriffszeit (UMA)

Bei UMA-Architekturen liegen die durch Parallelität verursachte Latenzvergrößerung nicht wesentlich über der normalen Speicherlatenz. Es wird daher nicht zwischen lokalen und globalen Zugriffen unterschieden. Alle Speicherzugriffe werden gleich behandelt und haben näherungsweise die gleiche Latenz.

Der Vorteil solcher Architekturen besteht darin, daß die Effizienz eines Programms nur geringfügig von der Verteilung der Daten auf die Prozessoren und dem Kommunikationsmuster abhängt. Man kann daher mit Hilfe eines einfachen, portablen Programmiermodells effiziente Programme für ein breites Spektrum von Anwendungen schreiben. Auch kommunikationsintensive Programme können ohne großen Aufwand parallelisiert werden.

Leider konnte das UMA Modell bisher nur auf kleinen (üblicherweise 4 bis 16 Prozessoren) Rechnern mit gemeinsamem Speicher und Adreßraum implementiert werden. Bei größeren Prozessorzahlen bzw. verteiltem Speicher war man bisher nicht in der Lage, eine ausreichend geringe Latenz von globalen Speicherzugriffen zu realisieren.

Nicht-Uniforme Zugriffszeit (NUMA)

Bei NUMA-Architekturen sind die durch Parallelität verursachten Kosten gegenüber den normalen Speicherzugriffskosten nicht vernachlässigbar. Zu dieser Rechnerklasse gehören vor allem große Parallelrechner mit verteiltem Speicher und Arbeitsplatzrechner-Cluster. Dabei können die Kosten für globale Zugriffe auf bis zu 10000 Prozessorzyklen steigen. Würde man hier wie im UMA-Modells die unterschiedlichen Latenzen ignorieren, so würde dies zu uniform sehr langsamem Speicher führen. Um dies zu vermeiden, unterscheidet das NUMA explizit zwischen lokalen Zugriffen auf die privaten Daten eines Prozessors und globalen Zugriffen auf gemeinsame, parallel genutzte Daten. Das Speichersystem ist so konzipiert, daß die Kosten für die lokalen Zugriffe in der Nähe der

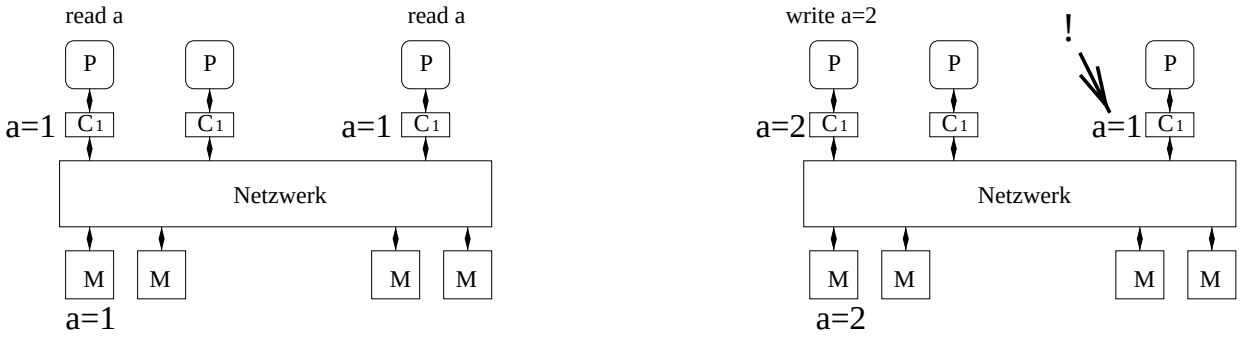


Abbildung 2.2: Das Problem der Cache-Kohärenz. Das linke Bild zeigt den Zustand eines Systems nachdem der erste und der letzte Prozessoren die Variable a gelesen haben, die zu dem Zeitpunkt den Wert 1 hat. Das rechte Bild zeigt das System nachdem der erste Prozessor den Wert 2 in die Variable a geschrieben hat, ohne den Zugriff zu den anderen Caches zu propagieren.

Hauptspeicher-Zugriffskosten sequentieller Rechner liegen. Die Optimierung der Latenz globaler Zugriffe wird als zweitrangig betrachtet. Sie wird nur dann durchgeführt, wenn sie die Effizienz lokaler Zugriffe nicht beeinträchtigt. Die Motivation für die NUMA-Architektur liegt in der Erkenntnis, daß

1. bei vielen parallelen Algorithmen Kommunikationsoperationen nur einen kleinen Anteil an Anweisungen ausmachen und
2. es möglich ist, die globalen Speicherzugriffe eines Programms auf die vom Algorithmus vorgegebenen Kommunikationsanweisungen zu beschränken.

Hinzu kommen viele sog. Latenzverbesserungs-Techniken (z.B. [26, 113]), die die Auswirkung der hohen Latenz globaler Zugriffe auf die Programmlaufzeit mildern. Somit lassen sich durch eine geschickte Implementierung viele Anwendungen auf einem NUMA Rechner trotz hoher Latenz globaler Zugriffe effizient parallelisieren.

Verglichen mit der UMA-Architektur hat die NUMA-Speicherarchitektur zwei große Nachteile (siehe auch Abschnitt 2.5): sie schränkt die Menge der effizient parallelisierbaren Anwendungen stark ein und läßt kein einfaches und gleichzeitig effizientes Programmiermodell zu. Beide Probleme hängen mit der Notwendigkeit zusammen, die Anzahl der globalen Speicherzugriffe zu minimieren. Zum einen können deswegen grundsätzlich keine kommunikationsintensiven Algorithmen effizient implementiert werden. Dadurch wird eine Vielzahl von interessanten Anwendungsgebieten von der Parallelisierung ausgeschlossen. Zum anderen ist auch bei nicht kommunikationsintensiven Problemen eine effiziente Implementierung oft sehr aufwendig oder gar nicht möglich. Dies liegt daran, daß die Anzahl globaler Speicherzugriffe nicht nur von dem Kommunikationsaufkommen des Algorithmus, sondern auch ganz entscheidend von der Verteilung der Daten auf die Prozessoren abhängt. Die durch den Algorithmus vorgegebene Anzahl von Kommunikationsoperationen stellt lediglich eine untere Schranke dar, die nur durch eine optimale Datenverteilung erreicht werden kann.

2.3.4 Cache-Kohärenz

In einem Parallelrechner führt die Speicherhierarchie dazu, daß im System gleichzeitig mehrere Kopien von Speicherinhalten existieren können. Im Extremfall kann sogar jeder Prozessor in seinem Cache eine Kopie einer und derselben Speicherstelle besitzen. Um eine korrekte Programmausführung zu garantieren, müssen bei den meisten Schreibzugriffen alle Kopien auf den neusten Stand gebracht werden. Ein Schreibzugriff muß also nicht nur zum Hauptspeicher, sondern gleichzeitig auch zu allen Caches geschickt werden, in denen sich die Kopie des Datums befindet (Abbildung 2.2). Man spricht dabei von der Erhaltung der Cache-Kohärenz(siehe z.B. [164]). Um die Bedeutung der Cache-Kohärenzzu veranschaulichen, betrachten wir das folgende Programm. In dem Programm

```
  P1              P2
work();           A=1;
A=0;           while(A!=0);
```

benutzt Prozessor 1 die Variable `A`, um Prozessor 2 aus der `while` Warteschleife zu entlassen. In einem hierarchischen System muß man davon ausgehen, daß Prozessor 2 die Variable `A` im Cache hat. Er würde die `while`-Schleife daher nie verlassen, wenn der Schreibzugriff von Prozessor 1 nur zum Hauptspeicher weitergeleitet werden würde.

Wie in Kapitel 3 genauer erläutert, erfordert die Erhaltung der Cache-Kohärenzeinen sehr großen Aufwand. Aus diesem Grund wird sie bei manchen Speichersystemen nicht automatisch gewährleistet, sondern dem Programmierer überlassen. Solche Systeme werden nicht Cache-kohärent (NCC) genannt. Systeme, bei denen das Speichersystem die Erhaltung der Cache-Kohärenzsicherstellt, werden entsprechend als Cache-kohärent (CC) bezeichnet.

NCC-Speichersysteme

Bei NCC-Systemen werden die Cache-Subsysteme der einzelnen Prozessoren als voneinander unabhängig betrachtet. Jeder Prozessor kann also Daten in seinen Cache laden und beschreiben, ohne daß die anderen Prozessoren etwas davon mitkriegen. Der Vorteil besteht hierbei darin, daß das Speichersystem einfach gehalten werden kann. Insbesondere wird die Latenz der Zugriffe auf die einzelnen Caches nicht von der Anzahl der Prozessoren und der Effizienz des Netzwerks beeinflußt.

Die Nachteile liegen in der schwierigen Programmierbarkeit. Der Programmierer (oder der Compiler) muß sich bei jedem Schreibzugriff auf ein Datum im Cache überlegen, ob dieses Datum eventuell in einem anderen Cache vorhanden ist, und wenn ja, ob es aktualisiert werden muß. Da dies im allgemeinen nicht mit vertretbarem Aufwand möglich ist, werden bei NCC-Systemen oft gemeinsam genutzte Daten grundsätzlich nicht im Cache zwischengespeichert. Dies vereinfacht zwar die Programmierung, hat aber bei vielen Anwendungen eine deutliche Verschlechterung der Effizienz zur Folge.

CC-Systeme

Bei CC-Architekturen stellt das Speichersystem sicher, daß jede Veränderung eines Datums in allen den Caches sichtbar wird, in denen sich dieses Datum befindet. Dies kann auf zwei Arten verwirklicht werden. Zum einen kann jede Schreiboperation grundsätzlich für alle Caches sichtbar sein. Zum anderen kann für jede Cache-Zeile in einem Verzeichnis gespeichert werden, in welchen Caches sie sich befindet, damit der veränderte Wert gezielt an diese Caches weitergeleitet werden kann. Das Problem beider Lösungen besteht darin, daß sie die Implementierung der Cache- und Speicherarchitektur erschweren. Außerdem werden zum Teil die Cache-Zugriffe verlangsamt. Besonders problematisch ist die Tatsache, daß sowohl die Komplexität der Implementierung als auch die Verlangsamung der Cache-Zugriffe mit steigender Prozessorzahl zunehmen. Aus diesem Grund stellt die effiziente Realisierung der Cache-Kohärenz für große Prozessorzahlen zur Zeit eines der wichtigsten ungelösten Probleme der Rechnerarchitektur dar. Eine Übersicht über den Stand der Forschung ist z.B. in [46, 27] zu finden. Die wichtigsten Ansätze werden auch im nachfolgenden Kapitel erläutert.

2.3.5 Speicherkonsistenz

Die Speicherkonsistenz [160, 55, 14] definiert das genaue Verhalten eines Speichersystems bei parallelen Zugriffen. Dabei geht es zum einen um die Frage, in welcher Reihenfolge parallele Speicherzugriffe den Hauptspeicher erreichen. Zum anderen muß geklärt werden, wann durch Schreibzugriffe verursachte Veränderungen des Hauptspeichers für welche Prozessoren sichtbar werden. Die Speicherkonsistenz bestimmt somit die Timing-Anforderungen an das Cache-Kohärenz Protokoll. Die Speicherkonsistenz wird vor allem durch vier Gesichtspunkte bedingt:

1. die asynchrone Arbeitsweise der Prozessoren,
2. die Abhängigkeit der Speicherzugriffslatenz von der Prozessornummer und der Speicheradresse,
3. die Existenz von lokalen Kopien von Daten (in den Caches oder in einem Schreibpuffer),
4. Optimierungstechniken, die die Reihenfolge von Speicherzugriffen eines Prozessors verändern können (z.B. out of order execution, nichtblokierende Zugriffe etc.).

Um die Problematik zu verdeutlichen, betrachten wir das nachfolgende Programmstück.

```
  P1              P2

B = 1;          A = 0;
A = 1;          B = 0;
```

Im PRAM-Modell würde es 1 in `A` und 0 in `B` schreiben, da die beiden Prozessoren ihre Anweisungen synchron ausführen und alle Speicherzugriffe in Einheitszeit erfolgen. In der Realität ist dagegen je nach Speicherkonsistenz-Modell eine beliebige Permutation von 0 und 1 möglich. Eine interessante Möglichkeit ergibt sich in einem System mit verteiltem Speicher und gemeinsamem Adreßraum, in dem:

1. die Prozessoren asynchron arbeiten,
2. jede Prozessor seine eigenen Anweisungen in der durch das Programm vorgegebenen Reihenfolge ausführt,
3. die Schreibzugriffe nicht blockierend sind,
4. `A` eine lokale Variable von Prozessor 1, und `B` eine lokale Variable von Prozessor 2 ist,
5. und die Prozessoren auf ihre eigenen lokalen Variablen wesentlich schneller als auf fremde Variablen zugreifen können.

Die Differenz in den Speicherzugriffszeiten auf `A` und `B` führt hier dazu, daß die Schreiboperationen den Hauptspeicher nicht in der Reihenfolge erreichen, in der sie von dem jeweiligen Prozessor ausgeführt wurden. So ist es möglich, daß das Programm eine 1 in `B` und eine 0 in `A` schreibt. In einem System, in dem die Zugriffe eines jeden Prozessors in der Reihenfolge den Speicher erreichen, in der sie ausgeführt wurden, ist ein solches Ergebnis dagegen nicht möglich. Aus $B = 1$ würde nämlich folgen, daß die Anweisung `B=1` von Prozessor 1 nach der Anweisung `B=0` von Prozessor 2 ausgeführt wurde. Unter der Annahme, daß jeder Prozessor seine eigenen Anweisungen in der vorgegebenen Reihenfolge ausführt, würde das aber bedeuten, daß die Anweisung `A=1` von Prozessor nach der Anweisung `A=0` von Prozessor 2 erfolgt ist. Aus $B = 1$ würde also zwangsläufig $A = 1$ folgen.

Die beiden wichtigsten Speicherkonsistenzmodelle sind die sequentielle Konsistenz sowie verschiedene Varianten der sog. schwachen Konsistenz. Das Prinzip dieser beiden Modelle ist im Nachfolgenden erläutert. Die Entwicklung neuer Modelle stellt ein aktuelles Forschungsthema dar. Eine ausführliche Einführung in die Problematik der Speicherkonsistenz ist in [5] zu finden.

Sequentielle Konsistenz

Die sequentielle Konsistenz ([97]) definiert eine einfache, einem sequentiellen Rechner sehr ähnliche Speichersemantik. Es gilt : Ein Speichersystem ist sequentiell konsistent, wenn das Ergebnis einer parallelen Berechnung mit dem Ergebnis einer sequentiellen Berechnung identisch ist, bei der

1. alle Speicherzugriffe in einer wohldefinierten Reihenfolge sequentiell durchgeführt werden,
2. die Speicherzugriffe eines jeden einzelnen Prozessors in der vom Programm vorgegebenen Reihenfolge durchgeführt werden.

Ein großer Nachteil der sequentiellen Konsistenz besteht darin, daß es blockierende Schreibzugriffe erfordert. D.h., daß ein Prozessor seine Arbeit nicht fortsetzen kann, sobald ein Schreibzugriff in das Verbindungsnetzwerk geschickt wurde. Er muß vielmehr warten, bis er zu allen anderen Prozessoren durchgedrungen ist. Dies kann zu erhebliche Effizienzverlusten führen.

Schwache Konsistenz

Die schwache Konsistenz (z.B. [189]) ist durch die Beobachtung motiviert, daß die strenge Einhaltung der sequentiellen Konsistenz nur bei wenigen Speicherzugriffen benötigt wird. Dabei handelt es sich vor allem um die Synchronisation und Zugriffe die mit ihr zusammenhängen. Sie fordert daß nur die folgenden Anweisungen sequentiel konsistent sind:

1. die Anweisungen eines Prozessors untereinander,
2. Synchronisationsanweisungen und
3. Zugriffe, die durch eine Synchronisationsbarriere getrennt sind

Die schwache Konsistenz erlaubt eine Steigerung der Leistung und Vereinfachung des Speichersystems, ohne daß deswegen die Programmierbarkeit spürbar verschlechtert wird. Sie wird heute daher in fast allen CC-Rechner verwendet.

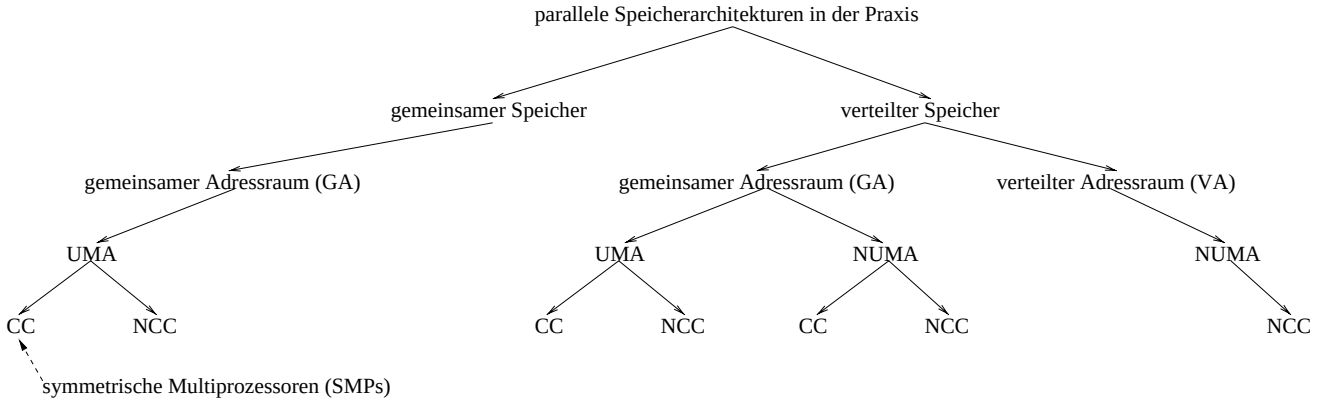


Abbildung 2.3: Ein Überblick über die in der Praxis relevanten parallelen Speicherarchitekturen.

2.3.6 Zusammenfassung

Die verschiedenen Varianten paralleler Speicherarchitektur, die sich durch die Kombination der in diesem Abschnitt beschriebenen Faktoren ergeben, sind in Abbildung 2.3 dargestellt. An oberster Stelle steht die Unterscheidung zwischen gemeinsamem und verteiltem Speicher. Diese beiden Oberklassen können wie folgt weiter eingeteilt werden:

Gemeinsamer Speicher

Im Falle eines gemeinsamem Speicher kann man immer von einem gemeinsamen Adreßraum und einer uniformen Zugriffszeit ausgehen. Beide ergeben sich automatisch aus der GS-Architektur, und gehören zu ihren wichtigsten Vorteilen. Die GS-Speichermodelle unterscheiden sich somit nur in der Frage der Cache-Kohärenz. Bei den Cache-kohärenten Architekturen stellt sich zusätzlich die Frage nach dem Modell der Speicherkonsistenz. Dabei stellt das Cache-Kohärenz-UMA-GS Modell mit sequentieller bzw. schwacher Speicherkonsistenz die beste Näherung an das PRAM-Speichermodell dar. Es erlaubt dem Programmierer, die Frage der Datenverteilung und der Kommunikation vollkommen außer acht zu lassen, ohne daß dadurch die Ausführungseffizienz des Programms beeinträchtigt wird.

Verteilter Speicher

Bei Architekturen mit verteiltem Speicher wird zunächst zwischen gemeinsamem und getrenntem Adreßraum unterschieden. Da bei getrenntem Adreßraum die lokalen Speicher der einzelnen Prozessoren keinen einheitlichen Speicher bilden, geht man von nicht-uniformer Zugriffszeit und einem nicht cache-kohärenten Speichermodell aus. Im Falle des gemeinsamen Adreßraumes wurden hingegen sowohl UMA als auch NUMA Architekturen implementiert. In beiden Fällen sind sowohl CC als auch NCC-Varianten sowie unterschiedliche Speicherkonsistenz-Modelle denkbar.

2.4 Parallele Programmiermodelle

Das Programmiermodell stellt die logische Sicht des Rechners dar. Es gibt die Konstrukte vor, mit dessen Hilfe der Programmierer die parallelen Algorithmen implementieren kann. Um die Semantik dieser Konstrukte festzulegen, definiert es gleichzeitig einen abstrakten 'virtuellen Parallelrechner'. Der Programmierer schreibt sein Programm für den abstrakten Rechner. Dieses wird dann durch den Compiler und ein Laufzeitsystem auf den echten Rechner abgebildet. Dadurch wird der Programmierer von den technischen Details der Programmausführung abgeschirmt.

Die Frage nach dem besten Programmiermodell bzw. der besten Programmiersprache ist heiß umstritten. Dies gilt für Parallelrechner genauso wie für sequentielle Rechner. Unumstritten sind allerdings die folgenden zwei Punkte:

1. Ein Programmiersystem soll eine möglichst abstrakte Formulierung von Algorithmen zulassen. Der virtuelle Parallelrechner soll also möglichst nah an dem PRAM Modell oder einem anderen abstrakten Modell der Parallelverarbeitung liegen. Dadurch werden zwei Dinge erreicht:

   (a) Die Implementierung von parallelen Algorithmen wird weniger arbeitsaufwendig und fehleranfällig.

   (b) Das Programm kann leichter zwischen verschiedenen parallelen Systemen portiert werden.

2. Es muß möglich sein, die in dem Modell formulierten Programme effizient auf reale Rechnerarchitekturen abzubilden. Das Programm muß also genug Information erhalten, um dem Übersetzer die Erzeugung eines Codes zu ermöglichen, der die Eigenheiten der reellen Maschine berücksichtigt.

Die Schwierigkeit bei der Realisierung eines guten Programmiermodells besteht darin, daß diese beiden Forderungen in der Regel miteinander unvereinbar sind. Je mehr die Formulierung des Programms von der reellen Rechnerarchitektur abstrahiert, um so schwieriger wird es für den Compiler und das Laufzeitsystem, einen effizienten Code zu erzeugen. Dies gilt insbesondere für das Speichermodell. Wie in Abschnitt 2.5 genau erläutert, ist z.B. eine optimale Abbildung eines PRAM-artigen Programms auf eine CC-NUMA-GS oder gar VS Maschine zur Übersetzungszeit gar nicht möglich.

Dies hat dazu geführt, daß sich viele Programmiermodelle stark an bestimmten konkreten Architekturen orientieren, und so die einfache Programmierbarkeit und Portabilität der Effizienz opfern. Im Nachfolgenden wird ein Überblick über die wichtigsten heutigen parallelen Programmiermodelle gegeben. Dabei werden die Modelle im Hinblick auf vier Aspekte klassifiziert:

Verwaltung der Parallelität: Ein paralleles Programmiermodell muß eine Möglichkeit bieten, parallele Aktivitäten zu erzeugen, zu verwalten und ihnen bestimmte Anweisungen zur Bearbeitung zuzuordnen.

Der Kommunikationsmechanismus: Der Kommunikationsmechanismus erlaubt den parallelen Aktivitäten untereinander Informationen austauschen zu können.

Der Synchronisationsmechanismus: Der Synchronisationsmechanismus wird benutzt, um die parallelen Aktivitäten miteinander zu koordinieren.

Das Scheduling-Verfahren: Das Scheduling-Verfahren ordnet die abstrakten parallelen Aktivitäten des Programmiermodells den Prozessoren eines Parallelrechners zu.

Das Ziel des nachfolgenden Überblicks besteht vor allem darin, die Probleme aufzeigen, die sich durch die Anlehnung der Modelle an konkrete Architekturen, insbesondere bestimmte Speichermodelle, ergeben. Dadurch wird die Bedeutung des CC-UMA-GS Speichermodells und damit die Bedeutung der SMP-Architekturen unterstrichen, dessen skalierbare Implementierung das Ziel der Arbeit darstellt.

2.4.1 Verwaltung der Parallelität

Die Grundvoraussetzung für paralleles Programmieren sind Konstrukte, die Anweisungen parallel ausführen lassen. Solche Konstrukte müssen zwei Aufgaben erfüllen. Sie müssen erstens die Erzeugung einer Menge paralleler Aktivitäten erlauben, die im Nachfolgenden als Prozesse bezeichnet werden. Außerdem müssen sie eine Möglichkeit bieten, Anweisungen bzw. Anweisungsgruppen den virtuellen Prozessoren zuzuordnen. Die heute gängigsten Möglichkeiten hierfür sind parallele Schleifen (Schleifen-Parallelität), leichtgewichtige Prozesse, sog Fäden (Fäden-Parallelität) und parallel ablaufende vollwertige Programm (Auftraggeber/Auftragnehmer-Parallelität.

Schleifen-Parallelität

Eine sehr elegante Möglichkeit, Parallelität in ein Programm zu integrieren, stellt eine parallele Schleifenanweisung dar. Ihre Syntax und Semantik sind an die sequentielle Schleife angelehnt. Letztere gibt für eine bestimmte Anweisungsgruppe an, wie oft diese hintereinander ausgeführt werden soll. Analog gibt eine parallele Schleife für eine Anweisungssequenz an, wie diese parallel ausgeführt werden soll.

Eine parallele Schleife wird meisten als `FORALL` bezeichnet. Sie hat die Syntax

```
FORALL <index> = <von> TO <bis> DO
 <Anweisungen>
END
```

Jede Anweisung innerhalb des `FORALLS` wird gleichzeitig von allen virtuellen Prozessoren ausgeführt, deren Nummern zwischen `von` und `bis` liegen. Die Laufvariable `index` stellt in Anlehnung an die Laufvariable einer sequentiellen Schleife einen Zähler für die parallele Aktivität dar. Sie hat also auf dem Prozessor 1 den Wert 1, auf dem Prozessor 2 den Wert 2, usw. So kann jeder Prozessor auf andere Daten zugreifen, indem er die Laufvariable für die Indizierung von Datenfeldern benutzt. Die `FORALL` Anweisung kann prinzipiell in eine beliebige Programmiersprache integriert werden. Beispiele für solche Sprache sind FORTRAN 90, HPF [173] und MODULA-2* [139].

Faden-Parallelität

Die Grundidee hinter dem Faden-Parallelen Modell besteht darin, Prozeduren im Rahmen parallel ablaufender, sog. leichtgewichtiger Prozesse (Fäden) auszuführen. Die Fäden sind immer Bestandteile, sozusagen Unterprozesse, eines Prozesses. Sie besitzen keinen eigenen Adreßraum, sondern laufen im dem Adreßraum des Prozesses ab, von dem sie ein Bestandteil sind. Der Vorteil von Fäden gegenüber normalen Prozessen besteht darin, daß sie einfach aufgebaut sind und dadurch schnell erzeugt und vernichtet werden können. Um eine Anweisungsgruppe im Faden-Modell parallel auszuführen, muß man die Anweisungen in einer Prozedur unterbringen, und für jeden virtuellen Prozessor einen Faden erzeugen, der diese Prozedur ausführt. Dies kann in einem Programm z.B. so aussehen:

```
VAR Thread_id: my_thread[P]

PROCEDURE thread_procedure(int id)
BEGIN
 <Anweisungen>
END

BEGIN
 FOR p=1 TO P
  my_thread[p]=create_thread(thread, thread_procedure);
 END
END
```

Das Programm erzeugt P Fäden, die alle die Prozedur `thread_procedure(int id)` ausführen. Das Feld `my_thread` beinhaltet die Kennungen der Fäden, die Funktion `create_thread` dient zu ihrer Erzeugung. Sie bekommt zwei Argumente. Das erste ist die Adresse der Prozedur, die in dem der Faden ablaufen soll. Das Zweite ist ein Wert, der an diese Prozedur übergeben wird, sobald der Faden gestartet wird. Er erfüllt die gleiche Rolle wie die Laufvariable der `FORALL` Anweisung.

Faden-Parallelität ist heute ein fester Bestandteil vieler Programmiersprachen und Betriebssysteme (z.B. Java, Modula-3 ([64]), die meisten UNIX-Varianten [8] und Windows NT [130]). Die genaue Syntax und Semantik von Faden-Bibliotheken ist von System zu System unterschiedlich. Das obige Beispiel ist an die POSIX-Faden Bibliothek [19] angelehnt, die in den meisten UNIX-Systemen integriert ist.

Auftraggeber/Auftragnehmer-Parallelität

Dieses Model geht davon aus, daß mehrere Kopien eines Programms gestartet werden. Jede Kopie läuft als eigenständiger Prozeß ab. Um eine Anweisungsfolge parallel auszuführen, muß man dafür sorgen, daß sie von mehreren Kopien des Programms zur gleichen Zeit ausgeführt wird. Dies funktioniert in der Regel nach dem Auftraggeber/Auftragnehmer Prinzip. Dabei wird ein Prozeß als Auftraggeber ausgezeichnet und mit der Ausführung der sequentiellen Anweisungen und der Ablaufkontrolle betraut. Die anderen Prozesse warten in einer Schleife auf Anforderungen vom Auftraggeber. Diese Anforderungen werden vom Auftraggeber dann geschickt, wenn ein Unterprogramm parallel ausgeführt werden soll. Die Auftragnehmer führen daraufhin die entsprechenden Anweisungen aus. Sobald sie fertig sind, benachrichtigen sie den Auftraggeber und gehen wieder in die Warteschleife.

Das Auftraggeber/Auftragnehmer-Modell ist vor allem im Zusammenhang mit Arbeitsplatzrechner-Clustern beliebt. Zu diesem Zweck wurden verschiedene Programmiersysteme entwickelt. Die bekanntesten davon sind das PVM- [165, 54] und das MPI- [65, 119] System.

2.4.2 Kommunikation

Der Datenaustausch zwischen den Prozessoren des abstrakten Parallelrechners kann grundsätzlich auf zwei Arten erfolgen: durch Zugriffe auf gemeinsam genutzte Variablen oder durch Verschicken von Nachrichten. Dementsprechend unterscheidet man in Bezug auf den Kommunikationsmechanismus zwei Arten von Programmiermodellen: speichergekoppelten und nachrichtengekoppelten. Die Programmiermodelle in der erste Klasse gehen alle von einem abstrakten Parallelrechner aus, der auf einem Speichermodellen mit gemeinsamem Adressraum basiert. Diese Klasse kann daher nach dem Muster dieser Speichermodelle weiter in CC-UMA, NCC-UMA, CC-NUMA und NCC-NUMA unterteilt werden. In Bezug auf die Speicherkonsistenz gilt, daß heute fast alle CC-Systeme die schwache Speicherkosistenz realisieren.

Um die Unterschiede zwischen den einzelnen Modellen zu verdeutlichen, wird im Nachfolgenden ein einfaches Programm verwendet, das für eine $S \times Z$ Matrix M die Elemente von ausgewählten, durch einen Vektor $\vec{A}$ vorgegebenen P Spalten sortiert. Es wird also die folgende Operation durchgeführt. Bei gegebenen M und $\vec{A} = (A_1, ..., A_P)$ mit

$$M \in \mathrm{IN}^Z \times \mathrm{IN}^S \quad A \in [1..S]^P$$

stellt das Programm also den Zustand:

$$\forall i \in [1...P] : \forall j, k \in [1..Z] : j \leq k \Rightarrow M_{A_i,j} \leq M_{A_i,k}$$

her. Wir gehen im Nachfolgenden davon aus, daß die Sortierung der Spaltenwerte Teil eines größeren Programms ist. Das Programm generiert in einem als `<Vorberechnung>` bezeichnen Abschnitt die Werte für M und $\vec{A}$ in den Variablen `A` und `M`. Das Programm kann sequentiell wie folgt formuliert werden:

```
VAR INT A[P], M[S][Z]
BEGIN

 <Vorberechnung>

 FOR i = 1 TO P DO
  sort(M[A[i]]);
 END
END
```

In der parallelen Version soll jede, der durch $\vec{A}$ gegebenen Spalten durch einen anderen virtuellen Prozessor sortiert werden. Es wird angenommen, daß die parallelen Aktivitäten mit Hilfe von `FORALL`-Schleifen generiert werden. Dies erlaubt eine übersichtliche, klare Darstellung der Programmstücke.

CC-UMA-Speicherkoppelung

Den Programmiermodellen mit CC-UMA-Speicherkoppelung liegt das Modell des gemeinsamen Speichers mit Cache-Kohärenz und einheitlicher Zugriffszeit zugrunde. Der Datenaustausch findet also über Programmvariablen statt, auf die jeder Prozeß in gleicher Weise zugreifen kann. Die Kommunikation erledigt sich dabei sozusagen von selber und muß vom Programmierer nicht extra berücksichtigt werden. Dementsprechend gibt es auch keine gesonderten Kommunikationsmechanismen. Das Programm für die Sortierung der Matrixspalten sieht in diesem Modell wie folgt aus:

```
VAR INT A[P], M[S][Z]
BEGIN

 <Vorberechnung>

 FORALL i = 1 TO P DO
  sort(M[A[i]);
 END
END
```

Es unterscheidet sich von einem sequentiellen Programm nur dadurch, daß anstelle einer sequentiellen eine parallele Schleife benutzt wird.

NCC-UMA

Das NCC-UMA Modell unterscheidet sich von dem CC-UMA-Modell darin, daß es keine Cache-Kohärenz garantiert. D.h., daß ein Schreibzugriff auf ein Datum, daß sich gleichzeitig in den Caches mehrerer Prozessoren befindet, nicht von allen Prozessoren gesehen wird. Das Problem wurde ausgiebig in 2.3.4 erläutert. Es kann, falls es bei der Programmierung nicht angemessen berücksichtigt wird, zu inkorrektem und indeterministischem Programmverhalten führen. Die Schwierigkeit bei der Bewältigung dieses Problems besteht darin, daß der Programmierer den Inhalt der einzelnen Caches nicht kennt. Die Cache-Verwaltung wird automatisch vom Prozessor bzw. seinem Cache-Kontroller durchgeführt. Eine korrekte und deterministische Programmausführung kann also nur gewährleistet werden, wenn bestimmte Variablen von der Zwischenspeicherung in den Caches ausgeschlossen werden. Dies geschieht in der Regel mit Hilfe eines speziellen Schlüsselwortes (z.B. `SHARED`) bei der Variablenvereinbarung.

Syntaktisch unterscheidet sich das NCC-UMA von dem CC-UMA Modell nur durch die Variablenvereinbarungen mit dem `SHARED`-Schlüsselwort. In dem Beispielprogramm müssen die Variablen `A` und `M` z.B. als

```
VAR SHARED INT A[P], M[S][Z]
```

deklariert werden. Die Formulierung der Programme wird also kaum schwieriger. Allerdings muß berücksichtigt werden, daß die Zugriffe auf `SHARED` Variablen im Durchschnitt wesentlich langsamer sind, als die Zugriffe auf normale Daten. Anders als bei Daten, die im Cache zwischengespeichert werden dürfen, erfahren diese Zugriffe jedes Mal die volle Hauptspeicherlatenz. Dies bedeutet zum einen, daß der Programmierer versuchen muß, so wenige Variablen wie möglich als `SHARED` zu deklarieren. Zum anderen kann es sinnvoll sein, Daten aus `SHARED` Variablen in normale Variablen zu kopieren. Nach einer solchen Kopieroperation können die Daten wieder in den Cache geladen werden. Wann eine solche Operation sinnvoll ist, hängt von der Speicherlatenz, der Cache-Größe und der Cache-Zeilenlänge des Systems ab. Effiziente Programmierung wird durch das NCC-Modell also nicht nur aufwendiger sonder auch systemabhängiger.

CC-NUMA-Speicherkoppelung

Wie in 2.3.3 beschrieben, wird beim NUMA-Speichermodell zwischen lokalen und globalen Daten unterschieden. Um dieser Unterscheidung Rechnung zu tragen gibt es bei NUMA-speichergekoppelten Programmiermodellen zwei Arten von Variablen: lokale und globale. Für die syntaktische und semantische Handhabung solcher Variablen wurde in verschiedenen Programmiersystemen eine Vielzahl von Konzepten vorgeschlagen. Die meisten basieren auf dem folgenden Ansatz:

1. Globale und lokale Variablen werden durch ein zusätzliches Schlüsselwort in der Vereinbarung unterschieden (z.B. `LOCAL` und `GLOBAL`).

2. Von jeder als lokal vereinbarten Variablen werden vom Compiler mehrere Kopien erzeugt, eine für jeden virtuellen Prozessor.

Ein Programm für die Sortierung der Matrixspalten kann im NUMA-speichergekoppelten Modell unter den obigen Annahmen wie folgt aussehen:

```
VAR GLOBAL INT A[P], M[S][Z]
    LOCAL  INT local_M[Z]
BEGIN

 <Vorberechnung>

 FORALL i = 1 TO P DO
   local_M=M[A[i]];
   sort(local_M);
   M[A[i]]=local_M;
 END
END
```

Syntaktisch gesehen, unterschiedet sich das Programm nur unwesentlich von der UMA-Version. Die Probleme des NUMA-Modells werden erst deutlich, wenn man sich mit der Effizienz beschäftigt. Dabei muß berücksichtigt werden, daß die Zugriffe auf die globale Variable `M` wesentlich langsamer als die Zugriffe auf das lokale Feld `local_M` sind. Wie in 2.3.3 erläutert, kann der Unterschied zwischen einem Faktor von 10 und 1000 liegen. Dies bedeutet, daß eine parallele Version bei einer naiven Implementierung mit vielen globalen Zugriffen sehr langsam, unter Umständen sogar langsamer als eine parallele Version wäre. Um die Auswirkungen der nicht-uniformen Zugriffszeit auf die Programmeffizienz zu verringern, gibt es zwei Möglichkeiten:

1. Globale Daten müssen in den lokalen Speicher kopiert werden, wenn ein Prozessor öfters auf sie zugreifen will. Aus diesem werden die Spalten vor dem Sortieren aus der globalen Matrix in die lokalen Variablen kopiert und danach wieder zurückgeschrieben.

2. Es muß versucht werden, Daten in den lokalen Variablen der virtuellen Prozessoren abzulegen, die auf sie am meisten zugreifen. Es muß also eine gute Datenverteilung gefunden werden. Dazu sind zum einen eine detaillierte Kenntnis des Speicherzugriffsmusters der gesamten Anwendung notwendig. Außerdem müssen die genauen Kosten für die globalen Zugriffe bekannt sein. Diese sind in der Regel nicht nur von Rechner zu Rechner, sondern oft auch von Zugriff zu Zugriff verschieden. Das Problem der Datenverteilung kann aufgrund obiger Probleme auch von erfahrenen Programmierern oft nur unbefriedigend gelöst werden. Die Gründe hierfür werden in Abschnitt 2.5.2 genauer erläutert.

Die Notwendigkeit, die obigen Punkte zu berücksichtigen, macht die Programmierung im CC-NUMA Model aufwendig und komplex. Außerdem müssen die Programme auf die Eigenschaften konkreter Architekturen hin optimiert werden.

NCC-NUMA

Beim NCC-NUMA-Modell wird für die, als global definierten Variablen, keine Cache-Kohärenz garantiert. Aus diesem Grund werden globale Variablen in der Regel nicht im Cache zwischengespeichert. Dies hat keine Auswirkung auf die Syntax des Programms. Allerdings ergibt sich dadurch ein zusätzlicher Faktor, der für eine effiziente Implementierung von Anwendungen berücksichtigt werden muß. So kann man beim CC-NUMA Modell davon ausgehen, daß eine globale Variable nach dem ersten Zugriff lokal im Cache vorhanden ist. Weitere, kurzfristig erfolgende Zugriffe sind also keine langsamen globalen Speicherzugriffe, sondern schnelle lokale Cache-Operationen. Dadurch bekommt eine gute Datenverteilung eine noch größere Bedeutung für die Programmeffizienz. Es ist außerdem, wie auch beim NCC-UMA Modell, sinnvoll, Daten aus globalen in lokale Variablen zu kopieren. Dadurch können sie anders als globale Variablen im Cache zwischengespeichert werden.

Nachrichtenkoppelung

Nachrichtengekoppelte Programmiermodelle gehen davon aus, daß die virtuellen Prozessoren keinen Zugriff auf gemeinsame Variablen besitzen. Für den Datenaustausch gibt es statt dessen spezielle Kommunikationsroutinen. Mit Hilfe solcher Routinen können Prozesse einander Nachrichten schicken und so Daten austauschen. Für die Syntax und Semantik von Kommunikationsroutinen gibt es viele Varianten. Gleiches gilt für die Deklaration und Sichtbarkeit von Variablen. Eine genau Diskussion dieser Varianten würde den Rahmen der vorliegenden Zusammenfassung sprengen. Sie ist z.B. in [178] zu finden. Um das Prinzip der nachrichtengekoppelten Programmiermodelle zu verdeutlichen, gehen wir im nachfolgenden Beispielprogramm von folgenden Annahmen aus:

1. Im Programm global definierter Variablen sind lokale Variablen des virtuellen Prozessor 0, und nur für diesen sichtbar. Dabei ist der Prozeß 0 für die Ausführung der sequentiellen Anweisungen zuständig.

2. Variablen, die für die anderen virtuellen Prozessoren sichtbar sind (sich also in derem Speichern befinden), müssen in den entsprechenden FORALL Anweisungen definiert werden.

3. Für die Kommunikation zwischen den virtuellen Prozessoren gibt es `send`- und `receive`-Routinen, die eine Variable an einen Prozessor senden bzw. empfangen können. Beide Prozeduren sind nicht-blockierend. Das Senden ist immer erfolgreich. Die `receive` Prozedur gibt, je nachdem, ob ein Datum eingetroffen ist oder nicht, `TRUE` bzw. `FALSE` zurück. Sie kann also zum Warten auf Daten in einer Schleife benutzt werden.

Unter den obigen Annahmen kann das Sortierprogramm für die Matrixspalten wie folgt formuliert werden:

```
VAR INT A[P], M[S][Z];

BEGIN

 <Vorberechnung>

 FOR i=1 TO P
  send(M[A[i]],i);
 END;
 FORALL i = 1 TO P DO
   VAR local_M[S];
     WHILE  NOT receive(local_M,0) DO END;
    sort(local_M);
    send(local_M,0);
  END
 FOR i=1 TO P
  WHILE NOT receive(M[A[i]],i); DO END;
 END;
END
```

Fazit

Eine abstrakte, systemunabhängige Programmierung ist lediglich mit Hilfe des CC-UMA-speichergekoppelten Modells möglich. Bei Verwendung der CC- bzw. NCC NUMA-Speicherkoppelung wird die syntaktische Formulierung des Programms zwar nicht wesentlich schwerer, es muß aber um der Effizienz Willen die Frage der Datenverteilung berücksichtigt werden. Dabei müssen in Abhängigkeit von den Systemeigenschaften und dem Programm die Daten so auf die lokalen Speicher verteilt werden, daß die Anzahl von globalen Zugriffen minimiert wird. Dies ist eine komplexe, nicht automatisierbare Aufgabe, die die Programmierung aufwendig und die Programme unportabel macht. Am aufwendigsten ist die Programmierung im nachrichtengekoppelten System. In diesem Modell ist, wie bei den NUMA-Systemen, eine Datenverteilung notwendig, die für möglichst wenige nicht-lokale Zugriffe sorgt. Hinzu kommt, daß alle nicht-lokalen Daten explizit von den Besitzer-Prozessoren geholt werden müssen. Dies macht die Programmierung sehr umständlich.

2.4.3 Synchronisation

Die meisten parallelen Algorithmen erfordern ein gewisses Maß an Synchronisation zwischen den virtuellen Prozessen. Hierfür gibt es in Programmiermodellen zwei Möglichkeiten: eine streng synchrone Ausführung von parallelen Anweisungssequenzen, sowie explizite Synchronisationsanweisungen.

Synchrone Ausführung

In manchen Programmiermodellen wird in Anlehnung an die SIMD Hardware-Architektur (2.2.1) bzw. das PRAM-Modell (2.1.1) eine Möglichkeit geboten, Anweisungen streng synchron auszuführen. Es handelt sich dabei in der Regel um Varianten der `FORALL` Anweisung, die oft `FORALL IN SYNC` genannt werden. Solche Anweisungen sind mit den, in Abschnitt 2.4.1 beschriebenen Anweisungen bis auf die Tatsache identisch, daß die parallele Ausführung der Anweisungen innerhalb des `FORALL`s strickt synchron vor sich geht.

Explizite Synchronisation

Bei Programmiermodellen, die keine synchrone Ausführung beinhalten, sind für die Synchronisation spezielle Anweisungen vorgesehen. Für die Funktionalität solcher Anweisungen wurden verschiedenen Möglichkeiten untersucht und realisiert. Am weitesten verbreitet sind Sperren (in der Literatur vom englischen her meist Lock genannt) und Barrieren. Ihre Funktion kann wie folgt zussamengefaßt werden:

Sperre: Eine Sperre dient dazu, den Zugang zu einem sog. kritischen Teil eines Programms zu sequentialisieren. Es ist eine Variable, auf der ein Faden zwei Operationen ausführen kann: sie in seinen Besitz nehmen

(`LOCK`) und sie wieder freigeben (`UNLOCK`). Das besondere an einer Sperre ist, daß zu jedem nur ein Faden der Besitzer sein kann. Wenn ein Faden im Besitz einer Sperre ist, dann schlägt jeder weitere `LOCK`-Aufruf auf diese Sperre fehl. Erst wenn der Besitzer sie durch eine `UNLOCK`-Anweisung freigegeben hat, kann ein anderer Faden zum Besitzer werden.

Barriere: Eine Barriere dient dazu, eine Menge von Fäden an einer Stelle im Programm zusammenzuführen. Sie läßt zwei Arten von Operationen zu: eine Initialisierung und die eigentliche `BARRIER`-Anweisung. Bei der Initialisierung wird der Barriere mitgeteilt, wieviele Fäden sie zusammenführen soll. Die `BARRIER`-Anweisung wird dort im Programm plaziert, wo sich die Fäden treffen sollen. Führt ein Faden diese Anweisung aus, so wird er zunächst blockiert. Er wird erst dann freigegeben, wenn die Anzahl der Fäden, die die `BARRIER`-Anweisung ausgeführt haben, gleich der, bei der Initialisierung bestimmten Faden-Anzahl ist.

Das besondere an den obigen Synchronisationsanweisungen ist, daß sie auf einem CC-UMA Speichersystem durch einfache Speicheroperationen effizient realisiert werden können.

2.4.4 Scheduling

Die Aufgabe des Schedulings besteht darin, die virtuellen Prozessoren des Programmiermodells den Prozessoren eines realen Parallelrechners zuzuordnen. Dabei müssen zwei Faktoren berücksichtigt werden

1. Es ist weder möglich noch sinnvoll, jedem virtuellen Prozessor einen echten Prozessor zuzuordnen. Dies liegt zum einen daran, daß die Anzahl virtueller Prozessoren die der realen Prozessoren übersteigen kann. Zum anderen ist es möglich, daß die Kosten für die Verteilung der virtuellen Prozessoren so hoch sind, daß sie den Gewinn einer voll parallelen Ausführung übersteigen. In diesem Fall ist es sinnvoll, mehrere Prozesse hintereinander auf einem echten Prozessor auszuführen.

2. Bei NUMA-Architekturen muß die Zuteilung der virtuellen Prozessoren mit der Datenverteilung abgestimmt werden.

Im Hinblick auf das Scheduling unterscheidet man Programmiermodelle mit automatischem Scheduling sowie solche, bei denen das Scheduling vom Programmierer durchgeführt werden muß. In beiden Fällen gilt, daß eine gute Verteilung der virtuellen auf die echten Prozessoren im allgemeinen nur auf CC-UMA-Rechnern möglich ist.

2.5 Das CC-UMA-Programmiermodel in der Praxis

Im vorigen Abschnitt wurden verschiedene Aspekte paralleler Programmiermodelle vorgestellt. Dabei wurde deutlich, daß das CC-UMA Speichermodell entscheidend ist, damit ein Programmiermodell eine einfache, portable und gleichzeitig effiziente Formulierung von Programmen erlaubt. Dieser Abschnitt beschäftigt sich mit den Problemen der Implementierung dieses Speichermodells auf den in Abschnitt 2.3 beschriebenen Hardware-Speicherarchitekturen. Das wichtigste Ergebnis dieses Abschnitts ist die Feststellung, daß eine effiziente Implementierung im allgemeinen nur dann möglich ist, wenn die Hardware bereits das CC-UMA Modell unterstützt. Eine Simulation des CC-UMA Modells auf anderen Speicherarchitekturen, z.B. NCC, NUMA oder gar auf getrenntem Adreßraum ist nur sehr eingeschränkt möglich.

Je nachdem, auf welcher Hardware das CC-UMA- Speichermodell realisiert werden soll, müssen der Übersetzer und das Laufzeitssystem eine oder mehrere der folgenden drei Aufgabe bewältigen

1. Im Falle eines getrennten Adreßraumes müssen Speicherzugriffe auf nicht-lokale Daten in geeignete Kommunikationsanweisungen übersetzt werden.

2. Bei NUMA-Speichersystemen müssen die Daten so auf die Speicherbänke verteilt werden, daß die Anzahl nicht-lokaler Zugriffe minimiert wird.

3. Bei NCC-Systemen muß die Cache-Kohärenz gewährleistet werden.

Wie im Nachfolgenden erläutert wird, kann die erste Aufgabe grundsätzlich ohne Probleme, die zweite in manchen Fälle und die dritte nur für seltene Spezialfälle effizient gelöst werden.

2.5.1 Simulation des gemeinsamen Adreßraumes

Bei einem Speichersystem mit verteiltem Adreßraum liegen die Daten in den Speichern der einzelnen Prozessoren und besitzen keine globalen Adressen. Will ein Prozessor auf ein nicht-lokales Datum zugreifen, so muß er eine Nachricht an den Besitzer dieses Datums schicken und auf eine Antwort warten. Um dem Programmierer unter diesen Umständen die Illusion eines gemeinsamen Speichers zu präsentieren, sind zwei Dinge notwendig. Zum einen muß ein virtueller globaler Adreßraum definiert werden. Dieser muß eine eindeutige Zuordnung von virtuellen globalen Adressen zu Prozessornummern und lokalen Adressen beinhalten. Zum anderen müssen alle Zugriffe auf globale Adressen in Kommunikationsanweisungen übersetzt werden. Beides kann ohne weiteres durch einen entsprechenden Übersetzer und Laufzeitsystem erledigt werden. Dies wurde von mehreren Systemen mit sog. virtuellem gemeinsamen Speicher bereits demonstriert (z.B. [25]).

2.5.2 Datenverteilung

Wie bereits beschrieben, hängt die Effizienz eines Programms auf einer Speicherarchitektur mit nicht-uniformer Zugriffszeit sehr stark von der Anzahl der globalen Speicherzugriffe ab. Um diese Anzahl zu minimieren, muß man dafür sorgen, daß die Daten so oft wie möglich in den lokalen Speichern der Prozessoren untergebracht werden, die sie für eine Berechnung benötigen. Da der Programmierer im UMA-Speichermodell keine Kenntnis von der Lage der Daten und den unterschiedlichen Zugriffszeiten hat, muß die Datenverteilung vom Übersetzer und dem Laufzeitsystem bewerkstelligt werden. Die Schwierigkeit dabei besteht darin, daß die Erstellung einer guten Datenverteilung eine detaillierte Kenntnis des Speicherzugriffsmusters der gesamten Anwendung erfordert. Bei einem so einfachen Problem wie z.B. der Vektoraddition, stellt dies kein Problem dar. Bei größeren Anwendungen hängt das Speicherzugriffsmuster aber in der Regel in einer komplexen Art und Weise von den Eingabedaten und dem Verlauf der Berechnung ab. Die automatische Generierung einer guten Datenverteilung ist ein aktuelles Forschungsthema. Dabei werden zwei Ansätze verfolgt: die statische Datenverteilung zur Übersetzungszeit auf der Basis des Quellcodes und eine Verteilung zur Laufzeit unter Berücksichtigung des Programms und der Daten. Wie im Nachfolgenden gezeigt, funktioniert jeder dieser Techniken für bestimmte Anwendungen. Es ist jedoch nicht möglich, ein Verfahren zu finden, daß für jedes Programm und jeden Datensatz automatisch ein gute Datenverteilung generiert.

Statische Datenverteilung

Bei der statischen Datenverteilung muß der Übersetzer versuchen, aus dem Quellcode herauszulesen, welche Prozessoren auf welche Daten zugreifen. Im Fall einer `FORALL`-basierten Verwaltung der Parallelität erfolgt dies mit Hilfe von drei Regeln:

1. Alle Zugriffe außerhalb von `FORALL`-Anweisungen werden vom Prozessor 0 (für den sequentiellen Teil verantwortlich) durchgeführt.

2. Variablen innerhalb einer `FORALL`-Anweisung, dessen Adresse nicht von der Laufvariablen des `FORALL`s abhängt, werden von allen Prozessoren benötigt.

3. Bei Variablen in einem `FORALL`, dessen Adresse von der Laufvariablen abhängt, wird die Nummer des zugreifenden Prozessors aus dem Ausdruck errechnet [17]. Bei solchen Variablen handelt es sich in der Regel um Felder, dessen Indexausdruck die Laufvariable beinhaltet.

Nachdem ermittelt wurde, welcher Prozessor auf welche Variablen zugreift, muß der Übersetzer eine optimale Plazierung der Daten errechnen. Problematisch sind dabei die Variablen, auf die mehrere Prozessoren zugreifen. Sie müssen den Prozessoren zugewiesen werden, die auf sie am meisten zugreifen bzw. für die die Zugriffskosten am höchsten sind. Dies stellt ein Optimierungsproblem dar, das mit Hilfe geeigneter Gleichungssysteme gelöst wird. Das Problem der statischen Datenverteilung besteht darin, daß sie nur in einfachen Spezialfällen möglich ist. Dies liegt an drei Dingen:

1. Es ist nicht immer möglich, aus dem Quellcode herauszulesen, welcher Prozessor auf ein Datum zugreifen muß. Das Problem ergibt sich, wenn in den Feldindizies außer den Prozeßindizies auch andere Variablen vorkommen. Es ist nämlich im allgemeinen nicht möglich, statisch aus dem Programmcode den Wert zu berechnen, den eine Variable zur Laufzeit annehmen wird. Ein Beispiel hierfür stellt das Beispielprogramm

aus Abschnitt 2.4.2 dar. Dort hing die Spalte, die ein Prozessor sortieren mußte, von dem Element eines Vektors ab. Unter der Annahme, daß dieser Vektor nicht als Konstante im Programm definiert wurde, sind seine Werte erst zur Laufzeit bekannt.

2. Die Anzahl der Zugriffe eines bestimmten Prozessors auf eine bestimmte Variable ist nicht immer aus dem Quellcode ersichtlich. Hier gilt die gleiche Argumentation wie bei den Feldindizies.

3. Die Gleichungen, die sich aus der Analyse der Feldindizies ergeben, sind nicht immer maschinell (also vom Übersetzer) lösbar.

Im allgemeinen funktioniert die statische Datenverteilung sehr gut für Anwendungen, dessen Speicherzugriffsmuster regelmäßig und statisch durch Konstanten festgelegt ist. Bei irregulären und dynamischen Problemen ist eine optimale statische Datenverteilung dagegen in der Regel nicht möglich.

Datenverteilung zur Laufzeit

Das Problem der statischen Datenverteilung besteht, wie oben beschrieben, darin, daß das Speicherzugriffsmuster von Variablenwerten abhängen kann, die erst zur Laufzeit bekannt sind. Dieses Problem kann durch eine dynamische Verteilung zur Laufzeit vermieden werden. Dabei werden zwei Strategien verfolgt: einfache dynamische Umverteilung (siehe z.B. [137]) und das Inspector-Executor-Verfahren (siehe z.B. [149])

Dynamische Umverteilung: Bei dynamischer Umverteilung werden die Daten, die statisch nicht gut plaziert werden konnten, zur Laufzeit umverteilt. Dies erfolgt, sobald die Variablenwerte verfügbar sind, die für die statische Datenverteilung fehlten. Das Problem dieses Verfahren besteht darin, daß der Aufwand für die Umverteilung sehr groß sein kann. Im ungünstigsten Fall kann er sogar so groß werden, daß das Programm durch die dynamische Umverteilung verlangsamt statt beschleunigt wird. Der Umverteilungsaufwand resultiert vor allem daraus, daß die für die Berechnung der optimalen Verteilung notwendige Information nicht immer zentral auf einem Prozessor verfügbar ist. Dies ist z.B. der Fall, wenn die Häufigkeit der Zugriffe der einzelnen Prozessoren auf ein bestimmtes Datum benötigt wird, wenn die neue Verteilung zu berechnen.

Inspector-Executor: Dieses Verfahren wurde entwickelt, um den Aufwand für die Datenverteilung während der Berechnung zu vermeiden. Es basiert auf der Idee, daß man die für die Datenverteilung notwendige Information durch einen kurzen Probelauf des Programms (Inspector-Lauf) bekommen kann. Ein solcher Probelauf wird mit den für die Berechnung vorgesehenen Eingabedaten vor dem echten Berechnungslauf (Executor-Lauf) gestartet. Es versucht an Hand einer extrem verkürzten Ausführung möglichst viele Informationen über das Speicherzugriffsmuster zu gewinnen. Diese Informationen werden dann im Berechnungslauf für die Datenverteilung benutzt. Das Problem des Inspector-Executor Verfahrens besteht in der Annahme, daß ein kurzer Probelauf repräsentativ für die gesamte Programmausführung ist. Diese Annahme trifft für viele Probleme mit regulärem Berechnungsmuster zu, versagt aber im allgemeinen bei irregulären Problemen.

Insgesamt gilt, daß zur Laufzeit für viele Anwendungen eine gute Datenverteilung gefunden werden kann, für die sich eine solche statisch nicht ermitteln läßt. Allerdings ist es auch zur Laufzeit nicht möglich, eine gute Datenverteilung für jedes Programm zu finden.

2.5.3 Simulation der Cache-Kohärenz

Wie in Abschnitt 2.3.4 dargelegt, erfordert die Erhaltung der Cache-Kohärenz, daß jeder Schreibzugriff zu allen den Caches propagiert wird, in deen sich eine Kopie des betroffenen Datums befindet. Hierzu ist es einerseits notwendig, daß nach jedem Schreibzugriff eine entsprechende Nachricht an alle Caches verschickt wird. Zum anderen müssen diese Nachrichten von den Caches empfangen werden, woraufhin der Cache-Inhalt überprüft, und bei Bedarf aufgefrischt werden muß. Ersteres kann ohne weiteres auf einem System ohne Cache-Kohärenz Unterstützung vom Laufzeitsystem bewerkstelligt werden. Allerdings ist die Überprüfung und Auffrischung des Caches ohne eine Unterstützung durch die Hardware nicht möglich. Daher besteht die einzige Möglichkeit für die Emulation eines CC-Modells auf einer NCC Hardware darin, alle Zugriffe auf, von mehreren Prozessoren genutzte Variablen vom Cache auszuschließen. Dies würde aber, wie bereits erläutert (siehe 2.3.4), zu einem großen Effizienzverlust führen.

2.6 Symmetrische Multiprozessoren

Die bisherige Betrachtung hat gezeigt, daß das CC-UMA Speichermodell die Voraussetzung für eine einfache, portable Programmierbarkeit von Parallelrechnern darstellt. Gleichzeitig wurde deutlich, daß ein solches Modell nur dann effizient implementiert werden kann, wenn es von der Hardware des Speichersystems zur Verfügung gestellt wird. Es wurde außerdem gezeigt, daß eine asynchrone, dem MIMD-Modell entsprechende Programmausführung mit effizienten Synchronisationsoperationen für die meisten Anwendungen von Vorteil ist. Rechner die dieser Beschreibung entsprechen, werden oft als *symmetrische Multiprozessoren* kurz *SMP*s bezeichnet. Diese Bezeichnung soll unterstreichen, daß alle Prozessoren eines solchen Rechners gleichberechtigt auf alle Systemresourcen, also auch auf den Hauptspeicher, zugreifen können.

Dank ihrer gute Programmierbarkeit und Effizienz haben die SMPs als einzige parallele Rechnerklasse eine hohe Akzeptanz im kommerziellen Bereich gefunden. Leider können sie heute auf Grund ihrer beschränkten Skalierbarkeit nur in einigen wenigen Anwendungsbereichen eingetzt werden. Um welche Bereiche es sich dabei handelt und welche Vorteile von der Erweiterung ihres Einsatzgebietes zu erwarten sind wird im Nachfolgenden zusammengefaßt.

2.6.1 Stand der Technik beim Einsatz von SMPs

Die Einsatzgebiete von Parallelrechnern im allgemeinen liegen heute in den folgenden Gebieten:

Multiprozessor PCs: Durch den Preisverfall im Bereich der Mikroprozessoren finden Multiprozessoren auch zunehmend in PCs Einzug. Dabei handelt es sich um Maschinen mit maximal 2 bis 4 Prozessoren.

Multiprozessor Arbeitsplatzrechner: Auch bei Arbeitsplatzrechnern haben die sinkenden Prozessorpreise zusammen mit steigenden Rechenleistungs-Anforderungen parallele Architekturen motiviert. Arbeitsplatzrechner werden heute mit bis zu 16 Prozessoren ausgestattet.

Multiprozessor Server: Server stellen eine Zwischenstufe zwischen Arbeitsplatzrechnern und Superrechnern dar. Sie werden vor allem für große Datenbanken, WWW-Server und teilweise auch für numerische Berechnungen eingesetzt. Sie werden heute fast ausschließlich als Parallelrechner mit 8 bis 64 Prozessoren implementiert.

Parallele Superrechner: Bei den Superrechnern ist in letzer Zeit die Parallelverarbeitung zum Standard geworden. Dabei sind Rechner mit bis zu 1024 Prozessoren gebaut worden []. Bei Superrechnern steht vor allem die Rechenleistung im Vordergrund. Die Programmierbarkeit und Kosten sind zweitrangig.

PC- und Arbeitsplatzrechner-Cluster: Einer der am schnellsten wachsenden Bereiche der Parallelverarbeitung stellt das Cluster-Computing dar. Dabei werden gewöhnliche Arbeitsplatzrechner mit Hochleistungs-Netzwerkkarten verbunden und können wie ein Parallelrechner benutzt werden.

Von den obigen fünf Bereichen werden symmetrische Multiprozessoren heute nur in den ersten beiden eingesetzt. Bei den Servern und parallelen Superrechnern scheitert die Implementierung an der hohen Prozessorzahl. Die Koppelung von Arbeitsplatzrechnern zu einem symmetrischen Multiprozessor ist wegen der großen Systemausdehnung nicht möglich, da dies die für einen symmetrischen Multiprozessor notwendige Netzwerkleistung nicht zuläßt. Die Hürden bei der Realisierung symmetrischer Multiprozessoren für hohe Prozessorzahlen und große Systemausdehnungen werden im nächsten Kapitel beschrieben.

2.6.2 Motivation für ein breiteres Einsatzgebiet

Durch die Erweiterung des Einsatzgebietes symmetrischer Prozessoren können drei Dinge erreicht werden:

1. Die Programmierung von Servern, Superrechnern und Arbeitsplatzrechner-Clustern wird einfacher, da auf solchen Maschinen ein CC-UMA basiertes Programmiermodell implementiert werden kann.

2. Es wird ein wesentlich breiterer Kreis von Anwendungen effizient auf großen Parallelrechnern und Arbeitsplatzrechner-Clustern parallelisiert werden können, da kommunikationsintensive Anwendungen nicht mehr durch die hohen Kosten globaler Datenzugriffe verlangsamt werden.

3. Programme werden zwischen den verschiedenen parallelen Rechnerklassen portabel. Dadurch wird vor allem die Entwicklung von parallelen Programmen vereinfacht. So wird es möglich, Anwendungen auf einem Arbeitsplatzrechner oder gar PC zu entwickeln und mit kleinen Datensätzen zu testen, um sie dann unverändert auf einem Server, einem Supercomputer oder einem großen Arbeitsplatzrechner-Cluster laufen zu lassen.

Kapitel 3

Das Speichersystem symmetrischer Multiprozessoren

Das vorliegende Kapitel beschreibt den Zusammenhang zwischen der Architektur des Speichersystems, und der Skalierbarkeit heutiger symmetrischer Multiprozessoren. Es zeigt, warum es bisher nicht gelungen ist, die SMP-Architektur für große Prozessorzahlen zu realisieren. Damit wird das Problem definiert, das die vorliegende Arbeit zu lösen versucht.

Im Abschnitt 3.1 werden zunächst einige Grundüberlegungen zu der Architektur und Leistung heutiger Speichersysteme erläutert. Da die Speichersysteme von SMPs auf sequentiellen Speichersystemen basieren und mit ihrer Leistungsfähigkeit verglichen werden, werden im Abschnitt 3.2 die wichtigsten Aspekte sequentieller Speichersysteme beschrieben. Danach werden in den Abschnitt 3.3 und 3.4 die beiden wichtigsten Aspekte der Speichersysteme symmetrischer Multiprozessoren beschrieben: das Cache-Kohärenz Protokoll und die Architektur des Verbindungsnetzwerk erörtert. Darauf aufbauend werden im Abschnitt 3.5 die Gründe für die schlechte Skalierbarkeit der SMP-Speichersysteme zusammengefaßt. Abschließend wird ein Überblick über den Stand der Technik bei parallelen Architekturen mit cache kohärenten gemeinsamen Speicher gegeben. Dabei wird gezeigt, wie sich die vorliegende Arbeit von verwandten Arbeiten in diesen Bereichen unterscheidet.

3.1 Grundüberlegungen

Die Architektur heutiger Speichersysteme basiert auf drei Grundannahmen:

1. Die Latenz eines Speichers nimmt mit seiner Größe stark zu. Es ist nur möglich, kleine schnelle Speicher oder große langsame Speicher zu bauen.

2. Speichersysteme können trotzt hoher Latenz der Speicherbausteine eine hohe Bandbreite besitzen. Beim Zugriffen auf größere, zusammenhängende Datenblöcke fällt daher die hohe Latenz nur beim Zugriff auf das erste Element an. Auf die nachfolgenden Elemente kann wesentlich schneller zugegriffen werden.

3. Die Speicherzugriffe sind nur bei den wenigsten Programmen zufällig und gleichmäßig auf die Speicheradressen verteilt. Zeitlich kurz hintereinander liegende Zugriffe beziehen sich oft auf im Speicher benachbarte Daten. Man spricht in diesem Zusammenhang von räumlicher Lokalität der Zugriffe. Sie ist darauf zurückzuführen, daß die vom Programm benötigten Daten in der Regel in einer geordneten Form (z.B. als lineares Feld) im Speicher vorliegen.

4. In den meisten Programmen werden immer wieder die gleichen Speicherstellen genutzt. Dieses Verhalten wird als temoporale Lokalität der Speicherzugriffe bezeichnet. Sie hat ihre Ursachen in der Tatsache, daß die meisten Programme den Großteil der Rechenzeit in Schleifen verbringen. Dies bedeutet, daß immer wieder der gleiche Teil des Programmcodes ausgeführt wird und immer wieder auf die gleichen Variablen zugegriffen wird.

Obige Überlegungen gelten sowohl für sequentielle als auch für parallele Speichersysteme, also auch für symmetrische Multiprozessoren.

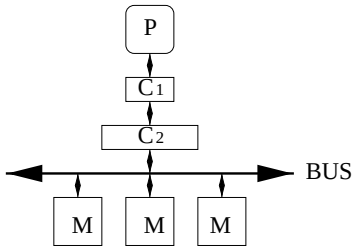


Abbildung 3.1: Schematische Darstellung des Speichersystems eines sequentiellen Rechners bestehend aus einem Prozessor (P), zwei Cache-Stufen (C_1 und C_2), und drei Speicherbänken (M) des Hauptspeichers.

Wie im vorigen Kapitel angesprochen, haben sie zu einer hierarchischen Speicherarchitektur geführt. Die unterste Stufe dieser Hierarchie bildet der Hauptspeicher, der groß genug für alle Daten ist, dafür aber verhältnismäßig langsam ist. Er besteht in der Regel aus mehreren Speicherbänken, die über ein Verbindungsnetzwerk (in der Regel einen Bus) mit dem restlichen Speichersystem verbunden ist. Dieser besteht aus mehreren Cache-Stufe, wobei die Speicherkapazität von Stufe zu Stufe abnimmt, während die Zugriffsgeschwindigkeit zunimmt. Diese Speicherarchitektur is schematisch in Abbildung 3.1 dargestellt. Sie verringert die durchschnittliche Speicherzugriffszeit indem sie die Nutzung der hohen Speicherbandbreite und der Lokalität der Zugriffe unterstützt. Ersteres wird dadurch erreicht, daß bei jedem Lesezugriff große Datenblöcke (Cache-Zeilen) auf einmal gelesen und im Cache für nachfolgende schnelle Zugriffe zwischengespeichert werden. Dabei macht man sich die räumliche Lokalität der Speicherzugriffe zu nutze. Um die temporare Lokalität auszunutzen werden die Daten nach dem Zugriff nicht sofort aus dem Cache entfernt. Will der Prozessor kurz darauf nochmals auf diese Daten zugreifen, so müssen sie nicht mehr aus dem Hauptspeicher geholt werden.

3.1.1 Speichersysteme symmetrischer Multiprozessoren

Auf die Verwendung von Caches kann auch in Parallelrechner nicht verzichtet werden. Die Probleme und Speicherarchitektur-Varianten, die sich daraus ergeben wurden im vorigen Kapitel ausführlich beschrieben. Wie in 2.6 erläutert zeichnen sich symmetrische Multiprozessoren durch einen cache-kohärenten gemeinsamen Speicher mit uniformer Zugriffszeit (CC-UMA-GS Speichermodel). Dies bedeutet, daß die Prozessoren mit ihren Caches so an die Speicherbänke angeschlossen werden müssen, daß

1. zur Erhaltung der Cache-Kohärenz jeder Schreibzugriff auf gemeinsam genutzte Daten nicht nur im Hauptspeicher, sondern auch in den Caches aller Prozessoren durchgeführt, die eine Kopie des betroffenen Datums besitzen (siehe 2.3.4)

2. zur Gewährleistung der uniformen Zugriffszeit die Latenz der Speicherzugriffe weitgehend unabhängig von der Lage des Datums im Speicher und dem Zugriffsmuster der Prozessoren sein muß

3. die Latenz der Speicherzugriffe vergleichbar mit der Latenz der Speicherzugriffe in einem sequentiellen Rechner sein muß.

Für die Implementierung des Speichersystems eines SMPs sind also zwei Dinge notwendig. Zum einen muß ein Speicherzugriffs-Protokoll definiert werden, das für die Fortpflanzung der Schreibzugriffe zu den betroffenen Caches sorgt. Ein solches Protokoll wird als Cache-Köhärenz Protokoll bezeichnet. Zum anderen wird für die Verbindung der Prozessoren und der Speicherbänke eine leistungsfähiges Netzwerk benötigt. Dieses Netzwerk muß trotz der Zusatzbelastung durch mehrere Prozessoren und das Cache-Kohärenz-Protokoll ungefähr die gleiche Latenz wie der Speicherbus eines sequentiellen Rechners besitzen. Die Fragen des Cache Kohärenz Protokolls und der Effizienz des Verbindungsnetzwerks werden in den Abschnitten 3.3 und 3.4 genauer erläutert.

3.2 Speichersysteme sequentieller Prozessoren

Das Speichersystem heutiger sequentieller Prozessoren ist in der Regel, in drei Hierarchiestufen aufgebaut.

1. Auf dem Prozessorchip befinden sich 64-256KByte L1-Cache, die die erste Stufe der Speicherhierarchie darstellen.
2. Die zweite Stufe bildet ein externer L2-Cache, der zwischen 0.5 bis 4MB groß sein kann.
3. Die untere Speicherstufe bildet der DRAM-Hauptspeicher der heute eine Größe von 32 bis 1024 MByte haben kann.

Für die Verwaltung der beiden Caches ist ein Cache-Kontroller zuständig, der sich meistens auf einem separaten Chip befindet. Er ist durch ein Verbindungsnetzwerk mit dem Hauptspeicher verbunden, der in der Regel aus 2 bis 8 Speicherbänken besteht. An den Speicher sind oft auch Ein/Ausgabe-Einheiten angeschlossen, die gleichberechtigt mit dem Cache-Kontroller auf die Daten zugreifen können.

Im Nachfolgenden werden zunächst die wichtigsten Möglichkeiten für die Realisierung der Cache-Stufen und des Verbindungsnetzwerks zusammengefaßt. Abschließend wird die Latenz der heutigen sequentiellen Speichersysteme erötert.

3.2.1 Cache-Architektur

Ein Cache-Speicher besteht aus einer Reihe von Cache-Zeilen, die jeweils in einen sog. Attribut und einen Datenteil unterteilt sind. Der Datenteil beinhaltet eine Folge von Datenwörten, der Attribut die Speicheradresse der Folge sowie zusätzliche Verwaltungsinformation. Die wichtigsten Faktoren in einem Cache-Design sind die Zuordnung von Speicheradressen zu Cache-Zeilen, die Strategie bei der Ersetzung von alten Cache-Zeilen und die Frage der Länge der Cache-Zeilen. Hinzu kommen die Vorgehensweise beim Propagieren von Schreibzugriffen innerhalb der Speicherhierarchie und die Interaktion zwischen dem L1- und dem L2-Cache.

Adressierung

Da ein Cache wesentlich kleiner als der Hauptspeicher ist, gibt es zwischen den Speicheradressen und den Cache-Zeilen keine 1:1 Korrespondenz. Es wird daher ein Verfahren benötigt, das beim Laden eines Datums in den Cache eine Zuordnung der Adresse zu einer Speicher-Zeile vornimmt. Hierfür gibt es drei Möglichkeiten: ein direktes Abbildungsverfahren ein Assoziativverfahren oder ein gemischtes Verfahren.

Direkte Caches: Beim direkten Abbildungsverfahren ergibt sich die Adresse eines Datums im Cache aus einem Teil seiner Speicheradresse. Bei einer b_m langen Speicheradresse und 2^{b_c} Cache-Zeilen benutzt man normalerweise die niederwertigen b_c Bits der Speicheradresse für die Cache-Zeilen Zuordnung. Demnach würde z.B. ein Wort mit der binär kodierten Speicheradresse 100001 in einem Cache mit $8 = 2^3$ Zeilen in die Zeile 001 geladen werden. Der hochwertige Teil der Adresse (100) würde dann in dem Attribut gespeichert werden. Er würde dann bei einem Zugriff auf den Cache mit dem hochwertigen Teil der gewünschten Adresse verglichen werden, um festzustellen, ob sich das gesuchte Datum im Cache befindet. Der Nachteil des obigen Verfahrens besteht in einer schlechten Ausnutzung der Cache-Kapazität. Durch die eindeutige Zuordnung einer jeder Speicheradresse zu einer Cache-Zeile kann es vorkommen, daß ein Datum in eine bereits besetzte Zeile geladen wird, obwohl es genug andere, freie Zeilen gibt. Die Vorteile des direkten Cache sind der geringe Imlpementierungsaufwand und ein schneller Zugriff.

Assoziative Caches: Bei assoziativen Caches kann jede Speicheradresse in jede Cache-Zeile geladen werden. Das Auffinden von Daten im Cache erfolgt nach dem assoziativen Speicherprinzip: die gesuchte Adresse wird parallel mit den Attributen aller Cache-Zeilen verglichen. Der Vorteil des assoziativen Verfahrens besteht in der optimalen Nutzung der Cache-Kapazität. Leider erfordert die Implementierung der assoziativen Suche einen sehr hohen Hardwareaufwand. Aus diesem Grund wird das assoziative Verfahren nur bei kleinen Caches angewendet.

Set-Assoziative Caches: Set-assoziative Caches sind eine Kombination aus den beiden obigen Verfahren. Dabei kann ein Datum zwar nicht in eine beliebige Zeile geladen werden, es ist aber nicht mehr auf eine einzige beschränkt. Jeder Speicheradresse werden mehrere Zeilen zugeordnet, die einen 'Set' bilden. Innerhalb des

Sets wird das gesuchte Datum nach dem Assoziativverfahren verwaltet. Set-assoziative Caches haben den Vorteil einer recht guten Cache-Ausnutzung bei vertretbarem Implementierungsaufwand.

Ersetzungsstrategie

Soll ein neues Datum in den Cache aufgenommen werden, so muß in der Regel ein anderes Datum dafür Platz machen. Während bei direkten Caches die zu ersetzende Zeile durch die Adresse bestimmt ist, muß bei assoziativen und gemischten Caches eine Auswahl getroffen werden. Um die temporale Lokalität möglichst gut zu nutzen, sollte man dabei ein Datum wählen, das mit hoher Wahrscheinlichkeit in nächster Zeit nicht benötigt wird. Hierfür wird in der Regel das sog. LRU (für least recently used) Verfahren verwendet. Dabei wird im Attribut einer jeden Cache-Zeile der Zeitpunkt des letzten Zugriffs durch einen Zeitstempel vermerkt. Für die Ersetzung wird dann immer die Zeile mit dem ältesten Zeitstempel genommen. Aus diesem Grund werden oft auch ein zufälliges Ersetzungsverfahren (RAND) oder ein FIFO-Verfahren verwendet.

Cache-Zeilenlänge

Die Frage der Cache-Zeilenlänge ist entscheidend für die Nutzung der räumlichen Datenlokalität. Je länger die Cache-Zeile, um so größer ist die Wahrscheinlichkeit, daß eines ihrer Elemente bei einem späteren Zugriff wiederverwendet wird. Andererseits dürfen die Cache-Zeilen auch nicht zu lang werden. Zum einem würde bei einer beschränkten Cache-Größe eine zu hohe Cache-Zeilenlänge die Anzahl der Cache-Zeilen zu stark reduzieren. Dadurch könnten nur wenige unterschiedliche Adressen im Cache untergebracht werden, was wiederum die Miss-Rate erhöhen würde. Hinzu kommt, daß mit steigender Cache-Zeilenlänge die Anforderungen an die Bandbreite des Netzwerks und des Hauptspeichers wachsen.

Schreibstrategie

Bei einem Schreibzugriff müssen sowohl der Inhalt der Cache-Zeile als auch aller darunterliegenden Stufen der Speicherhierarchie (weitere Caches bzw. der Hauptspeicher) aktualisiert werden. Passiert dies sofort im Anschluß an einen Schreibzugriff, so spricht man von einem durchschreibenden Cache. Alternativ kann die Aktualisierung solange verschoben werden, bis das Datum aus dem Cache entfernt wird. Ein solches sog. zurückschreibendes Verfahren hat bei wiederholten Schreibzugriffen den Vorteil, daß die Anzahl der Zugriffe auf die darunterliegende Speicherebenen reduziert wird. Andererseits bringt es aber auch einen höheren Verwaltungsaufwand mit sich. So muß dem Attribut jeder Cache-Zeile ein zusätzliches Zustandsbit hinzugefügt werden, das nach einem Schreibzugriff gesetzt wird. Es zeigt an, daß die Cache-Zeile modifiziert wurde und beim Entfernen aus dem Cache zurückgeschrieben werden muß.

Relation zwischen den verschiedenen Cache-Stufen

Jedes Datum, das sich in einem Cache befindet, ist trivialer Weise auch irgendwo im Hauptspeicher vorhanden. Gleiches kann, muß aber nicht für L1- und L2-Caches gelten. Im ersten Fall spricht man von einer logischen Inklusion. Die logische Inklusion vereinfacht viele Verwaltungsaufgaben, insbesondere im Bereich der Multiprozessoren. Ihr Nachteil besteht darin, daß sie eine Angleichung der Struktur und der Verwaltungsprotokolle der beiden Caches verlangt.

3.2.2 Die Verbindungsstruktur

Bei dem Verbindungssystem ist vor allem die Verbindung zwischen dem Cache-Kontroller, dem Hauptspeicher und den Ein/Ausgabe-Einheiten interessant. Die anderen Verbindungen sind einfache Punkt-zu-Punkt die keiner weiteren Erläuterung bedürfen.

Für die Verbindung zwischen dem Cache-Kontroller und dem Hauptspeicher wird in sequentiellen Rechnern meistens ein sog. leitungsvermittelnder Bus verwendet.

Leitungsvermittelnder Bus

Ein Bus besteht aus einem parallelen Datenkanal, und einer Menge von Kontrolleitungen. Der Datenkanal besitzt in den heutigen Bussen 64 bis 256 Leitungen die mit einer Frequenz zwischen 66 und 100 MHz betrieben werden.

Die Funktion und Anzahl der Kontrollleitung ist von System zu System unterschiedlich. In den meisten Systemen gibt es fünf Arten solcher Leitungen:

1. Eine CLOCK-Leitung auf der ein Taktsignal zur Synchronisation der Datenübertragung gesendet wird.
2. Eine STROBE bzw. SUPPLY genannte Leitung, die die Übertragung gültiger Daten anzeigt.
3. Eine BUSY-Leitung, die immer dann gesetzt ist, wenn der Bus belegt ist.
4. Eine REQUEST und eine GRANT-Leitung für jede Einheit, die über den Bus auf die Speicherbänke zugreifen kann. Über diese Leitungen wird der Zugriff auf den Bus und die Speicherbänke koordiniert.

Die Busleitungen sind über Schnittstellenbausteine an eine Menge von Knoten (Prozessor, Speicherbänke und Ein/Ausgabeeinheiten) und einen Buskontroller angeschlossen. Letzterer hat die Aufgabe den Zugriff auf den Bus zwischen dem Cache-Kontroller und der Ein/Ausgabeeinheit zu koordinieren. Ein Zugriff auf eine an den Bus angeschlossene Speicherbank, der oft als Bustransaktion bezeichnet wird, läuft in einem leitungsvermittelnden Bus wie folgt ab:

1. Falls ein Busknoten (Prozessor bzw. die Ein/Ausgabeeinheit) auf eine Speicherbank zugreifen möchten, dann signalisiert er dies mit Hilfe seiner REQUEST-Leitung.
2. Sobald der Bus frei ist, erlaubt der Buskontroller den Zugriff und teilt dies dem Knoten durch Setzen seiner GRANT-Leitung mit. Der Knoten wird dadurch zum Besitzer des Busses. Ab diesem Zeitpunkt bis zum Ende der Transaktion dürfen nur der Besitzer und die von ihm angesprochene Speicherbank auf den Bus zugreifen. Der Bus wird also für die Dauer der Transaktion zu einer dedizierten Leitung zwischen dem Besitzer und der Speicherbank (daher die Bezeichnung 'leitungsvermittelter' Bus).
3. Der Besitzer setzt die BUSY-Leitung, schickt die Anfrage an die Speicherbank, und wartet auf die Antwort. Während der Wartezeit gilt der Bus als besetzt.
4. Sobald die Speicherbank die Anfrage bearbeitet hat, schickt sie die Antwort über den Bus.
5. Nach dem Empfang der Antwort setzt der Besitzer die BUSY-Leitung auf 0 und gibt den Bus damit frei.

3.2.3 Latenz heutiger Speichersysteme

Bei der Betrachtung der Latenz von Speichersystemen muß zwischen der Latenz einzelner Stufen der Speicherherarchie und der durchschnittlichen Gesamtlatenz unterschieden werden.

Latenz der einzelnen Stufen des Speichersystems

Der L1-Caches ist in der Regel auf dem Prozessorchip integriert und kann mit der Prozessorfrequenz betrieben werden. Seine Latenz beträgt daher 1 bis 2 Prozessorzyklen. Der L2-Cache befinden sich meistens auf einem anderen Chip. Durch die damit verbundene höhere Verbindungslatenz liegt seine Latenz bei 4 bis 8 Prozessorzyklen. Die Speicherbänke selbst haben eine Latenz von 20 bis 50 Zyklen. Hinzu kommt die Verzögerung durch den Speicherbus, die im Bereich mehrerer Hundert Nanosekunden liegt. Insgesamt summiert sich damit die Hauptspeicherlatenz auf ca. 100 bis 200 Prozessorzyklen.

Durchschnittliche Gesamtlatenz eines Speichersystems

Die großen Unterschiede in den Zugriffszeiten der einzelnen Speicherhierarchie-Stufen führen dazu, daß die Effizienz eines Programms stark von der Verteilung der Speicherzugriffe auf die Caches und den Hauptspeicher abhängt. Ein Programm, dessen Speicherzugriffe sich allein auf den L1-Cache beschränken, läuft wesentlich schneller als eines, das überwiegend auf den Hauptspeicher zugreift. Um diese Abhängigkeit zu berücksichtigen, betrachtet man zur Charakterisierung eines Speichersystems in der Regel die durchschnittliche Speicherlatenz t_{av}. Sie ist definiert als eine durch sog. Hit-Raten h_{L1} und h_{L2} gewichtete Summe aus den Cache- und Hauptspeicherlatenzen t_{L1} (L1-Cache), t_{L2} (L2-Cache) und t_{RAM} (Hauptspeicher).

$$t_{av} = h_{L1} * t_{L1} + h_{L2} * t_{L2} + (1 \Leftrightarrow p_{L1} \Leftrightarrow p_{L2}) * t_{RAM} \tag{3.1}$$

Dabei geben die Hit-Raten die Wahrscheinlichkeiten an, daß ein Datum bei einem Zugriff im L1-Cache (h_{L1}) oder im L2-Cache (h_{L2}) gefunden wird. Die Maximierung der Hit-Raten für ein möglichst breites Spektrum von Programmen ist eines der Hauptanliegen beim Entwurf von Speichersystemen. Es erfordert eine Anpassung der Cache-Struktur und der Verwaltungsstrategie an das Speicherzugriffsverhalten von möglichst vielen Anwendungen. Bei den meisten heutigen Systeme liegt die Hitrate für typische Anwendungen für den den L1-Cache zwischen 0.6 und 0.9 und für den L2-Cache zwischen 0.90 und 0.99.

3.3 Cache-Kohärenz Protokolle

Für die Erhaltung der Cache-Kohärenz ist es notwendig daß bei jedem Schreibzugriff:

1. möglichst schnell ermittelt wird, welche Prozessoren von dem Zugriff betroffen sind,
2. diese Prozessoren von dem Schreibzugriff in Kenntnis gesetzt werden und
3. die Daten in den Caches dieser Prozessoren auf den neusten Stand gebracht werden.

Je nach dem wie ein cache-Kohärenz Protokoll mit den ersten beiden Anforderungen umgeht, unterscheidet man zwischen sog. snoopy [12]und verzeichnis [22] -Protokollen:

Snoopy-Verfahren: Bei Snoopy-Verfahren wird jeder Schreibzugriff grundsätzlich an alle Prozessoren durch einen Rundruf verschickt. Die Cache-Kontroller der Prozessoren suchen sich dann selbst die Daten heraus, die in ihrem Cache enthalten sind, und führen die zur Erhaltung der Cache-Kohärenz erforderlichen Modifikationen durch. Der Vorteil dieser Protokolle besteht darin daß sie mit einem geringen Verwaltungsaufwand verbunden sind und daher sehr effizient sind.

Verzeichnis-Verfahren: Bei Verzeichnis-Protokollen wird in einem Verzeichnis der Verbleib einer jeden Cache-Zeile protokolliert. Die Schreibzugriffe können dann gezielt nur an die Besitzer der betroffenen Zeile geschickt werden. Verzeichnis-Protokolle haben den Vorteil, daß das Verbindungsnetzwerk und die Cache-Kontroller nicht mit unnötigen Nachrichten belasten. Dafür erfordern sie einen wesentlich höheren Verwaltungsaufwand als Snoopy-Protokolle und führen dadurch zu einer höheren Speicherlatenz.

Sowohl bei Snoopy- als auch bei Verzeichnis-Protokollen kann im Hinblick auf den Zeitpunkt der Aktualisierung der Daten zwischen Aktualisierungs- und Invalidierungsverfahren unterschieden werden.

Aktualisierungsverfahren Bei Auktualisierungsverfahren wird der neue geschriebene Wert in allen betroffenen Caches aktualisiert. Der Vorteil dieses Verfahrens besteht darin, daß alle Prozessoren sofort über den aktuellen Wert verfügen und nicht mehr auf den Hauptspeicher zugreifen müssen.

Invalidierungsverfahren: Beim Invalidierungsverfahren wird nur das Datum im Cache des schreibenden Prozessors bzw. im Hauptspeicher tatsächlich geschrieben. In allen anderen Caches wird es lediglich als ungültig markiert. Der neue Wert wird erst bei einem erneuten Zugriff geholt. Der Vorteil des Invalidierungsverfahrens besteht darin, daß nicht die gesamte Cache-Zeile, sondern nur ihre Adresse zu den Prozessoren propagiert werden muß. Da eine Adresse 64 Bit lang ist, während eine Cache-Zeile bis zu 128 Byte, wird so für das Cache-Kohärenz-Protokoll deutlich weniger Bandbreite benötigt.

Die Frage der Cache-Kohärenz ist ein wichtiges Thema in der Parallelrechner-Forschung. Es wurde daher eine Vielzahl von verschiedenen Protokollen entwickelt und untersucht. Im Nachfolgenden wird ein Überblick über die wichtigsten dieser Protokolle gegeben Dabei werden die Probleme bei der Skalierbarkeit und der Effizienz unterstrichen.

3.3.1 Snoopy-Protokolle

Bei Snoopy-Protokollen wird wie beschrieben jede Schreiboperation per Rundruf an alle Prozessoren verschickt. In der Praxis wird hierzu in der regel ein Bus verwendet, an dem alle Prozessoren angeschlossen sind. Daher werden Snoopy-Protokolle oft auch als Bus-Protokolle bezeichnet.

Die einfachsten Snoopy-Protokolle sind das direkte Aktualisierungsverfahren. und das sog. WTI (write through invalidate) Protokoll. Beide gehen von einem durchschreibenden Cache aus und teilen ohne Ausnahmen immer

alle Schreibzugriffe dem gesamten System mit. Beim direkten Aktualisierungsverfahren werden die Cache-Zeilen der betroffenen Prozessoren aktualisiert und können sofort weiterverwendet werden. Beim WTI-Protokoll werden sie lediglich als ungültig markiert und müssen beim nächsten Zugriff neu aus dem Speicher geholt werden.

Der große Vorteil von Snoopy-Protokollen besteht darin, daß sie nur einen sehr geringen Verwaltungsaufwand erfordern. So ist es möglich, die Cache-Kohärenz zu gewährleisten, und gleichzeitig eine, mit sequentiellen Speichersystemen vergleichbare, Speicherlatenz zu wahren. Hinzu kommt, daß sie optimal für die Implementierung mit Hilfe eines Busses geeignet sind. Aus diesen Gründen werden zur Zeit in fast allen SMPs Snoopy-Protokolle eingesetzt. Leider steigen bei Snoopy-Protokollen mit steigender Prozessorzahl und Taktrate die Anforderungen an den Speicherbus, an die Netzwerkschnittstellen und an die Cache-Kontroller sehr rapide an. Von besonderer Bedeutung sind dabei vor allem drei Dinge:

1. Die Anzahl der Schreibzugriffe im System steigt linear mit der Anzahl der Prozessoren. Somit steigt auch die benötigte Busbandbreite linear mit der Prozessorzahl. Bei mehrstufigen Netzwerken, bei denen die Kommunikation auf Punkt-zu-Punkt und nicht auf Rundruf Basis stattfindet, ist der Anstieg der benötigten Bandbreite sogar quadratisch. Dies liegt darin, daß Schreibzugriffe an immer mehr Prozessoren weitergeleitet werden müssen.

2. Die Anzahl von Transaktionen, die jeder Netzwerkschnittstelle überwachen muß, steigt linear mit der Anzahl der Prozessoren und deren Taktfrequenz. Dadurch wird die Schnittstelle bei zu hoher Prozessorzahl überfordert. Dies liegt allerdings nicht an der Rechenleistung des Chips. Der begrenzende Faktor ist vielmehr die IO-Bandbreite des Schnittstellen Chips.

3. Die Anzahl von Invalidierungen bzw. Aktualisierungen, die in jedem Cache pro Zeiteinheit durchgeführt werden müssen, steigt ebenfalls linear mit der Anzahl der Prozessoren und deren Taktfrequenz. Da die Cache-Zugriffsgeschwindigkeit beschränkt ist, kann der Cache ab einer gewissen Prozessorzahl die benötigten Operationen nicht mehr rechtzeitig durchführen. Dieses Problem wird oft als das Problem der *cache-update pressure* beueichnet.

Obige Faktoren haben dazu geführt, daß die anfangs beschriebenen naiven Protokolle kaum noch Verwendung finden. Statt dessen wird auf intelligente Protokolle gesetzt, die nur einen Teil der Schreibzugriffe per Rundruf verschicken. Solche Protokolle gehen von einer zurückschreibenden Cache-Architektur aus. Ihnen liegen einer oder mehrere der folgenden drei Überlegungen zugrunde:

1 Mit Hilfe zusätzlicher Zustände kann zwischen Daten unterschieden werden, die sich entweder nur in einem oder in mehreren Caches befinden. Im ersten Fall braucht bei einem Schreibzugriff keine Invalidierung zu erfolgen. So werden unnötige Buszugriffe vermieden.

2 Fordert ein Prozessor ein Datum an, daß sich bereits in einem anderen Cache befindet, so kann die Anfrage nicht nur aus dem Hauptspeicher, sondern auch aus dem Cache befriedigt werden. Letzteres verringert die Zugriffslatenz und die Hauptspeicherlatenz und entlastet die Speicherbänke.

3 Es kann für jede Cache-Zeile ein Besitzer bestimmt werden. Er hat in der Regel das alleinige Schreibrecht und stellt dem System bei weiteren Anfragen die aktuelle Version der Daten zur Verfügung.

Die Entwicklung von effizienten Cache-Kohärenz-Protokollen ist ein aktuelles Forschungsfeld (siehe z.B. [95]). Im Nachfolgenden werden vier der bekanntesten Verfahren erläutert.

Beispiel Snoopy-Aktualisierungsprotokolle

Zu den wichtigsten Aktualisierungsprotokollen gehören die Firefly- und Dragon-Protokolle. Beide unterschieden zwischen privaten und gemeinsam genutzten Daten, um nicht bei jedem Schreibzugriff einen Rundruf durchführen zu müssen. Das Prinzip dieser beiden Protokolle kann wie folgt zusammengefaßt werden:

Firefly-Protokoll: Das Firefly-Protokoll [12] gehört zu den einfachsten Snoopy-Aktualisierungsprotokollen. Es benutzt drei Zustände: ReadPrivate, DirtyPrivate und ReadShared. Die beiden ersten Zustände sind für Daten bestimmt, die sich ausschließlich in dem Cache eines einzigen Prozessors befinden. Dabei bezeichnet DirtyPrivate Cache-Zeilen, die durch einen Schreibzugriff verändert wurden und irgendwann in den Hauptspeicher zurückgeschrieben werden müssen. Zeilen, auf die nur gelesen wurde, sind durch den ReadPrivate

Zustand gekennzeichnet. Eine Cache-Zeile geht aus einem der beiden Private Zustände dann in ReadShared über, wenn sie von einem weiteren Prozessor angefordert wird. In diesem Fall wird die Anfrage nicht durch den Hauptspeicher, sondern durch den Cache erfüllt, der das Datum aus dem Speicher geholt hatte. Dies gilt sowohl für Schreib- als auch für Lesezugriffe. Ein Schreibzugriff auf eine ReadShared Zeile muß immer durchgeschrieben werden, damit die Daten in den anderen Caches aktualisiert werden können. Hierbei wird der neue Wert auch vom Hauptspeicher übernommen. Eine gemeinsam genutzte Zeile kann somit nie 'Dirty' sein. Ob die Zeile nach der Schreiboperation im ReadShared Zustand verbleibt, oder in den ReadPrivate Zustand wechselt, hängt von der Shared-Leitung ab. Sie muß, während die Daten des Schreibzugriffs auf dem Bus liegen, von allen Prozessoren gesetzt werden, die diese in ihrem Cache haben. Falls die Leitung nicht gesetzt wird, nimmt der schreibende Prozessor an, daß die Zeile aus allen anderen Caches entfernt wurde. In diesem Fall geht sie in den ReadPrivate Zustand über.

Dragon-Protokoll: Das Dragon-Protokoll [118] unterscheidet sich vom Firefly-Protokoll dadurch, daß ein Schreibzugriff auf eine gemeinsam genutzte Cache-Zeile nicht sofort zum Hauptspeicher weitergeleitet wird. Aus diesem Grund wird ein zusätzlicher DirtyShared Zustand benötigt. Er gibt an, daß der Inhalt der Zeile in den Hauptspeicher zurückgeschrieben werden muß, bevor er aus dem Cache entfernt wird. Da die Daten nur einmal zurückgeschrieben werden müssen, kann eine Zeile nur in einem Cache DirtyShared sein. In allen anderen Caches hat sie den ReadShared Zustand. Ein Übergang in den DirtyShared Zustand ist aus allen vier Zuständen möglich. Er findet bei einem Schreibzugriff auf ein Datum statt, das sich bereits in einem anderem Cache befindet. Dabei wird der Zustand der Zeile bei allen anderen immer auf ReadShared gesetzt.

Snoopy Invalidierungsprotokolle

Zu den wichtigsten Invalidierungsverfahren zählen die MESI Protokollklasse und das Berkeley-Protokoll.

MESI-Protokolle: Der Name MESI stammt von den vier Zuständen, die diese Protokollklasse verwendet: **M**odified, **E**xclusiv, **S**hared und **I**nvalid (siehe z.B. [138]). Im Gegensatz zu dem einfachen WTI Protokoll unterscheiden MESI-Protokolle zwischen Daten, die sich ausschließlich in einem einzigen Cache befinden und solchen, die mehreren Caches gemeinsam sind. Die ersten drei Zustände entsprechen weitgehend den DirtyPrivate, ReadPrivate und ReadShared Zuständen des Firefly-Protokolls. Ein Unterschied ergibt sich lediglich bei Schreibzugriffen. Da MESI-Protokolle nach dem Invalidierungsprinzip arbeiten, gehen nach einen Schreibzugriff auf eine Zeile alle Kopien in den Invalid Zustand über. Der neue Wert wird in den Hauptspeicher geschrieben und die Zeile geht im Cache des schreibenden Prozessors in den Exclusiv Zustand über.

Berkeley-Protokoll: Das Berkley-Protokoll [80] unterscheidet sich von den MESI-Protokollen in zwei Punkten. Zum einem wird jeder Cache-Zeile ein eindeutiger Besitzer zugewiesen, der die Daten bei einer Anfrage zur Verfügung stellen muß. Dies kann entweder ein Cache oder der Hauptspeicher sein. Zum anderen unterschiedet es nur bei Schreibzugriffen zwischen privaten und gemeinsam genutzten Daten. Es verwendet vier Zustände: Invalid, ReadOnly, SharedDirty und PrivateDirty. Daten, die nur gelesen wurden, befinden sich immer in dem ReadOnly Zustand. Dabei spielt es keine Rolle, ob sie privat sind, oder sich in mehreren Caches gleichzeitig befinden. Solange auf die Zeile kein Schreibzugriff stattfindet,verbleibt sie in diesem Zustand. Nach einer Schreiboperation wird der schreibende Prozessor zum Besitzer der Zeile und versetzt sie in den DirtyPrivate Zustand. In allen anderen Caches wird sie invalidiert. Ein Übergang in den SharedDirty Zustand findet nur dann statt, wenn ein anderer Prozessor die Zeile lesen möchte. In diesem Fall wird die Zeile vom Besitzer zur Verfügung gestellt, der sie in seinem Cache als SharedDirty markiert. In den anderen Caches ist die Zeile ReadOnly. Wird auf die im SharedDirty Zustand befindliche Zeile erneut schreibend zugegriffen, so muß eine Invalidierungsmeldung an alle anderen Prozessoren geschickt werden. Diejenigen, die die Zeile in ihrem cache besitzen (im ReadOnly Zustand) markieren sie dann als ungültig. Im Cache des schreibenden Prozessors geht die Zeile dagegen von dem SharedDirty in den DirtyPrivate Zustand über.

3.3.2 Verzeichnis-Protokolle

Beim Verzeichnis-Protokollen (siehe [58] für einen Überblick) wird in einem Verzeichnis für jede Cache-Zeile mitprotokolliert, in welchen Caches sich eine Kopie dieser Zeile befindet. Das Verzeichnis wird bei jedem Schreibzugriff

konsultiert. So können Schreibzugriffe auf gemeinsam genutzte Daten gezielt nur zu den Caches weitergeleitet werden, die tatsächlich eine Kopie der Daten enthalten. Damit das Verfahren funktioniert, muß das Verzeichnis ständig aktualisiert werden. Dazu muß jeder Prozessor, der Daten in seinen Cache lädt oder sie aus dem Cache entfernt, dies in dem Verzeichnis vermerken.

Der Vorteil des Verzeichnis-Verfahrens gegenüber dem Snoopy-Verfahren ergibt sich daraus, daß die Kommunikationsstruktur der meisten parallelen Anwendungen eine starke Lokalität aufweist. So werden die meisten globalen Variablen immer nur von einer kleinen, von der Prozessorzahl unabhängigen Anzahl von Prozessoren genutzt. Dies bedeutet, daß die durchschnittliche Anzahl von Caches, von denen eine Cache-Zeile gemeinsam genutzt wird, gering ist. Sie ist vor allem auch weitgehend unabhängig von der Anzahl der Prozessoren.

1. In einem mehrstufigen Netzwerk steigen die Anforderungen an die Netzwerkbandbreite nicht mehr quadratisch wie im Falle der Snoopy-Protokolle, sondern nur noch linear mit der Anzahl der Prozessoren. Damit kann auch für hohe Prozessorzahlen problemlos die benötigte Bandbreite zur Verfügung gestellt werden. Die Bandbreite mehrstufiger Netzwerke steigt nämlich in der Regel linear mit der Anzahl der Netzknoten an.

2. Die Anzahl der Nachrichten, die die einzelnen Cache-Kontroller bearbeiten müssen, steigt nicht mit der Anzahl der Prozessoren an. Somit erübrigt sich das Problem der 'cache update preassure', das die Skalierbarkeit von Snoopy-Protokollen beschränkt.

Aus obigen Überlegungen folgt, daß sich Verzeichnis-Protokolle sehr gut für massiv parallele Rechner eignen. So nimmt man heute als gesichert an, daß sich skalierbare Multiprozessoren nur mit Hilfe dieser Protokollklasse realisieren lassen [95]. Andererseits ist es bisher nicht gelungen, Verzeichnis-Protokolle mit der gleichen Effizienz wie Snoopy-Protokolle zu implementieren. Aus diesem Grund geht man davon aus, daß nur NUMA-Multiprozessoren skalierbar sind, währen SMPs grundsätzlich auf wenige Prozessoren beschränkt sind.

Um die Gründe für die mangelnde Effizienz von Verzeichnis-Protokollen zu verstehen, muß man die verschiedenen Möglichkeiten ihrer Implementierung betrachten. Wie auch bei Snoopy-Protokollen unterschiedet man bei Verzeichnis-Protokollen zunächst zwischen Invalidierungs- und Aktualisierungsverfahren, sowie 'write through' und zurückschreibenden Systemen. Hinzu kommt als weiteres Kriterium der Aufbau des Verzeichnisses. Hierzu gibt es zwei Möglichkeiten: ein zentrales Verzeichnis (ZV) oder ein verteiltes Verzeichnis (VV).

Zentrales Verzeichnis (ZV)

Beim zentralen Verzeichnis (z.B. [6, 23]) wird die Information über den Verbleib jeder Cache-Zeile an einer vorbestimmten, allen Prozessoren bekannten Stelle gespeichert. Dies erfolgt in der Regel in der Speicherbank, aus der die Cache-Zeile stammt. In dem Verzeichnis gibt es für jede Cache-Zeile des Hauptspeichers einen eigenen Eintrag. Im einfachsten Fall beinhaltet der Eintrag ein Bitfeld, in dem jedes Bit einem Prozessor zugeordnet ist. Dabei zeigt ein gesetztes Bit an, daß eine Kopie der Zeile im Cache des korrespondierenden Prozessors vorhanden ist. Die größten Probleme des ZV Verfahrens sind der Speicherplatzbedarf und die Zugriffsgeschwindigkeit. Bei 256 Prozessoren wären für jede Cache-Zeile 32 Byte nötigt. Dies ist halb soviel wie die typische Cache-Zeilenlänge von 64 Byte.

Verteiltes Verzeichnis(VV)

Die Idee hinter dem VV (z.B. [172]) besteht darin, über den Verbleib von der Cache-Zeilen in Form einer verzeigerten Liste zu verwalten, die auf die Prozessoren verteilt ist. Jedes Element der Liste enthält einen Verweis auf den nächsten Prozessor, der die Zeile in seinem Cache hat. Die Adresse des ersten Elementes der Liste wird zentral (üblicherweise in der Speicherbank) gespeichert. Ein Prozessor, der eine Zeile in seinen Cache holt, liest diese Adresse und fügt sich vor das bisherige erste Element in die Liste ein. Sobald die Zeile aus seinem Cache entfernt wurde, löscht er sich aus der Liste, indem er den Zeiger seines Vorgängers von sich auf seinen Nachfolger umsetzt. Zur Erhaltung der Cache-Kohärenzmuß bei jedem Schreibzugriff eine Invalidierung bzw. Aktualisierung durch die gesamte Liste geschickt werden.

Fazit

Snoopy-Protokolle sind einfach und können daher theoretisch auch für große Prozessorzahlen eine geringe uniforme Speicherzugriffszeit realisieren. Leider erfordern sie ein Rundruf-Netzwerk, dessen Bandbreite linear mit der

Anzahl der Prozessoren ansteigt. Auch die Bandbreitenanforderungen an die Netzwerkschnittstellen und die Cache-Verwaltung stegen bei Snopy-Protokolle linear mit der Anzahl der Prozessoren. Sie sind daher nicht skalierbar und konnten bisher in der Praxis nur für wenige Prozessoren verwirklicht werden.

Im Hinblick auf die Skalierbarkeit vielversprechender sind Verzeichnis-Protokolle. Sie besitzen aber einen hohen Verwaltungsaufwand, der mit der Anzahl der Prozessoren stark ansteigt. Aus diesem Grund sind sie für die Implementierung einer uniformer Speicherzugriffszeit für große Prozessorzahlen nicht geeignet.

3.4 Verbindungsnetzwerk

Die einfachste Möglichkeit, einen SMP zu realisieren besteht darin, an den Speicherbus eines sequentiellen Rechners (Abbildung 3.1) weitere Prozessoren und Speicherbänke anzuschließen (Abbildung 3.2a).

Der Nachteil einer Busarchitektur besteht darin, daß ein Bus aus nur einem Datenkanal besteht, an den alle Prozessoren und Speicherbänke angeschlossen sind. Um die mit steigender Prozessorzahl steigende Bandbreitenanforderungen zu befriedigen, muß Bandbreite dieses Kanals linear mit der Anzahl der Prozessoren zunehmen. Wie im Kapitel 4 erläutert wird, ist die Realisierung eines Datenkanals mit hoher Bandbreite und hohem Fanout mit heutiger (elektronischer) Technologie nur schwer möglich. Ja größer der Fanout und die physikalische Ausdehnung eines elektronischen Datenkanals, um so geringer ist die maximale erreichbare Bandbreite. Hinzu kommt, daß mit steigenden Fanout die Latenz stark zunimmt. Dies beschränkt die Anzahl der Prozessoren die in der Praxis mit Hilfe eines leitungsvermittelnden Busses verbunden werden können auf einige wenige. Außerdem wird dadurch die Ausdehnung des Systems auf weniger als 1m eingeschränkt.

Dieses Problem kann durch die Verwendung anderer Netzwerke umgangen werden. Die wichtigsten darunter sind paketvermittelnde Busse, Mehrbus-Systeme, Punkt-zu-Punkt Netzwerke, so wie gemischte Netzwerke, die ein Punkt-zu-Punkt mit einem Bus kombinieren. Sie werden im Nachfolgenden kurz beschrieben. Ein ausführlicher Überblick über Verbindungsnetzwerke ist z.B. in [171] zu finden.

3.4.1 Paketvermittelnder Bus

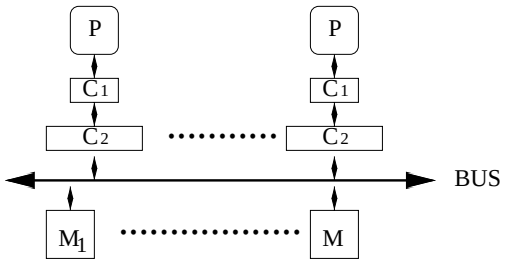
Bei einem paketvermittelnden Bus bleibt der Prozessor nicht für die gesamte Dauer der Transaktion im Besitz des Busses. Stattdessen schickt er seine Anfrage an die Speicherbank und gibt den Bus wieder frei. Während die Speicherbank die Anfrage bearbeitet, ist der Bus damit anders als bei der Leitungsvermittlung frei für andere Aufgaben. Erst wenn die Speicherbank die Bearbeitung der Anfrage beendet hat, fördert sie den Zugriff auf den Bus und schickt die Daten zum Prozessor zurück. Das Verfahren hat den Vorteil daß die Bandbreite des Busses besser genutzt wird. Aus diesem Grund wird es heute bei vielen Hochleistungsbussen verwendet (siehe z.B. [18]). Allerdings ändert es nichts an der fundamentalen Beschränkung der Busbandbreite, die sich durch die Verwendung eines einzigen Datenkanals ergibt. Somit kann die Skalierbarkeit eines Systems durch die Nutzung der Paketvermittlung nur um einen kleinen Faktor erhöht werden. Er liegt in der Praxis bei 1,5 bis 2.

3.4.2 Mehrbus-Systeme

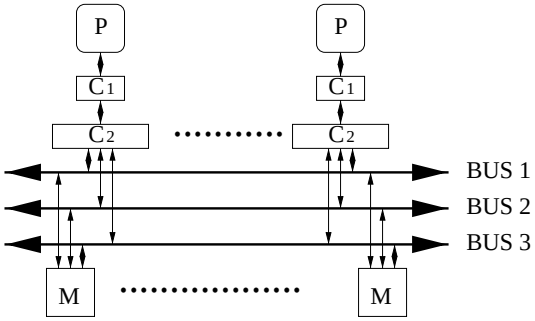
Eine naheliegende Möglichkeit die Bandbreitenbeschränkung eines Busnetzwerks zu umgehen, besteht darin, mehrere Busse zu benutzen [170] (Abbildung 3.2 b). Dieser Ansatz wird in verschiedenen Architekturen benutzt und wurde in zahlreichen Arbeiten untersucht (z.B. [15, 184]). Allerdings kann die Skalierbarkeit von Bussystemen mit Hilfe von Mehrbusarchitekturen aus zwei Gründen nicht über 32 bis 64 Prozessoren erhöht werden. Zum einen trägt die Benutzung mehrere Busse nichts dazu bei, die Probleme bei der effizienten elektronischen Realisierung von Rundrufleitungen mit hohen Fanout zu lösen. Mit steigender Prozessorzahl sinkt die Bandbreite der einzelnen Busse während die Latenz und der Implementierungsaufwand immer höher werden. Zum anderen ist die Anzahl der Busse an die man jeden einzelnen Prozessor anschließen kann durch die Leistung der Schnittstellen und der hohen Platzbedarf elektronischer Busse auf 2 bis 4 beschränkt.

3.4.3 Mehrstufige Punkt-zu-Punkt Netzwerke

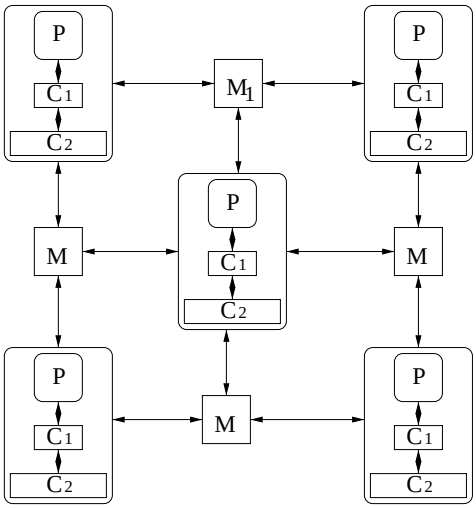
Das Bandbreitenproblem von Bussen kann mit Hilfe von mehrstufigen Punkt-zu-Punkt Netzwerken umgangen werden. In solchen Netzwerken ist jeder Knoten über dedizierte Leitungen mit einem oder mehreren anderen Knoten oder Schaltern verbunden. Beispiel für weit verbreitete Punkt-zu-Punkt Netzwerke sind ein Ring, ein



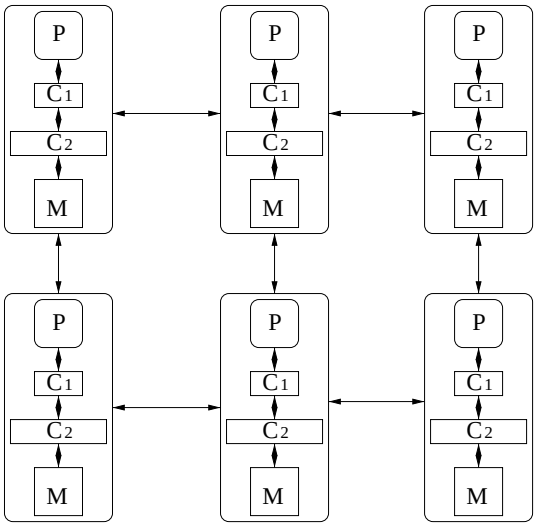
a) ein einfaches Bussystem



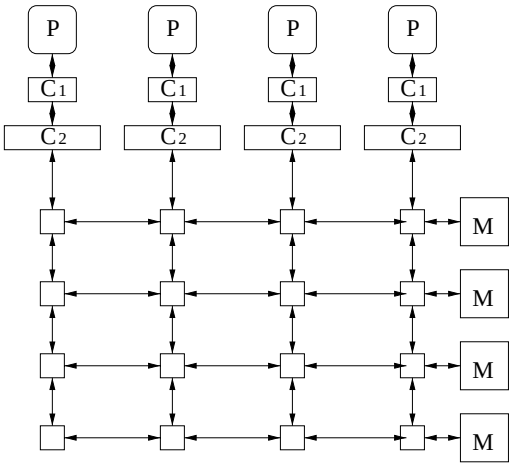
b) ein Mehrbussystem mit 3 Bussen



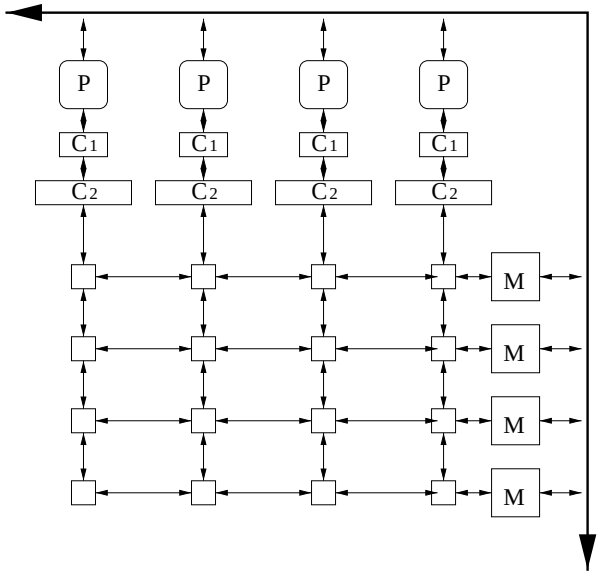
c) ein Gitternetzwerk in einem GS rechner



d) ein Gitternetzwerk in eine VS-Rechner



e) ein Crossbar-System



f) ein gemischtes Crossbar-Bus Netzwerk

Abbildung 3.2: Verschiedene Verbindungsnetzwerke für die Prozessor-Speicherkoppelung in Multiprozessoren.

Gitter, ein Gitter und Hyperwürfel. Der Einsatz eines solchen Netzwerks zur Prozessor Speicher-Koppelung in Multiprozessoren ist in Abbildung 3.2 c zu sehen

Mehrstufige Punkt-zu-Punkt Netzwerke haben gegenüber einem Bus den Vorteil, daß mit steigender Prozessorzahl auch die Anzahl der Datenkanäle ansteigt, da mit jedem zusätzlichen Knoten auch zusätzliche Verbindungen zum System hinzukommen. Damit steigt die Gesamtbandbreite des Netzwerks, ohne daß gleichzeitig die Datenrate der einzelnen Kanäle erhöht werden muß. Hinzu kommt, daß die Leitungen keinen Fanout haben, was die Realisierung einer hohen Bandbreite erleichtert (siehe Kapitel4).

Leider sind Punkt-zu-Punkt Netzwerke aus zwei Gründen nur bedingt für die Implementierung von SMPs geeignet.

1. Die Latenz steigt in den meisten Netzwerken stark mit der Anzahl der Prozessoren an. Dies liegt daran, daß es in der Regel nicht zwischen allen Knoten eine direkte Verbindung gibt. Nachrichten müssen daher über mehrere Stationen wandern, bevor sie ihr Ziel erreichen. Dabei wächst die Anzahl der Stationen, die eine Nachricht im durchschnitt durchwandern muß, mit der Anzahl der Knoten. Dies führt dazu, daß in großen Punkt-zu-Punkt Netzwerken die durchschnittliche Kommunikationslatenz im Bereich vom Mikrosekunden liegt. Punkt-zu-Punkt Netzwerke werden daher meistens in NUMA und VS Rechnern verwendet. Ein Beispiel für eine solche Anwendung ist in Abbildung 3.2d zu sehen.

2. Die Vorteile von Punkt-zu-Punkt Netzwerken können nur im Zusammenhang mit Verzeichnis-Protokollen genutzt werden, die wie im vorigen Abschnitt beschrieben nicht effizient genug für die Realisierung des UMA-Modells sind. Bei Snoopy-Protokollen wird die höhere Bandbreite durch den Aufwand zunichte gemacht, der für die Implementierung von Rundruf-Operationen notwendig ist. Im Punkt-zu-Punkt Netzwerk wird ein Rundruf durch die Vervielfältigung von Nachrichten durchgeführt. Es wird also eine eigenständige Nachricht an jeden Prozessor geschickt. Die für einen Rundruf benötigte Netzwerkbandbreite ist also proportional zur Anzahl der Prozessoren. Da die Anzahl der Rundruf-Operationen pro Zeiteinheit ebenfalls zur Anzahl der Prozessoren proportional ist, steigt die benötigte Bandbreite quadratisch mit der Anzahl der Prozessoren. Dies bedeutet, daß entweder die Anzahl der Punkt-zu-Punkt Verbindungen superlinear mit der Anzahl der Prozessoren steigen muß, oder daß die Datenrate einzelner Kanäle wie beim Bus mindestens linear zunehmen muß.

3.4.4 Crossbars-Netzwerke

Eine Möglichkeit die hohe Latenz der mehrstufigen Netzwerke zu vermeiden, bietet ein Crossbar-Netzwerk (Abbildung 3.2e). Dabei hat jeder Prozessor die Möglichkeit über eine Schaltermatrix eine direkte Verbindung zu einer beliebigen Speicherbank herzustellen. In einem solchen Netzwerk wächst die Anzahl der Leitungen und damit die Gesamtbandbreite mit der Anzahl der Prozessoren ohne daß die Latenz stark zunimmt. Allerdings ist auch ein Crossbar-Netzwerk schlecht zur Realisierung der für Snoopy notwendigen Rundruf-Operationen geeignet. Wie auch bei mehrstufigen Punkt-zu-Punkt Netzwerke steigen die Bandbreitenanforderungen quadratisch mit der Prozessorzahl an. Außerdem steigt die Anzahl der Schalter in der Schaltmatrix quadratisch mit der Prozessorzahl.

3.4.5 Gemischte Netzwerke

In den meisten Snoopy-Protokollen wird ein Rundruf nur für Schreiboperationen auf gemeinsam genutzte Daten benötigt. Andere Operationen erfordern lediglich eine Punkt-zu-Punkt Kommunikation zwischen einem Prozessor und der an der Operation beteiligten Speicherbank. Um dies auszunutzen verwenden einige Systeme ein gemischtes Netzwerk, daß aus einem Bus und einem Punkt-zu-Punkt Netzwerk besteht. Von besonderer Bedeutung sind dabei Netzwerke, die einen Bus mit einem Crossbar kombinieren (z.B. die Gigaplane Architektur von Sun [121]). Der Bus wird für alle Cache-Kohärenz-Operationen verwendet. Alle anderen Zugriffe werden über das Punkt-zu-Punkt Netzwerk abgewickelt. Dadurch sinken die Bandbreitenanforderungen an den Bus je nach Anwendung um einen Faktor von 2 bis 3. Das System kann also für eine größere Anzahl von Prozessoren realisiert werden. Allerdings ändert auch dieses Verfahren nichts an dem linearen Anstieg der benötigten Busbandbreite. Hinzu kommen die bereits erwähnten Probleme bei der elektrischen Realisierung eines großen Fanouts, die zu einem Absinken der Bandbreite bei steigender Latenz führen.

3.4.6 Fazit

Busse können theoretische eine sehr geringe Latenz besitzen und eignen sich optimal für die Implementierung effizienter snoopy-Protokolle. Sie haben aber den Nachteil, daß die Anzahl der Datenkanäle mit der Anzahl der Prozessoren nicht wächst. Dies bedeutet daß mit steigender Prozessorzahl die Bandbreite des Datenkanals linear steigen muß, was elektronisch nur schwer realisierbar ist, insbesondere dann wenn ein hoher Fanout benötigt wird.

Das Problem wird durch Punkt zu Punkt Netzwerke umgangen. Sie haben aber eine hohe Latenz und eignen sich nicht für die Implementierung von Snoopy Protokollen

3.5 Schranken der Skalierbarkeit von SMPs

Bei der Frage nach der Skalierbarkeit von SMPs geht es darum, wie viele Prozessoren zu einem SMP zusammengefaßt werden können, ohne daß die durchschnittliche Speicherlatenz dadurch wesentlich über die Latenz sequentieller Rechner ansteigt. Die fundamentalen Schranken der Skalierbarkeit ergeben sich aus der durch die Lichtgeschwindigkeit c beschränkten Signalausbreitungsgeschwindigkeit und dem unvermeidbaren Anstieg der Systemdimensionen mit steigender Prozessorzahl. Ein System mit vielen Prozessoren ist größer als ein sequentieller Rechner. Dementsprechend größer ist auch die Signallaufzeit zwischen den Prozessoren und dem Hauptspeicher und damit die durchschnittliche Speicherlatenz. Allerdings spielen die fundamentalen Schranken gegenwärtig in der Praxis keine Rolle. Bei einer Lichtgeschwindigkeit von $c \sim 30cm/ns$ würde die Signallaufzeit erst bei einem Systemdurchmesser d von ca. $d = 100m$ in den Bereich der heute üblichen Hauptspeicherlatenz von 200ns gelangen. Hinzu kommt, daß der Systemdurchmesser mit der Kubikwurzel aus der Anzahl von Prozessoren ansteigt. Somit bleiben selbst bei Rechnern mit mehreren Tausend Prozessoren auf absehbare Zeit allein die technologischen Schranken für die Speicherlatenz verantwortlich.

Die technologischen Schranken bestehen vor allem in der effizienten Implementierung der Cache-Kohärenz für große Prozessorzahlen und Systeme mit großer räumlicher Ausdehnung. Die Ursachen hierfür wurden bereits bei der Beschreibung der jeweiligen Protokolle angesprochen. Im Nachfolgenden werden sie nochmals zusammengefaßt und an Hand einiger Technologieparameter verdeutlicht. Da sich die vorliegende Arbeit mit Snoopy-Protokollen beschäftigt, liegt bei der nachfolgenden Betrachtung die Betonung auf den Problemen dieser Protokoll-Klasse.

3.5.1 Effizienz der Verzeichnis-Protokolle

Die Effizienz der Verzeichnis-Protokolle ist durch 3 Faktoren beschränkt

1. Im Bezug auf die Skalierbarkeit haben die Verzeichnis-Protokolle gegenüber Snoopy-Protokollen vor allem den Vorteil, daß sie mit Punkt-zu-Punkt Netzwerken realisiert werden können. Ihre Verwendung macht nur zusammen mit solchen Netzwerken Sinn. Wie im vorigen Abschnitt erläutert, besitzen Punkt-zu-Punkt Netzwerke aber eine hohe Latenz, die stark mit der Prozessorzahl steigt. Sie kann im Bereich von Mikrosekunden liegen.

2. Bei Protokollen mit verteilten Verzeichnis müssen die alle Schreibzugriffe hintereinander alle Prozessoren abwandern, die eine Kopie des Datums haben. Dadurch wird die Auswirkung der hohen Netzwerklatenz potenziert. Ein Schreibzugriff kann zwischen 10 und 100 μs dauern.

3. Protokolle mit zentralen Verzeichnis können wie bereits beschrieben die Speicherdichte halbieren. Sie sind in ihrer naiven Form daher für hohe Prozessorzahlen nicht praktikabel. Fortgeschrittene Protokolle, die sparsamer mit dem Speicher umgehen, haben dagegen einen recht hohen Verwaltungsaufwand, der die Latenz stark erhöht.

3.5.2 Skalierbarkeit von Snoopy-Protokollen

Die Probleme bei der Skalierbarkeit der Snoopy-Protokolle entspringen daraus, daß ein jeder Prozessor bei jeder Schreiboperation oder bei jedem Cache-Miss eine Nachricht an den Speicher und an *alle* Prozessoren schicken muß. Die genaue Häufigkeit, mit der ein einzelner Prozessor solche Nachrichten verschickt, ist von Anwendung zu Anwendung und von Protokoll zu Protokoll verschieden. Sie bleibt aber immer mit steigender Prozessorzahl entweder konstant oder steigt sogar aufgrund von Invalidierungen leicht an. Die Anzahl von Rundruf-Vorgängen

steigt also mindestens linear mit der Anzahl der Prozessoren an. Dieser linearer Anstieg führt dazu, daß mit steigender Prozessorzahl das Netzwerk, die Netzwerkschnittstellen und die Caches selber von der großen Datenmenge überfordert werden.

Anzahl und Länge der Nachrichten

Die Anzahl der Nachrichten, die jeder Prozessor im Durchschnitt pro Zeiteinheit per Rundruf verschickt, hängt von vielen Faktoren ab. Sie wurde für eine Menge von numerischen Anwendungen in [95] und [187] untersucht. Dabei wurde ein Wert von 1 bis 10 Nachrichten pro 100 Prozessoroperationen ermittelt. Die Länge der Nachrichten ist unterschiedlich. Die kürzesten bestehen lediglich aus einem Kommando-Wort und einer Adresse, die längsten beinhalten zusätzlich noch eine oder mehrere Cache-Zeilen. Da in den meisten Protokollen vor allem Invalidierungen und Datenanforderungen einen Rundruf erfordern, wird in der nachfolgenden Betrachtung von kurzen Nachrichten ausgegangen, deren Länge auf ca. 100 Bit geschätzt wird (eine 64-Bit Adresse und ein 32 Bit-Kontrollcode). Damit wird pro Prozessor mindestens eine Rundruf-Bandbreite von 1 bis 10 Bit pro Prozessorzyklus benötigt. Geht man von einer Prozessorfrequenz von 500 MHz aus, sind dies zwischen 500 Mbit/s und 5Gbit/s. Für ein effizientes Speichersystem muß man zusätzlich berücksichtigen, daß die Prozessoren ihre Nachrichten in der Regel nicht gleichmäßig über die Zeit verteilt verschicken. Um lange Verzögerungen zu 'Stoßzeiten' zu vermeiden, wird daher eine um einen Faktor von 2 bis 4 höhere Bandbreite benötigt.

Busbandbreite

Die die Anzahl der Rundruf-Operationen linear mit der Anzahl der Prozessoren ansteigt, muß wie beschrieben auch die für der Rundruf zur Verfügung stehende Bandbreite im gleichen Maße steigen. Die besten heutigen Busse erreichen eine Bandbreite ('aggregate bandwidth') von ca. 30Gbit/s. Dies beschränkt die Prozessoranzahl auf bestenfalls 60 und schlechtestenfalls 6.

Wie bereits beschrieben kann das Problem auch nicht durch die Verwendung eines Punkt-zu-Punkt Netzes gelöst werden. Da der Rundruf durch Vervielfältigung von Nachrichten durchgeführt wird steigt in einem solchen Netz die benötigte Gesamtbandbreite quadratisch mit der Anzahl der Prozessoren an. Hinzu kommt die hohe Latenz, die mit 1 bis 10 μs um einen Faktor von ca. 10 über der sequentiellen Speicherzugriffszeit liegt.

Schnittstellenleistung

Die hohe Anzahl von Rundruf-Operationen stellt nicht nur für die reine Datenübertragung ein Problem dar. Sie bedeutet auch, daß die Netzwerkschnittstellen der Prozessoren eine Datenmenge empfangen und verarbeiten müssen, die linear mit der Prozessorzahl wächst.

Wie in Kapitel 4 erläutert, stellt dabei vor allem die IO-Bandbreite der VLSI-Bausteine eine starke Beschränkung dar. Heutige VLSI-Bausteine besitzen zwischen 500 und 1000 Pins, über die sie mit der Außenwelt kommunizieren. Diese Pins werden in der Regel mit 100 MHz, also ca. einem Fünftel der heute üblichen Prozessorfrequenz betrieben. Bei 100 Bit-langen Nachrichten heißt das, daß 1 bis 2 Nachrichten pro Prozessorzyklus eingelesen werden können.

Cache-Update-Pressure

Selbst wenn es gelänge, die benötigte Netzwerk- und Schnittstellenbandbreite zu realisieren, so steht man immer noch vor dem Problem der Cache-Geschwindigkeit. Jede Nachricht muß nicht nur von jedem Prozessor empfangen werden, sondern auch mit dem Cache-Inhalt verglichen werden. Bei Bedarf muß außerdem entweder der Cache-Inhalt modifiziert werden, oder ein Datum aus dem Cache geholt werden. Da bei Snoopy-Protokollen jede Nachricht von allen Prozessoren untersucht wird, ist damit die maximale Datenrate durch die Anzahl von Anfragen beschränkt, die ein einzelner Cache pro Zeiteinheit verarbeiten kann. Die Schranke wird wie bereits erwähnt oft als das *cache-update-pressure*-Problem bezeichnet. Unter der Annahme, daß der Cache mit der Prozessorgeschwindigkeit betrieben werden kann, kann also maximal eine Nachricht pro Prozessorzyklus übertragen werden. Geht man wie beschrieben davon aus, daß jeder Prozessor zwischen 1 und 10 Nachrichten pro Hundert Operationen generiert, so ist die Anzahl der Prozessoren damit unabhängig von der Netzwerk- und Schnittstellenleistung auf 10 bis 100 beschränkt.

Eine Möglichkeit, diese Beschränkung zu umgehen, besteht darin, den Zugriff auf den Cache zu parallelisieren. Dabei stellt sich aber das Problem, daß die Dichte eines Speicherbausteins mit mehreren parallelen Zugriffsports quadratisch mit der Anzahl dieser Ports abnimmt. Daher sind mehr als 2 bis 4 Ports nicht praktikabel.

3.6 Cache-kohärente parallele Rechnerarchitekturen

Der vorliegende Abschnitt gibt einen Überblick über parallele Rechnerarchitekturen mit cache-kohärenten gemeinsamem Speicher. Es berücksichtigt sowohl kommerzielle Produkte als auch die wichtigsten Forschungsprojekte. Dabei wird deutlich, daß die Entwicklung großer SMPs weder kommerziell noch im Bereich der Forschung verfolgt wird. Aufgrund der im vorigen Abschnitt geschilderten Probleme nimmt man es als gesichert an, daß nur NUMA-Architekturen für Rechner mit mehr als 32 bis 64 Prozessoren geeignet sind.

3.6.1 Kommerzielle Rechner

Im Bereich kommerzieller Parallelrechner stellen SMPs, PCs, Arbeitsplatzrechner und kleine Server die am meisten verbreitete parallele Architektur dar. Sie sind die erste und bisher einzige parallele Architektur, die eine breite Akzeptanz im Massenmarkt gefunden hat.

SMPs

Kleinere (bis 4 Prozessoren) SMPs, Arbeitsplatzrechner und PCs werden von fast allen namhaften Herstellern angeboten. Unter den größeren Rechner aus dem Server-Bereich sind vor allem der Enterprise 10000, die DEC-Alpha-Server und die POWERCHALLANGE-Architektur von SGI erwähnenswert.

SUN Enterprise Server 10000: Der Enterprise 10000 Server der Firma Sun [122] ist der zur Zeit größte symmetrische Multiprozessor. Er unterstützt bis zu 64 Prozessoren bei einer maximalen Hauptspeicherlatenz von 500ns. Um die hohe Prozessorzahl zu realisieren, werden zwei Netzwerke verwendet: ein paketvermittelnder Bus für die Daten, die unbedingt einen Rundruf erfordern, und ein Crossbar-Netzwerk für alle anderen Operationen. Das Busnetzwerk hat eine Bandbreite von ca. 20 Gbit/s, die mit 256 Leitungen bei 83MHz Betriebsfrequenz erreicht wird.

DEC-Alpha-Server 4xxx und 8xxx: Die Rechner der 4xxx und 8xxx Server Serie von DEC [112, 81] können bis zu 12 Prozessoren beinhalten. Die Hauptspeicherlatenz beträgt 300ns. Als Verbindungsnetzwerk dient ein Hochleistungsbus mit einer Bandbreite von ca. 16Gbit/s.

SGI Challenge: Der Challenge Rechner von SGI [53] war der erste größere kommerzielle SMP Rechner. Basierend auf den R3000- und später R10000 Prozessoren wurde ein Snoopy-Protokoll basiertes CC-UMA Speichersystem für bis zu 36 Prozessoren realisiert. Die Architektur wurde inzwischen von SGI zugunsten der Origin2000 CC-NUMA-Architektur aufgegeben (siehe unten).

NUMA-Rechner

Große Server und sogar parallele Superrechner werden zunehmend als CC-NUMA Architekturen implementiert. Gute Beispiele für solche Systeme sind der Sequent-Symmetry, die ??? von SGI und SCI Maschinen.

SGI Origin 2000: Die Origin 2000 Architektur von SGI [92] erlaubt die Integration von bis zu 1024 MIPS 10000 Prozessoren zu einem CC-NUMA Rechner. Dabei werden Bus-basierte 2-Prozessor Knoten mit einem Hypercube-ähnlichem Netzwerk verbunden. Zur Erhaltung der Cache-Kohärenz wird ein Verzeichnis-Protokoll mit zentralem Verzeichnis verwendet. Das System hat eine globale Speicherlatenz von bis zu $1\mu s$.

SCI: SCI definiert eine Standardschnittstelle für die Realisierung des CC-NUMA Speichermodells mit beliebigen Punkt-zu-Punkt Netzwerken [76, 61]. Es wird vor allem im Bereich von Arbeitsplatzrechner-Clustern eingesetzt. Die globale Speicherlatenz liegt beim SCI-System daher in der Regel bei mehreren μs.

Sequent NUMA-Q: Das Numa-Q System ist das Nachfolgesystem der Sequent Symmetry Rechners [158]. Es verbindet bis zu 1024 Pentium-Prozessoren zu einem CC-NUMA System mit Hilfe eines, auf dem SCI Protokoll basierendem Netzwerk. Die Latenz globaler Speicherzugriffe beträgt bei größeren Systemen zwischen 5 und 15 μs [159].

3.6.2 Forschungsprojekte

Die im vorigen Abschnitt beschriebenen Schwierigkeiten bei der Realisierung von SMP-Architekturen haben dazu geführt, daß sich fast alle Forschungsprojekte auf CC-NUMA-Speichermodelle konzentrieren. Hinzu kommen einige NCC-UMA Ansätze [59, 9], die allerdings nur wenig mit der vorliegenden Arbeit zu tun haben. Das einzige UMA Projekt, das eine gewisse Ähnlichkeit mit der vorliegenden Arbeit aufweist, ist das SBPRAM-Projekt der Universität Saarbrücken.

SBPRAM

Im Rahmen des an der Uni-Saarbrücken durchgeführten Projektes wird eine direkte Implementierung der PRAM angestrebt [3]. Das System basiert auf einer Verbesserung des in [148] vorgeschlagenen Emulationsverfahrens. Es beinhaltet ein Netzwerk, das Anfragen zur gleichen Speicherstelle gemäß der PRAM-Speicherzugriffssemantik kombiniert [32]. Um die hohe Speicherlatenz zu verstecken, werden auf jedem Prozessor mehrere virtuelle Prozessoren simuliert.

NUMA-Architekturen

Die Entwicklung effizienter skalierbarer CC-NUMA Architekturen gehört zu den wichtigsten Forschungsschwerpunkten im Bereich paralleler Rechnerarchitektur. Zu den bedeutendsten Arbeiten auf diesem Gebiet gehören das MIT-Alwife Projekt, die DASH /FLASH Architekturen sowie das S3.mp Projekt von SUN.

MIT-Alwife: Die Alwife [84, 7, 48] Maschine ist ein CC-NUMA Rechner, der am MIT entwickelt wurde. Es verwendet SPRAC-Prozessoren, ein 2D-Gitternetzwerk und realisiert ein verzeichnisbasiertes Cache-Kohärenz-Protokoll für bis zu 512 Prozessoren.

DASH: DASH [94] ist ein skalierbarer CC-NUMA Rechner, der aus SMP-Knoten besteht. Jeder Knoten besteht aus vier Prozessoren und Speicherbänken, die untereinander einen gemeinsamen CC-UMA Speicher realisieren. Die Knoten sind durch ein Gitternetzwerk verbunden. Über dieses Netzwerk wird ein verzeichnisbasiertes zentrales Cache-Kohärenz-Protokoll mit schwachem Speicherkonsistenz-Modell realisiert. Die Speicherlatenz beträgt innerhalb des lokalen Knotens ca. 30 Prozessorzyklen, und im globalen Speicher bis zu 130.

FLASH: FLASH ([153, 68]) ist das Nachfolgeprojekt von DASH. In seinem Rahmen wurde das CC-NUMA Speichermodell mit einer effizienten hardwarenahen Schnittstelle für die Nachrichtenkoppelung verbunden. Dadurch wurde dem Programmierer die Möglichkeit gegeben, ausgewählte Programmteile durch gezieltes Verschicken von Nachrichten zu optimieren.

S3.mp: S3.mp ist ein Prototyp von der Firma Sun, der eine effiziente Koppelung von Arbeitsplatzrechnern zu einem CC-NUMA Rechner erlaubt [143, 131]. Das besondere an der S3.mp-Architektur ist, daß sie Netzwerkbausteine verwendet, die eine direkte Hochleistungsverbindung zwischen VLSI-Bausteinen erlauben. Es handelt sich dabei um sequentielle elektrische Verbindungen mit einer Datenrate von bis zu 1Gbit/s, von denen drei direkt mit dem Substrat des Netzwerkchips verbunden sind. In dieser Hinsicht ähnelt das Konzept dem Ansatz der vorliegenden Arbeit.

Ein Überblick über weitere Arbeiten ist in [27, 95] zu finden. Da sie nur entfernt mit der vorliegenden Arbeit verwandt sind, werden sie hier nicht weiter erläutert.

3.7 Fazit

Die Betrachtung in diesem Kapitel hat gezeigt, daß die Skalierbarkeit von symmetrischen Multiprozessoren auf zwei Arten erhöht werden könnte: entweder durch die Verbesserung der Skalierbarkeit von snoopy-Protokollen und Busarchitekturen oder durch die Verringerung der Latenz von Verzeichnis-Protokollen und Punkt-zu-Punkt Netzwerken. Dabei wurde deutlich, daß keine dieser beiden Absätzen mit konventioneller elektronischer Technologie erfolgversprechend ist. Die Skalierbarkeit der snoopy-Protokolle ist durch die Schranken elektrischer Rundruf-Leitungen und Netzwerkschnittstellen bestimmt. Die hohe Latenz von Verzeichnisprotokollen folgt aus

dem unvermeidbaren Verwaltungsaufwand und der hohen Latenz von Punkt-zu-Punkt Netzwerken. Aus diesem Grund wird heute als gesichert angenommen, daß für größere Prozessorzahlen nur NUMA Architekturen in Frage kommen. Daher beschäftigen sich alle Forschungsprojekte im Bereich skalierbarer Multiprozessoren mit NUMA-System.

Die vorliegende Arbeit zeigt dagegen, daß die obige Annahme nicht für Systeme mit opto-elektronische Prozessor Speicher Koppelung gilt. Dabei wird die Tatsache ausgenutzt, daß optisch Rundrufkanäle mit hohem Fanout, hoher Bandbreite und geringer Latenz möglich sind. Außerdem ermöglicht die Kombination optischer Ein/Ausgabekanäle mit VLSI Schaltkreise die Realisierung extrem leistungsfähiger Netzwerkschnittstellen.

Kapitel 4

Elektronik, Optik und die Datenübertragung

Das vorliegende Kapitel beschäftigt sich mit der opto-elektronischen Netzwerktechnik und ihren Anwendungsmöglichkeiten in der Rechnerarchitektur. Es richtet sich vor allem an den aus der Informatik kommenden, mit der Optik nicht vertrauten Leser. Ihm wird eine sanfte Einführung in die Problematik der optischen Datenübertragung und der dazugehörenden Technologie geboten. Dabei wird gezeigt, warum die Optik trotz fundamentaler Vorteile als Datenübertragungsmedium bisher für den Einsatz in der Rechnertechnik nicht geeignet war und wie neue technologische Entwicklungen dies zu ändern versprechen. Abschließend wird ein Überblick über aktuelle Forschungsarbeiten im Bereich optisch verbundener Parallelrechner gegeben und die vorliegende Arbeit gegen diese abgegrenzt.

4.1 Theoretische Vorteile der Optischen Datenübertragung

Eine Datenübertragung ist immer mit der physikalischen Ausbreitung eines Informationsträgers verbunden. Die Eigenschaften dieses Informationsträgers bestimmen die fundamentalen Eigenschaften einer Datenübertragungstechnologie. So ergeben sich auch die Vorteile der Optik gegenüber der Elektronik aus den physikalischen Unterschieden zwischen den jeweiligen Informationsträgern. Die elektrische Nachrichtenübertragung bedient sich bewegter Ladungen in Form eines elektrischen Stroms. In der Optik findet die Datenübertragung mit Hilfe des Lichts, einer extrem hochfrequenten (also kurzwelligen) elektro-magnetischen Welle statt.

Im Nachfolgenden werden die fundamentalen Unterschiede zwischen der Optik und der Elektronik zusammengefaßt. Dabei wird auch eine 'Blitzeinführung' in die Grundbegriffe der Optik gegeben. Das Ziel des Abschnitts besteht darin, dem Leser ein qualitatives Verständnis der physikalischen Hintergründe zu vermitteln. Es wird daher weitgehend auf Formeln und exakte Definitionen verzichtet.

4.1.1 Elektrische Datenübertragung

In der Elektronik findet der Informationstransport mit Hilfe des elektrischen Stroms, also der Bewegung massiver Ladungsträger statt. Auf Grund der elektrostatischen Abstoßung und der Wechselwirkung der Ladungsträger mit der Umgebung ist es kaum möglich, einen solchen Strom im Form eines fokusierten Strahls im freien Raum zu übertragen. Daher werden für die elektrische Datenübertragung Leiter benötigt.

Beschreibung der Signalausbreitung

Die Ausbreitung eines elektrischen Signals in einem Leiter kann durch eine lokale Veränderung der Ladungsdichte beschrieben werden, die sich wie ein Wellenberg entlang des Leiters ausbreitet (Abbildung 4.1). Sie wird durch die Spannung ausgelöst, die der Sender an der Leitung anlegt. Der Grad der Verdichtung ist proportional zu dieser Spannung, während die Anzahl der beteiligten Ladungen dem Strom entspricht. Eine aus mehreren Signalen zusammengesetzte Nachricht besteht aus mehreren solchen Wellenbergen. Sie gleicht also einem, sich entlang des Leiters fortbewegendem Wellenpaket. Dabei ist die Form des Wellenpakets durch den Inhalt und die

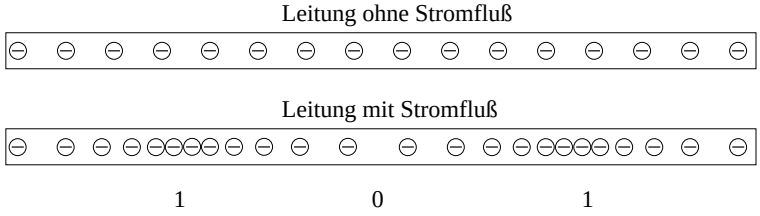


Abbildung 4.1: Das Prinzip der elektrischen Datenübertragung in einer Leitung.

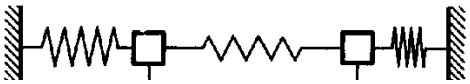


Abbildung 4.2: Die mechanische Analogie zur Ausbreitung eines elektrischen Signals in einer Leitung.

Datenfrequenz der Nachricht bestimmt. Die Ausbreitung solcher Wellenpakete kann mit Hilfe der Fourieranalyse auf die Fortpflanzung von harmonischen Schwingungen zurückgeführt werden. Der Anteil der einzelnen Fourierkomponenten hängt von der Form des Wellenpakets und damit von dem Inhalt und der Datenrate der Nachricht ab. Jede Komponente stellt eine erzwungene, gedämpfte, harmonische Welle einer anderen Frequenz dar. Die Ausbreitungseigenschaften solcher harmonischen Wellen sind bekannt und können verhältnismäßig einfach anhand des Widerstands, der Kapazität und Induktivität des Leiters sowie der Frequenz der Welle beschrieben werden. Um sie zu veranschaulichen, ist es nützlich, auf eine aus dem Alltag bekannte mechanische Analogie zurückzugreifen. Dazu wird die Leitung als eine Kette gekoppelter mechanischer Oszillatoren dargestellt. Jeder Oszillator steht für einen im Grenzfall infinitisemalen Teil der Leitung, wobei die Federstärke der Induktivität, die Masse der Kapazität und die Reibung dem Widerstand entspricht. Das elektrische Signal korrespondiert in dieser Analogie zu einer horizontalen Auslenkung, die sich entlang der Kette ausbreitet.

Eigenschaften elektrischer Datenübertragung

Die Probleme der elektrischen Datenübertragung können wie folgt zusammengefaßt werden:

Signalverzerrung durch Wellendämpfung: Versucht man die in Abbildung 4.2 dargestellte Oszillatorkette zum Schwingen zu zwingen, so stellt man fest, daß dies für bestimmte Schwingungsfrequenzen besser und für andere schlechter funktioniert. Analog gilt für einen elektrischen Leiter, daß die harmonischen Wellen verschiedener Frequenzen unterschiedlich gut geleitet werden. Dies liegt daran, daß die Dämpfung einer harmonischen Welle vom Abstand ihrer Frequenz von der sog. Eigenfrequenz des Systems abhängt. Die Eigenfrequenz ergibt sich beim Leiter aus ihrer Kapazität, Induktivität und ihrem Widerstand. Für ein zu übertragendes Signal, zu dem viele unterschiedliche Fourierkomponenten beitragen, bedeutet obiges, daß das Signal nicht nur gedämpft, sondern auch ganz erheblich verzerrt wird.

Signalverzerrung durch Wellenreflexion: Regt man eine Oszillatorkette an einem Ende an, so breitet sich die Erregung nur in den seltensten Fällen bis zum anderen Ende aus und wird dort absorbiert. Meistens wird sie entweder am Ende oder sogar schon vorher ganz oder teilweise reflektiert. Dabei sind auch mehrere Teilreflexionen möglich, die ihrerseits durch Teilreflexionen weiter aufgespaltet werden und hin und her reflektiert werden.

Das gleiche Phänomen kann man bei elektrischen Leitungen beobachten. Bei Unstetigkeiten in der Kapazität oder Induktivität der Leitung können Reflexionen entstehen, die die Signalqualität erheblich beeinträchtigen können. Solche Unstetigkeiten können durch Variationen in der Leitungsdicke, schwankende Materialeigenschaften oder durch an der Leitung angeschlossene zusätzliche Empfänger oder Sender verursacht werden. Die Wellenreflexionen stellen eines der Hauptprobleme bei der Implementierung von elektrischen Broadcast-Kanälen mit hoher Bandbreite und hohem Fanout dar. Je mehr Sender und Empfänger an einer Leitung angeschlossen sind, um so schwieriger wird es, Unstetigkeiten zu vermeiden.

Übersprechen und Verluste durch Abstrahlung: Die Erzeugung von Ladungsverdichtungen bei der Signalübertragung ist mit der Bewegung und Beschleunigung von Ladungsträgern verbunden. Dabei wird Energie in Form elektro-magnetischer Wellen abgestrahlt. Diese Strahlung führt zum einen zur Abschwächung des Signals. Zum anderen verursacht sie ein Übersprechen mit benachbarten Leitungen, in denen nach dem Induktionsgesetz durch die Strahlung Störströme induziert werden. Die abgestrahlte Leistung steigt quadratisch mit der Datenrate und linear mit der Leitungslänge.

Bandbreitenbeschränkung durch den Skin-Effekt: Eine weitere Folge der Beschleunigung von Ladungen stellt der sog. Skin-Effekt dar. Er wird durch sog. magnetische Wirbelfelder hervorgerufen, die oszillierende Ladungsströme in einem Leiter erzeugen. Diese Felder sorgen dafür, daß die bewegten Ladungen in die Außenhaut des Leiters verdrängt werden, und zwar mit einer zu ihrer Geschwindigkeit proportionalen Kraft. Da die Geschwindigkeit der Oszilation von der Bandbreite abhängt, fließt der Strom mit steigender Datenrate immer mehr nur in der Außenhaut des Leiters. Dadurch steigt der effektive Leitungswiderstand, da der für die Stromleitung verfügbare Querschnitt sinkt. Mit steigender Datenrate kommt also ein immer geringerer Teil der gesendeten Leistung beim Empfänger an. Dies bedeutet, daß es bei vorgegebener Senderleistung eine Grenz-Datenrate gibt, ab der keine fehlerfreie Signalübertragung möglich ist.

Störungen durch gemeinsame Spannungsversorgung: Wenn ein Sender ein starkes Signal senden möchte, dann muß dieses Signal mit einem hohen Strom verbunden sein. Dieser hohe Strom muß der Schaltung über die Spannungsversorgung zugeführt und über die GND-Leitung wieder abgeführt werden. Gleiches gilt, wenn mehrere Sender gleichzeitig ein schwaches Signal senden. Da die Versorgungs- und GND-Leitung einen von 0 verschiedenen Widerstand haben, führt der hohe Strom zu einem Spannungsabfall an der Versorgungsleitung und einem Spannungsanstieg an der GND-Leitung. Dieser Spannungsabfall beeinträchtigt die Funktion aller anderen Systemkomponenten, die auf eine konstante Versorgungsspannung angewiesen sind. Er führt unter anderem auch dazu, daß an freien Leitungen 'Scheinsignale' registriert werden.

Quadratischer Latenzanstieg auf kurzen Strecken: Bei Leitungen, die kürzer als die Wellenlänge eines Signals sind, findet keine echte Wellenausbreitung statt. Bleibt man bei der mechanischen Analogie, so korrespondiert eine solche Leitung zu einem einzelnen Oszillator. Die Übertragung eines Signals entspricht der Auslenkung des Oszillators aus der Ruhelage. Bei vorgegebener Kraft ist die Auslenkgeschwindigkeit umgekehrt proportional zur Masse. Analog ist die elektrische Signalgeschwindigkeit proportional zur Kapazität der Leitung. Da die Kapazität der Leitung aber linear mit der Leitungslänge zunimmt, ist die Latenz bei konstanter Signalstärke proportional zum Quadrat der Leitungslänge.

Begrenzte Verbindungskomplexität: Die Komplexität einer elektrischen Verbindungstopologie ist durch die Tatsache beschränkt, daß für die Signalübertragung Leitungen benötigt werden, die sich nicht ohne weiteres kreuzen können. Dies gilt insbesondere für VLSI-Schaltkreise bzw. Platinen, auf denen für die Leitungen nur eine beschränkte Anzahl von Leitungslagen zur Verfügung stehen. Für solche Systeme wurde in [43] gezeigt, daß die Fläche, die für eine Netzwerk mit einer Bisektionsbreite B benötigt wird, proportional zu B^2 ist.

4.1.2 Optische Datenübertragung

Bei der optischen Datenübertragung werden Signale mit Hilfe von Lichtimpulsen verschickt. Physikalisch gesehen werden also für die Datenübertragung hochfrequente elektro-magnetische Wellen verwendet. Aus verschiedenen Gründen wird dabei in der Regel sog. monochromatisches Licht benutzt. Dies bedeutet, daß in dem Strahl nur eine bestimmte Frequenz und damit nur eine bestimmte Wellenlänge (Farbe) vertreten ist. Diese Frequenz (ν) liegt bei den heutigen Systemen zwischen 10^{14} und 10^{15} Herz (also zwischen einhunderttausend und eine Million Ghz). Dies entspricht rötlichem bis infraroten Licht. Die Ausbreitungsgeschwindigkeit (c) beträgt ca. $3 \times 10^8 m/s = 30 cm/ns$, wobei der genaue Wert vom Ausbreitungsmedium abhängt. Damit folgt für eine Wellenlänge des verwendeten Lichts λ ein Wert zwischen ca. $500 nm$ und $1,5 \mu m$.

Für die Beschreibung der optischen Signalübertragung ist die Tatsache entscheidend, daß die Frequenz der Lichtwellen um mehrere Größenordnungen über der Datenübertragungsfrequenz liegt. Die optische Datenübertragung kann daher als eine Amplitudenmodulation einer Trägerwellenfrequenz betrachtet werden (Abbildung 4.3). Beim monochromatischen Licht handelt es sich bei der Trägerwelle um eine harmonische Welle fester Frequenz.

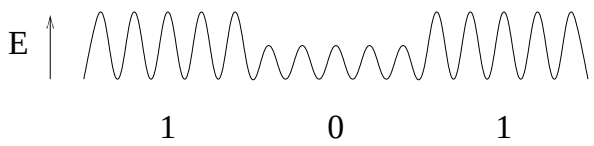


Abbildung 4.3: Das Prinzip der optischen Datenübertragung durch Modulation einer Trägerwelle.

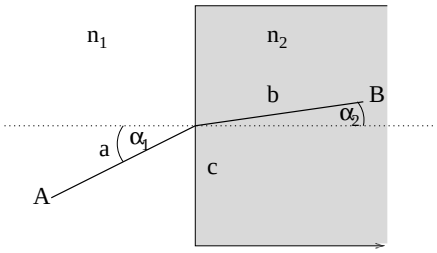


Abbildung 4.4: Das Prinzip der Brechung eines Lichtstrahls an der grenze zweier Medien mit unterschiedlichen Brechungsindizies n_1 und n_2.

Die Signalqualität hängt nur von der Frequenz der Trägerwelle (Lichtfarbe) und den Eigenschaften des Übertragungsmediums ab. Anderes als bei der Elektronik werden sie weder von der Datenrate noch vom Inhalt der übertragenen Nachricht beeinflußt.

Bei der Beschreibung der Ausbreitung von Lichtsignalen müssen zwei Aspekte berücksichtigt werden: die lokale Wechselwirkung mit Materie und die Folgen dieser Wechselwirkung auf die globale Ausbreitung des Lichts. Im Nachfolgenden werden nach einer einfachen Betrachtung der Wechselwirkung mit Materie zwei Näherungsverfahren zur Beschreibung der Lichtausbreitung skizziert: die Strahlenoptik und die Fourieroptik.

Wechselwirkung mit Materie

Die Übertragungseigenschaften eines Mediums für das Licht einer bestimmten Wellenlänge können in guter Näherung durch drei Parameter beschrieben werden: den Reflexionskoeffizienten, den Absorbtionskoeffizienten und den Brechungsindex. Ersterer gibt an, welcher Anteil des auf die Oberfläche des Materials einfallenden Lichts reflektiert wird. Der Absorbtionskoeffizent bestimmt, wieviel Licht pro Längeneinheit in dem Medium absorbiert wird. Der Brechungsindex gibt schließlich das Verhältnis der Lichtgeschwindigkeit im Medium zur Lichtgeschwindigkeit im Vakuum an.

Beschreibung der Signalausbreitung durch die Strahlenoptik

Die Strahlenoptik nimmt an, daß man die Lichtausbreitung durch die Ausbreitung von inifinitisemal schmalen, linienförmigen Strahlen annähern kann [66]. Sie beschäftigt sich mit Systemen aus drei Arten von Komponenten: vollständig reflektierende Flächen beliebiger Form, vollständig absorbierende Flächen beliebiger Form, sowie durchsichtige Bereiche mit unterschiedlichem Brechungsindex und beliebig geformten Grenzflächen. Die Ausbreitung der Strahlen durch solche Systeme wird durch drei Regeln beschrieben.

1. Trifft ein Strahl auf eine absorbierende Fläche so endet er dort und propagiert nicht weiter.
2. Trifft ein Strahl auf eine reflektierende Oberfläche, so wird er unter dem Auftreffwinkel reflektiert.
3. Ein zwischen zwei Punkten verlaufender Lichtstrahl folgt immer dem Weg mit der kürzesten Laufzeit.

Die ersten beiden Regeln bedürfen keiner weiteren Erläuterung. Aus der dritten folgt, daß sich das Licht in einem homogenen Medium geradlinig ausbreitet. Beim Durchgang durch Gebiete mit unterschiedlichen Brechungsindex wird es dagegen so abgelenkt, daß der Weg durch Bereiche mit geringer Ausbreitungsgeschwindigkeit minimiert wird. Durch eine entsprechende Gestaltung der einzelnen Gebiete und deren Grenzflächen kann man so einen bestimmten Strahlenverlauf erzwingen. Dies ist in Abbildung 4.4 für die Brechung eines Strahls an einer geraden Grenzfläche zwischen zwei Medien dargestellt. Wir gehen dabei davon aus, daß die Ausbreitungsgeschwindigkeit im rechten Medium geringer ist (der Brechungsindex größer ist). Der Weg von A nach B über die Strecken a und b ist zwar länger als der gerade Weg über c, dafür wird aber eine kürzere Strecke innerhalb des rechten

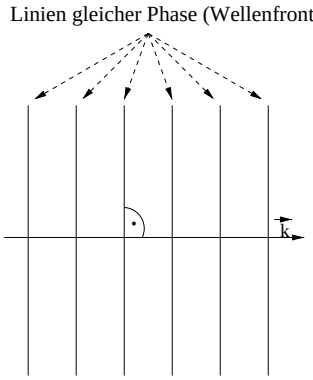


Abbildung 4.5: Die Abbildung zeigt einen Ausschnitt einer zweidimensionalen ebenen Welle. Die gesamte Welle hat eine unendliche Ausdehnung.

Abbildung 4.6: Der Einfluß verschiedener optischer Elemente auf eine ebene Welle. Bild a) zeigt Brechung an der Grenzfläche zwischen zwei Medien mit unterschiedlichen Brechungsindizies. Es entspricht der Brechung eines parallelen Strahlenbündels. Im Bild b) ist die Brechung durch Graustufen bzw. Phasenmuster zu sehen.

Mediums zurückgelegt. Mit einfachen geometrischen Überlegungen läßt sich zeigen, daß für das Verhältnis des Einfallswinkels α_1 zum Ausfallswinkel α_2 die folgende Formel gilt:

$$\frac{\sin(\alpha_1)}{\sin(\alpha_2)} = \frac{n_1}{n_2} \tag{4.1}$$

Dabei bezeichnen n_1 und n_2 die Brechungsindizies der beiden Medien. Die obige Formel gilt für beliebige Medien, Auftreffwinkel und Grenzflächen (bei gekrümmten Flächen wird die Tangente im Auftreffpunkt betrachtet). Sie erlaubt die Herleitung des Strahlengangs durch alle klassischen optischen Komponenten wie Linsen und Prismen.

Beschreibung der Signalausbreitung durch die Fourieroptik

Die Strahlenoptik berücksichtigt nur sehr unzureichend die Wellennatur des Lichts. Sie kann keine Phänomene erklären, die mit Beugung und Interferenz zu tun haben. Diese Phänomene sind aber sowohl für die Leistungsgrenzen optischer Systeme als auch für die Funktionsweise vieler moderner optischer Komponenten von entscheidender Bedeutung.

Eine gute Basis für das Verständnis solcher Phänomene bildet die Fourieroptik [157]. Sie basiert auf der Betrachtung sog. ebener Wellen. Es handelt sich dabei um unendlich ausgedehnte, gerade, dreidimensionale Wellenfronten. Das Prinzip einer solchen Welle ist schematisch anhand einer zweidimensionalen Welle in Abbildung 4.5 dargestellt. Der Unterschied zu einer dreidimensionalen Welle besteht darin, daß dreidimensionale Orte gleicher Phase (z.B. Wellenberge) keine Linien sondern eine Ebene sind. Eine ebene Welle wird durch zwei Parameter beschrieben: ihre Frequenz und die Ausbreitungsrichtung. Sie entspricht einem unendlich breiten parallelen Strahlenbündel, dessen Strahlen senkrecht auf den Ebenen gleicher Phase stehen.

Die Fourieroptik geht davon aus, daß eine monochromatische Lichtwelle durch eine Fourierzerlegung in Elementarwellen, sog. ebene Wellen unterschiedlicher Ausbreitungsrichtung aufgespaltet werden können. Die Frequenz aller dieser Wellen ist mit der Frequenz der Ursprungswelle identisch. Die Zerlegung in Ebenen gilt für einen einfachen Laserstrahl genauso wie für das Bild einer Landschaft, das man durch ein Fenster sieht. Die Eigenschaften

eines optischen Systems auf eine Lichtwelle werden anhand der Wirkung des Systems auf die einzelnen ebenen Wellen ermittelt. Diese Wirkung kann nach folgenden drei Regeln ermittelt werden:

1. Eine ebene Welle wird bei der Ausbreitung durch ein homogenes Medium nicht verändert.

2. Bei Durchgang durch die Grenzfläche zweier Medien unterschiedlicher Brechungsindizies wird die Welle 'verbogen'. Die Verbiegung entsteht dadurch, daß die Teile der ebenen Welle, die zuerst die Grenzfläche erreichen, als erste ihre Ausbreitungsgeschwindigkeit ändern. Sie folgt damit der Form der Grenzfläche. Dies ist in Abbildung 4.6 links zu sehen. Dabei wird auch klar, daß sich aus dieser Regel das Brechungsgesetz der Strahlenoptik ableitet.

3. Der Einfluß einer Fläche mit einem beliebigen Graustufen- oder Phasenmuster auf eine ebene Welle ist durch die Fouriertransformation des Musters gegeben. Dazu wird eine Zerlegung des Musters in zweidimensionale Wellen unterschiedlicher Frequenz betrachtet. Die ebene Welle wird beim Durchgang durch die Fläche in mehrere ebene Wellen aufgefächert, von denen jeder zu einer Fourierkomponente des Musters korrespondiert. Die Intensität jeder dieser Wellen ist proportional zur Intensität der korrespondierenden Fourierkompenete im Muster. Die Korrespondenz basiert auf einer Abbildung der räumlichen Frequenzen der Musterwellen auf Winkel. Diese Abbildung ordnet jeweils der Frequenz ν den Winkel θ mit

$$sin(\theta_\nu) = \lambda \cdot \nu \tag{4.2}$$

zu, wobei λ die Wellenlänge des Lichts bezeichnet. Je höher also die Frequenz, um so höher ist der korrespondierende Winkel. Der Winkel jeder der erzeugten ebenen Wellen setzt sich zusammen aus dem Winkel der ursprünglichen Welle und dem, nach der obigen Gleichung ermittelten Winkel. Dies ist in Abbildung 4.6 dargestellt. Ein allgemeines Muster erzeugt viele verschiedene ebene Wellen. Ein Muster, das nur aus einer Fourierkomponente besteht, verändert lediglich den Winkel der einfallenden ebenen Welle um einen Betrag, der um so größer ist, je höher die Frequenz der Fourierkomponente ist.

Eigenschaften der Datenübertragung

Als Datenübertragungsmedium hat das Licht den elektrischen Signalen gegenüber die folgenden Vorteile:

Hohe Zeitbandbreite: Da die Datenrate des Signals keinen Einfluß auf die Übertragungseigenschaften hat, erlaubt die Optik sehr hohe Bandbreiten. Die Datenrate ist im Prinzip nur durch die Frequenz des Lichts beschränkt. Damit die Übertragung als Amplitudenmodulation einer Trägerwelle betrachtet werden kann, muß die Signalfrequenz um einen Faktor von 10 bis 100 über der Lichtfrequenz liegen. Damit ergibt sich für die Bandbreite eines optischen Kanals eine obere Grenze von ca. 10 bis 100 TBit/s.

Hohe Ortsbandbreite: Zusätzlich zu der hohen Datenrate einzelner Kanäle erlaubt die optische Datenübertragung eine hohe räumliche Kanaldichte. Da im Gegensatz zur Elektronik die Datenrate keinen Einfluß auf das Übersprechen benachbarter Kanäle hat, ist auch die Kanaldichte von der Datenrate der einzelnen Kanäle unabhängig. Sie ist lediglich durch die Wellenlänge und den Durchmesser des optischen Systems beschränkt. Sie zeigen den kleinsten Durchmesser eines Punktes d_{min}, der vom System noch sauber aufgelöst werden kann als:

$$d_{min} = \frac{1,22 f \lambda}{D} \tag{4.3}$$

Dabei sind f die Brennweite der benutzten Linse, D ihr Durchmesser und λ die Wellenlänge.

Mehrere Trägerwellenfrequenzen (Lichtfarben) sind möglich: Ein großer Vorteil optischer Datenübertragung stellt die Möglichkeit dar, unterschiedliche Wellenlängen als unabhängige Datenkanäle zu benutzen. Übertragungssysteme, die diese Möglichkeit nutzen, werden als WDM-Systeme (für wavelength division multiplexing), bezeichnet. In der Praxis wird das WDM-Verfahren benutzt, um die Bandbreite und die Kanalanzahl zu steigern. Es kann allerdings die theoretische Bandbreite eines optischen Systems nicht verändern. Dies liegt an der Unschärferelation, die für das Produkt aus der Zeit- und Frequenzunschärfe eines Lichtimpulses einen konstanten Wert vorgibt.

Verlustarme Ausbreitung: Ein Großteil der Verluste bei der elektrischen Datenübertragung hängt damit zusammen, daß sie mit Hilfe massive, geladener Teilchen durchgeführt wird. Als elektromagnetische Welle

besitzt das Licht dagegen weder Masse noch Ladung. Es wird im Teilchenbild mit Hilfe neutraler masseloser Partikel dargestellt: der Photonen. Dadurch ist es möglich, Medien mit einer sehr geringen Absorbtion zu finden. Sie bewegt sich im Bereich von 0.1dB/Km, d.h., daß das Signal beim Durchlaufen eines 1km dicken Mediums gerade mal um 1% geschwächt wird. Hinzu kommt, daß es keine Verluste durch Abstrahlung gibt.

Hohe Signalqualität: Die optische Datenübertragung zeichnet sich durch eine hohe Signalqualität aus. Bei den für die relevanten Entfernungen $\leq 20m$ ist die Qualität des Signals in guter Näherung von der Länge der Übertragungsstrecke und dem Fanout unabhängig. Die Signale werden weder durch Abstrahlung verzerrt noch gibt es optische Analogien zum Skin-Effekt. Ebenso entfällt das Problem der unterschiedlichen Übertragungseigenschaften der einzelnen Fourierkomponenten der Nachricht, da die Übertragungseigenschaften nur von der Frequenz der Trägerwelle abhängig sind.

Geringes Übersprechen: Da zwei die Photonen, anderes als zwei Elektronen, nicht miteinander wechselwirken gibt es bei der optischen Datenübertragung wesentlich weniger Probleme mit Übersprechen. Von besonderer Bedeutung sind dabei die folgenden Punkte.

1. Das Maß des Übersprechens hängt bei der optischen Datenübertragung nicht von der Datenrate ab.
2. Bei den für die Rechnertechnik relevanten Entfernungen gibt es bei der Übertragung in Fasern ('Lichtkabeln, siehe Abschnitt 4.2.4) gar kein Übersprechen zwischen benachbarten Kanälen. Dies gilt sowohl für viele, dicht gepackte Einzelfasern als auch für die WDM-Übertragung in einer Faser.

Ein gewisses Problem stellt in der Optik das Übersprechen in Systemen dar, in den die Signale mit hoher Dichte im freien Raum übertragen werden. Dies hat zwei Gründe: die Wellennatur des Lichtes und die Fehler optischer Abbildungselemente (Abberationen). Die Auswirkungen dieser Faktoren auf das Übersprechen in einem System hängen von der Art des Systems ab, und können nicht auf eine einfache allgemeingültige Formel gebracht werden. Eine genaue Behandlung des Übersprechens in verschiedenen optischen Systemen würde den Rahmen der vorliegenden Arbeit sprengen. Wir belassen es hier daher bei der Feststellung, daß trotz der oben genannten Faktoren optische Freiraumsysteme mit mehreren Tausend Kanälen pro mm^2 realisiert werden können, und zwar ohne das die Signalqualität von der Datenrate abhängt.

Hohe Verbindungskomplexität: Da Lichtsignale im freien Raum verlaufen können und sich dabei beliebig kreuzen können, sind komplexe Verbindungstopologien optisch deutlich einfacher zu implementieren als elektrisch. Es kann gezeigt werden, daß ein Netzwerk mit der Bisektionsbreite B auf einer Grundfläche $O(B)$ implementiert werden kann. In der Elektronik wird hierfür eine Fläche von $O(B^2)$ benötigt.

4.2 Optische Datenübertragungssysteme

Ein optisches Datenübertragungssystem besteht aus einer Menge von Sendern, einer Menge von Empfängern und einem optischen System, das beide miteinander verbindet. Bei den Sendern und Empfänger handelt es sich heute in der Regel um Halbleiterdioden. Das optische System kann auf zwei Arten implementiert werden: als Freiraumsystem oder als Lichtleiter-System. Im vorliegenden Abschnitt wird ein Überblick über die heute verwendeten Komponenten, deren Funktionsweise und Leistungsschranken gegeben. Ich beschränke mich dabei auf die Technologie, die heute kommerziell verfügbar und auf breiter Front eingesetzt wird. Damit soll das Verständnis der Gründe ermöglicht werden, die bisher den Einsatz der Optik in der Rechnerarchitektur verhindert haben. Neue technologische Entwicklungen werden in den Abschnitten 4.5 bis 4.7 besprochen.

4.2.1 Optische Sender

Ein optischer Sender muß digitale elektronische Signale in Lichtsignale wandeln. Dies kann auf zwei Arten geschehen: durch Lichtquellen oder durch Lichtmodulatoren. Im Nachfolgenden werden die heute üblichen Komponenten beschrieben und deren Leistungsschranken aufgezeigt.

Lichtquellen

Eine Lichtquelle, die als Sender funktioniert, wandelt digitale elektronische Signale in Lichtimpulse um. Dazu werden zwei Komponenten benötigt: ein Bauelement zur Lichterzeugung, und einen elektronischen Treiberbaustein. Der letztere generiert aus den digitalen Signalen die von dem Wandler benötigte Stromstärke und Spannung.

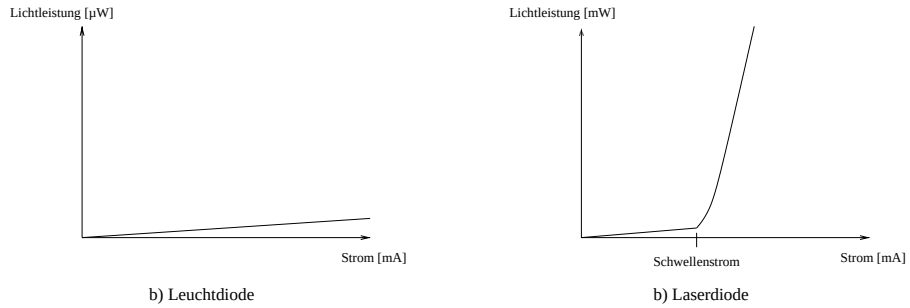


Abbildung 4.7: Die Abhängigkeit der emittierten Lichtleistung von der Stromstärke bei einer LED (Abbildung a) und bei einer Laserdiode (Abbildung b).

Für die Lichterzeugung werden in der Regel Halbleiterdioden verwendet. Sie nutzten die Tatsache aus, daß die Ladungsträger beim Übergang zwischen den unterschiedlich dotierten Halbleiterschichten ihre Energieniveaus wechseln. Die Energiedifferenz wird bei solchen Übergängen meist in Form von Licht freigesetzt.

LED: Eine LED ist eine einfache Halbleiterdiode, bei der die dotierten Bereiche so gewählt wurden, daß ihre Energiedifferenz der benötigten Lichtwellenlänge entspricht. Wie in Abbildung 4.7 gezeigt, ist die emittierte Lichtleistung zunächst proportional zum Strom. Die Steigung, die die I/W-Kurve bestimmt, gibt die Effizienz der LED an. Sie liegt typischerweise bei ca. 10 bis 100 $\mu W/mA$. D.h. daß nur ca. 1% der der LED zugeführten Energie in Licht umgewandelt wird. Der Rest wird in Form von Wärme dissipiert.

Wird die Stromstärke über einen gewissen Wert gesteigert, so setzt ein Sättigungsprozess ein, bis irgendwann eine weitere Erhöhung der Lichtleistung nicht mehr möglich ist. Diese Sättigungsgrenze bestimmt die maximale Ausgangsleistung der LED. Sie liegt meist bei 0.1 bis 0.5 mW.

LD: Die Laserdioden nutzten einen Effekt, der induzierte Emission genannt wird. Er bewirkt, daß die Wahrscheinlichkeit für die Emission von Licht einer bestimmten Wellenlänge durch die Anwesenheit des Lichts gleicher Wellenlänge vergrößert wird. Die Lichtleistung einer LED kann also dadurch gesteigert werden, daß der aktive (lichtemitierende) Bereich mit Licht der passenden Wellenlänge bestrahlt wird.

Um die induzierte Emission zu nutzen, besteht eine LD aus einem nichtlinearen Resonator. Der Resonator ist i.a. ein sog. Fabry-Perot-Resonator [180] mit wenigstens einem teildurchlässigen Spiegel. Bei LDs für die Datenübertragung werden die 'Spiegel' häufig als Gitter implementiert. Bei einfachen Bauteilen wird auch einfach die Reflexion beim Übergang vom Halbleitermaterial in Luft ausgenutzt. Diese Reflexion wird durch den unterschiedlichen Brechungsindex des Halbleiters und der Luft verursacht.

Die Spiegel reflektieren einen Teil des emittierten Lichts zurück in den aktiven Bereich. Das zurückreflektierte Licht induziert weitere Lichtemission und bewirkt eine Rückkopplung, die zu einem steilen Anstieg der Lichtleistung führt. Dies ist in dem I/W Graphen einer Laserdiode in Abbildung 4.7 zu sehen. Bei geringer Stromstärke verhält sich eine LD wie eine LED. Ab einem gewissen Strom macht sich dann die Rückkoppelung bemerkbar. Dieser Strom wird als Schwellenstrom bezeichnet und ist eine wichtige Kenngröße. Oberhalb des Schwellenstroms hat eine LD eine sehr hohe Effizienz. Sie liegt typischerweise bei ca. $0,6W/A$. Dabei werden bis zu 60% der Energie in Licht umgewandelt. In der Praxis ergibt sich die obere Leistungsstärke dadurch, daß ab einer bestimmten Lichtleistung die halbdurchlässigen Spiegel zerstört werden. Sie beträgt zur Zeit einige wenige Watt. Bei den in der Datenübertragung heute am meisten verwendeten sog VCSEL (vertical cavity surface emmiting laser) LDs, die nachfolgend noch genauer betrachtet werden, liegt diese allerdings Grenze bei einigen wenigen mW.

Modulatoren

Ein Lichtmodulator verändert ausgewählte Eigenschaften (z.B. die Intensität) des einfallenden Lichts in Abhängigkeit von einem Steuersignal. Die vermutlich bekannteste Variante von Lichtmodulatoren sind Flüssigkristalle, die unter anderem in Anzeigen Anwendung finden. Ein großer Vorteil des Lichtmodulators als Sender besteht darin, daß die zum Schalten notwendige eletrische Leistung von der Signalleistung entkoppelt ist. So ist die Leistung, die man zum Aufbau eines Bildes auf einem Flüssigkeitsdisplay braucht, unabhängig von der Intensität des einfallenden Lichts. Hinzu kommt die sog. optische Transparenz. Dies bedeutet, daß der Schalter eine bestimmte

Eigenschaft des Lichts moduliert, die anderen aber unverändert läßt. So könnnen mit einem einzigen Modulator z.B. mehrere Signale auf unterschiedlichen Wellenlängen moduliert werden.

Trotz der oben genannten Vorteile ist heute das Einsatzgebiet von Modulatoren bei der Datenübertragung begrenzt. Sie werden zur Zeit nur bei der Fernübertragung (über Hunderte von Kilometern) verwendet. Dort wird die Tatsache ausgenutzt, daß man durch Modulatoren die Verbreiterung des Lichtspektrums vermeiden kann, die die direkte Modulation einer LD oder LDs mitsichbringt. Diese Verbreiterung stellt bei sehr langen Übertragungsstrecken ein erhebliches Problem dar. Bei der Übertragung über kürzere Strecken spiele Modulatoren dagegen zur Zeit praktisch keine Rolle. Dies liegt daran, daß es bisher keine kompakten Komponenten gab, die mit geringem Strom und geringer Spannung Datenraten von mehr als 1 bis 10KBit/s erreichen konnten. Statt dessen werden direkt modulierte LDs und LEDs benutzt.

Leistungsschranken

Die Bandbreite von LED- und LD-basierten Sendern ist durch zwei Faktoren bestimmt: die Zeit, die die LED oder LD für die Veränderung der Lichtstärke braucht, und die Zeit, die der Treiber für die zugehörige Veränderung der Stromstärke benötigt.

Bei LEDs ist ersteres für die Bandbreitengrenzen verantwortlich. Die maximale Datenrate liegt dabei um 1Gbit/s. Bei Laserdioden ist theoretisch eine Modulation im Bereich von mehreren Hundert GHz möglich. In der Praxis scheitert eine so schnelle Modulation jedoch an der Leistung der Treiberbausteine, die als Halbleiterverstärker an die Schaltgeschwindigkeiten von Transistoren gebunden sind. So können mit Hilfe spezieller GaAs Treiber heute maximal 10Gbit/s erreicht werden. Konventionelle Silizium CMOS Treiber ermöglichen Datenraten zwischen 1 und 2 Gbit/s.

4.2.2 Optische Empfänger

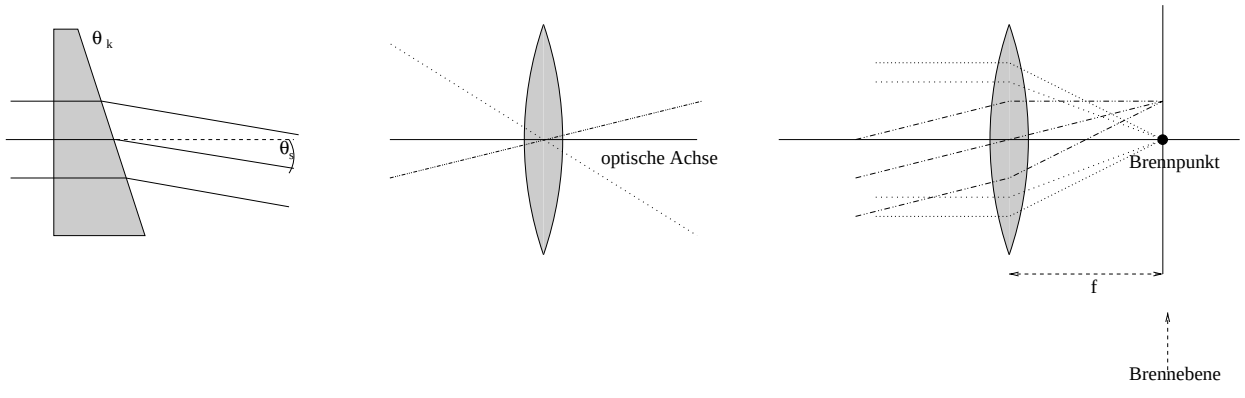
Optische Empfänger müssen Lichtimpulse in Stromimpulse umwandeln und diese digitalisieren. Sie bestehen aus zwei Komponenten: einem Bauteil zur Umwandlung von Licht in Strom, und einem elektronischen Empfänger zur Anpassung des Signals an das digitale Spannungsniveau.

Als Licht/Strom-Wandler werden meistens in Sperrichtung gepolte Halbleiterdioden, sog. Photodioden, benutzt. Sie nutzen die Tatsache aus, daß durch die Lichteinstrahlung Ladungsträger im Halbleiter in einen energetisch höheren Zustand gehoben werden. Dies versetzt sie in die Lage, die für den Sperreffekt verantwortliche Potentialbarierre zu überwinden und einen Strom in Sperrrichtung zu erzeugen. Dieser Photostrom ist proportional zur Lichtintensität. Der Proportionalitätsfaktor wird als Quanteneffizienz bezeichnet und gibt an, wieviel Prozent der einfallenden Photonen von den Ladungsträgern absorbiert werden. Der Strom bzw. der damit verbundene Spannungsabfall wird von dem elektronischen Empfänger verstärkt, mit einem Schwellenwert verglichen und dementsprechend als 0 oder 1 interpretiert. Der Entwurf effizienter schneller Verstärker ist ein aktuelles Forschungsthema in der Elektronik. In der Regel bestehen solche Bausteine aus mehreren hintereinandergeschalteten Transistor-Verstärkerstufen.

Leistungsgrenzen

Die fundamentale Schranke für die Bandbreite einer Photodiode stellt die Zeit dar, die Ladungsträger für den Energieübergang und die Durchquerung der Sperrschicht benötigen. Sie liegt je nach Art der Diode zwischen 10ps und 100ps. Dies entspricht einer Datenrate von 10 bis 100Gbit/s. Eine weitere Beschränkung der Bandbreite ergibt sich aus der Notwendigkeit, bei der verfügbaren Signalstärke eine wohldefinierte maximale Fehlerrate zu garantieren. Dazu muß berücksichtigt werden, daß die Lichtdetektion in der Photodiode einen statistischen Prozess darstellt. So führt einerseits nicht jedes auf die Photodiode einfallendes Lichtquant dazu, daß ein Ladungsträger in ein höheres Energieniveau angehoben wird und die Sperrschicht überqueren kann. Gleichzeitig gibt es auch Ladungsträger, die, ohne daß sie einen Lichtquant absorbiert haben, zufällig die für das Überqueren der Sperrschicht notwendige Energie erlangen. Damit trotz dieses Rauschens mit einer hohen Wahrscheinlichkeit der richtige Wert empfangen wird, muß also in der Photodiode eine bestimmte minimale Anzahl von Photonen ankommen. Damit ist eine minimale Lichtenergie W gegeben, die zum Empfang eines Signals notwendig ist. Bei einer vorgegebenen Lichtleistung des Signals P ergibt sich daraus die für den Empfang notwendige Zeit t als

$$t = \frac{P}{W} \tag{4.4}$$



a) Parallele Strahlen im Keilprisma b) Mittelpunktsstrahlen einer Linse b) Parallele Strahlen in einer Linse

Abbildung 4.8: Die wichtigsten refraktiven optischen Bauelemente. Das linke Bild zeigt den Strahlenverlauf durch ein Keilprisma. Das mittlere und das linke Bild illustrieren die beiden Regeln, mit den man den Strahlenverlauf durch eine Linse bestimmen kann.

Um die in der Datenverarbeitung übliche Fehlerrate von BER= 10^{-9} zu ermöglichen, sind in den meisten Photodioden einige Tausend Photonen notwendig.

In der Praxis ist die Bandbreite durch die Leistungsfähigkeit der elektronischen Empfänger beschränkt. Zum einen verursachen sie ein zusätzliches Rauschen, das die benötigte minimale Signalstärke weiter anhebt. Zum anderen sind sie an die Schaltzeit der Transistoren in den Verstärkerstufen gebunden, die bei einigen Hundert Pikosekunden (ps) liegt.

4.2.3 Optische Signalübertragung im freien Raum

Ein optisches Freiraumsystem besteht aus Feldern von Sendern und Empfängern und einem Abbildungssystem. Das Abbildungssystem sorgt dafür, daß das Licht eines jeden Senders zu den durch die Verbindungstopologie vorgegebenen Empfängern gelangt. Es besteht in der Regel aus mehreren optischen Komponenten für die Strahlablenkung, die mit Hilfe eines mechanischen Systems zu einer stabilen Einheit zusammengefügt sind. Im Nachfolgenden wird zunächst die Funktionsweise der beiden wichtigsten Arten optischer Komponenten erläutert. Danach werden die Probleme des mechanischen Aufbaus und die Leistungsschranken von Freiraumsystemen besprochen.

Refraktive Komponenten

Unter refraktiven Komponenten versteht man Bauteile, die für die Strahlablenkung die Berechung von Lichtstrahlen an Grenzflächen von Materialien mit unterschiedlichen Brechungsindizies nutzen (siehe 4.1.2). Die wichtigsten refraktiven Komponenten sind Prismen und Linsen. Ihre Funktion kann wie folgt zusammengefaßt werden:

Prismen: Ein Prisma ist ein Bauteil mit mehreren ebenen Stirnflächen. Das bekannteste Prisma ist das sog. Keilprisma (Abbildung 4.8), das dem Namen entsprechend die Form eines Keils besitzt. Es lenkt ein paralleles Strahlenbündel um einen bestimmten, von Keilwinkel und Einfallswinkel abhängenden Winkel ab. Betrachtet man ein Bild durch ein solches Prisma, so erscheint das Bild um eine bestimmte Strecke verschoben. Das Ausmaß der Verschiebung hängt von dem Keilwinkel ab.

Linsen: Wie in Abbildung 4.8 zu sehen, sind Linsen flache, symmetrische Bauelemente mit mindestens einer runden Stirnfläche. Ihre Wirkung auf den Strahlenverlauf kann näherungsweise durch zwei Regeln beschrieben werden:

1. Ein Strahl, der durch die Mitte der Linse geht (Mittelpunktsstrahl), wird von der Linse nicht beeinflußt.
2. Zwei zueinander parallele Strahlen, die auf der einen Seite auf die Linse auftreffen, werden so abgelenkt, daß sie sich auf der anderen Seite in der sog. Brennebene der Linse schneiden. Wie in Abbildung 4.8 zu sehen, ist die Brennebene eine zu der Stirnfläche tangentiale Ebene. Die Entfernung der Brennebene von der Linse ist eine für jede Linse charakteristische Größe, die als die Brennweite f bezeichnet wird.

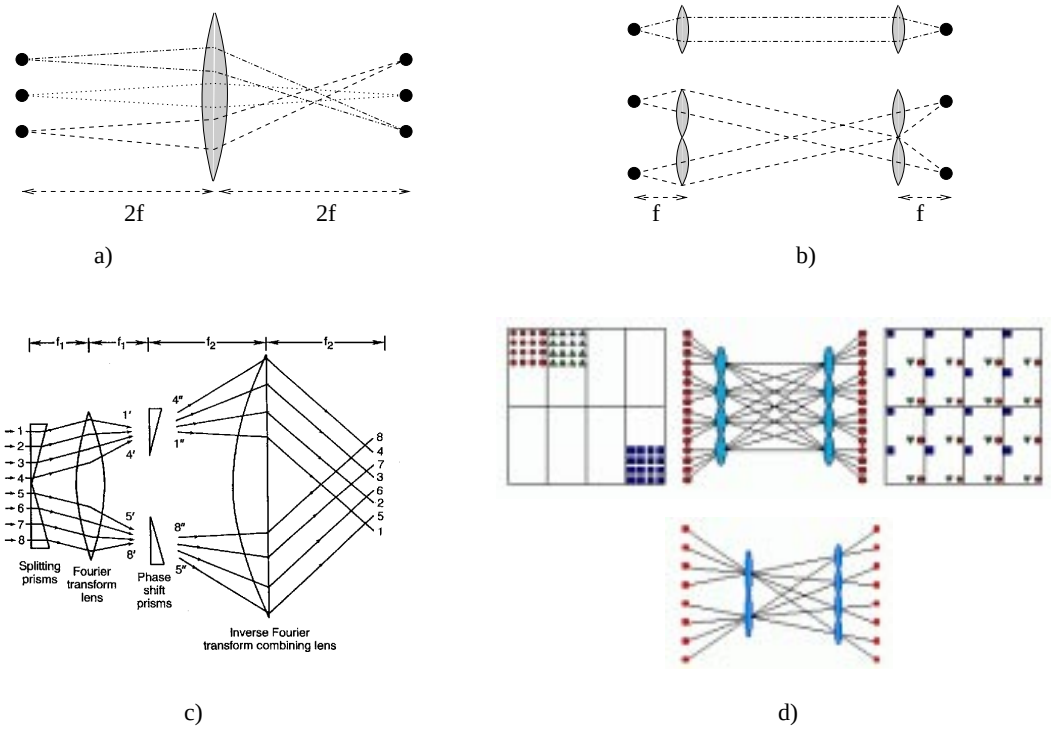


Abbildung 4.9: Beispiele für die realisierung verschiedener Verbindungsstrukturen mit Hilfe von Linsen und Prismen. Die Systeme sind im Text beschrieben.

> Um den Weg eines Strahls durch die Linse zu bestimmen, muß man demnach zuerst einen parallelen Strahl durch die Mitte der Linse zeichnen und seinen Schnittpunkt mit der Brennebene bestimmen. Aus der zweiten Regel folgt dann, daß auch der gesuchte Strahl die Brennebene in diesem Punkt schneiden muß.

Um den Einsatz refraktiver Komponenten in Verbindungssystemen zu veranschaulichen, wurden in Abbildung 4.9 vier verschiedene Abbildungssysteme dargestellt. Das System in Bild a) zeigt ein Abbildungssystem für eine reguläre Punkt-zu-Punkt Verbindung zwischen drei Empfängern und drei Sendern mit Hilfe einer großen Linse. Das in Bild b) dargestellte System kann mit Hilfe eines Linsenfeldes eine beliebige Punkt-zu-Punkt Verbindungstopologie implementieren. Die Ablenkung des Senderstrahls wird dabei durch die laterale Verschiebung der kleinen Linsen in Bezug auf den Sender erreicht. Die Abbildungen c) und d) sind Beispiele aus der Literatur. Abbildung c) zeigt einen Entwurf für ein optisches Shuffle-Netzwerk [114]. Bild d) ist dem OTIS-Verbindungssystem gewidmet [42]. Es basiert auf zwei Feldern von Linsen, so daß man mit Hilfe jeder Linse des ersten Feldes jede Linse des zweiten Feldes ausleuchten kann. Der obere Teil der Abbildung zeigt ein vollverbundenes Netzwerk für 16 Prozessoren. Die Sende- und Empfangseinheit eines jeden Prozessors besitzt ein 4x4 Feld aus Sendern bzw. Empfängern und eine eigene kleine Linse. Die Sender- und Empfängereinheiten sind in einem zweidimensionalen 4x4 Gitter angeordnet. Jeder Sender einer Sendereinheit dient der Datenübertragung an eine andere Empfangseinheit. Gleichzeitig ist in jeder Empfangseinheit jeder Sendereinheit ein Empfänger zugeordnet. Unten im Bild wird gezeigt, wie man mit Hilfe des OTIS-Konzeptes ein Shuffle implementieren kann, indem man zwei Linsenfelder mit unterschiedlich vielen Linsen (2 im ersten und 4 im zweiten) verwendet.

Diffraktive Komponenten

Diffraktive Bauelemente basieren auf der Beugung von Lichtwellen beim Auftreffen auf Oberflächen mit feinen Strukturen. Bei den Strukturen kann es sich sowohl um Graustufenmuster als auch um kleine Dickenunterschiede handeln. Die wichtigsten diffraktiven Bauelemente können wie folgt beschrieben werden:

Diffraktive Ablenkkomponenten: Diffraktive Komponenten können leicht die Funktionalität von Prismen und Linsen nachbilden. Für eine einfache Ablenkung eines Strahls ist eine Platte mit Graustufen oder Phasenmuster einer einzigen räumlichen Frequenz (siehe 4.1.2) notwendig. Für die Nachbildung der Linsenfunktionalität wird hingegen ein Muster aus konzentrischen Ringen verwendet, die nach außen hin immer

dünner werden und immer kleinere Abstände haben. Damit existiert außen ein Muster hoher Frequenz, während innen ein Muster geringer Frequenz vorhanden ist. Daraus folgt, daß das Licht außen stark und nach innen hin schwächer gebrochen wird. Dadurch wird die Fokusierfunktion einer Linse realisiert.

Wellenlängenmultiplexer und -demultiplexer: Der Gleichung (4.2) zufolge hängt der Winkel, unter dem ein refraktives Prisma eine ebene Welle ablenkt, von der Wellenlänge der Welle ab. Damit kann ein diffraktives Ablenkelement zur Trennung verschiedener Wellenlängen benutzt werden.

Beliebige Routingelemente: Wie in 4.1.2 erklärt, kann jede Lichtwelle in ebene Wellen zerlegt werden. Dies gilt auch für eine Strahlenkombination eines beliebigen Routingelementes. Da man andererseits durch ein entsprechend berechnetes Graustufenmuster eine beliebige Kombination ebener Wellen erzeugen kann, kann man auch diffraktiv beliebige Routingelemente realisieren.

Strahlteiler: Für die Realisierung von Broadcast-Verbindungen sowie die Ausleuchtung von großen Feldern von Lichtmodulatoren werden Strahlteiler benötigt. Solche Felder können nur schwer mit Linsen und Prismen hergestellt werden. Daher werden hier entsprechende diffraktive Bauteile benutzt.

Neben der höheren Flexibilität hat die diffraktive Optik gegenüber der refraktiven den Vorteil, daß sie sich besser für die Miniaturisierung und Massenherstellung eignet. Dies liegt daran, daß die Herstellung von Graustufenmuster z.B. mit Lithographie, einfacher ist als komplexe, dicke Oberflächenstrukturen.

Opto-Mechanisches System

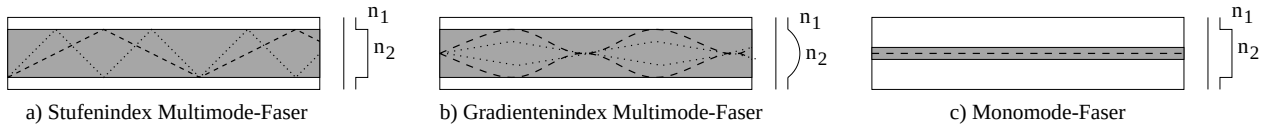
Die beiden wichtigsten Vorteile optischer Freiraumsysteme sind die große Kanaldichte und die Möglichkeit, eine hohe Verbindungskomplexität auf kleinem Raum zu erreichen. Um sie auszunutzen, müssen die Sender und Empfänger klein und nah beieinander sein. Dies bedeutet, daß das Abbildungssystem sehr genau sein muß. Die Toleranzen liegen dabei typischerweise zwischen 10 und 100 μm. Um dies zu erreichen, müssen alle Komponenten in Bezug aufeinander genau justiert werden. Da bei der Justierung in der Regel für jede Komponente alle 6 Freiheitsgrade berücksichtigt werden müssen, stellt dies bei komplexen Systemen eine schwierige Aufgabe dar. Hinzu kommt, daß für den Einsatz in der Praxis die Justierung in Bezug auf Umwelteinflüsse wie Erschütterungen und Temperaturschwankungen stabil sein muß. In der klassischen Optik werden solche Aufbauten auf speziellen, vibrationsisolierten optischen Tischen verwirklicht. Die optischen Komponenten werden in massive Metallfassungen untergebracht, die über Präzisionsversteller justiert werden. Für kompakte Aufbauten wurde in letzter Zeit die Möglichkeit entwickelt, die Komponenten mit Hilfe spezieller, stabiler Stangen oder Röhren zu verbinden. Sie sind allerdings nur für einfache Aufbauten geeignet und wie jede Art von Präzisionsmechanik sehr teuer. Außerdem sind sie immer noch nicht kompakt genug für den Einsatz in den meisten Stellen der Rechnerarchitektur. Damit stellt des mechanische System die größte Hürde für den Einsatz der Freiraumoptik in der Rechnerarchitektur dar.

4.2.4 Optische Wellenleiter

Für die Anwendung der Freiraumoptik ist es erforderlich, daß zwischen dem Sender und dem Empfänger eine ungestörte Sichtverbindung besteht. Hinzu kommen die bereits erwähnten Probleme bei der Justierung und beim robusten Aufbau. Eine Alternative, die diese Schwierigkeiten umgeht stellen Lichtwellenleiter (kurz LWL) Systeme dar. Bei solchen System wird das Lichtsignal durch unterschiedliche Brechungsindizies benachbarter Materialschichten 'eingesperrt' und gezwungen, dem Verlauf des Leiters zu folgen. Um LWL-System Netzwerken für die Rechnertechnik einsetzen zu können, sind drei Arten von Komponenten notwendig: 'Lichtkabel', Steckverbinder bzw. eine andere Möglichkeit, Kabel miteinander und mit Sendern und Empfängern zu verbinden, und Funktionseinheiten wie z.B. Verzweiger oder Wellenlängenfilter.

Glasfaser-LWL

Eine besondere Stellung under den LWL besitzen die Glasfaser (im nachfolgenden auch kurz Faser genannt). Sie stellen das optische Analogon zu elektrischen Kabeln dar. Das Funktionsprinzip von Glasfasern ist in Abbildung 4.10 dargestellt. Er besteht aus einem transparenten Kern, der von einem Mantel umgeben ist. Dabei ist das Material des Mantels so beschaffen, daß das aus dem Kern stammende Licht an der Grenzfläche zwischen Kern und Mantel reflektiert wird. Das Licht verbleibt also im Kern und breitet sich entlang der Faser durch fortwährende Reflexion aus. Dabei wird ein Strahl immer unter dem gleichen Winkel reflektiert, unter dem er in



a) Stufenindex Multimode-Faser b) Gradientenindex Multimode-Faser c) Monomode-Faser

Abbildung 4.10: Das Prinzip verschiedener Arten von Glasfaser-LWLs.

die Faser eingetreten ist. Da der Durchmesser von Fasern in der Größenordnung der Wellenlänge des Lichts liegt, darf die Ausbreitung allerdings nur näherungsweise als eine einfache Folge von Reflexionen betrachtet werden. In Wirklichkeit korrespondiert jeder Reflexionswinkel zu einer bestimmten Wellenfront, deren Ausdehnung durch die Grenze zwischen dem Kern und dem Mantel beschränkt ist. Dabei sind nur Wellen zugelassen, die bestimmten, diskret verteilten Winkeln entsprechen. Diese Wellen werden Moden genannt. Die Anzahl der in einer Faser möglichen Moden ist ein wichtiger Leistungsparameter. Bei Fasern mit einem Durchmesser, der in etwa gleich der Wellenlänge ist, ist nur eine Mode möglich. Solche Fasern werden Monomode-Fasern genannt. Mit steigendem Faserdurchmesser werden mehrere Moden möglich (Multimode-Faser). Bei einer Faserdicke von ca. 100λ sind es bereits mehrere Tausend.

Verbindungen

Um zwei elektrische Leitungen zu verbinden, muß man nur zwischen diesen beiden einen Kontakt herstellen. Bei Glasfasern sieht die Sache wesentlich komplizierter aus. Es müssen die Kerne der beiden Faser zusammengefügt werden, und zwar mit einer sehr hohen Genauigkeit. Trotzdem können Glasfaserverbindungen heute ähnlich einfach wie elektrische Verbindungen gehandhabt werden. Dafür sorgen spezielle Faserstecker, in den die beiden Faser durch Präzisionsmechanik fixiert. Die Stecker werden dann durch Führungsstifte zueinander justiert und festgeklemmt. In einer ähnlichen Weise können Faser an Laserdioden und Empfänger angebunden werden.

Neben der Steckverbindung gibt es auch noch verschiedene Möglichkeiten, Faser fest miteinander zu verbinden. Die einfachste ist das sog. 'Splicen'. Dabei werden die beiden Faser in einem speziellen kompakten Gerät zusammengeschweißt. Dies entspricht dem elektrischen Löten.

Alle obigen Techniken werden heute standardmäßig nur auf einzelne Fasern angewendet. Die mechanischen Justier- und Stabilitätsprobleme haben die Herstellung von kompakten mehradrigen Fasersteckern und Anschlüssen bisher verhindert. Dadurch war es auch nicht möglich, Faserverbindungen ähnlich einem elektrischen Flachbandkabel parallel aufzubauen.

LWL-Komponenten: Integrierte Optik

Für die Implementierung sinnvoller optischer Netzwerke werden neben einfachen Kabelverbindungen auch noch Verzweiger und Koppler benötigt. Außerdem braucht man für die Nutzung des Wellenlängenmultiplexings Bausteine zum Trennen und Vereinigen von Datenströmen verschiedener Wellenlänge in einer Faser. Solche Komponenten werden mit einer Technologie, die Integrierte Optik genannt wird, hergestellt. Dabei werden die Lichtleiter ähnlich den Leitungen auf eine Platine oder einem VLSI-Chip in einem Substrat fest integriert. Das Prinzip ist in Abbildung 4.11 dargestellt. Neben der Leitungen können in dem Substrat auch Funktionselemente wie Strahlteiler oder Beugungsgitter zur Wellenlängentrennung integriert werden.

Die wichtigsten heute verfügbaren Komponenten sind:

1xn Verzweiger: Hierbei handelt es sich um Broadcast-Elemente, die das Signal der ankommenden Leitung auf alle Ausgänge verteilen. Dabei sind heute bis zu 64 Ausgänge möglich.

nxm Sternverzweiger: Sternverzweiger verteilen das Signal der ankommenden Leitungen auf alle Ausgänge. Für n und m sind heute Werte von bis zu 64 möglich.

Wellenlängenmultiplexer: Ein Wellenlängermultiplexer verteilt mehrere in einer Faser verlaufende Wellenlängensignale auf unterschiedliche Ausgänge. Heutige Multiplexer können 16 bis 64 Wellenlängen trennen.

Wellenlängendemultiplexer: Ein Wellenlängendemultiplexer stellt die Umkehrung des Wellenlängenmultiplexers dar. Mehrere auf unterschiedlichen Leitungen ankommenden Signale unterschiedlicher Wellenlänge werden in einer einzigen Faser vereinigt.

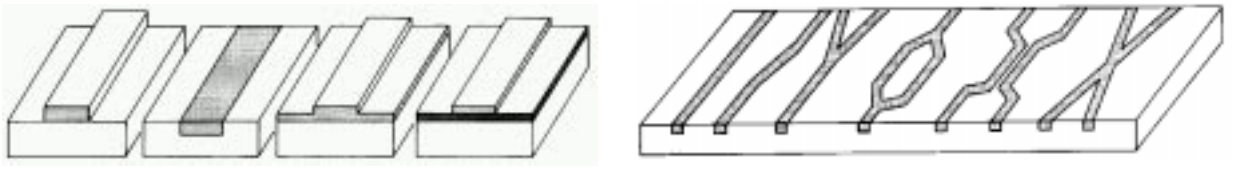


Abbildung 4.11: Das Prinzip der integrierten Optik: unterschiedliche Typen integrierter Lichtwellenleiter (rechts) und der Aufbau einiger Komponenten (links).

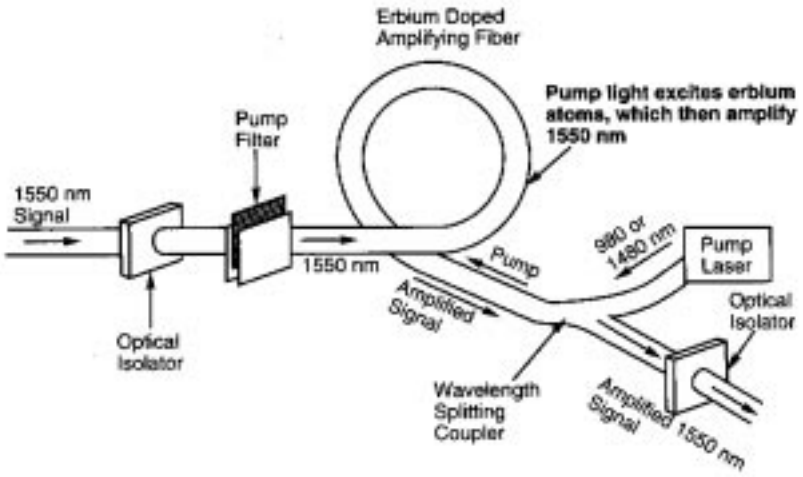


Abbildung 4.12: Der Aufbau eines Faserverstärkers (aus [67]).

Die obigen Komponenten werden als kompakte, wenige Zentimeter große Bauelemente hergestellt. Sie sind an den Ein- und Ausgängen mit Fasern und Fasersteckern versehen und somit einfach handhabbar.

LWL-Komponenten: Faserverstärker

Lichteleiterfasern zeichnen sich durch sehr geringe Leitungsverluste aus. Bei sehr langen Übertragungsstrecken oder bei der Realisierung eines hohen Fanouts kann es aber trotzdem zu einer Signalabschwächung kommen, die eine Verstärkung notwendig macht. Hierfür wurden spezielle Faserverstärker entwickelt (Abbildung 4.12). Es handelt sich dabei besondere Faserabschnitte die die Energie eines sog. Pumplasers aufnehmen und sie dem Signal hinzufügen können. Das Prinzip besteht darin, daß der Pumplaser die Ladungsträger in dem Faserabschnitt zunächst auf ein hohes Energieniveau anhebt. Sie verbleiben in diesem Zustand solange, bis ein Signal durch die Faser geschickt wird. In diesem Fall geben sie ihre Energie wie in einem Laser durch induzierte Emission ab. Dabei wird Licht mit genau der gleichen Frequenz emittiert, die die Emission verursacht hat, nämlich der Signalfrequenz.

Faserverstärker werden heute routinemäßig in der Telekommunikation eingesetzt. Sie können Signale mit einer Stärke von ca. 0.1mW auf bis zu 10mW verstärken ([31]). Die Verstärkung findet dabei unabhängig von der Frequenz und Datenrate statt und verursacht keine zusätzliche Latenz.

Leistungsparameter von Glasfasern

Die Bandbreite und Signalqualität eines Glasfaser werden durch drei Faktoren bestimmt: die Dämpfung, die Modendispersion und die spektrale Dispersion.

Dämpfung: Bei der Fortpflanzung in einer Faser wird das Signal durch Lichtabsorption gedämpft. Diese Dämpfung ist allerdings sehr gering. Sie liegt bei den heute in der Telekommunikation üblichen Glasfasern in der Größenordnung von 0.1 bis 1dB/Km, wobei der genaue Wert von der Art der Faser und der Wellenlänge abhängt.

Modendispersion: Aufgrund der unterschiedlichen Reflexionswinkel müssen die Moden bei einer gegebenen Faserlänge unterschiedliche Weglängen durchlaufen. Dementsprechend unterschiedlich sind auch ihre Laufzeiten. Ein Signal, das mehrere Moden beinhaltet, wird so mit steigender Entfernung immer unschärfer.

Dieser Effekt wird Modendispersion genannt. Eine Datenübertragung ist in einer Faser nur dann möglich, wenn die Laufzeitunterschiede zwischen den Moden eines Signals unterhalb des zeitlichen Abstands zwischen nachfolgenden Bits bleiben. Ansonsten würden sich benachbarte Bits überlagern und so Übertragungsfehler verursachen. Da ein Signal in einer Multimode-Faser immer aus mehreren Moden besteht, wird die Bandbreite von Multimode-Fasern durch die Modendispersion beschränkt.

Wellenlängendispersion: Da die Lichtgeschwindigkeit in einem Medium von der Wellenlänge abhängt, gibt es auch in Bezug auf die Wellenlänge eine Dispersion. Signale mit einer nicht verschwindenden spektralen Breite verlieren beim Durchlaufen langer Fasern an Schärfe und können somit nur eine beschränkte Datenrate haben. Mit steigender Datenrate nimmt die spektrale Breite eines Signals aufgrund der Unschärferelation zu. Die Wellenlängendispersion stellt somit eine fundamentale Beschränkung der Zeitbandbreite von Signalen in Fasern dar.

Für Anwendungen in der Rechnerarchitetur sind in der Praxis lediglich die durch die Modendispersion bedingten Schranken von Bedeutung. In guten Glasfasern machen machen sich die Dämpfung und die spektrale Dispersion erst bei Entfernungen im Kilometerbereich bemerkbar. Auf der für die Rechnerarchitektur benötigten Strecke von bis zu 10m können in einer Monomode-Faser problemlos Datenraten von mehreren hundert TBit/s übertragen werden. In Multimode-Fasern ist die Bandbreite auf etwa 1Tbit/s/m, also etwa 100Gbit/s bei 10m beschränkt. Berücksichtigt man zusätzlich die Möglichkeit des Wellenlängenmultiplexings, so sind in Monomode-Fasern Datenraten von mehreren Tausend TBit/s möglich. In Multimode-Fasern könnten bis etwa 100TBit/s/m erreicht werden.

Trotz ihrer geringeren Bandbreite sind Multimode-Fasern für die Anwendung in der Rechnerarchitektur von großer Bedeutung. Dies liegt daran, daß sie einen deutlich größeren Durchmesser als die Monomode-Faser haben($50\mu m$ bis $250\mu m$ gegenüber ca. $1\mu m$), und somit wesentlich leichter zu handhaben sind. Dies macht sich vor allem beim Ein- und Auskoppeln des Lichts aus der Faser bemerkbar. Um die Verluste gering zu halten, muß beim Einkoppeln das Signal so genau wie möglich fokussiert werden. Bei Monomode-Fasern erfordert dies eine Positioniergenauigkeit im Bereich von ca $0.2\mu m$. Bei Multimode-Fasern reicht dagegen eine Genauigkeit zwischen $10\mu m$ und $50\mu m$.

4.2.5 Fazit

Zur Zeit werden in der Datenübertragung lediglich einfache, faserbasierte Systeme eingesetzt. Die Verwendung von Freiraumsystemen scheitert an der Größe und mangelnder Robustheit mechanischer Aufbaukomponenten.

Bei den heutigen Fasersystemen handelt es sich um sequentielle Übertragungssysteme mit einer Bandbreite von bis zu 2,4Gbit/s. Für die Kommunikation auf kurze Distanzen (z.B. bei LAN-Netzwerken) werden Multimode-Faser, sonst Singlemodefaser benutzt. Es sind zur Zeit alle die in 4.2.4 beschriebenen Funktionskomponenten, also Verzweiger, Sternverzweiger und Wellenlängenmultiplexer bzw. demultiplexer als kommerzielle Produkte verfügbar. Gleiches gilt für Sender und Empfänger. Sie werden als integrierte Module angeboten, die wie gewöhnliche elektronische Komponenten auf eine Platine aufgelötet werden können. Für die Verbindung der Systemkomponenten werden Stecker verwendet, die wie normale elektrische Verbindungen gehandhabt werden.

Das größte Manko der heutigen Fasertechnik ist das Fehlen integrierter paralleler Faserverbindungen in der Art elektronischer Flachbandkabel. Will man mehrere Leitungen verwenden, dann sind auch mehrere diskrete Faser, jeder mit einem eigenen Sender und Empfänger notwendig. Diese belegen verhältnismäßig viel Platz auf der Platine, so daß parallele Faserverbindung nicht praktikabel sind. Damit kann das Bandbreite-Potential der Optik nur sehr mangelhaft genutzt werden.

4.3 Elektronische und optische Datenübertragung in der Praxis

Im vorliegenden Abschnitt wird gezeigt, an welchen Stellen sich die Schwächen der elektronischen Datenübertragung in der Praxis bemerkbar machen und wie die prinzipiellen Vorteile der Optik an diesen Stellen Abhilfe schaffen können. Gleichzeitig wird erläutert, warum in den meisten Fällen die Optik in der Praxis noch nicht zum Einsatz kommt.

Beim Vergleich der Leistung der optischen und elektronischen Datenübertragung in der Praxis ist es es sinnvoll, zwischen 5 Klassen von Verbindungen zu unterscheiden:

1. **On-Chip Verbindungen:** Hiermit sind Verbindungen zwischen den Komponenten eines VLSI Chips gemeint.

2. **Lokale Chip-to-Chip Verbindungen:** In diese Klasse fallen Verbindungen zwischen benachbarten Chips, also z.B. zwischen dem Prozessor und dem second level Cache.

3. **Globale Chip-to-Chip Verbindungen:** Hierbei handelt es sich um alle Verbindungen auf einer Platine, die nicht zu den lokalen Chip-to-Chip Verbindungen zählen. Dazu gehört auch die Prozessor-Speicher Koppelung in sequentiellen Rechnern und in kleinen symmetrischen Prozessoren.

4. **Board-to-Board Verbindungen:** Hier sind Verbindungen zwischen den verschiedenen Platinen gemeint. Sie dienen zum einen der Verbindung der Hauptplatine mit Peripherieeinheiten. Zum anderen werden sie bei Parallelrechnern zur Implementierung des Kommunikationsnetzwerks benötigt. In diese Klasse fällt auch die Prozessor-Speicher Koppelung bei größeren symmetrischen Multiprozessoren.

5. **Rack-to-Rack Verbindungen:** In diese Klasse fallen Verbindungen zwischen verschiedenen Rechnereinheiten. Hierzu gehören vor allem Verbindungen zwischen Arbeitsstationen sowie LAN- und WAN-Netze. Ebenfalls dieser Klasse zuzurechnen sind Verbindungen zwischen einzelnen Einheiten großer Supperrechner. Will man das Modell der symmetrischen Multiprozessoren auf Arbeitsplatzrechner-Cluster und Supperrechner erweitern, so muß die Prozessor-Speicher Koppelung als Rack-To-Rack Verbindung implementiert werden.

4.3.1 On-Chip Verbindungen

Die elektronische Datenübertragung hat ihre Stärken bei sehr kurzen Distanzen, wie sie auf einem VLSI-Chip vorkommen. Bei modernen VLSI-Bausteinen können Punkt-zu-Punkt Verbindungen bei einer Leitungsdicke von 1 bis 2 μm Datenraten im GHz-Bereich erreichen. Die Implementierung von Broadcastverbindungen und Bussen führt zwar auch bei On-Chip Verbindungen, parasitären Kapazitäten und Crosstalk zu Problemen, diese können aber mit leistungsfähigen Treiberbausteinen weitgehend zufriedenstellend gelöst werden. So können prozessorinterne Busse bei einer Breite von 64 bis 256-Bit Datenraten im Bereich der Prozessorfrequenz (200 bis 500 MHz) arbeiten.

Bei On-Chip Verbindungen liegt die größte Schwäche der Elektronik bei komplexen Verbindungsstrukturen. Dadurch, daß in VLSI nur eine geringe (3-6) Anzahl von Schichten für die Datenleitungen zur Verfügung stehen, nehmen Netze mit komplexen, dichten Topologien eine große Fläche ein. Wie in Abschnitt 4.1.2 erläutert, wird für Verbindungen mit einer gegeben Bisektionsbreite B eine Fläche von $O(B^2)$ benötigt. Dies ist besonders bei der Implementierung von Multiport-Speichereinheiten hinderlich, bei denen die Bisektionsbreite linear mit der Anzahl der Ports ansteigt. Hier könnte durch die Anwendung von optischen Freiraumverbindungen Abhilfe geschaffen werden, bei denen die Grundfläche nur linear mit der Bisektionsbreite ansteigt. Allerdings wird dazu die Möglichkeit benötigt, komplexe optische Systeme in einem mikroskopischen, robusten Aufbau zu implementieren. Außerdem machen optische Verbindungen in diesem Zusammenhang nur dann Sinn, wenn die optischen Ein/Ausgabekanäle direkt an die Elektronik der VLSI-Bausteine angekoppelt werden. Dies war bisher nicht möglich.

4.3.2 Lokale Chip-to-Chip Verbindungen

Lokale Chip-to-Chip Verbindungen sind wenige Zentimeter lange Punkt-zu-Punkt Leitungen zwischen VLSI-Bausteinen. Sie werden entweder als sog. Multichipmodule [161] mit Hilfe spezieller Keramiksubstrate oder als kurze Leitungen auf gewöhnlichen Platinen implementiert. Im ersten Fall wird das Substrat durch Bonden direkt mit dem VLSI-Schaltkreis befestigt. Im zweiten Fall werden an dem VLSI-Schaltkreis zunächst sog. Bonding-Drähte befestigt. Sie verbinden den Schaltkreis mit dem Chip-Gehäuse, das wiederum über die Pins mit der Platine verbunden ist.

Lokale Chip-to-Chip Verbindungen sind elektronisch schwieriger zu implementieren und weniger leistungsfähig als On-Chip Verbindungen. Dies liegt vor allem daran, daß die Leitungen länger sind, so daß sich die für die elektronische Datenübertragung typischen Störeffekte bemerkbar machen. Ein weiterer Störfaktor ist der Übergang zwischen verschiedenen Übertragungsmedien (VLSI-Leitung, Bonding-Draht, Pin, Platine). Trotzdem ist es mit Hilfe differentialer Signale und aufwendiger Treiberbausteine möglich, Datenraten von weit über 1Gbit/s pro Leitung zu erreichen. Die Leitungsbreite liegt dabei sowohl bei Platinen als auch bei MCM-Substraten um ein

bis zwei Größenordnungen über der VLSI-Leitungsdicke. Allerdings wird dies weitgehend durch die Größe der zur Verfügung stehenden Fläche und eine größere Anzahl von Leitungslagen (bei Platinen bis zu 24) weitgehend wettgemacht. Somit stellt die auf Platinen und MCM-Substraten im lokalen Chip-to-Chip Bereich verfügbare Bandbreite keine Beschränkung für die Leistung von Rechnern dar.

Vom Standpunkt der Rechnerarchitektur stellt zur Zeit der Flaschenhals beim Übergang vom Chip auf die Platine bzw. das MCM-Substrat (*Pin-Beschränkung*) das größte Problem der der lokalen Chip-to-Chip Verbindungen dar.

Pin-Beschränkung

Moderne Mikroprozessoren haben auf einem Chip um 10^7 Transistoren. Dies entspricht in etwa $2 \cdot 10^6$ Millionen einfacher Logikgatter (z.B. AND-Gatter). Bei einer durchschnittlichen Gatterschaltzeit von ca. 500ps folgt daraus eine theoretische Anzahl von $4 \cdot 10^{16}$ logischer Operationen pro Sekunde. Demgegenüber steht bei den meisten heutigen Prozessoren eine IO Leistung von ca. 100 Gbit/s ($= 10^{11} Bit/s$). Die Pin-Beschränkung ist die Ursache für das in Abschnitt 3.5.2 erwähnte Schnittstellenproblem der Snooping-Basierten Cache-Kohärenz Protokolle. Auf einem Chip mit 2 Millionen Logikgattern ließen sich ca. 20000 64Bit Vergleicher realisieren. Unter der Annahme, daß jeder Vergleicher mit der Prozessorfrequenz (500MHz) betrieben wird, könnte ein Chip so einen Datenstrom von 10TBit/s überwachen. Auf Grund der Pin-Beschränkung ist es aber nicht möglich, die benötigten Daten in den Chip zu laden.

Für die Pin-Beschränkung gibt es zwei Gründe: Zum einen machen die schlechten Leitungseigenschaften der externen Verbindungen aufwendige Treiber notwendig, die viel Platz auf dem Chip benötigen. Dies trifft insbesondere dann zu, wenn die Verbindungen mit mehr als 50 bis 100MHz betrieben werden sollen. Zum anderen ist es schwierig, mehr als einige Hundert bis 1000 externe Anschlüsse an einem nur wenige Quadratzentimeter großen VLSI-Chip zu befestigen. Dies ist unabhängig davon, ob man die Anschlüsse wie bei einfachen Chips üblich nur am Rand anbringt oder über die ganze Chipfläche verteilt. Bei Multichipmodulen hat man das Problem, daß die Leitungen planar in einer festen Anzahl von Leitungslagen verlaufen. Damit ist auch die Anzahl von Leitungen beschränkt, die von der Seite kommend die über die Chipfläche verteilten Anschlüsse kontaktieren können. Dies liegt daran, daß die am Rande liegenden Anschlüsse den Weg zu den inneren Anschlüssen versperren. Im Falle konventioneller Platinen stellt die Größe der Bonding-Drähte und die Präzision, mit der sie befestigt werden können, die entscheidende Schwierigkeit dar. So benötigt ein Draht einen sog. Pad mit einem Durchmesser von ca. 50 bis 100 μm. Hinzu kommt, daß die Anzahl und Dichte der Drähte durch Probleme mit dem elektrischen Übersprechen beschränkt ist.

Einsatzmöglichkeiten der Optik

Das Problem der Pin-Beschränkung könnte man theoretisch durch direkte optische Verbindungen zwischen VLSI-Chips lösen. Dazu wären vier Dinge notwendig:

1. Große Felder (1000 bis 10000 Elemente) von opto-elektronischen Sendern und Empfängern,
2. die Möglichkeit, solche Felder auf VLSI-Schaltkreisen zu integrieren,
3. einfache Treiber für die optischen Sender und Empfänger, deren Platzbedarf auf dem VLSI-Chip mit dem Platzbedarf einfacher Gatter vergleichbar ist, und
4. die Möglichkeit, optische Systeme, die für die Übertragung der vielen Datenkanäle zwischen den Chips benötigt werden, in einer kompakten, stabilen Form zu implementieren.

Diese Voraussetzungen waren bisher nicht oder nicht ausreichend erfüllt. Daher spielt die Optik zur Zeit bei lokalen Chip-zu-Chip keine Rolle.

4.3.3 Globale Chip-to-Chip Verbindungen

Globale Chip-to-Chip Verbindungen stellen die Verbindung zwischen weit entfernten Komponenten auf einer Platine her. Sie haben eine Länge von bis zu 50cm und werden sowohl als Punkt-zu-Punkt, als auch als Broadcast-Leitungen verwendet.

Durch die große Leitungslänge machen sich bei globalen Chip-to-Chip Verbindungen die in 4.1.1 beschriebenen Störeffekte stark bemerkbar. Es muß also mit Übersprechen, Signalverzerrung durch Wellenabsorbtion, Wellenreflexionen und Spannugsschwankungen auf den Versorgungsleitungen gerechnet werden. Bei Punkt-zu-Punkt Leitungen geringer Dichte lassen sich trotzdem durch aufwendiges Platinen-Layout und spezielle Treiber, Datenraten bis zu 1GHz pro Leitung erreichen [186]. Bei Broadcast-Verbindungen sowie dichten parallelen Leitungsbündeln sind die Probleme dagegen so groß, daß die Bandbreite auf 100 bis 200 MHz reduziert wird. Hinzu kommt eine hohe Latenz, die vor allem durch die Wellenreflexionen verursacht wird. So muß am Anfang eines Zugriffs abgewartet werden, bis die Reflexionen abgeklungen sind, die Leitung also 'zur Ruhe' gekommen ist. Bei den heutigen Bussen liegt aus diesem Grund die Latenz zwischen 50 und 500ns.

Vom Standpunkt der Rechnerarchitektur stellt zur Zeit die Leistungsfähigkeit der globalen Chip-to-Chip Verbindungen das größte Problem dar. Sie ist maßgeblich verantwortlich für die Beschränkungen der Leistungsfähigkeit und der Skalierbarkeit symmetrischer Multiprozessoren.

Optische Verbindungen sind im Prinzip hervorragend geeignet, um die Probleme der Chip-to-Chip Verbindungen zu lösen. Wie in 4.1.2 dargelegt, sind die Signalqualität und die Latenz weitgehend unabhängig von der Datenrate, der Leitungslänge und dem Fanout. Außerdem stellt auch bei hoher Kanaldichte das Übersprechen wie Anfangs des Kapitels beschrieben ein wesentlich geringeres Problem als bei der Elektronik dar. Insbesondere hängt das Maß des Übersprechens nicht von der Datenrate ab. Trotzdem findet die Optik heute mit Ausnahme einiger experimenteller Systeme in diesem Bereich noch keine Anwendung. Dies liegt an zwei Dingen: zum einen gehört die Technologie zum kompakten, robusten und billigen Aufbau solcher Systeme heute noch in den Bereich der Forschung. Zum anderen wären zum Erreichen der benötigten Bandbreite auch optisch mehrere Leitungen notwendig. Die Bandbreite kommerzieller optischer Sender liegt heute bei bei maximal ca. 2,4Gbit/s. Damit wären 4 bis 8 Leitungen notwendig, um die Bandbreite eines Hochleistungsbusses zu erreichen. Jeder Busknoten müßte also 8 bis 16 optische Sender und Empfänger besitzen. Dies wäre nur mit integrierten parallelen optischen Sendern und Empfängern sinnvoll. Mit heutigen Einzelsendern und -empfängern wäre eine solche Anordnung aus Platzgründen nicht praktikabel. Allein die Sender und Empfänger einiger weniger Busknoten würden Platz einer gesamten Platine benötigen.

4.3.4 Board-to-Board Verbindungen

Board-to-Board Verbindungen haben eine Länge von ca. 1m bis 2m. Sie werden in der Regel durch Flachbandkabel implementiert, die über spezielle Treiberbausteine angesteuert werden. Manchmal werden sie auch in Form sog. Backplanes realisiert. Dabei handelt es sich um spezielle Verbindungsplatinen, auf denen sich massiv parallele Hochleistungsverbindungen befinden. Solche Backplanes werden bevorzugt in Parallelrechnern eingesetzt, die aus mehreren Prozessorplatinen bestehen.

Elektrische Board-to-Board Verbindungen haben mit den gleichen Problemen wie elektrische globale Chip-to-Chip Verbindungen zu kämpfen. Aufgrund der größeren Leitungslänge treten die Schwierigkeiten allerdings deutlich stärker auf. Dies führt zu einer noch höheren Latenz und geringeren maximalen Betriebsfrequenz. Besonderes betroffen sind davon Broadcast-Kanäle, wie z.B. Hochleistungsbusse. Die mangelnde Leistungsfähigkeit, insbesondere die hohe Latenz von Board-to-Board Verbindungen ist eines der Hauptprobleme bei der Implementierung effizienter, großer Parallelrechner. Sie ist auch für die Beschränkung der Skalierbarkeit von symmetrischen Multiprozessoren entscheidend.

Die Anwendung der Optik bei den Board-to-Board Verbindungen scheitert an den gleichen Problemen wie bei den globalen Chip-to-Chip Verbindungen: der zu geringen Bandbreite einzelner Kanäle und dem zu hohen Aufwand für parallele Verbindungen.

4.3.5 Rack-to-Rack Verbindungen

Rack-to-Rack Verbindungen haben eine Länge zwischen einigen wenigen und ca. 100m. Bei solchen Entfernungen sind elektrisch selbst Datenraten von einigen wenigen Hundert MBit/s pro Leitung nur sehr schwer zu realisieren. Sie werden in der Regel mit seriellen Koaxialleitungen verwirklicht. Bei solchen Leitungen sind die einzelnen Treiber und Anschlüsse ähnlich aufwendig wie optische Faseranschlüsse. Damit ist eine Steigerung der Bandbreite durch viele parallele Leitungen nicht praktikabel. Auch die Latenz ist aufgrund serieller Übertragung ähnlich hoch wie bei konventionellen optischen Systemen.

Aus obigen Gründen werden heute bei Rack-to-Rack Verbindungen routinemäßig optische Fasersysteme eingesetzt. Die 2,4Gbit/s, die heutige kommerzielle Systeme bieten, liegen deutlich über der Bandbreite elektrischer

Systeme. Hinzu kommt, daß diese Bandbreite unabhängig von der Entfernung und bei guter Signalqualität erreicht werden kann.

Vom Standpunkt der Rechnerarchitektur aus, besteht das Problem der Rack-to-Rack Verbindungen darin, daß die Bandbreite sowohl der elektrischen als auch der optischen Verbindungen um einen Faktor von mindestens 10 kleiner ist als die Bandbreite auf einer Platine oder in einer Backplane. Aus diesem Grund ist es zur Zeit nicht möglich, Arbeitsplatzrechner mit Hilfe des SMP-Speichermodells zu koppeln. Abhilfe könnten hier kompakte parallele optische Übertragungsysteme schaffen. Solche Systeme stehen zur Zeit kurz vor der Markteinführung, oder sind gerade eben verfügbar geworden. Sie hatten daher noch keine Auswirkung auf die Rechnerarchitektur.

4.4 Zwischenbilanz

Die bisherigen Ausführungen dieses Kapitels können wie folgt zusammengefaßt werden:

Die elektronische Datenübertragung hat ihre Stärken bei einfachen Verbindungsstrukturen mit geringem Fanout und bei sehr kurzen Distanzen, wie sie auf einem VLSI-Chip vorkommen. Ihre Leistungsfähigkeit nimmt mit steigender Komplexität, Länge und Fanout rapide ab, wobei die Implementierung mit immer mehr Problemen verbunden ist. Dieser Leistungsabfall ist durch die fundamentalen Eigenschaften der elektrischen Datenübertragung bedingt. Er ist maßgeblich für die Beschränkung der SMP-Architektur auf Rechner mit wenigen Prozessoren und kleiner räumlicher Ausdehnung (also keine Arbeitsplatzrechner-Cluster) verantwortlich.

Bei optischen Verbindungen sind die Übertragungseigenschaften weitgehend unabhängig von der Datenrate, der Leitungslänge und dem Fanout. Dadurch sind sie prinzipiell geeignet, die obigen Probleme zu lösen. Allerdings stand die dafür notwendige Technologie bisher nicht zur Verfügung. Es fehlten vor allem:

1. die Möglichkeit, optische Ein/Ausgabekanäle direkt auf VLSI-Schaltkreisen zu integrieren,
2. kompakte robuste optische Systeme für die Realisierung komplexer Verbindungsstrukturen,
3. kompakte parallele optische Verbindungen mit einer geringen Latenz.

In den nachfolgenden Abschnitten wird gezeigt, daß neue Entwicklungen im Bereich der Opto-Elektronik diese Mängel beseitigen können.

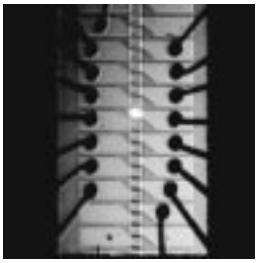
4.5 Neue Entwicklungen: Optische Sender und Empfänger

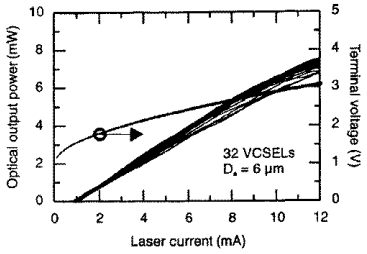
Im Bereich der Sender und Empfänger sind vor allem drei Entwicklungen für die Rechnerarchitektur von Bedeutung:

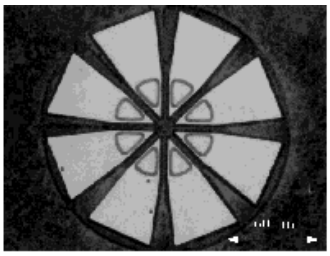
1. Es wurden billige, leistungsfähige Laserdioden mit geringem Energieverbrauch und hoher Integrationsdichte entwickelt. Solche Laserdioden sind eine der Voraussetzungen für parallele optische Verbindungen.
2. Es wurden Möglichkeiten gefunden, Sender zu bauen, die auf mehreren verschiedenen Wellenlängen senden können. Dies eröffnet die Möglichkeit paralleler optischer Verbindungen auf der Basis des WDM Verfahrens.
3. Es wurden neue Modulatoren entwickelt, die eine hohe Bandbreite bei geringem Schaltstrom besitzen und ebenfalls eine hohe Integrationsdichte zulassen. Solche Modulatoren vereinfachen die Realisierung von Verbindungen, die gleichzeitig eine hohe Bandbreite und einen hohen Fanout haben.
4. Es sind Verfahren entwickelt worden, um große zweidimensionale Felder von Laserdioden, Modulatoren und Empfängern direkt auf konventionellen VLSI-Bausteinen zu befestigen. Dadurch wird eine wichtige Voraussetzung für die Aufhebung der Pin-Beschränkung und die Implementierung direkter optischer Verbindungen zwischen VLSI-Bausteinen geschaffen.

4.5.1 Laserdioden

Betrachtet man die Technologie der Halbleiterlaser unter dem Aspekt paralleler, mit VLSI integrierter optischer Verbindungen, so sind vor allem zwei Dinge wichtig:







a) b) c)

Abbildung 4.13: Die Abbildung zeigt ein Feld von VCSEL Laserdioden (a), die optische Leistung eines 4x8 Feldes von VCSELs (b, aus [20]), sowie ein Feld von 8 Laserdioden unterschiedlicher Wellenlänge, die in eine Faser eingekoppelt werden (c, aus [90])

1. Es wird die Möglichkeit benötigt, Laserdioden billig in großen, vorzugsweise zweidimensionalen Feldern mit einer Dichte um 100 Komponenten /mm^2 herzustellen. Dies ist notwendig, um auf einem maximal 2x2cm großen VLSI-Chip Tausende von optischen Kanäle unterbringen zu können.

2. Der Energieverbrauch der einzelnen Laserdioden muß auch bei hohen Datenraten so gering wie möglich ($\leq 1mW$) pro Laser sein. Anderenfalls wäre der Betrieb großer Felder solcher Komponenten auf Grund von Wärmedisspiations- und Energieversogungsproblemen nicht möglich. Dies ist vor allem bei der Integration mit VLSI von großer Bedeutung, da heutige VLSI Bausteine selbst eine Wärmedissipation von bis zu 50W besitzen.

Beide Forderungen werden durch Komponenten erfüllt, die in letzter Zeit in verschiedenen Forschungsprojekten entwickelt wurden. Solche Komponenten sind zum Teil sogar kommerziell bzw. als Laborprototypen erhältlich.

Integration

Laserdioden sind Halbleiterbauelemente, die in ähnlicher Weise wie konventionelle VLSI Komponenten hergestellt werden. In der Theorie spricht also nichts gegen eine billige Herstellung großer, integrierter Felder. In der Praxis sind sie jedoch lange Zeit an technischen Problemen wie Ausbeute, Uniformität der Lasereigenschaften und Lebensdauer der Dioden gescheitert. Hinzu kam, daß Laserdioden lange Zeit nur als sog. kantenemittierende Bauelemente hergestellt werden konnten. Bei solchen Bauelementen tritt das Licht parallel zur Waferoberfläche seitlich aus. Dies macht es unmöglich, ein zweidimensionales Feld von LDs auf einem Chip herzustellen.

Motiviert vor allem durch die Anforderungen aus der Telekommunikation wurden inzwischen die meisten der technischen Probleme weitgehend gelöst. Die Beschränkung auf eindimensionale Felder wurde mit der Entwicklung von oberflächenemmitierenden LDs (sog. VCSEL für **V**ertical **C**avity **S**urface **E**mitting **L**aser, siehe z.B. [157]) aufgehoben. VCSELs emittieren das Licht vertikal zur Waferoberfläche. Sie wurden erfolgreich mit einer Dichte von bis zu 1000 Bauteilen pro mm^2 ([78]) in Feldern mit Tausenden von Elementen hergestellt. Kleinere Felder (8x8 bzw. 16x16) wurden bereits erfolgreich in verschiedenen Prototypen und Laboraufbauten getestet [82, 136]. Tests mit Feldern von 32x32 Elementen sind für die nächste Zeit geplant [21]. Die Entwicklung großer, in der Praxis einsetzbarer Felder mit mehreren Tausend Elementen wird zur Zeit in mehreren Forschungsprojekten (z.B. das MEL-ARI Projekt der EU [1]) und in der Industrie (z.B. bei Honywell) vorangetrieben.

Kleinere Laserdiodenfelder sind zur Zeit auch zum Teil kommerziell bzw. in Form von Laborprototypen erhältlich. Ein Beispiel zeigt Abbildung 4.13. Es ist ein Laserdiodenfeld mit 16 Elementen, das von unserem Institut 1995 zu Forschungszwecken erworben wurde. Es ist ein gutes Beispiel für den Preisverfall, der in der Opto-Elektronik in letzter Zeit stattgefunden hat. So hat vor 4 Jahren der gezeigte Laserchip ca. 20.000 DM gekostet. Heute kann man solche Bausteine für wenige Hundert DM erwerben.

Leistungsparameter

Ein wichtiger Forschungsschwerpunkt im Bereich von Laserdioden-Felder ist die Verbesserung ihrer Leistungsparameter. Dazu gehören vor allem die Effizienz und die maximale Ausgangsleistung. So sind in zahlreichen Arbeiten VCSEL-Laserdiodenfelder mit einer Effizienz von über 50 % demonstriert worden. Dabei konnte ei-

ne Ausgangsleistung von 2 bis 4 mW pro Diode erreicht werden. Gibt man sich mit einer geringeren Effizienz zufrieden so kann sogar eine Leistung zwischen 6 und 8mW erreicht werden (siehe Abbildung 4.13).

4.5.2 Wellenlängenselektion

Für die Nutzung des Wellenlängenmultiplexings ist es notwendig, die Wellenlänge des emittierten Lichts sehr genau zu kontrollieren. Insbesondere werden Lichtquellen benötigt, die je nach Bedarf auf verschiedenen Wellenlängen senden können. So kann ein Sender auf unterschiedlichen Kanälen Daten übertragen. Durch gleichzeitiges Senden auf mehreren Kanälen ist außerdem entweder ein Rundruf oder ein Multicast möglich.

Für die Steuerung der Wellenlänge gibt es heute zwei Möglichkeiten. Zum einen gibt es LDs, bei denen der Abstand zwischen den beiden halbdurchlässigen Spiegeln durch elektrische Impulse um Bruchteile einer Wellenlänge verstellt werden kann. Dadurch kann die Wellenlänge des Lichtes gewählt werden, die durch die induzierte Emission verstärkt wird. Dabei wird die Tatsache ausgenutzt, daß der Abstand zwischen den Spiegeln ein ganzzahliges Vielfaches der Wellenlänge sein muß, damit das Licht zwischen den Spiegeln hin und her reflektiert wird. Sender, die nach diesem Prinzip arbeiten, wurden u.a. in [71] beschrieben. Zum anderen können Felder von LDs hergestellt werden, deren Elemente auf unterschiedlichen Wellenlängen senden können. Solche Bausteine wurden mit bis zu 10 unterschiedlichen Wellenlängen und einer Bandbreite von bis zu 2,4Gbit/s pro Wellenlänge demonstriert [183, 90]. Sie wurden in einem groß angelegten Testnetzwerk in den USA verwendet [166], das von dem AON-Konsortium mehrerer Universitäten und Forschungslabors implementiert wurde.

4.5.3 Modulatoren

Die Entwicklung schneller kompakter Modulatoren ist seit geraumer Zeit wichtiges Forschungsthema. Dabei wurden für die Faseroptik integrierte optische Modulatoren entwickelt, die in Fasersystemen Datenraten von bis zu 100Gbit/s erlauben. Darüber hinaus sind mit den sog. MQW-Modulatoren auch für den Einsatz in der Freiraumoptik schnelle Modulatoren implementiert worden. Die Leistung und Funktionsweise beider Modulatorklassen wird im Nachfolgenden zusammengefaßt.

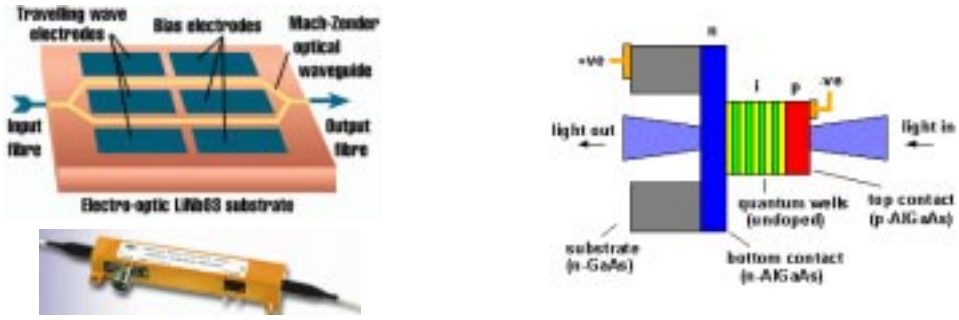
Integrierte Elektro-Optische Modulatoren

Für Fasersysteme wurden integrierte optische Modulatoren entwickelt, die Datenraten zwischen 10 und 100GHz [93]erreichen können. Sie funktionieren in der Regel nach dem sog. Mach-Zehnder Interferometer Prinzip. Dabei wird das ankommende Signal in zwei Äste aufgeteilt, und danach wieder vereinigt. In einem dieser Äste kann durch das Anlegen einer Spannung die Ausbreitungsgeschwindigkeit des Signals verzögert werden. Falls die Verzögerung genau eine halbe Wellenlänge beträgt, dann findet beider Vereinigung der Strahlen eine destruktive Interferenz statt, so daß kein Signal übertragen wird.

Integrierte optische Modulatoren mit eine Datenrate zwischen 10 und 40Gbit/s sind zur Zeit kommerziell erhältlich (z.B. [91]).

MQW-Modulator-Felder

Mach-Zehnder Modulatoren sind für zweidimensionale Felder, wie sie in der Freiraumoptik benötigt werden, nicht geeignet. Dies liegt vor allem daran, daß für eine hohe Schaltgeschwindigkeit die beiden Äste des Modulators eine Länge im Zentimeter-Bereich haben müssen. Eine Alternative stellen hier die sog. Multiple Quanten Well-Modulatoren [127, 125] dar. Sie basieren auf der Veränderung des Absorptionsspektrums sog. Excitonen in übereinander gestapelten, wenige Mikrometer dicken Schichten unterschiedlicher Halbleiter (Quantum Wells genannt). Excitonen sind Pseudoatome, die sich in Halbleiterkristallen bilden, wenn ein freies Elektron von einer nicht ganz neutralisierten positiven Ladung eines Gitteratoms angezogen wird. Unter normalen Umständen ist die Bindungsenergie solcher Pseudoatome so gering, daß sie bereits von extrem niederenergetischer Strahlung zerstört werden. Innerhalb eines Quantum Wells werden die Excitonen jedoch eingesperrt. Dies führt dazu, daß die Bindungsenergie höher wird, und ganz bestimmte Wellenlängen zwischen 850 und 1500 μm absorbiert werden. Welche Wellenlänge genau absorbiert wird, kann man dabei durch das Anlegen einer Spannung verändern, da sich durch eine angelegte Spannung auch die Bindungsenergie verändert. Dadurch kann man einen sehr schnellen Schalter für bestimmte Wellenlängen bauen. Der Aufbau eines solchen Schalters ist in Abbildung 4.14 zu sehen.



a) b)

Abbildung 4.14: Ein Mach-Zehnder Modulator und sein Aufbauprinzip (a aus [91]) und der Aufbau eines MQW-Modulators für Freiraumsysteme (b).

MQW-Modulatoren wurden mit einer Schaltzeit von 33ps und weniger demonstriert [52]. Dabei wurden zweidimensionale Felder von bis zu 64000 Elementen hergestellt. Sie sind somit sehr gut für massiv parallele Verbindungen mit sehr hoher Bandbreite geeignet. Solche Verbindungen wurden auch in verschiedenen Experimenten demonstriert [126, 116, 117].

4.5.4 Smart Pixels

Wie in Abschnitt 4.3 verdeutlicht, werden für eine effiziente Nutzung optischer Verbindungen in der Rechnerarchitektur direkte optische Kanäle zwischen VLSI-Bausteinen benötigt. Solche Verbindungen sind nur möglich, wenn es gelingt, optische Sender und Empfänger direkt auf konventionellen VLSI-Komponenten zu integrieren. Bausteine, die dies bewerkstelligen, werden Smart Pixels (SP) genannt.

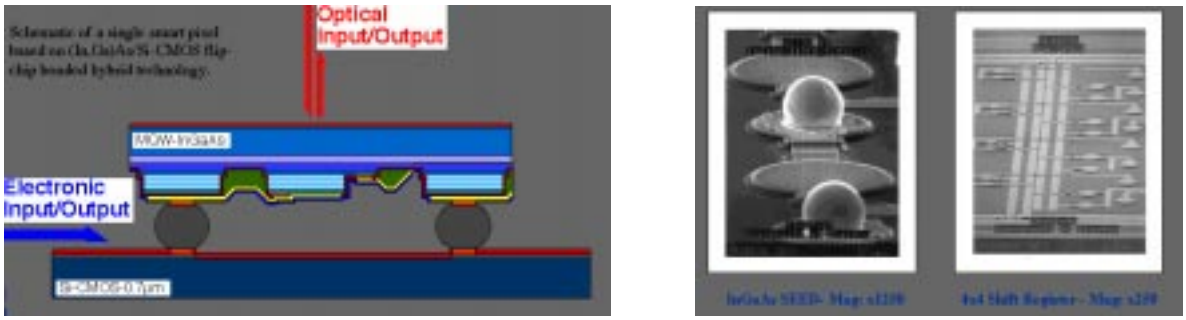
Die Herausforderung bei der Herstellung von SPs besteht darin, daß die meisten optisch aktiven Bauteile nicht kompatibel mit den Standard-VLSI Prozessen sind. Dies liegt daran, daß die Lichterzeugung und zum Teil auch die Detektion die Energiedifferenzen verschiedener Ladungsträgerzustände eines Halbleiters ausnutzen. Dazu ist es notwendig, daß diese Energiedifferenze der Energie von Lichtquanten einer gut nutzbaren Wellenlänge (sichtbares oder infrarotes Licht) entspricht. Bei dem Grundstoff für konventionelle VLSI-Schaltkreise, Silizium, ist dies leider nicht der Fall. Optisch aktive Bauteile müssen deswegen aus einem anderen Halbleiter (z.B. GaAs) hergestellt werden. Aufgrund unterschiedlicher Kristallstrukturen lassen sich verschiedene Halbleiter aber nur schwer auf dem gleichen Wafer vereinigen. Um das Problem zu umgehen, wurden drei Ansätze untersucht: die Verwendung von GaAs Logik, die Entwicklung spezieller Zwischenschichten zur Integration von GaAs auf Silizium-Wafern, und die nachträgliche Verbindung.

Zu den bedeutendsten Entwicklungen in Bereich der Opto-Elektronik gehören die Fortschritte bei der Integration opto-elektronischer Ein/Ausgabe Komponenten mit komplexer VLSI-Logik.

Ein Überblick über neue Entwicklungen auf diesem Bereich ist in [86] zu finden. Die nachfolgende Betrachtung konzentriert sich auf das Flip-Chip Bonding Verfahren, da dies die vielversprechendste Technologie ist.

Das Prinzip des Flip-Chip-Bondens

Das Flip-Chip-Bonden ist ein Verfahren, das bereits sein geraumer Zeit zum Befestigen von VLSI-Komponenten auf Multichip-Substraten und zur Stapelung von Chips verwendet wurde. Das Prinzip ist in Abbildung 4.15 zu sehen. Zunächst werden auf der Chipoberfläche und dem Substrat passende Metallkontakte angebracht. Die Kontakte sind den Bondflächen für Kontaktdrähte ähnlich und können mühelos in den normalen VLSI-Herstellungsprozess integriert werden. Auf die Kontakte werden Tropfen aus einem weichen Metall, in der Regel Gold, aufgetragen. Der Chip wird dann umgedreht (daher flip-chip) und mit den Metalltropfen auf das Substrat (oder den anderen Chip) gesetzt. Um die Verbindung zu verfestigen, wird als letztes das Metall durch Wärme und/oder Druck geschmolzen.



a) b)

Abbildung 4.15: Das Prinzip des Flip-Chip Bondens am Beispiel von MQW-Feldern und Silizium CMOS VLSIs (a, aus [134]). Im Bild b sind der GaAs Chip mit den Metalltropfen (links) und der zum Bonden fertige CMOS-Wafer zu sehen (ebenfalls aus [134]).

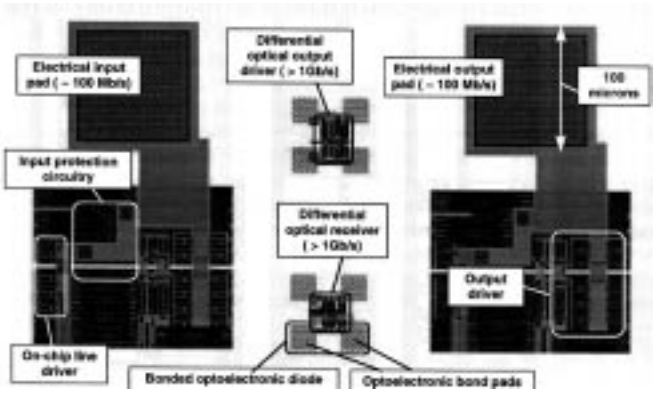


Abbildung 4.16: Größenvergleich der Treiber und Kontakte, die für einen konventionellen elektrischen Anschluß an einen VLSI-Baustein und einen optischen Anschluß an einen SP benötigt werden (aus [123]).

Eigenschaften von SPs

Für die Herstellung von SPs ist das Flip-Chip-Bonden aus drei Gründen besonderes gut geeignet:

1. Das Verfahren erlaubt die Verbindung mehrere Zentimeter großer Chips in einem Produktionsschritt. Hinzu kommt, daß die Metalltropfen durch ihre Oberflächenspannung beim Verbinden leichte Positionierungsungenauigkeiten automatische korrigieren. Dies führt dazu, daß eine billigen Massenherstellung von SPs möglich ist.

2. Die benötigten Kontakte können sehr klein sein (10 bis 20 μm Durchmesser). Sie sind insbesondere um einen Faktor 5 bis 10 kleiner als die für konventionelle Kontaktdrähte benötigten Pads.

3. Die Verbindung über kleine Metallkontakte und Goldtropfen hat eine geringe Kapazität und eine geringen Widerstand. Sie verhält sich bei der Datenübertragung ähnlich einer dicken On-Chip Leitung. Damit werden keine aufwendigen Treiber benötigt. Gleichzeitig können eine hohe Bandbreite und geringe Latenz realisiert werden.

Punkte 2 und 3 führen dazu, daß bei SPs das Verhältnis zwischen der Rechenleistung und der Ein/Ausgabe Bandbreite um einen Faktor von 10 bis 100 besser ist als bei konventionellen VLSI-Chips. Damit wird die Pin-Beschränkung beseitigt. Hinzu kommt, daß die Bandbreite der Ein/Ausgabe mit der Leistung der VLSI-Technologie skaliert. Dies liegt daran, daß die Bandbreite der optischen Ein/Ausgabe Kanäle durch die Leistung der elektronischen Treiber beschränkt ist. Die eigentlichen optischen Sender und Empfänger lassen, wie bereits beschrieben, eine Datenrate im Bereich von 100 Gbit/s pro Kanal zu. Auch die Verbindung über die Kontakte und die Metalltropfen stellt keine nennenswerte Beschränkung dar. Die Skalierbarkeit der Ein/Ausgabe-Bandbreite von SP-Bausteinen mit der Leistungsfähigkeit der VLSI-Technologie wurde in [16] theoretisch für auf MQW-Dioden basierten SPs nachgewiesen.

Ein weiterer wichtiger Aspekt der SP-Bausteine ist die geringe, im wesentlichen durch die Signallaufzeit und Gatterschaltzeit bestimmte Latenz der optischen Ein/Ausgabekanäle. Die Schaltzeit optischer Ein/Ausgabe-Bausteine beträgt Bruchteile einer Nanosekunde. Trotzdem liegt die Latenz der meisten kommerziell verfügbaren optischen Datenübertragungssysteme zwischen 10 und 100ns. Dies liegt daran, daß vor der eigentlichen Übertragung in der Regel eine Serialisierung und eine spezielle Kodierung der Daten stattfinden muß. Ersteres ist notwendig, weil der optische Sender seine Daten über gewöhnliche elektrische Leitungen auf einer Platine bekommt. Da diese eine wesentlich niedrigere Bandbreite besitzt, werden viele (10 bis 100) parallele Leitungen benötigt, um den Sender mit ausreichend Daten zu versorgen. Dieser niederfrequente, parallele Datenstrom muß vor dem Senden in einen sequentiellen, hochfrequenten Datenstrom umgewandelt werden. Die Datenkodierung wird benötigt, weil die klassische optische Datenübertragung seriell in einer einzelnen Faser stattfindet. Es gibt also kein gesondertes Clock-Signal, um die Übertragung zu synchronisieren. Aus diesem Grund können nur geeignet kodierte Daten fehlerfrei übertragen werden. Bei der Datenübertragung zwischen SP-Bausteinen spielen die oben beschriebenen Faktoren praktisch keine Rolle. Da die optischen Ein/Ausgabekanäle direkt an die Logik angeschlossen sind, können sie mit der Betriebsfrequenz des Chips mit Daten versorgt werden. Bei einer Datenrate, die maximal um einen Faktor von 4 bis 8 über dieser Frequenz liegt, wird damit nur ein geringes Maß an Sequentialisierung benötigt. Gleichzeitig erlaubt die große Anzahl von Ein/Ausgabekanälen auf einem SP-Chip eine gesonderte Clock-Leitung für jeden Kanal. Damit ist die Latenz nur durch die Signallaufzeit und die Schaltzeiten der Sender und Empfänger bestimmt. Sie liegt damit im Bereich der Latenz von langen On-Chip Verbindungen.

Beispiel für SPs: MQW-Dioden basierte SPs (OEVLSI)

Zu den ersten und inzwischen am weitesten fortgeschrittenen SPs gehören Bausteine, die Felder von MQW-Dioden mit konventionellen CMOS-Bausteinen kombinieren. Ihre Entwicklung wurde vor allem bei den Bell-Labs in USA [40] und an der Heriot-Watt Universität in Schottland vorangetrieben. Sie werden in der Literatur oft als OEVLSI bezeichnet. Ein großer Vorteil des OEVLSI-Konzeptes besteht darin, daß die MQW-Dioden sowohl als Modulatoren (Sender) als auch als Photodioden (Empfänger) benutzt werden. Welche Funktion eine Diode erfüllt, hängt lediglich von dem VLSI-Treiber ab, an den sie angeschlossen ist. Darüber hinaus haben MQW-Modulator-Sender eine geringere Energie- und damit Wärmedissipation als Laserdioden. Der größte Nachteil der OEVLSI-Bauteile besteht darin, daß die Nutzung von Modulatoren als Sender ein komplexes optisches System erfordert.

MQW-basierte SPs wurde für viele Forschungsprototypen verwendet. Die Bell-Labs haben bereits zweimal ausgewählten Forschungsteams die Möglichkeit geboten, eigene SPs zu entwerfen. Es handelte sich dabei um 2x2 bzw. 4x4 Millimeter große CMOS-Chip mit 200 bzw. 400 MQW-Dioden. Dabei wurden u.a. ein einfacher Prozessor mit optischer Ein/Ausgabe [83], ein Router-Chip für Hochleistungsnetzwerke [175] und ein assoziativer opto-elektronischer Chip implementiert. Der Autor der vorliegenden Arbeit hat in diesem Rahmen drei Chips entworfen: einen einfachen opto-elektronischen Speicherbuffer [108], einen einfachen Prototypen für die PHOTOBUS -Architektur [104, 105] und einen opto-elektronischen Multiport-Speicherbaustein [105]. Die Technologie wird zur Zeit von Bell-Labs halbkommerziell als Forschungsprototyp angeboten. Um ihre Leistung zu demonstrieren, wurden bei Bell-Labs verschiedene Prototypen hergestellt. Zu den bedeutendsten gehören:

- Der bisher größte OEVLSI Baustein wurde mit 128x128 Dioden realisiert.
- Die größte Dichte der MWQ-Dioden wurde in einem Feld von 32x64 Modulatoren erreicht, das auf einer Fläche von 2,3x2,3 mm untergebracht war. Solche Chips können heute routinemäßig mit einer Ausbeute von mehr als 99,95%, also maximal einer defekten Diode pro Feld, hergestellt werden.
- Als Beispiel für eine Anwendung in der Vermittlungstechnik wurde ein 7x7mm große CMOS Crossbar-Schalter demonstriert. Auf dem Chip war in einem 5,44x5,44mm großen Bereich ein 64x68 Feld von MQW-Dioden integriert [98]. Der Chip war in $0.7\mu m$ Technologie hergestellt und beinhaltete 140.000 Transistoren, von denen weniger als 10 % auf die Treiber der opto-elektrponischen Ein/Ausgabe entfielen. Die optischen IOs wurden mit einer Datenrate von bis zu 400MHz betrieben. Damit konnte eine Gesamtbandbreite von ca. 1,6TBit/s erreicht werden. In der $0.7\mu m$ Technologie entsprechen 400MHz in etwa der vierfachen Prozessortaktrate, was zu eine Übertragungsleistung von 256, 64-Bit Worten pro Prozessorzyklus führt.
- Es wurde ein Chip implementiert, der eine optische Eingabe mit 2,48Gbit/s pro Kanal realisiert [169].

Die obigen Beispiele zeigen, daß mit der OEVLSI-Technologie die Integration von Tausenden von optischen Kanälen auf konventionellen VLSI-Bausteinen an der Schwelle zur Serienreife angelangt ist.

4.5.5 Beispiel für SPs: VCSEL/Detektor basierte SPs

Das komplexe optische System, das für die Anwendung der Modulatoren als Sender notwendig ist, hat ein großes Interesse an VCSEL-Laserdioden basierten SPs hervorgerufen. Dabei wurden Felder von bis zu 256 Laserdioden erfolgreich auf CMOS-Schaltkreisen integriert [38]. Auch gemischte Felder von Laserdioden und Photodioden wurden hergestellt [144]. Zu Zeit wird bei der Firma Honywell an der Integration mehrerer Tausend Laserdioden gearbeitet [96]. An größeren Feldern wird zur Zeit intensiv gearbeitet.

4.6 Fortschritte beim Aufbau optischer Übertragungsstrecken

Im Abschnitt 4.3 wurde deutlich, daß Probleme mit stabilen, kompakten Aufbauten eines der Haupthindernisse für die Nutzung optischer Datenübertragung im Bereich der Rechnerarchitektur darstellen. Dies ist vor allem bei Freiraumsystemen ein Problem. Dort müssen eine große Anzahl disjunkter Komponenten im Bezug aufeinander bis auf wenige Mikrometer genau justiert und fixiert werden. Das System muß so aufgebaut sein, daß die Justierung über einen längeren Zeitraum bestehen bleibt und nicht von Umweltfaktoren wie Erschütterungen und Temperaturschwankungen beeinflußt wird. Ein ähnliches Problem stellt sich bei hochparallelen Fasersystemen. Bei solchen Systemen ist vor allem die Implementierung von Steckverbindungen und die Anbindung der Faser an die Sender und Empfänger problematisch. In beiden Fällen müssen alle Faser des Systems in einer stabilen ein- oder zweidimensionalen Anordnung fixiert werden. Je nach Art der Faser wird dazu eine Genauigkeit im Bereich eines Mikrometers erforderlich.

In letzer Zeit wurden im Bereich der mechanischen Aufbauten enorme Fortschritte erzielt. Zum einen wurden Konzepte für einen sehr kompakten und stabilen Aufbau von konventionellen optischen Bänken entwickelt. Zum anderen sind durch die Mikro- und Nanotechnologie Möglichkeiten geschaffen worden, komplexe mechanische und optische Systeme sehr stark zu miniaturisieren. Aufgrund der großen Bedeutung der zweiten Entwicklung wird im Nachfolgenden zunächst ein kurzer Überblick über die Mikrosystem-Technik gegeben. Danach werden die verschiedenen neuen Ansätze für den Aufbau von optischen Freiraumsystemen und parallelen Faserbündeln erläutert. Ein ausgiebiger Überblick über die Technologie ist z.B. in [87, 155] zu finden.

4.6.1 Mikrosystem-Technik

Die billige Herstellung komplexer Strukturen mit einer Auflösung und Genauigkeit unterhalb eines Mikrometers ist bei der Produktion von VLSI-Schaltungen Routine. Sie basiert auf der Lithographie. Das Verfahren besteht aus drei Schritten: Zuerst werden die benötigten Strukturen auf eine Maske geschrieben. Diese Maske wird zur Belichtung einer Photolack-Schicht verwendet, die zuvor auf einen Siliziumwafer aufgetragen wurde. Als nächstes wird die Photolackschicht entwickelt. Dabei werden die belichteten Stellen entfernt, so daß die Siliziumoberfläche nur an den durch die Maske vorgegebenen Stellen zum Vorschein kommt. An dieser Stelle setzt die eigentliche Siliziumberabeitung ein. Dazu wird der gesamte Wafer einer Bearbeitung unterzogen, die auf das Silizium, nicht aber auf den übriggebliebenen Photolack wirkt. Dabei kann es sich um einfaches Ätzen, die Ablagerung von Zusatzschichten oder besondere Arten von Bestrahlung handeln, die die Materialeigenschaften verändern. Als letztes wird mit einem speziellen Lösungsmittel die restliche Photolackschicht entfernt.

Leider eignet sich die gewöhnliche Lithographie nur zur Herstellung von zweidimensionalen Strukturen, deren Tiefe zwischen $1\mu m$ und $10\mu m$ beträgt. Dies liegt daran, daß sowohl das Belichten als auch das Ätzen nur bis zur einer geringen Tiefe wohldefinierte Strukturen erzeugen kann. Das Licht wird mit steigender Belichtungstiefe zunehmend gestreut. Somit wird nicht nur die durch die Maske bestimmte Struktur, sondern auch ein diffuser, undefinierter Bereich bestrahlt. Ein ähnliches Problem tritt beim Tiefen-Ätzen auf. Das Ätzmittel wirkt in der Regel nicht nur vertikal, sondern auch horizontal. Mit zunehmender Tiefe läuft die Struktur also auseinander und entspricht nicht mehr der Maske.

Um trotz obiger Probleme komplexe, mikroskopische Systeme herzustellen, wurden verschiedene Alternativverfahren entwickelt. Die wichtigsten davon sind: mechanische Präzisionswerkzeuge, Silizium-Oberflächenbearbeitung, fortgeschrittene Ätzverfahren und LIGA. Sie werden im Nachfolgenden kurz geschildert.

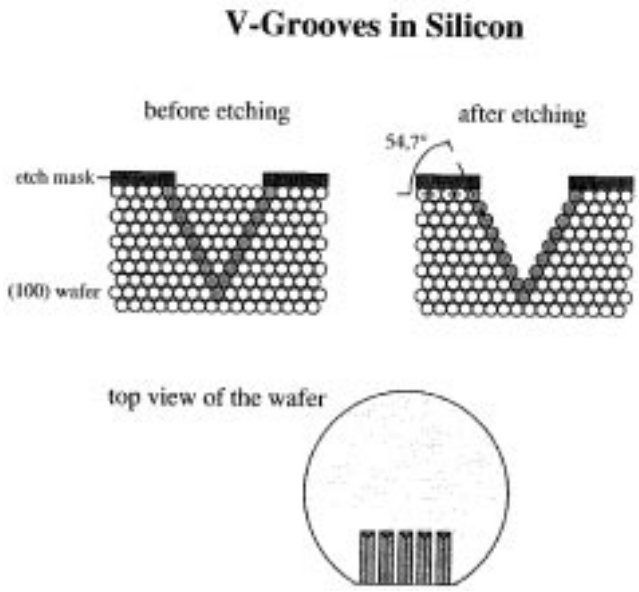


Abbildung 4.17: Das Prinzip der Herstellung von V-Nuten durch unsymmetrisches Ätzen (aus [155]).

Präzisionswerkzeuge

Computergesteuerte, mechanische Präzisionswerkzeuge sind heute in der Lage, eine Genauigkeit von weniger als $10\mu m$ zu erreichen. Damit sind sie prinzipiell für die Herstellung vieler optischer Systeme geeignet. Eine noch bessere Genauigkeit kann durch die Verwendung von Laserstrahlen anstelle mechanischer Werkzeuge erreicht werden. Dabei wird ein sog. Excimer-Laser mit einigen Watt Leistung dazu benutzt, den Werkstoff an vorbestimmten Stellen zu verdampfen. Das als Laserablation bekannte Verfahren hat eine Genauigkeit von bis zu $2\mu m$ und erlaubt eine Strukturtiefe von bis zu einem mm.

Ein großes Problem bei der Verwendung von Präzisionswerkzeugen zur Herstellung komplexer Mikrostrukturen besteht darin, daß die einzelnen Strukturelemente sequentiell erzeugt werden. Dadurch wird die Genauigkeit mit wachsender Anzahl von Strukturelementen immer schlechter, da sich die Positionierfehler von Schritt zu Schritt summieren. Hinzu kommt, daß so keine billige Massenproduktion möglich ist.

Silizium-Oberflächenberarbeitung

Die Oberflächenbearbeitung erlaubt es, trotz der geringen Strukturtiefe dreidimensionale Elemente durch herkömmliche Lithographie auf Siliziumwafern zu erzeugen. Dabei werden zunächst dünne Bauteile auf der Waferoberfläche liegend erzeugt. Sie sind mit speziellen elektro-mechanischen Vorrichtungen versehen, die sich beim Anlegen einer Spannung ausdehnen oder verbiegen. Diese Vorrichtungen werden bei fertigen Chips benutzt, um die liegenden Strukturen aufzurichten und zu fixieren (Abbildung 4.23a). Das Verfahren wurde erfolgreich zur Herstellung von mikroskopischen optischen Bänken auf der Oberfläche von Silizium-Chips verwendet. Ein Beispiel dafür ist in Abbildung 4.23b zu sehen.

Fortgeschrittene Ätzverfahren

Die Tatsache, daß das Ätzen nicht nur vertikal sondern auch horizontal stattfindet, kann man zur Herstellung von Nuten für die Positionierung von Fasern und anderen Komponenten nutzten. Dabei verwendet man Kristalle, bei denen die horizontale Ätzgeschwindigkeit um einen wohldefinierten Faktor geringer ist als die vertikale [87]. Dies führt dazu, daß beim Tiefen-Ätzen eine V-förmige Rille entsteht (Abbildung 4.17). Die Maße dieser Rille können mit einer Genauigkeit von ca. $1\mu m$ kontrolliert werden. Ihre Lage, und die Breite der Öffnung werden wie bei der Herstellung von VLSI-Strukturen lithographisch festgelegt.

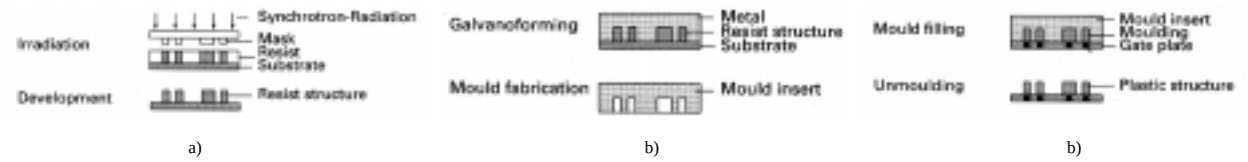
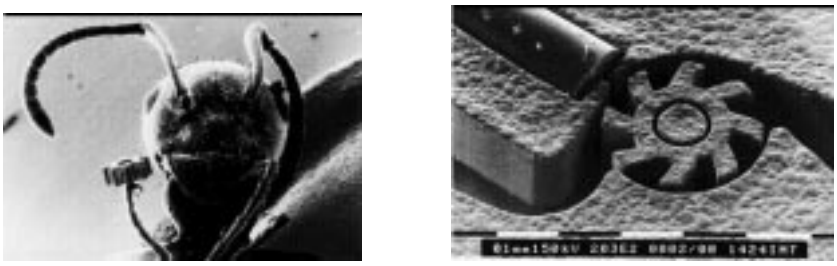


Abbildung 4.18: Die drei Schritte des LIGA-Verfahrens (aus [87]): Herstellung einer Form durch Lithograhie (a), Herstellung eines Matallstempels durch Galvanisierung der Form (b) und die Massenproduktion durch Abformung mit Hilfe des Stempels (c).



a) b)

Abbildung 4.19: Beispiele für mit LIGA hergestellte Strukturen: ein mikroskopisches Zahnrad vom KFK Karlsruhe (a) und eine am IMM-Mainz hergestellte mikroskopische Pumpe (b aus [51]).

LIGA

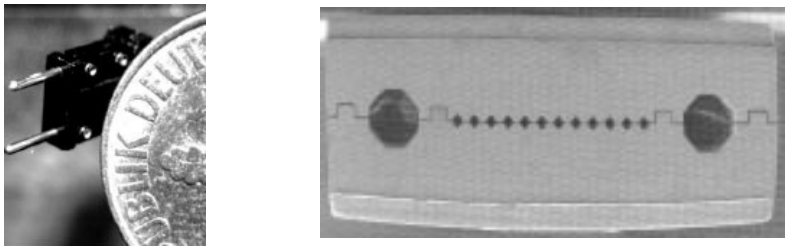
Wie oben geschildert besteht das Problem bei der lithographischen Herstellung tiefer Strukturen darin, daß mit zunehmender Tiefe die Belichtungsstrahlung immer stärker gestreut wird. Das LIGA-Verfahren (**Li**thographie, **G**alvanoformung und **A**bformung) umgeht dieses Problem, indem es eine sehr hochenergetische Strahlung (Synchrotron Strahlung) für die Belichtung verwendet [35]. Bei solcher hochenergetischer Strahlung nimmt die Streuung erst bei einer wesentlich größeren Tiefe ein relevantes Maß an. So können bis zu 2mm tiefe Strukturen hergestellt werden. Allerdings ist diese Art von Lithographie sehr aufwendig, und somit nicht für die billige Massenherstellung geeignet. Aus diesem Grund wird die Lithographie beim LIGA-Verfahren nur zur Herstellung von Vorlagen verwendet. Die eigentlichen Werkstücke werden durch Prägen oder Gießen aus diesen Vorlagen abgeformt. Da die für die Lithograpie verwendeten Materialien in der Regel nicht robust genug für die Abformung sind, wird für die Herstellung der Präge- und Gießformen ein zusätzlicher Schritt benötigt. Dabei wird durch galvanische Metallablagerungen an der Vorlage eine Metallform hergestellt. Die drei Schritte des LIGA-Verfahrens sind in Abbildung 4.18 dargestellt.

Das LIGA-Verfahren erlaubt die billige Produktion von Strukturen mit einer Tiefe bis zu 2mm und einer Genauigkeit im Bereich von $1\mu m$. Die Gesamtgröße der Werkstücke kann ohne Einbußen bei der Genauigkeit mehrere Zentimeter betragen. Damit ist LIGA eine der wichtigsten Technologien im Bereich der Mikrosystem-Technik und Mikrooptik. Zwei Beispiele für ihre Anwendung sind in Abbildung 4.18 zu sehen.

4.6.2 Fasersysteme

Die Fortschritte im Bereich der Mikrosystemtechnik haben dazu geführt, daß eindimensionale Faserbündel mit bis zu 32-Elementen inzwischen kommerziell erhältlich sind. Die exakte Positionierung der Faser wird durch geeignete V-Nuten erreicht (Abbildung 4.17), in denen die Faser von oben eingelegt werden. Mit dieser Technik ist es möglich, sowohl Stecker herzustellen als auch eine Anbindung von Faserbündeln an Sender- und Empfängerfelder zu gewährleisten. Ein Beispiel für einen kommerziellen Stecker ist in Abbildung 4.20 zu sehen.

Neben eindimensionalen Bündeln wurden auch zweidimensionale demonstriert, die über 1000 Elemente beinhalten können. Um die Faser in dem benötigten zweidimensionalen Muster zu fixieren, werden sie in Platten gesteckt, in denen vorher mit Hilfe der Mikrosystem-Technik geeignete Öffnungen hergestellt wurden. Die Schwierigkeit besteht dabei darin, daß das Einfügen der Faser in die Öffnung eine sehr präzise Positionierung erfordert. Dadurch ist eine billige Massenherstellung nicht möglich. Allerdings werden zur Zeit verschiedene Ansätze untersucht, den Positioniervorgang zu vereinfachen und zu automatisieren. Sie sind als Sonderanfertigungen zum Teil sogar kommerziell erhältlich. Ein solches Bündel der Firma Fiberguide ist in Abbildung 4.21) dargestellt.



a) b)

Abbildung 4.20: Stecker für parallele eindimensionale Faserbündel. Abbildung a) zeigt einen Stecker für 12 Faser neben eine 10Pf-Münze. Abbildung b zeigt eine Vergrößerung der Frontpartie des Steckers (aus [51]).

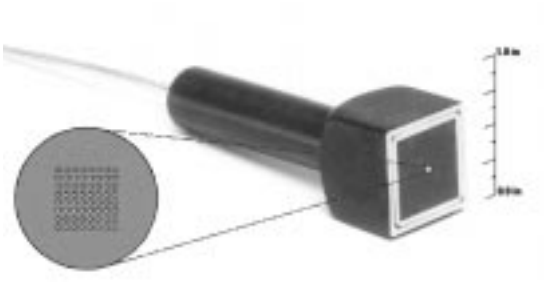


Abbildung 4.21: Beispiel für zweidimensionale Faserbündel: ein Bündel der Firma Fiberguide mit bis zu 1024 Fasern.

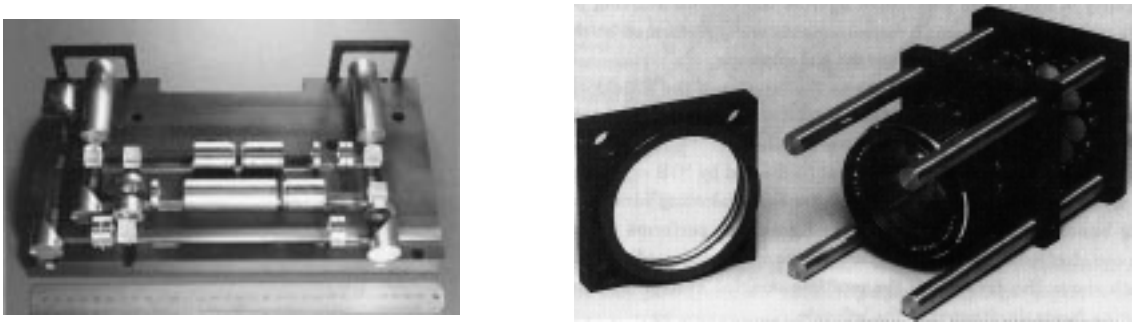
4.6.3 Freiraumoptik

Mikroskopische integrierte Systeme

Die Stärke der Freiraumoptik liegt vor allem darin, daß sie komplexe Verbindungsstrukturen und hohe Kanalzahlen auf engstem Raum zuläßt. Um sie voll auszunutzen, ist es notwendig, optische Systeme mit Hilfe der Mikrosystemtechnik zu miniaturisieren. Solche miniaturisierten Systeme können unter anderem benutzt werden, um mehrere SP-Chips miteinander und mit massiv parallelen Faserbündeln zu verbinden.

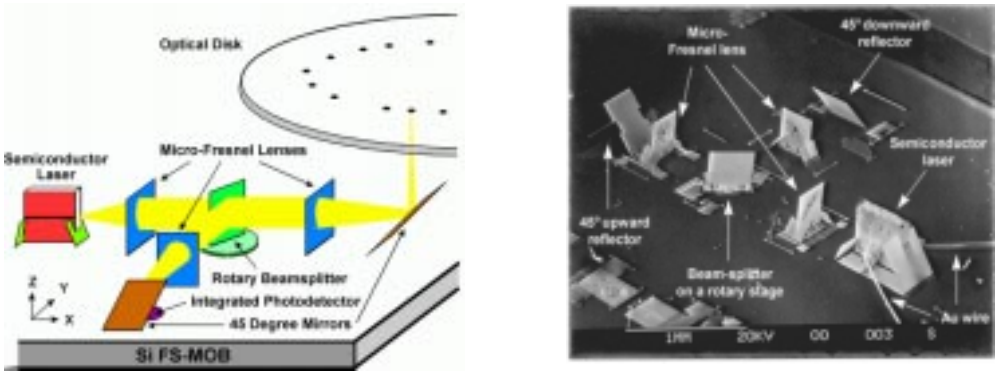
Einfache integrierte Freiraumoptik wird bereits im Bereich der Sensorik und zum Teil in optischen Diskspeichern kommerziell verwendet. Komplexere, für den Einsatz in der Rechnerarchitektur geeignete Freiraumoptik befindet sich zur Zeit in der Entwicklung und wurde von verschiedenen Forschergruppen in Form von Prototypen demonstriert. Dabei werden für den Systemaufbau vor allem zwei Ansätze verwendet: mikroskopische optische Bänke, integrierte Schichtoptik und integrierte planare Optik.

Mikroskopische Optische Bänke: Mikroskopische optische Systeme nach dem Vorbild klassischer optischer Bänke können mit LIGA Technologie oder Oberflächenbearbeitung von Silizium hergestellt werden. Mit dem



a) b)

Abbildung 4.22: Beispiele für neuartige mechanische Systeme zum kompakten Aufbau von Freiraumoptiken: slotted baseplate System (a aus[4]) und Stangensystem (b, aus [188])



a) b)

Abbildung 4.23: Ein CD-Lesekopf als Beispiel für mikroskopische optische Bänke und Silizium Oberflächenbearbeitung (aus [89]).

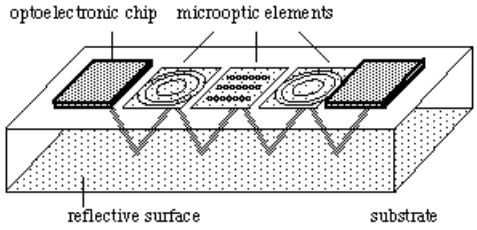


Abbildung 4.24: Das Prinzip der planar integrierten Freiraumoptik (aus [74]).

LIGA-Verfahren können in einer Platte Löcher erzeugt werden, in denen entsprechende Mikrokomponenten plaziert werden ([10]). Ein Beispiel für komplexe optische Bänke, die mit Oberflächenbearbeitung hergestellt wurden, zeigt Bild 4.23. Es handelt sich dabei um einen auf einer Chipoberfläche integrierten Lesekopf für optische Diskspeicher, der an der UCLA in USA realisiert wurde [89]. Das System zeigt, daß heute bereits komplexe, in der Praxis nutzbare Systeme möglich sind.

Planare Integrierte Freiraumoptik: Die integrierte planare Optik stellt ein Systemkonzept dar, das einfach herzustellen ist und eine hohe Systemstabilität bietet [77, 75, 154]. Die Grundidee ist in Abbildung 4.24 dargestellt. Das optische System wird 'gefaltet' und auf die Oberfläche eines Glasträgers aufgetragen. Bis auf wenige wohldefinierte Einkoppelstellen werden dabei die Ober- und Unterseite des Glasträgers mit einer reflektierenden Schicht bedeckt. Die optischen Bauelemente werden in diese Schicht eingeätzt. Sie müssen dazu so modifiziert werden, daß sie reflektiv, statt transmittiv wirken. Die Strahlausbreitung findet bei der integrierten Freiraumoptik vollständig im Inneren der Glasplatte statt. Dadurch ist das System gegenüber Umwelteinflüssen unempfindlich. Gleichzeitig kann es einfach mit lithographischen Methoden hergestellt werden.

An der Entwicklung der integrierten planaren Optik wird zur Zeit an der Universität Hagen gearbeitet. Dort wurden unter anderem ein Taktverteilungssytem für große SP-Chips und Multichipmodule, und ein Datenbus mit einer Kanaldichte von 1024 Kanällen pro $1,6mm^2$ demonstriert [154]. Dabei wurde auch gezeigt, daß sich solche Systeme durch Flip-Chip-Bonding mit LD-Feldern und SP-Bausteinen kombinieren lassen.

Kompakte Makroskopische Systeme

Obwohl die integrierten optische Systeme große Fortschritte gemacht haben, befinden sich komplexe Systeme immer noch im Entwicklungsstadium. Wesentlich näher an der Anwendung sind verbesserte, kompakte aber dennoch makroskopische Systeme. Solche System sind in Form von Sonderanfertigungen selbst für komplexe Aufbauten kommerziell erhältlich. Zwei Beispiele hierfür sind in Abbildung 4.22 zu sehen. Es handelt sich dabei um ein sog. slotted baseplate System und ein stangenbasiertes System.

Slotted Baseplates: Die slotted baseplate Technik wurde bei den Bell-Labs zum Aufbau von Demonstratoren optischer Freiraumnetzwerke entwickelt. Sie basiert auf Platten aus stabiler Metalllegierung, in denen passend zur Geometrie des optischen Systems präzise Nuten (slots) eingearbeitet sind. Die optischen Bauteile befinden sich in Fassungen, die in diese Rillen eingelegt und mit Hilfe von Magneten befestigt werden können.

Stangensysteme: Stabile, kompakte Aufbauten können durch Verbindung der optischen Komponenten durch speziell gefertigte Stangen erreicht werden. Die Anwendung solcher Systeme wurde z.B. in [188] demonstriert.

Weitere Konzepte für stabile opto mechanische Systeme wurden [120] und [70] beschrieben und demonstirert.

4.7 Fortschritte in der Systemintegration

Die vorigen beiden Abschnitte haben die Fortschritte bei der Herstellung verschiedener Komponenten aufgezeigt, die die Voraussetzung für die Realisierung paralleler optischer Verbindungen sowie direkter optischer Verbindungen zwischen VLSI-Bausteinen bilden. Fortschritte bei der Integration dieser Komponenten zu vollständigen Systemen und das Potential dieser Systeme in der Rechnertechnik können wie folgt zusammengefaßt werden.

1. Eindimensionale parallele Faserübertragungssysteme mit einer Bandbreite bis zu 30Gbit/s sind kommerziell bzw. als Laborprototypen erhältlich. Damit sind optische Verbindungen auf Rack-to-Rack und Board-to-Board Ebene auf der Basis kommerzieller Komponenten möglich. Die Bandbreite und Latenz solche Verbindungen wäre mit den leistungsfähigsten heutigen globalen Chip-to-Chip Verbindungen vergleichbar. Damit wäre eine Implementierung des SMP-Speichermodels bei vernetzten Arbeitsplatzrechnern möglich.

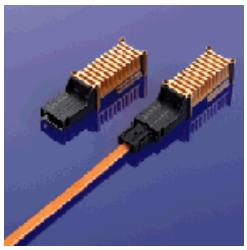
2. Integrierte opto-elektronische Systeme, die Verzweiger und Wellenlängenmultiplexer für mehrere Fasern mit aktiven Komponenten wie Sendern, Verstärkern und Empfängern in einem Halbleitermodul vereinigen. Mit Hilfe solcher Module ist es z.B. möglich, einen Fanout mit anschließender Verstärkung für ein Faserbündel in einem kompakten, billigen Baustein zu realisieren. Dies ist für die Realisierung von Rundruf-Architekturen von großer Bedeutung.

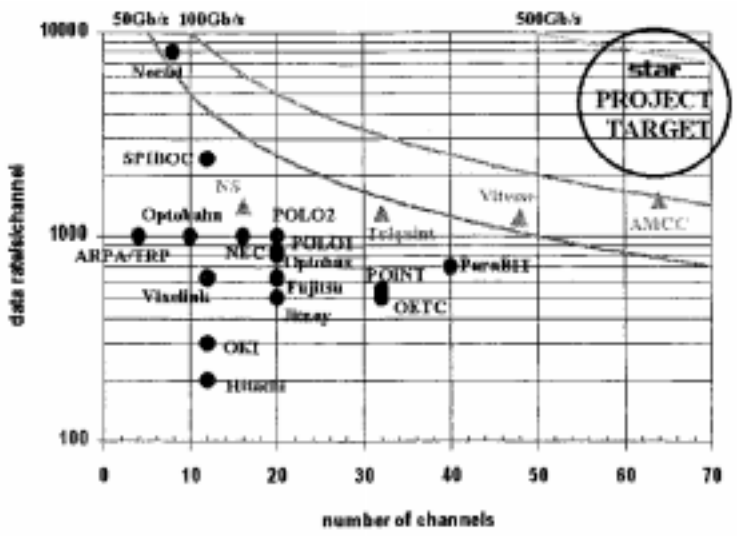
3. Zweidimensionale Fasersysteme mit Hunderten von Fasern, die über ein einfaches optisches System an einen SP-Baustein angebunden sind, wurden von mehreren Forschergruppen demonstriert. Sie machen eine Bandbreite zwischen 100Gbit/s und 1TBit/s möglich und können sowohl in globalen Chip-to-Chip als auch in Board-to-Board und Rack-to-Rack Verbindungen sinnvoll eingesetzt werden. Sie können aus Komponenten aufgebaut werden, die als Sonderanfertigungen bzw. Laborprototypen verfügbar sind. Gleichzeitig bieten sie eine Bandbreite, die die besten heutigen elektronischen Netzwerke um Faktor 10 bis 100 übertrifft. Sie ermöglichen so unter anderem eine deutliche Verbesserung der Skalierbarkeit von SMPs.

4. Komplexe System die Faser, SPs und mikroskopische Freiraumoptik integrieren, sind von verschiedenen Forschergruppen demonstriert worden. So scheint es plausibel, daß durch ein gezieltes Forschungsprogramm ein System optisches System für den Einsatz als Verbindungsnetzwerk auf lokaler Chip-to-Chip Ebene oder gar auf On-Chip Ebene gebaut werden kann. Ein solches System würde auch die Anbindung massiv paralleler zweidimensionaler Faserbündel mit Wellenlängenmultiplexing an SPs ermöglichen. Solche Systeme wären zu einer Bandbreite von weit über ein TBit/s fähig. Dies würde unter anderem den Weg für SMPs mit Hunderten von Prozessoren eröffnen.

Parallele Faserübertragungssysteme

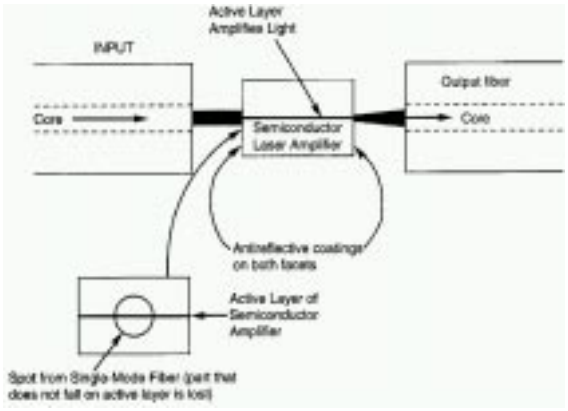
Die Entwicklung von parallelen Faserübertragungssystemen wird von vielen Firmen intensiv vorangetrieben. Sie benutzen Laserdioden-Zeilen und V-Gruben für das Befestigen der Faser. Zu kommerzieller Reife wurden bisher zwei Systeme gebracht: das OPTOBUS-System von Motorola und das PAROLI System von Siemens. Ersteres verfügt über 10 Faser, jede mit einer Bandbreite von 800MBit/s. Letzteres besitzt 12 Kanäle 1,2Gbit/s. Eine Übersicht über andere parallele Fasersysteme ist in Abbildung 4.25 zu sehen. Zur Zeit wird an 2D- Faserünertragungssystemen gearbeitet, die mit SP-Bausteinen verbunden sind [115].



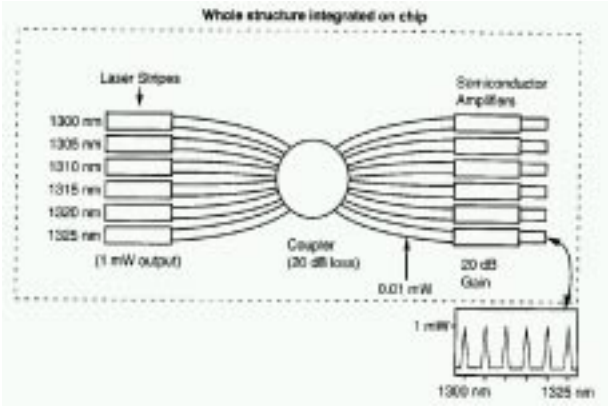


a) b)

Abbildung 4.25: Abbildung a zeigt zwei Beispiele für parallele Faserünertragungssysteme: der OPTOBUS von Motorola (oben) und das PAROLI-System von Siemens (unten). Die Tabelle in Abbildung b gibt einen Überblick über verschiedene parallele Fasersysteme (aus [177]).

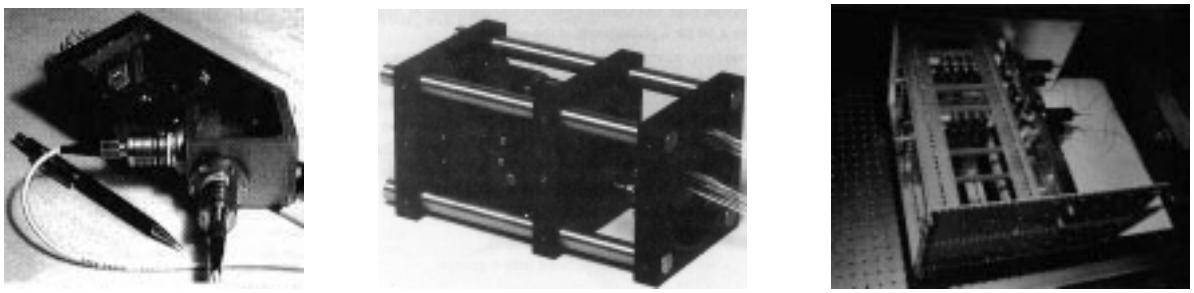


a) Halbleiterverstärker



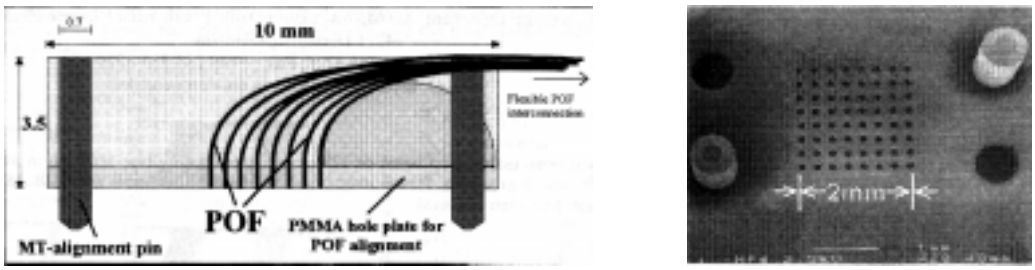
b) integriertes opto-elektronisches system

Abbildung 4.26: Das Linke Bild zeigt das Prinzip eines Halbleiter-Lichtverstärkers (SOA). das rechte Bild zeigt ein integriertes opto-elektronisches System mit Sendern verschiedener Wellenlänge, einem Sternverteiler und Verstärkern (aus [67]).



a) b) c)

Abbildung 4.27: Beispiele für die makromechanische Integration von Faserbündeln, SPs und optischen Komponenten. Abbildung a zeigt das AMOEBA-System von Bell-Labs [85]. Abbildung b zeigt ein Teil des Kaleidoskop-Systems aus Delft ([188]). In Bild c ist der Aufbau des Hyperplane Freiraumsystems der McGill Univeristät zu sehen ([181]).



a) b)

Abbildung 4.28: Mikromechanische Integration von Faserbündeln und SP-Bausteinen am Beispiel des POF-Systems der Universität Dortmund (aus [110]). Abbildung a zeigt das Prinzip, Abbildung b eine fertige Befestigungsplatte für 64 Faser.

Integrierte opto-elektronische Systeme

Integrierte optische Verzweiger und Sternkoppler können aus Halbleitermaterialien hergestellt werden, in den sich auch aktive Komponenten realisieren lassen. Somit ist es möglich, integrierte opto-elektronische Komponenten zu bauen, die Sender, Empfänger und verschiedene Verzweiger auf einem 'opto-elektronische IC' vereinigen. Mit der Entwicklung von optischen Signalverstärkern auf Halbleiterbasis (SOA für semiconductor optical amplifier [36]) Verstärkern wurde auch die Möglichkeit geschaffen, Verstärker in solche Systeme zu integrieren. Ein Beispiel für ein solches system ist in Abbildung 4.26 zu sehen. Ein solcher Bauteil wurde z.B. im Rahmen des LAMBDABUS-Projektes ([11]) gebaut.

Makromechanische, SP-Basierte Systeme

SPs und zweidimensionale Faserbündel können genauso wie konventionelle optische Bauelemente in die in 4.6.3 beschriebenen, kompakten mechanischen Systeme integriert werden. Der Vorteil solcher Systeme besteht darin, daß alle Komponenten kommerziell gegebenenfalls in Form von Sonderanfertigungen verfügbar sind. Sie wurde vielmals zum Aufbau von Prototypen verwendet. So wurde beispielsweise bei den Bell-Labs über 500 Faser an den OEVLSI Crossbar-Chip angebunden ([98]). Ebenfalls bei den Bell-Labs wurde das in Abbildung 4.27 a dargestellte System zur Verbindung eindimensionaler Faserbündel mit einem SP-Baustein entwickelt ([85]). Das besondere an diesem Baustein ist, daß in jeder Faser bis zu 16-Wellenlängen übertragen werden können. Das optische System verteilt sie auf verschiedene Eingänge des Chips. So wird die Funktionalität eines zweidimensionalen Faserbündels mit insgesamt 160 Kanälen simuliert. Eine makromechanische Anbindung eines Faserbündels wurde auch im Zusammenhang mit dem Kaleidoskop-System demonstriert (Abbildung 4.27b).

Ein weiteres Beispiel für eine komplexe Anwendung ist in Abbildung 4.27c zu sehen. Es handelt sich dabei um eine optische Freiraum-Backplane, die an der McGill Universität gebaut wurde. Der hier gezeigt Prototyp besaß 4 SPs, jeder mit 16 optischen Kanälen [181]. Zur Zeit wird an einem Prototypen mit mehreren Hundert Kanälen gearbeitet.

Integration mit Hilfe der Mikrosystem-Technik

Bei den mikroskopischen Systeme muß man zwischen einfachen Anordnungen (z.B. zur Anbindung mittelgroßer Faserbündel an SP-Bauteile) und komplexen optischen Systemen unterscheiden. Erstere können heute im Labor problemlos hergestellt werden. Ein Beispiel für ein solches System ist in Abbildung 4.28 zu sehen. Es ist ein zweidimensionales System für 64 Fasern, das an der Universität Dortmund entwickelt wurde [110]. Ebenfalls in diese Kategorie fallen einfache planar integrierte Systeme, die in Hagen durch Bonden mit einem Feld von Laserdioden verbunden wurden.

Auch komplexe Systeme sind mit Hilfe der Mikrosystemtechnik in verschiedenen Labors realisiert worden. Beispiele sind der bereits angesprochene CD-Lesekopf von der UCLA und ein 8x8 Banyan Netzwerk, das auf der Oberfläche eines Siliziumchips verwirklicht wurde [88], und ein planar integrierter, optischer Mustererkenner [182].

4.8 Optische Verbindungen in der Rechnerarchitektur

Die Forschung im Bereiche der Anwendung optischer Netzwerke in der Rechnerarchitektur kann in drei Bereiche eingeteilt werden. Der erste beschäftigt sich mit der optischen Implementierung verschiedener, aus der Rechnerarchitektur bekannten Netzwerke [39, 150]. Dies ist bei weitem der größte Forschungsbereich. Da sein Schwerpunkt aber auf technischen Problemen der Optik und der Opto-Elektronik liegt, ist er für die vorliegende Arbeit weitgehend uninteressant. Der zweite Bereich untersucht die Anwendung optischer Freiraumverbindungen in feinkörnigen massiv parallelen Architekturen und Speichern. Zu dieser Klasse gehören massiv parallele SIMD-Maschinen, Spezialarchitekturen für Bild und Signalverarbeitung, sowie neue Konzepte für Parallelität auf Instruktionsebene [41, 135, 44]. Diese Architekturen sind nur sehr entfernt mit der vorliegenden Arbeit verwandt, so daß auch dieser Bereich im Nachfolgenden nicht weiter betrachtet wird. Der dritte Bereich beinhaltet die für die vorliegende Arbeit interessante Forschung zu optischen Verbindungen in MIMD-Rechnern. Bei nachfolgenden Beschreibungen werden die Arbeiten in vier Klassen eingeteilt:

1. Rechner, in denen elektrische Verbindungen durch einfache optische Netzwerke ersetzt wurden, ohne daß die Rechner- oder Netzwerkarchitektur dabei verändert wurde. Die Bedeutung dieser Rechnerklasse liegt darin, daß zu ihr fast alle tatsächlich implementierten opto-elektronischen Rechner gehören. Sie zeigt somit den Stand der Technik bei der praktischen Nutzung optischer Verbindungen.

2. Optische Netzwerke, die im Hinblick auf die Anwendung in Parallelrechnern entwickelt wurden. Die Arbeiten in diesem Bereich unterscheiden sich von der vorliegenden Arbeit vor allem darin, daß sie sich nur am Rande mit Fragen der Rechnerarchitektur beschäftigen. Sie konzentrieren sich statt dessen auf die Netzwerkarchitektur, -technologie und -protokolle. Ein Vergleich mit einigen Projekten aus diesem Bereich ist trotzdem sinnvoll, da der Netzwerkentwurf ein wichtiger Bestandteil der vorliegenden Arbeit ist. Wir unterscheiden dabei zwischen klassischen optischen Netzwerken und opto-elektronischen Netzwerken, die auf der Nutzung von SP-Bausteinen basieren.

3. Rechnerarchitekturen, die zusammen mit geeigneten optischen Netzwerken entwickelt wurden. In diesem Bereich sind die eigentlich verwandten Arbeiten zu finden.

4.8.1 Einfache Verbindungen in funktionsfähigen Rechnern

Diese Klasse von Arbeiten ist durch zwei Dinge motiviert. Zum einen geht es oft darum, eine bestimmte Technologie in der Praxis zu testen. Beispiele für solche Projekte sind das COSINE und das MEMSY Projekt. Zum anderen werden verschiedene optische Hochleistungsverbindungen bei der Koppelung von Arbeitsplatzrechnern eingesetzt. Man erreicht dabei eine hohe Bandbreite auch bei sehr weit voneinander entfernten Rechnern.

COSINE: im COSINE-Projekt [156] wurden Freiraumverbindungen zwischen Prozessorplatinen für die Implementierung eines Transputer-basierten Rechners mit 64 Prozessoren benutzt .

MEMSY: Im Rahmen des SFB 182 wurde in Erlangen ein optischer Bus in den Parallelrechner MEMSY integriert [109].

Rainbow: Als Test für ihre WDM-Technologie hat IBM ein 'Broadcast-and-Select' WDM-Netzwerk aus Arbeitsplatzrechnern aufgebaut. Jeder Rechner kann auf einer festen, ihm zugeordneten Wellenlänge über einen Sternkoppler einen Rundruf durchführen. Gleichzeitig kann er über einen wellenlängensensitiven Empfänger entscheiden, welche Wellenlänge er empfangen möchte. Das demonstrierte System bestand aus 32 Arbeitsplatzrechnern, jeder mit einem 1Gbit/s Sender, die über eine Entfernung von 1km verbunden waren.

Optische Verbindungen für SCI-Architekturen: Von der Nutzung paralleler Faserübertragungssysteme in SCI Arbeitsplatz-Clustern wurde in [37] berichtet.

4.8.2 Netzwerkarchitekturen und Protokolle

Optische Netzwerke mit konventionellen, nicht SP-basierten Schnittstellen stehen vor dem Problem, daß ihre Netzwerkschnittstellen nur einen Bruchteil der möglichen Zeitbandbreite und Kanalanzahl optischer Verbindungen verarbeiten können. Die meisten Arbeiten in diesem Bereich beschäftigen sich daher mit dem Entwurf von Netzwerkarchitekturen, die trotz der geringen Schnittstellenleistung eine Nutzung der Vorteile der Optik erlauben. In den meisten Fällen wird versucht, durch eine hohe Anzahl von Kanälen den Netzwerkdurchmesser zu reduzieren, und Rundruf- und Multicast-Operationen zu vereinfachen. Die Überlastung der Netzwerkschnittstelle wird durch besondere Zugriffsprotokolle verhindert. Um eine hohe Kanalanzahl einfach und robust zu realisieren, greifen die meisten Arbeiten auf faserbasierte WDM-Systeme zurück. Dies hat den Vorteil, daß der mechanische Aufbau aus einer einzigen bzw. einigen wenigen Fasern besteht.

Im Nachfolgenden werden zunächst die wichtigsten WDM-Ansätze geschildert. Danach wird ein Verfahren zur Nutzung der hohen Zeitbandbreite von optischen Systeme vorgestellt, das auch mit vergleichsweise langsamen Schnittstellen funktioniert. Zum Schluß werden zwei Ansätze zur Nutzung hoher Ortsbandbreite vorgestellt.

WDM-Netzwerke

Für die Anwendung des Wellenlängenmultiplexings in einem Netzwerk gibt es drei Möglichkeiten. Die erste Möglichkeit stellt die Nutzung mehrerer Wellenlängen zur bitparallelen Datenübertragung in einem einzelnen physikalischen Kanal (z.B. einer Faser) dar. Dieser Ansatz wird z.B. in [152] für ein 8-Bit paralleles Faserübertragungssystem benutzt. Die zweite Möglichkeit besteht darin, jedem Netzwerkknoten bzw. jeder Gruppe von Netzwerkknoten einen eigenen Kanal zuzuordnen, auf dem sie Daten senden können. Unter der Annahme, daß jeder Knoten einen wellenlängenverstellbaren Empfänger besitzt, kann so ein leistungsfähiges 'Broadcast-and-Select' Netzwerk einfach implementiert werden. Dazu müssen alle Knoten lediglich über einen Sternkoppler verbunden werden. Die letzte Möglichkeit stellt die Nutzung der Wellenlängen zur Adressierung dar. Dabei wird jedem Knoten bzw. jeder Knotengruppe eine Wellenlänge zugeordnet, auf der er Daten empfangen kann. Zum Senden besitzen alle Knoten wellenlängen-verstellbare Sender. Um eine Nachricht zu einem bestimmten Ziel zu schicken, muß ein Knoten nur seinen Sender auf die richtige Wellenlänge einstellen. Falls der Sender mehrere Wellenlängen gleichzeitig senden kann (weil er z.B. als ein Laserdiodenfeld realisiert ist), dann können auch Multicast- und Rundruf-Operationen einfach durchgeführt werden.

Das Problem obiger Verfahren besteht zum einen darin, daß die Anzahl der Wellenlängen in einem System durch die Leistung der verstellbaren Sender und Empfänger beschränkt ist. Sie liegt in der Regel zwischen 8 und 32. Für große Prozessorzahlen müssen sich mehrere Knoten jeweils eine Wellenlänge teilen. Dies führt zu Konflikten und macht geeignete Zugriffsprotokolle notwendig. Zum anderen muß sichergestellt werden, daß kein Knoten durch zuviele an ihn gleichzeitg gerichtete Nachrichten überfordert wird. Es ist also eine Flußkontrolle notwendig. Der Entwurf von Netzwerkarchitekturen, die die obigen Probleme lösen und trotzdem einen geringen Protokolloverhead und Netzwerkdurchmesser haben, ist das Hauptthema der Forschung im Bereich der WDM-Netzwerke.

LAMBDABUS: Eine Architektur für einen symmetrischen Multiprozessor auf der Basis mehrerer, optischer Busse wurde an dem Lawrence Livermore Laboratory untersucht [11]. Die LAMBDABUS genannte Architektur wurde als WDM-System entworfen. Es wurde also jedem Bus eine eigene Wellenlänge zugeordnet. Der Schwerpunkt der Forschung lag auf der Entwicklung der opto-elektronischen Komponenten. Um ihren Nutzen für die Rechnerarchitektur zu bewerten, wurden die Leistungsparameter des Netzwerks in einen Simulator für UMA Multiprozessoren eingegeben. Es wurde gezeigt, daß mit 8 bis 32 Wellenlängenkanälen a 1Gbit/s eine gute Leistung für bis zu 256 Prozessoren erreicht werden kann.

LIGHTNING: Das LIGHTNING-Netzwerk ist ein rekonfigurierbares hierarchisches WDM-Netzwerk, das zu Koppelung von Supperrechnern zu einem Cluster mit gemeinsamem Speicher konzipiert wurde [145, 146, 69]. Es basiert auf einer Baumarchitektur, in der die Prozessoren an den Blättern und elektronische Vermittlungseinheiten an den inneren Knoten angebracht sind. Es ordnet jedem Unterbaum eine oder mehrere Wellenlängen für die interne Kommunikation zu. Das Besondere an der Architektur ist, daß die Anzahl der einem Unterbaum zugeordneten Wellenlängen dynamisch in Abhängigkeit von den Bandbreitenanforderungen verändert werden kann.

HORN: Das Hierarchical Optical Ring Network ist eine Architektur für WDM basierte hierarchische Ringe. An den Ringen unterster Hierarchiestufe sind die Prozessoren angeschlossen. Wie die lokale Kommunikation innerhalb dieser Ringe, besitzt jeder Prozessor eine eigene Wellenlänge. Diese Wellenlängezuordnung ist in allen Ringen identisch, um die Anzahl der Wellenlängen klein zu halten. Für die Kommunikation zwischen den Ringen wird jedem Ring eine Wellenlänge zugeordnet, auf der alle Prozessoren dieses Ringes empfangen können. Für die Flußkontrolle wird ein auf Zeitfenstern basiertes Verfahren vorgeschlagen.

SLMH/OMCH: In [99] wird eine auf dem WDM-Verfahren basierte Erweiterung der Hypercube-Architektur vorgeschlagen. Dabei werden mehrere Hypercubes durch ein Gitternetzwerk (OMCH) oder durch ein Feld von Bussen verbunden. Ähnlich wie bei den Ringen der HORN werden in allen Hypercubes dieselben Wellenlängen für die lokale Kommunikation vergeben. Für die globale Kommunikation wird jedem Hypercube eine andere Wellenlänge zugeordnet.

Optical Time Domain Multiplexing (OTDM)

Das Optical Time Domain Multiplexing Verfahren (OTDM) nutzt die hohe Zeitbandbreite optischer Verbindungen, ohne daß die einzelnen Sender und Empfänger mit einer hohen Frequenz betrieben werden müssen. Die Idee besteht darin, mehrere niederfrequente aber schmale Signale zeitversetzt nebeneinander in einer Faser zu übertragen. Um in einer Faser eine Gesamtbandbreite von 500Gbit/s zu erreichen, können z.B. 500 Signale a 1Gbit/s benutzt werden. Die Signale sind gegeneinander um 1ps versetzt. Damit es zwischen den Kanälen kein Übersprechen gibt, dürfen die Signale dabei nur eine ps breit sein.

Eine genaue Erklärung der technischen Realisierung von OTDM-Netzen würde in diesem Rahmen zu weit führen (siehe z.B. [132]). Das Prinzip kann wie folgt zusammengefaßt werden: Um die Signale genau in den gewünschten Zeitfenstern zu positionieren, wird ein globaler Takt benutzt. Dieser Takt wird von einem Speziallaser erzeugt, der sehr kurze Impulse wohldefinierter zeitlicher Breite mit der Betriebsfrequenz der Sender (in unserem Beispiel 1Gbit/s) erzeugt. Jeder Knoten im Netzwerk empfängt das Taktsignal und verzögert ihn entsprechend dem ihm zugeordneten Zeitfenster (also im obigen Beispiel: 1ps für das erste Zeitfenster, 2 für das zweite, usw.). Beim Senden eines Signals schickt jeder Netzknoten eine AND-Verknüpfung des verzögerten Taktes und eines normalen breiten Signals. Um das Signal eines bestimmten Zeitfensters zu empfangen, wird zunächst das Taktsignal um den zum gewünschten Zeitfenster passenden Betrag verzögert. Das verzögerte Taktsignal wird dann benutzt, um durch eine optische AND-Verknüpfung das erwünschte Datensignal von den anderen zu trennen und zu lesen.

Das OTDM-Verfahren wird zur Zeit vor allem im Bereich der Telekommunikation untersucht. Es wird in vielem Arbeiten im Bereich optischer Verbindungen als Alternative zum bzw. Ergänzung zum WDM-Verfahren erwähnt, aber nicht weiter verfolgt. Dies liegt vor allem an dem hohen technischen Aufwand, der zu ihrer Implementierung notwendig ist. Eine Ausnahme bildet ein Gemeinschaftsprojekt der Firma SUN und der Princeton University [133]. Es schlägt vor, ein OTDM-System als Crossbar für bis zu 512 Prozessoren zu nutzen. Jeder am Crossbar angeschlossene Einheit kann in genau einem Zeitfenster Daten empfangen und in allen senden. Damit kann jede Einheit durch die Wahl des Zeitfenster, in dem sie sendet, jede andere Einheit in einem Schritt erreichen. Um zu verhindern, daß mehrere Einheiten gleichzeitig an das gleiche Ziel senden, wurde ein spezielles Arbitrierungsprotokoll auf der Basis einer optischen 'wired-or' Operation entwickelt. Der Vorteil des OTDM-Verfahrens besteht hier darin, daß im Gegensatz zum konventionellen elektronischen Verfahren die Anzahl der Schaltelemente nicht quadratisch mit der Anzahl der Prozessoren ansteigt. Dies liegt daran, daß die Wahl des Zeitfensters (und damit die Adressierung) durch ein verstellbares Verzögerungsglied realisiert wird, dessen Komplexität nur leicht mit der Anzahl der Zeitfenster steigt.

Die Bedeutung des OTDM-Crossbar Systems für die vorliegende Arbeit besteht darin, daß nichtblockierende Crossbars zur Zeit als Alternative zu Bussen bei der Implementierung des gemeinsamen Speichers benutzt werden.

Damit stellt das System ein Konkurenzkonzept zu dem in der Arbeit beschriebenen Architekturen dar. Allerdings eignen sich solche Architekturen nicht für Snoopy-Protokolle und damit eher für NUMA-Rechner.

Netzwerke mit hoher Ortsbandbreite

Die meisten Forschungsprojekte zur Nutzung der hohen Ortsbandbreite beschäftigen sich mit den technischen Problemen beim Aufbau massiv paralleler Freiraumverbindungen. Zu den wichtigsten Ansätzen im Bereich der Netzwerkarchitektur und -protokolle gehören das POPS und das OTIS Projekt.

POPS: Das Paralell Optical Partioned Star Network [147] ist ein 'Broadcast-and-Select' Netzwerk, das durch ein geeignetes Zugriffsprotokoll verschiedene Netzwerktopologien emuliert. Da die Topologie durch das Protkoll gegeben ist, kann das Netzwerk leicht rekonfiguriert werden.

OTIS: Das Optical Transpose Interconnect System ist eine massiv parallele Freiraumnetzwerkarchitektur [42]. Die Netzwerktopologie wurde bereits in Abschnitt 4.2.3 als Beispiel für Freiraumnetze besprochen. Auf der Basis dieser Architektur wurde ein hierarchisches Architekturkonzept entwickelt. Auf unterster Ebene befinden sich Gruppen von elektrisch verbundenen Prozessoren. Die Gruppen sind auf globaler Ebene über eine OTIS-Freiraumverbindung gekoppelt.

4.8.3 SP basierte Netzwerke

Die Entwicklung von SP-Bausteinen mit einer hohen Anzahl optischer Kanäle und leistungsfähiger CMOS VLSI-Logik hat eine Vielzahl neuer opto-elektronischer Netzwerkarchitekturen motiviert. Diese Architekturen können in zwei Klassen eingeteilt werden: auf einem SP-Chip integrierte Netzwerke und Netzwerke, die SP-Chips als Schnittstellen nutzen. Im Nachfolgenden werden die wichtigsten Arbeiten in beiden Bereich beschrieben. Sie unterscheiden sich alle von der vorliegenden Arbeit darin, daß sie keinen direkte Bezug zur Speicherarchitektur und der Frage der Skalierbarkeit symmetrischer Multiprozessoren herstellen.

SP-Netzwerke

Mehrere Projekte beschäftigen sich zur Zeit mit der Realisierung eines elektronischen Schaltnetzwerks auf SP-Bausteinen. Sie nutzen die optischen Ein/Ausgabekanäle, um möglichst viele Knoten mit möglichst großer Bandbreite und geringer Latenz an das Netzwerk anzuschließen.

AMOEBA: Bei Bell-Labs wurde ein SP-basierter Crossbar-Schalter für parallele Fasernetzwerke entwickelt und implementiert [85]. Es handelt sich dabei um ein 8 Bit breites, 16x16 Crossbar, das auf einem CMOS ($0.8\mu m$ Technologie) mit einem Feld von 32x64 MQW-Dioden realisiert wurde.

Inteligent Optical Network: Für die Verbindung von Arbeitsplatzrechnern wurde an der McGill Universität in Toronto an einem SP-basierten Schalter für parallele Fasernetzwerke gearbeitet. Der Schalter implementiert auf dem Chip ein 'Broadcast-and-Select' Netzwerk mit Paketvermittlung. [167]

WARP: An der University of Southern California wurde ein opto-elektronischer Router-Chip entwickelt. Der WARP [175, 174] genannte Chip realisiert ein Paketvermittlungsverfahren auf der Basis eines modifizierten Wormhole-Algorithmus.

SPOEC An der Heriot-Watt Universität in England wird zur Zeit ein opto-elektronischer Freiraum-Crossbar-Schalter entwickelt [163]. Er soll mit einem SP-Baustein als Schalter einen Durchsatz von 1TBit/s erreichen.

Netzwerke mit SP-Schnittstellen

Die Nutzung von SP-Bausteinen als Netzwerkschnittstellen wird zur Zeit in zwei Projekten untersucht.

Hyperplane: Ein Konsortium kanadischer Universitäten arbeitet an einer optischen Freiraum-Backplane (Hyperplane), die eine Bandbreite im TBit/s Bereich für Board-to-Board Verbindungen bereitstellen soll ([181, 168]). Um eine hohe Bandbreite zu bewerkstelligen, werden als Netzwerkschnittstellen OEVLSI-Bausteine verwendet. Der Schwerpunkt des Hyperplane Projektes liegt bei der Entwicklung der Technologie für die Freiraumverbindungen.

STAR: [115] Das STAR System ist eine Erweiterung der kommerziellen parallelen Fasersysteme. Dabei werden als Sender und Empfänger SP-Bausteine verwendet. Das System soll mit 32 Fasern eine Bandbreite von mehr als 100Gbit/s erreichen.

4.8.4 Spezielle Architekturen

Die Optik gilt innerhalb der Computerarchitektur-Gemeinde immer noch als eine exotische Technologie. Dementsprechend wenige Arbeiten gibt es, die sich mit dem Entwurf auf diese Technologie basierender Architekturen beschäftigen. Neben der unten beschriebenen DPxx , GLORI und SMOEC Architekturen gibt es lediglich eine Reihe von Überlegungen zur Auswirkung optischer Netzwerke auf Cache-Kohärenz Protokolle. Hinzu kommen einige theoretische Arbeiten, die sich mit der Realisierung bestimmter theoretischer Modelle auf optischen Netzwerke beschäftigen. Projekte, die sich wie die vorliegende Arbeit mit großen symmetrischen Multiprozessoren beschäftigen, sind dem Autor nicht bekannt.

Die DPxx-Rechner

An der TU Delft wird seit Anfang der 80er Jahre an parallelen Rechnerarchitekturen mit optischen Netzwerken gearbeitet. Dabei wurden drei Generationen von Rechnern entworfen und teilweise implementiert: DP81, DP86 und DP91 [30, 34, 50, 49]. Es handelt sich dabei um massiv parallele MIMD Rechner mit optischem 'Broadcast-and-Select' und vollverbundene Netzwerken. Die Datenübertragung finde mit Hilfe von Faserbündeln statt. Für den Rundruf wird eine spezielle Spiegelanordnung, das sog. Kaleidoskop, verwendet. Um das Problem der Netzwerkschnittstellen zu lösen, wurde ein opto-elektronischer VLSI-Speicherbaustein, das sog. POWERRAM entwickelt.

Von allen dem Autor bekannten Projekten haben die DPxx Architekturen die meisten Gemeinsamkeiten mit der vorliegenden Arbeit. Das DPxx Projekt setzt auf ähnliche technologische Ansätzen und gehört zu den wenigen Projekten, die sich gleichzeitig intensiv mit Fragen der Technologie und der Rechnerarchitektur beschäftigen.

Das GLORI-Konzept

An der Stanford Universität wurde das GLORI genannte Konzept für einen optisch verbundenen MIMD Rechner vorgeschlagen und untersucht [140, 142]. Es schlägt die Nutzung optischer Freiraumverbindungen für lokale, busgekoppelte Prozessorcluster vor, die das UMA-Speichermodell implementieren. Die Cluster sind über ein optisches Hypercube-Fasernetzwerk zu einem CC-NUMA-Rechner gekoppelt. Das GLORI-Projekt ähnelt der vorliegenden Arbeit insofern, als es sich intensiv sowohl mit der Rechnerarchitektur als auch mit der Technologie beschäftigt. Allerdings sind sowohl die untersuchte Speicherarchitektur als auch die verwendete Technologie deutlich anderes.

SMOEC

Der Shared Memory Opto-Elelctronic Computer [185] ist eine MIMD Architektur mit gemeinsamem Speicher, die auf einem optischen perfect-shuffle Netzwerk [24] basiert. Die Architektur wurde als Gedankenexperiment zur Demonstration der Anwendungsmöglichkeiten des optischen Shuffle Netzwerks entwickelt. Sie wurde weder durch Implementierung noch durch Simulation bewertet. Auch die Fragen der Cache-Kohärenz wurden nur oberflächlich behandelt. Das SMOEC-Konzept ist daher nur sehr eingeschränkt mit der vorliegenden Arbeit vergleichbar.

Cache-Kohärenz-Protokolle für optische Architekturen

Mehrere Arbeiten haben Varianten bekannter Cache-Kohärenz-Protokolle untersucht, die sich besonderes gut für unterschiedliche Arten opto-elektronischer Verbindungen eignen. Dazu gehören die in [29, 28] beschriebenen Protokolle für einen optischen Bus und Ring so wie das SPEED-Protokoll [141] [63]. Auch spezielle Synchronisationsverfahren wurden vorgeschlagen [151, 162]. Obige Untersuchungen unterscheiden sich von der vorliegenden Arbeit darin, daß sie einen bestimmten engen Aspekt der Architektur auf einer hohen Abstraktionsebene betrachten. Sie brauchen daher hier nicht genauer erläutert zu werden.

4.8.5 Fazit

Die vorliegende Arbeit beschäftigt sich gezielt mit der Nutzung der Vorteile von SP-basierten Netzwerken zur Erweiterung des Anwendungsbereiches des SMP-Speichermodells. Sie zeichnet sich dadurch aus, daß sie sich gleichermaßen mit Fragen der Technologie wie mit der Rechnerarchitektur beschäftigt. Trotz einiger Überschneidungen mit unterschiedlichen Bereichen setzt sie sich damit in ihrer Gesamtheit klar von allen anderen dem Autor bekannten Projekten ab. Die meisten Gemeinsamkeiten gibt es zu dem Dpxx Projekt, das sich allerdings nicht explizit mit dem SMP-Speichermodell beschäftigt.

Teil II

Beiträge

Kapitel 5

Ansätze und Vorgehensweise

Das vorliegende Kapitel beschreibt die Ansätze und Vorgehensweisen, die dem Entwurf und der Bewertung aller betrachteten Architekturen zugrunde liegen. Dazu werden zunächst das Cache-Kohärenz-Protokoll und das Synchronisationsverfahren spezifiziert, die in allen Architekturen benutzt werden. Danach wird die für die Bewertung der Architekturen benutzte Simulationsumgebung beschreiben. Abschließend wird erläutert, wie die Ergebnisse der Simulation durch theoretische Modellierung verifiziert werden.

5.1 Grundidee

Die Betrachtung in den Kapiteln 2, 3 und 4 hat gezeigt daß,

1. die symmetrischen Multiprozessoren mit ihrem CC-UMA-GS Speichermodell dank ihrer guten Programmierbarkeit und hohen Effizienz für ein breites Spektrum von Anwendungen die attraktivste Parallelrechner-Klasse sind (Kapitel 2),

2. die Realisierung von symmetrischen Multiprozessoren für hohe Prozessorzahlen und großer räumlicher Ausdehnung durch die Leistungsfähigkeit von elektronischen Rundruf-Leitungen und Netzwerkschnittstellen beschränkt ist (Kapitel 3),

3. die Opto-Elektronik die Realisierung von Verbindungen mit extrem hoher Bandbreite, großem Fanout, geringen Latenz und direkter Anbindung an VLSI-Schnittstellen erlaubt (Kapitel 4).

Motiviert durch die obigen Erkenntnisse wird in der Arbeit gezeigt, wie optische Rundrufleitungen und optoelektronische Netzwerkschnittstlellen zu Erhöhung der Skalierbarkeit von SMPs beitragen können. Die Grundidee des in der Arbeit verfolgten Ansatzes wurde bereits in Kapitel 1 beschrieben. Die wichtigsten Punkte können im Licht der Erkenntnisse aus Kapiteln 2, 3 und 4 wie folgt zusammengefaßt werden:

1. Die vorgeschlagenen Architekturen basieren auf einem gemeinsamen Speicher auf Hardwarebenene.

2. Für die Erhaltung der Cache-Kohärenz wird ein Snoopy, durchschreibender Protokoll benutzt.

3. Es werden drei Netzwerktopologien vorgeschlagen:

   (a) ein paketvermittelnder Bus (PHOTOBUS-Architektur)

   (b) ein gemischtes Bus/Crossbar Netzwerk (PHOTOBAR-Architektur)

   (c) ein vom Autor für die Lösung des 'cache-update-pressure' Problems konzipiertes Mehrbussystem (PHOTON-Architektur)

4. Das elektronische Verbindungsnetzwerk wird auf einen SP-Chip 'geschrumpft', der über zwei Arten von optischen Kanäle mit den Prozessoren und den Speicherbänken verbunden ist.

(a) Optische Punkt-zu-Punkt Kanäle (P- und M-Kanäle genannt), die direkt an die optischen Ein/Ausgabefenster des SP-Chips angeschlossen sind. Alle Nachrichten, die ein Prozessor oder eine Speicherbank an das System verschicken wollen, gehen zuerst über diese Kanäle an den Chip. Dieser speichert sie zwischen, führt im Fall von Konflikten eine Arbitrierung durch und leitet sie dann weiter. Bei der PHOTOBAR-Architektur sind diese Leitungen biderektional um damit nicht alle Nachrichten über den Rundruf-Kanal verschickt werden müssen

(b) Eine unidirektionale optische Rundruf-Leitung (B-Kanal genannt) , die die Verlängerung des auf dem SP-Chip befindlichen Busses ist. Über diese Leitung kann der Chip Daten an alle Prozessoren und Speicherbänke durch einen Rundruf weiterleiten.

Da der Rundruf an die Prozessoren und die Speicherbänke optisch stattfindet, kann unabhängig von der Anzahl der Prozessoren und der Ausdehnung des Systems auch hier eine hohe Bandbreite und geringe Latenz garantiert werden. Auch die P- und M-Kanäle können dank der Vorteile optischer Verbindungen und SP-Schnittstellen eine geringe Latenz, hohe Bandbreite und große Ausdehnung haben. Durch die geringe Leitungslänge und günstige elektrische Leitungseigenschaften kann innerhalb des Chips ebenfalls eine hohe Bandbreite bei niedriger Latenz und hohem Fanout erreicht werden. Gleichzeitig hat ein auf einem SP-Chip basiertes System gegenüber einem rein optischen Bussystem zwei Vorteile:

(a) Für einen Rundruf wird ein Transmitter mit hoher Bandbreite und einer zur Prozessorzahl proportionalen Ausgangsleistung benötigt. Für hohe Prozessorzahlen sind solche Transmitter aufwendig und teuer. Da in einem rein optischen Bussystem jeder Prozessor und jede Speicherbank auf dem Rundrufkanal senden können muß, steigt die Anzahl solche Sender im System linear mit der Anzahl der Prozessoren. In dem hier vorgeschlagenen System wird nur an dem SP-Chip mit dem 'geschrumpften' Netzwerk ein solcher aufwendiger Transmitter benötigt.

(b) Auf dem VLSI-Chip befindet sich zu jedem Zeitpunkt die Information über alle im System vorhandenen Anfragen. Dies ermöglicht eine effiziente und intelligente Arbitrierung. Außerdem können auf dem Chip verschiedene Optimierungen (z.B. eine effiziente Synchronisation) durchgeführt werden.

5. Um das Schnittstellenproblem zu lösen, werden bei hohen Prozessorzahlen die Prozessoren mit SP-Bausteinen als Netzwerkschnittstellen ausgestattet.

Eine Schematische Darstellung der Architekturen ist in Kapitel 1 in Abbildung 1.3.2 zu finden. Sie werden in den Kapiteln 7 8 und 9 genau beschrieben.

5.2 Das Cache-Kohärenz-Protokoll

Die in der Arbeit untersuchten Grundkonzepte können für eine Vielzahl von Snoopy-Protokollen verwendet werden. Für eine genaue Untersuchung konkreter Architekturen muß man sich allerdings für ein bestimmtes Protokoll entscheiden. In der Arbeit wurde dazu das Berkeley-Protokoll ausgewählt.

Das Berkeley-Protokoll wurde in Kapitel 3 grob beschrieben. Die Grundidee besteht darin, das Schreibrecht auf eine Cache-Zeile auf einem Prozessor zu beschränken. Dieser sog. Besitzer-Prozessor ist dafür verantwortlich, den aktuellen Inhalt der Zeile anderen Prozessoren zur Verfügung zu stellen. Da bei der Beschreibung der Architekturen in den nachfolgenden Kapiteln ein genaues Verständnis des Protokolls vorausgesetzt wird, wird es im vorliegenden Abschnitt nochmals detailliert erläutert. Dazu werden zunächst die Zustände, Operationen und Zustandsübergänge genau spezifiziert. Danach werden die Anforderungen zusammengefaßt, die das Protokoll an die Prozessor-Speicher-Koppelung der untersuchten Architekturen stellt.

5.2.1 Die Zustände und Operationen des Berkeley-Protokolls

Die Definition eines Cache-Kohärenz-Protokolls beinhaltet drei Dinge: eine Menge von Cache-Zeilen-Zuständen, eine Menge von zulässigen Übergängen zwischen diesen Zuständen und eine Menge von Speicher-Operationen. Sie spezifiziert die Zustandsübergänge einer Cache-Zeile und die durchzuführende Speicheroperation in Abhängigkeit von drei Faktoren:

1. dem Zustand der Zeile

2. der durch den eigenen Prozessor gestellten Cache-Anfrage und

3. der über das Prozessor-Speicher Netzwerk empfangenen Speicher-Operationen, die andere Prozessoren auf einer im eigenen Cache enthaltenen Zeile durchführen.

Für die Beschreibung der Zustandsübergänge werden in der Regel zwei Übergangsdiagramme benutzt. Das eine zeigt, wie ein Prozessor durch seine Anfragen die Zustände der Zeilen in seinem Cache beeinflußt. Das andere gibt an, wie sich die Zustand einer Zeile verändert, wenn ein anderer Prozessor eine Operation auf einem Datum durchführt, das in dieser Cache-Zeile enthalten ist. Die Übergangsdiagramme abstrahieren von den Details der Speicheroperationen. Sie geben lediglich an, daß eine bestimmte Anfrage, bzw. Operation zum Übergang aus dem einen in den anderen Zustand zur folge hat. Was bei diesem Übergang zu passieren hat (als daß z.B. Daten in den Cache eingelesen werden müssen) wird nicht durch das Übergangsdiagramm sondern durch die Beschreibung der Speicheroperationen angegeben.

Im Nachfolgenden beschreiben wir zunächst die möglichen Cache-Zeilen Zustände des Berkeley-Protokolls und dessen Bedeutung. Danach wird beschrieben, welche Zustandsübergänge und Speicheroperationen durchgeführt werden müssen, wenn ein Prozessor eine Anfrage an seinen Cache stellt. Abschließend werden die Speicheroprationen erläutert. Dabei wird zum einen dargelegt, welche Funktion sie erfüllen. Zum anderen wird beschrieben welche Auswirkungen sie auf die Caches anderer Prozessoren haben. Zur Verdeutlichung sind in Abbildung 5.1 die beiden Übergangsdiagramme dargestellt.

Zustände

Im Berkeley-Protokoll kann eine Cache-Zeile vier Zustände haben: Invalid, ReadOnly, PrivateModified und SharedModified. Die einzelnen Zustände werden in der Literatur oft unterschiedlich bezeichnet. In der Arbeit werden die Bezeichnungen übernommen, die auch bei der Implementierung und Beschreibung des verwendeten Simulators benutzt werden. Ihre Bedeutung kann wie folgt zusammengefaßt werden:

Invalid: Dieser Zustand bedeutet, daß die Zeile keine gültigen Daten beinhaltet.

ReadOnly: Die Zeile im ReadOnly-Zustand beinhaltet gültige Daten, auf die allerdings nur lesend zugegriffen werden kann. Der Zustand gibt keine Auskunft darüber, ob sich die Zeile nur in einem oder in mehreren Caches befinden.

PrivateModified: Im PrivateModified Zustand beinhaltet die Zeile gültige Daten, auf die sowohl lesend als auch schreibend zugegriffen werden kann. Der Zustand bedeutet, daß sich die Daten nur in diesem einen Cache befinden.

SharedModified: In diesem Zustand befinden sich in der Zeile gültige Daten, auf die sowohl lesend als auch schreibend zugegriffen werden kann. Er signalisiert, daß Kopien dieser Daten auch in anderen Caches existieren. Allerdings sind alle diese Kopien im ReadOnly Zustand, da immer nur ein Prozessor das Schreibrecht auf eine Cache-Zeile besitzen kann.

Auswirkung der Prozessoranfragen

Je nachdem, ob der Prozessor auf ein Datum lesend oder schreibend zugreifen möchte, und ob das Datum im Cache vorhanden ist, müssen vier Arten von Operationen unterschieden werden: Read-Hit, Read-Miss, Write-Hit und Write-Miss. Die Auswirkung dieser Operationen auf die Cache-Zeilen Zustände sowie die durch sie verursachten Speicheranfragen können wie folgt zusammengefaßt werden:

Read-Hit: Von einem Read-Hit spricht man, wenn die Daten, die der Prozessor lesen möchte im Cache gefunden wurden. Die Daten werden an den Prozessor übergeben und es findet keine Zustandsänderung der Cache-Zeile statt.

Read-Miss: Bei einem Read-Miss ist das Datum, das gelesen werden soll, nicht im Cache vorhanden. Falls sich in der zugehörigen Cache-Zeile Daten im SharedModified oder PrivateModified Zustand befinden, so werden diese zunächst zurückgeschrieben (siehe WriteBack-Speicheroperation). Danach werden die benötigten Daten vom Speichersystem mit einer FetchRead-Speicheroperation eingelesen, in die Cache-Zeile hineingeschrieben und an den Prozessor weitergeleitet. Dabei wird der Zustand der neu gelesenen Zeile auf ReadOnly gesetzt.

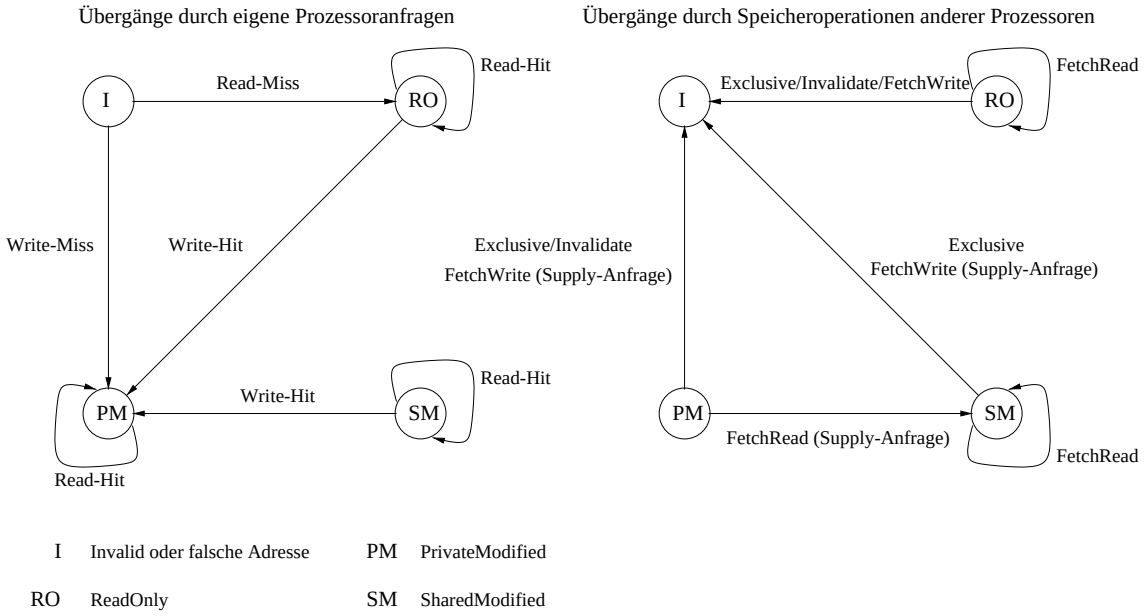


Abbildung 5.1: Die Zustandsübergänge des Berkeley Cache-Kohärenz Protokolls. Links sind die durch eigene Prozessoranfragen verursachten übergänge dargestellt. Die rechten Übergänge werden durch Anfragen anderer Prozessoren hervorgerufen.

Write-Hit: Bei einem Write-Hit soll ein Datum geschrieben werden, daß sich bereits im Cache befindet. Was dabei passiert, hängt von dem Zustand der zugehörigen Cache-Zeile ab. Wenn sich die Zeile im PrivateModified Zustand befindet, dann wird der Schreibzugriff durchgeführt und der Zustand der Zeile bleibt unverändert. Im Fall des SharedModified Zustands wird eine Invalidate-Nachricht an alle anderen Caches geschickt und die Zeile geht in den PrivateModified zustand über. Bei einem Schreibzugriff auf eine Zeile im ReadOnly Zustand wird der Zustand der Zeile in PrivateModified geändert werden. Zuvor muß die Änderung des Zustandes durch eine Exclusiv-Operation dem System mitgeteilt werden.

Write-Miss: Bei einem Write-Miss ist das zu schreibende Datum nicht im Cache vorhanden. Die entsprechende Zeile wird daher mittels einer FetchWrite-Operation eingelesen und in den PrivateModified Zustand versetzt. Der Schreibzugriff wird dann auf der neu eingelesenen Zeile im Cache durchgeführt.

Die durch Prozessor-Anfragen verursachten Zustandsübergänge sind in Abbildung 5.1 verdeutlicht.

Speicheroperationen

Als Speicheroperationen werden alle Operationen betrachtet, bei denen der Prozessor Anforderungen oder Daten an die Speicherbänke und die Caches anderer Prozessoren schickt. Die Realisierung des Berkeley-Protokolls erfordert die folgenden fünf Arten solcher Operationen

FetchRead: Eine FetchRead Operation wird nach einem Read-Miss durchgeführt. Sie liest die benötigte Zeile in den Cache ein und setzt ihren Zustand auf ReadOnly.

FetchWrite: Die FetchWrite Operation holt eine Cache-Zeile für einen Schreibzugriff. Die Zeile wird also nach dem Einlesen in den PrivateModified Zustand versetzt.

Exclusive: Die Exclusiv-Operation wird benötigt, wenn ein Prozessor in eine Zeile schreiben möchte, die sich in seinem Cache im ReadOnly Zustand befindet. Sie teilt allen Prozessoren mit, daß sie die Zeile in ihrem Cache in den Invalid-Zustand versetzten sollen, und führt sie im eigenen Cache in den PrivateModified Zustand über.

Invalidate: Eine Invalidate-Operation findet immer dann statt, wenn ein Prozessor einen Schreibzugriff auf eine Zeile im SharedModified Zustand durchführt. Durch sie werden alle anderen Prozessoren veranlaßt, die Zeile in ihren Caches in den Invalid-Zustand zu versetzen.

WriteBack: Falls ein Prozessor eine Zeile im PrivateModified oder SharedModified Zustand aus seinem Cache entfernen möchte, dann muß er vorher die Inhalte der Zeile in den Speicher zurückschreiben. Dies erfolgt durch eine WriteBack-Operation.

Supply: Die Supply-Operation wird benötigt, wenn ein Prozessor eine FetchRead oder FetchWrite Operation auf einer Zeile ausführt, die sich im Cache eines anderen Prozessors im PrivateModified oder im SharedModified Zustand befindet. In diesem Fall sind die im Speicher enthaltenen Daten ungültig. Die Zeile muß daher von dem Prozessor zur Verfügung gestellt werden, der das Schreibrecht auf sie besitzt. Dazu führt dieser Prozessor eine Supply-Operation durch. Je nachdem, ob die Supply-Operation als Antwort auf eine FetchRead oder eine FetchWrite Operation durchgeführt wurde, wird die Zeile entweder in den SharedModified oder in den Invalid-Zustand versetzt.

5.2.2 Anforderungen an die Prozessor-Speicher Koppelung

Die Definition des Berkeley-Protokolls basiert auf der Annahme, daß die Prozessoren und die Speicherbänke an einen gemeinsamen, leitungsvermittelnden Bus angeschlossen sind. Sie geht daher davon aus, daß

1. jede Speicheroperation für alle Prozessoren und alle Speicherbänke sichtbar ist und
2. ein Prozessor, der eine Speicheroperation ausführt, bis zur Beendigung der Operation den Bus blockiert, so daß Speicheroperationen nicht parallel ausgeführt werden können.

Obige Annahmen treffen auf die in der Arbeit entwickelten und untersuchten Architekturen nicht zu. Bei der PHOTOBAR-Architektur werden Nachrichten über die bidirektionalen P- und M-Kanäle an einzelne Prozessoren und Speicherbänke verschickt. Es ist also nicht jede Nachricht für alle sichtbar. Außerdem funktionieren die Netzwerkbausteine nach dem Paketvermittlungsprinzip, so daß mehrere Anfragen gleichzeitig bearbeitet werden können. Um trotzdem eine korrekte Ausführung des Berkeley-Protokolls zu garantieren, müssen die Architekturen die folgenden Anforderungen an den Datenfluß, die Flußkontrolle und das Timing erfüllen:

Anforderungen an den Datenfluß

Der Datenfluß, der für die einzelnen Speicheroperationen benötigt wird, ist in Abbildung 5.2 verdeutlicht. Die Abbildung zeigt für jede Speicheroperation den Datenfluß zwischen dem Prozessor, der die Operation durchführt (P) und den Caches der anderen Prozessoren so wie dem Hauptspeicher. Dabei sind die Caches unterschiedlich markiert, je nach dem ob, und in welchem Zustand sie das betroffene Datum beinhalten. Dabei wird die Größe der Nachrichten durch die Dicke der Linien angedeutet. Die dünnen Linien symbolisieren Nachrichten, die nur aus einer Adresse und einem Befehlswort bestehen. Die dicken Linien stellen Nachrichten dar, die einen ganze Cache-Zeile beinhalten müssen. Zusätzlich wird zwischen zwei Arten von Nachrichten unterschieden: absolut notwendigen Nachrichten und eventuell verzichtbaren Nachrichten. In die letzte Klasse fallen Nachrichten, die nur deswegen verschickt werden müssen, weil man nicht weiß, welcher Cache welches Datum in welchem Zustand beinhaltet. Sie können im Rahmen einer Optimierung durch Zwischenspeichern geeigneter Informationen in den Netzwerkbausteinen wegoptimiert werden.

Anforderungen an die Flußkontrolle

Die Möglichkeit paralleler Ausführung mehrerer Speicheroperationen führt dazu, daß mehrere Anfragen gleichzeitig an die gleiche Speicherbank oder den gleichen Prozessor gerichtet werden können. Dies kann leicht zur Überlastung führen, wenn an einer Komponenten mehr Nachrichten ankommen, als sie verarbeiten und zwischenspeichern kann. Daher muß eine Flußkontrolle implementiert werden, die sicherstellt, daß die Prozessoren und die Speicherbänke nie mehr Nachrichten erhalten, als sie in der Lage sind zu verarbeiten. Nachrichten dürfen also nur dann verschickt werden, wenn die betroffenen Knoten bereit sind, sie zu verarbeiten.

Anforderungen an das Timing

Das Timing von Operationen ist beim Berkeley-Protokoll nur in einem Fall von Bedeutung: bei den Supply-Operationen. Wenn sich eine durch Fetch angeforderte Zeile in einem Cache im PrivateModified oder SharedModified Zustand befindet, dann muß der Besitzer die benötigten Daten übertragen, bevor der Hauptspeicher auf

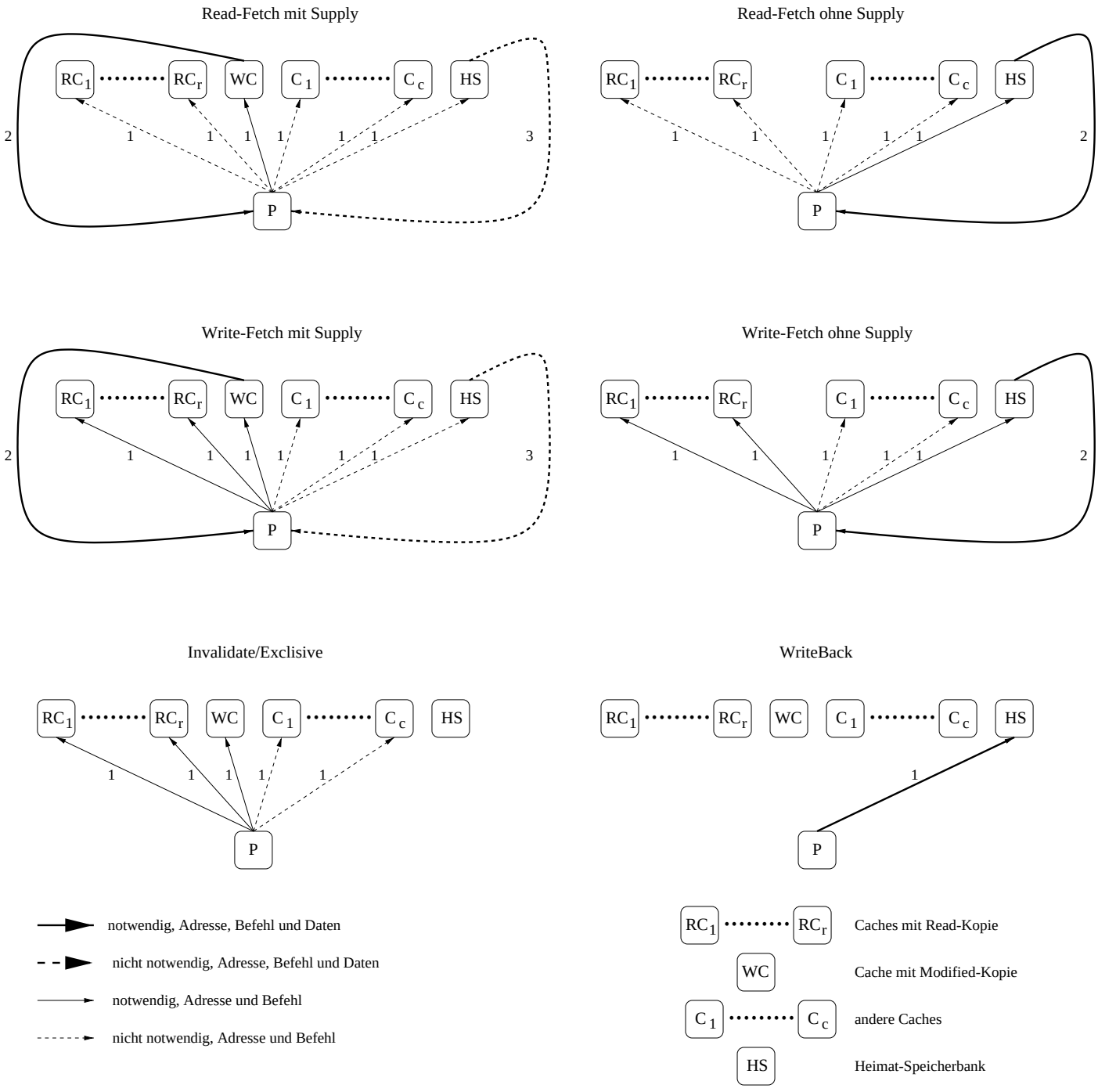


Abbildung 5.2: Der für die Realisierung des Berkeley Cache-Kohärenz Prptokolls notwendige Datenfluß zwischen dem Prozessor, der eine Operation durchführt (P) und den Caches der anderen Prozessoren so wie dem Hauptspeicher. Die Caches unterschiedlich markiert, je nach dem ob, und in welchem Zustand sie das betroffene Datum beinhalten.

die Anfrage antworten kann. In einem 'circuit-switched' Bus basierten System wird diese Forderung weitgehend automatisch erfüllt. Dies liegt daran, daß

1. die Anforderung den Hauptspeicher und die Prozessoren zur gleichen Zeit erreicht,
2. der Cache die Anfrage schneller bearbeiten kann als der Hauptspeicher,
3. der Cache die Antwort sofort übertragen kann, da der Bus die ganze Zeit für die laufende Operation reserviert ist.

In einem paketorientierten System treffen Punkt 1 und 3 nicht unbedingt zu. Die zeitgerechte Durchführung der Supply-Operationen muß also durch einen gesonderten Mechanismus sichergestellt werden.

5.3 Behandlung der Synchronisation

Für die Synchronisation gilt genauso wie für das Cache-Kohärenz-Protokoll, daß die Grundüberlegungen der entworfenen Architektur eine Vielzahl von Verfahren zulassen, für eine detaillierte Architekturstudie aber ein konkretes Verfahren ausgewählt werden muß. Wie in Kapitel 2 erläutert, kann die Synchronisation entweder mit Hilfe eines speziellen Synchronisationsnetzwerkes oder über den Speicher realisiert werden. Damit eine Speicherbasierte Realisierung möglich ist, muß das Speichersystem entweder grundsätzlich sequentiell konsistent sein, oder es muß für die Synchronisation spezielle Operationen zur Verfügung stellen. Die vorliegende Arbeit geht davon aus, daß das Speichersystem eigene Operationen für die Realisierung von Sperren zur Verfügung stellt. Für diese Entscheidung gibt es zwei Gründe. Der wichtigste Grund ist die Tatsache, daß sich solche Operationen mit Hilfe der betrachteten Architekturen besonders effizient realisieren lassen. Dies wird in den nachfolgenden Kapiteln eingehend diskutiert. Zum anderen wird dieser Ansatz auch in dem, für die Bewertung der Architekturen verwendeten Simulationssystem verfolgt.

Für die Implementierung von Sperren werden zwei neue Speicheroperationen zur Verfügung gestellt: ein `LOCK`-Befehl und ein `UNLOCK`-Befehl. Diese Befehle können direkt in den Maschinencode eines Programms eingefügt werden. Sie werden vom Prozessor an den Cache-Kontroller und von dort an das Speichersystem weitergegeben. Sie bekommen beide als Argument die Adresse einer Sperrvariable und haben die folgende Funktion:

LOCK-Befehl: Eine `LOCK`-Operation prüft zunächst den Wert, der an der angegeben Adresse gespeichert ist. Falls dieser 0 ist, wird er auf 1 gesetzt und die Operation liefert eine Erfolgsmeldung zurück. Damit wird die Sperre von dem Urheber der Operation in Besitz genommen. Anderenfalls wird ein Fehlschlag zurückgemeldet.

UNLOCK-Befehl: Die UNLOCK-Operation setzt die Variable an der angegeben Adresse auf 0 zurück. Sie gibt die Sperre also frei.

Die obigen Operationen werden in der Speicherbank ausgeführt, ohne daß irgendwelche Daten im Cache zwischengespeichert werden.

5.4 Die Simulationsumgebung

Mit der Simulation werden in der Arbeit zwei Ziele verfolgt: In erster Linie soll gezeigt werden, wie gut die entworfenen Architekturen skalierbar sind und wie ihre Effizienz von den Technologieparametern abhängt. Darüber hinaus soll die Simulation die Korrektheit der Systeme untermauern. Um das erste Ziel zu erreichen, muß die Simulationsumgebung drei Anforderungen erfüllen:

1. Sie muß auf einem repräsentativen Satz von Benchmark-Programmen basieren.
2. Bei der Simulation der parallelen Ausführung der Benchmarks muß das Speicherverhalten realitätsgetreu nachgebildet werden. Dabei muß insbesondere die Verteilung der Zugriffe auf den Cache und den Hauptspeicher korrekt simuliert werden.
3. Es müssen die Antwortzeiten der opto-elektronischen Verbindungsstruktur und des Speichersystems auf die Speicheranfragen simuliert werden.

Für die Simulation der Korrektheit ist es zusätzlich notwendig, den Weg der Datenpakete durch das System genau nachzubilden. Das heißt, daß die Architektur der Simulationssoftware die untersuchte Architektur des Netzwerks nachahmen muß.

Um die obigen Anforderungen zu erfüllen, wurden in der Arbeit ausgewählte SPLASH-2 Benchmarks [187] und eine entsprechend erweiterte Version des LIMES [176] Multiprozessor-Simulators benutzt. Für die Wahl der Benchmarks waren drei Faktoren entscheidend: die weite Akzeptanz der SPLASH-2 als Maß für die Leistungsfähigkeit von Multiprozessoren, die Verfügbarkeit einer genauen Charakterisierung des Laufzeitverhaltens der einzelnen Programme und die Verfügbarkeit von optimierten Implementierungen, die mit minimalen Anpassungen in der verwendeten Simulationsumgebung ausführbar waren. Bei der Auswahl des Simulators ging es vor allem um die Effizienz und eine einfache Erweiterbarkeit.

Im Nachfolgenden werden zunächst die verwendeten Benchmarks genauer beschrieben. Danach werden der Aufbau des LIMES-Simulators und die im Rahmen der Arbeit durchgeführten Erweiterungen erläutert.

5.4.1 Die Benchmarks

Die SPLASH-2 Benchmarksammlung wurde an der Stanford-Universität im Rahmen des FLASH-Projektes (siehe 3.6.2) zusammengestellt. Sie beinhaltet vier Kerns und acht größere Anwendungsprogramme. Die Programme wurden so ausgewählt, das sie für ein möglichst breites Spektrum von Problemen repräsentativ sind. Das besondere an der SPLASH-2 Sammlung ist, daß nicht nur die Ausführungszeit und die Beschleunigung der Programme gemessen wurden, sondern das gesamte Speicherverhalten genau untersucht wurde.

Im Nachfolgenden werden zunächst die verwendeten Programme und ihre Implementierung beschrieben. Danach werden das Speicherverhalten der Programme sowie ihre Effizienz auf einem idealen Parallelrechner erläutert.

Benchmark-Programme und ihre Implementierung

Im Rahmen der Arbeit werden für die Architekturbewertung alle Kerns und zwei Anwendungen verwendet. Die Einschränkung auf diese beiden Anwendungen wurde durch den Simulationsaufwand bedingt. Alle anderen Anwendungen sind so rechenintensiv, daß eine Simulation für größere Prozessorzahlen nicht in vertretbarer Zeit möglich gewesen wären. Bei der Simulation wurden die in [187] empfohlenen Problemgrößen verwendet. Die Funktionalität und Problemgröße der verwendeten Benchmarks kann wie folgt zusammengefaßt werden:

FFT: Das Programm berechnet eine komplexe 1D-FFT Transformation. Die normale Problemgröße ist 16^2.

Radix: Dieser Kern implementiert den parallelen Radix-Sortieralgorithmus. Er wird standardmäßig mit 2^{20} Ganzzahlen betrachtet.

LU: Der LU-Kern faktorisiert eine dicht besetzte $n \times n$ Matrix. Dabei wird standardmäßig eine 512×512 Matrix betrachtet.

Cholesky: Das Programm führt eine Cholsky-Zerlegung einer schwach besetzten $n \times n$ Blockmatrix durch.

Water: Das Water-Programm ist eine Simulation des Verhaltens von n Wasserteilchen. Sie hat den Aufwand $O(n^2)$.

Ocean: Dies ist eine Simulation von großflächigen Meeresströmungen. Sie basiert auf einem Gitterverfahren. Die Standardgröße beträgt 512 Gitterpunkte.

Für die obigen Benchmarks existiert eine C-Implementierung für ein Thread-basiertes Programmiermodell mit einfachen Synchronisationsoperationen. Um diese Implementierung auf einem bestimmten System laufen zu lassen, müssen lediglich einige wenige systemspezifische Makros implementiert werden. Dazu gehören die Synchronisationsoperationen sowie Makros zur Erzeugung und Vernichtung von Threads. Die Verwendung dieser Standardimplementierung zur Bewertung von Architekturen hat den Vorteil, daß die Wahrscheinlichkeit einer Verfälschung der Ergebnisse geringer wird. Hinzu kommt, daß die Ergebnisse dann besser mit anderen Arbeiten vergleichbar sind. Aus diesen Gründer wurde in der vorliegenden Arbeit die Standardimplementierung verwendet.

Absolute Anazahl in Millionen				Cache-Aussetzraten						
Programm	Instr.	Read	Write	1	2	4	8	16	32	64
cholesky	539,17	75,87	23,31	0,18	0,20	0,25	0,34	0,46	0,65	0,89
fft	34,79	4,05	2,87	1,98	1,55	1,17	1,05	1,11	1,15	1,16
lu	494,05	93,20	44,74	0,39	0,20	0,02	0,04	0,07	0,11	0,15
ocean	379.93	80,26	17,27	3,57	2,76	1,88	1,38	0,80	0,64	1,18
radix	50,99	12,06	7,03	2,35	2,35	1,87	1,39	1,41	1,57	2,20
water	460,52	69.07	26.60	0,0001	0,01	0,02	0,03	0,06	0,11	0,20

Abbildung 5.3: Die Speicherzugriffscharachteristik der SPLASH-2 Benchmarks (aus [187] und [179]). Die zweite Spalte zeigt die absolute Anzahl von Prozessoroperationen. die dritte und vierte die Anzahl darunter befindlicher Lese- bzw. Schreiboperationen. Alle diese Angaben sind in Millionen. Die Spalten Zeigen die Cache-Aussetzraten der Programme in Abhängigkeit von der Anzahl der Prozessoren. Die Raten sind in Prozent angegeben.

Speicherzugriffs-Charakteristik der Programme

Für das Verständnis des Leistung der Programme und für die theoretische Betrachtung ihrer Effizienz sind Kenntnisse des Speicherzugriffsmusters notwendig. Dabei geht es vor allem darum, wie oft ein Prozessor auf den Speicher zugreift und damit das Speichersystem belastet. Dies ergibt sich aus dem Verhältnis von Speicheroperationen zu anderen Prozessoroperationen sowie der Cache-Aussetzrate. Beide Werte wurden in [187] und [179] für jedes der obigen Programme untersucht. Da die Aussetzrate auch von der Anzahl der Invalidierungen abhängt, wurde sie für verschiedene Prozessorzahlen untersucht. Das Ergebnis dieser Untersuchung ist in Tabelle 5.3 zusammengefaßt.

Beschleunigung auf einem idealen Parallelrechner

Um die Effizienz der Programme auf den untersuchten Architekturen zu bewerten, wird ein Vergleichsmaßstab benötigt. Dieser Maßstab muß so gewählt werden, daß ein Vergleich mit den untersuchten Architekturen die Ineffizienzen des parallelen Speichersystems widerspiegelt. In vielen Arbeiten wird dazu das PRAM-Modell als der ideale Parallelrechner verwendet. Dabei wird die Ausführungszeit der betrachteten Benchmark-Programme auf einer PRAM für verschiedene Prozessorzahlen ermittelt und dann mit der Ausführungszeit auf den untersuchten Architekturen verglichen. Der Nachteil dieses Ansatzes besteht darin, daß im PRAM-Model kein Cache vorhanden ist und alle Speicherzugriffe in einem Prozessorzyklus ausgeführt werden. Dies führt dazu, daß die Ergebnisse durch Effekte verfälscht werden, die nichts mit dem parallelen Speichersystem zu tun haben, und allein auf das Vorhandensein des Caches zurückzuführen sind:

1. Mit steigender Prozessorzahl nimmt die absolute Cache-Menge zu. Bei vielen Anwendungen wird dadurch mit steigender Prozessorzahl die Anzahl der Hauptspeicherzugriffe geringer. Es kann sogar passieren, daß ein Programm plötzlich vollständig aus dem Cache läuft. Dadurch ist eine Beschleunigung zu beobachten, die nichts mit der parallelen Ausführung zu tun hat. Wegen dem großen Latenzunterschied zwischen dem Cache und dem Hauptspeicher kann diese Beschleunigung so groß sein, daß sie die Ineffizienzen des parallelen Speichersystem überschattet.

2. Bei einem System mit hoher Hauptspeicherlatenz verbringen die Prozessoren in Programmteilen, die nicht aus dem Cache laufen können, einen Großteil ihrer Zeit mit dem Warten auf den Hauptspeicher. In vielen Anwendungen kommen solche Programmteile vor allem in den parallel ausführbaren Programmbereichen vor. Dies bedeutet, daß der Zeitanteil der parallelisierbaren Programmteile an der Gesamtausführungszeit deutlich höher ist als bei der PRAM.

Bei einigen vom Autor durchgeführten Experimenten haben die obigen Faktoren dazu geführt, daß die Beschleunigung auf den untersuchten Architekturen besser als bei der PRAM war (die absolute Ausführungszeit war natürlich länger). Um das Problem zu umgehen, wird in der vorliegenden Arbeit als Vergleichsmaßstab ein ideales Speichersystem mit Cache verwendet. In einem solchen System besitzen die Prozessoren einen Cache und Hauptspeicherbänke, die mit der untersuchten Architekturen identisch sind, also die gleiche Größe, Latenz, Zeilenlänge, etc. besitzen. Auch das Cache-Kohärenz-Protokoll ist gleich. Allerdings ist die Dauer eines jeden Hauptspeicherzugriffs immer gleich und durch die feste Latenz gegeben. Diese Latenz gibt nur die DRAM-Latenz wieder. Sie beinhaltet keine Netzwerklatenz und hängt nicht vom Zugriffsmuster der Prozessoren ab. Es ist also

sichergestellt, daß die Laufzeitunterschiede zwischen dem idealen System und den untersuchten Architekturen allein auf die Ineffizienzen des parallelen Speichersystems (Netzwerklatenz, Sequentialisierung, etc.) zurückzuführen sind.

Die Ergebnisse der Simulation der ausgewählten Benchmarks mit einer solchen idealen Architektur sind in Abbildungen 5.4 und 5.5 dargestellt. Dabei wurde ein Cache von 1MB mit einer Zeilenlänge von 64Byte und einer Zugriffszeit von einem Zyklus verwendet. Die Speicherlatenz wurde auf 30 Prozessorzyklen gesetzt. Diese Werte entsprechen denen, die bei der Simulation der untersuchten Architekturen benutzt werden.

5.4.2 Der LIMES-Simulator

Das LIMES-System [176] ist ein Simulator für Intel x86 basierte Multiprozessoren mit gemeinsamem Speicher und einem Thread-parallelen Programmiermodell. Im Gegensatz zu vielen anderen Simulatoren, die den Prozessor in Software emulieren und die Anweisungen interpretieren, werden beim LIMES die Anweisungen tatsächlich vom Prozessor ausgeführt. Dies führt zu einer hohen Effizienz der Simulation, die die Ausführung komplexer Anwendungen bei hoher Prozessorzahl und realistischen, großen Datensätzen erlaubt. Die Ausführung eines Programms auf P Prozessoren wird durch ein abwechselndes, sequentielles Ausführen der Anweisungen der simulierten Prozessoren nachgeahmt. Um die Simulation des Speichersystems zu ermöglichen, wird die Ausführung bei jedem Speicherzugriff unterbrochen. Dazu wird durch einen speziellen Binder in den Assemblercode des simulierten Programms vor jede Speicheranweisung eine geeignete Sprunganweisung eingefügt.

Der LIMES-Simulator besteht aus zwei Teilen: dem Kern und einem Speichersimulator. Der Kern erfüllt drei Funktionen: die Messung der Simulationszeit, das Scheduling der Anweisungen der simulierten Prozessoren auf dem echten Prozessor und die Kommunikation mit dem Speichersimulator. Der Speichersimulator ermittelt die Kosten der Speicherzugriffe, und zwar unter Berücksichtigung aller, zum gegebenen Zeitpunkt im System befindlichen Speicheranfragen. Um ein Verständnis der im Rahmen der Arbeit durchgeführten Erweiterungen zu ermöglichen, wird im Nachfolgenden kurz beschrieben, wie die obigen Funktionen im LIMES-Simulator realisiert sind. Eine genaue Beschreibung des Aufbaus und der Funktionsweise der LIMES-Komponenten ist in [176] zu finden.

Zeitmessung

Die Zeitmessung muß zwei Dinge berücksichtigen. Zum einen kann die Zeit nicht mit Hilfe der Systemuhr gemessen werden, da die Speicherzugriffe in Software simuliert werden. Zwischen der Systemzeit und der simulierten Speicherzugriffszeit besteht also keine Korrelation. Aus diesem Grund benutzt LIMES für die Berechnung der Simulationsdauer einen eigenen Zähler: die simulierte Systemzeit. Zum anderen werden die parallelen Aktivitäten der simulierten Prozessoren und des Speichersystems in Wirklichkeit abwechselnd, sequentiell durchgeführt. D.h., daß sich der Simulator für jeden simulierten Prozessor merken muß, bis zu welchem Zeitpunkt ihre Aktivität bereits simuliert wurde. Zu diesem Zweck besitzen jeder Prozessor und der Speichersimulator einen eigenen Zähler: die lokale simulierte Zeit.

Um einen Bezug zwischen der simulierten und der echten Systemzeit herzustellen nimmt LIMES an, daß jede Anweisung (Speicherzugriffe ausgenommen) genau einen Zyklus benötigt. Die Bearbeitungszeit der Programmteile, die nicht auf den Speicher zugreifen, kann dann durch Zählen der ausgeführten Anweisungen bestimmt werden. Dazu wird der Assemblercode des simulierten Programms entsprechend instrumentiert.

Scheduling

Die einfachste Möglichkeit, eine parallele Programmausführung zu simulieren, besteht darin, abwechselnd jeweils eine Anweisung eines jeden simulierten Prozessors und einen Schritt der Speichersimulation auszuführen. Leider ist dieses Verfahren sehr ineffizient. Dies liegt darin, daß der Wechsel von der Ausführung des Programms eines simulierten Prozessors zu einem anderen bzw. zur Speichersimulation (Kontextwechsel) mit einem nicht vernachlässigbarem Verwaltungsaufwand verbunden ist. Um diesem Problem aus dem Weg zu gehen, führt LIMES einen Kontextwechsel erst dann aus, wenn ein simulierter Prozessor zu einer Speicheranweisung kommt. Die Simulation findet dabei in zwei, sich abwechselnden Phasen statt: einer Programmausführungsphase und einer Speichersimulationsphase. In der ersten Phase werden die Anweisungen eines jeden zur Ausführung bereiten simulierten Prozessors so lange ausgeführt, bis er auf einen Speicherzugriff trifft, und dann blockiert. Dabei wird nicht zwischen Cache-Hits und Cache-Misses des echten Prozessors unterschieden. D.h., daß die simulierte

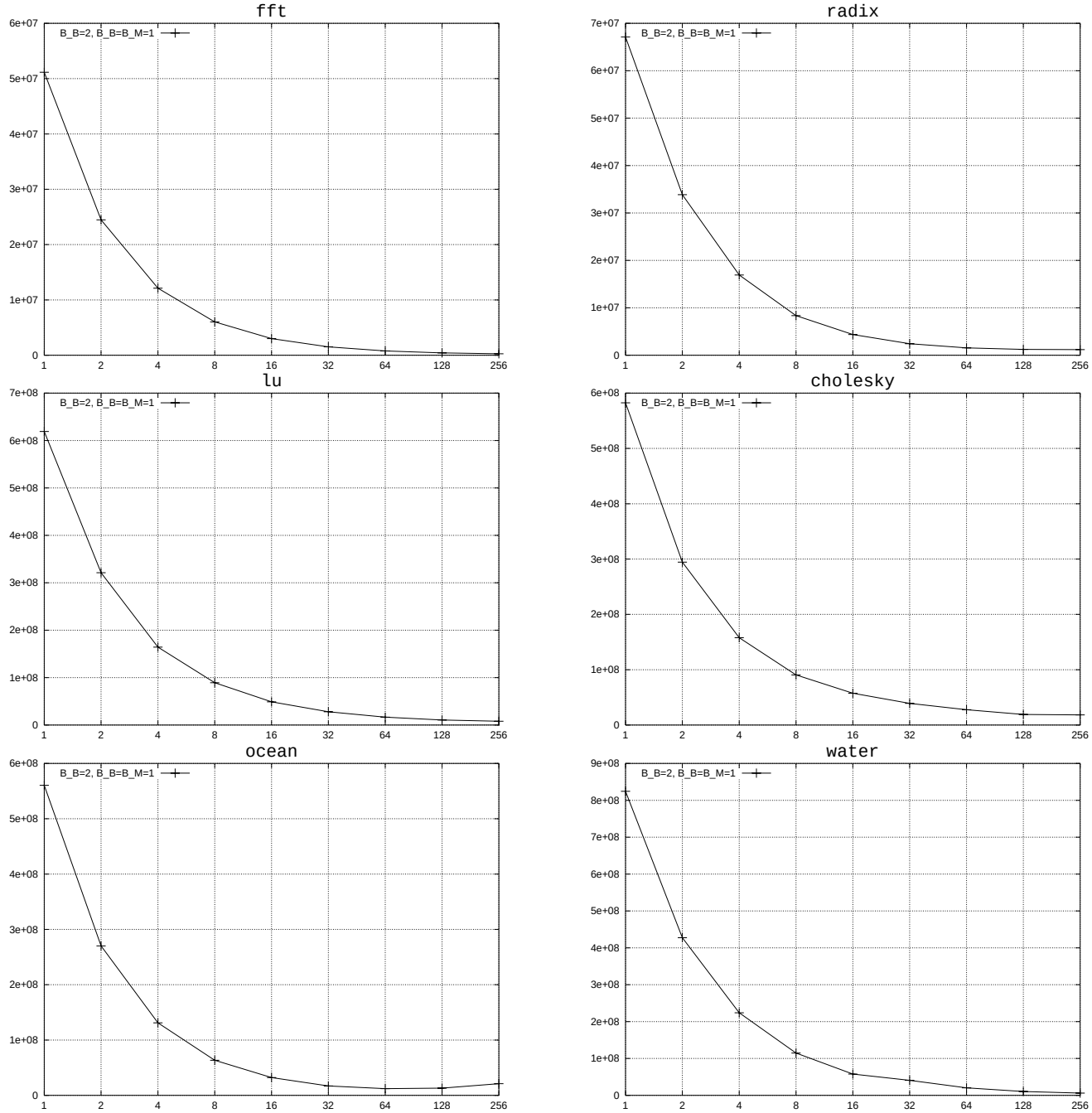


Abbildung 5.4: Die Ausführungszeit (in Zyklen) der untersuchten Benchmarks in Abhängigkeit von der Prozessorzahl für ein ideales Speichersystem mit Cache. Der Simulation basiert auf einer Speicherlatenz von 30 Zyklen und einer Cache Latenz von 1 Zyklus.

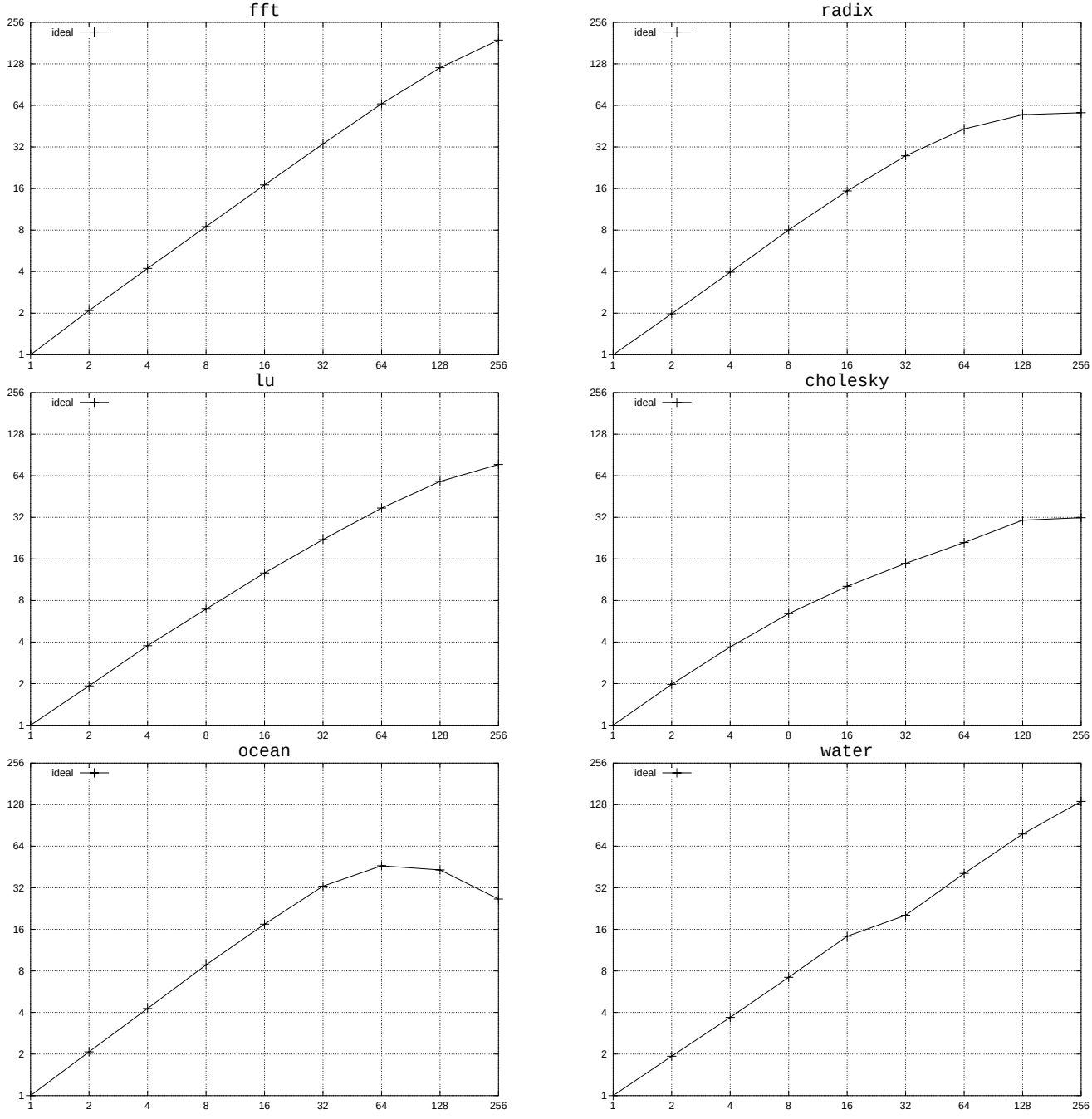


Abbildung 5.5: Die Beschleunigung der untersuchten Benchmarks in Abhängigkeit von der Prozessorzahl für ein ideales Speichersystem mit Cache. Der Simulation basiert auf einer Speicherlatenz von 30 Zyklen und einer Cache Latenz von 1 Zyklus.

Ausführung unabhängig davon blockiert wird, ob sich das benötigten Datum im Cache des echten Prozessor befindet oder nicht. Die Speichersimulation kann so sowohl die Caches der einzelnen simulierten Prozessoren, als auch das Netzwerk und die Hauptspeichermodule berücksichtigen. Sobald alle simulierten Prozessoren blockiert sind, werden dem Speichersimulator die Daten der Speicherzugriffe übergeben. In der zweiten Phase wird die Speichersimulation vorangetrieben, und zwar beginnend mit dem Zeitpunkt, in dem sie beim letzten Aufruf des Speichersimulators beendet wurde. Sie wird solange vorangetrieben, bis ein oder mehrere Speicherzugriffe beendet sind. Daraufhin werden die simulierten Prozessoren, deren Zugriffe beendet wurden, als bereit für die Ausführung markiert und die Simulation wird mit dem ersten Schritt fortgesetzt.

Der Kern und die Speichersimulation

Bei dem LIMES-System sind die tatsächliche Ausführung der Speicheroperationen und die Simulation ihrer Kosten getrennt. Die tatsächliche Ausführung wird vom (echten) Prozessor während der Programmausführungsphase vorgenommen, wobei auf den echten Speicher zugegriffen wird. Die Simulation der Zugriffskosten wird durch den Speichersimulator erledigt. Wie oben dargelegt, wird sie immer dann gestartet, wenn alle simulierten Prozessoren bei der Programmausführung zu einer Speicheranweisung gelangt sind. Der Kern macht keine Annahmen über die interne Funktion der Speichersimulation. Er verlangt lediglich, daß die Simulationsroutine

1. sich an die im Kern definierte Aufrufschnittstelle hält,
2. pro Aufruf die Simulation so weit vorantreibt, wie das simulierte System innerhalb eines Prozessorzyklus vorankommen kann (Operationen, die mehrere Prozessorzyklen benötigen, werden also im Verlauf mehrerer Aufrufe simuliert),
3. bei der Rückkehr dem Kern mitteilt, ob, und wenn ja, welcher Speicherzugriff erfolgreich beendet wurde.

Dies erlaubt die Einbindung einer breiten Palette von Speichersimulatoren, ohne daß der Kern modifiziert werden muß.

Im LIMES integrierte Speichersimulatoren

Das LIMES-System beinhaltet einen Speichersimulator für Bus-basierte Systeme mit drei verschiedenen Snoopy Cache-Kohärenz-Protokollen: einem einfachen WTI-Protokoll, dem Berkely-Protokoll und dem Dragon-Protokoll. Der Simulator modelliert die Komponenten des Speichersystems mit Hilfe von 3 Klassen: einer generischen Cache-Speicher Klasse (`Cache`), einer Klasse , die den Cache-Kontroller und das Cache-Kohärenz-Protokoll implementiert (`xxx_Cache`, wobei `xxx` durch die Bezeichnung des Cache-Kohärenz-Protokolls ersetzt wird) und einer Bus-Klasse (`Bus`). Die `xxx_Cache`- und `Bus`-Klassen besitzen für den Datenaustausch untereinander und die Kommunikation mit den simulierten Prozessoren spezielle Ein-/Ausgabefelder, die den Ein-/Ausgabekanälen der entsprechenden Komponenten nachempfunden sind. Dabei wird für die Darstellung von Busnachrichten eine eigene Klasse, `Channel_Signals`, verwendet, die die Daten und Kontrolleitungen des Busses modelliert. Der Ablauf der Simulation und der Datenfluß zwischen den Ein-/Ausgabefeldern der einzelnen Komponenten werden durch eine Kontrollklasse bewältigt (`Topology`). Sie versteckt die Topologie des Systems und die Details der Kommunikation vor den anderen Klassen, die dadurch in verschiedenen Systemvarianten unverändert eingesetzt werden können. So ließe sich das einfache Bussystem z.B. ohne Veränderungen in den Cache- und Bus-Klassen zu einer Multibusarchitektur umfunktionieren. Die Funktion dieser Klassen kann wie folgt zusammengefaßt werden:

Topology-Klasse: Die `Topology`-Klasse ist für drei Dinge zuständig: den Aufbau des Systems gemäß der vorgegebenen Topologie, die Kontrolle des Simulationsablaufs, und den Datentransfer zwischen den Komponenten. Ersteres beinhaltet die Erzeugung einer Instanz der `xxx_Cache`-Klasse für jeden simulierten Prozessor, sowie die Erzeugung und Initialisierung einer Instanz der Bus-Klasse. Die Kontrolle des Simulationsablaufs erfolgt mit Hilfe von `cycle`-Methoden, die die `xxx_Cache`- und `Bus`-Klassen zur Verfügung stellen müssen. Sie werden von der `Topology`-Klasse aufgerufen, um einen Zyklus der jeweiligen Komponente zu simulieren. Für die Kommunikation stellt die `Topology`-Klasse Methoden zur Verfügung, mit denen die anderen Systemkomponenten die an sie gerichteten Nachrichten lesen können. Diese Methoden übertragen die Daten aus den entsprechenden Ausgabefeldern der Quelle in das geeignete Eingabefeld des Aufrufers. Um welche Felder es sich dabei handelt, hängt von der Topologie und der Funktionsweise des Speichersystems ab.

Cache-Klasse: Diese Klasse ist für die Verwaltung des eigentlichen Cache-Speichers zuständig. Sie beinhaltet unter anderem Methoden zum Einfügen einer Zeile in den Cache, zur Veränderung ihres Zustands und zur Überprüfung, ob, und in welchen Zustand sich eine Adresse im Cache befindet. Die Cache-Klasse abstrahiert von der Länge, und den möglichen Zuständen der Cache-Zeile.

xxx_Cache-Klasse: Für jeden simulierten Prozessor wird bei der Simulation des Speichersystems eine Instanz der `xxx_Cache`-Klasse erzeugt. Sie implementiert die gesamte Cache-Verwaltung des zugehörigen Prozessors auf der Basis eines bestimmten Cache-Kohärenz-Protokolls. Um welches Protokoll es sich dabei handelt, kann an dem Präfix abgelesen werden, der anstelle der `xxx` in den Namen eingesetzt wird (also z.B. `WTI_Cache` für das WTI-Protokoll).

Die `xxx_Cache`-Klasse muß die Anfragen des zugehörigen Prozessors lesen, bearbeiten und den Kern benachrichtigen, sobald die Anfrage befriedigt wurde. Zu diesem Zweck beinhaltet sie eine Instanz der `Cache`-Klasse, sowie jeweils eine Kommunikationsschnittstelle für den Kern und den Bus. Die Busschnittstelle besteht aus zwei Feldern vom Typ `Channel_Signals`: einen für die Ausgabe auf den Bus, und einen für die Eingabe. Liegt an der Kern-Schnittstelle eine Anfrage vor, so wird zunächst geprüft, ob, und in welchem Zustand sich die entsprechende Zeile im Cache-Speicher befindet. Dazu wird eine entsprechende Methode der `Cache`-Klasse aufgerufen. Kann die Anfrage aus dem Cache befriedigt werden, so wird eine 'Fertig'-Meldung an die Kern-Schnittstelle weitergegeben und die Anfrage als erledigt betrachtet. Anderenfalls wird über den Bus-Ausgabekanal der Zugriff auf den Bus angefordert. Sobald dieser gewährt wird, erscheint im Bus-Eingabekanal eine entsprechende Benachrichtigung. Daraufhin schreibt die `xxx_Cache` die benötigte Busanfrage in den Ausgabekanal und wartet darauf, daß die Antwort des Speichersystems auf dem Bus-Eingabekanal erscheint. Sie fügt dann die neuen Daten in den Cache ein und benachrichtigt den Kern über den entsprechenden Kanal, daß die Anfrage erfolgreich beendet wurde.

Neben der Bearbeitung der Kern-Anfragen muß die `xxx-Cache`-Klasse die Zustände der Cache-Zeilen in Abhängigkeit von den Bus-Operationen gemäß der Spezifikation des jeweiligen Cache-Kohärenz-Protokolls verändern. Um dies zu ermöglichen, wird sichergestellt (siehe unten), daß jede über den Bus übertragene Schreibanfrage in dem Bus-Eingabekanal einer jeden Instanz der `xxx_Cache`-Klasse erscheint. Sie prüft dann mit Hilfe der `Cache`-Klasse, ob die zugehörige Cache-Zeile in dem eigenen Cache vorhanden ist und modifiziert bei Bedarf ihren Zustand.

Bus-Klasse: Die Bus-Klasse modelliert einen Bus, an den eine Speicherbank angeschlossen ist. Sie besteht aus einem Ein- und einem Ausgabekanal sowie einem Arbiter.

5.4.3 Simulation des opto-elektronischen Speichersystems

Im Rahmen der Arbeit wurden die Simulation des Cache-Kohärenz-Protokolls und des Hauptspeichersystems neu entwickelt und implementiert. Bei der Implementierung kam es vor allem auf zwei Aspekte an: die Zuverlässigkeit der Simulation und die Möglichkeit, durch die Simulation die Korrektheit der Protokolle zu untermauern.

1. Da der LIMES -Simulator ausgiebig getestet und verifiziert wurde [176], wurde versucht, so viele Funktionen wie möglich aus dem bestehenden Original-Simulator unverändert zu übernehmen. Dazu gehören der Kern und die Simulation der Cache-Speicher. Der Aufbau und die grundlegenden Datenstrukturen der Speichersimulation wurden ebenfalls beibehalten, allerdings mit einigen Veränderungen. Lediglich die Simulation des opto-elektronischen Speichersystems wurde neu implementiert.

2. Der Weg der Speicheroperationen durch das opto-elektronische Speichersystem wurde nach dem Prinzip eines Ereignis-gesteuerten Simulators realisiert. Dabei werden Instanzen einer Klasse, die die Speicheranfragen darstellen, durch Instanzen von Klassen 'durchgereicht', die die einzelnen Systemkomponenten simulieren. Das Durchreichen ist so realisiert, daß es den physikalischen Nachrichtenfluß möglichst genau nachbildet. Dazu gehört auch, daß Fehler in der Flußkontrolle und in dem Kommunikationsprotkoll genauso wie in einem echten Netzwerk zum Verlust von Nachrichten oder Verklemmungen führen können.

Im Nachfolgenden werden zunächst der Aufbau und der Ablauf der Simulation im Überblick beschrieben. Danach wird die Nachbildung der einzelnen Komponenten des Speichersystems erläutert.

Aufbau der Simulation

Für die Modellierung des opto-elektronischen Speichersystems wurde dem Simulator für jede Art von Systemkomponenten eine eigene Klasse (Komponenten-Klasse) hinzugefügt. Dazu gehören eine Cache-Kontroller-Klasse für das Berkeley Cache-Kohärenz-Protokoll (`Berk_Cache`), eine Klasse für die Netzwerkkanäle (`Net_Channel`), eine für die Schnittstellen zwischen dem Prozessor und dem Netzwerk (`PIU`), eine für die Netzwerkschnittstelle von Speicherbänken (`MIU`), und eine für den SP Chip (`OEBUS`). Jede der obigen Klassen besitzt eine Reihe von Eingabefeldern, eine Reihe von Ausgabefeldern und eine `cycle`-Methode. Die Ein-/Ausgabefelder dienen zur Modellierung der Kommunikationskanäle, über die die einzelnen Komponenten in dem simulierten System miteinander verbunden sind. So besitzt z.B. die `OEBUS`-Klasse ein Eingabefeld für jeden optischen Eingabekanal und ein Ausgabefeld für den optischen Broadcastkanal. Die `cycle`-Methode simuliert einen Zyklus der jeweiligen Komponente.

Bei der Simulation einer konkreten Systemvariante wird für jede Komponente des Systems eine Instanz der entsprechenden Klasse erzeugt. Um das System möglichst vielseitig zu gestalten, werden in Anlehnung an die original LIMES-Speichersimulation die Topologie des Verbindungsnetzwerks und die Kontrolle des Simulationsablaufs in die Kontrollklasse `Topology` ausgelagert.

Ablauf der Simulation

Nach dem Aufruf des Speichersimulators werden zunächst die anstehenden Speicheranfragen vom Kern gelesen. Daraufhin wird die Simulation solange vorangetrieben, bis einer der simulierten Cache-Kontroller in seinem Ausgabefeld anzeigt, daß seine Speicheranfrage erfolgreich beendet wurde. Dies erfolgt in Schritten, die jeweils einem Prozessorzyklus entsprechen. Diese Schritte werden von dem Simulator mitgezählt und bei der Rückkehr in den Kern diesem mitgeteilt. Sie werden dann zu der Ausführungszeit des Programms dazuaddiert.

Der Ablauf einer Speicheroperation wird simuliert, indem eine Instanz der `Channel_Signals`-Klasse, durch die Instanzen der Komponenten-Klassen durchgereicht wird. Die `Channel_Signals` stellt die Speicheranfrage dar, und beinhaltet all die Informationen, die zu ihrer Durchführung notwendig sind. In jedem Zyklus der Simulation werden die `cycle`-Methoden aller Komponenten-Klassen Instanzen in einer geeigneten Reihenfolge aufgerufen. Diese Methode simuliert die Funktionsweise der jeweiligen Komponenten in drei Schritten.

1. Im ersten Schritt werden alle an die Komponente gerichteten Nachrichten in die dafür bestimmten Eingabefelder eingelesen.

2. Im zweiten Schritt werden unter Berücksichtigung dieser Nachrichten die für einen Zyklus vorgesehenen internen Operationen durchgeführt. Dabei wird u.a. ermittelt, ob, und wohin die Nachrichten weitergeschickt werden sollen.

3. Die Nachrichten, die zur Weitergabe bereit sind, werden in die Ausgabefelder kopiert, die zu den entsprechenden Ausgabekanälen gehören. Diese Daten werden im nächsten Simulationszyklus in die Eingabefelder der entsprechenden Komponenten eingelesen.

Den Ausgangs- und Endpunkt bei der Simulation einer Speicheranfrage bildet immer eine Instanz der Cache-Kontroller-Klasse `Berk_Cache` . Eine Anfrage, die nicht aus dem Cache befriedigt werden kann, wird von der `Berk_Cache` an eine Instanz der `PIU`-Klasse weitergegeben, die sie an eine Instanz der Netzwerk-Kanal-Klasse `Net_Channel` weiterleitet. Da an den Ausgabekanal eines Cache-Kontrollers in der Regel nur der SP Chip angeschlossen ist, gelangt die Anfrage von dort in eine Instanz der `OEBUS`-Klasse. Von dort geht es wieder in eine Instanz der `Net_Channel`-Klasse. Daraufhin wird sie in die Eingabefelder aller, an diesem Kanal angeschlossenen Komponenten kopiert. Im Falle eines einfachen Bussystems mit nur einem Broadcast-Kanal handelt es sich dabei um die simulierten Netzwerkschnittstellen aller Cache-Kontroller und Speicherbänke, also alle im System vorhanden Instanzen der `PIU`-, `MIU`-Klassen. Dabei passieren gleichzeitig zwei Dinge:

1. Die `MIU`-Instanzen überprüfen, ob es sich um eine Anfrage an ihre Speicherbank handelt. Trifft dies für eine bestimmte Instanz zu, dann übernimmt sie die Anfrage. Sie wartet zunächst so viele Zyklen ab, wie die Speicherlatenz beträgt und schickt die Antwort an die Instanz der `Net_Channel`-Klasse, die den Netzwerkkanal zwischen der Speicherbank und dem SP-Chip repräsentiert. Von dort gelangt die Antwort zurück zu dem simulierten SP-Chip.

2. Die simulierten Prozessorschnittstellen betrachten die Art der Operation und schicken die Nachricht gegebenenfalls an die zugehörige `Berk_Cache` Instanz. Diese benutzt sie entweder zur Cache-Kohärenz Prüfung oder führt eine Supply-Operation durch. Im letzteren Fall wird das Ergebnis, wie oben für eine Speicheranfrage beschrieben, über die entsprechenden `PIU`- und `Net_Channel`-Instanzen an die `OEBUS`-Klasse zurückgeschickt.

Wenn der SP eine Antwort erhalten hat, dann schickt er sie auf den Weg zurück zu dem Cache-Kontroller, der sie abgeschickt hatte. Bei einem einfachen Bussystem bedeutet dies, daß die Anwort zunächst an die `Net_Channel`-Klasse weitergegeben wird, die den Broadcast-Kanal darstellt. Von dort wird sie wieder in die entsprechenden Eingabefelder aller `PIU`- und `MIU`-Instanzen kopiert.

Simulation der Cache-Kontroller

Der Cache-Kontroller besitzt ein Ein-/Ausgabefeld für die Kommunikation mit dem, durch den Kern simulierten Prozessor, und eines für den Datenaustausch mit der Netzwerkschnittstelle. Außerdem beinhaltet er eine Instanz der `Cache`-Klasse für die Simulation des eigentlichen Cache-Speichers.

Zu Beginn eines jeden Simulationszyklus einer Instanz der `Berk_Cache`-Klasse werden mit Hilfe der Kontrollklasse alle an den Cache-Kontroller gerichteten Nachrichten in die entsprechenden Eingabefelder kopiert. Dazu gehören alle im Ausgangsfeld der zugehörigen Netzwerkschnittstelle (=Instanz der `PIU`-Klasse) vorhanden Nachrichten sowie, falls vorhanden, die vom simulierten Prozessor kommende Speicheranfrage. Daraufhin wird die für die Erhaltung der Cache-Kohärenz zuständige Methode aufgerufen. Sie überprüft für jede der neuen Nachrichten, ob sie zu einer Veränderung des Zustands einer Cache-Zeile führt und nimmt bei Bedarf diese Veränderung vor. Als letztes wird die Simulation der Speicheranfrage um einen Zyklus vorangetrieben. Diese Simulation erfolgt in zwei Stufen. In der ersten Stufe wird geprüft, ob die Anfrage aus dem Cache befriedigt werden kann. Ist dies der Fall, so wird eine Erfolgsmeldung in das Ausgabefeld zum Prozessor geschrieben und der Zugriff ist beendet. Der Speicherzugriff dauert dann nur einen Zyklus. Anderenfalls wird gemäß der Spezifikation des Berkeley-Protokolls eine geeignete Anfrage an das Speichersystem geschickt. Dazu wird eine entsprechend initialisierte Instanz der `Channel_Signals`-Klasse an die zuständige Instanz der `PIU`-Klasse weitergegeben. Der simulierte Cache-Kontroller geht dann in einen Wartezustand über, und zwar solange, bis die bearbeitete Anfrage von der simulierten PIU zurückkommt. Wie viele Zyklen die Simulation der zweiten Stufe dauert, hängt davon ab, wie lange die Anfrage im Speichersystem verbleibt. Dies hängt wiederrum davon ab, durch welche Komponenten die Anfrage durchgehen muß und wie lange sie in ihnen jeweils verweilt. Beides ist durch die Parameter des simulierten System festgelegt (siehe auch unten).

Netzwerkschnittstellen der Prozessoren

Die Netzwerkschnittstellen auf der Prozessorseite besitzen einen Ein- bzw. Ausgabekanal für jeden Netzwerkkanal, an den der Prozessor in der untersuchten Architektur angeschlossen ist. Hinzu kommt ein Ein-/Ausgabekanal für den Datenaustausch mit dem Cache-Kontroller. Am Anfang des Simulationszyklus werden die, an den Ausgängen aller eingehenden Netzwerkanäle bereitstehenden Nachrichten in die entsprechenden Eingabefelder kopiert. Außerdem werden alle am Ausgangskanal des Cache-Kontrollers vorhanden Anfragen übernommen. Dies erfolgt mit Hilfe geeigneter Methoden der Kontrollklasse. Daraufhin wird geprüft, welche der vom Netzwerk gelesenen Nachrichten an den Cache-Kontroller weitergeleitet werden sollen. Diese werden in das entsprechende Ausgabefeld kopiert, von wo sie der Cache-Kontroller im nächsten Zyklus einlesen kann. Falls Anfragen seitens des Cache-Kontrollers in dem zugehörigen Eingabefeld vorhanden sind, dann werden sie an das Ausgangsfeld eines entsprechenden Netzwerkkanals weitergegeben.

Speicherschnittstelle und Speicherbank

Die `MIU`-Klasse beinhaltet Ein-/Ausgabefelder für die angeschlossenen Kanäle und eine Unterklasse für die Simulation einer Speicherbank. Sie untersucht alle in ihren Eingabefeldern vorliegenden Anfragen und sucht diejenigen heraus, die für sie bestimmt sind. Diese werden dann an die Speicherbank-Unterklasse übergeben und dort gemäß der Speicherlatenz verzögert. Danach werden die Antworten an das Netzwerk geschickt, indem eine entsprechend initialisierte Instanz der `Channel_Signals`-Klasse in ein Ausgabefeld geschrieben wird.

Die `MIU`-Klasse geht davon aus, daß sie niemals eine neue Anfrage zur Bearbeitung bekommt, solange sie noch mit einer alten beschäftigt ist. Falls dies trotzdem passiert, so wird die alte Anfrage in der Speicherbank-

Unterklasse überschrieben und geht damit verloren. Falls die Anfrage nicht von einem Besitzer-Prozessor befriedigt wird, so wartet ihr Urheber (eine Instanz der `BerkCache`-Klasse) also vergeblich auf Antwort. Das System 'hängt sich auf'. Dies ist ein Beispiel dafür, wie beim Durchreichen der Anfragen durch die Instanzen der Komponenten-Klassen Fehler in der Flußkontrolle bzw. dem Kommunikationsprotokll entdeckt werden können.

Simulation der SP Netzwerkbausteine

Die opto-elektronischen Netzwerkbausteine werden durch die `OEBUS`-Klasse simuliert. Sie besitzt für jeden an den Baustein angeschlossenen Netzwerkkanal ein Ein- bzw. Ausgabefeld sowie eine Menge von Unterklassen, die die internen Komponenten des Bausteins simulieren. Diese sind von Architektur zu Architektur unterschiedlich und werden in den jeweiligen Kapiteln beschrieben.

Simulation der Netzwerkkanäle

In der Simulation werden Netzkanäle modelliert, die aus einem Sender, einer einfachen, unidirektionalen Leitung, und einem Empfänger bestehen. In einem solchen Kanal setzt sich die Verzögerung, die eine Nachricht erfährt, aus zwei Faktoren zusammen: der Zeit, die für das Senden der Nachricht benötigt wird, und der Laufzeitverzögerung in der Leitung. Erstere ergibt sich aus der Bandbreite des Kanals und der Länge der Nachricht. Letzteres hängt von der Länge der Leitung und der Signalausbreitungsgeschwindigkeit ab. Dabei wird angenommen, daß der Sender sofort nach Erhalt einer Nachricht mit dem Senden beginnt. Eventuell vorhandene Setup-Zeiten werden nicht von dem Kanal, sondern von der sendenden Komponente simuliert. Für die Leistung des Speichersystems ist neben der Dauer einer Übertragung wichtig, wie schnell mehrere Nachrichten hintereinander verschickt werden können. Dabei muß berücksichtigt werden, daß das Senden einer neuen Nachricht beginnen kann, sobald der Sender mit der vorherigen Nachricht fertig ist. Es ist also nicht notwendig zu warten, bis die letzte Nachricht am Ausgang der Leitung angelangt ist. Dies ist insbesondere bei langen Kanälen mit hoher Latenz wichtig.

Um die obigen Eigenschaften eines Kanals zu simulieren, besitzt die `Net_Channel`-Klasse eine Liste von Kanalelementen (durch die `Channel_Element` Klasse dargestellt). Jedes Kanalelement besitzt ein Datenfeld vom Typ `Channel_Signals`. Die Länge der Liste entspricht der Latenz der Leitung. Dabei stellt das erste Element den Eingang und das letzte den Ausgang des Kanals dar. Eine Nachricht wird durch den Kanal transportiert, indem sie solange von einem Element ins nächste kopiert wird, bis sie beim letzten Element angelangt ist. Mit Ausnahme des Eingangs verweilen die Nachrichten in jedem Element genau einen Zyklus. Dadurch wird die Latenz des Kanals simuliert. Die für das Senden benötigte Zeit wird durch die Verzögerung im ersten Kanalelement simuliert. Sie gleicht dem Quotienten aus der Länge der Nachricht und der Bandbreite des Kanals.

5.5 Theoretische Analyse

Um die Zuverlässigkeit der Simulation zu überprüfen, werden ihre Ergebnisse mit der theoretischen Analyse der untersuchten Architekturen verglichen. Dazu werden einfache, schlangentheoretische Modelle der Architekturen aufgestellt und numerisch gelöst. Um die Last für die Lösung zu modellieren, werden die in Tabelle 5.3 angegeben Daten für die Speicherzugriffscharakteristik verwendet.

Im Nachfolgenden wird zunächst die grundsätzliche Vorgehensweise bei der Modellierung und beim Vergleich mit den Simulationsergebnissen erläutert. Danach werden die Grundideen der theoretischen Modellierung und die für die Modellierung verwendeten Lastparameter beschrieben. Abschließend werden die Näherungen, auf denen die Betrachtung basiert, zusammengefaßt und begründet.

5.5.1 Vorgehensweise

Die theoretische Modellierung von Rechnerarchitekturen ist ein Forschungsgebiet für sich. Die Schwierigkeit besteht dabei darin, daß mit zunehmendem Detail der Modellierung die resultierenden Gleichungen selbst numerisch kaum zu lösen sind. Hinzu kommt, daß die Lösungsverfahren nur unter der Annahme funktionieren, daß die Speicherzugriffe einer statistischen Verteilung, vorzugsweise der Gleichverteilung gehorchen. Dies ist in realen Programmen aber nur selten der Fall. Die theoretische Modellierung stellt daher immer nur eine Näherung dar. Allerdings wurde in zahlreichen Forschungsarbeiten gezeigt, daß diese Näherung mit entsprechendem Aufwand sehr genau gemacht werden kann [33]. Das Ziel der in der Arbeit vorgenommenen Modellierung besteht allerdings nicht darin, eine möglichst hohe Genauigkeit zu erreichen. Der hierfür benötigte Aufwand würde den Rahmen

der Arbeit sprengen. Statt dessen soll ein grobes, einfaches Modell aufgestellt werden, das eine Überprüfung der Plausibilität der Simulationsergebnisse ermöglicht. Dazu muß die Analyse zwei Anforderungen erfüllen:

1. Sie soll die Abhängigkeit der Effizienz von der Prozessorzahl bei gegebenen technologischen Parametern qualitativ korrekt wiedergeben. Die analytisch ermittelte Kurve für die Abhängigkeit der Programmlaufzeit von der Prozessorzahl soll also die korrekte Form haben. Insbesondere soll der Punkt, an dem die Leistung des System aufgrund der Überlastung der Broadcast-Kanäle einbricht, möglichst genau vorhergesagt werden.

2. Die analytisch ermittelten absoluten Werte für die Programmlaufzeit sollen größenordnungsmäßig korrekt sein.

5.5.2 Schlangentheoretische Modellierung des Systems

Das Problem bei der theoretischen Berechnung der Dauer bestimmter Operationen in einem Rechnersystem besteht darin, daß diese Dauer von der Systemlast abhängig ist. So hängt die Latenz eines Speicherzugriffs eines Prozessors davon ab, ob dieser Prozessor als einziger auf eine Speicherbank zugreift. Ist dies der Fall, so ergibt sich die Latenz des Zugriffs als Summe der Latenzen des Prozessor-Speichernetzwerks und der Speicherbank und kann einfach und exakt bestimmt werden. Anderenfalls müssen Wartezeiten berücksichtigt werden, die dann entstehen, wenn die Speicherbank oder Netzwerkkanäle durch andere Prozessoren besetzt sind. Die Länge der Wartezeiten hängt davon ab, wann welcher Prozessor welche Anweisungen ausführt. Somit hängt sie auch von den Wartezeiten bei früheren Zugriffen ab.

Eine Möglichkeit, ein solches System analytisch zu modellieren, besteht darin, die Speicherzugriffe als zufällig verteilt zu betrachten, und die Länge der Wartezeiten mit statistischen Mitteln zu berechnen. Dies wird im Rahmen der Warteschlangentheorie gemacht. Je nachdem, was für ein System analysiert werden soll und welche Genauigkeit benötigt wird, bietet die Warteschlangentheorie ein Fülle von Modellierungsmöglichkeiten. Ein Überblick ist z.B. in [33, 60] zu finden In der Arbeit werden einfache sog. geschlossene Modelle betrachtet. Im Nachfolgenden wird zunächst die Grundidee dieser Modelle erläutert. Danach wird skizziert, wie die untersuchten Architekturen modelliert werden und wie die Modelle gelöst werden.

Grundlagen von Warteschlangenmodellen

Das System wird als ein gerichteter Graph aus sog. Bedienstationen modelliert, in der sich eine bestimmte konstante Anzahl von Aufträgen befindet. Die Bedienstationen stellen die Systemkomponenten dar, die Aufträge die Prozessoranfragen. Die Kanten stellen mögliche Wege dar, die die Prozessoranfragen zwischen den Systemkomponenten nehmen können. Die Anzahl der Aufträge ist konstant, da jeder Auftrag die Tätigkeit eines Prozessors modelliert. Falls der Prozessor gerade keinen Speicherzugriff durchführt, dann befindet sich der Auftrag in der Bedienstation, die den Prozessor modelliert. Am sonsten belegt er eine Station, die eine Komponente des Speichersystems modelliert. Es durchaus möglich, daß von einer Bedienstation mehrere Kanten weitergehen. Dies bedeutet, daß die Anfragen nach dem Verlassen der korrespondierenden Komponente je nach Situation zu einer anderen Komponente weitergehen können (so wie Speicheranfragen vom Bus je nach Adresse zu einer anderen Speicherbank weitergeleitet werden können).

Das Modell geht davon aus, daß die Aufträge entlang der Kanten von einer Bedienstation zur anderen wandern. An jeder Bedienstation werden sie um einen Betrag verzögert, der sich aus einer Bearbeitungszeit und einer Wartezeit zusammensetzt. Um die Verzögerung nachzubilden, besteht jede Bedienstation aus einer oder mehreren Bedieneinheiten und einer Warteschlange. Dabei simuliert eine Bedieneinheit die Verzögerung, die bei der Bearbeitung eines Auftrags in der Komponente entsteht. Das Vorhandensein mehrerer Bedieneinheiten bedeutet, daß die Bedienstation mehrere Aufträge parallel bearbeiten kann. Falls bei der Ankunft eines Auftrags alle Bedieneinheiten besetzt sind, wird der Auftrag in die Warteschlange eingereiht. Dort verbleibt er, bis er an der Reihe ist, in einer freien Bedieneinheit bearbeitet zu werden.

Die Warteschlangentheorie erlaubt für ein, nach obigen Regeln aufgestelltes Modell eines Systems, die *durchschnittlichen* Verweilzeiten der Aufträge in den einzelnen Bedienstationen zu berechnen. Diese Verweilzeiten beinhalten sowohl die durchschnittliche Wartezeit als auch die Bearbeitungszeit. Aus ihnen kann durch einfaches Summieren der Beiträge entsprechender Komponenten die durchschnittliche Gesamtlatenz des Systems für die Prozessoranfragen berechnet werden.

Um die durchschnittlichen Verweilzeiten zu berechnen, wird in der Warteschlangentheorie angenommen, daß die Ankunftszeiten der Aufträge an den Bedienstationen zufällig verteilt sind. Außerdem werden die folgenden Informationen benötigt:

Die Anzahl der Aufträge K: Die Anzahl der Aufträge in einem geschlossenen Netz ist per Definion eines solchen Netzes konstant.

Die Bedienrate in der Station i, μ_i: Die Bedienrate für die Bedienstation i gibt an, wieviele Aufträge die Bedieneinheit dieser Station im Schnitt pro Zeiteinheit abarbeiten kann. Es ist eine positive reelle Zahl, die sowohl größer als auch kleiner als 1,0 sein kann.

Die Besuchshäufigkeit: Die Besuchshäufigkeit gibt für jede Station i an, wie oft ein Auftrag während eines Durchlaufs durch das System im Durchschnitt zu dieser Station gelangt. Es ist ebenfalls eine reelle Zahl, die sowohl kleiner als auch größer als 1,0 sein kann. Ersteres ist der Fall, wenn ein Auftrag je nach Situation von einer anderen Einheit bearbeitet wird. Letzteres bedeutet, daß eine Komponente für die Bearbeitung einer Anfrage mehrmals benötigt wird. So wäre in einem Speichersystem mit einem paketvermittelnden Bus und M Speicherbänken die Besuchshäufigkeit für Speicheranfragen für den Bus gleich 2 (einmal zur Speicherbank und einmal zurück) und für die einzelnen Speicherbänke gleich $1/M$.

Modellierung der Rechnerarchitekturen

Die genauen Modelle der einzelnen Architekturen werden in den zugehörigen Kapiteln beschrieben. Bei ihrer Erstellung wird wie folgt vorgegangen:

1. Für jeden Prozessor befindet sich im System ein Auftrag, die Anzahl der Aufträge ist also gleich der Prozessorzahl.

2. Die Prozessoren werden durch eine sog. Terminal-Bedienstation dargestellt. Eine solche Station zeichnet sich dadurch aus, daß sie für jeden Auftrag im System eine Bedieneinheit besitzt. An ihr treten also nie Wartezeiten auf. Aufträge, die bei einer Terminal-Station ankommen, werden um die Bedienzeit verzögert und dann wieder ins System geschickt. Das besondere ist, daß die Aufträge, die sich in der Terminal-Station aufhalten, keine Systemresource belegen. Sie verursachen damit keine Wartezeiten für andere Aufträge. Die Verweildauer in der Terminal-Station stellt also die Zeit dar, in der ein Prozessor mit Anweisungen beschäftigt ist, die keine Speicheranfragen sind. Die Bedienrate dieser Stationen gibt damit die Anzahl der Speicheroperationen an, die ein Prozessor pro Zeiteinheit generiert.

3. Die Netzwerkkanäle und die Speicherbänke werden durch Bedienstationen mit einer Bedieneinheit und einer FIFO Warteschlangenstrategie dargestellt. Die Bedienraten ergeben sich für diese Stationen aus der Leistungsfähigkeit der modellierten Komponenten und der Komplexität der Anfragen in der betrachteten Architektur.

Diese Vorgehensweise wird in Abbildung 5.6 für einen paketvermittelnden Bus mit zwei Prozessoren und zwei Speicherbänken verdeutlicht. Die beiden Prozessoren werden durch die Terminal-Bedienstation modelliert. Die Zeit, während der sich ein Auftrag in dieser Station befindet, stellt die Zeit zwischen zwei Speicheranfragen eines Prozessors dar. Von der Prozessor-Station gelangen alle Anfragen zu einer Station, die den Bus modelliert und von dort zu den Speicherbänken. Letztere werden durch zwei gleiche parallele Bedienstationen mit der Besuchshäufigkeit von jeweils $0,5$ modelliert. Diese Besuchshäufigkeit ergibt sich aus der Überlegung daß die Speicherzugriffe gleichmäßig auf die Bänke verteilt sind. Die Bedienrate dieser Stationen ist durch die Speicherlatenz gegeben. Von den Speicherbänken geht es zurück zur Bus-Station und von dort wieder zu den Prozessor-Stationen. Die die Aufträge zwei Mal bei der Bus-Station landen, besitzt diese die Besuchszahl 2. Wenn man zum ersten Mal mit solchen Modellen konfrontiert wird, stellt sich an dieser Stelle die Frage, wie an der Bus-Station zwischen den, vom Prozessor und den, von den Speicherbänken kommenden Anfragen unterschieden wird. Die Antwort ist:: 'gar nicht'. Für die Berechnung der Verweilzeiten ist es nur relevant, wie lange der Auftrag bei einem Durchlauf durch das System im Durchschnitt insgesamt die Bedieneinheit einer Bedienstation beschäftigt [33]. Die Vorstellung der Route der Aufträge durch das System ist nur zur Aufstellung des Modells und zum Aufsummieren der Verweilzeiten zur Gesamtverarbeitungszeit notwendig.

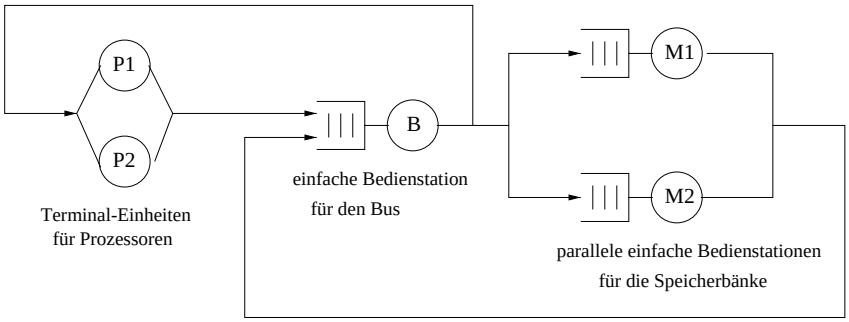


Abbildung 5.6: Ein einfaches schlangentheoretisches Modell eines Bussystems mit zwei Prozessoren und zwei Speicherbänken.

Berechnung der Verweildauer

Die bisherige Betrachtung beschäftigt sich mit der Aufstellung eines Rechnermodells. Dies ist aber nur der erste Schritt der Analyse. Anschließend müssen, ausgehend von den Modellparametern (der Anzahl der Aufträge, Bedienraten, und den Besuchshäufigkeiten) die gesuchten durchschnittlichen Verweildauern berechnet werden. Hierfür gibt es eine Vielzahl von exakten Verfahren sowie Näherungen. Für die Lösung wird ein Programmpaket verwendet, das in [60] beschrieben und dessen Quellcode mit dem Buch mitgeliefert wird. Von einer genauen Erläuterung des Verfahrens wird hier abgesehen. Der interessierte Leser kann sie z.B. in [60] oder in [33] finden.

5.5.3 Basis für den Vergleich mit der Simulation

Für einen Vergleich der theoretischen Analyse mit den Simulationsergebnissen müssen beide mit den gleichen Parametern durchgeführt werden. Außerdem wird eine gemeinsame Meßgröße benötigt, die zum Vergleich der Ergebnisse verwendet werden kann. Um diese Anforderungen zu erfüllen, wird wie folgt vorgegangen:

Parameterabgleich

Die Effizienz des Programms hängt bei den untersuchten Architekturen von zwei Dingen ab: der Latenz und Bandbreite der Komponenten des Speichersystems, sowie der Anzahl und Art von Speicheranfragen der Prozessoren. Die Angleichung der Latenz und der Bandbreite zwischen der Simulation und der theoretische Analyse stellt kein Problem dar. Die Werte sind für jede Komponente eines untersuchten Systems bekannt und müssen lediglich korrekt an die entsprechenden Teile der Simulation übergeben, und bei der Berechnung der Bedienzeiten der entsprechenden Bedienstationen berücksichtigt werden.

Bei der Häufigkeit der Speicherzugriffe ist die Angleichung nicht ganz so einfach. Das Problem besteht darin, daß sich bei der Simulation die Zeitpunkte der Speicheranfragen automatisch durch die Programmausführung ergeben. Ein konkreter Wert für die Häufigkeit der Zugriffe wird dabei nicht verwendet. Ein solcher Wert wird aber bei der theoretische Modellierung benötigt, um die Bedienrate der Prozessor-Stationen zu berechnen. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, aus der Simulation die Häufigkeit der Zugriffe zu berechnen. In Anbetracht der Tatsache, daß die theoretische Modellierung der Überprüfung der Simulation dienen soll, wäre dies jedoch eine methodisch sehr fragwürdige Vorgehensweise. Um das Problem zu umgehen, wird die Häufigkeit der Speicherzugriffe daher aus den Angaben in Tabelle 5.3 ermittelt. Für jedes Benchmarkprogramm ergibt sie sich aus dem Verhältnis der Anzahl der Speicherzugriffe zur Gesamtanzahl der Anweisungen und der Cache-Missrate.

Meßgröße

Wie in Abschnitt 5.4 dargelegt, wird für die Bewertung der Architekturen das Verhältnis der simulierten Ausführungszeit auf diesen Architekturen zu der Ausführungszeit der idealen Speicherarchitektur mit Cache betrachtet. Um die Simulation zu überprüfen, wird dieses Verhältnis aus den Ergebnissen der theoretischen Analyse geschätzt und mit den Resultaten der Simulation verglichen. Die Schätzung basiert auf der Überlegung, daß die Ausführungszeit T_{Prg} aus den Angaben über die Gesamtanzahl der Anweisungen N_A, der Anzahl der Speicherzugriffe N_{Sp} in Tabelle 5.3 und der durchschnittlichen Dauer der Speicheroperationen T_{Sp} bestimmt werden kann.

$$T_{Prg} = (N_{An} \Leftrightarrow N_{Sp}) + N_{Sp} \cdot T_{Sp} \tag{5.1}$$

Für die Ausführungszeit auf dem idealen Rechner wird als T_{SP} einfach die Speicherlatenz genommen. Für die Dauer der Ausführung auf einer bestimmten Architektur wird hingegen die, durch theoretische Analyse ermittelte durchschnittliche Latenz verwendet.

5.5.4 Näherungen und ihre Rechtfertigung

Wie bereits erläutert, stellt eine Warteschlangen-Analyse immer nur eine Näherung dar, da in echten Anwendungen die Speicherzugriffe nicht zufällig verteilt sind. Zusätzlich dazu beinhaltet die in der Arbeit vorgenommene Analyse noch drei weitere Näherungen:

1. Es wird die Tatsache nicht berücksichtigt, daß es unterschiedliche Arten von Speicheranfragen gibt, die zum Teil unterschiedliche Anforderungen an die Komponenten des Speichersystems stellen. So wird z.B. bei Invalidierungen lediglich eine Adresse über den Bus übertragen.

2. Es wird die Belastung des Speichersystems durch Synchronisationsoperationen nicht berücksichtigt.

3. Die Angaben über das Speicherverhalten der Benchmarkprogramme sind nur bis zu 64 Prozessoren veröffentlicht. Für größere Prozessorzahlen werden daher die Werte für 64 Prozessoren benutzt.

5.6 Zusammenfassung der Beiträge

Im vorliegenden Kapitel wurden neben der Erläuterung der Voraussetzung für den Architektur zwei wichtige Beiträge der Arbeit beschrieben: die vom Autor entwickelte Simulationsumgebung und die für das für die Verifikation dieser Umgebung verwendete Modellierungsverfahren.
Die Simulationsumgebung zeichnet sich durch drei Dinge aus:

1. Sie ist modular aufgebaut und einfach erweiterbar, so daß sie leicht für weitere, verwandte Forschung übernommen werden kann

2. Sie benutzt einen Ereignis-gesteuerten Ansatz, der eine genaue Nachbildung des Weges der Nachrichten durch das System erlaubt. Sie kann so nicht nur zur Leistungsbewertung sondern auch zur Überprüfung der Korrektheit der verwendeten Protokolle benutzt werden.

3. Sie verwendet unverändert viele Komponenten, die im Rahmen anderer Arbeiten entwickelt und ausgiebig getestet wurden. Dazu gehören der Simulator-Kern und die Benchmarks. Dadurch wird das Potential wird eine hohe Zuverlässigkeit der Ergebnisse erreicht.

Die theoretische Analyse untermauert die Korrektheit der Simulation. Sie zeichnet sich dadurch aus, daß sie dank geeigneter Näherungen verhältnismäßig einfach ist und trotzdem relevante Aussagen liefert. Außerdem basiert die Analyse auf Daten über das Speicherzugriffsverhalten, die aus der Literatur entnommen und von der Simulation völlig unabhängig sind. Dies garantiert eine eine methodisch saubere Überprüfung der Simulation

Kapitel 6

Technologiestudie

In dem vorliegenden Kapitels werden Wege zur Implementierung der vorgeschlagenen Architekturen aufgezeigt und die Leistungsparameter der Betrachteten Implementierungsansätze abgeschätzt (Abschnitte 6.1, 6.2 und 6.3). Diese Abschätzung dient als Grundlage für die Simulation und die theoretische Analyse in Kapiteln 7, 8 und 9. Darüberhinaus wird ein Konzept für ein modulares opto-elektronisches System 'Plug-and-Play' beschrieben, daß vom Autor in Zusammenarbeit mit Zusammenarbeit mit Wissenschaftlern aus den Bereichen der Optischen Nachrichtentechnik der Fernuniversität Hagen, und der Mikromechanik am IMM Mainz entwickelt wurde Abschnitt (6.4). Das System stellt die Möglichkeit in Aussicht, eine Vielzahl von opto-elektronischen Systemen einfach durch Zusammenstecken von Standardkomponenten aufzubauen. Um die Machbarkeit des Systems zu unterstreichen, wurden nach den Vorgaben des Autors einige einfache Prototypen am IMM-Mainz und bei Bell-Labs hergestellt. Diese werden im Abschnitt 6.5 beschrieben.

6.1 Grundüberlegungen

Bevor man sich mit der Auswahl und der Leistungsbewertung beschäftigt, muß man sich über drei Dinge klar werden. Erstens muß man die benötigten Komponenten genau spezifizieren. Zweitens müssen die Leistungsparameter definiert werden, die man für die gewählte Technologie bestimmen möchte. Schließlich muß man berücksichtigen, daß die Frage nach der Verfügbarkeit und Leistung der Technologie eng mit der Frage der betrachteten technologischen Reife und des betrachteten Zeithorizontes zusammenhängt.

6.1.1 Benötigte Komponenten

Der grobe Aufbau der drei in der Arbeit vorgeschlagenen Architekturen wurde in den Abschnitt 1.3.2 und 5 beschrieben. Dabei wurden zwei Eckpunkte für die Technologie festgelegt:

1. Die optischen Kanäle sollen nicht durch Freiraumoptik sondern mit Fasern bzw. Faserbündeln implementiert werden. Dies ist für die Koppelung von Arbeitsplatzrechnern unerläßlich, die in einem Raum bzw. sogar Gebäude verteilt sein können.

2. Die Netzwerke werden elektronisch auf SP-Bausteinen implementiert, die über die optischen Faserkanäle mit den Prozessoren und Speicherbänken verbunden sind.

Aus der Beschreibung kann man entnehmen, daß alle drei Architekturen aus den folgenden Komponenten bestehen:

1. opto-elektronischen VLSI-Netzwerkbausteinen, auf denen die eigentlichen Netzwerke elektronisch realisiert sind,

2. optischen Faserkanälen für die Punkt-zu-Punkt Kommunikation zwischen einzelnen Prozessoren bzw. Speicherbänken und den Netzwerkbausteinen (P- und M-Kanälen),

3. optischen Broadcast-Kanälen (B-Kanälen), über die die Netzwerkbausteine Nachrichten an das ganze System verschicken können,

4. opto-elektronischen Netzwerkschnittstellen, die die Prozessoren und Speicherbänke an die optischen Kanälen anbinden.

Die Funktionsweise dieser Komponenten und die Anforderungen, die an sie gestellt werden, können wie folgt zusammengefaßt werden:

Netzwerkbausteine

Die opto-elektronischen Netzwerkbausteine beinhalten optische Anschlüsse für die P- und M-Kanäle aller Prozessoren und Speicherbänke, einen Anschluß an den Transmitter des B-Kanals und ein auf dem Chip integriertes elektronisches Netzwerk. Bei der PHOTOBUS und der PHOTON-Architektur handelt es sich bei den Anschlüssen für die P- und M-Kanäle um Eingänge. Es werden also nur optische Empfänger benötigt. Bei der PHOTOBAR-Architektur sind die P- und M-Kanäle bidirektional, so daß sowohl Empfänger als auch Sender auf den Netzwerkbausteinen integriert werden müssen.
Die Netzwerkbausteine müssen in allen drei Architekturen vor allem zwei Anforderungen erfüllen:

1. Die Leistung und die Anzahl der Schaltkreise müssen mit heutigen Hochleistungsprozessoren vergleichbar sein.

2. Die Bausteine müssen je nach Größe des Systems Hunderte bis Tausende optische Ein/Ausgabekanäle besitzen. Diese hohe Kanalanzahl darf die Leistungsfähigkeit der VLSI-Bausteinen nicht beeinträchtigen.

P-und M-Kanäle

Die P-und M- Kanäle dienen der Punkt-zu-Punkt Kommunikation zwischen den Netzwerkbausteinen und einzelnen Prozessoren und Speicherbänken. In allen drei Architekturen haben sie zwei Dinge gemeinsam:

1. Die Bandbreite der P- und M-Kanäle ist für die Systemleistung nicht kritisch. Insbesondere braucht sie nicht mit der Anzahl der Prozessoren zu skalieren. Dies liegt daran, daß die Kanäle jeweils nur von einem Prozessor bzw. Speicherbank benutzt werden.

2. Da jeder Prozessor und jede Speicherbank einen Anschluß an einen P- bzw. M-Kanal hat, ist es wichtig, daß diese Anschlüsse möglich einfach und billig realisierbar sind.

Darüber hinaus gibt es bei der Realisierung der P- und M-Kanäle architekturbedingte Unterschiede. Zum einen werden bei der PHOTOBAR-Architektur bidirektionale Leitungen benötigt. Zum anderen müssen die P-Kanäle bei dem PHOTON-System einen Fanout besitzen, der der Anzahl der Bus-Chips im System gleicht.

B-Kanäle

Die B-Kanäle verlaufen von den Netzwerkbausteinen zu allen Prozessoren und den Speicherbänken und dienen dem Broadcast an das ganze System. Ihre Funktion ist in allen der Architekturen identisch. Sie zeichnen sich dadurch aus, daß

1. die benötigte Bandbreite von der Prozessorzahl abhängt und möglichst hoch sein soll und

2. der Fanout proportional zur Prozessorzahl ist.

Für die Implementierung der B-Kanäle sind zwei Dinge notwendig:

1. Transmitter, die trotz des hohen Fanouts die benötigte Bandbreite liefern,

2. eine Möglichkeit, das Signal des Transmitters auf viele Fasern mit möglichst geringem Aufwand zu verteilen.

Bei den B-Kanälen steht die Leistungsfähigkeit im Vordergrund. Insbesondere beim Transmitter ist der Implementierungsaufwand zweitrangig, da dieser nur einmal bzw. bei der PHOTON-Architektur nur wenige Male im System vorhanden ist.

Schnittstellen zur Elektronik

Die optischen Signale der P, M und B-Kanäle müssen von den Prozessoren und den Speicherbänken in elektrische umgewandelt und an den Cache-Kontroller bzw. Speicherkontroller zur Verarbeitung weitergegeben werden. Die Anforderungen, die von den einzelnen Architekturen an die opto-elektronische Schnittstelle gestellt werden, können wie folgt zusammengefaßt werden:

1. In den PHOTOBUS und PHOTOBAR Architekturen wird jeweils nur ein P- bzw. M-Kanal und ein B-Kanal an einen Prozessor bzw. eine Speicherbank angeschlossen. An die Schnittstelle werden daher keine besonderen Anforderungen gestellt. Sie soll vor allem möglichst einfach und billig realisierbar sein.

2. In der PHOTON-Architektur ist jeder Prozessor an mehrere B-Kanäle angeschlossen. Das heißt, daß er mehrere, hochfrequente Kanäle gleichzeitig überwachen und deren Daten verarbeiten muß. Um mit der großen Datenmenge fertig zu werden, muß jeder Prozessor eine SP-Schnittstelle besitzen, die die optischen Eingänge mit dem Cache-Kontroller vereinigt.

Anbindung der Faser

Um die Signalübertragung zwischen den SP-Bausteinen und den Fasern der P-, M- und B-Kanäle zu ermöglichen, müssen zwei Dinge gewährleistet werden:

1. Die Faserbündel müssen an den SP-Bausteinen befestigt und exakt justiert werden. Dabei wird eine Genauigkeit im Bereich weniger Mikrometer benötigt. Diese Genauigkeit darf nicht durch Umwelteinflüsse wie Termperaturschwankungen und Erschütterungen beeinträchtigt werden.

2. Da man die Faserenden in der Regel nicht direkt auf den Ein/Ausgabefenstern befestigen kann, wird ein optisches System für die Abbildung der Enden der Faser auf die Ein/Ausgabefenster der SPs benötigt. Dieses System muß zum einen die Unterschiede im Abstand und in der Größe zwischen den Fasern und den optischen Ein/Ausgabefenstern kompensieren. Zum anderen muß es in manchen Systemen spezielle Funktionen, wie die Trennung verschiedener Wellenlängen oder die Realisierung eines Fanouts übernehmen.

6.1.2 Technologieparameter

Vom Standpunkt der Rechnerarchitektur sind in Bezug auf einen Parallelrechner zwei Dinge interessant: seine Geschwindigkeit und wie gut er skaliert. Die benötigten Technologieparameter sind also die Bandbreite und Latenz der optischen Kanäle sowie die Anzahl der Prozessoren und Speicherbänke, die an die einzelnen Systemkomponenten angeschlossen werden können. Erstere können aus der Latenz und Bandbreite der einzelnen Kanäle und Schnittstelle ermittelt werden. Letztere hängen von zwei Faktoren ab: Der Anzahl von Kanälen, die an die opto-elektronischen Netzwerkbausteine angeschlossen werden können, und dem maximal realisierbaren Fanout der Broadcast-Kanäle. Für die spätere Betrachtung werden deswegen die folgenden Parameter definiert:

B_P, B_M, B_B: Die Bandbreite der P- (B_P), M-(B_M) und B-Kanäle (B_B). Sie wird entweder in Byte/Prozessorzyklus oder in Byte/s angegeben.

L_P, L_M, L_B: Die Latenz der P- (B_P), M-(B_M) und B-Kanäle (B_B), Angaben erfolgen in Prozessorzyklen oder Nanosekunden.

$Fout_P$, $Fout_B$: Der maximale Fanout des P ($Fout_P$) und des B-Kanals ($Fout_B$)

In_P, In_M **und** In_B: Die maximale Anzahl optischer P-(In_P), M-(In_M) und B In_B- Eingabekanäle, die auf einem SP-Baustein untergebracht werden können.

Out_P, Out_M **und** Out_B: Die maximale Anzahl optischer P-(In_P), M-(In_M) und B In_B- Ausgabegabekanäle, die auf einem SP-Baustein untergebracht werden können. Sie ist mit der Anzahl der Eingabekanäle verbunden. Allerdings muß sie gesondert betrachtet werden, da auf Grund ihrer hohen Wärmedissipation wesentlich weniger Sender als Empfänger auf einem Chip untergebracht werden können.

Die Werte für die obigen Parameter hängen von der für die Implementierung des Systems gewählten Technologie ab. Sie können wie folgt aus den Leistungsparametern der einzelnen Systemkomponenten ermittelt werden:

Die Bandbreite

Die Bandbreite eines Kanals hängt von der Datenrate und der Anzahl von Leitungen des Kanals ab. Dabei ergibt sich die Datenrate aus dem Minimum der maximalen Datenraten der Sender und der Empfänger. Wir definieren daher die folgenden Parameter

B_P^S, B_M^S, B_B^S: Die Bandbreite der Sender der P- (B_P^S), M-(B_M^S) und B-Kanäle (B_B^S).

B_P^E, B_M^E, B_B^E: Die Bandbreite der Empfänger der P- (B_P^E), M-(B_M^E) und B-Kanäle (B_B^E).

N_P, N_M, N_B: Die Anzahl der Leitungen der P- (N_P), M-(N_M) und B-Kanäle (N_B).

Mit diesen Parametern kann die Bandbreite der P- M- und B-Kanäle wie folgt geschrieben werden

$$B_P = min(B_P^S, B_P^E) \cdot N_P \quad (6.1)$$
$$B_M = min(B_M^S, B_M^E) \cdot N_M \quad (6.2)$$
$$B_B = min(B_B^S, B_B^E) \cdot N_B \quad (6.3)$$

Die Latenz

Die Latenz eines Punkt-zu-Punkt Datenkanals kann in fünf Faktoren aufgeteilt werden: die Verzögerung auf dem Weg zum optischen Sender, die Latenz des Senders, die Verzögerung in der Leitung, die Latenz des Empfänger und die Verzögerung auf dem Weg vom optischen Empfänger zur elektrischen Verarbeitungseinheit. Bei einem Broadcast-Kanal muß zusätzlich eine eventuell durch die Verstärkung hervorgerufene Verzögerung hinzugezogen werden. Um diese Faktoren zu erfassen, werden die folgenden Größen definiert:

L_P^{ZS}, L_M^{ZS}, L_B^{ZS}: Die Verzögerung des Signals auf dem weg zum Sender der P- (L_P^{ZS}), M-(L_M^{ZS}) und B-Kanäle (L_B^{ZS}).

L_P^S, L_M^S, L_B^S: Die Latenz der Sender der P- (L_P^S), M-(L_M^S) und B-Kanäle (L_B^S).

L_P^L, L_M^L, L_B^L: Die Leitungsverzögerung der P- (L_P^L), M-(L_M^L) und B-Kanäle (L_B^L).

L_P^E, L_M^E, L_B^E: Die Latenz der Empfänger der P- (L_P^E), M-(L_M^E) und B-Kanäle (L_B^E).

L_B^{VE}: Die Verzögerung des Signals auf dem vom Empfänger des B-Kanals zum Cache-Kontroller. Für die P- und M-Kanäle braucht die Verzögerung auf dem Weg vom Empfänger nicht betrachtet zu werden, da sie immer direkt in den SP-Netzwerkbaustein eingekoppelt werden.

L_P^B, L_B^B: Die Verzögerung, die durch die Verstärkung des Signals bei einem Broadcast auf den P- (L_P^B) und B-Kanälen (L_B^B) ensteht. Die M-Kanäle sind immer Punkt-zu-Punkt, brauchen hier daher nicht berücksichtigt werden.

Die Latenz der einzelnen Kanäle ergibt sich damit als:

$$L_P = L_P^{ZS} + L_P^S + L_P^L + L_P^E + L_P^B \quad (6.4)$$
$$L_M = L_M^{ZS} + L_M^S + L_M^L + L_M^E \quad (6.5)$$
$$L_B = L_B^{ZS} + L_B^S + L_B^L + L_B^E + L_B^{VE} + L_P^B \quad (6.6)$$

Der Fanout

Bei der Betrachtung des Fanouts muß man zwischen zwei Dingen unterscheiden: dem maximal möglichen Fanout eines einzelnen Transmitters, und dem maximalen Fanout, der durch Signalverstärkung und mehrere Fanout-Stufen erreicht werden kann. Letzterer stellt die eigentliche Schranke für den Fanout der P- und B- Kanäle dar.

Der Fanout eines einzelnen Transmitters ist durch die Tatsache beschränkt, daß die Empfänger eine bestimmte minimale optische Leistung des Signals benötigen, um einen korrekten Empfang zu gewährleisten. Diese Leistung hängt von zwei Dingen ab: der Datenrate und der maximal zulässigen Fehlerrate. Der Fanout eines Transmitters ergibt sich damit bei gegebener Daten- und Fehlerrate aus dem Quotienten der Ausgangsleistung des Transmitters

und der minimalen optischen Leistung, die bei der gewünschten Datenrate von den einzelnen Empfängern benötigt wird. Bei hohen Fanouts muß man außerdem die im optischen System auftretenden Verluste in die Betrachtung mit einbeziehen. Zur Bestimmung des maximalen Fanouts eines P- bzw. B-Kanal Transmitters werden die folgenden Parameter benötigt:

P_P^S, P_B^S: Die maximale optische Ausgangsleistung der Sender der P- (P_P^S) und B-Kanäle (P_B^S).

P_P^E, P_B^E: Die minimale optische Leistung, die von den Empfängern der P- (P_P^E) und B-Kanäle (P_B^E) benötigt wird. Sie wird bei der üblicherweise in der Nachrichtentechnik verwendeten Fehlerrate von 10^{-14} und der zum Erreichen der Bandbreiten B_P und B_B notwendigen Datenraten betrachtet.

V_P, V_B: Der Anteil der Senderleistung, der im optischen System verloren geht; V_P für die P- und V_B für die B-Kanäle).

Damit ist der Fanout eines einzelnen Transmitters $Fout_P^1$ und $Fout_B^1$ durch die folgende Gleichung gegeben:

$$Fout_P = \frac{P_P^S \cdot V_P}{P_P^E} \tag{6.7}$$

$$Fout_B = \frac{P_B^S \cdot V_B}{P_B^E} \tag{6.8}$$

In einem System mit Verstärkung wird nach dem Fanout jedes einzelne Signal in einen Verstärker geleitet. Danach wird das verstärkte Signal wieder aufgeteilt und dann entweder zu den Empfängern oder zu den Verstärkern der nächsten Stufe geleitet. Das System ist also wie ein Baum aufgebaut, wobei der B- bzw. P-Kanal Sender die Wurzel, und die Empfänger die Blätter bilden. Die Knoten bestehen jeweils aus einem Verstärker und dem nachfolgenden Fanout.

Der Fanout, der durch die Verstärkung erreicht werden kann, hängt also von drei Faktoren ab: der Anzahl der Verstärkerstufen, der minimalen Stärke des Eingangssignals der Verstärker jeder Stufe und der Ausgangsleistung der Verstärker jeder Stufe. Zu Berechnung des Fanouts werden bei einem System mit Verstärkung daher die folgenden Parameter benötigt:

S_P, S_B: Die Anzahl der Verstärkerstufen der P- (S_P) und B-Kanäle (S_B).

$P_S^{S,i}$, $P_B^{S,i}$: Die maximale optische Leistung, die am Ausgang der Stufe i der P- ($P_P^{S,i}$) und B-Kanäle ($P_B^{S,i}$) zur Verfügung steht,

$P_S^{E,i}$, $P_B^{E,I}$: Die minimale optische Leistung, die am Eingang der Stufe i der P- ($P_P^{E,i}$) und B-Kanäle ($P_B^{E,i}$) an jeder Leitung benötigt wird,

V_P^i, V_B^i: Der Anteil der Senderleistung, der im optischen System der Stufe i verloren geht (V_P^i für die P- und V_B^i für die B-Kanäle).

Im Nachfolgenden bezeichnen wir die Transmitter als Ausgang der Stufe 0 und die Empfänger als Eingang der Stufe $S_P + 1$ bzw. $S_B + 1$. Damit kann man den Fanout eines P- bzw. B-Kanals mit S_P bzw. S_B-Stufen wie folgt schreiben:

$$Fout_P = \prod_{i=0}^{S_P} \left(\frac{P_P^{S,i} \cdot V_P^i}{P_P^{E,i+1}} \right), \quad P_P^{S,0} = P_P^S, \quad V_P^0 = V_P, \quad P_P^{E,S_P+1} = P_P^E \tag{6.9}$$

$$Fout_B = \prod_{i=0}^{S_B} \left(\frac{P_B^{S,i} \cdot V_B^i}{P_B^{E,i+1}} \right), \quad P_B^{S,0} = P_B^S, \quad V_B^0 = V_B, \quad P_B^{E,S_P+1} = P_B^E \tag{6.10}$$

Für Systeme ohne Verstärkung ($S_P = 0$ bzw. $S_B = 0$) gehen obige Gleichungen in Gleichungen (6.7) bzw. (6.8) über.

Die Anzahl der Ein/Augabekanäle von SP-Bausteinen

Die Anzahl der optischen Datenkanäle eines SP-Bausteins hängt von zwei Faktoren ab: der Anzahl einzelner Kanäle (Leitungen) in einem Datenkanal und der maximalen Anzahl solcher einzelnen optischen Kanäle, die an dem Baustein angekoppelt werden können. Die Anzahl der Leitungen in einem Datenkanal wurde bei der Betrachtung der Bandbreite der P-, M-, und B-Kanäle als N_P, N_M und N_B definiert.

Die Anzahl der einzelnen optischen Kanäle ist durch drei Parameter bestimmt: die Anzahl von Empfängern bzw. Sendern auf dem SP-Baustein, die Anzahl der Faser, die an den Baustein angekoppelt werden können, sowie die Anzahl der Kanäle, die das optische System von den Fasern auf die Empfänger bzw. Sender abbilden kann. Hinzu kommt die Frage nach der Anzahl der Wellenlängen, die im System benutzt werden können. Von den obigen Faktoren ist lediglich der erste für Ein- und Ausgabefenster unterschiedlich. Dies liegt an der Wärmedissipation, die bei Sendern in der Regel wesentlich höher als bei Empfängern ist. Um die maximale Anzahl der Ein/Augabekanäle von SP-Bausteinen zu bestimmen werden daher folgende Parameter benötigt:

N_{SP}^{in}, N_{SP}^{out}: Die maximale Anzahl optischer Ein- (N_{SP}^{in}) und Ausgabefenster (N_{SP}^{out}), die auf einem SP-Baustein untergebracht werden kann,

N_{mech}: Die maximale Anzahl von Fasern, die auf einem SP-Baustein befestigt werden kann,

N_{WDM}: Die maximale Anzahl von Wellenlängen, die in einer Faser übertragen, und aus dieser Faser auf die Ein/Ausgabefenster des SP-Bausteins abgebildet werden kann,

N_{opt}: Die maximale Anzahl von Kanälen, die auf die Ein/Ausgabefenster des SP-Bausteins abgebildet werden kann.

Die Betrachtung von Eingabe- und Ausgabekanälen unterscheidet sich also nur darin, daß für die Eingabefenster N_{SP}^{in} und für die Ausgabefenster N_{SP}^{out} betrachtet werden muß. Die Anzahl einzelner Kanäle ist damit in beiden Fällen durch das Minimum aus N_{SP}^{in} bzw. N_{SP}^{out} und der Anzahl der Kanäle gegeben, die an dem Baustein befestigt sind und auf die Ein/Ausgabefenster abgebildet werden können. Letzeres ergibt sich als Minimum von N_{opt} und dem Produkt aus N_{mech} und N_{WDM}. Unter der Annahme, daß die P- und M-Kanäle genauso viele Leitungen besitzen, ergibt sich damit für die Anzahl der Ein- und Ausgabekanäle:

$$In_P = In_M = \frac{min\left(N_{SP}^{in}, min\left(N_{opt}, N_{mech} \cdot N_{WDM}\right)\right)}{N_P} \tag{6.11}$$

$$In_B = \frac{min\left(N_{SP}^{in}, min\left(N_{opt}, N_{mech} \cdot N_{WDM}\right)\right)}{N_B} \tag{6.12}$$

$$Out_P = Out_M = \frac{min\left(N_{SP}^{out}, min\left(N_{opt}, N_{mech} \cdot N_{WDM}\right)\right)}{N_P} \tag{6.13}$$

$$Out_B = \frac{min\left(N_{SP}^{out}, min\left(N_{opt}, N_{mech} \cdot N_{WDM}\right)\right)}{N_B} \tag{6.14}$$

Die obigen Gleichungen gelten jeweils für den Fall, daß sich auf einem Baustein nur eine Art von optischen Datenkanälen befindet. Sie dürfen nicht unabhängig voneinander betrachtet werden, wenn mehrere Arten von Datenkanälen gleichzeitig auf einem Baustein untergebracht werden sollen. In diesem Fall gilt, daß die Summe der einzelnen optischen Kanäle aller Datenkanal-Arten ein bestimmtes Maximum Max_{IO} nicht übersteigen darf. Dieses Maximum gleicht der maximalen Anzahl optischer Kanäle von der Sorte, von der am meisten auf dem Chip untergebracht werden können. Wir bezeichnen die maximalen Anzahlen von Datenkanälen, die auf einem Chip mit mehreren Arten optischer Ein/Ausgabekanäle untergebracht werden können, mit In_P^{mix}, In_M^{mix}, Out_P^{mix}, Out_M^{mix}, In_B^{mix} und Out_B^{mix}. Damit gilt:

$$Max_{IO} \geq \left(In_P^{mix} + In_M^{mix} + Out_P^{mix} + Out_M^{mix}\right) \cdot N_P + \left(In_B^{mix} + Out_B^{mix}\right) \cdot N_B \tag{6.15}$$

$$Max_{IO} = max\left(In_P \cdot N_P, Out_P \cdot N_P, In_B \cdot N_B, Out_B \cdot N_B\right) \tag{6.16}$$

6.1.3 Betrachtete Technologiereife und Zeitrahmen

Die Frage nach der technologischen Machbarkeit eines Systems macht nur Sinn, wenn gleichzeitig ein Zeitrahmen und der benötigte Reifegrad der Technologie vorgegeben werden. So gibt es einen enormen Unterschied zwischen

Systemen, die sofort aus kommerziellen Komponenten aufgebaut werden können, und solchen, die ein Stab von Wissenschaftlern in einem großen Forschungszentrum innerhalb einiger Jahre realisieren kann. Bei der nachfolgenden Betrachtung werden im Hinblick auf den Zeitrahmen und den Entwicklungsstand der Technologie drei Stufen unterschieden:

1. **Weitgehend ausgereifte, sofort verfügbare Technologie:** Als weitgehend ausgereift und sofort verfügbar werden drei Arten von Komponenten eingestuft: handelsübliche Standardkomponenten, Komponenten, die als Sonderanfertigungen bestellt werden können sowie Bauteile, die als Labormuster von Forschungsanstalten angeboten werden. Sie zeichnen sich alle dadurch aus, daß sie mit der benötigten Leistung und Robustheit kommerziell erhältlich sind.

2. **Prototypische, kurzfristig verfügbare Technologie:** Als prototypisch und kurzfristig verfügbar wird die Technologie eingestuft, die mit der benötigten Robustheit und Komplexität im Labor demonstriert wurde, zur Zeit jedoch noch nicht kommerziell erhältlich ist.

3. **Experimentelle, mittelfristig verfügbare Technologie:** Als experimentell und mittelfristig verfügbar werden Komponenten bezeichnet, die zwar als einfache Prototypen bereits demonstriert wurden, bei denen aber noch weitere Entwicklungsarbeit notwendig ist, bevor die für die praktische Anwendung benötigte Leistung und Robustheit erreicht werden.

6.2 Technologiewahl

Der vorliegende Abschnitt beschäftigt sich mit der Frage, wie die drei in der Arbeit untersuchten Architekturen am besten in jeder der im vorigen Abschnitt definierten Technologieklassen implementiert werden können.

Im Nachfolgenden wird zunächst für jede der benötigten Komponenten die Implementierungsmöglichkeit aufgezeigt. Abschließend werden die Implementierungsmöglichkeiten für jede Architektur zusammengefaßt.

6.2.1 Netzwerkbausteine und Schnittstellen

Die opto-elektronischen Netzwerkbausteine müssen eine komplexe Hochleistungs-VLSI Schaltung mit einer großen Anzahl hochfrequenter optischer IOs verbinden. Hierfür bietet sich die 'Flip-Chip' Verbindung konventioneller CMOS-VLSI Chips mit entsprechenden opto-elektronischen Chips an. Das Verfahren wurde ausführlich in Abschnitt 4.5.4 beschrieben. Es ist sowohl für heutige als auch für kurz- und mittelfristig verfügbare Systeme geeignet. Die Unterschiede zwischen der sofort und der künftig verfügbaren Technologie liegen in der Art und Anzahl der optischen IOs.

Sofort verfügbare Lösung

Zur Zeit sind SP-Bausteine lediglich mit MQW-Dioden (Abschnitt 4.5.3) kommerziell verfügbar. Obwohl die MQW-Dioden selbst sowohl als Empfänger als auch als Modulator-Sender verwendet werden können, ist die Nutzung MQW-basierter SPs als Sender mit Hilfe kommerzieller Technologie nur schwer möglich. Dies liegt daran, daß die Modulatoren zum Senden mit einer externen Lichtquelle ausgeleuchtet werden müssen. Gleichzeitig muß dafür gesorgt werden, daß das von den Modulatoren reflektierte und modulierte Licht zu den Ausgängen gelangt. Dies erfordert ein komplexes optisches System, das für große Felder schwer zu realisieren ist. Wir gehen daher davon aus, daß mit der sofort verfügbaren, ausgereiften Technologie nur SPs mit Empfängern möglich sind.

Kurzfristig verfügbare Lösung

Wie in 4.5.5beschrieben, wurde in verschiedenen Labors das 'Flip-Chip' Bonden von Feldern von VCSEL-Laserdioden und Photodioden auf CMOS VLSI Chips demonstriert. Solche Bausteine werden für die PHOTOBAR-Architektur benötigt, damit die Netzwerkbausteine auf den P-und M-Kanälen nicht nur empfangen, sondern auch senden können. Außerdem würden sie eine direkte Integration des B-Kanal Senders auf dem Netwerkbaustein ermöglichen.

Mittelfristig verfügbare Lösungen

Auf längere Sicht sind vor allem SPs interessant, auf denen Laserdioden verschiedener Wellenlänge angebracht sind. Solche Bausteine können bitparallele Sender realisieren, die jedes Bit auf einer anderen Wellenlänge senden, und somit alle Bits in einer Faser übertragen können. Dadurch kann die Anzahl von Fasern im System stark reduziert werden. Dies ist insbesondere für große PHOTON-System wichtig (siehe 6.2.7).

6.2.2 P- und M-Kanäle

Die P- und M-Kanäle sind die einfachsten Komponenten der hier betrachteten Architekturen. Mit Hilfe heute verfügbarer Technologie können sie am besten durch kommerzielle parallele Fasersysteme implementiert werden (4.7). Kurzfristig wird es möglich sein, die Sender der P- und M-Kanäle auf SP-Chips zu integrieren. Dadurch können die Signallatenz und die Komplexität des Systems reduziert werden. Mittelfristig sind vor allem Sender wichtig, die bitparallel auf mehreren Wellenlängen senden können. Solche Sender können die Anzahl der Faser im System, und damit die Systemkomplexität erheblich reduzieren.

6.2.3 B-Kanäle: Sender

Die B-Kanäle zeichnen sich durch eine hohe Bandbreite und einen hohen Fanout aus. Das größte Problem bei ihrer Implementierung stellt die Ausgangsleistung des Senders dar, die linear mit der Anzahl des Fanouts wachsen muß. Dies ist in allen Technologiestufen nur mit Hilfe von Verstärkern zu erreichen. Je nach Anzahl der Prozessor wird sogar eine mehrstufige Verstärkung benötigt. Die Unterschiede zwischen den Technologieklassen liegen dabei darin, wie einfach und effizient diese Verstärkerstufe realisiert werden kann.

Sofort verfügbar

Um mit heute verfügbarer Technologie die benötigte hohe Bandbreite zu realisieren, sind parallele Fasersysteme unabdingbar. Um für solche Systeme eine Verstärkung zu implementieren, gibt es zwei Möglichkeiten: opto-elektronische Verstärkung und die Verwendung von Faserverstärkern (siehe 4.2.4).

Opto-Elektronische Verstärkung: Bei der opto-elektronischen Verstärkung besitzt jede Verstärkerstufe einen parallelen Faserempfänger und -sender. Die optischen Signale von der vorhergehenden Stufe werden von den Empfängern in elektrische Signale umgewandelt, an den Sender weitergegeben (elektrisch) und von dort an die nächste Stufe optisch weitergesendet. Dadurch wird nach jeder Fanoutstufe die ursprüngliche Signalstärke wiederhergestellt. Der Nachteil dieser Methode besteht darin, daß durch den mehrmaligen Übergang zwischen optischem und elektrischem Signal die Latenz des Systems vergrößert wird.

Faserverstärker: Bei der Verwendung von Faserverstärkern wird der Ausgang des Senders nach einer ersten Fanout-Stufe an einen handelsüblichen optischen Faserverstärker angeschlossen. Da solche Verstärker heute nur für Einzelfaser erhältlich sind, müssen die parallelen Faserbündel vorher in Einzelfaser aufgeteilt und mit passenden Steckern versehen werden. Die Ausgänge der Faserverstärker gehen zum weiteren Fanout zu Faserverteilern und von dort entweder zu den Empfängern oder zur nächsten Verstärkungsstufe. Diese Methode hat den Vorteil, daß durch die Verstärkung die Signallatenz nicht vergrößert wird. Ihr Nachteil besteht darin, daß für jedes Bit ein teurer Verstärker benötigt wird.

Kurzfristig

Kurzfristig sind für die Realisierung der B-Kanäle vor allem SP-Bausteine mit einer hohen Zahl von Laserdioden interessant. Sie erlauben eine hohe Sendeleistung durch die Verwendung mehrerer Transmitter, die gleichzeitig die gleichen Daten senden. Dies ist möglich, da auf einem SP-Baustein Hunderte oder gar Tausende Laserdioden untergebracht werden können, während der Sender lediglich 8 bis 64 Bit breit ist. Damit können auf dem Chip viele Kopien des Transmitters untergebracht werden, die alle durch die VLSI-Schaltung mit Daten versorgt werden.

Durch die Verwendung mehrerer Transmitterkopien ist es zum einen möglich, einen beträchtlichen Fanout ohne weitere Verstärkung zu erreichen. Dies ist vor allem für die P-Kanäle der PHOTON-Architektur interessant. Für die B-Kanäle spielt diese Möglichkeit deswegen keine Rolle, weil auf den Netzwerkbausteinen nicht genug Platz für mehr als einen Transmitter vorhanden ist. Statt dessen werden spezielle Broadcast-SP-Bausteine benötigt,

die die Funktion einer opto-elektronischen Verstärkerstufe übernehmen. Sie bestehen aus einem Empfänger und vielen Transmitterkopien. Der Empfänger empfängt das Signal eines auf dem Netzwerkbaustein integrierten Tansmitters und leitet es elektronisch an die Transmitterkopien weiter, die es optisch weitersenden. Diese Art opto-elektronischer Verstärkung hat den Vorteil, daß sie nur eine geringe Latenz mit sich bringt. Dies liegt daran, daß der Empfänger und die Transmitter auf dem gleichen SP-Chip integriert sind. Ihre Latenz sowie die Verzögerung durch die elektronische Datenübertragung liegen im Bereich eines Prozessorzyklus.

Mittelfristig

Mittelfristig wird die Verwendung von Faserverstärkern wieder attraktiv. Dies liegt einerseits daran, daß durch das WDM-Verfahren alle Bits in einer Faser übertragen werden können. Da ein Faserverstärker mehrere Wellenlängen gleichzeitig verstärken kann, wird so für den ganzen Kanal nur ein Verstärker benötigt. Hinzu kommt, daß mittelfristig integrierte optische Module verfügbar werden, die die Verteiler und mehrere Halbleiter-Faserverstärker vereinigen. Somit kann die ganze Sender- und Verstärkerstufe, inklusive des Fanouts in einem kompakten Baustein untergebracht werden. Eine solche Senderstufe würde wie der in Abbildung 4.26 im Kapitel4 dargestellte Baustein aussehen. Hinzu kommt, daß Halbleiterverstärker wesentlich billiger als konventionelle Faserverstärker sind, so daß auch mehrere Verstärkerstufen benutzt werden können, ohne die Systemkosten übermäßig in die Höhe zu treiben.

6.2.4 B-Kanäle: Fanout

Wie in Kapitel 4 erläutert, sind zur Zeit Faserverteiler mit einem Fanout von bis zu 64 kommerziell verfügbar. Durch ein Hintereinanderschalten mehrere Schichten solcher Verteiler läßt sich theoretisch ein beliebig hoher Fanout erreichen. Das Problem besteht dabei darin, daß die heutigen Verteiler für Einzelfaser ausgelegt sind. D.h., daß die parallelen Faserbündel aufgespalten und einzeln in die Verteiler eingesteckt werden müssen. Bei einem hohen Fanout und einer großen Datenbreite des Kanals führt dies zu einem schwer handhabbarem 'Kabelsalat'. Soll z.B. ein 64 Bit breiter Kanal auf 512 Prozessoren verteilt werden, so hat man es mit 32768 einzelnen Fasern und Steckern zu tun. Das Problem kann kurzfristig durch, mit parallelen Fasersteckern versehenen, 1xF Verteilern gelöst werden. Sie vereinfachen die Realisierung des Fanouts, da nun nicht mehr für jede einzelne Faser ein eigenständiger Verteiler benötigt wird. Sie können auf zwei Arten implementiert werden: durch entsprechend konfektionierte, fortgeschrittene integrierte optische Komponenten (siehe 4.7), oder mit Hilfe planar integrierter optischer Freiraumsysteme (4.6.3). Im letzten Fall werden die Ein- und Ausgabe-Faserbündel auf einer Platte befestigt, in der eine Fanout-Optik realisiert ist. Mittelfristig wird, wie im vorigen Abschnitt angesprochen, die Faseranzahl durch das Wellenlängenmultiplexing und die Verwendung von Verteilern mit integrierten Verstärkern reduziert werden.

6.2.5 Schnittstellen zwischen der Optik und der Elektronik

Diese Schnittstelle zwischen der Optik und der Elektronik kann auf zwei Arten realisiert werden: durch konventionelle elektronische Faserempfänger und Treiberbausteine oder durch die Integration der optischen Ein-/Ausgabefenster und der Elektronik auf einem SP-Baustein. Im ersten Fall werden die Daten von dem optischen Empfänger zu den Treiberbausteinen, und von dort zu dem Cache bzw. Speicherkontroller übertragen. Dabei findet die elektrische Datenübertragung über normale Leitungen auf einer Platine statt. Dies ist zur Zeit Stand der Technik.

Im zweiten Fall sind die Cache bzw. Speicherkontroller mit Hilfe von SP-Bausteinen realisiert, die direkt an die optischen Kanäle angeschlossen sind. Dies erlaubt eine höhere Bandbreite und reduziert die Latenz, da keine elektrische Datenübertragung auf einer Platine notwendig ist.

6.2.6 Anbindung der Faser an die SP-Bausteine

Die Anbindung der Faser an die SP-Bausteine stellt eines der schwierigsten Probleme bei der Implementierung der in der Arbeit vorgeschlagenen Architekturen dar. Je nach Größe des Systems müssen zwischen einigen Hundert und einigen Tausend Kanälen in Relation zu den optischen Fenstern des Chips exakt positioniert und stabil befestigt werden.

Sofort verfügbare Lösung

Zur Zeit gibt es zwei Möglichkeiten, die Faserbündel mit den SP-Chips zu verbinden: makroskopische mechanische Aufbauten und die direkte Befestigung einzelner Faser an den optischen Ein-/Ausgabefenstern. In beiden Fällen werden die Faser zunächst mit Hilfe einer speziell gefertigten Lochplatte zu einem 2D-Bündel zusammengefaßt (siehe Abschnitt 4.6.2). Dieses 2D-Bündel kann dann, wie in 4.6.3 beschrieben, zusammen mit dem SP-Chip und einer geeigneten Optik in einem kompakten mechanischen Aufbau integriert werden. Alternativ ist es möglich, ein solches 2D-Bündel durch eine geeignete Mikromechanik über der Oberfläche des SP-Chips zu befestigen (siehe 4.6.3). Beide Möglichkeiten haben den Nachteil, daß die Herstellung des 2D-Bündels aufwendig und teuer ist. Der große Aufwand für die Implementierung ist auch der Hauptnachteil der makromechanischen Varianten. Das Problem der direkten Befestigung am SP-Baustein besteht darin, daß sie keine bzw. nur eine sehr einfache optische Abbildung zwischen den Fasern und den optischen Ein-/Ausgabefenstern erlaubt. Dies bedeutet zum einen, daß die Faser in dem 2D-Bündel in genau der gleichen Geometrie wie die optischen Fenster auf dem SP-Chip angeordnet sein müssen. So ist es nicht möglich, die Herstellung des 2D-Bündels z.B. dadurch zu vereinfachen, daß man mehrere 1D-Stecker anstelle einzelner Faser an der Lochplatte befestigt. Zum anderen können weder die für die Nutzung von Modulatoren als Sender notwendige Ausleuchtung, noch ein Wellenlängenmultiplexing durchgeführt werden.

Kurzfristig

Kurzfristig wird es möglich sein, mehrere 1-D Faserstecker mit Hilfe planar integrierter Freiraumoptik an einem SP-Chip zu befestigen. Das Verfahren wird nachfolgend in Abschnitt 6.4 nochmals genauer erläutert. Die Grundidee besteht darin, zunächst mit Hilfe einer geeigneten Lochplatte mehrere parallele Faserstecker zu einem 'Pseudo-' 2D-Bündel zusammenzufassen. Ein solches Bündel wird dann auf einer mikrosopischen planar integrierten Frairaumoptik befestigt. Dies kann einfach durch Aufkleben der Lochplatte auf der Oberfläche der planaren Optik erfolgen. An der gleichen planar integrierten Optik wird auch der SP-Baustein befestigt. Die Optik bildet jede Faser des Bündels auf das zugehörige Ein-/Ausgabefenster des SP-Bausteins ab.

Mittelfristig

Mittelfristig ist vor allem die Steigerung der Anzahl von Kanälen zu erwarten, die an einem SP-Chip befestigt werden können. Sie wird auf zwei Faktoren zurückzuführen sein. Zum einen werden mehr Faser an der planaren Optik befestigt werden können. Zum anderen werden durch die Anwendung des Wellenlängenmultiplexings in jeder Faser mehrere Kanäle übertragen.

6.2.7 Zusammenfassung

Die wichtigsten Unterschiede zwischen den Implementierungsmöglichkeiten der Technologieklasssen können wie folgt zusammengefaßt werden:

Sofort verfügbar: parallele Faserbündel ohne WDM für die Datenübertragung, kommerzielle parallele Fasersender und -Empfänger an den Prozessoren und Speicherschnittstellen, VLSI-Chips, die auf dem MQW-Dioden-Empfänger durch 'Flip-Chip' Bonding befestigt sind als Netzbaustein, und ein kompaktes makromechanisches System zur Anbindung der Faser an den SP-Chip.

Kurzfristig verfügbar: parallele Faserbündel ohne WDM für die Datenübertragung, VLSI-Chips mit durch 'Flip-Chip' Bonding integrierten Feldern von Laserdioden und Photodioden als Prozessorschnittstellen, Speicherschnittstellen und Netzwerkbausteine, und einfache planar integrierte Freiraumoptik zur Anbindung der Faser an den SP-Chip.

Mittelfristig verfügbar: parallele Faserbünbdel mit 8 bis 32 Wellenlängen für die Datenübertragung, VLSI-Chips mit durch 'Flip-Chip' Bonding integrierten Feldern von Laserdioden unterschiedlicher Wellenlänge, und Photodioden als Prozessorschnittstellen, Speicherschnittstellen und Netzwerkbausteine und komplexes planar integriertes Freiraumsystem mit Wellenlängenmultiplexing für die Anbindung der Faser an die SP-Chips, integrierte optische Bausteine mit Sendern verschiedener Wellenlänge, Verzweigern und optischen Verstärkern für den B-Kanal.

Bezogen auf die einzelnen Architekturen bedeutet dies die folgenden Implementierungsmöglichkeiten:

	sofort	kurzfristig	mittelfristig
Bus-Chip	MQW-dioden durch Flip-Chip Bonding mit CMOS-VLSI Bausteinen verbunden		
B-Kanal Fanout	kommerzielle Faserverteiler für Einzelfaser	Faserverteiler für Faserbündel und planar integrierte Freiraumoptik	integrierte optische Bausteine und/oder integrierte Freiraumoptik
B-Kanal Sender	parallele Faserssysteme mit Signalverstärkung durch Wiederholung oder Faserverstärker	SP-Bausteinen mit Laserdiodenfeldern	integrierte optische Bausteine mit WDM-Sendern und Halbleiterverstärkern
P und M-Kanal	parallele Fasersysteme	parallele Fasersysteme und SP-Bausteine	optische Ein/Aussgabe direkt vom Prozessorchip
Faseranbindung	makroskopische Mechanik	planar integrierte Freiraumoptik und MT-Stecker	planar integrierte Freirraumoptik mit WDM und MT-Steckern

Tabelle 6.1: Die Tabelle faßt die Implementierungsmöglichkeiten für die PHOTOBUS-Architektur in den verschiedenen Technologieklassen zusammen.

PHOTOBUS

Das besondere an der PHOTOBUS-Architektur ist, daß ein funktionsfähiges System aus sofort verfügbaren Komponenten aufgebaut werden kann (Tabelle 6.1). In einem solchen System bekommen jeder Prozessor und jede Speicherbank einen kommerziell parallelen Fasersender und -empfänger (z.B. das PAROLI-System von Siemens). Sowohl die Sender als auch die Empfänger sind über elektrische Leitungen an konventionelle Schnittstellen angeschlossen. Alle Sender sind über parallele Faserbündel mit einer kompakten makroskopischen Optik verbunden. Die Optik bildet die einzelnen Faser auf die entsprechenden Eingänge des Netzwerkbausteins ab. Der Netzwerkbaustein ist ein CMOS-VLSI-Chip auf dem MQW-Dioden-Empfänger durch 'Flip-Chip' Bonding befestigt sind. Er ist elektronisch mit dem Sender des B-Kanals verbunden, der wie die P- und M-Kanal-Sender mit Hilfe eines kommerziellen parallelen Fasersenders realisiert ist. Um das Fanout zu realisieren, wird der Ausgang des Faserbündels des B-Kanal-Senders in einzelne Faser aufgeteilt, die an kommerzielle 1xn Verzweiger angeschlossen werden. Die Faser an den Ausgängen der Verzweiger werden zur Verstärkung an kommerzielle Faserverstärker angeschlossen, dessen Ausgänge wieder zu Faserbündel zusammengefügt und mit den Empfängern der Prozessoren und Speicherbänke verbunden werden. Alternativ können die Ausgänge der Verzweiger zwecks mehrstufigen Fanouts an parallele Empfänger-/Sender-Kombinationen angeschlossen werden. Kurzfristig ist zu erwarten, daß die Implementierung der PHOTOBUS-Architekturen durch eine breitere Anwendung der SP-Bausteine vereinfacht und ihre Leistung gesteigert werden kann. Diese Anwendung wird durch zwei Entwicklungen möglich: die sinkenden Kosten solcher Bausteine und die einfachere Anbindung der Faser mit Hilfe planar integrierter optischer Systeme. Besonderes wichtig ist die Möglichkeit, den Fanout und die Verstärkung durch einen SP-Broadcast-Baustein und einer planaren Freiraumoptik zu verwirklichen. Hinzu kommt die Integration aller optischen Kanäle auf SP-Bausteinen. Beides führt zu einem kompakten Systemaufbau und einer geringeren Signallatenz. Mittelfristig ist vor allem die Vereinfachung des B-Kanal Senders und Fanouts durch integrierte optische Systeme von Bedeutung.

PHOTOBAR

Zur Zeit sind weder die Integration von Laserdioden auf VLSI-Bausteinen noch die für die Ausleuchtung von Modulator-Feldern notwendigen optischen Systeme ausgereift und verfügbar. Daher kann ein PHOTOBAR-System nicht aus sofort verfügbaren Komponenten aufgebaut werden. Die Realisierung mit Hilfe prototypischer Technologie ist in Tabelle 6.2 skizziert. Jeder Prozessor und jede Speicherbank besitzen einen parallelen Fasersender und -empfänger für den P- bzw. M-Kanal sowie einen parallelen Faserempfänger für den B-Kanal. Diese können sowohl als kommerzielle, über elektronische Schnittstellen angeschlossene Komponenten als auch in Form von SP-Bausteinen realisiert werden. Im letzeren Fall würden sich alle drei Anschlüsse auf dem Cache-Kontroller bzw. Prozessor-Chip befinden. Die Faserbündel der P- und M-Kanäle sind an einer planar integrierten Freiraumoptik befestigt, die sie auf die Ein-/Ausgabefenster des Netzwerkbausteins abbildet. Der Netzwerkbaustein ist ein CMOS-VLSI-Chip, auf dem ein Feld von Laserdioden und Empfängern durch 'Flip-Chip' Bonding befestigt ist. Damit kann auch der Sender des B-Kanals auf dem Netzwerkbaustein untergebracht werden. Für die Implementierung des Fanouts werden mehrere, auf einem SP-Chip integrierte Laserdioden-Sender und eine planar

	kurzfristig	mittelfristig
Netz-Chip	Laser- und Photodioden durch Flip-Chip Bonding auf CMOS integriert	Laserdioden unterschiedlicher Wellenlänge und Photodioden durch Flip-Chip Bonding auf CMOS integriert
B-Kanal Fanout	Faserverteiler für Faserbündel und planar integrierte Freiraumoptik	integrierte Optik
B-Kanal Sender	SP-Bausteinen mit Laserdiodenarrays	integrierte optische Bausteine mit WDM-Sendern, Halbleiterverstärkern
P und M-Kanal	parallele Fasersysteme und SP-Bausteine	optische Ein/Aussgabe direkt vom Prozessorchip
Faseranbindung	planar integrierte Freiraumoptik und MT-Stecker	planar integrierte Freirraumoptik mit WDM und MT-Steckern

Tabelle 6.2: Die Tabelle faßt die Implementierungsmöglichkeiten für die PHOTOBAR-Architektur in den verschiedenen Technologieklassen zusammen.

integrierte Freiraumoptik verwendet. Je nach Systemgröße können diese Sender direkt auf dem Netzwerkchip oder auf einem separaten SP-Baustein (Broadcast-Baustein) untergebracht werden. Im letzten Fall würde man auf dem Netzwerkchip trotzdem einen Sender für den B-Kanal unterbringen, um so den Netzwerkchip mit dem Broadcast-Baustein effizient optisch zu verbinden. Mittelfristig außerdem wie auch bei der PHOTOBUS-Architektur die Vereinfachung des B-Kanals durch die nutzung integrierte opto-elektronischer Komponenten zu erwarten.

Langfristig ist vor allem zu erwarten, daß durch die Verwendung mehrerer Wellenlängen mehr Kanäle an den Netzwerkchip angeschlossen werden können. Dies liegt daran, daß mehrere Bits in einer einzelnen Faser auf unterschiedlichen Wellenlängen übertragen werden. So werden pro Kanal weniger Fasern benötigt, es können also mehr Kanäle mit der gleichen Faseranzahl realisiert werden.

PHOTON

Die PHOTON-Architektur kann theoretisch aus sofort verfügbaren Komponenten aufgebaut werden. Dabei muß man allerdings bedenken, daß sie für jeden Prozessor einen SP-Baustein samt den für die Anbindung der Faser notwendigen optischen und mechanischen System benötigt wird. Insbesondere das optische System ist mit der heute verfügbaren Technologie nur mit einem großen Aufwand realisierbar, da es einen komplexen makroskopischen, opto-mechanischen Aufbau erfordert. Bedenkt man, daß die PHOTON-Architektur für große Prozessorzahlen gedacht ist, dann erscheint aufgrund dieses hohen Aufwandes für jeden Prozessor eine Implementierung mit kommerzieller Technologie wenig sinnvoll. Wir beschränken uns hier deswegen auf die in Tabelle 6.1 zusammengefaßte Realisierung mit Hilfe prototypischer Technologie.

In einem PHOTON-System besitzt jede Speicherbank einen parallelen Fasersender für den M-Kanal und einen Empfänger für den ihr zugeordneten B-Kanal. Jeder Prozessor besitzt einen SP-Baustein, an den der Sender des-P Kanals, die Empfänger für alle B-Kanäle sowie die PIU-Logik untergebracht sind. Der SP-Baustein ist ein CMOS-Chip, der durch 'Flip-Chip' Bonding mit einem Feld von Laserdioden und einem Empfängerfeld verbunden ist. Die Faserbündel der Übertragungskanäle sind über eine geeignete planar integrierte Optik an den Chip angebunden. Die Optik realisiert auch den benötigten Fanout der P-Kanäle. Je nach Anzahl der Bus-Chips im System können für den Fanout dabei mehrere Laserdioden-Sender benutzt werden. Damit wird die notwendige Sendeleistung sichergestellt. Da die Anzahl der B-Kanäle mit maximal 32 verhältnismäßig gering ist, gibt es auf dem PIU-Baustein auf jeden Fall genug Platz für die zusätzlichen Laserdioden. Die Netzwerkbausteine samt Anbindung an die Faser der P- und M-Kanäle sind mit denen eines, mit prototypischer Technologie realisierten, PHOTOBUS-Systems weitgehend identisch. Es sind CMOS-VLSI Chips, die durch 'Flip-Chip' Bonding jeweils mit einem Feld von Empfängern und Laserdioden integriert sind. An ihnen ist jeweils eine planar integrierte Freiraumoptik befestigt, über die die Faser der P- und M-Kanäle an die Empfänger angebunden werden. Die Laserdioden werden für den B-Kanal benötigt. Er wird durch einen bzw. bei einem sehr großen System mehrere externe Broadcast-Bausteine realisiert.

Das größte Problem bei der Realisierung der PHOTON-Architektur mit prototypischer Technologie stellt die Anzahl der Faser dar, die an die Netzwerkbausteine angebunden werden müssen. Bei 512 Prozessoren und einer 16 Bit parallelen Übertragung wäre es allein für die P-Kanäle 8192 Fasern. Selbst mit Hilfe fortgeschrittener

	kurzfristig	mittelfristig
Netz-Chip	MQW-Dioden durch Flip-Chip Bonding mit CMOS-VLSI Bausteinen verbunden	
B-Kanal Fanout	Faserverteiler für Faserbündel und planar integrierte Freiraumoptik	integrierte Optik
B-Kanal Sender	SP-Bausteinen mit Laserdiodenarrays	integrierte optische Bausteine mit WDM-Sendern, Halbleiterverstärkern
P und M-Kanal	SP-Bausteine	optische Ein/Aussgabe direkt vom Prozessorchip
Faseranbindung	planar integrierte Freiraumoptik und MT-Stecker	planar integrierte Freirraumoptik mit WDM und MT-Steckern

Abbildung 6.1: Die Tabelle faßt die Implementierungsmöglichkeiten für die PHOTON-Architektur in den verschiedenen Technologieklassen zusammen.

Mikrosystemtechnik ist die stabile exakte Befestigung einer so hohen Anzahl von Fasern kaum machbar. Abhilfe kann man hier nur durch die Verwendung der WDM-Technologie schaffen. Dazu müssen die P-Kanäle jedes Bit auf einer anderen Wellenlänge senden. Dies ist mit den heute noch experimentellen Laserdiodenfeldern möglich, in denen jede Diode eine andere Frequenz besitzt. Solche Laserdiodenfelder erlauben die Realisierung eines P-Kanals mit Hilfe einer einzigen Faser, in der jedes Bit mit einer anderen Wellenlänge übertragen wird. Die P-Kanäle mehrerer benachbarter Prozessoren können jeweils zu einem parallelen Faserbündel zusammengefaßt werden, um die Anbindung an die planare Optik zu erleichtern und den 'Kabelsalat' zu reduzieren.

6.3 Leistungsparameter

Die in 6.1.2 beschriebenen Leistungsparameter werden bei der Bewertung der Architekturen durch Simulation genutzt. Dabei wird geprüft, wie die Effizienz der Architekturen von der Leistungsfähigkeit der Technologie abhängt. Damit diese Betrachtung Sinn macht, muß allerdings ein realistischer Wertbereich für die obigen Parameter geschätzt werden. Eine solche Abschätzung wird im Nachfolgenden für alle drei Technologieklassen beschrieben.

6.3.1 Sofort verfügbare Technologie

Die Leistungsfähigkeit der sofort verfügbaren Technologie ist vor allem durch zwei Faktoren beschränkt: die Leistung kommerzieller paralleler Fasersysteme und der dazugehörigen elektronischen Schnittstellen, sowie die Anzahl der Kanäle, die an die Netzwerkbausteine angebunden werden können. Für die einzelnen Parameter ergeben sich dabei die folgenden Werte:

Bandbreite der P- und M-Kanäle

Die P- und M-Kanäle benutzen kommerzielle parallele Fasersysteme als Sender und auf SP-Bausteinen integrierte MQW-Dioden als Empfänger. Die Schranken der Bandbreite heutiger paralleler Fasersender kann man aus [177, 62, 128] entnehmen. Sie liegen zwischen 800MBit/s und 2,4Gbit/s (siehe auch 4.7). Ähnliche Werte sind auch für MQW-Empfänger in SP-Bausteinen demonstriert worden (siehe [86]). Damit gilt:

$$800MBit/s \leq B_P^S = B_P^E = B_M^S = B_M^E \leq 2,4Gbit/s \tag{6.17}$$

Wie bereits erläutert, sind nicht die Bandbreite der P- und M-Kanäle, sondern deren Implementierungsaufwand entscheidend. Man kann deswegen davon ausgehen, daß jeder P- und M-Kanal mit einem parallelen Fasersender realisiert wird. Damit ist die Anzahl der Faser auf 10 bis 16 festgelegt. Da mindestens zwei Faser für Kontrollsignale (CLOCK und STROBE) benutzt werden, gilt:

$$8 \leq N_P = N_M \leq 14 \tag{6.18}$$

und mit Gleichung (6.1)

$$6,4Gbit/s \simeq 8Gbit/s \leq \quad B_P = B_M \quad \leq 33,6Gbit/s \simeq 32Gbit/s$$

$$\begin{aligned} 1GByte/s \leq & \quad B_P = B_M \leq & 8GByte/s \\ 2Byte/cycle \leq & \quad B_P = B_M \leq & 16Byte/cycle \end{aligned} \tag{6.19}$$

Die letzen beiden Zeilen gehen von der Annahme aus, daß die Prozessor-Taktrate bei 500MHz, die Zykluszeit also bei 2ns liegt.

Bandbreite der B-Kanäle

Die B-Kanäle basieren wie die P- und M-Kanäle auf parallelen Faser-Übertragungssystemen. Somit gilt:

$$800MBit/s \leq B_B^S = B_B^E \leq 2,4Gbit/s \tag{6.20}$$

Da die Bandbreite des B-Kanals entscheidend für die Systemleistung ist, kann man davon ausgehen, daß bis zu zwei parallele Sender pro Kanal benutzt werden. Damit ist

$$16 \leq N_B \leq 28 \tag{6.21}$$

Die maximale Bandbreite des B-Kanals ist also doppelt so hoch wie die der P- und M-Kanäle

$$\begin{aligned} 2GByte/s \leq & \quad B_P = B_M \leq & 16GByte/s \\ 4Byte/cycle \leq & \quad B_P = B_M \leq & 32Byte/cycle \end{aligned} \tag{6.22}$$

Fanout der B-Kanäle

Die Sendeleistung kommerzieller paralleler Fasersysteme liegt zwischen 0,5mW und 1mW ([62, 128]). Die Empfänger benötigen eine Leistung zwischen 50μW und 100μW. Unter der Annahme, daß die Verluste durch das optische System bei $0,5$ liegen, gilt:

$$0,5mW \leq \quad P_B^S \quad \leq 1mW \tag{6.23}$$

$$0,05mW \leq \quad P_B^E \quad \leq 0,1mW \tag{6.24}$$

$$V_B \quad = 0,5 \tag{6.25}$$

Damit gilt für den maximalen Fanout eines einfachen B-Kanal-Transmitters $Fout_B^0$ nach Gleichung (6.7):

$$\frac{0,5 \cdot 0,5}{0,1} \simeq 2 \leq Fout_B^1 \leq \frac{1,0 \cdot 0,5}{0,05} \simeq 8 \tag{6.26}$$

Bei der Betrachtung der Verstärkung muß zunächst zwischen einer opto-elektronischen und einer auf Faserverstärkern basierenden Implementierung unterschieden werden. Wir bezeichnen die zugehörigen Fanouts mit F_B^{oe} und F_B^{multi}. Außerdem muß man zwischen dem maximalen theoretischen Fanout und dem maximalen, mit vertretbarem Aufwand realisierbaren Fanout unterscheiden. Da man theoretisch beliebig viele Verstärkerstufen hintereinander schalten kann, kann auch ein beliebiger Fanout erreicht werden. In der Praxis muß man allerdings die Latenzzunahme sowie die Kosten des Systems bei der Betrachtung berücksichtigen. Für die maximalen mit den beiden Verfahren erreichbaren Fanouts ergibt sich dann folgendes:

Opto-elektronische Verstärkung: Bei Verwendung mehrerer Stufen paralleler Fasersender und -empfänger zur Verstärkung sind die Ein- und Ausgangsleistungen aller Verstärker gleich den Ein- und Ausgangsleistungen der Sender bzw. Empfänger . Der Fanout eines solchen Systems $Fout_B^{oe}$ steigt also exponentiell mit der Anzahl der Stufen:

$$Fout_B^{oe} = (F_B^1)^{S_B+1} \tag{6.27}$$

Da jede Stufe mit hohem Aufwand und einer Latenzzunahme verbunden ist, gehen wir davon aus, daß maximal eine Stufe verwendet wird. Dies bedeutet, daß der maximale Fanout eines solchen Systems bei 4 bis 64 liegt.

$$2^2 = 4 \leq Fout_B^{oe} \leq 8^2 = 64 \tag{6.28}$$

Verstärkung durch Faserverstärker: Optische Faserverstärker benötigen heute eine Eingangsleistung von mindestens $0,05\mu$W bis $0,1\mu$W. Sie können Signale bis auf ca. 10 mW verstärken. Damit gilt:

$$P_B^{S,i} \leq 10mW \tag{6.29}$$

$$0,05mW \leq P_B^{E,i} \leq 0,1mW \tag{6.30}$$

$$V_B^i = 0,5 \tag{6.31}$$

Da optische Faserverstärker zur Zeit sehr teuer sind, muß das System mit möglichst wenigen solchen Bauteilen auskommen. Man kann man daher zum einen davon ausgehen, daß nur eine Verstärkerstufe verwendet wird. Außerdem nehmen wir an, daß vor dem Eingang in die Faserverstärker kein Fanout stattfindet, (also $F_B^1 = 1$). Dadurch wird für jedes Bit des N-Kanals nur ein Faserverstärker benötigt. Da die Ausgangsleistung des Faserverstärkers um einen Faktor von ca. 10 über der Ausgangsleistung der parallelen Fasertransmtter liegt, wird der Fanout gegenüber einem System ohne Verstärkung trotzdem erheblich erhöht. Mit Gleichungen (6.10) und (6.23) folgt für den maximalen, mit Faserverstärkern erreichbaren Fanout $Fout_B^{faser}$:

$$50 \leq Fout_B^{faser} \leq 100 \simeq 64 \tag{6.32}$$

Die obige Betrachtung zeigt, daß mit beiden Verstärkungsmethoden der maximale, mit vertretbarem Aufwand erreichbare Fanout in etwa gleich ist. Es gilt:

$$8 \leq Fout_B \leq 64 \tag{6.33}$$

Latenz der P- und M-Kanäle:

In der PHOTOBUS-Architektur, die als einzige mit sofort verfügbarer Technologie sinnvoll realisierbar ist, sind die P- und M- Kanäle gleich. Für die einzelnen Terme, die zu ihrer Latenz beitragen (siehe Gleichungen (6.4) und (6.5)), gilt:

L_P^{ZS}, L_M^{ZS}: Die Daten für P- und M- Kanäle müssen über gewöhnliche elektrische Leitungen auf der Platine zu den Treibern der optischen Sender gelangen. Dies ist eine lokale elektrische Chip-to-Chip Kommunikation, wie sie auch zwischen dem Prozessor und dem Cache-Kontroller stattfindet. Sie benötigt je nach System zwischen 2ns und 8ns. Es gilt:

$$2ns \leq L_P^{ZS} = L_M^{ZS} \leq 8ns \tag{6.34}$$

L_P^S, L_M^S: Die Latenz des Senders setzt sich zusammen aus der Anstiegszeit der Laserdioden, sowie der Latenz der Treiber. Erstere liegt im Bereich von einigen Hundert Picosekunden, kann also vernachlässigt werden. Letztere beinhaltet die Wandlung zwischen dem niederfrequenten massiv parallelen Datenstrom des Cache-Kontrollers und dem hochfrequenten, 8 bis 14 Bit breiten Datenstrom für den optischen Sender. Aus den entsprechenden Datenblättern kommerzieller Bausteine kann für die Treiber eine Latenz zwischen 4 und 10ns entnommen werden. Damit gilt:

$$4ns \leq L_P^S = L_M^S \leq 10ns \tag{6.35}$$

L_P^L, L_M^L: Die Leitungsverzögerung einer optischen Verbindung ist durch die Lichtgeschwindigkeit in der Faser c_{faser} und die Entfernung d gegeben.

$$L_P^L = L_M^L = \frac{d}{c_{faser}} \tag{6.36}$$

L_P^E, L_M^E: Wie in 4.5.4 erläutert, ist die Latenz der MQW-Empfänger auf dem SP-Chip durch die Schaltgeschwindigkeit der elektronischen Verstärker gegeben. Da diese aus einigen wenigen, hintereinander geschalteten Transistoren bestehen, beträgt sie in heutiger Technologie maximal eine Nanosekunde:

$$L_P^E = L_M^E \leq 1ns \tag{6.37}$$

Für die Gesamtlatenz der P-und M- Leitungen ergibt sich damit nach Gleichung (6.4) und (6.5):

$$7ns + d/c_{faser} \leq L_P = L_M \leq 20ns + d/c_{faser} \tag{6.38}$$

$$3,5cycles + d/c_{faser} \leq 4cycles + d/c_{faser} \leq L_P^E = L_M^E \leq 10cycles + d/c_{faser} \tag{6.39}$$

Latenz der B-Kanäle

Im Hinblick auf die Latenz unterscheiden sich die B-Kanäle von den P- und M- Kanälen in zwei Punkten. Zum einen sind die Empfänger nicht auf einem SP-Baustein integriert, sondern genauso wie die Sender durch einen kommerziellen parallelen Faserbaustein und elektronische Schnittstellen realisiert. Zum anderen führen die B-Kanäle einen Broadcast durch, besitzen also unter Umständen eine oder mehrere Verstärkungsstufen. Dies bedeutet, daß sich die Latenz der B-Kanäle von der der P- und M- Kanäle nur in drei Termen unterschiedet: der Verzögerung bei der Datenübertragung vom Empfänger zur Logik (L_B^{VE}), der Latenz des Empfängers (L_B^E) und der Verzögerung durch das Broadcast (L_B^B). Die anderen Parameter sind mit denen der P- und M- Kanäle identisch und können aus Gleichungen (6.34), (6.35), (6.36) übernommen werden:

$$2ns \leq L_B^{ZS} \leq 8ns \tag{6.40}$$

$$4ns \leq L_B^S \leq 10ns \tag{6.41}$$

$$L_B^L = c_{faser}/d \tag{6.42}$$

Für die zusätzlichen Parameter gilt folgendes:

L_B^{VE}: Der Weg vom Empfänger zum Cache-Kontroller ist die Umkehrung des Weges vom Cache-Kontroller zum Sender. Der Wert für L_B^{VE} kann daher aus der Gleichung für L_P^{ZS} (6.34) übernommen werden:

$$2ns \leq L_B^{VE} \leq 8ns \tag{6.43}$$

L_B^E: Die Sender und Empfänger kommerzieller Faserübertragungsysteme haben, die gleiche Latenz. Wir können daher für L_B^E den Wert für L_B^S übernehmen.

$$4ns \leq L_B^E \leq 10ns \tag{6.44}$$

L_B^B: Die Latenz des Verstärkersystems hängt von seiner Realisierung ab. Ein auf Faserverstärkern basiertes System bringt lediglich eine durch die höhere Leitungslänge bedingte Verzögerung mit sich. Diese liegt im Bereich von 1 bis maximal 2 Nanosekunden. Im Falle einer opto-elektronischen Verstärkung ergibt sich die Latenz für jede Verstärkerstufe aus der Sender- und Empfängerlatenz sowie der elektronischen Leitungsverzögerung. Da für die Verstärkung die gleichen Bausteine wie für die eigentlichen Sender und Empfänger benutzt werden, kann ihre Latenz aus Gleichungen (6.41) und (6.43) übernommen werden. Gleichzeitig nehmen wir an, daß die Verzögerung durch die elektrische Leitung zwischen dem Sender und dem Empfänger in etwa der Leitungsverzögerung auf dem Weg vom Cache-Kontroller zum Sender des P-Kanals entspricht.

$$1 \leq L_B^B \leq L_B^S + L_B^E + L_B^{ZS} \leq 10ns + 10ns + 8ns \leq 30ns \tag{6.45}$$

Zusammengefaßt ergibt sich für die Latenz des B-Kanals:

$$12ns + c_{faser}/d \leq \quad L_B \quad \leq 50ns + c_{faser}/d \tag{6.46}$$

$$6cycles + c_{faser}/d \leq \quad L_B \quad \leq 25cycles + c_{faser}/d \tag{6.47}$$

Anzahl der Ein-/Ausgabekanäle der SP-Bausteine

In sofort verfügbarer Technologie werden die Faserbündel mit einer makrosokopischen Optik an einen SP-Baustein angekoppelt. Die SP-Bausteine bestehen aus MQW-Dioden, die durch 'Flip-Chip' Bonding mit CMOS-Schaltkreisen vereinigt sind. Sie werden lediglich als Empfänger benutzt, so daß nur die Anzahl der Eingabekanäle betrachtet werden muß. Da die B-Kanäle nicht an SP-Bausteinen angekoppelt sind, kann außerdem In_B aus der Betrachtung ausgelassen werden. Aus der Beschreibung der Technologien in Kapitel 4 kann damit folgendes entnommen werden:

N_{SP}^{in}: Die Anzahl der optischen Ein-/Ausgabefenster auf einem SP-Baustein hängt von drei Faktoren ab:

1. der maximalen Größe des Feldes opto-elektronischer Komponenten, die fehlerfrei hergestellt werden kann,

2. der maximalen Größe des Feldes opto-elektronischer Komponenten, die den 'Flip-Chip' Vorgang ohne Beschädigung überstehen kann und
3. der Wärmedissipation der opto-elektronischen Komponenten.

Diese Faktoren wurden für MQW basierte SPs in [16] ausgiebig untersucht. Da sie eng mit der Entwicklung der VLSI-Technologie verbunden sind, wurde dabei die Anzahl und Datenrate der optischen Ein-/Ausgabefenster als Funktion der zu erwartenden Entwicklung der minimalen Strukturgröße auf CMSO-VLSI Bausteinen betrachtet. Für die heute gängige 0,35μ m Technologie wird dabei von einer Obergrenze von 6000 Dioden pro Chip ausgegangen. Solche Bausteine können als Sonderanfertigungen von Bell-Labs bestellt werden [100]. Wir nehmen daher für N_{SP} an:

$$N_{SP} \leq 6000 \tag{6.48}$$

N_{mech}: Die Anzahl der Faser, die an einem SP-Baustein befestigt werden können, hängt vor allem von der maximalen Anzahl von Fasern in einem 2D-Bündel ab. Wie in 4.7 beschrieben, wurden bereits Systeme mit mehreren Hundert Fasern implementiert. Wir nehmen daher hier folgendes an:

$$500 \leq N_{mech} \leq 1000 \tag{6.49}$$

N_{opt}: Die Anzahl der Kanäle, die ein makroskopisches optisches System abbilden kann, ist durch die Größe des Abbildungsfeldes beschränkt, die das System mit der benötigten Qualität realisieren kann. Diese kann bei speziell entwickelten Systemen einen Durchmesser von bis zu einem Zentimeter haben. Bei einer Dichte der Faser und der optischen Ein-/Ausgabefenster im Bereich von 100/mm^2 folgt daraus, daß bis zu 10000 optische Fenster abgebildet werden können:

$$N_{opt} \leq 10000 \tag{6.50}$$

N_{WDM}: Die Nutzung mehrerer Wellenlängen in einem makromechanischen System ist im Prinzip unproblematisch. Sie kommt in einem, auf sofort verfügbarer Technologie basierendem System trotzdem nicht in Frage, da die kommerziellen parallelen Fasersysteme zur Zeit kein WDM unterstützen. Es gilt daher

$$N_{WDM} = 1 \tag{6.51}$$

Mit Gleichung (6.11) folgt daraus, daß zwischen 500 und 1000 einzelne optische Kanäle in die Netzwerkbausteine eingekoppelt werden können. Geht man davon aus, daß die Datenbreite der P- und M-Kanäle zwischen 8 und 14 Bit liegt, so folgt daraus

$$\frac{500}{14} \simeq 32 \leq \quad In_P \quad \leq \frac{1000}{8} \simeq 128 \tag{6.52}$$

$$\frac{500}{14} \simeq 32 \leq \quad In_M \quad \leq \frac{1000}{8} \simeq 128 \tag{6.53}$$

6.3.2 Kurzfristig verfügbare Technologie

Die Leistungssteigerung durch kurzfristig verfügbare Komponenten ist vor allem auf die Nutzung der SP-Bausteine an den Prozessor- und Speicherschnittstellen, sowie für die Verstärkung des B-Kanal Signals zurückzuführen. Diese Nutzung wird durch zwei Dinge ermöglicht: die Verfügbarkeit von SP-Bausteinen mit integrierten Laserdioden, sowie die Vereinfachung der Anbindung der Faser durch die Verwendung planar integrierter Freiraumoptik.

Bandbreite der P- und M-Kanäle

Die P- und M-Kanäle sollen kurzfristig mit Hilfe von SP-Bausteinen implementiert werden. Dies bedeutet, daß die Bandbreite einzelner Sender und Empfänger im Bereich von 2 bis 8 Bit/Prozessorzyklus liegt (siehe [16]):

$$\begin{aligned} 2Bits/cycle \leq B_P^S = B_P^E \leq 8Bits/cycle \\ 2Bits/cycle \leq B_M^S = B_M^E \leq 8Bits/cycle \end{aligned} \tag{6.54}$$

Durch die Verwendung von SP-Sendern sind die P-und M- Kanäle in ihrer Datenbreite theoretisch nicht mehr eingeschränkt. Allerdings ist eine Erhöhung der Kanalbreite nicht sinnvoll, da sie gleichzeitig die Anzahl der Faser erhöhen würde, die pro Prozessor an den Netzbaustein angekoppelt werden müssen. Damit würde gleichzeitig die Anzahl der P- und M- Kanäle sinken, die an die Netzbausteine angebunden werden können. Wir nehmen daher in Anlehnung an die sofort verfügbare Technologie

$$8 \leq N_P = N_M \leq 16 \tag{6.55}$$

an, was für die Bandbreite der P- und M-Kanäle zu

$$2Byte/cycle \leq \quad B_P \quad \leq 16Byte/cycle \tag{6.56}$$
$$2Byte/cycle \leq \quad B_M \quad \leq 16Bits/cycle \tag{6.57}$$

führt.

Bandbreite der B-Kanäle

Die Betrachtung im vorigen Abschnitt kann auch für die B-Kanäle übernommen werden. Der einzige Unterschied besteht darin, daß es hier durchaus Sinn macht, die Datenbreite zu erhöhen. Dies liegt daran, daß die Bandbreite der B-Kanäle, wie bereits erläutert, kritisch für die Systemleistung ist. Wir gehen daher für den B-Kanal von den folgenden Annahmen aus:

$$2Bits/cycle \leq \quad B_P^S = B_P^E \quad \leq 8Bits/cycle \tag{6.58}$$
$$16 \leq \qquad N_B \qquad \leq 32 \tag{6.59}$$

Die Kanalbreite wird also gegenüber dem sofort verfügbaren System verdoppelt. Damit gilt für die Bandbreite des B-Kanals:

$$4Bytes/cycle \leq B_B \leq 64Bytes/cycle \tag{6.60}$$

Fanout der B-Kanäle

Der Transmitter und der Empfänger des B-Kanals kurzfristig auf dem Netzbaustein integriert werden. Die Ausgangsleistung sowie die von den Empfängern benötigte Leistung sind somit durch die Parameter von Komponenten gegeben, die auf VLSI-Bausteinen integriert werden können. Wie in 4.5.1 erläutert, gilt daher (siehe auch [124]):

$$1mW \leq \quad P_B^S \quad \leq 2mW \tag{6.61}$$
$$0,05mW \leq \quad P_B^E \quad \leq 0,1mW \tag{6.62}$$
$$\tag{6.63}$$

Unter der Annahme, daß die Verluste im optischen System wie auch bei sofort verfügbaren Systemen bei maximal :

$$V_B \leq 0,5 \tag{6.64}$$

liegen, gilt für den Fanout eines Transmitters ohne Verstärkung nach Gleichung 6.8

$$4 \leq Fout_{Net}^1 \leq 20 \tag{6.65}$$

Die Verstärkung findet mit Hilfe von Broadcast-SP-Bausteinen statt. Es handelt sich dabei um VLSI-Chips mit einer großen Anzahl integrierter Laserdioden sowie einem Empfänger für das zu verstärkende Signal. Die Laserdioden senden gleichzeitig die auf dem B-Kanal übertragenen Daten, funktionieren also als mehrere Kopien des Transmitters. Jede dieser Kopien ist mit dem Transmitter des B-Kanals identisch. Dies bedeutet, daß sich die optische Leistung des Senders der Verstärkungsstufe i $P_B^{S,i}$ aus der Leistung eines einzelnen Transmitters (P_B^S) multipliziert und der Anzahl der Transmitterkopien (NT) ergibt:

$$P_B^{S,i} = P_B^S \cdot NT \tag{6.66}$$

Die Anzahl der Kopien hängt von der Anzahl der Laserdioden auf dem SP-Baustein und der Datenbreite des B-Kanals N_B ab. Erstere liegt, wie in 4.5.4 und nachfolgend in beschrieben, zwischen 1000 und 2500. Letzere ist durch Gleichung 6.59 gegeben. Damit folgt für die Anzahl der Transmitterkopien:

$$\frac{1000}{32} \simeq 32 \leq NT \leq \frac{2500}{16} \simeq 128 \tag{6.67}$$

Wir gehen davon aus, daß aus Kostengründen nicht mehr als eine Verstärkerstufe verwendet wird. Dies bedeutet mit Gleichung 6.10, daß ein Fanout zwischen 64 und 1280 möglich ist:

$$128 \leq Fout_{Net} \leq 2560 \simeq 2048 \tag{6.68}$$

Latenz der M- und P-Kanäle

Da sowohl die Empfänger als auch die Sender der M- und P-Kanäle zusammen mit der Logik auf SP-Bausteinen integriert sind, ist ihre Latenz allein durch die Sender- und Empfängerlatenz sowie die optische Leitungsverzögerung gegeben. Es ist also:

$$L_P^{ZS} = L_M^{ZS} = 0 \tag{6.69}$$
$$\tag{6.70}$$

Wie in 4.5.4 erläutert, liegt die Latenz von SP-basierten Sendern und Empfängern im Bereich weniger Gatterschaltzeiten. Wir nehmen daher an:

$$L_P^S = L_P^E \leq 1cycle \tag{6.71}$$
$$L_M^S = L_M^E \leq 1cycle \tag{6.72}$$

Zusammen mit der Gleichung für die Leitungsverzögerung eines optischen Signals (6.36) folgt für die Latenz der P- und M- Kanäle:

$$L_P \leq 2cycle + c_{faser}/d \tag{6.73}$$
$$L_M \leq 2cycle + c_{faser}/d \tag{6.74}$$

Latenz der B-Kanäle

Wie die P- und M-Kanäle verlaufen die B-Kanäle direkt zwischen SP-Bausteinen. Dies bedeutet, daß die elektrischen Leitungsverzögerungen L_B^{ZS} und L_B^{VE} entfallen. Außerdem können für die Sender- und Empfängerlatenzen, sowie für die Leitungsverzögerung, die Werte vom P- und M-Kanal aus den Gleichungen (6.71) und (6.36) übernommen werden. Es ist also:

$$L_B^{ZS} = L_B^{VE} = 0 \tag{6.75}$$
$$L_B^S = L_B^E \leq 1cycle \tag{6.76}$$
$$L_B^L \leq c_{faser}/d \tag{6.77}$$

Damit muß zur Bestimmung der Latenz der B-Kanäle nur noch die Verzögerung durch die Verstärkung L_B^B ermittelt werden. Sie ergibt sich für jede Verstärkungsstufe aus der Latenz der Empfänger und Sender des Broadcast-Chips sowie der Zeit, die für die Verteilung des Signals auf die Transmitterkopien notwendig ist. Da wir von nur einer Verstärkerstufe ausgehen, gilt:

$$L_B^B \leq 4cycle \tag{6.78}$$

Die Latenz des B-Kanals ist also:

$$L_B \leq 6cycles + c_{faser}/d \tag{6.79}$$

Anzahl der optischen Datenkanäle an den SP-Bausteinen

Die kurzfristig verfügbaren Lösungen unterscheiden sich wie besprochen in zwei Punkten von den sofort verfügbaren. Zum einen beinhalten die SP-Bausteine nicht nur MQW-Dioden-Empfänger, sondern auch Laserdioden-Sender. Dies bedeutet, daß sowohl die Eingabe-, als auch die Ausgabekanäle berücksichtigt werden müssen Zum anderen findet die Anbindung der Faser an die SP-Bausteine mit Hilfe planar integrierter Freiraumoptik statt. Die Auswirkung dieser Veränderungen auf die, für den Fanin relevanten Parameter kann wie folgt zusammengefaßt werden:

N_{SP}^{in}: Für die kurzfristig verfügbare $0,18\mu$ Technologie wird in [16] eine Anzahl von bis zu 24000 Dioden angenommen, die auf einem SP-Chip integriert werden können. Wir setzen daher

$$N_{SP}^{out} \leq 24000 \tag{6.80}$$

N_{SP}^{out}: Da Laserdioden in einer ähnlichen Technologie hergestellt und auf, durch 'Flip-Chip' Bonding mit CMOS-VLSI verbunden werden können, kann man diesen Wert auch für Laserdioden-basierte SPs übernehmen. Allerdings muß man bei solchen SPs zusätzlich noch die Wärmedissipation berücksichtigen. Geht man von einer Laserdiodenleistung von 2mW und einer Effizienz von 0,5 aus, so dissipiert eine Laserdiode $2mW$. Dies wären bei 24000 Laserdioden 48W pro Chip, also in etwa soviel wie von einem Chip durchschnittlicher Größe ohne Wasserkühlung abgeleitet werden kann [57]. Dies bedeutet, daß bei 24000 Laserdioden keine Wärmekapazität mehr für die Logik und die Treiber übrig wäre. Um die Wärmemenge, die die Laserdioden dissipieren, auf maximal ein Viertel des gesamten Wärmebudgets zu beschränken, nehmen wir an:

$$N_{SP}^{out} \leq 5000 \tag{6.81}$$

N_{mech}: Die Verwendung von planarer Freiraumoptik und Mikromechanik zur Anbindung von 1D-Fasersteckern an die SP-Chips macht die Herstellung einfacher und billiger. Die Anzahl der Faser, die an einem Chip befestigt werden können, wird dabei allerdings nicht wesentlich vergrößert. Dies liegt daran, daß die Größe planar integrierter Systeme auf wenige Zentimeter Durchmesser beschränkt ist. Gleichzeitig ist die Faserdichte, die mit Hilfe mehrerer paralleler 1D-Stecker erreicht werden kann, recht gering. Im Nachfolgenden gehen wir von einer planaren Optik aus, die auf einem 6x6cm großen Substrat integriert ist. Solche Substrate wurden bereits demonstriert. Wir nehmen gleichzeitig an, daß der Chip selbst 2x2cm groß, und in der Mitte angebracht ist. Dies läßt für die Befestigung von Fasersteckern einen Randbereich mit 2cm Breite. Die heutigen kommerziellen Faserstecker für 12 Faser sind ca. 1cm breit und 4mm dick. Da die Faser in einem Abstand von 125μm angebracht sind, würde sich die Breite beim Übergang zu 32 Faser um 4mm auf in etwa 1,5cm erhöhen. Solche Faserstecker können heute bereits als Sonderanfertigungen bestellt werden. Auf dem Rand der planaren Optik können also ohne weiteres 80 Stecker a 32, oder 100 a 12 Faser untergebracht werden. Damit gilt:

$$1200 \simeq 1000 \leq N_{mech} \leq 2560 \simeq 2500 \tag{6.82}$$

N_{opt}: Die Anzahl der Kanäle in einem planar integrierten optischen System vor allem ist durch den minimalen Winkel gegeben, den das Übertragungssystem auflösen kann. Die Herleitung der maximalen Kanalmenge wurde in [111, 74] erläutert. Für eine Substratdicke von 5mm und einer Wellenlänge von 850nm wurden dabei ca. 3500 als obere Grenze der Kanalzahl ermittelt. In der Praxis wurden bereits Systeme mit bis zu 2500 Kanälen demonstriert [154]. Bei der Anbindung der Faserstecker an einen SP-Chip muß zusätzlich noch berücksichtigt werden, daß die Signale von den am Rand gelegenen Steckern von vier Seiten zu dem in der Mitte befindlichen Chip geleitet werden. Wir haben es dadurch mit vier unabhängigen Übertragungssystemen zu tun, so daß die Kanalanzahl vervierfacht wird. Es gilt also:

$$N_{opt} \leq 10000 \tag{6.83}$$

N_{WDM}: Da SP-basierte Sender kurzfristig keine Möglichkeit zum Wellenlängenmultiplexing bieten, nehmen wir, wie im Falle sofort verfügbarer Technologie,

$$N_{WDM} = 1 \tag{6.84}$$

an.

Aus obiger Betrachtung kann man ersehen, daß die Anzahl einzelner optischer Kanäle an einem SP-Baustein durch N_{mech} beschränkt ist. Sie ist somit für Ein- und Ausgabekanäle gleich und liegt zwischen 1000 und 2500. Bei einer Datenbreite zwischen 8 und 16 für die P- und M- Kanäle sowie 16 bis 32 für den B-Kanal folgt damit:

$$\frac{1000}{16} \simeq 64 \leq \quad In_P = Out_P \quad \leq \frac{2500}{8} \simeq 256 \tag{6.85}$$

$$\frac{1000}{16} \simeq 64 \leq \quad In_M = Out_M \quad \leq \frac{2500}{8} \simeq 256 \tag{6.86}$$

$$\frac{1000}{32} \simeq 32 \leq \quad In_B = Out_B \quad \leq \frac{2500}{16} \simeq 128 \tag{6.87}$$

Dies ist nicht wesentlich mehr als bei der sofort verfügbaren Technologie. Der Vorteil liegt hier in der einfacheren billigeren Herstellung und der Kompaktheit der Systeme.

6.3.3 Mittelfristig verfügbar

Mittelfristig wird die Leistung der hier untersuchten Systeme von zwei Dingen profitieren: Der Möglichkeit, die Kanalanzahl durch die Verwendung des WDM-Verfahrens zu steigern, und der Verbesserung der Leistungsfähigkeit von Laserdioden-basierten SP-Bausteinen. Von den Verbesserungen sind vor allem das Fanout des B-Kanals und der Fanin der SP-Bausteine betroffen. Die Latenz bleibt unverändert. Die Bandbreite kann durch die höhere Anzahl zur Verfügung stehender Kanäle leicht erhöht werden.

Bandbreite der P- und M-Kanäle

Da wir gemäß [16] annehmen, daß die Bandbreite der optischen Ein-/Ausgabekanäle mit der VLSI-Leistung skaliert, bleibt die Datenrate der Kanäle in Prozessorzyklen ausgedrückt unverändert:

$$2Bits/cycle \leq B_P^S = B_P^E \leq 8Bits/cycle \tag{6.88}$$

$$2Bits/cycle \leq B_M^S = B_M^E \leq 8Bits/cycle \tag{6.89}$$

Die Verwendung des WDM-Verfahrens bedeutet, daß nun bis zu 32 Kanäle pro Faser geschickt werden können. Daraus folgt, daß die Datenbreite der P- und M- Kanäle erhöht werden kann, da die Anzahl der Faser, die an die Netzbausteine angeschlossen werden, nicht mehr kritisch ist. Damit gilt:

$$8 \leq N_P = N_M \leq 32 \tag{6.90}$$

so daß die Bandbreite der P- und M-Kanäle zu

$$2Byte/cycle \leq \quad B_P \quad \leq 64Byte/cycle \tag{6.91}$$

$$2Byte/cycle \leq \quad B_M \quad \leq 64Bits/cycle \tag{6.92}$$

wird.

Bandbreite der B-Kanäle

Die Betrachtung im vorigen Abschnitt kann auch für die B-Kanäle übernommen werden, und zwar inklusive der Verdoppelung der Datenbreite. Es gilt also:

$$2Bits/cycle \leq \quad B_P^S = B_P^E \quad \leq 8Bits/cycle \tag{6.93}$$

$$16 \leq \qquad N_B \qquad \leq 64 \tag{6.94}$$

und damit:

$$4Bytes/cycle \leq B_B \leq 128Bytes/cycle \tag{6.95}$$

Fanout der B-Kanäle

Das Fanout kann mittelfristig entweder genauso wie im Falle der kurzfristig verfügbaren Technologie oder mit integrierten opto-elektronischen Komponenten mit optischen Halbleiterverstärkern realisiert werden. Im ersten Fall ist der mit einer Verstärkerstufe erreichbare Fanout identisch mit der kurzfristig verfügbaren Technologie. Dies liegt daran daß einerseits die Leistung der Laserdioden um einen Faktor von 2 höher angenommen wird, andererseits aber die Anzahl der Transmitterkopien auf dem Chip wegen der Verdoppelung der Datenbreite um die Hälfte sinkt. Allerdings kann man davon ausgehen, daß mittelfristig die SP-Bauteile sowie die zugehörige Optik und Mechanik billiger wird, so daß man für große Rechner ohne weiteres zwei Verstärkerstufen nutzen kann. Damit gilt für den maximalen Fanout:

$$128^2 = 16384 \leq Fout_{Net} \leq 1024^2 = 1048576 \tag{6.96}$$

Bei der Verwendung integrierter optischer Systeme als Verstärker kann man pro Stufe von einem Fanout zwischen 10 und 20 ausgehen. Da die Bausteine kompakt und mittelfristig billig sein werden und außerdem keine zusätzliche Latenz verursachen kann man davon ausgehen daß ohne weiteres 4 bis 8 Stufen benutzt werden können so daß ein ähnliche Fanout wie in Gleichung (6.96) angegeben erreicht werden kann.

Fanout der P-Kanäle

In der PHOTON-Architektur müssen die P-Kanäle einen Fanout besitzen. Da dieser nicht besonderes groß sein muß, gehen wir davon aus, daß er ohne eine Verstärkerstufe auskommt. Für die Leistung des P-kanal Senders P_P^S, die vom Empfänger benötigte Leistung P_P^E und die Verluste des optischen Systems V_P nehmen wir die gleichen Werte wie beim B-Kanal an. Es gilt:

$$2mW \leq \quad P_P^S \quad \leq 2mW \tag{6.97}$$

$$0,05mW \leq \quad P_P^E \quad \leq 0,1mW \tag{6.98}$$

$$V_P \quad \leq 0,5 \tag{6.99}$$

Damit liegt der Fanout eines einfachen Transmitters wie auch beim B-Kanal zwischen 4 und 20. Da auf der dem SP-Baustein der Prozessorschnittstelle wesentlich weniger optische Eingänge als auf dem Bus-Chip untergebracht werden müssen, kann man davon ausgehen, daß 2 bis 4 Kopien des Transmitters ohne weiteres auf dem Chip Platz finden. Damit ist

$$8 \leq Fout_P \leq 80 \simeq 64 \tag{6.100}$$

Latenz der P-, M- und B-Kanäle

Wie am Anfang des Abschnitts angedeutet, ist mittelfristig keine Änderung der Latenz zu erwarten. Sie ist bereits bei der kurzfristig verfügbaren Technologie überwiegend durch die Leitungsverzögerung beschränkt. Es gilt also:

$$L_P \quad \leq \quad 2cycle + c_{faser}/d \tag{6.101}$$

$$L_M \quad \leq \quad 2cycle + c_{faser}/d \tag{6.102}$$

$$L_B \quad \leq \quad 6cycle + c_{faser}/d \tag{6.103}$$

Anzahl der optischen Ein-/Ausgabekanäle an den SP-Bausteinen

N_{SP}^{in}**:** Für die mittelfristig verfügbare $0,1\mu$ Technologie wird in [16] eine Anzahl von bis zu 50000 Dioden angenommen, die auf einem SP-Chip integriert werden können. Wir nehmen daher für die Anzahl der Empfänger auf einem SP-Baustein an:

$$N_{SP}^{in} \leq 50000 \tag{6.104}$$

N_{SP}^{out}**:** Die Anzahl der Laserdioden, die auf einem SP-Baustein untergebracht und betrieben werden können, wird auch mittelfristig durch die Wärmedissipation bestimmt sein. Wir gehen hier davon aus, daß die Verbesserungen in der Effizienz und Wärmeableitungs-Technik die Steigerung der Laserleistung weitgehend kompensiert. Dies bedeutet, daß die maximale Anzahl von Laserdioden auf einem SP-Baustein mit der kurzfristig verfügbaren Technologie gleich ist:

$$N_{SP}^{out} \leq 5000 \tag{6.105}$$

N_{mech}: Bei Verwendung von 1D-Fasersteckern kann für die Anzahl der Faser an einem SP-Baustein die Betrachtung aus Abschnitt 6.3.2 übernommen werden. Allerdings ist mittelfristig, wie in Kapitel4 beschrieben, davon auszugehen, daß große 2D-Faserbündel durch automatische Massenherstellung preiswert verfügbar sein werden. Dadurch können die Fasern mit einer wesentlich höheren Dichte auf der planaren Optik untergebracht werden. Der typische Faserabstand in solchen Bündeln liegt bei 125μ bis 250 μm. Damit ergibt sich eine Dichte von 16 bis 64 Fasern pro mm^2. Bezüglich der für die Faserbefestigung verfügbaren Fläche gehen wir von den in Abschnitt 6.3.2 beschriebenen Annahmen bezüglich der Größe des SP-Chips (ca. 2X2cm) und der planaren Optik (ca. 6X6 cm) aus. Damit steht eine Fläche von bis zu 3200 mm^2 für die Faser zur Verfügung. Sie reicht für über 100000 Faser aus. Allerdings sind 2D-Faserbündel mit mehr als 10000 Faser auch mittelfristig nicht zu erwarten []. Wir nehmen daher an, daß an allen vier Seiten des SP-Chips 1000 zu einem Bündel zusammengefaßte Fasernbefestigt werden können. Damit gilt:

$$1000 \leq N_{mech} \leq 40000 \tag{6.106}$$

Die untere Schranke wurde hier aus der Betrachtung für ein System aus 1D-Fasersteckern übernommen.

N_{opt}: Bereits bei der kurzfristig verfügbaren Technologie wurde mit der maximalen theoretischen Anzahl von Kanäle gerechnet, die in der planaren Optik realisiert werden kann. Daher kann mittelfristig keine weitere Steigerung erwartet werden.

$$N_{opt} \leq 10000 \tag{6.107}$$

N_{WDM}: Die Anzahl der Wellenlänge ist vor allem durch die maximale Anzahl von Wellenlängen in einem Laserdiodenfeld beschränkt. Wie in 4.5.2 beschrieben, sind in verschiedenen Labors Laserdiodenfelder mit 8 bis 16 verschiedenen Wellenlänge demonstriert worden. Wir setzen daher:

$$8 \leq N_{WDM} \leq 16 \tag{6.108}$$

Die obige Betrachtung zeigt, daß die Anzahl einzelner Eingabekanäle durch die Leistung des optischen Systems während die Anzahl der Ausgabekanäle durch die Wärmedissipation der Laserdioden beschränkt ist. Sie beträgt für Eingabekanäle 10000 und für Ausgabekanäle 5000. Damit ergibt sich aus der, durch Gleichungen (6.90) und (6.94) gegebenen Datenbreite für die einzelnen Kanäle:

$$\frac{10000}{16} \simeq 512 \leq \quad In_P = In_M \quad \leq \frac{10000}{8} \simeq 1024 \tag{6.109}$$

$$\frac{5000}{16} \simeq 256 \leq \quad Out_P = Out_M \quad \leq \frac{5000}{8} \simeq 512 \tag{6.110}$$

$$\frac{10000}{32} \simeq 256 \leq \quad In_B = \quad \leq \frac{10000}{16} \simeq 512 \tag{6.111}$$

$$\frac{5000}{32} \simeq 128 \leq \quad Out_B = \quad \leq \frac{10000}{16} \simeq 256 \tag{6.112}$$

6.3.4 Zusammenfassung

Die Ergebnisse der obigen Betrachtung sind für die einzelnen Komponenten und Technologieklassen in Tabelle 6.3 zusammengefaßt.

6.4 Entwurf eines modularen 'Plug-and-Play' Systems

In den vorigen Abschnitten wurden verschiedene Möglichkeiten zur Implementierung der untersuchten Architekturen vorgeschlagen. Aufbauend auf dieser Betrachtung hat der Autor in Zusammenarbeit mit Wissenschaftlern aus den Bereichen der Optischen Nachrichtentechnik der Fernuniversität Hagen, und der Mikromechanik am IMM Mainz ein Konzept für ein modulares opto-elektronisches Baukastensystem entwickelt. Das System verfolgt zwei Ziele:

1. Es soll zur Implementierung von Prototypen der hier beschriebenen Architekturen im Rahmen eines von der DFG geförderten Forschungsprogramms benutzt werden.

Zeitrahmen	sofort	kurzfristig	mittelfristig
B_P,B_M (Bytes/cycles)	2 bis 16	2 bis 16	2 nis 64
B_B (Bytes/cycles)	4 bis 32	4 bis 64	4 bis 128
L_P,L_M (cylces)	4+d/c bis 10+d/c	$\leq$ 2+d/c	
L_B (cylces)	6+d/c bis 25+d/c	$\leq$ 6+d/c	
$Fout_P$ (mit B_P,B_M)	-	-	8 bis 64
$Fout_B$ (mit B_B)	8 bis 64	64 bis 2048	$\geq$ 16384
In_P, In_M (Kanäle mit B_P,B_M)	32 bis 128	64 bis 256	512 bis 1024
In_B (Kanäle mit B_P,B_M)	-	32 bis 128	256 bis 512
Out_P, Out_M (Kanäle mit B_P,B_M)	-	64 bis 256	256 bis 512
Out_B (Kanäle mit B_P,B_M)	-	32 bis 128	128 bis 256

Tabelle 6.3: Möglichen Werte für die Leistungsparameter der untersuchten Architekturen für die verschiedenen Technologieklassen

2. Es soll den Rechnerarchitekten die Möglichkeit geben, opto-elektronische Verbindungen als 'Plug-und-Play' Komponenten in ihre Systeme einzubauen. Dadurch soll der Opto-Elektronik zu einer breiteren Anwendung im Bereich der Rechnerarchitektur verholfen werden.

Um die obigen Ziele zu erreichen, faßt das System SP-Bausteine, planar integrierte Freiraumoptik und Fasersysteme zu modularen Komponenten zusammen, die durch elektrische und optische Steckverbindungen miteinander beliebig verbunden werden können. So ist es möglich, eine Vielzahl von opto-elektronischen Systemen einfach durch Zusammenstecken von Standardkomponenten aufzubauen. Man kann also die Vorteile von SP-Bauteilen, von komplexen optischen Freiraumverbindungen und von hochparallelen Faser-Übertragungssystemen nutzen, ohne sich mit der Optik oder der Opto-Elektronik auseinandersetzten zu müssen.

Im Nachfolgenden wird zunächst die Grundidee des Baukastensystems erläutert. Daraufhin wird das Prinzip des modularen Aufbaus verschiedener opto-elektronischer Systeme aus obigen Komponenten erläutert. Abschließend wird gezeigt, wie diese Komponenten zum Aufbau der, in der vorliegenden Arbeit vorgeschlagenen Architekturen benutzt werden können.

6.4.1 Grundidee

Das Baukastensystem basiert auf kurzfristig verfügbarer Technologie. Aus diesem Grund setzt es auf folgende Komponenten:

1. Parallele 1D-Faserbündel mit passenden Steckern für die Datenübertragung auf lange Distanzen,
2. Planar integrierte optische Freiraumsysteme für komplexe Verbindungsstrukturen und
3. SP-Bausteine, die optische Ein-/Ausgabefenster durch 'Flip-Chip' Bonding auf konventionellen CMOS-VLSI-Bausteinen für die Anbindung optischer Datenkanäle an VLSI-Schaltungen integrieren.

Es ist so konzipiert, daß es leicht auf die Nutzung mittelfristig verfügbarer Bauteile, insbesondere 2D-Faserbündel und WDM-Datenübertragung erweitert werden kann.

Das System integriert die obigen Komponenten in drei Arten von Modulen: Fasermodule, Topologiemodule und SP-Module. Diese Module können auf zwei Arten miteinander verbunden werden: durch konventionelle parallele Faserstecker oder durch eine spezielle opto-mechanische Stecker-Schnittstellen. Außerdem können die SP-Bausteine in handelsübliche Platinen gesteckt, und so miteinander und mit anderen Komponenten elektrisch verbunden werden. Die Funktionen der Schnittstelle und der drei Modularten kann wie folgt zusammengefaßt werden:

Opto-Mechanische Stecker-Schnittstelle

Die Schnittstelle erlaubt das Zusammenstecken verschiedener Module nach dem Vorbild heutiger kommerzieller Faserbündelstecker. Dazu definiert sie ein Raster optischer Fenster, um die herum genau positionierte Öffnungen für Justierstifte angebracht sind. Außerdem geht sie davon aus, daß die Lichtsignale die optischen Fenster immer als parallele, zur Oberfläche senkrechte Strahlenbündel verlassen bzw. in sie eintreten.

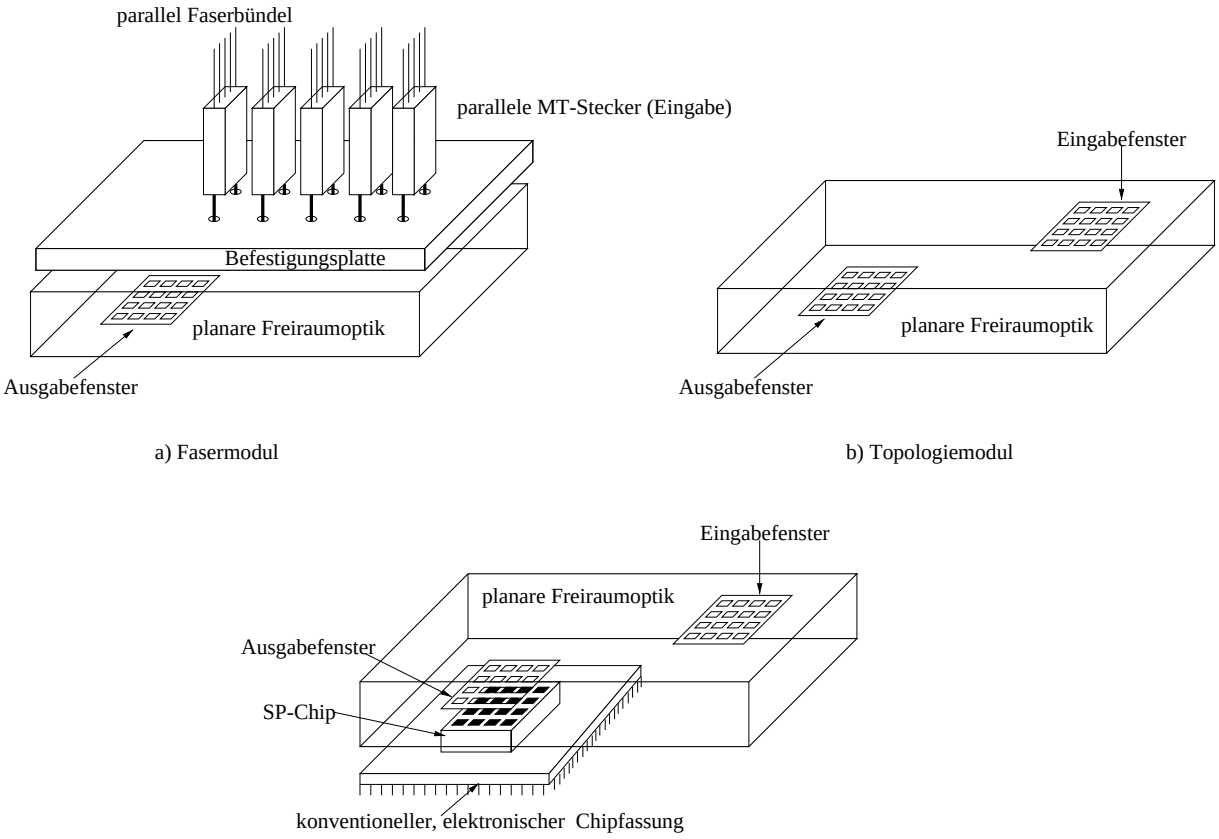


Abbildung 6.2: Die drei Arten von Modulen, aus den das vorgeschlagene 'Plug-and-Play' System besteht.

Sollen zwei Module zusammengesteckt werden, so werden Justierstifte in die entsprechenden Öffnungen gesteckt, die Module zusammengefügt und mit passenden Klammern befestigt. Falls die Verbindung permanent sein soll, können die Module auch geklebt werden.

Fasermodule

Ein Fasermodul verbindet eine Menge von parallelen, mit Fasersteckern versehenen Faserbündeln mit einer opto-mechanischen Stecker-Schnittstelle (siehe Abbildung 6.2a). Dazu werden die einzelnen Stecker zunächst mit Hilfe einer Fixierplatte zusammengefaßt und befestigt. Die Ausgänge der befestigten Stecker werden dann mit Hilfe einer planaren Optik auf die optischen Fenster der Schnittstelle abgebildet.

Die anderen Enden der Faserbündeln können entweder ebenfalls in einem Fasermodul zusammengefaßt sein, oder mit konventionellen Fasersteckern versehen werden. Im ersten Fall kann das Faserbündel als 1:1 Verbindung zwischen zwei Modulen des Baukastensystems verwendet werden. Im zweiten Fall dient das Bündel als Schnittstelle zu kommerziellen Bausteinen. So können sowohl parallele Fasersender und -empfänger, als auch handelsübliche Faserverteiler und Faserverstärker mit dem System verbunden werden. Außerdem kann ein Fasermodul durch einzelne Stecker mit mehreren anderen Fasermodulen verbunden werden. So kann eine komplexe Netzwerktopologie aufgebaut werden, die mehrere Module des Baukastensystems miteinander verbindet.

SP-Module

Ein SP-Modul bindet die optischen Ein-/Ausgabefenster eines SP-Bausteins an eine opto-mechanische Stecker-Schnittstelle an. Dazu wird eine entsprechende planare Freiraumoptik an einem SP-Baustein befestigt. Sie bildet seine optischen Ein-/Ausgabefenster auf das Raster der Schnittstelle ab (Abbildung 6.2c).

Damit ein SP-Modul in konventionellen elektronischen Aufbauten integriert werden kann, muß er in einen Sockel oder einen Slot gesteckt werden können. Hierzu ist es am besten, wenn sich der Baustein in einer konventionellen IC-Fassung befindet, und samt Fassung an die opto-mechanische Stecker-Schnittstelle angebunden wird.

Topologiemodule

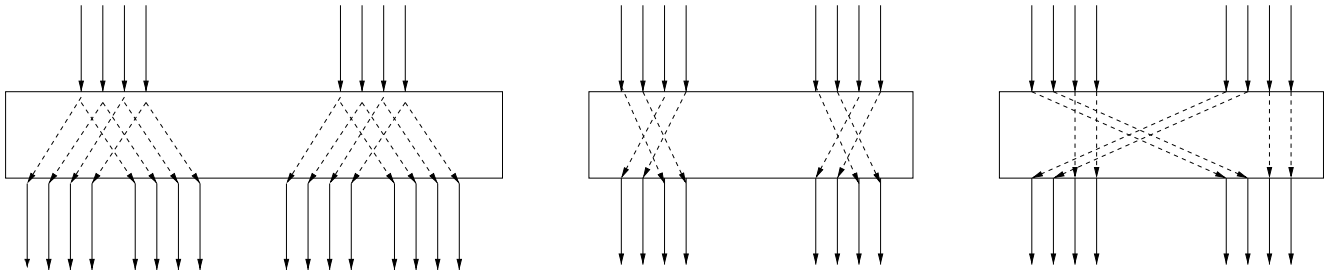
Mit Hilfe von Fasermodulen mit Übergang auf einfache Faserstecker lassen sich beliebige Verbindungstopologien realisieren. Dazu müssen lediglich die einzelnen Stecker entsprechend verbunden werden. Auch Broadcast und Multicast-Oprationen lassen sich realisieren, wenn man die einzelnen Faser an konventionelle 1xn Verzweiger anschließt. Allerdings ist dieses Verfahren bei massiv parallelen Verbindungen mit Hunderten von Fasern sehr aufwendig und mit einem schwer handhabbaren 'Kabelsalat' verbunden. Dies kann mit Hilfe geeigneter Topologiemodule vermieden werden, die die gewünschte Abbildung für ganze Faserbündel in einem Schritt realisieren. Solche Module bestehen aus einer planar integrierten Freiraumoptik, die beidseitig mit opto-mechanischen Stecker-Schnittstellen versehen ist (Abbildung (6.2b)). Sie bilden die optischen Fenster der einen Schnittstelle nach einer vorgegeben Topologie auf die Fenster der anderen Schnittstelle ab. Damit können z.B. zwei, jeweils an ein Fasermodul angeschlossene Faserbündel nach einem vorgegeben Muster miteinander verbunden werden. Analog könnten die Faser eines Faserbündels auf die optischen Fenster eines, an ein SP-Modul angeschlossenen SP-Bausteins abgebildet werden.

6.4.2 Standardkomponenten

Die große Stärke der Elektronik liegt in der Verfügbarkeit von Standardkomponenten aus denen sehr flexibel viele Standardsysteme aufgebaut werden können. So greift der Rechnerarchitekt nur in Sonderfällen zum VLSI-Design und entwirft einen eigenen Baustein. Um die Opto-Elektronik für den breiten Einsatz attraktiv zu machen, werden auch hier Standardkomponenten benötigt. Im Nachfolgenden wird, basierend auf dem obigen modularen System, ein solcher Satz von Standardkomponenten vorgeschlagen.

Schnittstellen-Standard

Je nach Anwendung werden unterschiedliche Mengen von Fasern benötigt. Daher ist es sinnvoll, die Schnittstelle für mehrere, unterschiedliche Faserzahlen zu definieren. Diese sollten allerdings jeweils ein Vielfaches der Faserzahlen in kommerziellen parallelen Fasersteckern (4, 8, 12, 16, 24) sein.



a) 2x(1x2) Rundruf-Modul a) Intra-Bündel Permutation c) Inter-Bündel Permutation

Abbildung 6.3: Die Funktionalität der Standard-Topologiemodule

Standard-Fasermodule

Um die verschiedenen, für die Schnittstelle definierten Faserzahlen auszunutzen, müssen entsprechende Fasermodule als Standardkomponenten verfügbar sein. Um außerdem eine flexible Verknüpfung von Modulen mit unterschiedlichen Schnittstellen sowie handelsüblichen Faserkomponenten zu ermöglichen, werden außerdem Standardkomponenten vorgeschlagen die unterschiedliche Fasermodule und kommerzielle Stecker miteinander Verbinden. Sie bestehen aus einem Faserbündel an deren beiden Enden unterschiedliche Fasermodule bzw. ein Fasermodul und die gewünschten handelsüblichen Stecker angebracht sind. Die wichtigsten solchen Module sind:

1:1 Verbindungen: 1:1 Verbindungen bestehen aus einem Faserbündel, an dessen beiden Enden identische Fasermodule angebracht sind. In einem Standardbaukasten sollten für jede Schnittstellengröße entsprechende 1:1 Verbinder vorhanden sein.

1:N Verbindungen: Eine 1:N Verbindung ist ein Faserbündel, das an einem Ende ein großes, und am anderen mehrere kleine Fasermodule besitzt. So können die Ausgänge mehrerer, aus wenigen Fasern bestehender Module zur Weiterverarbeitung zu einem großen Modul zusammengefaßt werden.

Übergang auf Faserstecker: Zur Anbindung an kommerzielle Faserkomponenten ist es notwendig, eine Schnittstelle zu kommerziellen Fasersteckern zu schaffen. Dazu werden die freien Enden der, an einem Fasermodul befestigten Fasern mit solchen Steckern konfektioniert. Dabei kann es sich sowohl um parallele Stecker als auch um Einfachstecker handeln.

Standard Topologiemodule

Für ein Standardsystem sind vor allem regelmäßige Verbindungen für große Faserbündeln sowie parallele Broadcast-Operationen mit großem Fanout interessant. Einfache Verbindungen zwischen wenigen Fasern können mit Hilfe von Fasermodulen mit einem Steckeranschluß realisiert werden. Für massiv parallele, unregelmäßige Topologien müssen spezielle maßgeschneiderte Topologiemodule realisiert werden. Es werden daher folgende Standard-Topologiemodule vorgeschlagen:

Kx(1xN) Broadcast-Module: Ein solches Modul realisiert gleichzeitig K Broadcast-Operationen, jede auf einem anderen Faserbündel. Das Modul besitzt K Eingangs-Faserbündel und K Gruppen von N Ausgangs-Faserbündel. Das Signal jedes Eingangsbündels wird auf alle N Bündel einer bestimmten Gruppe verteilt. Ein solches Modul wird benötigt, um das Signal eines Broadcast-SP-Bausteins auf die Empfänger zu verteilen.

Inter-Bündel Permutationen: Bei einer Inter-Bündel Permutation werden verschiedene Teile eines jeden Faserbündels am Eingang zu verschiedenen Ausgangs-Faserbündeln zugeordnet. Dabei sind vor allem regelmäßige Permutationen von Bedeutung bei denen die Faser jedes Eingangsbündels nach dem gleichen Muster auf die Ausgangsbündel abgebildet werden. Solche Pemutationen erlauben eine einfache Realisierung verschiedener Netzwerktopologien inklusive vollverbundener Netze sowie Broadcast- und Select Netzwerke.

Intra-Bündel Permutationen: Bei Intra-Bündel Permutatonen ist jedem Bündel am Eingang ein Bündel am Ausgang zugeordnet. Dabei werden die Faser eines jeden Eingangs-Faserbündels nach dem gleichen Muster

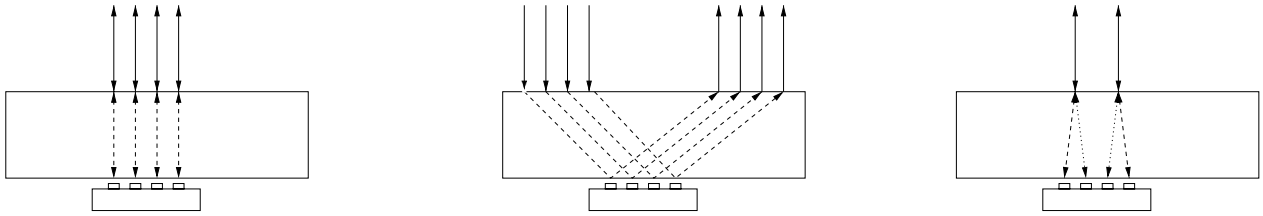


Abbildung 6.4: Die Funktionalität der vorgeschlagenen Standrad SP-Module

auf die Faser des korrespondieren Ausgangssteckers abgebildet. Als Abbildungsmuster sind dabei vor allem regelmäßige Permutationen sinnvoll.

Standard SP-Module

Da die Topologiemodule eine flexible Abbildung eines Eingabefeldes auf ein Faserbündel erlauben, werden für die SP-Module keine komplexen Abbildungsgeometrien zwischen den optischen Ein-/Ausgabefenstern der SP-Module und denen der opto-mechanischen Stecker-Schnittstelle benötigt. Da der Weg der Lichtstrahlen umkehrbar ist, muß außerdem nicht zwischen Eingabe- und Ausgabefenstern der SP-Bausteine unterschieden werden. Folglich müssen beim Entwurf von Standard-SP-Modulen nur zwei Gesichtspunkte bedacht werden. Zum einen sind SP-Module für die verschiedenen, für die Schnittstellen definierten Faserzahlen notwendig. Zum anderen muß zwischen Laserdioden und Modulator-basierten Sendern unterschieden werden.

1:1-Module: 1:1 Module bilden jedes optische Fenster des SP-Bausteins auf ein Fenster der opto-mechanischen Stecker-Schnittstelle ab (Abbildung 6.4a). Sie können sowohl für Eingabe als auch für Ausgabe mit Hilfe von Laserdioden benutzt werden.

Module mit Ausleuchtung: Module mit Ausleuchtung dienen zur Anbindung Modulator-basierter SPs an das System. Zu diesem Zweck sind die optischen Fenster der Schnittstelle in zwei Gruppen geteilt. Die Fenster der beiden Gruppen werden auf die optischen Ein-/Ausgabefenster des SP-Bausteins abgebildet. Diese Abbildung ist so beschaffen, daß das Licht von den Fenstern der einen Gruppe nach der Reflexion von den SP-Elementen zu den Fenstern der anderen Gruppe gelangt(Abbildung 6.4b). Damit dient die eine Gruppe als Eingang für die, zur Ausleuchtung dienenden Strahlen, während die andere den Ausgang darstellt. Das besondere an der Anordnung ist, daß die Ausgabefenster auch als Eingabefenster benutzt werden können. Somit kann das gleiche Modul benutzt werden, unabhängig davon, an welchen Stellen sich auf dem Chip die Modulator-Sender und an welchen sich die Empfänger befinden.

Module mit Wellenlängentrennung: Für die Nutzung des WDM-Verfahrens ist es notwendig, daß zur gleichen Faser gehörende Signale unterschiedlicher Wellenlänge auf unterschiedliche Ein-/Ausgabefenster des SP-Chips abgebildet werden. Hierzu ist ein spezielles WDM-taugliches SP-Modul notwendig (siehe Abbildung 6.4c).

6.4.3 Aufbau der untersuchten Architekturen

Alle drei in der Arbeit untersuchten Architekturen können aus den oben beschriebenen Standardkomponenten aufgebaut werden.

PHOTOBUS

Die Netzwerkbausteine sowie die Prozessor- und Speicherschnittstellen werden als 1:1 SP-Module realisiert. An diese SP-Module werden Fasermodule mit parallelen Fasersteckern (z.B. MT-Steckern) über die opto-mechanische Stecker-Schnittstelle angebunden. Dabei geht vom Netzwerkbaustein ein paralleler Faserstecker für jeden P-Kanal und jeden M-Kanal sowie für den Sender des B-Kanals ab. Passend dazu besitzt das SP-Modul jeder Prozessor- und Speicherschnittstelle jeweils einen Stecker für den Ausgang des B- bzw. M-Kanals und einen für den Eingang des B-Kanals. Die P- und M-Kanaleingänge des Netzwerkchips werden einfach mit den zugehörigen Ausgängen der

Prozessor- und Speicherschnittstellen zusammengesteckt. Der B-Kanalausgang wird zunächst über ein passendes Fasermodul an ein 1x(1xX) Broadcast-Modul angeschlossen. Dabei ist X der maximale Fanout des Transmitters. Die Ausgänge des Broadcast-Moduls sind an die Broadcast-SP-Bausteinen angeschlossen. Diese sind als 1:1 SP-Module implementiert und an ein Fasermodul mit Übergang zu parallelen Fasersteckern angesteckt. Jeder der anderen Stecker ist mit einer der Transmitterkopien des SP-Broadcast-Bausteins verbunden. Er wird an ein 1x(1xX) Broadcast-Modul angeschlossen, dessen Ausgänge an die B-Kanalstecker eines Teils der Prozessor- und Speichermodule angeschlossen sind.

PHOTOBAR

Die Implementierung der PHOTOBAR-Architektur unterscheidet sich von der der PHOTOBUS-Architektur nur in einem Punkt: der Anzahl der Stecker, die die Prozessor- und Speicherschnittstellen mit dem Netzbaustein verbinden. Da die P- und M-Kanäle bidirektional sind, besitzt das Fasermodul an dem SP-Modul mit dem Netzbaustein zwei Stecker für jeden Prozessor und jede Speicherbank: einen für die Eingabe und einen für die Ausgabe. Analog haben die Fasermodule an den SP-Moulen der Prozessor- und Speicherschnittstellen jeweils einen Eingabe- und einen Ausgabestecker für den P- bzw. M-Kanal.

PHOTON

Die Netzwerkbausteine und die B-Kanäle können bei der PHOTON-Architektur genauso wie bei der PHOTOBUS-Architektur realisiert werden. Einen Unterschied gibt es lediglich bei den Prozessorschnittstellen. Diese müssen zum einen mit mehreren B-Kanälen verbunden werden. Außerdem besitzen sie mehrere Kopien des P-Kanal-Transmitters, da die P-Kanäle einen Broadcast an alle Netzwerkbausteine durchführen müssen. Dementsprechend ist das 1:1 SP-Modul an den Prozessorschnittstellen mit einem Faser-Modul versehen, das einen Stecker für jeden B-Kanal und für jede Kopie des Transmitters besitzt. Die P-Kanal Stecker sind an ein Kx(1xX) Boradcast-Modul angekoppelt, dessen Ausgänge an die Netzwerkbausteine angeschlossen sind. Dabei ist K die Anzahl der Transmitterkopien, und X der Fanout eines jeden Transmitters, wobei das Produkt aus K und X der Anzahl der Netzwerkbausteine im System entspricht

6.5 Implementierungsfragen

Die einzelnen Komponenten des im vorigen Abschnitt vorgeschlagenen Systems sind alle bereits in Form von Prototypen in verschiedenen Forschungslabors demonstriert worden. Gleiches gilt für die Integration einiger Bauteile zu Gesamtsystemen, insbesondere die Integration von Faserbündeln und SP-Bausteinen. Dies wurde bereits in Kapitel 4 ausgiebig erläutert. Allerdings wurden dabei nicht genau die, bzw. nicht alle für das hier vorgeschlagene Baukastensystem notwendigen Komponenten integriert. Damit stellt die Frage der Realisierung des Baukastensystems ein offenes Forschungsproblem dar. Im Rahmen der Arbeit hat der Autor hierzu einige Möglichkeiten untersucht. Dabei wurden auch in Zusammenarbeit mit Projektpartnern aus Mainz erste einfache Prototypen für kritische Teilkomponenten realisiert. Weitere Forschungsarbeiten werden im Rahmen des laufenden DFG-Projektes durchgeführt.

Im Nachfolgenden werden zunächst die Probleme und die vorgeschlagenen Lösungen bei der Realisierung der verschiedenen Teile des Baukastensystems und seiner Standardkomponenten erläutert. Danach werden die implementierten Teilkomponenten beschrieben.

6.5.1 Fasermodule

Für die Realisierung der Fasermodule sind zwei Dinge notwendig:

1. Die Stecker der einzelnen Faserbündel müssen zusammengefaßt, justiert und befestigt werden.
2. Die Ausgänge der einzelnen Bündel müssen auf die optischen Fenster der opto-mechanischen Stecker-Schnittstelle abgebildet werden.

Die optische Abbildung soll mit Hilfe einer geeigneten planar integrierten Freiraumoptik realisiert werden. Hierzu werden zur Zeit an der Fernuniversität Hagen Forschungsarbeiten durchgeführt. Das in der Arbeit vorgeschlagene Prinzip der Integration mehrerer paralleler 1D- Faserstecker und ihrer Befestigung an der planaren Optik ist in

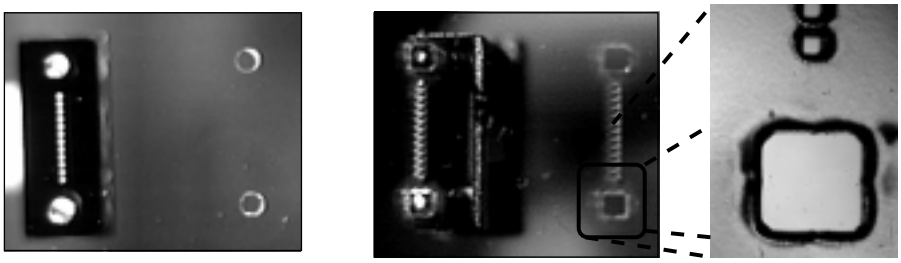


Abbildung 6.5: Die nach einer Idee des Autors am IMM-Mainz hergestellten Protypen einer Fixierplatte für die Fasermodule. Abbildung a zeigt eine mit Hilfe eines Mechanischen Verfahrens hergestellte Platte. Im Bild b ist ein mit Hilfe der Laserabalation realiserter Bauteil zu sehen. Eine Vergrößerung der Öffnungen ist in Bild c gezeigt. Die Größe Öffnung hat einen Durchmesser von 0,6mm, die kleinen ca 125μm.

Abbildung 6.5 verdeutlicht. Dabei werden die parallelen Faserstecker vertikal auf einer Fixierplatte positioniert und verklebt. Für die Positionierung werden die Justierstifte verwendet, die normalerweise für die Ausrichtung einer Steckverbindung benutzt werden. Zu diesem Zweck gibt es in der Platte für jeden Stecker passende Löcher, in die die Stifte eingeführt werden können. Bei Verwendung von Standard MT-Steckern müssen die Löcher einen Durchmesser von $600\mu m$ und einen Abstand von $4,6mm$ besitzen. Bauteile mit solchen Löchern können, wie in Abschnitt 4.6.1 beschrieben, mit Hilfe des LIGA-Verfahrens mit einer Genauigkeit unterhalb eines μm billig hergestellt werden.

Experimentelle Fixierplatten für die Faserstecker

Um die Möglichkeit zu testen, 1D-Faserstecker mit Hilfe einer Fixierplatte zu einem 2D-Feld zusammenzufassen, wurden am IMM-Mainz zwei Demonstrator gebaut. Die Bausteine sind in Abbildung 6.5 zu sehen. Da die Erzeugung der für das LIGA-Verfahren nötigen Maske sehr teuer ist, und sich nur bei einer hohen Stückzahl lohnt, wurden für die Herstellung mikromechanische 'rapid-prototyping' Verfahren angewendet. Im Fall des ersten Bausteins wurden mit einem Präzisionsbohrer 4 Paare von Justierlöchern gebohrt. Beim zweiten Demonstrator wurden in einer PMMA-Platte zwei Paare von Justierlöchern mit Hilfe der Laserabalation erzeugt. Um die Verluste beim Durchgang der Signale durch die PMMA-Platte zu vermeiden, wurden zusätzlich 12 kleine Öffnungen (eine pro Faser) zwischen den Justierlöchern angebracht

Bei allen Bauteilen konnte eine Genauigkeit von $\pm 2\mu m$ erreicht werden. Dies ist sogar mehr, als für die Positionierung von Multimode-Fasern notwendig ist. Laut Angaben des IMM würde sich die Genauigkeit bei Verwendung des LIGA-Verfahrens weiter verbessern, so daß sogar die Nutzung von Monomode-Fasern denkbar wäre.

6.5.2 Topologiemodule

Die Topologiemodule bestehen aus einer planaren Optik und zwei opto-mechanischen Stecker-Schnittstellen. Ihre Realisierung bringt zwei Probleme mit sich. Zum einen muß die gewünschte Abbildung zwischen den Ein- und Ausgabefenstern in der planaren Optik implementiert werden. Zum anderen müssen die Fixierplatten für die Schnittstelle an der planaren Optik positioniert und befestigt werden.

6.5.3 SP-Module

Bei der Realisierung der SP-Module müssen zwei Gesichtspunkte berücksichtigt werden: die Implementierung der SP-Bausteine selbst, sowie ihre Anbindung an die opto-mechanische Schnittstelle. Ersteres ist bereits im Abschnitt 4.5.4 ausgiebig diskutiert worden. Letzteres ist besonderes dann schwierig, wenn man wegen der Kompatibilität zur konventionellen Elektronik die SP-Bausteine in handelsüblichen Fassungen unterbringen möchte. Dies liegt vor allem daran, daß die Position des Chips im Sockel nicht genau definiert ist. Die gilt sowohl für die Position in der Ebene, als auch für die Höhe des Chips. Hinzu kommt, daß die elektrischen Bonding-Drähte eine direkte Befestigung eines planaren optischen Systems auf dem Chip schwierig machen.

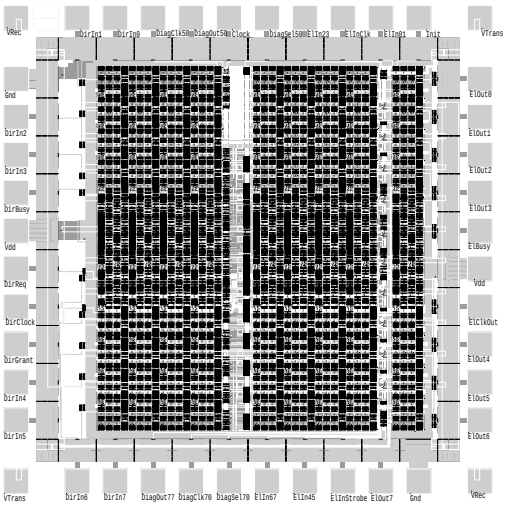


Abbildung 6.6: Die Abbildung zeigt das VLSI-Layout des vom Autor entworfenen und bei Bell-Labs produzierten einfachen PHOTOBUS Netzwerkchip-Protytyps.

Implementierungskonzept

Um die obigen Probleme zu umgehen, wird vorgeschlagen, auf dem Chip einen Positionierrahmen zu befestigen. Wie in Abbildung 6.9 dargestellt, sind an dem Rahmen mikroskopische Füße angebracht, die in die freien Räume zwischen den Bondpads, sowie auf bestimmte Positioniermarkierungen auf dem Chip passen. Die Positioniermarkierungen können beim Entwurf des Chips auf der Oberfläche z.B. als kleine, strukturierte Metall-Pads vorgesehen werden. So kann sichergestellt werden, daß der Rahmen in Bezug auf die optischen Fenster des Chips in einer exakt definierten Position in der Ebene angebracht wird. Da die Füße des Rahmens auf den Chip aufgesetzt werden, ist auch seine Höhe über dem Chip genau bestimmt. Der Rahmen ist von oben mit einer Platte wohldefinierter Dicke abgeschlossen, die an die Ränder der Fassung angeklebt ist. Diese Platte bildet die Schnittstelle zu dem planar integrierten optischen System, daß die optischen Fenster des SP-Chips auf die opto-mechanische Stecker-Schnittstelle abbildet. Für die Positionierung des optischen Systems sind an der Platte Justiermarkierungen mit einer genau definierten Position in Bezug auf den Rahmen angebracht.

Ein Problem des obigen Verfahrens besteht darin, daß der Rahmen einen Abstand zwischen 1mm und ca. 5mm zwischen der Oberfläche des SP-Chips und der planaren Optik schafft. Insbesondere dann, wenn die SP-Bausteine Laserdioden-Sender besitzen, kann dies bei der Abbildung zu Problemen führen, da auf diese Entfernung der Lichtkegel der Laserdioden einen beträchtlichen Durchmesser erreichen kann. Um dies zu vermeiden, kann man in der Mitte des Rahmens, also über den optischen Ein-/Ausgabefenstern des SP-Bausteins, ein Feld von Mikrolinsen in den Rahmen integrieren. Dieses Feld kann die optischen Fenster direkt auf die Eingänge der planaren Optik abbilden.

Für die Herstellung des Rahmens muß eine komplexe Struktur mit mehreren Millimetern Dicke und ein bis zwei Zentimetern Durchmesser mikrometergenau realisiert werden. Hierfür eignet sich am besten die LIGA-Technologie. Sie erlaubt, wie bereits erläutert, sowohl die notwendige Präzision als auch die, für die Füße des Rahmens notwendige Strukturtiefe. Mit ihrer Hilfe kann bei Bedarf ein Mikrolinsen-Feld in den Rahmen integriert werden (siehe 4.6.1).

Experimentelle SP-Chips

Der Autor hat im Verlauf der Arbeit zwei einfache SP-Bausteine entworfen. Beide wurden bei Bell-Labs in den USA im Rahmen eines sog. Foundry-Runs hergestellt. Bei beiden handelte es sich um 2x2mm große CMOS-VLSI Bausteine auf denen 200 MQW-Dioden gebondet waren. Der erste-Chip, der 1996 produziert wurde, beinhaltet lediglich einfache Shiftregister. Auf dem zweiten Chip wurde ein einfacher Prototyp des Bus-Chips für die PHOTOBUS-Architektur realisiert. Der Prototyp beinhaltet on Chip Bus, 8 optische Eingänge für die P -und M-Kanäle jewils mit einem 64 Bit-Buffer und einen elektrischen und und optischen Ausgang für den B-Kanal. Sowohl die optischen als auch die elektrischen Datenkanäle besitzen 8 Datenleitungen so wie eine CLOCK und eine STROBE Leitung.

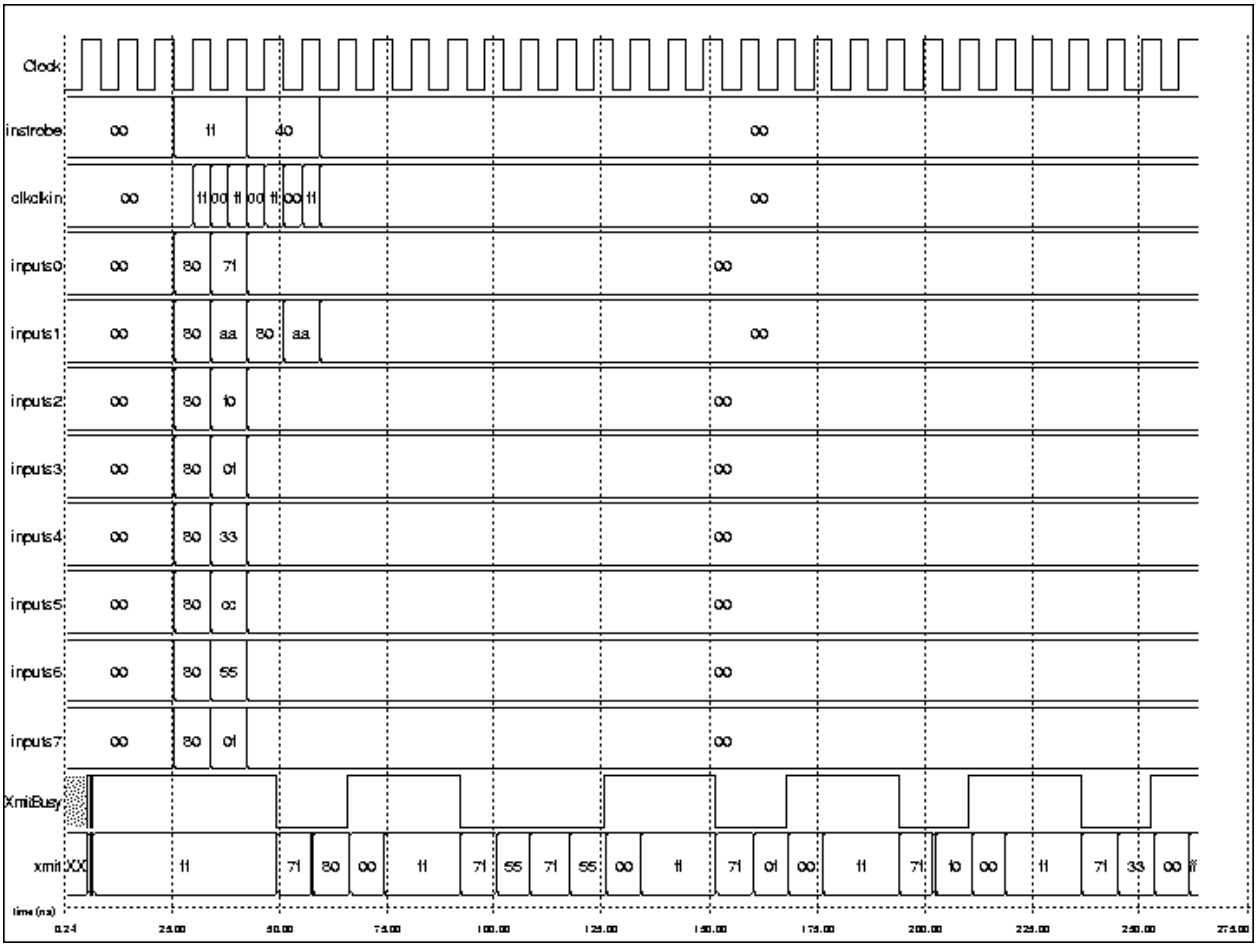


Abbildung 6.7: Das Ergebnis eine Simulation des vom Autor entworfenen Prototypen des PHOTOBUS Netzwerkchips. Die Zeitachsen ist in 25ns-Abschnitte eingeteilt. Bei t=25ns werden and die als input1 bis input7 bezeichneten optischen Eingänge verschiedene Daten gesendet. Diese werden dann nacheinander von dem als xmitt bezeichneten Transmitter übertragen, wobei auf Grund der von den Transmittern benötigten Spannungswerte das Inverse der Eingangssignale auf dem Transmitter erscheint.

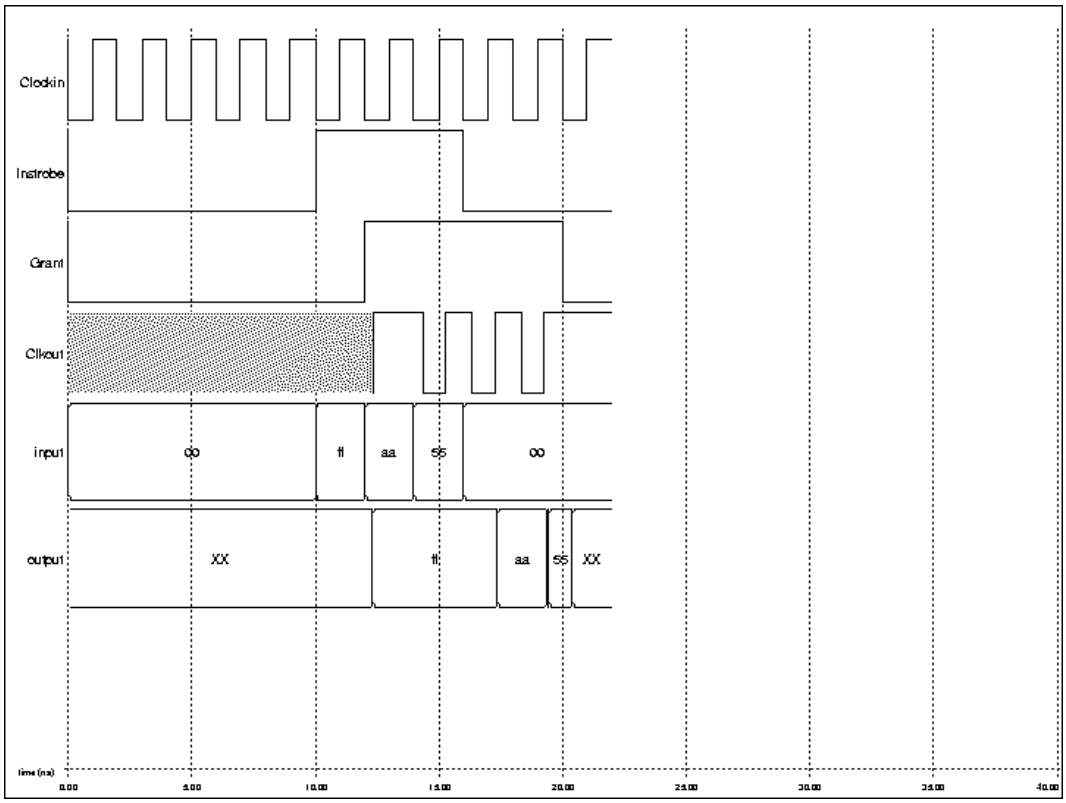


Abbildung 6.8: Das Ergebnis einer Simulation eines optischen Eingangs mit einer Betriebsfrequenz von 500 MHz. Die Daten, die über die Eingabeleitungen kommen (zu hexadezimalen input Signal zusammengefaßt) werden zwischengespeicher und 2,5ns später die die Ausgabeleitungen (zu hexadezimalen output Signal zusammengefaßt) korrekt weitergeleitet.

Das VLSI-Layout des Chips ist in Abbildung 6.6, der Chip selber in Abbildung 6.9 zu sehen. Da der Chip erst nach langer Zeit ausgeliefert wurde und einen aufwendigen opto-elektronischen Testaufbau erfordert, können in der Arbeit keine Testergebnisse präsentiert werden. Allerdings wurde das realisierte Design mit den SPICE und IRSIM VLSI-Simulatoren untersucht. Dabei wurden die vom Hersteller des Chips zur Verfügung gestellten Parameterdatein benutzt. Diese Paramterdatein wurde bei Bell-Labs vielfach benutzt, wobei eine gute Übereinstimmung von Simulation und Experiment festgestellt wurde. Die Simulation kann daher als ein wichtiges Indiz für die Leistung des Chips angesehen werden.

Die Ergebnisse sind in Abbildungen 6.7 und 6.8 zu sehen. In der Abbildung 6.7 wird die Arbitrierung und Übertragung von 8 Wörtern a 2 Byte demonstriert, die alle gleichzeitig auf den unterschiedliche Eingabekanäle ankommen. Dabei wird gezeigt, daß das System mit einer Frequenz von 200MHz betrieben werden kann und dabei eine Latenz von 8ns besitzt. Die Betriebsfrequenz ist in diesem Fall durch die Geschwindigkeit der Standard VLSI-Zellen beschränkt, die in dem Entwurf der Arbitrierungslogik benutzt wurden. Die einzelnen Eingabekanäle können mit einer deutlich höheren Frequenz betrieben werden. Dies wird in Abbildung 6.8 deutlich, die den Betrieb eines Eingabekanals mit einer Taktrate von 500 MHz demonstriert.

Experimentelle Positionierrahmen für SP-Chips

Ein Positionierrahmen, der am IMM auf der Basis der hier vorgestellten Idee hergestellt wurde, ist in Abbildung 6.9 zu sehen. Wie im Fall der Fixierplatten konnte aus Kostengründen für die Herstellung nicht das LIGA-Verfahren benutzt werden. Trotzdem wurden für den rahmen Genauigkeiten im Bereich weniger Mikrometer erreicht. Es wurde gezeigt, daß bei einer entsprechenden Vorbereitung des SP ein Rahmen mit geringen Aufwand auf dem Chip positioniert werden kann.

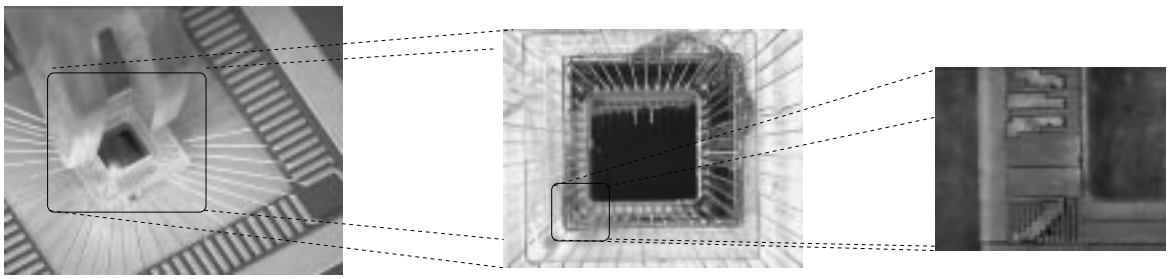


Abbildung 6.9: Der am IMM Mainz nach Vorgaben des Autors hergestellte Positionierrahmen für SP-Module. Abbildung a zeigt wie der Rahmen mit einem Manipulator auf dem Chips positioniert wird. In Bild b ist eine Nahaufnahme des Rahmens zu sehen. In Bild c ist eine Vergrößerung einer Ekce des Rahmens zu sehen, die die zur Justierung verwendeten Markierung zeigt.

6.6 Fazit

Die Betrachtung im vorliegenden Kapitel hat gezeigt, daß optische Verbindungen mit einer großen Anzahl dichtgepackter Kanäle möglich sind, die eine hohe Bandbreite, hohen Fanout und trotzdem eine geringe Latenz besitzen. Dank der Fortschritte in der Opto-Elektronik, der Mikrooptik und Mikromechanik können mehrere Hundert bis mehrere Tausend solche Kanäle direkt an konventionelle VLSI-Bausteinen angeschlossen werden. So können Netzwerkschnittstellen realisiert werden, die die hohe Bandbreite optischer Verbindungen voll ausnutzen. Gleichzeitig sorgt die direkte Anbindung an die VLSI Logik dafür, daß bei der Wandlung zwischen dem elektrischen und dem optischen Datenstrom keine nennenswerte Latenz entsteht. Die konkreten zu erwartenden Leistungsparameter sind in Tabelle 6.3 zusammengefaßt. Sie wurden auf der Basis einer umfassenden Literaturstudie abgeschätzt.

Für die Praktische Anwendung einer Technologie ist das Vorhandensein von Standardkomponenten entscheidend, aus den komplexe System einfach und billig aufgebaut werden können. Im Falle opto-elektronischer Netzwerke ist es insbesondere wichtig, daß Leistungsfähige System aufgebaut werden können, ohne daß man sich sich mit der technischen Problemen der Optik oder der Opto-Elektronik auseinandersetzten muß. Nur so kann diese Technologie eine Akzeptanz im Bereich der Rechnerarchitektur gewinnen. Aus dieser Überlegung heraus wurde im vorliegenden Kapitel ein Konzept für ein opto-elektronisches 'Plug- and-Play' System entwickelt und durch die Herstellung einfacher Komponenten verifiziert. Es faßt SP-Bausteine, planar integrierte Freiraumoptik und Fasersysteme zu modularen Komponenten zusammen, die durch elektrische und optische Steckverbindungen miteinander beliebig verbunden werden können. So ist es möglich, eine Vielzahl von opto-elektronischen Systemen einfach durch Zusammenstecken von Standardkomponenten aufzubauen. Auch die in der Arbeit vorgeschlagenen opto-elektronischen SMP-Architekturen können mit Hilfe des Systems implementiert werden.

Kapitel 7

PHOTOBUS

In diesem Kapitel wird das PHOTOBUS-System betrachtet, daß auf einem SP-Chip einen paket-vermittelnden Bus implementiert. Dabei liegen die Beiträge der Arbeit in vier Bereichen:

1. Es wird gezeigt, welche Voraussetzungen erfüllt werden müssen, damit das Berkeley- Protokoll korrekt und effizient realisiert werden kann (Abschnitt 7.1).

2. Für alle für die Realisierung des System benötigten Komponenten wird die Hardwarearchitektur beschrieben. Dazu gehören die optischen Sender und Empfänger (Abschnitt 7.2) Speicher- und Prozessorschnittstellen (Abschnitte 7.3 und 7.4) und der Bus-Chip (Abschnitt 7.5). Der Aufbau der Komponenten wird durch Strukturdiagramme, ihre Funktionalität durch endliche Automaten spezifiziert.

3. Basierend auf dem Architekturentwurf und den in Kapitel 6 beschriebenen Technologieparametern werden die Skalierbarkeit und Leistungsfähigkeit des Systems abgeschätzt. Dazu wird zunächst in Abschnitt 7.6 die Abhängigkeit der benötigten Bus-Chip Fläche von der Anzahl der Prozessoren betrachtet. Danach werden in Abschnitten 7.7 und 7.8 die Ergebnisse der Systemsimulation und der theoretischen Analyse präsentiert.

4. Es wird gezeigt, wie die Leistung des System gesteigert werden kann, indem auf dem Bus-Chip ein Synchronisationsmechanismus integriert wird (Abschnitt 7.9). Dazu wird zunächst das Prinzip der On-Chip Synchronisation erläutert. Danach werden die notwendigen Modifikationen in der Architektur und Funktionsweise des Bus-Chips und der Schnittstellen beschrieben. Anschließend werden Simulationsergebnisse präsentiert, die das Ausmaß der Leistungssteigerung dokumentieren.

7.1 Überblick

Die hier vorgeschlagene opto-elektronische PHOTOBUSArchitektur realisiert auf dem SP-Chip die Funktionalität eines paket-vermittelnden Busses. Wie in Abbildung 7.1 zu sehen, sind die Prozessoren und die Speicherbänke über unidirektionale optische Kanäle an einen SP-Chip (den Bus-Chip) angeschlossen. Diese Kanäle werden als P- und M-Kanäle bezeichnet. Der Bus-Chip kann auf einem ebenfalls unidirektionalen, optischen Kanal (B-Kanal) durch einen Rundruf an das ganze System verschicken. Die Prozessoren und die Speicherbänke schicken über ihre P- bzw. M-Kanäle Daten an den Bus-Chip. Dieser speichert sie zwischen und leitet sie dann nacheinander über den B-Kanal an das ganze System weiter. Damit wird die Funktionalität eines Busses nachgeahmt.

Wie in Kapitel 5 beschrieben, sind bei der Realisierung des Berkeley Cache Kohärenz Protokolls drei Dinge zu beachten: der Datenfluß, der Kontrollfluß, sowie das Timing der Supply-Operationen. Im Nachfolgenden werden die Grundideen beschrieben, auf denen die Bewältigung dieser Aufgaben in der PHOTOBUS-Architektur basiert. Die Details sind in den nächsten Abschnitten bei der Beschreibung der einzelnen Komponenten zu finden.

7.1.1 Ablauf der Speicheroperationen

Der logische Datenfluß zwischen den Prozessoren und den Speicherbänken, der zur Realisierung des Berkeley-Protokolls benötigt wird, wurde in 5.2 beschrieben und in Abbildung 5.2 dargestellt. In der PHOTOBUS-Architektur wird dieser Datenfluß wie folgt realisiert:

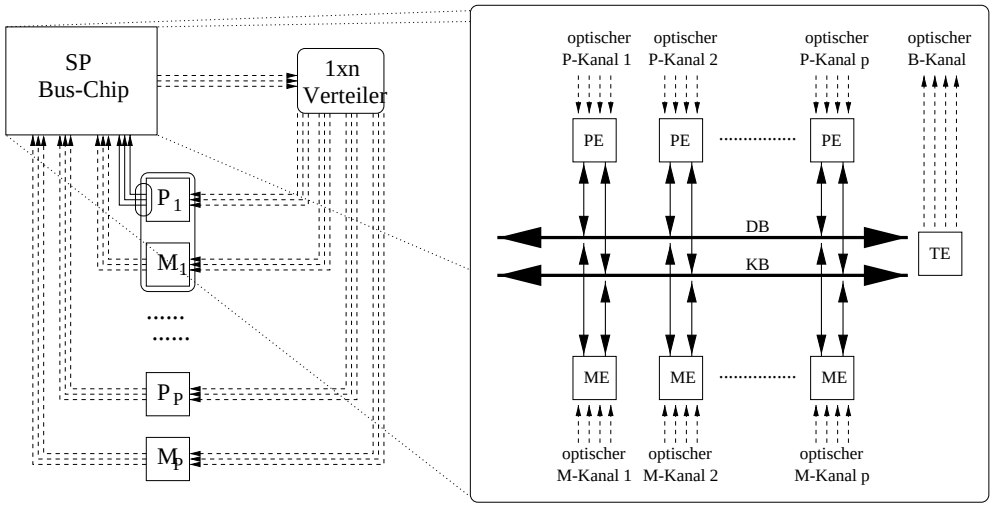


Abbildung 7.1: Der Aufbau der PHOTOBUS-Architektur.

FetchRead

Das Kommunikationsmuster für eine Leseoperation ist in Abbildung 7.9 oben dargestellt. Um eine neue Cache-Zeile nach eine Read-Miss zu holen, schickt ein Prozessor über seinen P-Kanal eine Nachricht an den SP-Buschip. Die Nachricht beinhaltet die Adresse der gewünschten Cache-Zeile und ein Befehlswort mit dem Lesebefehl. Sie wird von SP-Buschip in dem zugehörigen P-Puffer gespeichert und in die Warteschlange des B-Kanal Transmitters eingereiht. Sobald der Transmitter für sie verfügbar ist, wird die Nachricht über den B-Kanal an alle Prozessoren und Speicherbänke weitergeleitet. Die Anfrage wird von allen Speicherbänken überprüft, und von der, die diese Adresse beinhaltet, zur Bearbeitung übernommen. Gleichzeitig prüfen die Prozessoren, ob sie die Zeile in ihrem Cache besitzen und wenn ja, in welchem Zustand sie sich befindet. Der weiterer Ablauf hängt davon ab, ob ein Prozessor die Zeile im PrivateModified Zustand besitzt. Falls ja, dann liegt eine *Supply-Anforderung* vor. Der Besitzer der Zeile informiert sofort den Bus-Chip und führt die entsprechende Supply-Operation durch, indem er den Inhalt der geforderten Cache-Zeile über seinen P-Kanal schickt.

Ansonsten wartet der Bus-Chip, bis die Speicherbank das Datum gelesen hat, und es über ihren M-Kanal verschickt hat. Sobald der Bus-Chip die gewünschte Zeile erhalten hat (egal ob von einem anderen Prozessor oder von der Speicherbank), reiht er sie in die Warteschlange des Transmitters ein, um sie dann als Rundruf zu verschicken. Dabei wird in dem Befehlswort vermerkt, daß es sich bei der Übertragung um die Antwort des Speichers handelt. Damit wissen alle anderen Prozessoren und die Speicherbänke, daß sie die Nachricht nicht weiter betrachten müssen.

FetchWrite

Wie in Abbildung 7.2 oben zu sehen, geht ein Prozessor beim Holen einer Zeile für eine Schreiboperation fast genauso wie bei einem Lesezugriff vor. Der einzige Unterschied zum Lesezugriff besteht darin, daß ein Schreibzugriff eine Invalidierung verursacht. Alle Prozessoren, die die betroffene Zeile in ihrem Cache besitzen, markieren diese nach dem ersten Rundruf als ungültig. Dies gilt unabhängig davon, in welchem Zustand sich die Zeile in ihrem Cache befindet, also auch dann, wenn ein Prozessor die Zeile im PrivateModified Zustand besitzt. In diesem Fall schickt er sie zuerst an den Bus-Chip und markiert sie dann in seinem Cache als ungültig.

Invalidate-und Exclusive Operation

Das Anfordern des Schreibrechts für eine bereits gültige Cache-Zeile unterscheidet sich von den bisher betrachteten Operationen darin, daß keine Daten verschickt werden und der Prozessor keine Antwort auf seine Anfrage benötigt. Wie in Abbildung 7.2 unten rechts zu sehen, schickt der Prozessor zunächst die Adresse der Cache-Zeile zusammen mit einem entsprechenden Befehlswort an den Bus-Chip. Die Nachricht wird wieder gespeichert, in die Transmitter-Warteschlange eingereiht und über den B-Kanal an das ganze System geschickt. Die Prozessoren speichern die Adresse, prüfen, ob sie sie in ihrem Cache haben, und invalidieren gegebenenfalls die entsprechende

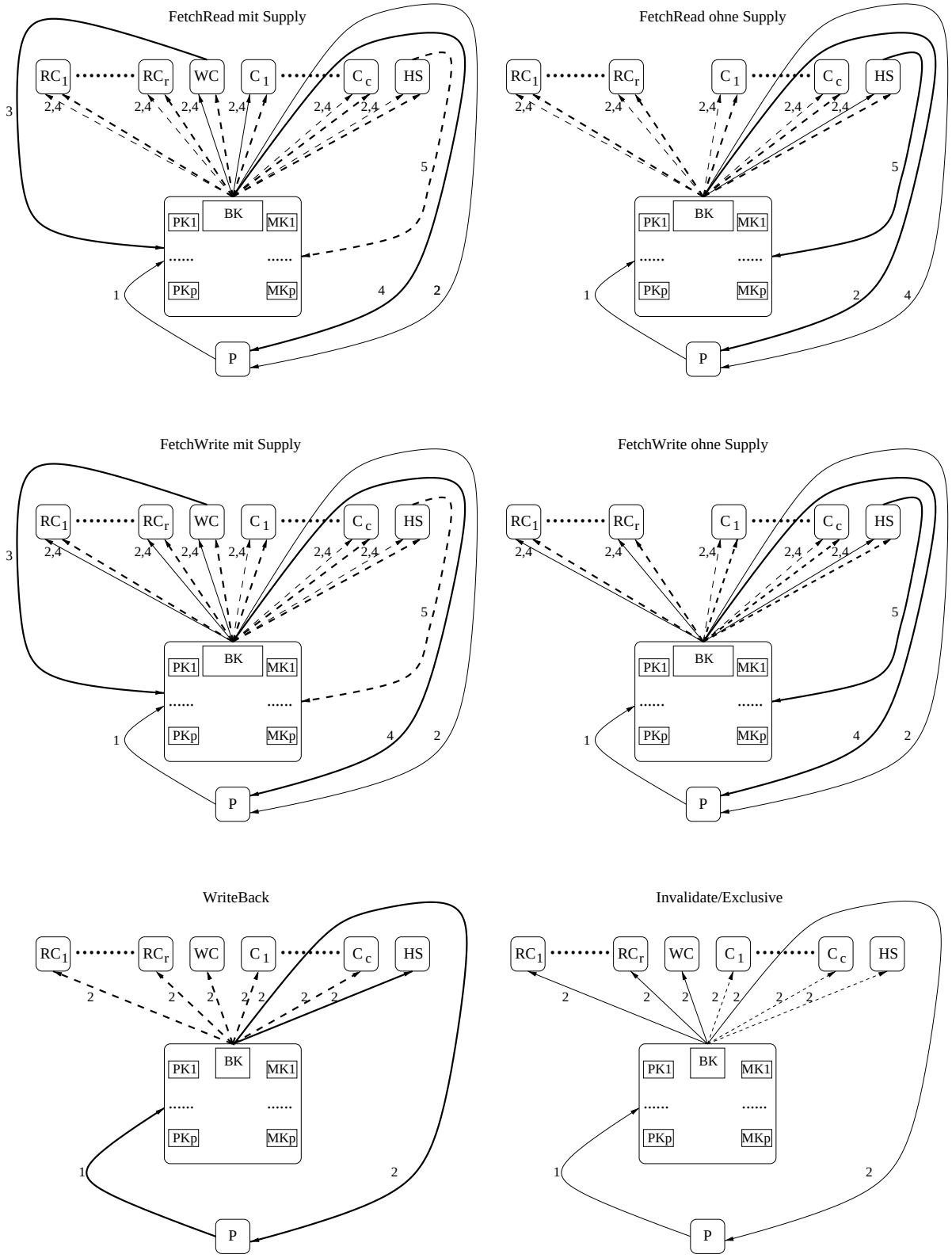


Abbildung 7.2: Der Ablauf der Speicheroperationen des Berkeley-Protokolls in der PHOTOBUS-Architektur. Die gestrichelte Linien bedeuten Kommunikationsschritte, die nicht unbedingt notwendig sind. Durch dicke und dünne Linien wird zwischen Nachrichten unterschieden, die aus einer ganzen Cache-Zeile oder nur aus einem Befehl und einer Adresse bestehen.

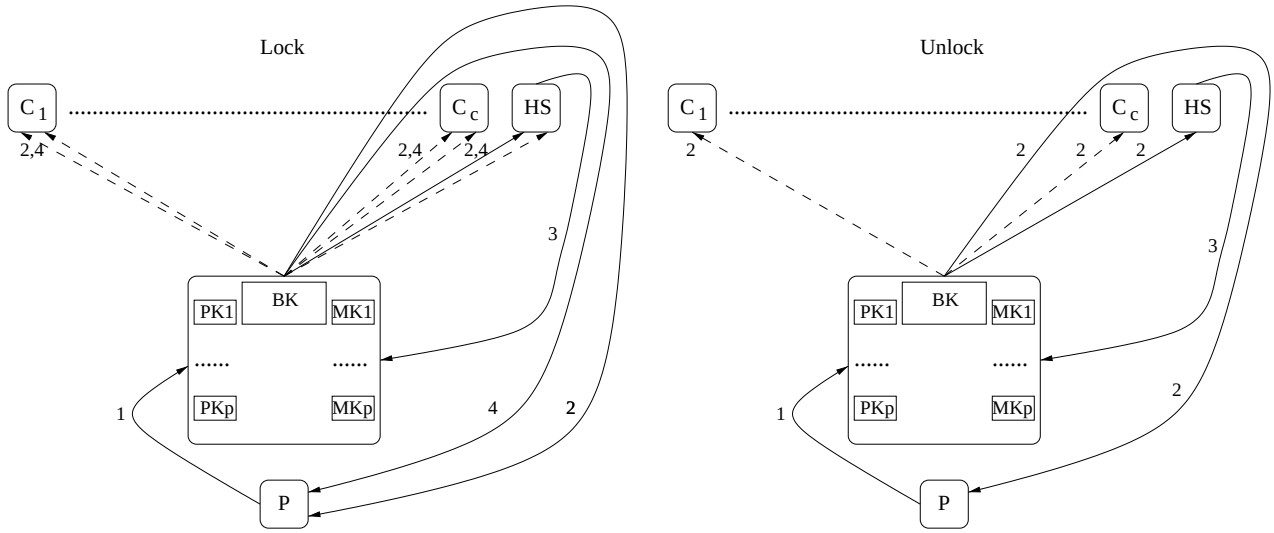


Abbildung 7.3: Der Ablauf von Lock- und Unlock-Operationen auf in der PHOTOBUS-Architektur. Gestrichelte Linien bedeuten Kommunikationsschritte, die nicht unbedingt notwendig sind. Durch dicke und dünne Linien wird zwischen Nachrichten unterschieden, die aus einer ganzen Cache-Zeile oder nur aus einem Befehl und einer Adresse bestehen.

Zeile. Die Speicherbänke ignorieren die Nachricht, da sie von ihr nicht betroffen sind. Die Invalidierung einer Zeile erfolgt in gleicher Weise wie das Anfordern des Schreibrechtes. Es bedarf daher keiner weiteren Erklärung.

WriteBack-Operation

Beim Zurückschreiben in den Speicher schickt der Prozessor die Adresse und den Inhalt der Zeile zusammen mit einem entsprechenden Befehlswort über seinen P-Kanal an den Bus-Chip. Wie in Abbildung 7.2 links unten dargestellt wird die Nachricht dann vom Bus-Chip über den B-Kanal an das ganze System geschickt. Die Speicherbänke überprüfen daraufhin, ob sie die Zeile besitzen, während die Prozessoren die Nachricht ignorieren. Die Speicherbank, die die Zeile beinhaltet, führt den Schreibzugriff durch und benachrichtigt den Prozessor über ihren M-Kanal, sobald sie fertig ist. Dies ist notwendig, damit der Bus-Chip weiß, daß die Bank nun frei ist und neue Anfragen akzeptieren kann.

Synchronisation

Die Kommunikationsstruktur bei der Realisierung von Sperren durch Lock und Unlock- Operationen ist in Abbildung 7.3 zu sehen. Für eine Lock-Operation schickt der Prozessor den Befehl und die Adresse an den Bus-Chip, der beide über den B-Kanal an das ganze System weiterleitet. Die Anfrage wird von der, zur Adresse einer Sperre gehörenden Speicherbank übernommen und bearbeitet. Das Ergebnis wird dem Bus-Chip über den M-Kanal mitteilt und gelangt durch einen Rundruf auf dem B-Kanal zu dem auf Antwort wartenden Prozessor.

Eine Unlock-Operation verläuft genauso wie ein WriteBack. Allerdings werden dabei lediglich der Befehl und die Adresse, und nicht der Inhalt einer gesamten Cache-Zeile verschickt.

7.1.2 Flußkontrolle

Die Flußkontrolle muß dafür sorgen, daß keine Komponente des Systems mehr Anfragen erhält als sie auf einmal verarbeiten kann. Bei der PHOTOBUS-Architektur müssen dabei vor allem drei Faktoren berücksichtigt werden: die Anzahl der Anfragen, die an den Bus-Chip geschickt werden, die Anzahl der an eine Speicherbank gerichteten Anfragen und die Anzahl der Nachrichten, die die Prozessorschnittstellen verarbeiten müssen.

Bus-Chip: Der Bus-Chip besitzt am Eingang eines jeden P- und M-Kanals einen Puffer, der maximal eine Nachricht speichern kann. Dies bedeutet, daß der zugehörige Prozessor bzw. die Speicherbank erst dann eine neue Anfrage an den Bus-Chip schicken kann, wenn die alte verarbeitet wurde. Entscheidend für die Flußkontrolle ist die Tatsache, daß eine Anfrage immer dann als vom Bus-Chip verarbeitet gilt, wenn sie

über den B-Kanal übertragen wurde. Dies bedeutet, daß die Prozessoren und die Speicherbänke mit dem Schicken der neuen Anfrage einfach nur warten müssen, bis sie die alte auf dem B-Kanal bemerken.

Speicherbänke: Die Koordinierung der Anfragen an die Speicherbänke wird zentral vom Bus-Chip erledigt. Dieser weiß für jede Speicherbank, ob sie gerade frei oder beschäftigt ist. Er arbitriert den Zugang zu den Speicherbänken in gleicher Weise, wie er den Zugriff auf den B-Kanal verwaltet.

Prozessoren: Bei den Prozessoren muß zwischen drei Arten von Nachrichten unterschieden werden:

1. den Antworten auf eigene Anfragen,
2. Invalidate- und Exclusiv-Operationen, die im Cache überprüft werden müssen und
3. FetchRead und FetchWrite Anfragen, die eine Supply-Operation erforderlich machen.

Die erste Art von Nachrichten führt trivialer Weise zu keinen Problemen. Unter der Annahme, daß die Bandbreite des B-Kanals nicht größer als die Zugriffsbandbreite des Caches ist, gibt es auch mit der zweiten Nachrichtenart keine Probleme. Das Problem bei den Supply-Operationen besteht darin, daß Supply-Anforderungen bei einem Prozessor schneller über den B-Kanal eintreffen, als die Prozessorschnittstelle über den P-Kanal die angeforderten Daten an den Bus-Chip übertragen kann. Dies liegt daran, daß die Anforderungen lediglich aus der Adresse und einem Befehlswort bestehen, während bei der Durchführung einer Supply-Operation eine ganze Cache-Zeile an den Bus-Chip übertragen werden muß. Hinzu kommt, daß die Bandbreite des P-Kanals geringer als die des B-Kanals ist. Um mit dem Problem fertig zu werden, muß jeder Prozessor einen Puffer für Supply-Operationen besitzen. Wie im nächsten Abschnitt erläutert wird, muß die Größe des Puffers gleich der Prozessoranzahl sein.

7.1.3 Timing der Supply-Operationen

Falls in kurzen Zeitabständen mehrere Supply-Anforderungen an den gleichen Prozessor ergehen, so führt dies an der entsprechenden Prozessorschnittstelle zu einer Stauung von anstehenden Supply-Anfragen. Für die Architektur der Prozessorschnittstelle bedeutet das zum einen, daß ausreichend Puffer-Platz für die Speicherung von solchen Anfragen verfügbar sein muß. Zum anderen muß man dafür sorgen, daß der Bus-Chips trotzt des Staus rechtzeitig über alle Supply-Operationen benachrichtigt wird. Wie in 7.1 erläutert, bedeutet 'rechtzeitig' in diesem Zusammenhang, daß die Benachrichtigung beim Bus-Chip vor der Antwort der Speicherbank eintreffen muß. Anderenfalls würde der Bus-Chip fälschlicher Weise die Daten von der Speicherbank als Ergebnis der Speicheranfrage über den B-Kanal weiterleiten.
Die hier vorgeschlagene Lösung des obigen Problems basiert auf drei Annahmen:

1. Die maximale Anzahl von Supply-Anfragen, die sich gleichzeitig im System befinden können, ist durch die Anzahl der Prozessoren oder ihr kleines Vielfaches beschränkt. Dies folgt aus der Tatsache, daß ein Prozessor nicht unbegrenzt viele ausstehende Speicheranfragen verwalten kann. Die meisten heutigen Prozessoren lassen maximal 2 bis 4 ausstehende Anfragen zu. Darüber hinaus wird der Prozessor so lange angehalten, bis eine der ausstehenden Anfragen befriedigt wurde.

2. Der Cache-Kontroller kann die Supply-Anfragen mit der gleichen Rate befriedigen, mit der sie über den B-Kanal ankommen. Bei dieser Annahme handelt es sich um eine Konkretisierung der Annahme vom vorigen Abschnitt, daß der Cache-Kontroller von dem Datenstrom auf dem B-Kanal nicht überfordert wird.

3. Eine kurze Benachrichtigung über eine anstehende Supply-Operation kann über den P-Kanal in der gleichen Zeit an den Bus-Chip verschickt werden, die eine Supply-Anforderung für die Übertragung auf dem B-Kanal benötigt. Diese Annahme ist dadurch begründet, daß eine solche Benachrichtigung nur aus einem Befehlswort und der Nummer der Speicherbank bestehen muß. Sie ist damit wesentlich kürzer als die einer Supply-Anforderung, die zusätzlich zum Befehlswort eine ganze 64-Bit lange Adresse beinhalten muß.

Aus der ersten Annahme folgt, daß der Puffer (Supply-Puffer) mit der Kapazität von P-Cache-Zeilen für die Speicherung ausstehender Supply-Anfragen ausreicht. Durch Annahmen zwei und drei ist es möglich, auch im Fall eines Staus den Bus-Chip rechtzeitig über ausstehende Supply-Operationen zu benachrichtigen. Das Vorgehen hierzu sieht wie folgt aus. Wird über den B-Kanal eine Fetch-Operation übertragen, so wird diese in einen Puffer geholt und an den Cache-Kontroller übergeben. Dieser prüft, ob eine Supply-Anforderung vorliegt, also ob

die betroffene Zeile sich im eigenen Cache im Modified-Modus befindet. Ist dies der Fall, so wird der Inhalt der Zeile in den Supply-Puffer übertragen. Gleichzeitig wird ein entsprechendes Befehlswort zusammen mit der Nummer der zur Adresse der Zeile gehörenden Speicherbank (die Supply-Benachrichtigung) über den P-Kanal verschickt. Falls zu diesem Zeitpunkt andere Daten über den P-Kanal übertragen werden, so wird diese Übertragung unterbrochen und nach der Benachrichtigung fortgesetzt. Entscheidend für die Korrektheit des Verfahrens ist, daß es sich aufgrund der letzten Annahme bei der unterbrochenen Übertragung niemals um eine andere Supply-Benachrichtigung handeln kann.

Die eigentliche Supply-Operation, bei der die Cache-Zeile aus dem Supply-Puffer an den Bus-Chip geschickt wird, folgt zu einem beliebigen späteren Zeitpunkt. Wann dies genau passiert, ist egal, da der Bus-Chip über die anstehende Supply-Operation informiert ist. Er weiß somit, daß er die vom Speicher kommenden Daten nicht weiterleiten darf, sondern auf den aktuellen Inhalt der Cache-Zeile vom Prozessor warten muß.

7.2 Die optische Datenübertragung

Die optische Datenübertragung zwischen dem Bus-Chip und den Prozessoren und Speicherbänken findet mit Hilfe paralleler Faserbündel-Systeme statt. Dabei müssen drei Aspekte berücksichtigt werden:

- Die Prozessoren, Speicherbänke und der Bus-Chip arbeiten nicht synchron. Dies bedeutet, daß es kein globales Taktsignal gibt, mit dem die Datenübertragung koordiniert werden kann. Statt dessen muß den Empfängern explizit mitgeteilt werden, wann gültige Daten an den Leitungen anliegen.

- Die optische Datenübertragung wird mit einer hohen Datenrate (bis 1 bis 2 Gbit/s), dafür aber mit einer geringen Anzahl von Leitungen (8 bis 10) durchgeführt. Innerhalb des Bus-Chips und auf den Prozessor- und Speicherplatinen wird dagegen mit einer geringeren Datenrate (100 bis 500MHz), dafür aber mit breiteren Datenpfaden gearbeitet. Die Empfänger und Sender müssen also eine Konvertierung zwischen einem hochfrequenten Datenstrom auf einem schmalen Datenpfad und einem niederfrequenten Strom auf einem breiten Datenpfad durchführen.

- Es muß ein Format für die Datenpakete definiert werden, das eine eindeutige Unterscheidung zwischen Befehlen, Adressen und Daten ermöglicht.

7.2.1 Synchronisation der Übertragung

Für eine fehlerfreie Datenübertragung ist es notwendig, daß der Empfänger weiß, wann gültige Daten an der Leitung anliegen und übernommen werden können. In einem synchronen System wird zu diesem Zweck die Übertragung in der Regel mit dem globalen Systemtakt synchronisiert. Dabei geht der Empfänger davon aus, daß er die an der Leitung anliegenden Daten mit der Taktflanke in ein Flip-Flop einlesen kann. Da es in einem asynchronen System keinen globalen Takt gibt, müssen für die Synchronisation andere Verfahren verwendet werden. Dabei gibt es grundsätzlich zwei Möglichkeiten: Es kann entweder ein Taktsignal in den Datenstrom integriert werden oder es kann eine zusätzliche Taktleitung verwendet werden. Die meisten konventionellen optischen Übertragungssysteme verwenden das erste Verfahren, da sie mit einer einzigen Leitung auskommen müssen. Dies hat den Nachteil, daß eine aufwendige Kodierung und Dekodierung notwendig ist, die zu einer hohen Latenz führt. Das PHOTOBUS-System nutzt die hohe Anzahl von Leitungen, die die parallelen Fasersysteme zur Verfügung stellen und verwendet das zweite Verfahren.

7.2.2 Konvertierung des Datenstroms

Die Konvertierung zwischen hochfrequenten und niederfrequenten Datenströmen ist eine Standardfunktion, die in jedem optischen Datenübertragungssystem zu finden ist. Dabei wird für jede Leitung des hochfrequenten Stromes ein Schieberegister benutzt, auf das sequentiell mit hoher, und parallel mit geringer Frequenz zugegriffen werden kann. Die hochfrequenten Daten werden seriell in das Register eingeschoben und, sobald das Register voll ist, parallel ausgelesen. In umgekehrter Richtung werden die Daten parallel in das Register eingelesen und dann mit hoher Frequenz seriell herausgeschoben. Das obige Verfahren setzt voraus, daß auf die Schieberegister abwechselnd parallel und seriell zugegriffen wird. Während des hochfrequenten Zugriffs muß also die niederfrequente Übertragung unterbrochen werden und umgekehrt. Dadurch wird ein kontinuierlicher Datenstrom verhindert

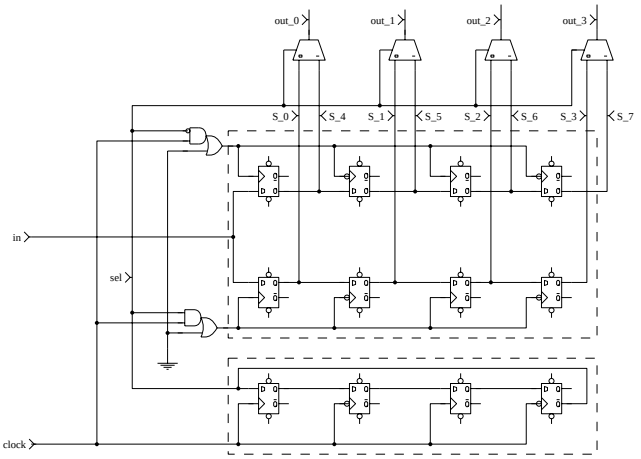


Abbildung 7.4: Die im Text beschriebene Konvertierschaltung, deinen seqientiellen hochfrequenten Datenstrom des optischen Kanals in einen parallelen (in der Abbildung 4 Bit breiten) niederfrequenten Datenstrom wandelt.

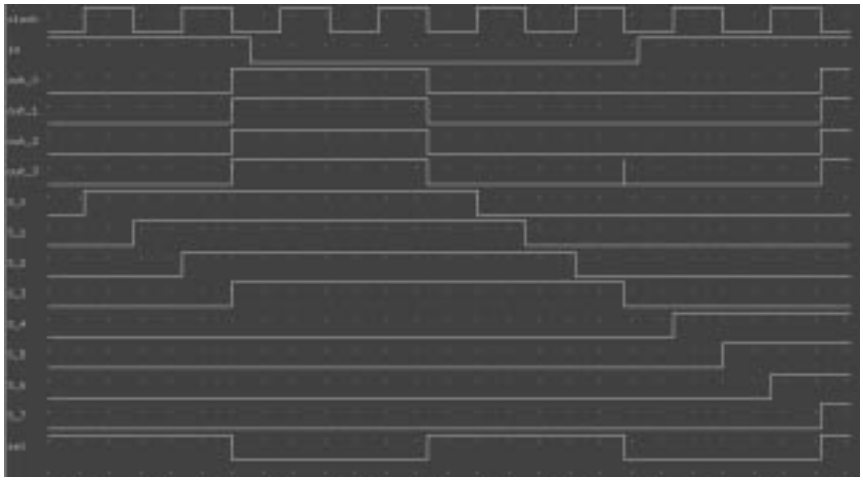


Abbildung 7.5: Das Ergebnis einer Simulation der Konvertierschaltung aus Abbildung 7.4.

und die Bandbreite halbiert. Um einen kontinuierlichen Datenfluß zu ermöglichen, müssen zwei Schieberegister mit der Länge des doppelten Frequenzunterschieds zwischen den Datenströmen verwendet werden. Die hochfrequenten und niederfrequenten Zugriffe finden dann gleichzeitig auf jeweils unterschiedlichen Registern statt. Während eines sequentiellen Zugriffs auf das erste Register findet also ein paralleler Zugriff auf das zweite statt. Da die Länge des Schieberegisters dem Frequenzunterschied entspricht, werden beide Zugriffe gleichzeitig beendet. Daraufhin werden die Register getauscht, es wird also parallel auf das erste und sequentiell auf das zweite zugegriffen.

Bausteine, die auf dieser Weise bis zu 10 hochfrequente Datenströme gleichzeitig auf jeweils 8 bis 16 niederfrequente Ströme reduzieren, sind heute kommerziell erhältlich. Damit kann die Konvertierung zwischen den Datenströmen bei den Prozessor- und Speicherschnittstellen des PHOTOBUS-Systems einfach mit Hilfe von Standardbausteinen verwirklicht werden. Auf dem Bus-Chip muß dagegen für jeden P- und M-Kanal eine entsprechende Schaltung implementiert werden.

Aufbau der Konvertierschaltung

Für die Realisierung der Konvertierschaltung werden vier Arten von Komponenten benötigt:

1. Schieberegister, auf die sowohl sequentiell als auch parallel zugegriffen werden kann,

2. eine Schaltung, die bestimmt, aus welchem Register die parallele Ausgabe stattfindet,

3. eine Schaltung zum Umschalten des sequentiellen Eingabestroms zwischen den beiden Registern,

4. einen Mechanismus, der erkennt, wenn ein Register vollständig beschrieben wurde.

Das Prinzip der Schaltung wird in Abbildung 7.4 für ein 4-Bit-System verdeutlicht. Die Schieberegister werden mit Hilfe von hintereinander geschalteten Flip-Flops verwirklicht. Sie sind in der Abbildungsmitte in der großen Umrandung zu sehen. Der Eingang des Ersten Flip-Flops jedes der beiden Register ist mit dem sequentiellen Dateneingang verbunden. Die Ausgänge der einzelnen Flip-Flops sind über Multiplexer mit den parallelen Datenausgängen verbunden, wobei jeweils der Ausgang des i-ten Flip-Flops jedes Registers zur i-ten Datenleitung geht. Die CLOCK-Eingänge aller Flip-Flops sind über eine Auswahlschaltung an die Taktleitung angeschlossen. Damit Daten sowohl mit der steigenden als auch mit der fallenden Flanke übernommen werden können, ist der CLOCK-Eingang bei jedem zweiten Flip-Flop invertiert. Die Auswahlschaltung sorgt dafür, daß in Abhängigkeit von dem Wert eines Auswahlsignals (sel) ein Register tatsächlich an das Taktsignal angeschlossen wird, während das andere mit einem konstanten 0-Signal verbunden wird. Damit wird bestimmt, wann welches Register durch den sequentiellen Datenstrom beschrieben wird. Das Auswahlsignal ist dasselbe, das für die Ausgangs-Multiplexer benutzt wird. Dabei sind die Auswahlschaltungen und Multiplexer so konfiguriert, daß immer ein anderes Register mit der parallelen Ausgabe und mit der sequentiellen Eingabe verbunden ist. Für die Erzeugung des Auswahlsignals ist ein weiteres Schieberegister zuständig, das als Zähler betrieben wird. Dieses Register ist im unteren Teil der Abbildung 7.4 in der kleinen Umrandung zu sehen. Das Register ist mit den Datenregistern identisch, wird aber nicht an den Eingabe-Datenstrom angeschlossen. Statt dessen wird sein Eingang mit seinem invertierten Ausgang verbunden. Außerdem sind die CLOCK-Eingänge der Flip-Flops direkt mit dem Taktsignal (bzw. dem invertierten Taktsignal) verbunden. Dadurch ändert der Ausgang des Registers seinen Zustand nach so vielen Taktsignalen, wie zum Beschreiben eines Datenregisters notwendig sind.

Zur Verdeutlichung der Funktionsweise der Konvertierschaltung ist in Abbildung 7.5 das Ergebnis einer digitalen Simulation der Schaltung aus Abbildung 7.4 dargestellt. Dabei wurde als Eingabe das Bitmuster `1111 0000 0000 1111` verwendet. In der Abbildung ist der Verlauf des Taktsignals (`CLOCK`), des Eingabesignals (`in`), des parallelen Ausgabesignals (`out0` bis `out3`), der Ausgabesignale der einzelnen Flip-Flops der Datenregister (`s0` bis `s3` für das erste und `s4` `s7` für das zweite Register) und des Auswahlsignals (`sel`) dargestellt. Anfangs sind alle Register und damit alle Ausgangssignale auf 0. Innerhalb der ersten zwei Taktzyklen werden die vier Einsen in das erste Register eingelesen. Dabei bleiben die parallelen Ausgänge unverändert, da das Auswahlsignal die ganze Zeit den Wert 1 hat, und somit die Daten aus dem zweiten Register zum Ausgang leitet. Mit der dritten Taktflanke geht das Auswahlsignal auf 0, wodurch die Register umgeschaltet werden. Die serielle Eingabe wird in das zweite Register geleitet, während die parallelen Ausgänge den Inhalt des ersten Registers anzeigen (`1111`). Dieser Vorgang wiederholt sich nun noch drei Mal mit den verbleibenden Bitfolgen `0000`, `0000` und `1111`.

7.2.3 Format der Nachrichten

Wie in 7.1 beschrieben, können die Nachrichten in einem PHOTOBUS-System unterschiedliche Länge und Struktur haben. Es wird daher ein Mechanismus benötigt, um den Empfängern das Erkennen von Ende und Anfang einer Nachricht zu ermöglichen. Außerdem müssen die Systemkomponenten wissen, wie sie die einzelnen Teile der Nachrichten zu interpretieren haben. Dabei muß auch berücksichtigt werden, daß die Übertragung einer Nachricht durch eine Supply-Benachrichtigung mitten drin unterbrochen werden kann.

Das hier vorgeschlagene Verfahren benutzt eine zusätzliche Leitung, um Befehlsworte zu markieren. Es basiert auf der Beobachtung, daß die Länge und der Aufbau einer Nachricht nur von der Art der Operation abhängt. So bestehen Supply-Benachrichtigungen lediglich aus einem Befehlswort und einer Speicherbank-Nummer, Invalidate-, Exclusive- und Synchronisations-Operationen aus einem Befehlswort und einer Speicheradresse, und schließlich FetchRead- FetchWrite-, WriteBack- und Supply-Operationen aus einem Befehl, einer Adresse und dem Inhalt einer Cache-Zeile. Wenn der Empfänger also ein Befehlswort an Hand der gesetzten Befehlsleitung identifiziert und interpretiert hat, dann weiß er, wie lang der Rest der Nachricht ist und wie er zu interpretieren ist. Falls zwischendurch eine Supply-Benachrichtigung eingeschoben wird, so kann dies ebenfalls an Hand der Befehlsleitung und des Kommandos erkannt und berücksichtigt werden.

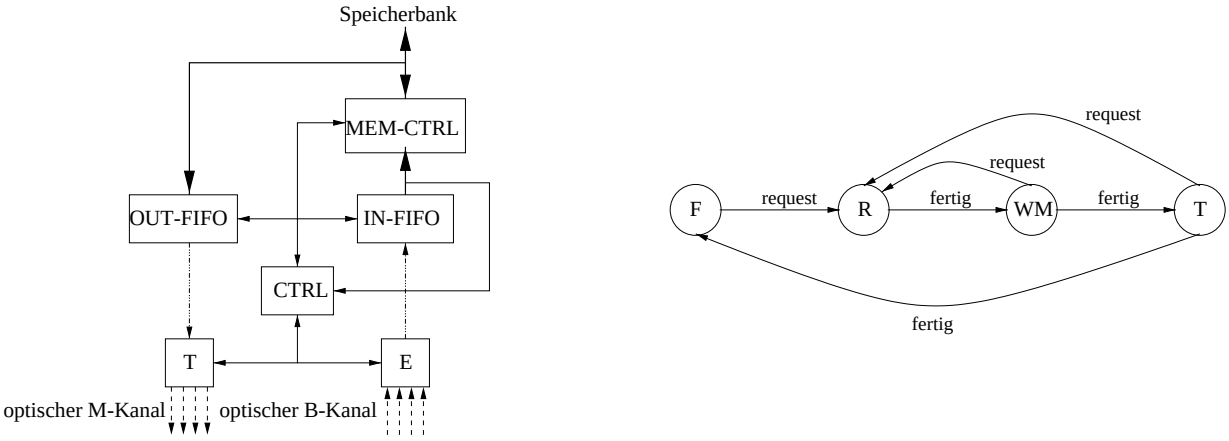


Abbildung 7.6: Der Aufbau (links) und das Zustands-Diagramm (rechts) der Speicherschnittstelle

7.3 Architektur der Speicherschnittstelle

Die Speicherschnittstelle hat drei Aufgaben: sie muß für die Optik-Elektronik Wandlung auf dem optischen M- und B-Kanal sorgen, vom B-Kanal die an ihre Speicherbank gerichteten Anfragen herausfiltern, und die DRAM-Verwaltung bewerkstelligen. Der Aufbau einer Schnittstelle, die obiges leistet, ist in Abbildung 7.6 dargestellt.

Für die Wandlung zwischen optischen und elektrischen Datenströmen sind ein optischer Faserbündel-Empfänger und -Sender sowie zwei spezial-FIFOs zuständig. Letztere sind notwendig, um den niederfrequenten, 64-128 Bit breiten TTL-Datenstrom von der Speicherbank in einen hochfrequenten 8 bis 12 Bit breiten ECL-Datenstrom zu wandeln, der von den Sendern und Empfängern benötigt wird. Die FIFOs sind an einem Speicherkontroller und einer Kontrolleinheit angeschlossen. Der Speicherkontroller ist für die Kommunikation mit der Speicherbank verantwortlich. Die Kontrolleinheit koordiniert die Funktion der übrigen Komponenten. Gleichzeitig hört sie den Datenverkehr auf dem B-Kanal ab und vergleicht die Adressen der Anfragen mit dem Adreßbereich der Speicherbank. Sobald eine Übereinstimmung entdeckt wird, veranlaßt diese den Speicherkontroller, die Anfrage an den Speicher weiterzuleiten. Falls zu diesem Zeitpunkt eine andere Anfrage bearbeitet wird, wird die Bearbeitung abgebrochen. Die Speicherschnittstelle geht in diesem Fall davon aus, daß die alte Anfrage bereits durch ein Supply von einem Prozessor befriedigt wurde. Wie in Abschnitt 7.5 erklärt wird, würde der Bus-Chip anderenfalls keine Anfrage an eine bereits beschäftigte Speicherbank zulassen.

Die Funktionsweise der Speicherschnittstelle kann formal durch ein Zustands-Übergangsdiagramm mit vier Zuständen dargestellt werden. Dies ist in Abbildung 7.6 geschehen. Die dort dargestellten Zustände haben die folgende Bedeutung:

Frei (F): Die Speicherschnittstelle befindet sich im F-Zustand, wenn sie keine Anfrage zu bearbeiten hat. Sie verläßt ihn und wechselt in den R-Zustand, sobald auf dem B-Kanal eine an sie gerichtete Anfrage festgestellt wird.

Speicheranfrage (R): Im R-Zustand wird die gerade empfangene Anfrage an die Speicherbank weitergegeben. Sobald die Anfrage von der Speicherbank zur Bearbeitung übernommen wurde, geht die Speicherschnittstelle in den WM-Zustand über.

Warten auf den Speicher (WM): Im WM wartet die Speicherschnittstelle darauf, daß die Speicherbank mit der Bearbeitung der Anfrage fertig wird, um dann in den T-Zustand zu wechseln. Falls während der Wartezeit eine neue Anfrage am R-Kanal ankommt, so wird die alte Anfrage verworfen und die Bearbeitung mit der neuen im R-Zustand fortgesetzt.

Datenübertragung (T): Im T-Zustand wird das Ergebnis der Speicheranfrage über den M-Kanal an den Bus-Chip gesendet. Sobald die Übertragung beendet ist, kehrt die Schnittstelle in den F-Zustand zurück. Die Ankunft einer neuen Anforderung während der Bearbeitung wird genauso wie im WM-Zustand gehandhabt und führt zum Übergang in den R-Zustand.

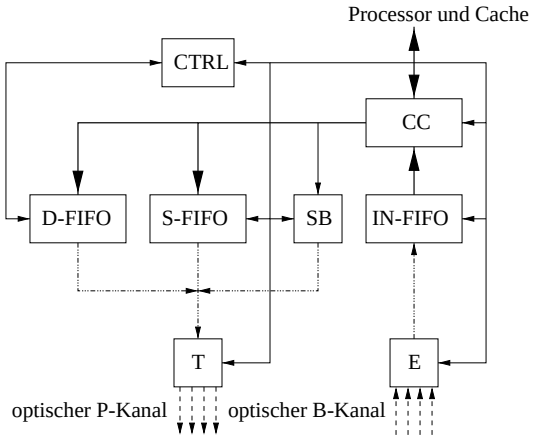


Abbildung 7.7: Der Aufbau der Prozessorschnittstelle.

7.4 Architektur der Prozessorschnittstelle

Die Funktion der Prozessorschnittstelle ist wesentlich komplexer als die der Speicherschnittstelle. Sie umfaßt fünf Aufgaben:

1. Wandlung zwischen optischen und elektrischen Signalen,
2. die Übertragung der Anfragen des Cache-Kontrollers (Fetch-, Invalidate-, Exclusive- und WriteBack-Operationen) auf dem M-Kanal an den Bus-Chip,
3. das Herausfiltern der Antworten auf die Anfragen des Cache-Kontrollers aus dem Datenstrom auf dem B-Kanal,
4. die Weiterleitung der für die Erhaltung der Cache-Kohärenz relevanten Daten (Schreib-Fetch-, Invalidate- und Exclusive- Operationen) vom B-Kanal an den Cache-Kontroller und
5. die Abwicklung von Supply-Operationen nach dem in Abschnitt 7.1.3 beschriebenen Verfahren.

7.4.1 Aufbau der Schnittstelle

Der Aufbau der Prozessorschnittstelle ist ist in Abbildung 7.7 links zu sehen. Für die optische Datenübertragung besitzt sie einen parallelen Fasersender und -empfänger. Die Daten des Empfängers gelangen über ein spezial-FIFO zu der Cache-Kontroller-Schnittstelle. Wie auch schon bei der Speicherschnittstelle, dient der FIFO-Baustein nicht nur der Zwischenspeicherung, sondern vor allem der Konvertierung zwischen einem schmalen (8 bis 12 Bit) hochfrequenten ECL-Datenstrom in einen breiten (64 bis 128 Bit) niederfrequenten TTL-Strom.

Auf der Senderseite gibt es für die gleiche Aufgabe drei unterschiedliche FIFO-Puffer: den Anfrage-Puffer (A-FIFO), den Supply-Puffer (S-FIFO) und den BenachrichtigungsPuffer (B-FIFO). Sie sind für die unterschiedlichen Arten von Nachrichten bestimmt, die über den P-Kanal verschickt werden können: Anforderungen des Prozessors (A-FIFO), ausstehende Supply-Operationen (S-FIFO) und Supply-Benachrichtigungen (B-FIFO). Für die Koordinierung der einzelnen Komponenten ist eine Kontrolleinheit verantwortlich. Sie bestimmt auch, welche der drei Puffer wann Daten an den Transmitter liefern darf.

7.4.2 Funktionsweise der Schnittstelle

Die Arbeitsweise der Prozessorschnittstelle kann man mit Hilfe zweier paralleler Prozesse beschreiben: eines Eingabe-Prozesses und eines Ausgabe-Prozesses. Der Eingabe-Prozeß überwacht den B-Kanal und leitet die relevanten Daten an den Cache-Kontroller und den Ausgabe-Prozeß weiter. Der Ausgabe-Prozeß verwaltet den Transmitter und koordiniert die Übertragung von Prozessoranforderungen, Supply-Operationen und Supply-Anforderungen über den P-Kanal. Die Funktionsweise der beiden Prozesse kann durch die in Abbildung 7.8 dargestellten Zustandsdiagramme eines endlichen Automaten formal spezifiziert werden.

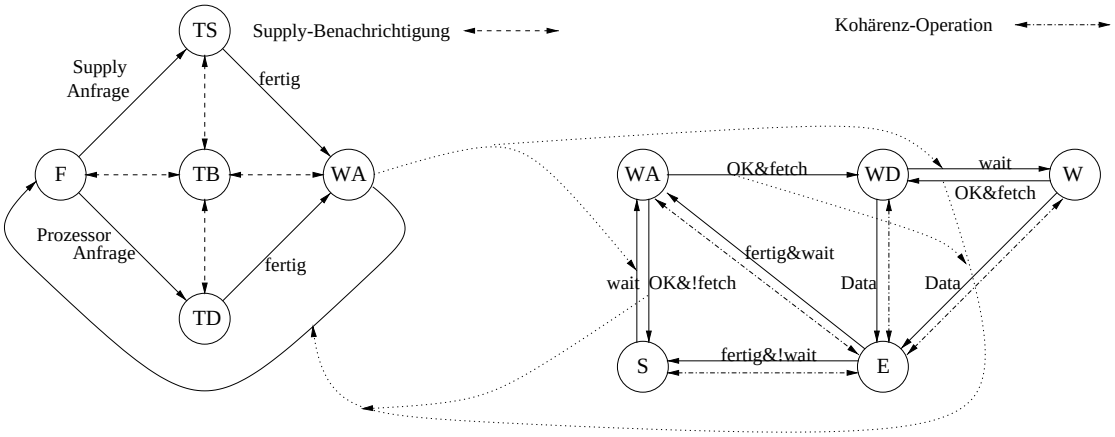


Abbildung 7.8: Die Zustandsdiagramme der beiden Prozesse der Prozessorschnittstelle.

Für den Eingabe-Prozeß werden fünf Zustände benötigt: ein Empfangszustand und vier unterschiedliche Überwachungszustände. Der Empfangszustand dient der Übertragung von Daten vom B-Kanal über das In-FIFO zum Cache-Kontroller. In den Überwachungszuständen beobachtet der Prozeß den B-Kanal und wartet auf für ihn relevante Nachrichten. Um welche Nachrichten es sich handelt, und was bei ihrem Eintreffen passiert, hängt von dem Zustand ab. Die Bedeutung der Zustände des Eingabe-Prozesses kann wie folgt zusammengefaßt werden:

Daten empfangen und weiterleiten (E): Im E-Zustand werden Daten vom B-Kanal empfangen und an den Cache-Kontroller weitergeleitet. Der Prozeß verläßt den E-Zustand sobald die Übertragung der Nachricht beendet ist.

Snoop (S): Der Eingabe-Prozeß ist im S-Zustand, wenn sich seitens des eigenen Prozessors keine Anfragen im System befinden. Er wartet darauf, daß eine Invalidate-, Exclusive- oder Fetch-Anfrage eines anderen Prozessors auf dem B-Kanal übertragen wird. Sobald eine solche Anfrage registriert wird, geht der Prozeß in den E-Zustand über.

Warten auf Antwort (WA): Der WA-Zustand bedeutet, daß der Ausgabe-Prozeß eine Nachricht an den Bus-Chip geschickt hat und nun darauf wartet, daß diese auf dem B-Kanal übertragen wird. Sobald der Eingabe-Prozeß die besagte Anfrage auf dem B-Kanal entdeckt, gibt er dem Ausgabe-Prozeß Bescheid und geht in den S-Zustand zurück. Im WA-Zustand werden genauso wie im S-Zustand die für die Cache-Kohärenz relevanten Operationen berücksichtigt. Entdeckt der Prozeß eine solche Operation auf dem B-Kanal, so wechselt er kurzzeitig in den E-Zustand, gibt sie an den Cache-Kontroller weiter und kommt wieder zum WA-Zustand zurück.

Warten auf Daten (WD): Wenn der Prozessor auf das Ergebnis einer Fetch-Operation wartet, dann befindet sich der Eingabe-Prozeß im WD-Zustand. Sobald die erwarteten Daten auf dem B-Kanal erscheinen, geht der Prozeß in den E-Zustand über und gibt sie an den Cache-Kontroller weiter. Ein Wechsel in den E-Zustand findet auch dann statt, wenn während der Wartezeit eine für die Erhaltung der Cache-Kohärenz relevante Operation auf dem B-Kanal festgestellt wird. Ein solcher Wechsel ist allerdings nur vorübergehend. Sobald die Daten an den Cache-Kontroller weitergegeben wurden, kehrt der Prozeß in den W-Zustand zurück. Aus dem WD-Zustand kann der Eingabe-Prozeß auch in den W-Zustand wechseln. Dies passiert, wenn während der Wartezeit der Ausgabeprozeß eine weitere Nachricht (z.B. eine Supply-Operation) an den Bus-Chip schickt und darauf wartet, daß diese auf dem B-Kanal übertragen wird.

Allgemeines Warten (W): Der W-Zustand ist eine Zusammenfassung der S-, WA- und WD-Zustände. Er bedeutet, daß die Prozessorschnittstelle eine Nachricht an den Bus-Chip abgeschickt hat, während der Prozessor auf das Ergebnis einer Fetch-Anfrage wartete. Der Eingabe-Prozeß kann aus dem W-Zustand entweder in den WD-Zustand zurückkehren oder auf dem Umweg über den E-Zustand in den WA-Zustand übergehen. Ersteres passiert, wenn die an den Bus-Chip geschickte Nachricht über den B-Kanal übertragen wurde, bevor die erwarteten Daten (das Ergebnis der Speicheroperation) eingetroffen ist. Letzeres ist der Fall, wenn die Daten zuerst eintriffe.

Der Ausgabe-Prozeß hat fünf Zustände: einen Ausgangszustand, drei Übertragungszustände und einen Wartezustand. Sie haben die folgende Funktion:

Fertig (F): Der Ausgabe-Prozeß befindet sich im F-Zustand, wenn er entweder gerade eine Übertragung vollständig beendet hat, oder wenn weder Prozessoranfragen noch Supply-Anfragen vorliegen. Er verläßt diesen Zustand, sobald mindestens eine Anfrage vorhanden ist, die über den P-Kanal übertragen werden muß. Dabei hängt der nächste Zustand von der Art der vorhandenen Anfrage ab. Bei einer Supply-Benachrichtigung geht der Prozeß in den TB-Zustand, bei einer Supply-Operation in den TS- und im Falle einer Prozessor-Anforderung in den TA-Zustand über. Für den Fall, daß mehrere unterschiedliche Anfragen vorliegen, werden zuerst Supply-Anforderungen, dann Supply-Operationen und an letzter Stelle Prozessoranfragen bearbeitet.

Übertragen einer Supply-Benachrichtigung(TB): Im TB-Zustand wird eine Supply-Benachrichtigung aus dem B-FIFO über den P-Kanal an den Bus-Chip übertragen. Ein Wechsel in diesen Zustand findet sofort statt, wenn dem Cache-Controller eine Supply-Anforderung vorliegt. Sobald die Übertragung beendet ist, geht der Prozessor in den Zustand über, aus dem er in den TB-Zustand gelangt ist. Falls es sich dabei um einen der anderen Übertragungs-Zustände handelt, dann wird die unterbrochene Übertragung fortgesetzt.

Übertragung von Supply-Daten (TS): Im TS-Zustand wird aus dem S-FIFO der Inhalt der Cache-Zeile an den Bus-Chip geschickt, die durch eine Supply-Anforderung bestellt wurde. Am Ende der Übertragung geht der Prozessor in den WA-Zustand, um auf die Antwort des Bus-Chips zu warten. Falls während der Übertragung eine Supply-Anforderung eintrifft, geht der Ausgabe-Prozeß vorübergehend in den TB-Zustand und sendet eine Supply-Benachrichtigung an den Bus-Chip weiter.

Übertragung einer Prozessor-Anfrage (TA): Im TA-Zustand wird eine Prozessoranfrage (eine Fetch-, Invalidate-, Exclusive- oder WriteBack-Operation) aus dem A-FIFO über den P-Kanal gesendet. Am Ende der Übertragung geht der Prozessor in den WA-Zustand, um auf die Antwort des Bus-Chips zu warten. Im TA-Zustand ist genauso wie im TS-Zustand ein vorübergehender Wechsel in den TB-Zustand möglich. Er wird durch die Ankunft einer Supply-Anforderung ausgelöst.

Warten auf Antwort (WA): Ein Prozessor im WA-Zustand wartet darauf, daß eine von ihm initiierte Anfrage über den B-Kanal übertragen wird. Beim Wechsel in diesen Zustand wird der Eingabe-Prozeß benachrichtigt, daß er in den WA- bzw. W-Zustand (siehe oben) wechseln und auf das Erscheinen entsprechender Daten auf dem B-Kanal warten soll. Der WA-Zustand dient der Flußkontrolle. Er sorgt dafür, daß die Prozessorschnittstelle nur dann Daten an den Bus-Chip schickt, wenn der Puffer des P-Kanals frei ist. Der Ausgabe-Prozeß verläßt den WA-Zustand und geht in den F-Zustand zurück. Sobald der Eingabe-Prozeß die Übertragung der Anfrage auf dem B-Kanal feststellt, gibt er ihm Bescheid und veranlaßt damit eine Rückkehr in den F-Zustand. Aus dem WA-Zustand ist auch ein kurzzeitiger Wechsel in den TB-Zustand möglich, wenn die Übertragung einer Supply-Benachrichtigung notwendig ist.

7.5 Architektur des Bus-Chips

Der Bus-Chip ist das Herz des PHOTOBUS Systems. Er empfängt auf den P- und M-Kanälen Daten von den Prozessoren und den Speicherbänken, speichert sie zwischen und schickt sie durch einen Rundruf über den B-Kanal an das ganze System. Dabei muß er sicherstellen daß

1. zu keinem Zeitpunkt mehr als eine Anfrage an eine bestimmte Speicherbank geschickt wird,
2. jede Prozessoranfrage nach einer endlichen Wartezeit bearbeitet wird und
3. Supply-Operationen korrekt bearbeitet werden.

Der vorliegende Abschnitt beschreibt eine Bus-Chip Architektur die den obigen Anforderungen gerecht wird. Im Nachfolgenden wird zuerst ein Überblick über den Aufbau und die Funktionsweise des Chips gegeben. Als nächstes werden die Arbitrierschaltung sowie der Kontrollmechanismus für die Koordination von Zugriffen auf die einzelnen Speicherbänke und den Transmitter beschrieben. Abschließend werden der Aufbau und die Funktionsweise der einzelnen Einheiten des Chips erläutert.

7.5.1 Überblick

Der Aufbau des Bus-Chips ist schematisch in Abbildung 7.1 dargestellt. Der Chip besitzt eine Transmittereinheit (TE), sowie eine Eingabeeinheit für jeden P- und M-Kanal (PE und ME), die durch einen Kontrollbus (KB) und einen Datenbus (DB) verbunden sind. Die TE bildet die Schnittstelle zum optischen B-Kanal. Sie wird von den MEs und den PEs über den DB mit Daten versorgt. Der KB ist für die Arbitrierung des Transmitters und der Speicherbänke verantwortlich. Er stellt sicher, daß eine Anfrage nur dann über den B-Kanal übertragen werden kann, wenn die betroffene Speicherbank frei ist. Die PEs und MEs sind selbständige Verarbeitungseinheiten, die für die Abwicklung von Prozessoranfragen verantwortlich sind. Sie empfangen die Daten auf den ihnen zugeordneten Kanälen, speichern sie zwischen und führen alle notwendigen Aktionen selbständig durch. Die PEs entscheiden auf der Basis des Befehlswortes, um was für eine Anfrage es sich handelt. Daraufhin fordern sie über den KB den Zugang zum Transmitter und bei Bedarf auch den Zugriff auf eine Speicherbank. Sobald dieser gewährt wird, schicken sie die erforderlichen Daten über den DB an den Transmitter. Falls es sich bei der Anfrage um eine Operation handelt, die den Zugriff auf eine Speicherbank erfordert, wird dies gleichzeitig der zugehörigen ME mitgeteilt. Diese weiß damit, daß sie Daten zu auf ihrem M-Kanal zu erwarten hat. Wenn sie eintreffen, fordert die ME den Transmitter an und leiten die Daten über den DB und an die TE weiter. Am Ende der Übertragung meldet sie über den KB, daß die zugehörige Speicherbank nun für weitere Anfragen frei ist.

Um eine korrekte Durchführung von Supply-Operationen zu ermöglichen, kann eine ME auch durch eine Nachricht von einer PE freigegeben werden. Dies passiert, falls eine PE eine entsprechende Supply-Benachrichtigung empfängt. Sie leitet die Benachrichtigung über den KB an die ME weiter, die sich dann sofort 'frei' meldet und die Daten von der Speicherbank ignoriert.

7.5.2 Der Kontrollbus

Der KB und die dazugehörende Logik sind für drei Dinge zuständig: die Arbitrierung des Transmitters, die Vergabe der Speicherbänke an die PEs und die Kommunikation zwischen den PEs und den MEs. Sie müssen sicherstellen daß:

1. die Arbitrierung verklemmungsfrei ist,
2. bei der Arbitrierung des Transmitters und der Speicherbänke immer eine sinnvolle und faire Vergabe-Reihenfolge eingehalten wird,
3. nach einer Supply-Benachrichtigung die PE in der Lage ist, die betroffene ME ohne Verzögerung zu benachrichtigen.

Gleichzeitig muß der Kontrollmechanismus effizient beschaffen sein und darf nicht zu viele Chip-Ressourcen verbrauchen. Insbesondere dürfen die Latenz der Arbitrierung und die Größe der Schaltung nicht zu stark mit der Prozessorzahl wachsen.

Im Nachfolgenden werden zuerst die Frage der Verklemmungsfreiheit diskutiert, und eine effiziente, verklemmungsfreie Arbitrierungsstrategie beschrieben. Als nächstes wird gezeigt, wie ein Arbitrierungmechanismus implementiert werden kann, der auch bei hoher Prozessorzahl effizient und einfach bleibt und dabei eine faire Arbitrierung garantiert. Daraufhin werden der Gesamtaufbau des Kontrollbusses beschrieben und der Ablauf einer Bustransaktion erläutert.

Die Verklemmungsfreiheit der Arbitrierung

Eine Verklemmung liegt dann vor, wenn zwei oder mehr Instanzen gegenseitig aufeinander warten und blockiert sind, wobei keine ihre Arbeit fortsetzen kann, ohne daß mindestens eine der anderen zuerst mit ihrer Arbeit fortfährt. Bei der Arbitrierung können in einem System Verklemmungen in zwei Fällen entstehen: bei der gleichzeitigen Arbitrierung mehrerer Ressourcen und wenn es zwischen den Instanzen, die sich um die gleiche Ressource bewerben, zusätzlich noch andere Abhängigkeiten bestehen.

Erstes führt dann zu Problemen, wenn eine Instanz eine Ressource besitzt, die sie solange nicht freigibt, bis sie den Zugriff auf eine andere Ressource bekommt. Falls die zweite Ressource aber von einer Instanz gehalten wird, die auf die erste Ressource wartet, dann werden beide Instanzen blockiert und es kommt zu einer Verklemmung. Eine solche Verklemmung kann auch durch transitive Abhängigkeiten mehrere Instanzen verursacht werden. Bei dem Bus-Chip kann diese Situation dann auftreten, wenn zwei PEs Daten an die gleiche Speicherbank schicken

möchten. Dazu benötigen sie beide sowohl den Zugriff auf die Speicherbank als auch auf den Transmitter. Eine Verklemmung könnte dabei dann auftreten, wenn eine PE den Transmitter bekommt, während der anderen der Zugriff auf die Speicherbank gewährt würde. Dies führt dazu, daß die erste PE blockiert ist bis die zweite die Speicherbank freigibt, während die zweite darauf wartet, daß sie von der ersten den Transmitter bekommt.

Der zweite Fall liegt dann vor, wenn eine Instanz die Dienste einer anderen benötigt, und zusätzlich beide Instanzen sich um den Zugriff auf die gleiche Ressource bewerben. Eine Verklemmung kommt in dieser Konstellation vor, wenn sich die erste Instanz im Besitz der Ressource befindet, die zweite auf die Ressource wartet, und gleichzeitig die erste zum Fortsetzen ihrer Arbeit auf die zweite angewiesen ist. Eine solche Situation könnte sich bei dem Bus-Chip bei der Vergabe des Transmitters an die MEs und PEs ergeben. Dies liegt daran, daß eine Speicherbank unter Umständen erst dann freigegeben werden kann, wenn ihre Daten von der ME über den B-Kanal weitergeleitet wurden (siehe Abschnitt 7.5.4). Damit bewirbt sich sowohl die PE als auch die Speicherbank (vertreten durch die ME) um den Transmitter, während die PE gleichzeitig den Zugriff auf die Speicherbank benötigen kann. Eine Verklemmung würde dann vorkommen wenn die ME einer Speicherbank auf den Transmitter wartet, und eine PE den Transmitter besitzt, die auf den Zugriff auf die zu der ME gehörende Speicherbank wartet.

Beide Arten von Verklemmungen lassen sich im Bus-Chip dadurch vermeiden, daß die Transmitter-Arbitrierung mit der Vergabe der Speicherbänke koordiniert wird. Dabei bewerben sich die PEs zuerst um die Speicherbänke. Eine Transmitter-Anfrage stellen nur die, denen entweder der Zugriff auf die Speicherbank gewährt wurde, oder die, die keine Speicherbank benötigen. Letzteres ist dann der Fall, wenn eine PE eine Exclusive-, Invalidate- oder Supply-Operation zu bearbeiten hat.

Die Arbitrierungslogik

Das Ziel der Arbitrierung besteht darin, den Zugriff auf eine Ressource, um die sich mehrere Instanzen bewerben, fair und effizient zu regeln. Dabei bedeutet effizient, daß die Latenz der Arbitrierung möglichst gering im Vergleich zur durchschnittlichen Nutzungszeit der Ressource sein sollte. Die Arbitrierung erfolgt in der Regel in drei Schritten. Zunächst wird durch die Arbitrierungslogik aus den vorliegenden Anfragen eine ausgewählt. Als nächstes wird der Urheber der Anfrage benachrichtigt. Sobald dieser die Ressource nicht mehr benötigt, gibt er sie frei und schickt dem Arbiter eine Benachrichtigung. Daraufhin initiiert der Arbiter den nächsten Arbitrierungszyklus. Die Latenz der Arbitrierung setzt sich damit aus zwei Faktoren zusammen: aus der Auswahlzeit und der Übertragungszeit für die beiden Benachrichtigungen.

Die Anforderungen beim Entwurf eines Arbitrierungs-Mechanismus für den Bus-Chip unterschieden sich in zwei Punkten von den Anforderungen an den Arbiter eines konventionellen Busses. Zum einen ist die durchschnittliche Nutzungszeit um eine bis 2 Größenordnungen geringer. Eine Transaktion auf einem konventionellen Bus dauert bis zu 1μs. Durch die hohe Bandbreite des B-Kanals sowie die Anwendung des Paketvermittlungsverfahrens liegen hingegen die Nutzungszeiten auf dem Bus-Chip im Bereich weniger Prozessorzyklen. Das heißt, daß ein sehr schnelles Arbitrierungsverfahren benötigt wird. Hinzu kommt, daß die Latenz der Arbitrierung im Bus-Chip im wesentlichen durch die Auswahlzeit bestimmt wird. Dies liegt daran, daß die Benachrichtigungen lediglich innerhalb des Chips verschickt werden. Die Leitungsverzögerungen liegen damit unterhalb eines Prozessorzyklus. In einem konventionellen Bus hingegen werden die Benachrichtigungen über 10 bis 100 cm lange Leitungen auf einer Platine übertragen. Die Leitungsverzögerungen sind so erheblich größer als die Schaltzeit der Logik. Sie ist dadurch für die Latenz der Arbitrierung unerheblich.

Um die benötigte Effizienz zu erreichen, wird für den Bus-Chip eine Variante des Daisy-Chain ([79]) Verfahrens vorgeschlagen, das auf einer sog. 'wired-or' Leitung basiert. Dabei geht eine, mit der Spannungsversorgung verbundene Leitung durch alle Knoten durch (Abbildung 7.9). Jeder Knoten kann den Ausgang der Leitung entweder mit dem Eingang oder mit Masse verbinden. Dies geschieht mit Hilfe eines Transmission-Gates. In einem Arbitrierungsschritt schalten alle die Knoten den Ausgang der Leitung auf Masse, die die Ressource anfordern wollen. Danach hat der erste der anfragenden Knoten an seinem Eingang eine Spannung anliegen, während alle nachfolgenden mit Masse verbunden sind. Damit steht dieser Knoten als Gewinner fest und kann auf die Ressource zugreifen. Sobald er sie nicht mehr benötigt, verbindet er den Ausgang der Leitung wieder mit dem Eingang. Damit liegt die Spannung am Eingang des nächsten wartenden Knoten an.

Der Nachteil dieses Verfahrens besteht darin, daß es nicht fair ist. So kann der erste Knoten die Ressource immer bekommen. Der letzte bekommt sie dagegen nur dann, wenn keine anderen Anfragen im System vorhanden sind. In einem System mit vielen Anfragen kann es dazu führen, daß die hinteren Knoten die Ressource nie bekommen. Um dieses Manko zu beseitigen, wird die Arbitrierung in zwei Schritten durchgeführt. Im ersten Schritt werden

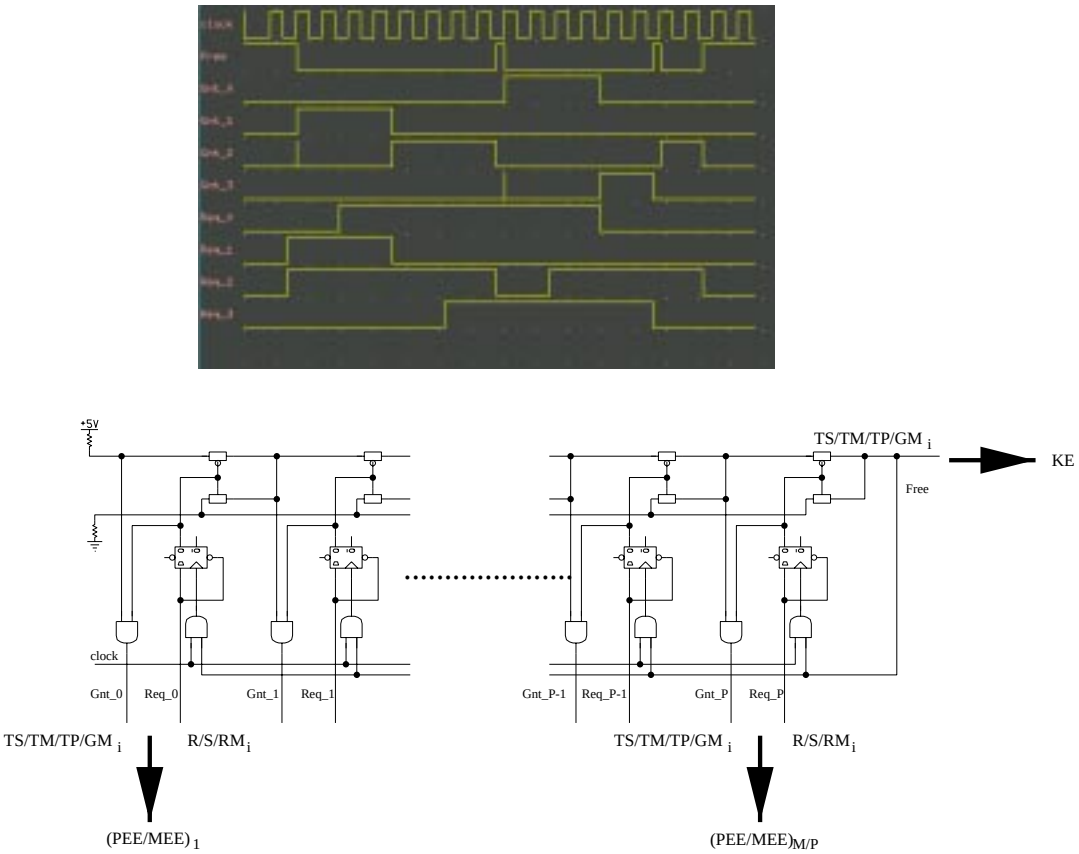


Abbildung 7.9: Das Prinzip der 'wired-or' Arbitrierung. Das untere Bild zeigt den Aufbau der Arbitrierschaltung. Darüber ist das Ergebnis einer Simulation des Arbitriervorgangs mit vier Einheiten dargestellt.

alle Anfragen gespeichert. Im nächsten Schritt wird dann eine Arbitrierung nach dem oben genannten Prinzip durchgeführt. Dabei werden allerdings nur die Anfragen berücksichtigt, die im ersten Schritt gespeichert wurden. Neue Anfragen werden erst dann berücksichtigt, wenn alle gespeicherten befriedigt wurden und die Arbitrierung wieder mit Schritt 1 fortgesetzt wird. Um die zweistufige Arbitrierung zu implementieren, besitzt jeder Knoten einen Flip-Flop, in dem er eine Anfrage speichern kann. Die CLOCK-Leitung der Flip-Flops ist über ein AND-Gatter mit dem Ausgang des letzten Knotens verbunden. Dies hat zur Folge, daß die Flip-Flops nur dann Daten speichern, wenn alle Knoten ihre Ausgänge mit dem Eingang verbunden haben. Dies ist wiederum genau dann der Fall, wenn alle Anfragen befriedigt wurden. Zusätzlich ist der Reset Eingang des Flip-Flops so konfiguriert, daß er gelöscht wird, sobald der Knoten die Ressource freigibt. Die Funktionsweise der Arbitrierschaltung ist in Abbildung 7.9 oben an Hand einer Simulation mit vier Knoten zu sehen.

Der Aufbau des Kontrollbusses

Das im vorigen Abschnitt beschriebene Verfahren ermöglicht die Arbitrierung einer Ressource mit einer 'wired-or' Arbitrierleitung. Auf dem Bus-Chip müssen ein Transmitter und M Speicherbänke arbitriert werden. Daher besitzt der KB insgesamt $M + 1$ solcher Leitungen. Zusätzlich wird ein Mechanismus benötigt, um die Arbitrierung der Speicherbänke mit der Transmitter-Arbitrierung zu koordinieren und mit der Bearbeitung der Supply-Benachrichtigung zu koppeln. Das Hauptproblem liegt dabei darin, daß die Speicherbänke anders als der Transmitter nicht von derselben Einheit freigegeben werden, die sie angefordert und als besetzt gekennzeichnet hat. Wie bereits beschrieben, wird eine Speicherbank von der PE als besetzt gekennzeichnet, die eine Anfrage an diese Speicherbank hat. Freigegeben wird sie dagegen entweder von der zugehörigen ME oder von der PE, die eine Supply-Benachrichtigung empfangen hat. Um eine solche Trennung der Inbesitznahme und der Freigabe des Transmitters zu ermöglichen, besitzt der KB für jede ME zusätzlich zu der Arbitrierleitung eine 'besetzt'- und eine 'frei'-Leitung.

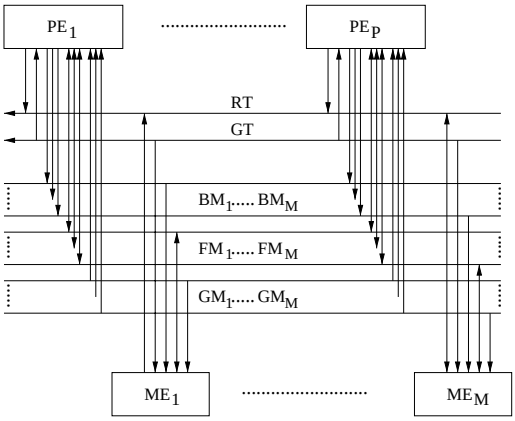


Abbildung 7.10: Der Aufbau des Konrollbusses (KB).

Der Aufbau des KB ist Abbildung 7.10 zu sehen. Die Arbitrierleitungen sind dort als (für Grant-Transmitter) und als GM_i (für grant memory i) gekennzeichnet. Die 'frei'- und 'besetzt'-Leitungen der MEs werden als FM_i (für free memory i) und BM_i (für busy memory i) bezeichnet. An die GT-Leitung sind alle PEs und alle MEs angeschlossen. Es gibt an der GT-Leitung also für jede PE und jede ME einen Transmission-Gate Schalter, der wie oben beschrieben über einen Flip-Flop gesteuert wird. Der Eingang des Flips-Flops ist mit dem RT (request transmitter) Ausgang der PE bzw. ME verbunden. Das von der Leitung kommende Grant-Signal geht in den GT (grant transmitter) Eingang. Analog sind die PEs an die GM-Leitungen angeschlossen. Dabei hat jede PE einen RMi-Ausgang (für request memory i) und einen GMi (für grant memory) Eingang für jede Speicherbank. Die BM-Leitungen können von jeder PE gesetzt werden und werden lediglich von der zugehörigen ME gelesen. Die FM-Leitungen sind dagegen echte Busleitungen, die sowohl von den PEs als auch von der ME gelesen und gesetzt werden.

Die Arbitrierung des Transmitters und der Speicherbänke der obigen Leitungen wird wie folgt durchgeführt:

Transmitter-Arbitrierung: Die Transmitterarbitrierung funktioniert exakt nach dem im vorigen Abschnitt beschriebenen Prinzip. Die PEs und MEs fordern den Transmitter durch das Setzen ihrer RT-Leitung auf 1. Sie werden über die Zuteilung des Transmitters über die GT-Leitung informiert. Diese geht auf 1 wenn der betroffenen Einheit der Zugang zum Transmitter gewährt wird. Sobald die Einheit den Transmitter nicht mehr benötigt, setzt sie ihre RT-Leitung wieder auf 0.

Speicherbank-Arbitrierung: Bei der Vergabe der Speicherbänke, muß wie bereits angedeutet, zwischen der eigentlichen Arbitrierung auf der einen Seite, und der Inbesitznahme sowie Freigabe auf der anderen unterschieden werden. Die Arbitrierung erfolgt wie beim Transmitter, wobei an Stelle der RT- und GT-Leitungen die zur benötigten Speicherbank korrespondierenden RM- und GM-Leitungen verwendet werden. Allerdings bedeutet eine 1 auf der GM-Leitung dabei nicht, daß die PE sofort auf die Speicherbank zugreifen kann. Sie bedeutet statt dessen, daß ein Zugriff möglich ist, sobald die Speicherbank durch eine 1 auf der korrespondierenden Leitung als frei gekennzeichnet wird. Außerdem erfolgen die Inbesitznahme und die Freigabe nicht mit Hilfe der RM-, sondern mit der BM- und der FM-Leitungen. Die BM-Leitung wird von den PE benutzt, um einer ME mitzuteilen, daß eine Anfrage an die zugehörige Speicherbank geschickt wurde. Die ME weiß dadurch, daß sie auf Daten von ihrer Speicherbank warten muß. Außerdem setzt sie die zu ihr gehörende FM-Leitung auf 0. Damit zeigt sie dem System, daß sie besetzt ist. Die FM-Leitung einer ME kann entweder durch die ME selbst oder durch eine PE auf 1 gesetzt werden. Ersteres geschieht, wenn die ME die erwarteten Daten von der Speicherbank bekommen hat, und sie an den Transmitter des B-Kanals weitergegeben hat. Letzteres passiert, wenn eine PE eine Supply-Benachrichtigung empfängt, die sich auf die Speicherbank der ME bezieht.

7.5.3 Ablauf einer PHOTOBUS Transaktion

Eine Transaktion läuft auf dem Bus-Chip in zwei Schritten ab. Zunächst zeigen die PEs und MEs durch das Setzen entsprechender Kontrolleitungen, in welchem Zustand sie sich befinden und welche Ressource sie bekommen

möchten (Anfragephase). Im nächsten Schritt werden die Ressourcen vergeben, wobei die Zuteilung durch entsprechende Grant- und Frei-Leitungen den PEs und MEs mitgeteilt (Vergabephase) wird. Diese ändern daraufhin ihre Zustände und zeigen dies in der nächsten Anfragephase an. Jede Phase dauert einen Taktzyklus. Sie beginnt mit der fallenden Taktflanke und wird durch die steigende Flanke in zwei Unterabschnitte geteilt: den Signal- und den Reaktionsabschnitt. Im ersten Abschnitt werden alle Signale gesetzt und mit der steigenden Taktflanke von allen Komponenten des Bus-Chips übernommen. Sie werden dann im Reaktionsabschnitt ausgewertet und veranlassen die betroffenen Komponenten zu einer Zustandsänderung. Diese wiederum bestimmt, welche Signale sie in der nächste Phase setzen.

1. Anfragephase Signale:
   - Alle PEs, die den Zugriff auf eine Speicherbank benötigen, setzen die korrespondierende RM-Leitung auf 1.
   - Alle PEs und MEs, die Daten über den B-Kanal übertragen wollen, setzen ihre RT-Leitung auf 1.
   - Alle PEs, denen in der vorhergehenden Vergabephase der Zugriff auf die angeforderte Speicherbank gewährt wurde, setzen die korrespondierende BM-Leitung auf 1. Gleichzeitig wird die RM-Leitung wieder auf 0 gesetzt.
   - Alle PEs, die eine Supply-Benachrichtigung empfangen haben, setzen die FM-Leitung der ME der zugehörigen Speichereinheit auf 1.
   - Die PE oder ME, der in der vorhergehenden Vergabephase der Zugriff auf den Transmitter gewährt wurde, fängt mit der Datenübertragung an.
   - Falls der Besitzer des Transmitters diesen nicht mehr benötigt, setzt er die RT-Leitung wieder auf 0.
2. Anfragephase Reaktionen:
   - Jede ME, deren BM-Leitung auf 1 gesetzt wurde, betrachtet sich als besetzt.
   - Jede besetzte ME, deren FM-Leitung auf 1 gesetzt wurde, geht in der Freizustand über
3. Vergabephase Signale:
   - Alle freien MEs setzen ihre FM-Leitung auf 1, die Besetzten auf 0.
   - Die GT-Leitung der PE bzw. ME, die bei der Arbitrierung des Transmitters gewonnen hat, geht auf 1.
   - Falls eine PE die Arbitrierung einer Speicherbank gewonnen hat, geht seine zugehörige GM-Leitung auf 1.
4. Vergabephase Reaktionen:
   - Ob eine PE oder ME auf den Transmitter zugreifen darf, entscheiden die GT-Leitung der ME bzw. PE. Ist sie 1, so weiß die PE oder ME, daß sie auf den Transmitter zugreifen darf. Anderenfalls ist jemand anderes im Besitz des Transmitters.
   - Der Zugriff auf eine Speicherbank wird durch die zu der Speicherbank korrespondierenden GM- und FM-Leitungen geregelt. Ist die zu einer Speicherbank gehörende GM-Leitung einer PE auf 1, und wird gleichzeitig durch eine 1 auf der FM-Leitung die Speicherbank als frei gekennzeichnet, so weiß die PE, daß sie auf die Speicherbank zugreifen darf. Falls lediglich die GM-Leitung der Speicherbank auf 1 ist, so weiß sie, daß sie als nächstes dran ist, muß aber mit dem Zugriff noch warten.

7.5.4 Aufbau und Funktion einer ME

Die ME ist für zwei Dinge zuständig: sie muß den Zugriff auf die zugehörige Speicherbank koordinieren und die von der Speicherbank kommenden Daten an die Prozessoren weiterleiten. Dazu muß sie

1. durch ihre FM_i-Leitung den Zustand der Speicherbank anzeigen,
2. die ihr zugeordnete BM_i-Leitung überwachen, und bei Bedarf die Speicherbank als besetzt kennzeichnen,

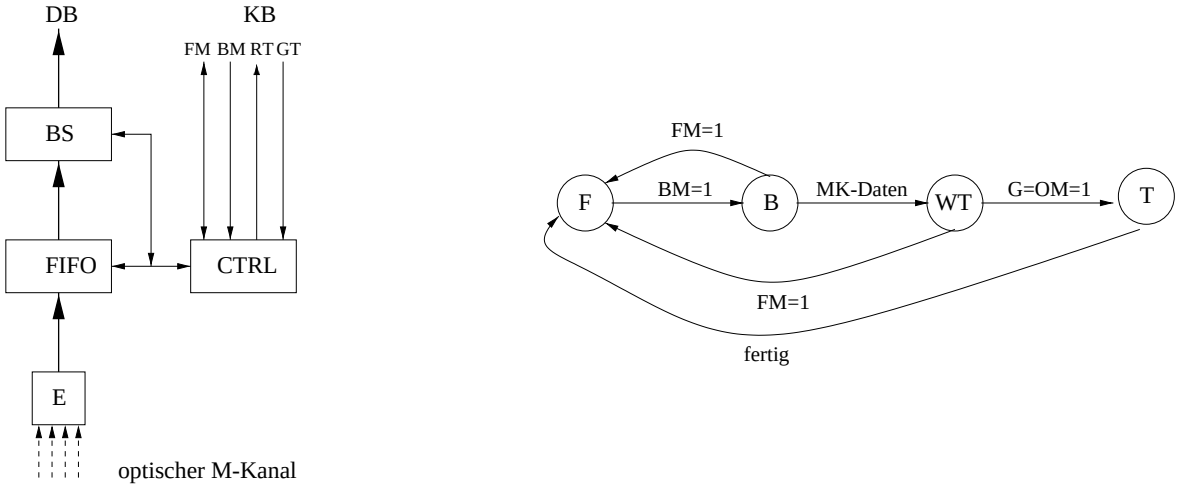


Abbildung 7.11: Die Abbildung zeigt den Aufbau (links) und das Zustandsübergangsdiagramm (rechts) einer ME.

3. ihre FM_i-Leitung überwachen und gegebenenfalls die Speicherbank freigeben,
4. Daten von der Speicherbank empfangen und zwischenspeichern,
5. bei Bedarf den Zugang zum Transmitter anfordern und die Daten über den B-Kanal übertragen,
6. am Ende einer Übertragung die Speicherbank freigeben.

Aufbau der ME

Der Aufbau einer ME ist in Abbildung 7.11 dargestellt. Sie besteht aus einem Empfänger (E) für die optischen Datenleitungen, einer Busschnittstelle (BS), einem DatenPuffer (FIFO) und einer Kontrollogik (CTRL). Der Empfänger ist an den Puffer angeschlossen und speichert dort die von der Speicherbank kommenden Daten zwischen. Dabei finden auch die Reduktion der Datenrate und der Übergang vom externen zum internen Takt statt. Damit können die Daten über die BS zum Transmitter des B-Kanals geschickt werden. Die Kommunikation mit dem KB findet über die Kotrolllogik statt. Sie überwacht die FM_i und BM_i GT-Leitungen, speichert den Zustand der Speicherbank und setzt die benötigten Signale auf der FM_i und RT Leitung. Außerdem übernimmt sie die Steuerung und Koordinierung der übrigen Komponenten.

Funktionsweise der ME

Die Funktionsweise der ME kann mit Hilfe eines endlichen Automaten mit 4 Zuständen beschrieben werden, dessen Zustandsübergänge durch die Leitungen des Kontrollbusses und die von der Speicherbank kommenden Daten gesteuert werden.

Frei (F): Eine ME befindet sich im F-Zustand, wenn die zugehörige Speicherbank keine Anfrage zu bearbeiten hat und der DatenPuffer der ME leer ist. Sie setzt während der Vergabephase des Buszyklus ihre FM_i-Leitung auf 1 und wartet darauf, daß sie eine PE durch Setzen ihrer BM_i-Leitung in den B-Zustand versetzt.

Besetzt (B): Der Besetzt-Zustand bedeutet, daß die zugehörige Speicherbank dabei ist, eine Anfrage zu bearbeiten. Eine ME im B-Zustand setzt in der Vergabephase des Buszyklus ihre FM_i-Leitung auf 0. Sie verläßt den B-Zustand in zwei Fällen: wenn eine PE in der Anfragephase ihre FM_i-Leitung auf 1 setzt, oder wenn das Ergebnis der Speicheroperation über den M-Kanal in ihrem DatenPuffer angekommen ist. Im ersten Fall wird die ME frei und geht in den F-Zustand über. Im zweiten Fall hängt der nächste Zustand von der Art der Speicheroperation ab. Bei einer WriteBack-Operation wechselt die ME in den F-Zustand, da die Speicherabfrage damit beendet ist. Bei einer Fetch-Operation dagegen begibt sich die ME in den WT-Zustand, um das Ergebnis des Speicherzugriffs an den Prozessor weiterzugeben.

Warten auf Transmitter (WT): Eine ME im WT-Zustand wartet auf den Zugang zum Transmitter des B-Kanals, um das Ergebnis des Speicherzugriffs an den Prozessor weiterzugeben. Sie setzt während der ersten

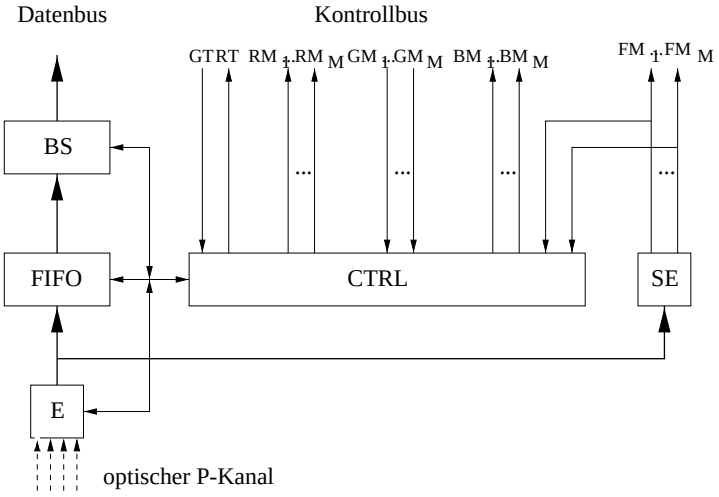


Abbildung 7.12: Der Aufbau der PE.

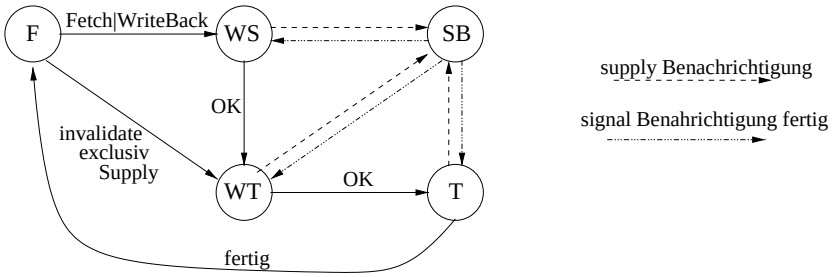


Abbildung 7.13: Das Zustandsübergangsdiagramm der PE.

Phase des Buszyklus ihre FM_i-Leitung auf 0 und ihre RT-Leitung auf 1. Die ME kann den WT-Zustand nur in dem T-Zustand verlassen. Dies geschieht, sobald die GT-Leitung der ME auf 1 geht, ihr also der Zugriff auf den Transmitter gewährt wird.

Datenübertragung (T): Die ME befindet sich in dem T-Zustand, während sie Daten aus ihrem Puffer an den Transmitter des B-Kanals über den Datenbus überträgt. Sobald die Übertragung abgeschlossen ist, geht sie in den F-Zustand über.

Das Zustandsdiagramm der ME ist in Abbildung 7.11 zu sehen.

7.5.5 Aufbau und Funktion einer PE

Die PE ist für zwei Aufgabenbereiche zuständig: sie muß die Speicheranfragen ihres Prozessors über den B-Kanal weiterleiten und für die Abwicklung von Supply-Operationen sorgen. Um die Speicheranfragen weiterzuleiten, muß die PE:

1. die Daten von dem P-Kanal empfangen und zwischenspeichern,
2. aufgrund des Befehlswortes entscheiden, ob, und wenn ja, welche Speicherbank angefordert werden muß,
3. die für die Anforderung der benötigten Ressourcen notwendigen Signale auf dem KB setzen,
4. nach Erhalt der Ressourcen die Datenübertragung durchführen und bei Bedarf die betroffene Speicherbank über die BM_i Leitung als besetzt kennzeichnen.

Bei der Abwicklung der Supply-Operationen muß zwischen der Benachrichtigung und dem eigentlichen Supply unterschieden werden. Für die Bearbeitung der Benachrichtigung muß die FM_i -Leitung der entsprechenden ME auf 0 gesetzt werden. Dabei muß berücksichtigt werden, daß eine Supply-Benachrichtigung wie in 7.1.3 beschrieben,

auch mitten in anderen Übertragung ankommen kann. Die PE muß also bei jedem ankommenden Datenpaket prüfen, ob es sich dabei um eine Supply-Benachrichtigung handelt. Die eigentliche Supply-Operation wird genauso wie eine normale Prozessoranfrage behandelt, die keine Speicherbank benötigt. Sie wird also empfangen, zwischengespeichert, und über den B-Kanal übertragen.

Im Nachfolgenden wird zunächst der Aufbau der PE beschrieben. Daraufhin wird ihre Funktionalität mit Hilfe eines endlichen Automaten genau spezifiziert.

Aufbau der PE

Eine PE besteht, wie in Abbildung 7.12 dargestellt, aus einem Empfänger (E) für die optischen Datenleitungen, einer Busschnittstelle (BS), einem DatenPuffer (FIFO), einer Kontrolllogik (CTRL) und einer Supply-Einheit (SE). Der Empfänger, der Puffer und die Busschnittstelle haben die gleiche Funktion wie bei der ME. Der Empfänger ist für die Optik-Elektronik Wandlung, und den Übergang von externen zum internen Takt zuständig. Er leitet die Daten an den DatenPuffer weiter, von wo sie über die Busschnittstelle an den Transmitter geschickt werden. Die Datenübertragung zum Transmitter wird durch die Kontrolllogik gesteuert. Sie ist an die RT-, GT-, RM_i-, GM_i-, FM_i- und BM_i-Leitungen des Kontrollbusses angeschlossen, über die sie die benötigten Ressourcen anfordert und wieder freigibt. Um entscheiden zu können, ob, bzw. welche Speicherbank angefordert werden muß, überwacht die Kontrolllogik die vom Empfänger kommenden Daten und liest von jeder Anfrage das Befehls- und Adreßwort. Die Behandlung der Supply-Benachrichtigungen wird von der SE gehandhabt. Sie ist dazu sowohl an die FM_i-Leitungen des KB als auch an den Ausgang des Empfängers angeschlossen. Sobald am Empfängerausgang eine Supply-Benachrichtigung erscheint, merkt sich die SE die Nummer der Speicherbank und setzt in der nächsten Anfragephase die korrespondierende FM_i-Leitung auf 1.

Funktionsweise der PE

Die Funktionsweise der PE kann durch einen endlichen Automaten mit 5 Zuständen beschrieben werden. Einer davon (F für frei) entspricht einer untätigen PE, die auf Anfragen wartet. Drei (WS für Warten auf Speicher, WT für Warten auf Transmitter und T für Senden zum Transmitter) bezeichnen die verschiedenen Phasen der Bearbeitung einer Prozessoranfrage. Hinzu kommt ein Spezialzustand, der für die Bearbeitung einer Supply-Benachrichtigung reserviert ist (SB). Da die Supply-Benachrichtigungen Vorrang vor allen anderen Anfragen hat, kann eine PE von jedem Zustand der ersten Gruppe die Durchführung einer Operation unterbrechen und in einen Zustand der zweiten Gruppe wechseln. Ist die Supply-Operation beendet, dann kehrt die PE in den Ausgangszustand zurück. Das Zustandsübergangsdiagramm ist in Abbildung 7.13zu sehen. Die der Zustände kann wie folgt zusammengefaßt werden:

Frei (F): Die PE befindet sich im F-Zustand, wenn sie keine Anfragen zu bearbeiten hat. Sie verläßt ihn, sobald eine Prozessoranfrage über den P-Kanal empfangen wird. Je nachdem, um was für eine Anfrage es sich dabei handelt, geht sie entweder in den WS- oder in den WT-Zustand über. Ersteres ist der Fall, wenn ein Zugriff auf eine Speicherbank notwendig ist (also eine Fetch- oder WriteBack-Operation durchgeführt werden soll). Bei allen anderen Operationen geht der Prozessor in den WT-Zustand über.

Suply-Benachrichtigung (SB): Sobald die PE eine Supply-Benachrichtigung auf dem P-Kanal empfängt, wechselt sie in den SB-Zustand. Dies erfolgt unabhängig davon, in welchem Zustand sie sich bei der Ankunft der Supply-Benachrichtigung befindet. Sie verbleibt im SB-Zustand nur solange, wie sie für das Setzen der FM_i der, von der Supply-Benachrichtigung betroffenen Speicherbank benötigt. Danach kehrt sie in den Zustand zurück, aus dem sie in den SB-Zustand gelangt ist.

Warten auf Speicherbank (WS): In den WS-Zustand gelangt die PE aus dem F-Zustand dann, wenn der Prozessor eine Fetch- oder eine WriteBack-Operation ausführen möchte. Dabei wartet sie auf die Verfügbarkeit der Speicherbank, auf die sich die Operation bezieht. Eine PE im WS-Zustand setzt in der Anfragephase des Buszyklus die RM_i-Leitung der gewünschten Speicherbank auf 1. Sie verläßt den WS-Zustand, sobald ihr durch eine 1 auf der GM_i- und FM_i-Leitung der Zugriff auf die Speicherbank gewährt wird. Sie geht dann in den WT-Zustand über.

Warten auf Transmitter (WT): Im WT-Zustand wartet die PE darauf, daß ihr der Transmitter des B-Kanals zugeteilt wird. Sie kann in diesem Zustand entweder, wie oben beschrieben aus den WS-Zustand, oder aus

Abbildung 7.14: Der Aufbau (links) und das Zustandsübergangsdiagramm (rechts) der TE.

dem F-Zustand gelangen. Letzteres passiert dann, wenn eine Invalidate- oder Exclusiv-Operation durchgeführt werden soll. Die PE setzt im WT-Zustand in der Anfragephase des Buszyklus ihre RT-Leitung auf 1. Falls sie den Transmitter bekommt und ihre GT-Leitung in der nachfolgenden Vergabephase auf 1 ist, geht sie in den T-Zustand über. Falls nicht, verbleibt die PE im WT-Zustand.

Datenübertragung (T): Im T-Zustand ist die PE mit der Übertragung der Daten aus Ihrem Puffer an den Transmitter beschäftigt. Sobald die Datenübertragung beendet ist, geht die PE in den F-Zustand über.

7.5.6 Aufbau der Transmittereinheit

Die TE liest Daten vom Datenbus und leitet sie an den optischen B-Kanal weiter. Da der SP Bus keine optischen Ausgabe auf dem Chip zuläßt, befindet sich der eigentliche Transmitter Off-Chip. Die TE muß also die Daten über geeignete Treiber vom Chip über die Platine an den externen optischen Transmitter übertragen. Sie beinhaltet eine einfache Kontrolllogik, eine Busschnittstelle und die Treiber Off-Chip Datenübertragung. Seine Funktion kann mit Hilfe von zwei Zuständen beschrieben werden: frei (F) und besetzt (B) (Abbildung 7.14). Der F-Zustand bedeutet, daß keine Daten übertragen werden, der Transmitter eben frei ist. Die TE geht in den B-Zustand über, sobald durch die STROBE-Leitung des Datenbusses angezeigt wird, daß Daten zur Übertragung vorliegen. Dabei werden die Daten über die Treiber an den externen Transmitter des optischen B-Kanals weitergegeben. Wenn die Datenübertragung beendet ist (STROBE=0), geht die TE in den F-Zustand zurück und wartet auf neue Anfragen.

7.6 Chipfläche

Einer der Faktoren, die die Skalierbarkeit eines PHOTOBUS-Systems bestimmen, ist der Platzbedarf der Schaltungen auf dem Bus-Chip. So ist die Anzahl von Prozessoren und Speicherbänken im System durch die maximale Anzahl von PEs und MEs beschränkt, die auf einem VLSI-Baustein untergebracht werden können. Sie hängt von drei Dingen ab: der Komplexität der einzelnen Schaltungen, der Art ihrer Implementierung und der zur Implementierung des Bus-Chips verwendeten Technologie. Das Ziel dieses Abschnittes besteht darin, die Komplexität auf der Basis der Diagramme in Abbildungen 7.12, 7.11, 7.14 grob abzuschätzen. Dabei geht es vor allem um zwei Dinge: den asymptotischen Anstieg der Fläche mit der Prozessorzahl, sowie eine Abschätzung der Größenordnung der pro Prozessor benötigten Fläche. Damit soll gezeigt werden, bis zu welcher Prozessorzahl man eine Implementierung der PHOTOBUS-Architektur in Bezug auf den Platzbedarf auf dem Chip als realistisch betrachten kann.

Die Fläche eines Buschips A setzt sich aus zwei Anteilen zusammen: der für die Schaltungen benötigten Fläche A_S und der Fläche der Busleitungen A_B:

$$A = A_S + A_B \tag{7.1}$$

Um den ersten Faktor zu ermitteln, betrachtet man am besten die Anzahl der logischen Gatter NG_S, die zur Implementierung der Schaltung notwendig sind. Aus ihr kann die Fläche entweder direkt berechnet, oder durch

Vergleich mit den Gatterzahlen kommerzieller Bausteine abgeschätzt werden.

Die Gatterzahl des Buschips setzt sich zusammen aus den Gatterzahlen von P PEEs (NG_P), den M MEEs (NG_M) sowie der Arbitrierschaltungen am Kontrollbus (NG_K) und der Transmittereinheit (NG_T):

$$NG_S = P \cdot NG_P + M \cdot NG_M + NG_K + NG_T \tag{7.2}$$

Wie den Abbildungen 7.12, 7.11 und 7.14 zu entnehmen ist, bestehen alle Komponenten des Bus-Chips vor allem aus vier Arten von Grundschaltungen: optischen Empfängern bzw. Transmittern, Busschnittstellen und Kontrollschaltungen. Im Nachfolgenden werden zunächst einige allgemeine Überlegungen zur Komplexität dieser Schaltungen dargelegt. Darauf basierend wird dann die Gatteranzahl der PEs, der MEs, und der TE betrachtet. Danach wird der Platzbedarf der Busleitungen abgeschätzt. Abschließend werden die Ergebnisse zusammengefaßt.

7.6.1 Grundschaltungen

Die Komplexität der Grundschaltungen kann wie folgt abgeschätzt werden:

Empfänger

Für den optischen Empfänger werden pro Leitung ein Verstärker und eine Konvertierschaltung benötigt. Der Verstärker generiert aus der Photospannung an der MQW-Diode ein digitales Signal und besteht aus einigen wenigen Transistoren. Seine Komplexität wird mit NG_{Vs} bezeichnet. Die Konvertierschaltung wandelt einen seriellen hochfrequenten in einen niederfrequenten parallelen Datenstrom. Sein Aufbau wurde im Abschnitt 7.2.2 genau beschrieben. Dabei wurde deutlich, daß seine Komplexität von dem Verhältnis der Übertragungsfrequenz des hochfrequenten, zu der des niederfrequenten Datenstroms V_F abhängt. Die Schaltung besteht aus zwei Schieberegistern, jeweils mit der Länge V_F, V_F Multiplexern und zwei Demutliplexern. Damit gilt für die Gatteranzahl des Empfängers einer optischen Datenleitung NG_E^1

$$NG_E^1 = V_F \cdot (2NG_{Bit} + NG_{Mux}) + 2NG_{Mux} \tag{7.3}$$

Zusätzlich zu den Datenleitungen besitzt jeder Empfänger eine Takt- und eine Kontrolleitung. Die Taktleitung ist an einen Zähler angeschlossen, der das Auswahlsignal für die Multiplexer und Demultiplexer generiert. Er besteht aus V_F Flip-Flops. Die Signale der Kontrolleitung markieren Befehlsworte und müssen zusammen mit den Datenbits gespeichert werden. Die Komplexität der zur Kontrolleitung gehörenden Schaltung ist somit mit der einer Datenleitung (NG_E^1) identisch. Mit der Anzahl der Datenleitungen im P-Kanal N_{PK} folgt daraus für die Komplexität des Empfängers:

$$\begin{aligned} NG_E &= (N_{PK}+1) \cdot NG_E^1 + V_F \cdot NG_{bit} \\ &= (N_{PK}+1) \cdot (V_F \cdot (2NG_{Bit} + NG_{Mux}) + 2NG_{Mux}) + V_F \cdot NG_{bit} \\ &= V_F \cdot (2N_{PK}+3) \cdot NG_{Bit} + (V_F+2) \cdot NG_{Mux} \end{aligned} \tag{7.4}$$

Busschnittstelle

Bei der Busschnittstelle wird für jede Leitung ein Treiberbaustein benötigt. Dabei müssen sowohl die Datenleitungen als auch die Kontrolleitungen berücksichtigt werden. Unter der Annahme, daß die Treiber für alle Leitungsarten in etwa gleich komplex sind und aus $NG_{Treiber}$ Gattern bestehen gilt:

$$NG_{BS} = (N_{DB} + N_{KB}) \cdot NG_{BTr} \tag{7.5}$$

In obiger Gleichung stehen N_{DB} für die Anzahl der Leitungen des Datenbusses und N_{KB} für die Anzahl der Anschlüsse an den Kontrollbus. Da der Datenbus in einem Buszyklus genauso viel Daten übertragen soll wie die optischen Datenleitungen, muß der DB $V_F$mal so viele Datenleitungen wie die optischen Kanäle besitzen. Hinzu kommen die üblichen Zusatzleitungen, zu denen mindestens eine CLOCK und eine STROBE-Leitung gehören. Damit gilt:

$$NG_{BS} = (N_{PK} \cdot V_F + 2 + N_{KB}) \cdot NG_{Treiber} \tag{7.6}$$

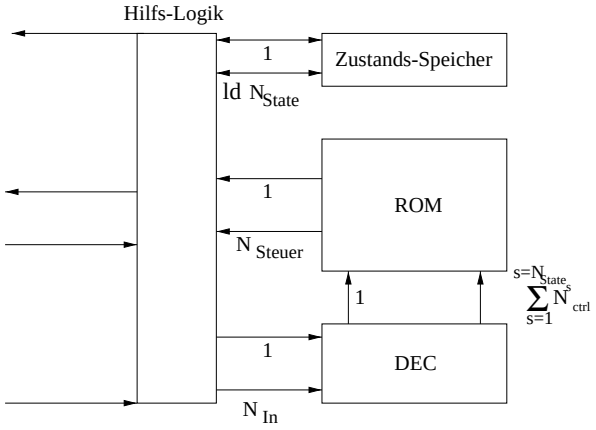


Abbildung 7.15: Das Model der Kontrollogik, daß für die Abschätzung der Komplexität der einzelnen Komponenten des Bus-Chips verwendet wird.

FIFO

Der FIFO-Puffer ist ein paralleles Schieberegister, dessen Wortbreite der Wortbreite des Datenbusses ($N_{PK} \cdot V_F$) entspricht. Wie beschrieben verfügt er über die Möglichkeit, nicht nur die letzte, sondern jede beliebige Stufe des Schieberegisters mit dem Ausgang zu verbinden. Um dies zu erreichen, besteht das FIFO aus einer Speichermatrix und einer Adreßschaltung. Die Gatteranzahl der Speichermatrix N^S_{FIFO} ergibt sich aus der Gatteranzahl eines Bit-Speichers und der Anzahl der zu speichernden Bits. Ersteres setzen wir mit der Gatteranzahl eines Flip-Flops NG_{Bit} gleich, die in Gleichung (7.4) benutzt wurde. Letzteres ergibt sich aus der Länge einer Cache-Zeile l_C in Bit, der Anzahl der zu speichernden Cache-Zeilen N_{CZ} und der Länge eines Adreß- und Befehlsworts l_A bzw. l_K:

$$NG^S_{FIFO} = (N_{CZ} \cdot l_C + l_A + l_K) \cdot NG_{bit} \tag{7.7}$$

Die Adreßschaltung besteht aus einem 1-Bit Schieberegister, das die aktuelle Position des Eingabestroms in der Speichermatrix markiert und einer Auswahlschaltung. Die Länge des Schieberegisters entspricht der Anzahl der Worte in der Speichermatrix. Sie gleicht der Kapazität der Speichermatrix ($N_{CZ} \cdot l_C + l_A + l_K$) dividiert durch die Wortlänge $N_{PK} \cdot V_F$. Die Auswahlschaltung besitzt für jedes Bit der Speichermatrix ein Transmission-Gate, über die das Bit an die ihm zugeordnete Datenleitung angeschlossen ist. Bezeichnet man die Komplexität der Transmission Gate als NG_{Tr} so folgt daraus für die Gatteranzahl der Adreßschaltung NG^{Addr}_{FIFO}

$$NG^{Addr}_{FIFO} = N_{PK} \cdot V_F \cdot NG_{Bit} + (N_{CZ} \cdot l_C + l_A + l_K) \cdot NG_{TrG} \tag{7.8}$$

Damit ergibt sich für die Gesamtzahl der Gatter im FIFO

$$\begin{aligned} NG_{FIFO} &= NG^S_{FIFO} + NG^{Addr}_{FIFO} \\ &= (N_{CZ} \cdot l_C + l_A + l_K) \cdot (NG_{Bit} + NG_{TrG}) + N_{PK} \cdot V_F \cdot NG_{Bit} \end{aligned} \tag{7.9}$$

Kontrollschaltkreise

Die Kontrollschaltkreise müssen in jedem der möglichen Zustände einer Einheit in Abhängigkeit von den Signalen des Kontrollbusses und den anliegenden Daten, entsprechende Steuersignale für die übrigen Komponenten generieren. Außerdem müssen sie den aktuellen Zustand speichern und bei Bedarf Zustandsübergänge durchführen. Die Struktur der Kontrollschaltkreise kann je nach benötigter Funktionalität sehr unterschiedlich sein. Eine genaue Analyse der Komplexität der Kontrollschaltung ist daher nur an Hand eines vollständigen Entwurfs möglich. Allerdings ist eine größenordnugsmäßige Abschätzung auch an Hand des in Abbildung 7.15 dargestellten einfachen Modells möglich. Es geht davon aus, daß die Kontrollfunktion mit Hilfe einer ROM-basierten Tabelle realisiert wird. Die Tabelle beinhaltet für jeden Zustand der Einheit und jede in diesem Zustand zulässige Kombination von Kontrollsignalen die benötigten Steuersignale. Dabei wird durch die Beschränkung auf die in jedem Zustand zulässigen Signale vermieden, daß die Kapazität der Tabelle exponentiell mit der Anzahl der Kontrollleitungen steigt, an die die Einheit angeschlossen ist. Sie wird durch spezielle Decoder realisiert. Insgesamt besteht die Kontrollschaltung damit aus fünf Komponenten: einem Zustandsspeicher, der ROM-Tabelle, einem Decoder, einem

Adreßgenerator und einer Zusatzlogik. Bezeichnet man ihre Gatterzahlen als NG_{State}, NG_{ROM}, NG_{Dec},NG_{Adr} und NG_{hilfs}, so gilt für die Anzahl der Gatter der Kontrollschaltung NG_{CTRL}:

$$NG_{CTRL} = NG_{State} + NG_{ROM} + NG_{Dec} + NG_{Adr} + NG_{hilfs} \tag{7.10}$$

Ihre Komplexität hängt von vier Faktoren ab:

1. der Anzahl der Zustände der Einheit: N_{State},
2. der Anzahl von Kontrollsignalen, die für die Zustandsübergänge relevant sind N_{ctrl},
3. der Anzahl der Eingabesignale der Einheit N_{in}, aus denen sich die obigen Kontrollsignale ableiten,
4. der Anzahl der Steuersignale, die von der Kontrolleinheit generiert werden müssen N_{Str}.

Die Werte für N_{State} und N_{ctrl} können aus den Zustandsübergangs-Diagrammen der Einheiten ermittelt werden. Die Anzahl der Eingabesignale N_{in} und der Steuersignale N_{Str} ergibt sich aus den Diagrammen der Einheiten. Mit diesen Werten kann die Komplexität der fünf Komponenten einer Kontrolleinheit wie folgt abgeschätzt werden:

Zustandsspeicher: Der Zustandsspeicher beinhaltet den aktuellen Zustand der Einheit. Er wird durch ein Register verwirklicht, das eine Zahl zwischen 1 und der Anzahl der Zustände N_{State} speichern kann. Ein solches Register benötigt $ld\,(N_{State})$ Bits und damit

$$NG_{State} = ld\,(N_{State}) \cdot NG_{bit} \tag{7.11}$$

Gatter.

Decoder: Der Decoder generiert aus den Eingabesignalen einer Einheit die Kontrollsignale, die die Zustandsübergänge steuern und über die Werte der Steuersignale bestimmen. So wird z.B. bei der PE aus der Speicherbanknummer und den FM_i Signalen das 'Speicherbank verfügbar' Signal erzeugt, das den Übergang aus dem WS in den WT Zustand bewirkt. Er beinhaltet für jedes Kontrollsignal einen Vergleicher für boolsche Vektoren der Dimension N_{in}. Dieser vergleicht die Eingabesignale mit den Werten, die zu dem Kontrollsignal gehören. Es werden also N_{ctrl} solcher Vergleicher benötigt. Da für einen b-Bit Vergleicher $2b \Leftrightarrow 1$ AND-Gatter benötigt werden, gilt für die Komplexität des Decoders N_{Dec}:

$$NG_{Dec} = 2(N_{in} \Leftrightarrow 1) \cdot N_{ctrl} \simeq 2N_{in} \cdot N_{ctrl} \tag{7.12}$$

ROM-Tabelle: Da in jedem Zustand einer Einheit jeweils nur wenige Kontrollsignale eine Bedeutung haben, wäre es sinnlos, für jeden Zustand für alle möglichen Kombinationen der Kontrollsignale einen eigenen Eintrag in der Tabelle zu generieren. Dies würde zur einer exponentiellen Abhängigkeit der Kapazität der Tabelle von der Anzahl der Kontrollsignale und damit zu einer sehr hohen Kapazität der Tabelle führen. Um dies zu vermeiden, wird für jeden Zustand nur für die Kombinationen der Kontrollsignale ein Eintrag generiert, die in diesem Zustand eine Bedeutung hat. Diese Anzahl wird als N^s_{ctrl}, $1 \leq s \leq N_{State}$ bezeichnet. Die benötigte ROM-Kapazität N_{ROM} ergibt sich dann aus N^s_{ctrl} sowie der Anzahl der Steuersignale N_{Str} und der Anzahl der Zustände N_{State} als :

$$N_{ROM} = \left(N_{Str} \cdot \sum_{s=1}^{N_{State}} N^s_{ctrl} \right) \tag{7.13}$$

Schätzt man die zur Realisierung einer ROM-Zelle notwendige Anzahl von Gattern durch die Gatterzahl eines Flip-Flops NG_{Bit} so ergibt sich für die Gatterzahl des ROM-Speichers:

$$NG_{ROM} = N_{ROM} \cdot NG_{Bit} = \left(N_{Str} \cdot \sum_{s=1}^{N_{State}} N^s_{ctrl} \right) \cdot NG_{bit} \tag{7.14}$$

Adreßgenerator: Der Adreßgenerator erzeugt aus Kontrollsignalen und der Nummer des aktuellen Zustands einer Einheit die Adresse des dazugehörenden Eintrags in der ROM-Tabelle. Er besitzt für jeden Zustand und jede in diesem Zustand gültige Signalkombination einen Vergleicher. Damit werden $\sum_{s=1}^{N_{State}} N^s_{ctrl}$

Vergleicher benötigt. Da jeder Vergleicher die Werte aller Kontrollsignale und die Nummer des Zustands berücksichtigen muß werden Vergleicher für binäre Vektoren der Dimension $N_{ctrl} + ld\,(N_{State})$ benötigt. Für die Komplexität des Adreßgenerators NG_{Adr} gilt also:

$$NG_{Adr} = 2(N_{ctrl} + ld\,(N_{State})) \cdot \sum_{s=1}^{N_{State}} (N^s_{ctrl}) \tag{7.15}$$

Zusatzlogik: Die Hilfsschaltung ist für die Funktionalität gedacht, die durch die ROM-Tabelle nicht ohne weiteres abgedeckt ist. Ihre Komplexität wird mit NG_{hlp} bezeichnet und muß für jede Einheit einzeln betrachtet werden.

Die Gatterzahl der Kontrollschaltung ergibt sich damit zu:

$$\begin{aligned} NG_{CTRL} &= NG_{State} + NG_{Dec} + NG_{ROM} + NG_{Adr} + NG_{hilfs} \\ &= ld\,(N_{State}) \cdot NG_{bit} + 2N_{in} \cdot N_{ctrl} + \left(N_{Str} \cdot \sum_{s=1}^{N_{State}} N^s_{ctrl} \right) \\ &\quad +2(N_{ctrl} + ld\,(N_{State})) \cdot \sum_{s=1}^{N_{State}} (N^s_{ctrl}) + NG_{hilfs} \end{aligned} \tag{7.16}$$

7.6.2 ME

Die ME besteht, wie der Abbildung 7.5.4 zu entnehmen ist, aus einem optischen Empfänger, einem FIFO-Puffer, einer Busschnittstelle und einer Kontrollschaltung. Bezeichnet man die zugehörigen Gatterzahlen mit NG^M_E, N^M_{FIFO}, NG^M_{BS} und NG^M_{CTRL}, dann gilt:

$$NG_M = NG^M_E + NG^M_{FIFO} + NG^M_{BS} + NG^M_{CTRL} \tag{7.17}$$

Die Gatteranzahl des Empfängers ist direkt durch Gleichung (7.3) gegeben, da es zwischen den Empfängern der ME und der PE keine Unterschiede gibt. Es gilt

$$NG^M_E = NG_E = V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux} \tag{7.18}$$

Da der ME-Puffer lediglich eine Cache-Zeile fassen muß, gilt für die Komplexität des Puffers mit Gleichung 7.7:

$$NG^M_{FIFO} = (l_C + l_A + l_K) \cdot NG_{bit} \tag{7.19}$$

Aus Abbildung 7.11 kann abgelesen werden, daß die ME 4 Anschlüsse an den Kontrollbus besitzt (RT, GT, FM_i und BM_i). Damit ergibt sich die Komplexität der Busschnittstelle aus:

$$NG^M_{BS} = (N_{PK} \cdot V_F + 2 + 4) \cdot NG_{Tr} \tag{7.20}$$

Um die Komplexität der Kontrolleinheit nach Gleichung (7.9.8) zu bestimmen, werden die Anzahl der Eingabeleitungen (N^M_{in}), die Anzahl der Steuersignale (N^M_{in}), die Anzahl der Kontrollsignale (N^M_{in}), die Anzahl der Zustände (N^M_{in}) sowie die Anzahl der in jedem Zustand gültigen Kontrollsignale (N^M_{in}) benötigt. Außerdem muß die Komplexität der Hilfsschaltung NG^M_{hilfs} ermittelt werden. Aus der Beschreibung der Architektur der ME in Abschnitt 7.5.4 können die gesuchten Werte wie folgt ermittelt werden:

Eingabesignale: Der Zustand der ME hängt von zwei Dingen ab: den Eingangssignalen des KB und den von der Speicherbank empfangenen Daten. Die Anzahl der Eingabesignale ergibt sich damit aus der Anzahl der Eingänge vom KB-Bus (2) und der Länge des Befehlswortes (l_K):

$$N^M_{in} = 2 + l_K \tag{7.21}$$

Steuersignale: Die Kontrolleinheit der ME muß das RT und das FM_i Signal für den KB sowie das STORBE-Signal für den DB generieren. Hinzu kommen Steuersignale für die Busschnittstelle, das FIFO und die Empfängerschaltung.

$$N^M_{Str} = 10 \tag{7.22}$$

Zustände und Kontrollsignale: Das Zustandsdiagramm besitzt 4 Zustände (siehe Abbildung 7.11), wobei die Zustandsübergänge durch insgesamt 6 Kontrollsignale (3 für die Art des Befehls, ein Frei-Signal, ein Grant-Signal und ein Signal, das das Ende einer Übertragung signalisiert):

$$N^M_{State} = 4 \tag{7.23}$$
$$N^M_{ctrl} = 6 \tag{7.24}$$
$$\tag{7.25}$$

Für die Anzahl gültiger Signalkombinationen in jedem Zustand gilt:

$$N^F_{ctrl} = 1, \quad N^B_{ctrl} = 2, \quad N^{WT}_{ctrl} = 2, \quad N^T_{ctrl} = 2, \tag{7.26}$$

und damit

$$\sum_{s=1}^{N^M_{State}} (N^s_{ctrl}) = 7 \tag{7.27}$$
$$\tag{7.28}$$

Setzt man die obigen Werte in Gleichung (7.9.8), so ergibt sich für die Komplexität der Kontrolllogik der ME:

$$\begin{aligned} NG^M_{CTRL} &= ld(4) \cdot NG_{bit} + (2 + l_k) \cdot 6 + 10 \cdot 7 \cdot NG_{Bit} + 2(6 + ld(4)) \cdot 7 \\ &\simeq 100 NG_{Bit} + 500 \end{aligned} \tag{7.29}$$

Zusammen mit Gleichungen für die restilchen Komponenten der ME daraus für ihre Gesamtkomplexität:

$$\boxed{\begin{aligned} NG_M \simeq\ & V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux} + (l_C + l_A + l_K) \cdot NG_{bit} \\ & + (N_{PK} \cdot V_F + 6) \cdot NG_{Tr} + 100 NG_{Bit} + l_k + 500 \end{aligned}} \tag{7.30}$$

7.6.3 PEs

Wie in Abbildung 7.12 zu sehen, besteht eine PE aus fünf Komponenten: einem Empfänger, einem FIFO-Puffer, einer Busschnittstelle, einem Kontrollschaltkreis sowie einer Supply-Einheit. Bezeichnet man die zugehörigen Gatterzahlen mit NG^P_E, N^P_{FIFO}, NG^P_{BS}, NG^P_{CTRL}, NG_{SE} so gilt also:

$$NG_P = NG^P_E + N^P_{FIFO} + NG^P_{BS} + NG^P_{CTRL} + NG_{SE} \tag{7.31}$$

Die Komplexität dieser Komponenten kann wie folgt berechnet werden:

Empfänger, FIFO

Die Gatteranzahl des Empfängers ergibt sich wie bei der ME direkt aus Gleichung (7.7). Auch die Komplexität des FIFOs ist mit der ME identisch, da es ebenfalls lediglich eine Cache-Zeile fassen muß

$$NG^M_{FIFO} = (l_C + l_A + l_K) \cdot NG_{bit} \tag{7.32}$$

Busschnittstelle

Aus Abbildung 7.12 kann abgelesen werden, daß die PE einen RT, einen GT, und jeweils M FM_i, RM_i, GM_i und BM_i Anschlüsse an den Kontrollbus besitzt. Damit ergibt sich die Komplexität der Busschnittstelle aus

$$NG^P_{BS} = (N_{PK} \cdot V_F + 2 + 4M + 2) \cdot NG_{Tr} \tag{7.33}$$

SE

Die Supply-Einheit hat lediglich die Aufgabe, die Ankunft einer Supply-Benachrichtigung zu erkennen und gemäß der Speicherbank-Nummer die entsprechende FM_i-Leitung auf 1 zu setzen. Dazu benötigt sie l_K AND-Gatter für die Interpretation des Befehlswortes sowie einen Decoder, der aus der Nummer der Speicherbank das richtige FM_i-Signal erzeugt. Da letzterer $ld(M) \cdot M$ Gattern benötigt, gilt damit:

$$NG_{SE} = ld(M) \cdot M + l_k \tag{7.34}$$

Kontrollogik

Um die Gatteranzahl der Kontrolllogik zu ermitteln, müssen wie bei der ME zunächst die Werte für die Anzahl der Eingabeleitungen (N_{in}^P), die Anzahl der Steuersignale (N_{St}^P), die Anzahl der Kontrollsignale (N_{ctrl}^P), die Anzahl der Zustände (N_{State}^P) sowie die Anzahl der in jedem Zustand gültigen Kontrollsignale bestimmt werden.

Eingabesignale: die Eingangssignale der Kontrolllogik sind alle GM_i, alle FM_i sowie das GT-Signal des Kontrollbusses. Hinzu kommen die Leitungen des Kontrollwortes. Damit ergibt sich für die Anzahl der Eingabesignale:

$$N_{in}^P = 1 + 2M + l_K \tag{7.35}$$

Steuersignale: Die Kontrolleinheit der PE muß das RT,das RM_i und das BM_i Signal für den KB sowie das STROBE-Signal für den DB generieren. Hinzu kommen Steuerungssignale für die Busschnittstelle, das FIFO und die Empfängerschaltung. Wir schätzen:

$$N_{Str}^P = 10 \tag{7.36}$$

Die obige Schätzung geht davon aus, daß in der ROM-Tabelle lediglich Steuersignale gespeichert werden müssen, die die Art der zu setzenden Leitung bestimmen (RM_i oder BM_i). Welche RM_i bzw. BM_i gesetzt wird, berechnet aus der Speicherbanknummer des Adreßwortes die Hilfslogik.

Zustände und Kontrollsignale: Die Anzahl der Zustände der PE beträgt 5. Aus dem Zustandsübergangsdiagramm im Abbildung 7.13 kann man ablesen, daß insgesamt 10 Signale die Zustandsübergänge steuern. So werden zunächst zwei Signale für die beiden möglichen Übergänge aus dem F-Zustand benötigt. Hinzu kommen jeweils ein Signal für den Übergang zwischen dem WS und dem WT, dem WT und T sowie die Rückkehr zum F-Zustand am Ende der Übertragung. Schließlich kommen 5 Signale hinzu, die für die Handhabung des SB-Zustands notwendig sind: eines für den Wechsel zum SB-Zustand und 4 für die 4 möglichen Rückwege. Es gilt also:

$$N_{State}^P = 5 \tag{7.37}$$
$$N_{ctrl}^P = 10 \tag{7.38}$$
$$\tag{7.39}$$

Für die Anzahl gültiger Signalkombinationen in jedem Zustand kann man aus Abbildung 7.13 folgendes ablesen:

$$N_{ctrl}^F = 3, \quad N_{ctrl}^{WS} = 2, \quad N_{ctrl}^{WT} = 2, \quad N_{ctrl}^T = 2, N_{ctrl}^{SB} = 4, \tag{7.40}$$

und damit

$$\sum_{s=1}^{N_{State}^P} (N_{ctrl}^s) = 13 \tag{7.41}$$
$$\tag{7.42}$$

Neben den obigen Signal-Anzahlen wird für die Bestimmung der Gatterzahl der Kontrolleinheit noch die Komplexität der Hilfslogik benötigt. Sie muß die Nummer der Speicherbank in das gewünschte BM_i, FM_i, bzw. RM_i Signal generieren. Dazu wird ein Decoder benötigt, der über $ldM \cdot M$ AND-Gattern verfügt. Es ist also:

$$NG_{hilfs}^P = ldM \cdot M \tag{7.43}$$

Setzt man die obigen Werte in Gleichung (7.9.8), so ergibt sich für die Komplexität der Kontrolllogik der PE:

$$\begin{aligned} NG_{CTRL}^P &= ld(5) \cdot NG_{Bit} + (2 + M + l_k) \cdot 10 + 10 \cdot 13 \cdot NG_{Bit} + 2(10 + ld(5)) \cdot 13 + ldM \cdot M \\ &\simeq 200NG_{Bit} + 10M + 500 + ldM \cdot M \end{aligned} \tag{7.44}$$

Zusammenfassung

Faßt man die Gleichungen für die einzelnen Komponenten der PE zusammen, so ergibt sich für ihre Gatterzahl:

$$\boxed{\begin{array}{rcl} NG_P & \simeq & V_F \cdot (2N_{PK}+3) \cdot NG_{Bit} + (V_F+2) \cdot NG_{Mux} + (l_C + l_A + l_K) \cdot NG_{bit} \\ & & +(N_{PK} \cdot V_F + 2 + 4M + 2) \cdot NG_{Tr} + ld(M) \cdot M + l_k \\ & & +200NG_{Bit} + 10M + 500 + ldM \cdot M \end{array}} \tag{7.45}$$

7.6.4 TE

Die Transmittereinheit ist die einfachste Komponente des Bus-Chips. Sie besteht aus einer Busschnittstelle, einem Treiber für die externe Datenübertragung und einer Kontrolleinheit. Bezeichnet man die dazugehörenden Gatteranzahlen als NG^T_{BS}, NG^T_{Tr} und NG^T_{CTRL} so kann man die Komplexität der TE NG_T als

$$NG_T = NG^T_{BS} + NG^T_{Tr} + NG^T_{CTRL} \tag{7.46}$$

schreiben. Da die TE lediglich an den Datenbus angeschlossen ist, beträgt die Gatteranzahl der Busschnittstelle

$$NG^T_{BS} = (N_{PK} \cdot V_F + 2) \cdot NG_{Tr} \tag{7.47}$$

Bedenkt man, daß die TE die Bussignale unverändert an den externen Transmitter weitergibt, dann kann man die Gatteranzahl der Treiber für die externe Datenübertragung durch

$$NG^T_{Tr} = (N_{PK} \cdot V_F + 2) \cdot NG_{Ext} \tag{7.48}$$

ausdrücken. Dabei bezeichnet NG_{Ext} die Komplexität des für eine Leitung benötigten Treibers.

Die Kontrolllogik der TE muß lediglich zwei Zustände (F und B) und ein Kontrollsignal (STROBE) berücksichtigen. Unter der Annahme, daß 2 Steuersignale generiert werden müssen ergibt sich damit aus Gleichung (7.16):

$$NG^P_{CTRL} = NG_{Bit} + 2 + 4 + 2 \cdot 2 = NG_{Bit} + 10 \tag{7.49}$$

Faßt man nun Gleichungen (7.47), (7.48) und (7.49) zusammen, dann gilt für die Gatterzahl der TE

$$\boxed{NG_T = (N_{PK} \cdot V_F + 2) \cdot (NG_{Tr} + NG_{Ext}) + NG_{Bit} + 10} \tag{7.50}$$

7.6.5 Logik des Kontrollbusses

Wie in 7.5.2 beschrieben, besitzt jede Arbitrierleitung des KB für jede an sie angeschlossene Instanz eine spezielle Schaltung. Sie besteht aus einem Flip-Flop, zwei einfachen logischen Gattern und zwei Transmission Gates. Bezeichnet man die Gatteranzahl der Arbitrierschaltung mit NG_{Arb} so gilt:

$$NG_{Arb} = 2NG_{Tr} + NG_{Bit} + 2 \tag{7.51}$$

Der KB besitzt eine Arbitrierleitung für den Transmitter und jeweils eine für jede der M Speicherbänke. An die Transmitter-Arbitrierleitung sind alle P PEs und alle M MEs angeschlossen. Daraus ergibt sich für die zugehörige Gatteranzahl NG^{Tr}_{Arb}.:

$$NG^{Tr}_{Arb} = (P+M) \cdot NG_{Arb} = (P+M) \cdot (2NG_{Tr} + NG_{Bit} + 2) \tag{7.52}$$

Da an die Arbitrierleitungen für die Speicherbänke nur die P PEs angeschlossen sind, kann deren Gatterzahl NG^{Sp}_{Arb} als

$$NG^{Sp}_{Arb} = P \cdot NG_{Arb} = P \cdot (2NG_{Tr} + NG_{Bit} + 2) \tag{7.53}$$

geschrieben werden. Zusammen folgt aus den beiden obigen Gleichungen für die Gatteranzahl des Kontrollbusses NG_{KB}:

$$\begin{array}{rcl} NG_{KB} & = & NG^{Tr}_{Arb} + M \cdot NG^{Sp}_{Arb} \\ & = & (P + (P+1) \cdot M) \cdot (2NG_{Tr} + NG_{Bit} + 2) \end{array} \tag{7.54}$$

Die letzte Zeile ergibt sich aus der Annahme, daß die Anzahl der Speicherbänke und damit die Anzahl der MEs gleich der Prozessoranzahl ist.

7.6.6 Busleitungen

Die Fläche, die von den Busleitungen beansprucht wird, hängt von zwei Dingen ab: von der für die Implementierung verwendetet VLSI-Technologie und der Komplexität der Verbindungsstruktur. Im Nachfolgenden werden zunächst kurz die relevanten Technologieparameter beschrieben. Auf der Basis dieser Parameter wird dann der Flächenbedarf der Verbindungsstruktur des Bus-Chips der PHOTOBUS-Architektur geschätzt.

Technologieparameter

In VLSI-Bausteinen gibt es in der Regel spezielle Metalllagen für die Implementierung von Leitungen. Die Leistungsfähigkeit der Technologie in Bezug auf die Implementierung von Verbindungsstrukturen ist dabei durch zwei Dinge bestimmt: die Anzahl dieser Lagen sowie die Dichte mit der die Leitungen in den einzelnen Lagen verlegt werden können. Die Anzahl der Lagen liegt zwischen 2 und 8. Sie wird im weiteren mit N_{Lag} bezeichnet. Die Leitungsdichte ist durch die minimale Leitungsbreite W_L und den minimalen Abstand zwischen zwei Leitungen D_L gegeben. Beide werden in der Regel als Vielfaches der minimalen Strukturgröße einer Technologie λ angegeben. Sie liegt heute zwischen $0,15\mu m$ und $0,35\mu m$. Wie im weiteren deutlich wird, ist die Breite eines Bündels paralleler Leitungen $D_{||}$ entscheidend für den Flächenbedarf der Verbindungsstruktur des Bus-Chips. Sie ergibt sich unter den oben beschriebenen Annahmen wie folgt aus der Anzahl der Leitungen N_L der

$$D_{||} = \frac{N_L \cdot (W_L + D_L \Leftrightarrow 1)}{N_{Lag}} \cdot \lambda \tag{7.55}$$

Verbindungsstruktur

Der Flächenbedarf der Verbindungsstruktur des Bus-Chips hängt von von zwei Faktoren ab: der Breite der eigentlichen Busleitungen und dem Platzbedarf der Anschlußleitungen, über die die PEs und die MEs an die Busleitungen angeschlossen sind. Um ihn für den DB und den KB zu bestimmen, gehen wir davon aus, daß die Busleitungen als paralleles Bündel quer über den Chip verlaufen. Die PEs und MEs befinden sich an den Rändern der Busse. Sie sind mit ihnen über Anschlußleitungen verbunden, die im rechten Winkel zu den Busleitungen verlaufen. Dabei wird angenommen, daß die Bus- und Anschlußleitungen in unterschiedlichen Lagen verwirklicht werden, so daß die Anschlußleitungen über den Busleitungen verlaufen können. In einer solchen Anordnung wird die Breite der Verbindungsstruktur durch die Breite der Busleitungen bestimmt. Die Länge ist nach unten durch die Summe der Breiten aller Anschlußleitungen aller PEs und MEs beschränkt. Damit ergibt sich der Flächenbedarf der Verbindungsstruktur A_V aus der Breite der beiden Busse D_{KB} und D_{DB}, der Breite der PE und ME Anschlußleitungen (D_{An}^{PE} und D_{An}^{ME}) als:

$$A_V = (D_{KB} + D_{DB}) \cdot (P \cdot D_{An}^{PE} + M \cdot D_{An}^{ME}) \tag{7.56}$$

Flächenbedarf der Busleitungen

Der DB besitzt, wie bereits beschrieben, $N_{PK} \cdot V_F$ Daten und 2 Kontrollleitungen. Der KB besteht aus $M + 1$ Arbitrierleitungen (eine für den Transmitter und M für die Speicherbänke) sowie jeweils M BM- und FM-Leitungen. Damit gilt mit Gleichung (7.55) für die Breite der Busleitungen $D_{KB} + D_{DB}$:

$$D_{KB} + D_{DB} = (N_{PK} \cdot V_F + 2 + 3M + 1) \cdot \frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \tag{7.57}$$

Die Anzahl der Anschlußleitungen einer der PEs und der MEs wurde bereits im Zusammenhang mit der Komplexität ermittelt. Sie beträgt $N_{PK} \cdot V_F + 4 + 4M$, bzw. $N_{PK} \cdot V_F + 6$. Für D_{An}^{PE} und D_{An}^{ME} folgt daraus:

$$D_{An}^{PE} = (N_{PK} \cdot V_F + 4 + 4M) \cdot \frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \tag{7.58}$$

$$D_{An}^{ME} = (N_{PK} \cdot V_F + 6) \cdot \frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \tag{7.59}$$

$$\tag{7.60}$$

Setzt man die obigen Gleichungen zusammen mit Gleichung (7.57) in Gleichung (7.56) ein so ergibt sich für die Gesamtfläche der Verbindungsstruktur des Bus-Chips:

$$A_B = ((P+M)N_{PK} \cdot V_F + 4P(M+1) + 6) \cdot (N_{PK} \cdot V_F + 3M + 3) \cdot \left(\frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \right)^2 \tag{7.61}$$

7.6.7 Zusammenfassung

Im vorliegenden Abschnitt werden die Ergebnisse der bisherigen Analyse zusammengefaßt und veranschaulicht. Dazu werden zunächst die Ausdrücke für die Gatteranzahl und den Flächenbedarf so umgeformt, daß die asymptotische Abhängigkeit von der Prozessor und Speicherbank deutlich wird. Danach wird an Hand konkreter Werte gezeigt, daß die Chipfläche bis zu 128 Prozessoren kein Problem für die Skalierbarkeit von der PHOTOBUS-Architektur darstellt.

Asymptotischer Anstieg der Schaltungsfläche

Die Gleichungen für die Gatterzahl der Komponenten des Bus-Chips beinhalten vier Arten von Termen: $M \cdot ld(M)$-Terme, P-Terme, M-Terme sowie Terme, die weder von der Prozessorzahl noch von der Speicherbankzahl abhängen. Berücksichtigt man zusätzlich, daß die Terme in den Ausdrücken für die Komplexität der PEs bzw. MEs mit P bzw. M multipliziert werden, kann die Gatteranzahl für die Schaltungen des Bus-Chips also als:

$$NG_S = P \cdot M \cdot ld(M) \cdot NG_S^{PMld(M)} + P \cdot M \cdot NG_S^{PM} + P \cdot NG_S^P + M \cdot NG_S^M + NG_S^C \tag{7.62}$$

geschrieben werden. Für die Werte der einzelnen Terme kann aus Gleichungen (7.30), (7.45), (7.50) und (7.54) folgendes abgelesen werden:

$$\begin{aligned}
NG_S^{PMld(M)} &= 2 && (7.63)\\
NG_S^{PM} &= 12 + 6NG_{Tr} + NG_{Bit} && (7.64)\\
NG_S^P &\simeq (V_F \cdot (2N_{PK} + 3) + 200 + (l_C + l_A + l_K)) \cdot NG_{Bit} \\
&\quad + (V_F + 2) \cdot NG_{Mux} + (N_{PK} \cdot V_F + 6) \cdot NG_{Tr} + l_k + 500 && (7.65)\\
NG_S^M &\simeq (V_F \cdot (2N_{PK} + 3) + 100 + (l_C + l_A + l_K)) \cdot NG_{Bit} \\
&\quad + (V_F + 2) \cdot NG_{Mux} + (N_{PK} \cdot V_F + 8) \cdot NG_{Tr} + l_K + 500 && (7.66)\\
NG_S^C &= N_{PK} \cdot V_F + 2) \cdot (NG_{Tr} + NG_{Ext}) + NG_{Bit} + 10 && (7.67)
\end{aligned}$$

Asymptotischer Anstieg der Fläche der Verbindungsstruktur

Der Ausdruck für den Flächenbedarf der Verbindungsstruktur (7.61) beinhaltet Terme, die von PM^2, PM, P, M abhängen, sowie von der Prozessor- und Speicherbankanzahl unabhängige Terme. Er kann also als

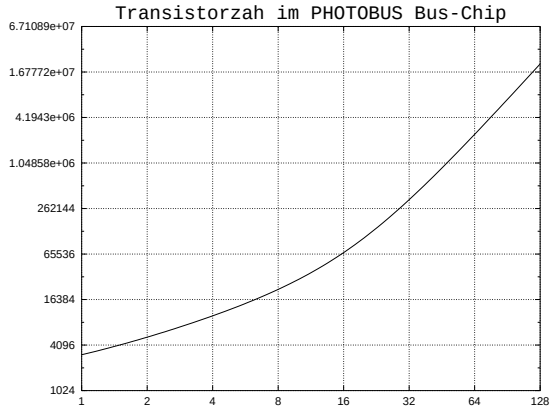
$$A_V = P \cdot M^2 A_V^{PM^2} + P \cdot M \cdot A_V^{PM} + P \cdot A_V^P + M \cdot A_V^M + A_V^C \tag{7.68}$$

geschrieben werden. Die einzelnen Terme können aus Gleichung (7.61) leicht durch Umformen und Zusammenfassen ermittelt werden. Es gilt:

$$\begin{aligned}
A_V^{PM^2} &= 12 && (7.69)\\
A_V^{PM} &= 12 + 3N_{PK} \cdot V_F && (7.70)\\
A_V^P &= 12 + (N_{PK} \cdot V_F)^2 + 7N_{PK} \cdot V_F && (7.71)\\
A_V^M &= 18 + 3N_{PK} \cdot V_F && (7.72)\\
A_V^C &= (N_{PK} \cdot V_F)^2 + 3N_{PK} \cdot V_F + 18 && (7.73)
\end{aligned}$$

Abschätzung möglicher Werte

Um konkrete Werte für den Flächenbedarf des Bus-Chips zu bekommen, werden zunächst die Anzahl der Leitungen der P- und M-Kanäle N_{PK} und das Verhältnis ihrer Datenrate zur On-Chip Datenrate V_F benötigt. Wir gehen im



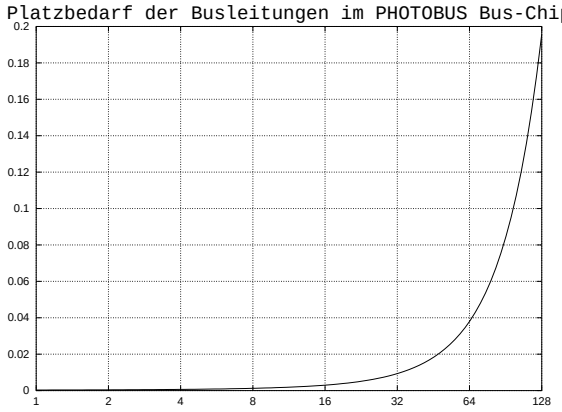


Abbildung 7.16: Das linke Diagramm zeigt die Abhängigkeit der auf dem Bus-Chip benötigten Gatter (vertikale Achse) von der Prozessorzahl (horizontale Achse). Das rechte Diagramm stellt den anstieg der für die Verbindungsstruktur benötigten Fläche in cm^2 (vertikale Achse) mit der Prozessorzahl. Beide Diagramme verwenden eine logarithmische Darstellung.

Nachfolgenden davon aus, daß 20 optische Leitungen pro Kanal verwendet werden, die jeweils mit dem 10fachen der On-Chip Datenrate betrieben werden:

$$N_{PK} = 20, \quad V_F = 10 \tag{7.74}$$

Dies deutlich mehr als in den meisten Fällen benötigt wird.

Desweiteren werden für die Berechnung als Längen der einzelnen Teile der Speicheranfragen, folgende Werte angenommen:

$$l_C = 512, \quad l_A = 64, \quad l_K = 8 \tag{7.75}$$

Wir gehen also von einem 64-Bit großen Adreßraum, eine Cache-Zeile von 64 Byte und eine 8 Bit Kontrollwort. Schließlich werden für die Bestimmung der gesamten Gatteranzahl die Gatterzahlen eines Flip-Flops NG_{Bit}, eines Transmission-Gates und eines Transmission-Gate basierten Multiplexers benötigt. Dafür werden folgende Werte angenommen:

$$NG_{Bit} = 2, \quad NG_{Mux} = 2, \quad NG_{Tr} = 1 \tag{7.76}$$

Durch Einsetzen dieser Werte und entsprechendes Aufrunden gelangt man zu der folgenden Gleichung für den Flächenbedarf der Schaltung:

$$\boxed{NG_S \quad \simeq \quad 2P \cdot M \cdot ld(M) + 50P \cdot M \cdot + 1000P \cdot + 1000M \cdot + 1000} \tag{7.77}$$

Um eine Abschätzung für die Fläche der Verbindungsstruktur zu bekommen, werden zusätzlich die Technolgieparameter W_L (minimale Dicke einer Leitung), D_L (minimaler Abstand zwischen zwei Leitungen) , N_{Lag} (die Anzahl der Leitungslagen) und die Strukturgröße λ benötigt. Wir gehen im Nachfolgenden von der heute üblichen Strukturgröße von 0.25μ mit 5 Leitungslagen aus und nehmen für die Dicke und den Abstand der Leitungen 3λ an:

$$W_L = 3, \quad D_L = 3, \quad N_{Lag} = 5, \quad \lambda = 0.25\mu \tag{7.78}$$

$$\left(\frac{W_L + D_L \Leftrightarrow 1}{NLag}\lambda\right)^2 [cm^2] = 6,25 \cdot 10^{-10} \tag{7.79}$$

Durch Einsetzen und Aufrunden ergibt sich damit:

$$\boxed{A_V[cm^2] \quad = \quad (12P \cdot M^2 + 1000P \cdot M \cdot + 40000P \cdot + 10000M + 50000) \cdot 6,25 \cdot 10^{-10}} \tag{7.80}$$

Die obigen Abschätzungen wurden in Abbildung 7.16 graphisch dargestellt. Dabei wurde angenommen, daß es im System für jeden Prozessor eine Speicherbank gibt, es gilt also $P = M$. Bedenkt man, daß heutige VLSI-Bausteine bis zu 10^7 Gatter auf einer Fläche von bis zu 4 cm^2 besitzen können, so kann man aus der Abbildung folgendes schließen:

Für bis zu 128 Prozessoren stellt die Fläche des Chips keine Einschränkung der Skalierbarkeit dar.

7.7 Leistungsparameter

Für die theoretische Modellierung und die Simulation einer Architektur werden die Bearbeitungszeiten der Speicheroperationen in den einzelnen Architekturkomponenten benötigt. Im Falle der PHOTOBUS-Architektur müssen für jede der 6 Speicheroperationen und Synchronisationsoperationen (FetchRead, FetchWrite, Exclusiv, Invalidate, WriteBack und Supply, Lock und Unlock im Nachfolgenden mit dem Suffix ^{Op} bezeichnet) die folgenden Parameter bekannt sein:

1. die Bearbeitungsdauer im Speicher (Speicherlatenz T_S^{Op}),
2. die Übertragungsdauer in den optischen Kanälen inklusive der Schnittstellen (P-, M- und B-Kanal): T_P^{Op}, T_M^{Op} und T_B^{Op},
3. die Bearbeitungsdauer Bus-Chip T_{BC}^{Op} mit den im vorliegenden Kapitel beschriebenen Teilkomponenten PE, ME, und TE und DB.

Der Speicherlatenz liegt heute bei ca. 30 Prozessorzyklen:

$$T_S^{Op} = 30 \tag{7.81}$$

Die anderen Parameter können wie nachfolgend beschrieben abgeschätzt werden.

7.7.1 Optische Kanäle

Die, die bei einer Operation Op auf die Nachrichtenübertragung Operation auf einem optischen Kanal Kan entfällt (T_{Kan}^{Op}) hängt von vier Dingen ab: der Bandbreite des Kanals B_{Kan}, der Latenz des Kanals L_{Kan}, der Anzahl der Nachrichten die auf dem Kanal übertragen werden müssen N_{Kan}^{Op} und der der Länge dieser Nachrichten $l_{Kan}^{Op,i}$, $1 \leq i <\leq N_{Kan}^{Op}$ Es gilt:

$$T_{Kan}^{Op} = \sum_{i=1}^{N_{Kan}^{Op}} \left(L_{Kan} + \frac{l_{Kan}^{Op,i}}{B_{Kan}} \right) \tag{7.82}$$

Die Latenz und Bandbreite der optischen Kanäle sind im Kapitel 6 für verschiedene Technologiestufen bereits abgeschätzt worden. Die Anzahl und Art der Nachrichten die für die Einzelnen Operationen über die Kanäle verschickt werden gehen aus der Beschreibung der Kommunikationsstruktur (Abschnitt 7.1.1) und des Nachrichtenformats (Anschnitt 7.2.3) hervor. Ihre Länge ergibt sich der Länge eines Befehlswortes l_K, einer Adreßangabe l_A ind einer Cachezeiel l_C. Diese Werte wurde bereits im Zusammenhang mit der Bestimmung der Chipfläche auf $l_C = 512$, $l_A = 64$, $l_K = 8$ festgelegt (Gleichung (7.75). Für die einzelnen Operationen ergibt sich damit:

$$T_{Fetch}^{P} = L_P + \frac{72}{B_P} \qquad T_{Fetch}^{B} = 2L_B + \tfrac{144+512}{B_B} \qquad T_{Fetch}^{M} = L_M + \frac{72+512}{B_M} \tag{7.83}$$

$$T_{WriteBack}^{P} = L_P + \frac{72+512}{B_P} \qquad T_{WriteBack}^{B} = L_B + \tfrac{72+512}{B_B} \tag{7.84}$$

$$T_{Inv/Excl}^{P} = L_P + \frac{72}{B_P} \qquad T_{Inv/Excl}^{B} = 2L_B + \tfrac{72}{B_B} \tag{7.85}$$

$$T_{(Un)Lock}^{P} = L_P + \frac{72}{B_P} \qquad T_{(Un)Lock}^{B} = 2L_B + \tfrac{144}{B_B} \qquad T_{(Un)Lock}^{M} = L_M + \frac{72}{B_M} \tag{7.86}$$

$$\tag{7.87}$$

In den obigen Gleichungen wurde übersichtshalber die FetchRead und FetchWrite Operationen zu Fetch die Invalidate und Exclusiv Operationen zu Inv/Excl so wie Lock und Unlock zu (Un)Lock zusammengefaßt.

7.7.2 Bearbetungszeit Zeiten im Buschip

Die Bearbeitungszeit der einzelnen Operationen im Buschip ist durch die Summe der Bearbeitungszeiten in der PE (T_{PE}^{Op}), der ME (T_{ME}^{Op}) und der TE (T_{TE}^{Op}):

$$T_{TE}^{Op} = T_{PE}^{Op} + T_{ME}^{Op} + T_{TE}^{Op} \tag{7.88}$$

Diese Zeit muß in Prozessorzyklen angegeben werden. Daher wird im Nachfolgenden als erstes die Frage nach dem Verhältnis der Taktfrequenz des Buschips zu Prozessorfrequenz untersucht. Danach werden die einzelnen Zeiten geschätzt.

Die Taktfrequenz

Die Dauer eines Taktzyklus ist durch die Dauer der zeitaufwendigsten Operation gegeben, die in einem Zyklus durchgeführt werden muß. Die Taktfrequenz hängt also von zwei Faktoren ab: der Leistung der verwendeten Technologie und der Komplexität der für einen Taktzyklus bestimmten Operationen. Dabei müssen sowohl die Schaltzeit als auch die Signallaufzeit berücksichtigt werden.

In der Arbeit wird angenommen, daß der Bus-Chip mit der gleichen Frequenz betrieben werden kann wie ein kommerzieller Mikroprozessor. Unter der Annahme, daß für die Implementierung die gleiche Technologie verwendet wird, bedeutet dies, daß die Arbeit der Chip-Komponenten in Schritte eingeteilt werden muß, die jeweils nicht aufwendiger als ein Prozessorbefehl sind.

Da die Ausführung eines Prozessorbefehls ein komplexer Vorgang ist, stellt die erste Forderung kein Problem dar. Bei heutigen CPUs wird die Logiktiefe eines Befehls als 10 bis 15 Gattern angenommen [73]. Wie in den nachfolgenden Abschnitten dargelegt wird, sind die einzelne Schritte der PE- und ME-Bearbeitung erheblich einfacher.

Bearbeitungszeit in der PE

Eine Nachricht vom Prozessor wird in der PE in vier Schritten bearbeitet::

1. Die Daten werden vom P-Kanal empfangen. Dabei findet die Konvertierung des hochfrequenten optischen in einen elektrischen Datenstrom mit der Chip-Frequenz statt.

2. Das Befehlswort wird durch die Kontrollogik ausgewertet.

3. Die benötigten Ressourcen (B-Kanal und bei Bedarf Speicherbank) werden angefordert.

4. Die Daten werden an den DB übergeben.

Mit dem Empfangen der Daten und der Konvertierung der Datenströme ist eine Latenz von einem Taktzyklus verbunden. Geht man davon aus, daß die Kontrolleinheit direkt an dem hochfrequenten Datenstrom mithört, so kann die Auswertung des Befehls mit der Konvertierung überlappt werden. Für die beiden ersten Schritte wird also insgesamt nur ein Taktzyklus benötigt.

Für die Anforderung des B-Kanals und der Speicherbank muß die PE die RT- bzw. RM_i-Leitung auf 1 setzen und auf die Antwort vom KB warten. Dabei muß das in 7.5.3 beschriebene zwei-Phasen Arbitrierungsverfahren befolgt werden. Da jede Phase einen Taktzyklus dauert, werden für die Anforderung jeder Ressource mindestens zwei Taktzyklen benötigt. Darüber hinaus muß man berücksichtigen, daß die Anforderung nur in der Anfragephase des KB beginnen kann. Je nachdem, wann der erste Schritt beendet ist, kann also ein zusätzlicher Wartezyklus notwendig sein.

Die Übergabe der Daten an den DB erfordert lediglich Sie wird hier mit einem Taktzyklus geschätzt.

Insgesamt beträgt die PE-Latenz für die Operationen, die keine Speicherbank benötigen (Exclusiv und Invalidate) 5, für alle anderen WriteBack-Operationen 7 Taktzyklen:

$$T_{PE}^{Inv} = T_{PE}^{Excl} = 5 \tag{7.89}$$

$$T_{PE}^{Fetch} = T_{PE}^{WriteBack} = T_{PE}^{Lock} = T_{PE}^{Unlock} = T_{PE}^{Supply} = 4 \tag{7.90}$$

Bearbeitungszeit in der ME

Die obige Betrachtung kann weitgehend auch für die ME übernommen werden. Es muß lediglich berücksichtigt werden, daß die ME immer nur den Zugriff auf den KB benötigt. Außerdem bearbeitet sie nur Fetch-, Lock- und Unlock-Operationen. Damit gilt:

$$T_{ME}^{Fetch} = T_{ME}^{Lock} = T_{ME}^{Unlock} = 4 \tag{7.91}$$

Bearbeitungszeit der TE

Die Transmittereinheit liest die Daten vom Kontrollbus und leitet sie sofort an den optischen Transmitter weiter. Die einzige Verzögerung entsteht dabei durch die Konvertierung des niederfrequenten in einen hochfrequenten Datenstrom. Diese beträgt einen Taktzyklus, so daß für alle Operationen gilt:

$$T_{TE}^{Op} = 1 \tag{7.92}$$

7.8 Leistungsbewertung

Im vorliegenden Abschnitt werden die Ergebnisse der Leistungsbewertung der PHOTOBUS-Architektur erläutert. Hierzu werden das in Kapitel 5 beschriebene Simulationssystem und die theoretische Modellierung verwendet. Als Basis für die Bewertung dienen die im vorigen Abschnitt ermittelten Leistungsparameter. Aufgrund der großen Breiten der technologisch möglichen Werte für die einzelnen Parameter ist es allerdings nicht sinnvoll, hier die Ergebnisse für alle Wertekombinationen zu präsentieren.

Im Nachfolgenden wird zuerst die Simulation des Bus-Chips beschrieben. Als Nächstes werden die Auswahl der Parameter und die mit diesen Parametern ermittelten Ergebnisse erläutert. Abschließend werden die Ergebnisse der Simulation mit der theoretischen Analyse des Systems verglichen.

7.8.1 Simulation des Bus-Chips

Die Simulation wird mit Hilfe des erweiterten LIMES-Simulators und einer Untermenge der SPLASH-2 Benchmark Sammlung durchgeführt. Beide wurden in 5.4.3 ausgiebig beschrieben. Die einzige Komponente, bei der die Details der Simulation offen gelassen wurde, ist der Bus-Chip, der weitestgehend architekturspezifisch ist.

Die Implementierung der Bus-Chip Simulation für die PHOTOBUS-Architektur basiert auf der gleichen Idee wie die Realisierung der übrigen Teile des Simulators. Dabei wird eine discreet-event Simulation durchgeführt mit dem Ziel, den Fluß der Nachrichten durch ein physikalisches System möglichst genau nachzubilden. Die `OEBUS`-Klasse beinhaltet dazu für jede Komponente des Chips eine Instanz einer entsprechenden Unterklasse. Dazu gehören: eine `PE`-Unterklasse für die PEs, eine `ME`-Unterklasse für die MEs, eine `Xmitter`-Unterklasse für den B-Kanal Transmitter und eine `Arbiter`-Unterklasse für die Simulation der Arbitrierungslogik des KB. Das Verhalten der einzelnen Komponenten wird in Form endlicher Automaten simuliert, die der Beschreibung in den vorigen Abschnitten entsprechen. Dabei wird der Weg der Nachrichten durch das Kopieren der korrespondierenden Instanzen der `Channel_Signals`-Klasse zwischen Ein-/Ausgabefeldern von Instanzen der Komponentenklassen nachgebildet. Dabei wird insbesondere darauf geachtet, daß die folgenden Punkte gewährleistet sind:

1. Fehler im Protokoll führen wie in einem echten System zum Nachrichtenverlust durch Überschreiben von Ein-/Ausgabefeldern. Da die Zustandsübergänge der simulierten Komponenten durch die in den Eingabefeldern ankommenden Nachrichten ausgelöst werden, führen außerdem fehlende Nachrichten zu Verklemmungen.

2. Der Weg der Nachrichten durch die Komponenten und deren Verzögerung kann sowohl für einzelne Nachrichten als auch für die Summe aller Nachrichten überprüft werden. Eine solche Überprüfung wurde stichprobenartig vorgenommen und zeigte ein korrekte Ausführung.

Dadurch sind eine hohe Konfidenz für die Korrektheit der Simulation sowie die Möglichkeit zur Überprüfung der Korrektheit der Protokolle gegeben.

7.8.2 Verwendete Parameter

Die technologisch bedingten Parameter wurden in Kapitel 6 definiert. Große Variationen sind vor allem bei der Bandbreite und der Latenz der optischen Kanäle zu finden. Gemäß Kapitel 6 müssen also die folgenden Parameter betrachtet werden: B_B, B_P B_M (Bandbreite des B-,P- und M–Kanals), und L_B, L_P L_M (Latenz des B-,P- und M–Kanals). hinzu kommen die Schaltzeiten der einzelnen Komponenten des Bus-Chips. Sie sind, wie im vorigen Abschnitt beschrieben, weitgehend festgelegt. Sie sind durch Gleichungen (7.89) bis (7.92) gegeben.

Bandbreite

Die PHOTOBUS-Architektur ist vor allem durch den Wunsch geprägt, mit sofort verfügbarer Technologie und möglichst geringem Aufwand einen opto-elektronischen Rechner realisieren zu können. Für die optischen Kanäle bedeutet dies, daß vor allem die Realisierung mit Hilfe konventioneller paralleler Fasersysteme interessant ist. Wie in Kapitel 4 beschrieben bedeutet dies eine Datenrate zwischen 1 und 4 GByte/s. Berücksichtigt man zusätzlich daß je eine optische CLOCK, STROBE und Kontrolleitung benötigt werden, dann sinkt diese Rate auf ca. 0.7 bis 3 GByte/s. Bei einer Prozessorfrequenz von 500MHz entspricht dies ca. 1,4 bis 6 Bytes /cycle. Für die Leistungsbewertung werden daher als realistischste Werte:

$$B_M = B_P = 1Byte/cycle, \quad B_B = 4Bytes/cycle \tag{7.93}$$

angenommen. Bei der Simulation werden dann für $B_B = 4$ verschiedene Werte für B_P und L_M Werte zwischen 1 und 4 betrachtet. Dazu werden für $B_P = B_M = 1$ B-Kanal Bandbreiten zwischen 2 und 32 untersucht.

Latenz

Die Frage des Einflußes der Latenz auf die Effizienz wird am Ende des Kapitels an Hand eines System mit on-Chip Synchronisation untersucht. Dies liegt daran daß ein solches System wie später gezeigt deutlich effizienter ist und in seiner Leistung weniger durch die Saturierung des B-Kanals beeinflußt wird. Dies hat für die Betrachtung der Latenz zwei Vorteile. Zum einen wird ihr Einfluß deutlicher sichtbar. Zum anderen ist der Simulationsaufwand deutlich geringer. Im Nachfolgenden gehen wir daher von der minimalen in der sofort verfügbaren Technologie möglichen Latenz aus. Wir gehen dabei davon aus, daß die Prozessoren alle in einem Radius von ca. 1M um den Bus-Chip angebracht sind und nehmen eine Leitungsverzögerung von 2 Zyklen an. Damit ergibt sich laut Tabelle 6.3

$$L_M = L_P = 6cycles, \quad L_B = 8cycles \tag{7.94}$$

7.8.3 Ergebnisse

Die Ergebnisse der Simulation sind in den Abbildungen 7.17 und 7.18 zu sehen. Die erste Abbildung zeigt auf der vertikalen Achse für jedes Benchmarkprogramm das Verhältnis der Laufzeit des Programms zur Laufzeit des idealen Parallelrechners mit Cache (siehe 5.4.1). In der zweiten Abbildung ist auf der vertikalen Achse die parallele Beschleunigung zu sehen. In beiden Digrammen ist auf den horizontalen Achse die Anzahl der Prozessoren aufgetragen. Die einzelnen Kurven ensprechen verschiedenen Werten für die Bandbreite der einzelnen Kanäle. Welche Kurve welcher Bandbreite entspricht ist den Abbildungen zu entnehmen.
Die wichtigsten Ergebnisse der Simulation, die den Graohen entnommen werden können, können wie folgt zusammengefaßt werden:

1. Die Bandbreite der P- und M-Kanäle hat nur eine geringe Auswirkung auf die Leistung. Sie ist vor allem bei Anwendungen mit einer hohen Miss-Rate und/oder einer hohen Anzahl von Synchronisationsoperationen wie FFT, Radix und insbesondere Ocean zu sehen. Auch dort ist sie nur für kleine Prozessorzahlen von Bedeutung, bei denen die Programme nahe an der optimalen Effizienz sind und die Überlastung des B-Kanals noch keine Rolle spielt.

2. Für die Abhängigkeit der benötigten Bandbreite des B-Kanals von der Prozessorzahl gilt:

   bis zu 8 Prozessoren: hier reicht jede der untersuchten Bandbreiten für eine optimale Leistung.

   16 Prozessoren: hier wird für eine optimale Leistung mindestens eine Bandbreite von 4Byte/Cycle benötigt.

   32 Prozessoren: eine gute Effizienz wird in diesem Bereich erst ab 8Bytes/Cycle erreicht.

   64 Prozessoren: hier werden 16 bis 32 Bytes/Cycle benötigt

3. Bei den Ocean und LU Benchmarks bricht die Leistung bei 32 bzw. 64 Prozessoren ein, und wird auch bei hoher Bandbreite des B-Kanals nicht besser. Grund hierfür ist eine hohe Anzahl der Synchronisationen, die zu einem Flaschenhals bei den Zugriffen auf die Speicherbank führt.

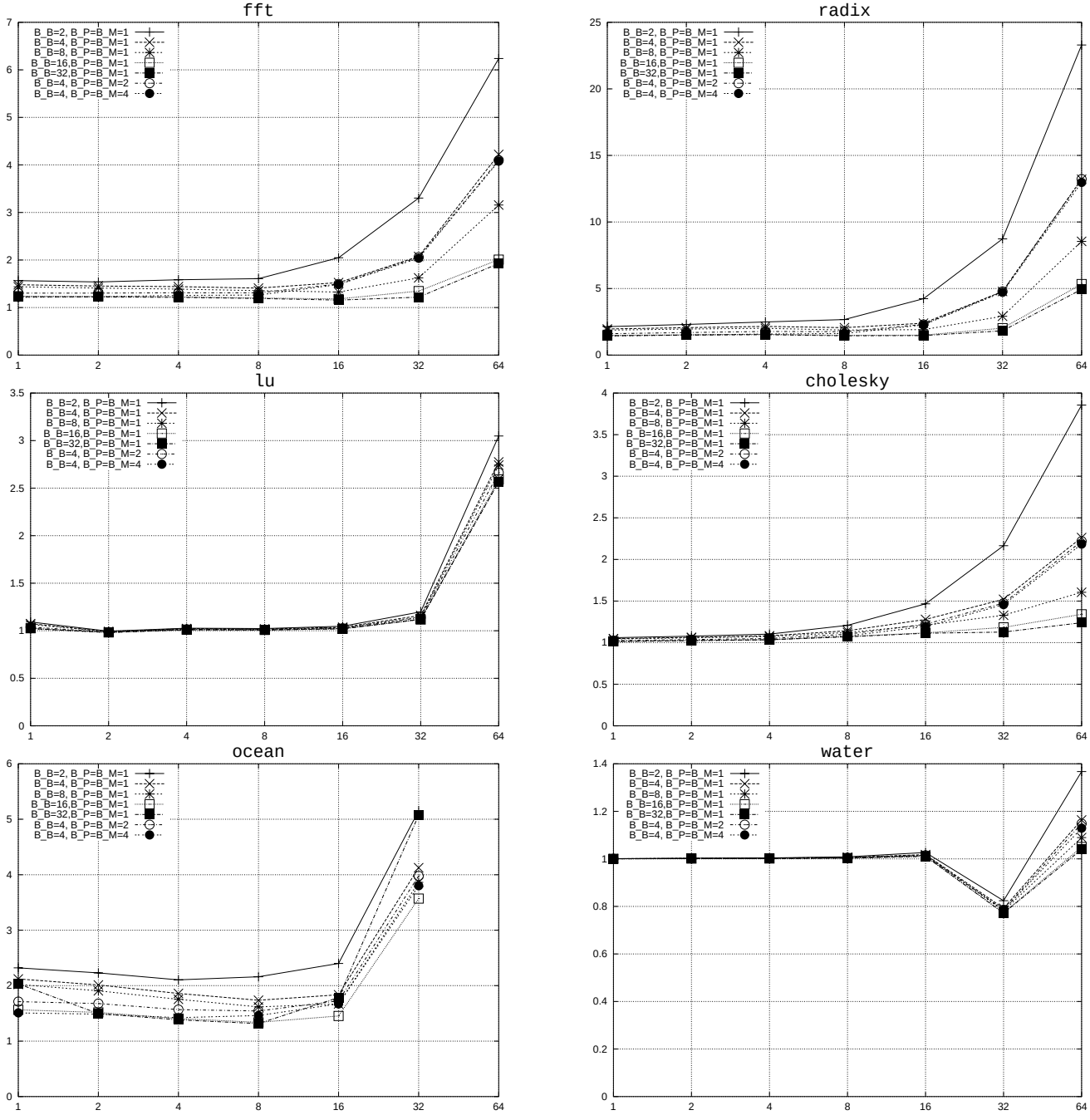


Abbildung 7.17: Die obigen Diagramme zeigen die durch Simulation ermittelte Ausführungszeit der einzelnen Benchmarks im Vergleich zur Ausführungszeit mit einem idealen Speichersystem

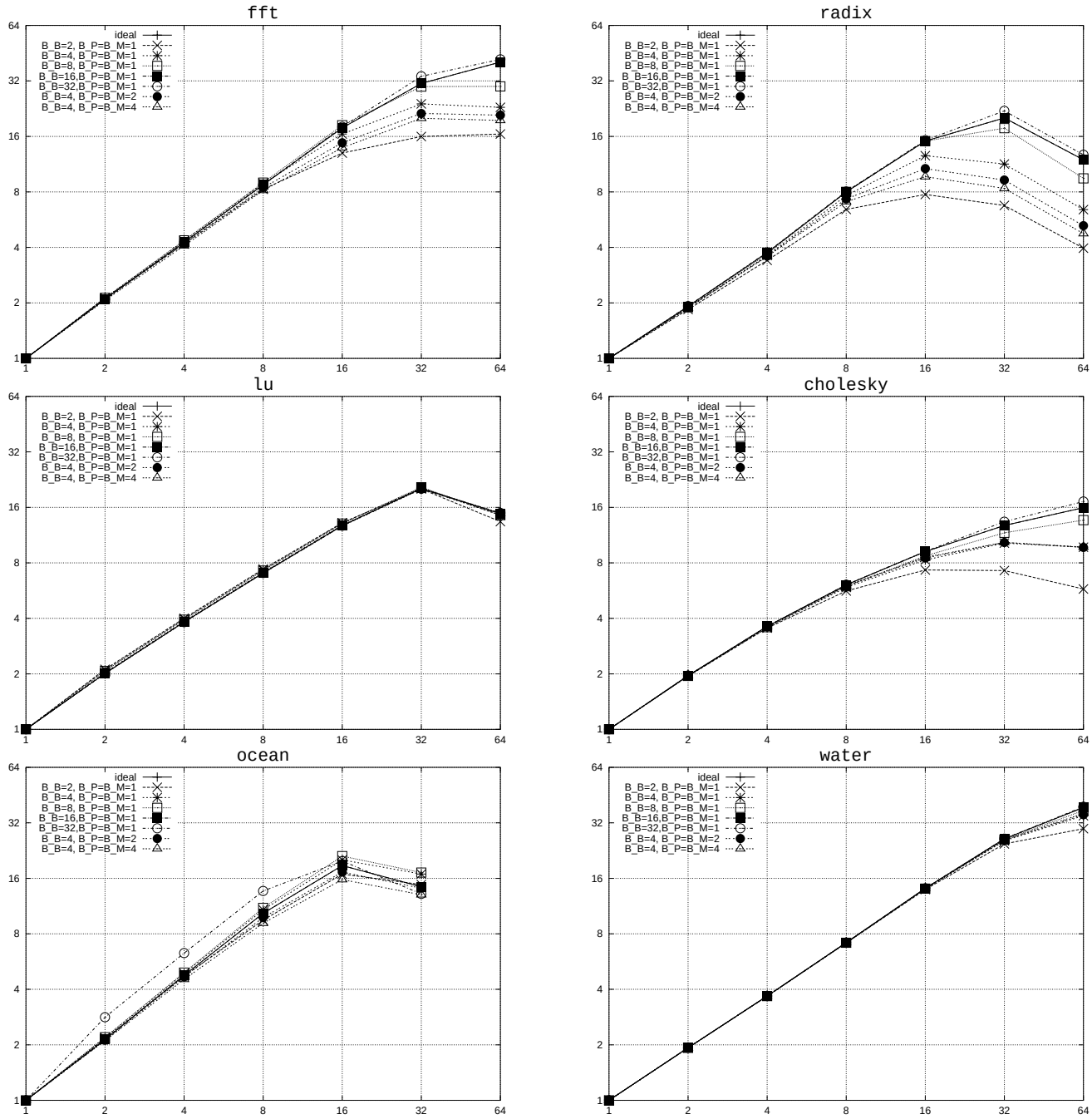


Abbildung 7.18: Die durch Simulation ermittelte Beschleunigung der einzelnen Benchmarkprogramme.

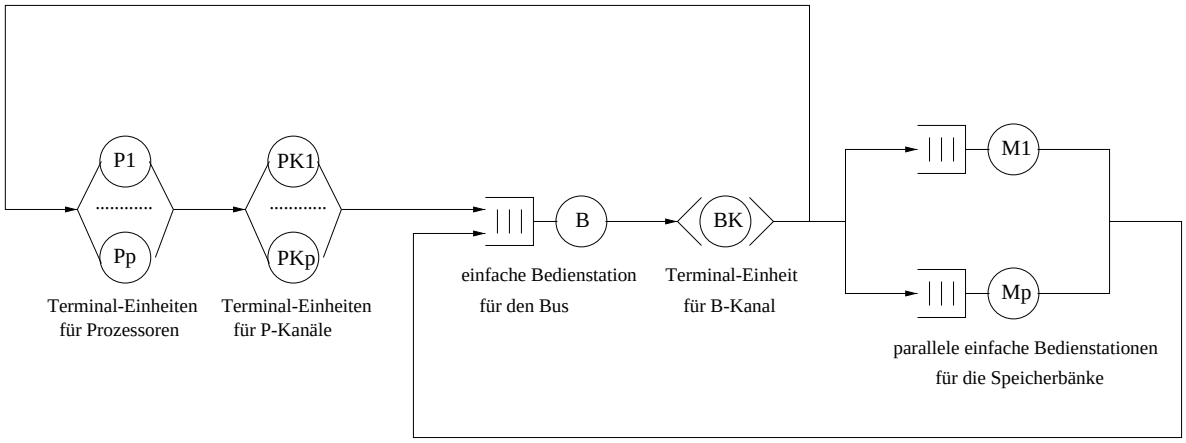


Abbildung 7.19: Das für die theoretische Analyse verwendete Model der PHOTOBUS-Architektur.

7.8.4 Vergleich mit theoretischer Modellierung

Wie in 5.5 beschrieben, werden bei der statistischen Modellierung mit Hilfe der Warteschlangentheorie Durchschnittswerte für die Wartezeiten der Anfragen auf dem B-Kanal und die Speicherbänke ermittelt.

Das Modell der PHOTOBUS-Architektur

Die PHOTOBUS-Architektur wird durch das in Abbildung 7.19 skizzierte einfache Modell dargestellt. Es basiert auf dem in Abschnitt 5.5.2 beschriebenen Modell eines paket-vermittelnden Busses, an dem mehrere Prozessoren und mehrere Speicherbänke angeschlossen sind. Es unterschiedet sich von dem Bus-Modell lediglich durch zwei zusätzliche Terminal-Bedienstationen: eine nach der Prozessor-Bedienstation und eine nach der Bus-Bedienstation.

Die erste Bedienstation modelliert die Verzögerung der Anfragen durch den P-Kanal. Darin sind die Bearbeitungszeit der Prozessorschnittstelle, die Latenz des optischen Kanals und die durch die Bandbreite und Nachrichtenlänge gegebene Sendezeit enthalten. Da jeder Prozessor eine eigene Schnittstelle und einen eigenen P-Kanal besitzt, können an dieser Stelle keine Warteschlangen auftreten. Aus diesem Grund wird für die Modellierung eine Terminal-Bedienstation verwendet.

Die zweite zusätzliche Bedienstation modelliert die reine Leitungs-Latenz des B-Kanals. Der Grund für eine getrennte Behandlung der Leitungs-Latenz-Verzögerung besteht darin, daß diese keinen Einfluß auf die Wartezeiten im Busknoten hat. So braucht der Transmitter mit dem Senden einer Nachricht nicht zu warten, bis die vorherige Nachricht tatsächlich beim Empfänger angekommen ist. Sobald die Daten einer Nachricht übertragen wurden, kann mit dem Senden der Nächsten begonnen werden. Dies bedeutet, daß die Leitungs-Latenz nicht in die Berechnung der durchschnittlichen Wartezeit im Bus-Chip eingehen darf, und daher nicht der Bearbeitungszeit der Bus-Chip-Bedienstation hinzugerechnet werden kann. Eine solche Trennung zwischen der Kanal-Latenz und der sonstigen Bearbeitungszeit kann hingegen nicht bei den Speicherbänken vorgenommen werden. Dies liegt daran, daß die Speicherbank erst freigegeben wird, wenn die Daten über den M-Kanal im Bus-Chip eingetroffen sind. Daraus folgt, daß die gesamte Übertragungszeit bei der Berechnung der Wartezeiten berücksichtigt werden muß. Die Kanal-Latenz muß also der Bearbeitungszeit der Speicherbank hinzugerechnet werden.

Vergleich mit der Simulation

Die Ergebnisse der Analyse sind in Abbildung 7.20 zu sehen. Dabei handelt es sich um das Verhältnis der Ausführungszeit auf einem idealen Parallelrechner mit Cache zu der Ausführungszeit auf der PHOTOBUS-Architektur. Die Berechnung dieser Werte wurde in Kapitel 5 erläutert. Ein Vergleich mit den in Abbildung 7.17 dargestellten Simulationsergebnissen zeigt, daß mit wenigen Ausreißern sowohl die Größenordnung der Werte als auch die Form der Kurve sehr gut mit der Simulation übereinstimmt. Bei den Ausreißern handelt es sich um die Programme, bei denen die Leistung bei hoher Prozessorzahl aufgrund der Belastung des Speichersystems durch Synchronisationsoperationen zusammenbricht. Dies ist besonderes deutlich bei Ocean ab 32, und bei LU bei

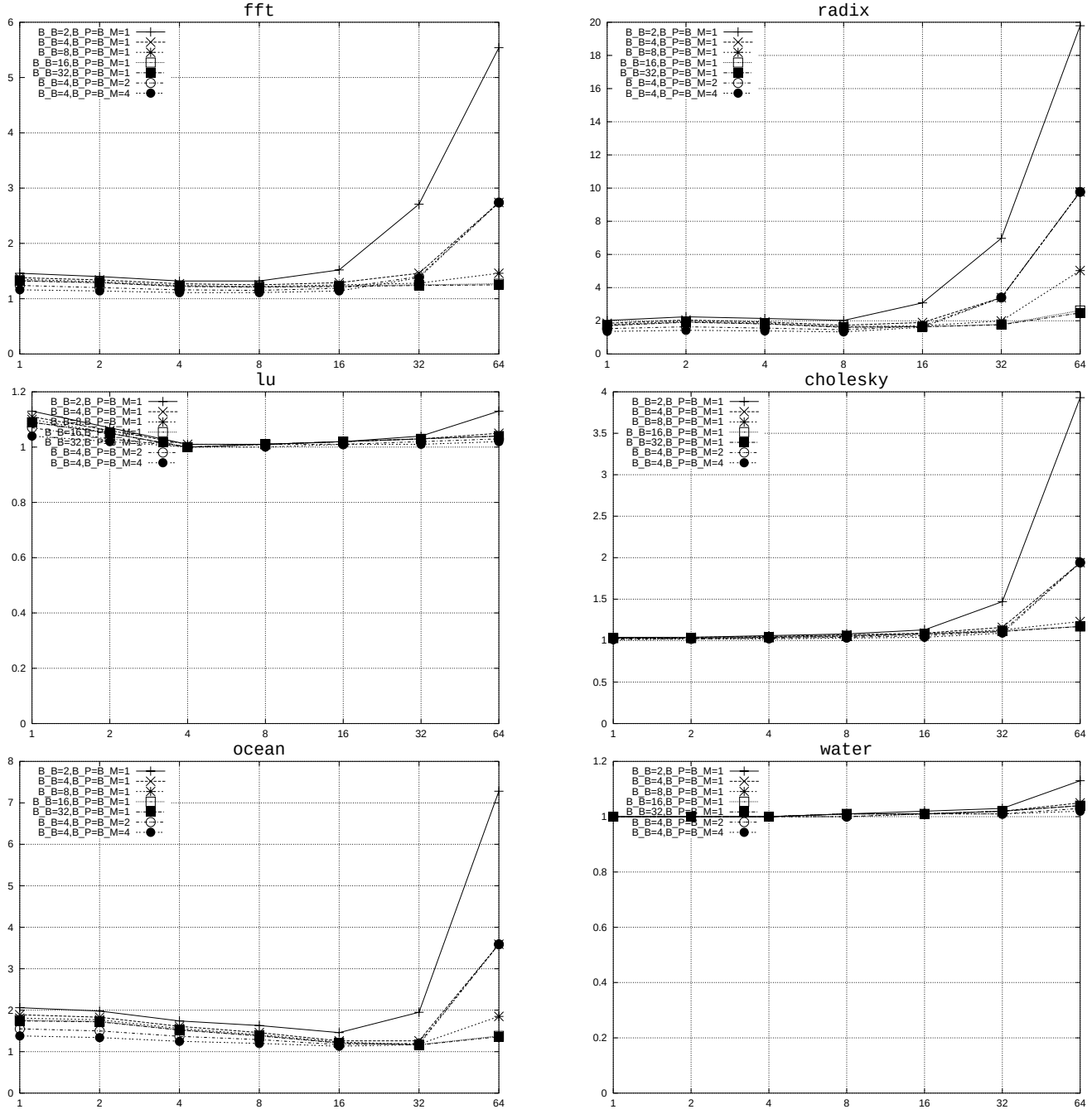


Abbildung 7.20: Die obigen Diagramme zeigen die durch theoretische Modellierung ermittelte Ausführungszeit der einzelnen Benchmarks im Vergleich zur Ausführungszeit mit einem idealen Speichersystem.

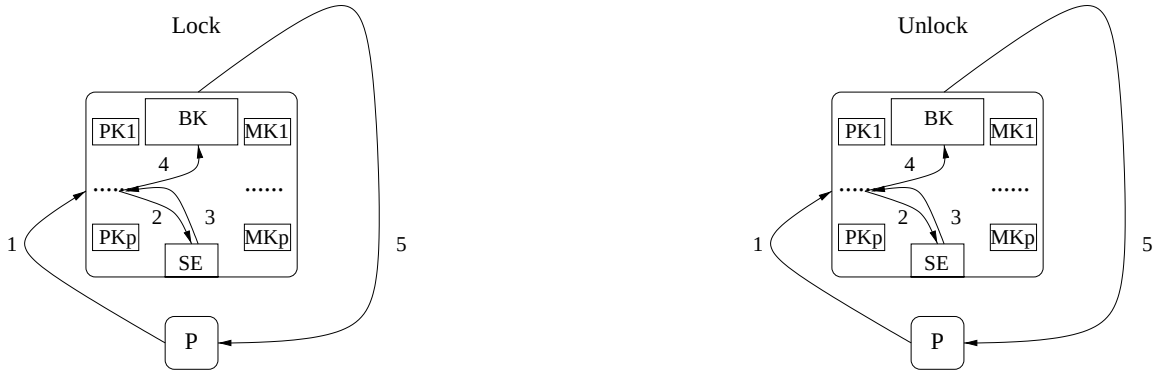


Abbildung 7.21: Der Ablauf der Kommunikation bei Lock- und Unlock-Operationen bei der on-Chip Synchronisation.

64 Prozessoren zu sehen. Das Phänomen wurde bereits im vorigen Abschnitt erläutert. Es kann von der hier betrachteten theoretischen Modellierung nicht wiedergegeben werden, da diese von den Synchronisationsoperationen abstrahiert.

Insgesamt kann man sagen, daß die Abweichungen zwischen der theoretischen Modellierung und der Simulation in dem Rahmen liegen, der aufgrund der gemachten Näherungen (siehe 5.5.4) zu erwarten war. Dies ist ein wichtiger Hinweis darauf, daß die Simulation das Verhalten der untersuchten Systeme weitgehend korrekt wiedergibt.

7.9 Optimierung der Synchronisation

Der Leistungsabfall des PHOTOBUS ab ca. 16 Prozessoren ist im hohen Maße auf die Synchronisation zurückzuführen. Dies liegt daran, daß die Synchronisation nach dem sog. Polling Prinzip durchgeführt wird. D.h., daß ein Prozessor nach einer fehlgeschlagenen Lock-Operation diese solange in kurzen Zeitabständen wiederholt, bis ihm die Sperre zugeteilt wird. Der Nachteil dieses Verfahrens besteht darin, daß es das Speichersystem sehr stark belastet. Wenn viele Prozessoren auf eine Sperre warten, dann wird der B-Kanal mit Lock-Anfragen überschüttet. Gleiches gilt für die Speicherbank, die Sperre beinhaltet. Dadurch wird die Arbeit der anderen Prozessoren verlangsamt, die nun sehr lange auf den B-Kanal bzw. auf die betroffene Speicherbank warten müssen. Dabei wird auch der Prozessor verlangsamt, der die Sperre besitzt. Dies führt dazu, daß die Wartezeit auf die Sperre steigt, was wiederum bedeutet, daß die hohe Belastung des B-Kanals länger anhält. Um das Problem zu lindern, gibt es verschiedene Verfahren (siehe z.B. [178]). So kann man das Laden der Sperr-Variablen in die Caches zulassen. Alternativ ist es möglich, die Abstände zwischen den wiederholten Anfragen der wartenden Prozessoren zu verlängern. Ein erheblicher Anstieg der Belastung des Speichersystems bei der Synchronisation vieler Prozessoren kann allerdings durch keines dieser Verfahren erreicht werden.

Ein Vorteil des PHOTOBUS Konzeptes besteht darin, daß es eine sehr effiziente Implementierung von Synchronisationsoperationen erlaubt, die eine Mehrbelastung des Speichersystems vermeidet. Die Grundidee besteht darin, die Synchronisation vollständig auf dem Bus-Chip abzuwickeln. Dazu bekommt der Chip, wie in Abbildung 7.22 zu sehen, eine Synchronisationseinheit (SE), die über einen Synchronisationsbus (SB) mit allen PEs verbunden ist. Die für die Synchronisation notwendige Kommunikation wird On-Chip über den SB durchgeführt. Eine Nachricht über den B-Kanal wird erst dann verschickt, wenn dem Prozessor die Sperre zugeteilt wurde.

Im Nachfolgenden wird zunächst der Ablauf der Synchronisation im Detail erläutert. Danach werden die benötigten Erweiterungen des Bus-Chips beschrieben. Die für diese Erweiterungen benötigte zusätzliche Chip-Fläche wird dann in Abschnitt 7.9.8 abgeschätzt.

7.9.1 Grundüberlegungen

Bei der On-Chip Synchronisation werden die, in den PEs ankommenden Lock- und Unlock-Anfragen nicht über den B-Kanal weitergeschickt. Statt dessen werden sie an die SE weitergegeben. Diese überprüft zunächst, ob die dazugehörende Sperr-Variable bereits besetzt ist, und teilt das Ergebnis der PE mit. Falls die Sperre frei

war, wird er als besetzt markiert. Er wird wieder freigegeben, wenn die SE von einer PE eine entsprechende Unlock-Anfrage bekommt.

Bei der Implementierung der On-Chip Synchronisation müssen vor allem zwei Aspekte bedacht werden:

1. Die SE kann nur eine begrenzte Anzahl von Sperren auf einmal verwalten. Wird diese Anzahl überschritten, dann kommt es zu einem Überlauf.

2. Die SE und die Sperren stellen zusätzliche Ressourcen, um die sich die PEs bewerben. Bei der Zuteilung dieser Ressourcen besteht die Gefahr von Verklemmungen.

Im Nachfolgenden werden die obigen Probleme und der in der Arbeit verfolgte Lösungsansatz genauer erläutert.

Überläufe in der SE

Das obige Verfahren setzt voraus, daß sich die SE bei jeder erfolgreichen Lock-Operation merkt, daß die Sperre mit der entsprechenden Adresse besetzt ist. Dies bedeutet, daß für jede besetzte Sperre in der ME eine Speicherstelle benötigt wird. Dabei wird man mit dem Problem konfrontiert, daß die Anzahl der Sperre nur durch die Größe des Adreßraums beschränkt ist. Theoretisch kann man für jede Adresse des Hauptspeichers eine Sperre vereinbaren und besetzten. Um alle diese Sperren zu verwalten, würde die ME die gleiche Speicherkapazität wie der Hauptspeicher benötigen. Dies ist aber nicht möglich. Es kann also passieren, daß der SE der Speicherplatz ausgeht und sie keine weiteren Lock-Operationen annehmen kann. Das Problem bei einer solchen Situation besteht darin, daß man die Ausführung der Lock-Operation nicht verzögern kann, bis der SE Speicher frei geworden ist. Im ungünstigen Fall kann das Freiwerden des Speicherplatzes nämlich von der Durchführung der neuen Lock-Operation abhängen. Dies wäre z.B. dann der Fall, wenn alle in der SE gespeicherten Sperren von dem gleichen Prozessor wie die neue Operation stammen. Um Verklemmung zu vermeiden, muß für die Ausführung der Lock-Operation in einem solchen Fall eine alternative Lösung gefunden werden. Eine Möglichkeit wäre eine reine Softwarelösung Betriebssystem-Ebene. Eine wäre allerdings schwer zu realisieren, da dazu das Betriebssystem jeder Zeit die Anzahl der im System vorhandenen Sperren kennen müßte. Diese Information ist aber nur auf dem Bus-Chip verfügbar und wäre ohne eine Hardware-Unterstützung für das Betriebssystem nicht verfügbar. Wir beschäftigen uns hier daher mit einer reinen Hardwarelösung. Unter der Annahme, daß der oben beschriebener Fall nur selten auftritt, muß diese Lösung nicht besonderes effizient sein. Sie muß aber sicherstellen, daß:

1. beliebig viele Sperren bearbeitet werden können,

2. durch die Verwendung von zwei unterschiedlichen Verfahren für die Sperren-Verwaltung keine Inkonsistenzen entstehen und

3. sobald im Speicher der SE wieder Platz ist, das System so schnell wie möglich wieder zu der effizienten SE-basierten Sperren-Verwaltung zurückkehrt.

Da die maximale Anzahl der Sperren durch die Größe des Hauptspeicher beschränkt ist, kann wegen Punkt 1 nur der Hauptspeicher selbst für die Speicherung der Sperren benutzt werden. Um Inkonsistenzen zu vermeiden, muß der Bus-Chip jederzeit wissen, ob Sperren nur in der SE oder auch im Hauptspeicher vorhanden sind. Im zweiten Fall folgt aus der Tatsache, daß eine Sperre nicht in der SE vermerkt ist, nicht unbedingt, daß er frei ist. Er kann auch im Hauptspeicher als besetzt gekennzeichnet sein. Sobald also auch nur eine Sperre in den Speicher ausgelagert wurde, muß bei jeder Lock-Operation auch im Hauptspeicher nachgesehen werden, ob die Sperre besetzt ist. Aus der dritten Forderung folgt, daß mit dem Freiwerden von Speicherstellen in der SE neu besetzte Sperren dort gespeichert werden müssen. Außerdem sollten auch Sperren aus dem Speicher in die SE verschoben werden.

Aus den obigen Überlegungen ergibt sich, daß für die Synchronisation drei Betriebsmodi benötigt werden:

Standardmodus: In diesem Modus wird die Synchronisation vollständig auf dem Bus-Chip abgewickelt. Dies ist der Regelfall, da es nur sehr wenige Anwendungen gibt, die viele verschiedene Sperren gleichzeitig verwenden. In der Regel werden Sperren verwendet, um Prozessoren an einer Stelle des Programms zu synchronisieren, so daß alle Prozessoren bzw. eine Prozessorgruppe auf die gleiche Sperre warten.

Überlaufmodus: Dieser Modus ist immer dann aktiv, wenn die SE 'voll' ist, also keine weiteren Sperren aufnehmen kann. Neue Sperren werden also im Speicher abgelegt. Lediglich die Synchronisationsanweisungen, die sich auf bereits in der SE gespeicherte Sperren beziehen, laufen auf dem Chip ab.

Rückkehrmodus: Der Rückkehrmodus wird nach dem Überlaufmodus aktiviert, wenn in der SE Platz frei wird. Neue Sperren werden wie im Standardmodus über die SE abgewickelt. Bei der Bearbeitung bestehender Sperren muß aber berücksichtigt werden, daß manche zuvor in den Speicher ausgelagert wurden. Diese werden nun nach und nach in die SE zurückgeholt, um eine Rückkehr zum Standardmodus zu ermöglichen.

Von den obigen Modi muß nur der erste effizient sein. Die anderen dienen der Behandlung von seltenen Ausnahmefällen. Wichtig ist lediglich, daß das System sobald wie möglich in den Standardmodus zurückkehrt, wenn in der SE Platz frei wird.

Verklemmungsfreiheit

In Abschnitt 7.5.2 wurde erläutert, daß eine Verklemmung immer dann auftritt, wenn zwei oder mehr Prozessoren gegenseitig aufeinander warten. Die Gefahr einer solchen Situation ergibt sich für das hier beschriebene Synchronisationsverfahren im Zusammenhang mit Supply-Operationen. Das Problem tritt auf, wenn ein Prozessor (P1) auf eine Sperre wartet, dessen Besitzer (P2) seinerseits auf eine Supply-Operation von P1 wartet. Um eine Verklemmung zu vermeiden, muß hier sichergestellt werden, daß die Supply-Operation durchgeführt wird, obwohl sich die PE von P1 im Wartezustand befindet. Würde man das für die Supply-Operation genutzte Kommunikationsprotokoll des ursprünglichen Bus-Chips (Abschnitt 7.1) ohne Änderungen übernehmen, wäre dies aber nicht der Fall. Dies liegt daran, daß der Prozessor nach diesem Protokoll keine neuen Daten an den Bus-Chip schickt, solange die PE nicht mit der Bearbeitung der alten Anfrage fertig ist und dies dem Prozessor mitgeteilt hat. Dies dient der Flußkontrolle und sorgt dafür, daß die PE nicht mit Anfragen überfordert wird. Bei der Synchronisation bewirkt es aber, daß die Supply-Operation nicht wie benötigt ausgeführt wird, was wiederum zu einer Verklemmung führt. Um dies zu vermeiden, müssen ausstehende Lock-Operation als Ausnahme behandelt werden. D.h., daß der Prozessor bei einer Lock-Operation zumindest Supply-Daten an den Bus-Chip schickt, ohne auf die Rückmeldung zu warten.

7.9.2 Ablauf der Synchronisation

Wie im vorigen Abschnitt erläutert, muß bei der Synchronisation zwischen drei Betriebsmodi unterschieden werden. Der Ablauf der Synchronisation in diesen Modi kann wie folgt zusammengefaßt werden:

Standardmodus

LOCK: Bei einer Lock-Operation schickt der Prozessor über seinen P-Kanal das Befehlswort und die Adresse der Sperre an den Bus-Chip. Sobald die zugehörige PE die Anfrage empfangen hat, bemüht sie sich um den Zugriff auf die SE und leitet die Anfrage über den SB an sie weiter. Falls die gewünschte Sperre frei ist, schickt die SE eine Erfolgsmeldung an die PE zurück und markiert die Adresse als besetzt. Von der PE gelangt die Nachricht über den B-Kanal zu dem wartenden Prozessor. Ist die Sperre bereits besetzt, so teilt die SE dies der PE mit. Diese geht daraufhin in einen Wartezustand über. Sie verbleibt in dem Zustand, bis sie eine Unlock-Operation auf die gewünschte Sperre auf dem SB bemerkt. Sie bemüht sich dann wieder um den Zugriff auf die SE und gibt die Anfrage erneut an sie weiter.

UNLOCK: Um eine Unlock-Operation durchzuführen, schickt der Prozessor das entsprechende Befehlswort und die Sperren-Adresse über seinen P-Kanal. Daraufhin verschafft sich die PE Zugriff auf die SE und teilt ihr die Freigabe der Sperre über den SB mit. Dabei wird die Unlock-Operation von allen den PEs mitgehört, die sich aufgrund einer fehlgeschlagenen Lock-Operation im Wartezustand befinden. Zum Schluß benachrichtigt die PE den Prozessor über den B-Kanal, daß sie die Unlock-Operation beendet. Der Prozessor weiß dadurch, daß er nun neue Anfragen an den Bus-Chip schicken kann.

Überlaufmodus

LOCK im Überlaufmodus: Nach Erhalt einer Lock-Anfrage prüft die SE, ob die entsprechende Sperre in ihrem Speicher als besetzt gekennzeichnet ist und teilt das Ergebnis über den SB der PE mit. Falls die Sperre nicht in der SE vermerkt war, dann fordert die PE den Zugriff auf die entsprechende Speicherbank und den B-Kanal an und leitet die Lock-Anfrage an den Speicher weiter. Die Speicherbank überprüft den Zustand der Sperre, schickt die Antwort über den M-Kanal an den Bus-Chip zurück und markiert die

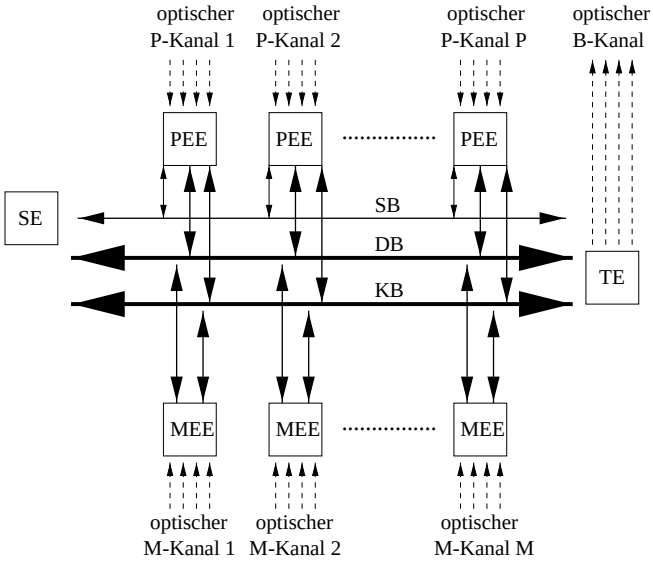


Abbildung 7.22: Die Architektur des Bus-Chips in der PHOTOBUS-Architektur mit on-Chip Synchronisation.

Sperre bei Bedarf als besetzt. Die Antwort wird von der zugehörigen ME empfangen und über den SB an die SE und die PE weitergegeben. Falls die Lock-Operation erfolgreich war, dann verschafft sich die PE erneut Zugang zum B-Kanal und schickt die Erfolgsmeldung an den Prozessor. Gleichzeitig erhöht die SE den Zähler für die im Speicher befindlichen Sperren. Falls die Operation fehlgeschlagen ist (weil die Sperre bereits besetzt war), dann geht die PE wie im Standardmodus in den Wartezustand über. Dies gilt unabhängig davon, ob die Sperre in der SE oder im Speicher als besetzt markiert ist.

UNLOCK im Überlaufmodus: Bei der Unlock-Operation im Überlaufmodus muß zwischen den in der SE, und den im Speicher vermerkten Sperren unterschieden werden. Falls die SE die entsprechende Sperre in ihrem Speicher findet, dann löscht sie den Eintrag und teilt den Erfolg den PEs über den SB mit. Diese verfahren dann wie im Standardmodus. Da nun eine Speicherstelle frei wird, geht die SE gleichzeitig in den Rückkehrmodus über.

Für den Fall, daß die Sperre nicht in der SE zu finden ist, wird zuerst geprüft, ob irgendeine der PEs auf genau diese Sperre wartet. Dazu schickt die ME eine entsprechende Nachricht auf dem SB. Wenn eine PE eine passende Lock-Anfrage besitzt, dann verschafft sie sich den Zugang zum SB und benachrichtigt die SE. Außerdem fordert sie den Zugriff auf den B-Kanal an und schickt eine Erfolgsmeldung über den B-Kanal. Diese Meldung gilt sowohl für den Urheber der Lock- als auch für den Urheber der Unlock-Anfrage. Die Operation ist damit beendet.

Falls sich in keiner der wartenden PEs eine passende Lock-Anforderung befindet, dann dekrementiert die SE den Zähler für die im Speicher befindlichen Sperren und gibt der PE Bescheid, von der die Unlock-Anfrage ausgegangen ist. Diese schickt dann über den B-Kanal eine Nachricht, die den Prozessor über das Ende der Operation informiert. Diese Nachricht veranlaßt gleichzeitig die betroffene Speicherbank, die Sperre als frei zu markieren.

Rückkehrmodus

LOCK im Rückkehrmodus: Falls die gewünschte Sperre bereits in der SE als besetzt markiert ist, dann verläuft eine Lock-Operation im Rückkehrmodus genauso wie im Standardmodus. Ansonsten wird zunächst die Sperre in der SE als besetzt gespeichert. Daraufhin wird wie im Überlaufmodus über den B-Kanal eine Nachricht an die zugehörige Speicherbank geschickt. Allerdings dient diese Nachricht nicht nur der Nachfrage, ob die Sperre bereits besetzt ist. Sie veranlaßt die Speicherbank auch, die betroffene Sperre nach der Überprüfung und dem Verschicken der Antwort aus dem Speicher zu entfernen. Das weitere Vorgehen hängt davon ab, ob die Sperre in der Speicherbank bereits als besetzt markiert war oder nicht. Im ersteren Fall wird nach dem Eintreffen der Antwort in der ME entweder die entsprechende PE wieder in

den Wartezustand versetzt. Gleichzeitig dekrementiert die SE den Zähler für die im Speicher verbliebenen Sperren. Im zweiten Fall wird die Erfolgsmeldung über den B-Kanal an den Prozessor weitergeben.

UNLOCK im Rückkehrmodus: Auch bei der Unlock-Operation unterscheidet sich der Rückkehrmodus nur dann von dem Standardmodus, wenn sich die betroffene Sperre im Hauptspeicher befindet. Falls die SE bei einer Unlock-Operation die Sperre nicht in ihrem Speicher findet, dann dekrementiert sie zunächst den Zähler für die im Speicher befindlichen Sperren. Daraufhin veranlaßt sie die PE, sich Zugriff auf die entsprechende Speicherbank und den B-Kanal zu verschaffen und eine Unlock-Anforderung zu übertragen. Da die PE nun nichts mehr mit der Anfrage zu tun hat, wird diese Anforderung gleichzeitig von dem Prozessor als Erfolgsmeldung interpretiert. Die Speicherbank löscht nach dem Empfang der Anforderung der Sperre und meldet den Erfolg über ihren B-Kanal an den Bus-Chip.

7.9.3 Der SB

Für die Systemeffizienz ist es vor allem wichtig, daß die Synchronisation den B-Kanal nicht übermäßig belastet. Da ein Zugriff auf den B-Kanal gleichbedeutend mit einem Zugriff auf den Datenbus ist, ist es nicht sinnvoll, den Datenbus für die Synchronisation zu benutzen. Aus diesem Grund wird die Kommunikation mit der SE durch einen speziellen Bus, den SB, abgewickelt. Da für die Synchronisationsoperationen lediglich Adressen übertragen werden, braucht der SB keine besonders hohe Bandbreite. Dies bedeutet, daß der Bus mit wesentlich weniger Leitungen als der Datenbus auskommt. Es ist außerdem möglich, ihn als einen leitungs-vermittelden Bus zu realisieren. Dadurch werden die Busverwaltung und die Zugriffsprotokolle vereinfacht.

7.9.4 Die Arbitrierungshardware

Bevor eine PE eine Synchronisationsoperation durchführen kann, muß sie sich den Zugriff auf die SE verschaffen. Dabei wird immer gleichzeitig der Zugriff auf die SB und den SE vergeben. Sie bleibt solange im Besitz der SE, bis sie eine eindeutige Antwort auf die Synchronisationsanfrage bekommen hat. Im Überlauf- und Rückkehrmodus bedeutet dies, daß eine Anfrage der PE an den Speicher weitergeleitet wird und die PE bis zur Ankunft der Antwort des Speichers im Besitz der SE und des SB bleibt. Dies wiederum bedeutet, daß die ME den Zugriff auf den SB nicht extra anfordern muß, um diese Antwort an die PE und die SE weiterzuleiten. Die Arbitrierung findet also ausschließlich unter PEs statt. Sie wird nach dem gleichen Prinzip wie die Arbitrierung des B-Kanals und der MEs durchgeführt. Dazu wird dem KB eine weitere Arbitrierleitung hinzugefügt: die GS-Leitung. Diese Leitung wird in gleicher Weise benutzt wie die GB- und die GM $_i$-Leitungen, die der Arbitrierung des B-Kanals und der Speicherbänke dienen. Sie verfügt also für jede PE über eine Arbitrierschaltung, wie sie in Abschnitt 7.5.2 beschrieben wurde. Die PEs sind an die zugehörigen Schaltungen über zwei Leitungen angeschlossen: eine Request-Leitung (RS-Leitung) und eine Grant-Leitung (GS). Sie können in der Anfrage-Phase über die RS-Leitung den Zugriff auf die SE anfordern und bekommen ihn über die GS-Leitung in der Antwort-Phase zugeteilt. Sobald eine PE die SE und den SB nicht mehr benötigt, setzt sie in der Anfrage-Phase die Request-Leitung auf Null und gibt sie damit frei.

7.9.5 Die SE

Die SE muß in der Lage sein,

1. über den SB mit den PEs und MEs zu kommunizieren,
2. die Adressen der besetzten Sperren zu speichern,
3. die verschiedenen Betriebsmodi (Standardmodus, Überlaufmodus und Rückkehrmodus) zu verwalten,
4. gemäß des Betriebsmodus und des Inhalts des Sperren-Speichers auf Lock- und Unlock-Anfragen zu reagieren.

Aufbau der SE

Eine entsprechende Architektur ist in Abbildung 7.23 zu sehen. Sie besteht aus einer Busschnittstelle (BS), einem Zwischenspeicher (ZS), Cache (C) , einem Zähler (Z), und einer Kontrolleinheit(CTRL). Die BS beinhaltet

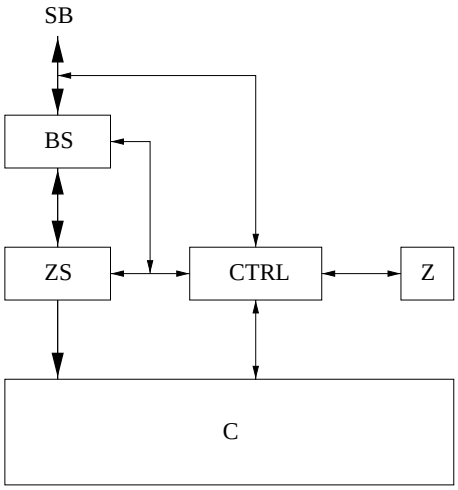


Abbildung 7.23: Der Aufbau der Synchronisationseinheit (SE).

bidirektionale Treiber für die Datenübertragung auf dem SB. Sie ist an den ZS angeschlossen, in dem alle an die SE gerichteten Anfrage gehalten und bearbeitet werden. Die ZS ist ihrerseits mit der Kontrolleinheit und dem Cache verbunden. An die Kontrolleinheit leitet sie die Befehlsworte der Anfragen weiter. Außerdem kann sie von der Kontrolleinheit neue Befehlsworte empfangen, um sie dann als Antwort über die BS an den SB weiterzugeben. Die Kontrolleinheit kann den ZS auch veranlassen, die in der Anfrage enthaltene Adresse an den Cache weiterzugeben. Dieser speichert für jeden besetzten Sperre dessen Adresse. Er ist über Kontrollleitungen mit der CTRL verbunden. Diese Leitungen bestimmen, was der Cache mit den Daten aus dem ZS machen soll.

Funktion der SB: Überblick

Das Prinzip der Arbeitsweise der SE wurde bereits in Abschnitt 7.9.2 deutlich. Im Nachfolgenden wird die Beschreibung formalisiert, indem für jeden der drei Betriebsmodi (Standardmodus, Überlaufmodus und Rückkehrmodus) ein endlicher Automat angegeben wird. Die Zustandsübergangsdiagramme der drei Automaten sind in Abbildung 7.24 dargestellt. Die Übergänge zwischen den Zuständen eines Betriebsmodus werden durch vier Arten von Ereignisse gesteuert:

1. die Anfragen der PEs ('LOCK', 'UNLOCK', bzw. 'NONE', falls keine Anfrage auf dem SB anliegt),

2. den Zustand einer Sperre im SE-Cache ('frei', falls die Sperre nicht im Cache vorhanden, bzw.' besetzt', falls er dort vermerkt ist),

3. die von einer ME übermittelte Antwort des Hauptspeichers auf eine Lock-Anfrage (ebenfalls 'frei' oder 'besetzt') und

4. das Ende bestimmter durch die SE durchgeführter Aktionen. Die zugehörigen Kanten wurden in Abbildung 7.24 mit 'fertig' markiert.

In allen drei Modi ist der Anfangszustand der Frei-Zustand (F). Er bedeutet, daß die SE untätig ist und auf Anfragen von den PEs wartet. Wird eine Nachricht auf dem SB empfangen, so wechselt die SE je nach Art der Anfrage in den LOCK (L) bzw. UNLOCK (U) Zustand. In beiden Fällen wird zunächst geprüft, ob die Sperre in der SE bereits gespeichert ist ('besetzt' ist) oder nicht ('frei' ist). Der weiterer Ablauf hängt von dem Ausgang der Abfrage und dem Betriebsmodus der SE ab.

Übergänge zwischen den Modi finden immer nur dann statt, wenn die SE in den F-Zustand zurückkehrt.

Funktion der SB: Standardmodus

Im Standardmodus werden für die Abwicklung der Synchronisationen nur 3 zusätzliche Zustände benötigt:

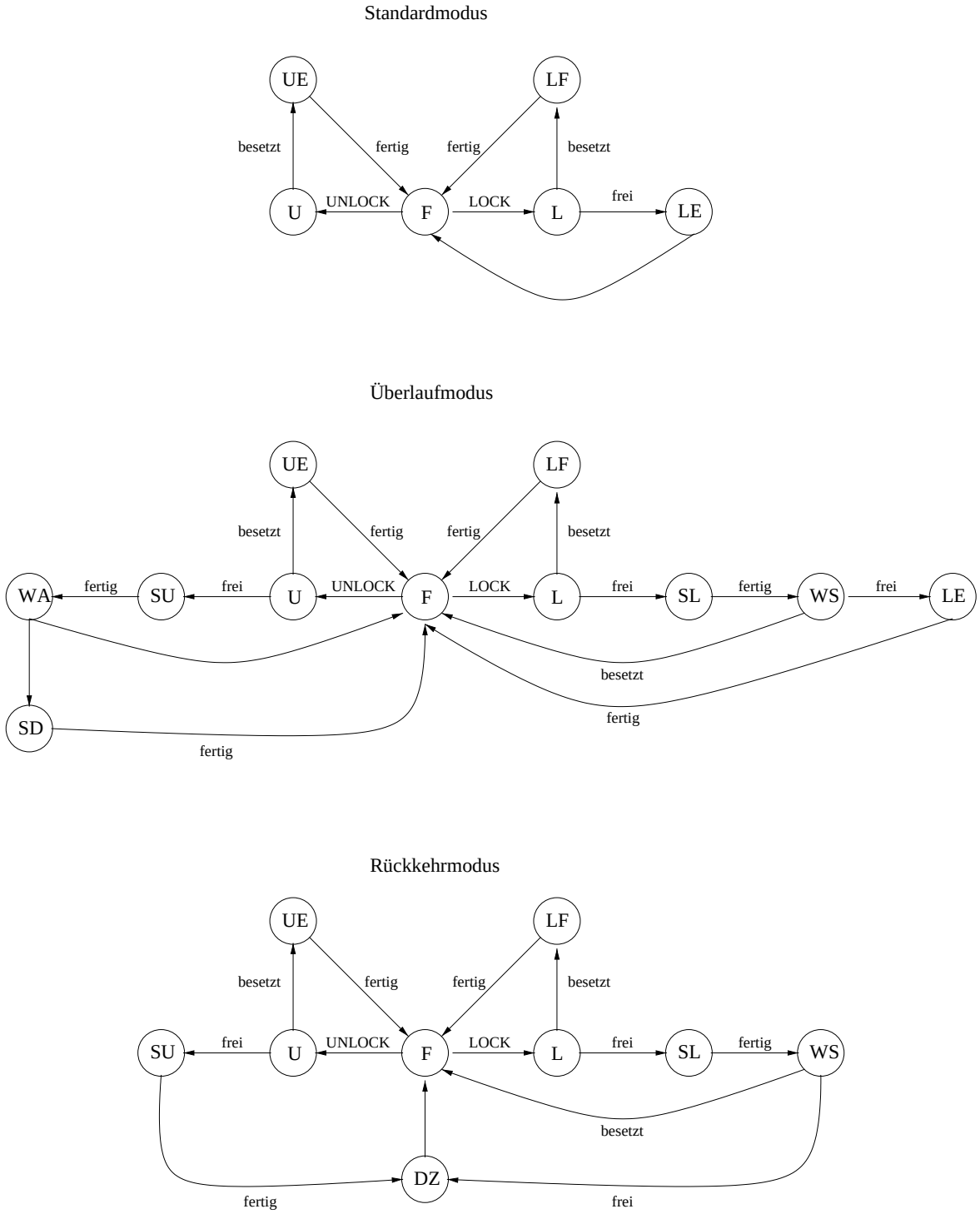


Abbildung 7.24: Die Zustandsübergangsdiagramme der SE in den einzelnen Betriebsmodi.

Standard LOCK Erfolg (LE): Die SE geht in den LE-Zustand über, wenn die Sperre noch nicht in der SE als besetzt vermerkt ist. Sie gibt auf dem SB eine Erfolgsmeldung aus, trägt die Adresse im Cache ein und geht in den F-Zustand über.

Standard LOCK Fehlschlag (LF): Die LE gelangt aus dem L-Zustand in den LF-Zustand, wenn der LOCk bereits im Cache vorhanden ist. Sie gibt dann eine Fehlschlag-Meldung auf dem SB aus und geht in den F-Zustand über.

Standard UNLOCK Erfolg (UE): Der UE-Zustand folgt aus dem U-Zustand, sobald die Sperre im Cache gefunden wurde. Die SE löscht den Eintrag aus ihrem Cache, schickt eine Erfolgsmeldung über den SB und geht in den F-Zustand über.

Funktion der SB: Überlaufmodus

Anfragen bezüglich Sperren, die sich nicht im SE-Cache befinden, müssen im Überlaufmodus an den Speicher weitergeleitet werden. Dies wird in dem zugehörigen Diagramm durch die zusätzlichen Zustände berücksichtigt, die die SE aus dem L- bzw. U- Zustand bei einer 'frei'-Antwort des Caches durchläuft.

Überlauf LOCK Fehlschlag (LF): Dieser Zustand ist mit dem Standard-LF-Zustand identisch.

Überlauf Speicher-LOCK (SE): Der SE-Zustand folgt aus dem L-Zustand, falls die Sperre nicht im Cache gefunden wurde. Die SE informiert daraufhin die PE, daß sie die Anfrage an den Speicher weiterleiten muß und geht in den WS-Zustand über.

Überlauf Warte auf Speicherantwort (WS): Die SE verbleibt im WS-Zustand, bis die Antwort auf eine Lock-Anfrage vom Speicher eingetroffen ist. Je nachdem, ob die Sperre frei oder besetzt ist, geht sie dann in den LE- oder den LF-Zustand über.

Überlauf LOCK Erfolg (LE): Der LE-Zustand bedeutet, daß der Speicher eine erfolgreiche Ausführung einer Lock-Operation gemeldet hat. Da die Anfrage an den Speicher durch die PE gehandhabt wird, braucht diese anderes als im Standardmodus nicht mehr benachrichtigt werden. Die SE inkrementiert daher nur den Zähler für im Speicher befindliche Sperren und kehrt in den F-Zustand zurück.

Überlauf UNLOCK Erfolg (UE): Dieser Zustand ist mit dem UE-Zustand des Standardmodus bis auf die Tatsache identisch, daß ihm ein Übergang in den Rückkehrmodus folgt.

Überlauf Speicher-UNLOCK (SU): Die SE gelangt in den SU-Zustand, wenn bei einer Unlock-Anfrage die Sperre nicht im Cache gefunden wird. Sie fragt daraufhin über den SB an, ob es wartende PEs mit einer Lock-Anforderung für den betroffenen Sperre gibt, und geht in den WA-Zustand über. Da die Nachricht auf dem SB auch von der PE mitgehört wird, von der die Unlock-Anfrage stammt, ist eine gesonderte Erfolgsmeldung nicht notwendig.

Überlauf Warte auf Speicherantwort (WA): Im WA wartet die SE darauf, daß eine PE die gerade freigegebene Sperre in Besitz nimmt. Sie verbleibt in diesem Zustand genau einen Zyklus lang. Falls sich innerhalb dieser Zeit keine PE meldet, geht sie in den SD-Zustand über. Sonst folgt der F-Zustand.

Überlauf Speicher-Delete (SD): Im SD-Zustand veranlaßt die SE die PE, von der die UNLock-Anfrage stammt, diese an den Hauptspeicher weiterzugeben. Sie dekrementiert dann den Zähler für im Speicher vorhandene Sperren und geht in den F-Zustand zurück.

Funktion der SB: Rückkehrmodus

Wie in Abbildung 7.24 unten zu sehen, ähnelt der Rückkehrmodus stark dem Überlaufmodus. So haben die U-,UE-, L-, LF- und WS-Zustände exakt die gleiche Funktion. Der Unterschied zwischen den beiden Modi besteht darin, daß Sperren aus dem Speicher zurück in den SE-Cache geholt werden. Dies bedeutet zum einen, daß Unlock-Operationen immer an den Speicher weitergeleitet werden. Dadurch entfällt der WA-Zustand. Gleichzeitig wird der SD-Zustand durch einen Dekrementiere-Zähler-Zustand (DZ) ersetzt. Zum anderen werden Lock-Operationen anders abgewickelt, so daß der LE-Zustand entfällt und die SU- und SL-Zustände eine andere Funktion haben. Die Änderungen können wie folgt beschrieben werden:

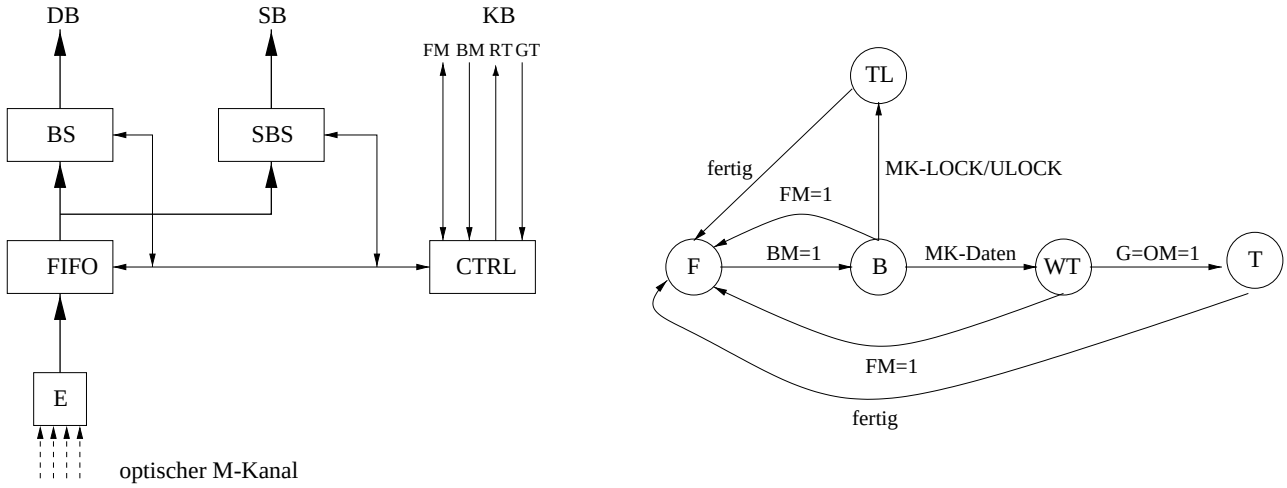


Abbildung 7.25: Der Aufbau (links) und das Zustandsdiagramm der ME in der PHOTOBUS-Architektur mit on-Chip Synchronisation.

Rückkehr Speicher-LOCK (SL): Im SL-Zustand wird die PE über den SB veranlaßt, die Lock-Anfrage an den Speicher weiterzuleiten. Anders als im Überlaufzustand soll dabei der Speicher auf jeden Fall die Sperre löschen. Gleichzeitig wird die Sperre im SE-Speicher als besetzt vermerkt.

Rückkehr Speicher-UNLOCK (SU): Die SE kommt aus dem U-Zustand in den SL-Zustand, wenn die Sperre nicht im SE-Cache gefunden wurde. Sie veranlaßt dann die PE, die Operation über den B-Kanal an den Speicher weiterzugeben. Da diese Nachricht auch von dem Prozessor als Erfolgsmeldung bewertet wird, ist die Operation damit beendet.

Rückkehr Dekrementiere Zähler (DZ): Im DZ-Zustand wird der Zähler für die im Speicher gespeicherten Sperren dekrementiert. Danach geht die SE in den F-Zustand zurück. Falls keine Sperren mehr im Speicher vorhanden sind, findet ein Wechsel in den Standardmodus statt.

7.9.6 Die ME

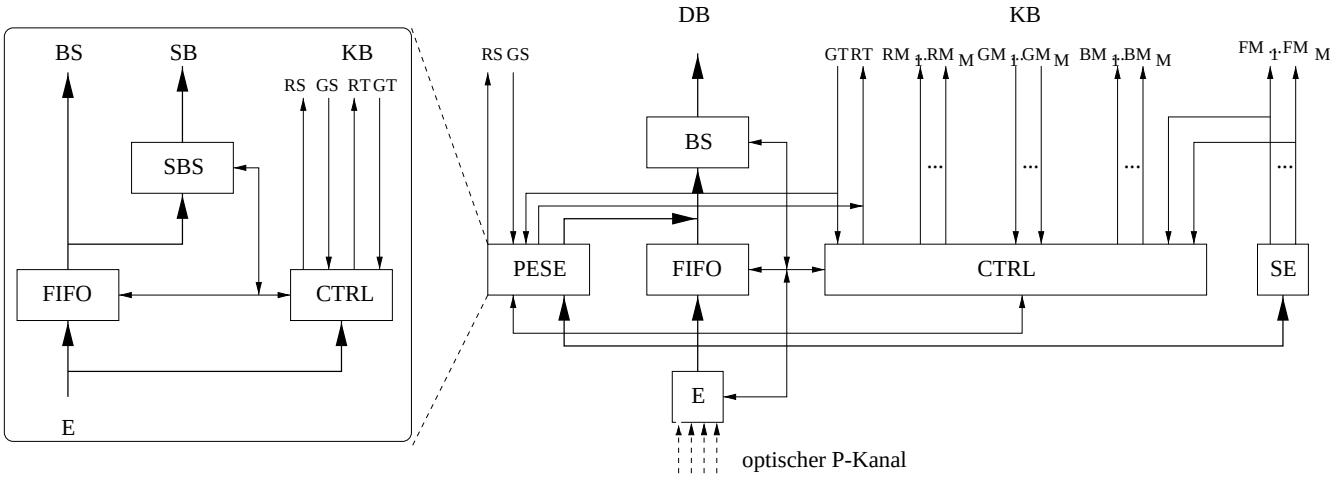
Antworten auf Lock-Anfragen, die an den Speicher weitergegeben wurden, müssen von der ME über den SB an die PEs und die ME weitergeleitet werden. Die ME kann dabei davon ausgehen, daß sie in dem Moment, in dem die Antwort von der Speicherbank eintrifft, sofort auf den SB zugreifen kann. Wie in Abschnitt 7.9.2 beschrieben, befinden sich zu diesem Zeitpunkt die SE und der SB im Besitz der PE, von der die Anfrage stammt. Sowohl die PE als auch die SB greifen in dieser Situation nur lesend auf den SB zu, da sie auf die Daten von der ME warten. Die ME muß den Zugang zum SB also nicht extra anfordern.

Um die obige Funktion zu erfüllen, benötigt die ME lediglich eine Schnittstelle zum SB. Außerdem muß die Kontrolleinheit so erweitert werden, daß sie Lock-Operationen erkennt und die Daten zum SB leitet. Die Arbeitsweise der modifizierten ME kann durch die Erweiterung des in 7.11 dargestellten endlichen Automaten formal beschrieben werden. Wie in Abbildung 7.25 zu sehen, wird hierzu nur ein zusätzlicher Zustand, der Übertrage-Lock (TL)-Zustand, benötigt. Die ME gelangt in diesen Zustand aus dem B, wenn die Rückmeldung über die Durchführung einer Lock-Operation auf dem M-Kanal eintrifft. Sie überträgt dann die Daten auf dem SB und geht in den F-Zustand zurück.

7.9.7 Die PE

Die Aufgabe der PE bei Synchronisationsoperationen geht aus der Beschreibung in Abschnitt 7.9.2 hervor. Sie kann wie folgt zusammengefaßt werden:

1. Die PE muß nach der Ankunft einer Synchronisations-Anfrage den Zugang zur SE anfordern, und die Anfrage an die SE weiterleiten.



a) PESE

a) Einbindung der PESE in die PE

Abbildung 7.26: Der Aufbau der PESE (Bild a) und ihre Einbindung in die PE (Bild b).

2. Die PE muß das Ergebnis der Operation über den B-Kanal an den Prozessor zurückgeben.

3. Falls die SE eine Anfrage mit einem L-MEMORY oder U-MEMORY Befehl beantwortet, dann muß die PE diese an die entsprechenden Speicherbank weiterleiten. Dazu muß sie sich Zugang zur Speicherbank und zum B-Kanal verschaffen und die Daten an den Transmitter weitergeben.

4. Nach einer fehlgeschlagen Lock-Operation muß die PE den SB überwachen und auf einen entsprechenden OK- oder L-CHANGE Befehl warten. Daraufhin muß sie sich Zugang zur SE verschaffen und die Anfrage erneut stellen, bzw. einen CONFIRM-Befehl absetzen.

Die Funktion der PE bei anderen Operationen ist von der On-Chip Synchronisation nicht betroffen und mit der in Abschnitt beschriebenen 7.5.5 identisch. Um die Betrachtung der für die On-Chip Synchronisation notwendigen Erweiterungen zu erleichtern, gehen wir daher davon aus, daß für die Synchronisation innerhalb der PE eine eigene Untereinheit zuständig ist. Sie wird als PESE (für PE-Synchronisationseinheit) bezeichnet. Die PESE holt alle über den P-Kanal ankommenden Synchronisationen aus dem Datenstrom heraus und bearbeitet sie. Die übrigen Komponenten der PE haben also mit der Synchronisation nichts zu tun. Sie entsprechen der Beschreibung in Abschnitt 7.5.5 und brauchen hier nicht weiter betrachtet zu werden.

Aufbau der PESE

Um die geforderte Funktionalität zu realisieren, benötigt die PESE eine SB-Schnittstelle, eine zusätzliche Kontrollogik und einen Zwischenspeicher. Ihr Aufbau ist in Abbildung 7.26 rechts dargestellt. Sie ist, wie in Abbildung 7.26 links zu sehen, an die eigentliche Kontrolllogik der PE, die Schnittstelle zum DB und zum KB, und den Ausgang des Empfängers angeschlossen.

Funktionsweise der PESE

Die Funktionsweise der PESE kann mit Hilfe eines endlichen Automaten mit 10 Zuständen formal dargestellt werden. Das zugehörige Zustandsübergangsdiagramm ist in Abbildung 7.27 zu sehen. Die Zustandsübergänge werden durch drei Arten von Ereignissen hervorgerufen: Die Synchronisationsanfragen des Prozessors, die auf dem SB übertragenen Befehle und die Signale auf den GS-, GM_i- und GT-Leitungen. Hinzu kommen Übergänge, die die PESE am Ende bestimmter Aktionen durchführt ('fertig').

Den Ausgangszustand des Automaten bildet der Frei (F) Zustand, der einer untätigen PE entspricht. Sie verläßt diesen Zustand, sobald eine Lock- oder Unlock-Anfrage über den P-Kanal eintrifft und beginnt mit ihrer Bearbeitung. Dabei durchläuft sie die folgenden Zustände:

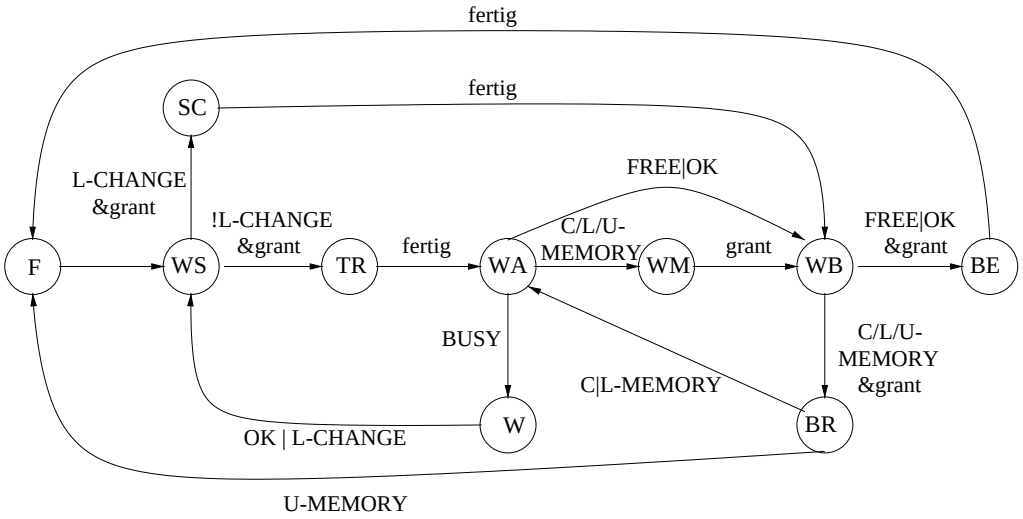


Abbildung 7.27: Das Zustandsdiagramm der PESE.

Warten auf die SE (WS): Im WS-Zustand wartet die PESE darauf, daß ihr der Zugriff auf die SE gewährt wird. Sie setzt daher in ihre RS-Leitung auf 1. Sobald ihr die SE zugeteilt wird, wechselt sie in den TC- oder TR-Zustand über.

Übertrage Confirm (TC): Der TC-Zustand ist für die Übertragung einer CONFIRM Meldung an die SE da. Am Ende der Übertragung geht die PESE in den WB-Zustand über, um die Rückmeldung an den Prozessor zu senden.

Übertrage Anfrage (TR): Im TR-Zustand wird die Prozessor-Anfrage an die SE weitergegeben. Die PESE geht dann zum WA-Zustand weiter.

Warten auf Antwort (WA): Die PESE befindet sich im WA-Zustand, wenn sie auf das Ergebnis einer Operation wartet. Sie verläßt den Zustand, sobald das Ergebnis auf dem SB übertragen wird. Falls es sich dabei um eine OK- oder FREE-Meldung handelt, folgt der WB-Zustand. Bei einer BUSY-Meldung geht die PESE in den Wartezustand W über. Wenn die SE mit einem L-, U- oder C-MEMORY antwortet, dann wechselt die PESE in den WS-Zustand.

Warten auf die Speicherbank (WM): Die PESE gelangt in den WM-Zustand, wenn sie den Zugriff auf eine Speicherbank benötigt. Sie setzt in der Anfragephase die entsprechende RM_i-Leitung auf 1 und wartet darauf, daß ihr die Speicherbank zugeteilt wird (die korrespondierende GM_i-Leitung auf 1 geht). Sie wechselt dann in den WB-Zustand.

Warten auf den B-Kanal (WB) Die PESE befindet sich im WB-Zustand, wenn sie auf den B-Kanal wartet. Sie verläßt ihn, sobald ihr der Zugriff auf den Kanal gewährt wird (die GT-Leitung auf 1 geht).

Broadcast Ergebnis (BE): Im BE-Zustand wird die Erfolgsmeldung über den B-Kanal an den Prozessor geschickt. Danach kehrt die PESE in den F-Zustand zurück.

Warten (W): Die PESE befindet sich im W-Zustand, wenn die von ihr angeforderte Sperre besetzt ist. Sie überwacht den SB und wartet auf eine 'OK' oder 'L-PROVIDE' Nachricht, die die von ihr gewünschte Sperre betrifft. Sie wechselt dann in den WS-Zustand, um den Zugriff auf die SE zu bekommen.

7.9.8 Für die Synchronisation zusätzlich benötigte Chipfläche

Die Synchronisationshardware wirkt sich in fünf Bereichen auf die Komplexität der Schaltungen des Bus-Chips aus:

1. Es werden zusätzliche Leitungen für den SB sowie die dazugehörigen Anschlüsse benötigt.

2. Es wird eine GS-Leitung mit den zugehörigen Arbitrierschaltungen für jede PE benötigt.
3. Es wird zusätzliche Chip-Fläche für die Implementierung der SE benötigt.
4. Jede ME benötigt eine SB-Schnittstelle und eine zusätzliche Logik
5. Jede PE benötigt eine PESE mit der dazugehörigen SB-Schnittstelle, FIFO und Kontrollogik.

Der Flächenbedarf der obigen zusätzlichen Komponenten kann an Hand der Beschreibung im vorhergehenden Abschnitten leicht abgeschätzt werden. Dabei kann die in Abschnitt 7.16 beschriebene Vorgehensweise für die Abschätzung des Flächenbedarfs des Bus-Chips ohne Synchronisationshardware übernommen werden. Im Nachfolgenden werden daher lediglich die Ergebnisse zusammengefaßt. Auf eine ausführliche Herleitung, wie sie in 7.16 zu finden ist, wird verzichtet.

Annahmen

Zusätzliche Leitungsfläche

Die für die Leitungen benötigte Fläche erhöht sich um die Fläche der N_{SB}-Leitungen des SB, die GS-Leitung und die Anschlüsse, die diese Leitungen mit den PEs, den MEs und der SE verbinden. Durch Einsetzen der zusätzlichen Leitungszahl in Gleichungen 7.57 ergibt sich für die gesamte Leitungsfläche:

$$A_B = ((P+M)(N_{PK} \cdot V_F + \mathbf{N_{SB}}) + 4P(M+1) + 6) \cdot (N_{PK} \cdot V_F + 3M + 3 + \mathbf{N_{SB}}) \cdot \left(\frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \right)^2 \quad (7.95)$$

Fläche der Arbitrierungslogik

Für die Durchführung der On-Chip Synchronisation wird lediglich eine zusätzliche Arbitrierungsleitung benötigt: die GS-Leitung. An diese Leitung werden nur die PE angeschlossen, da die ME den Zugriff auf die SE und den KB nicht extra anfordern müssen. Es wird also pro PE eine Arbitrierschaltung benötigt. Damit ergibt sich für die Fläche der Arbitierschaltungen in einem System mit On-Chip Synchronisation:

$$NG_{KB} = (P + M + \mathbf{P} + (P+1) \cdot M) \cdot (2NG_{Tr} + NG_{Bit} + 2) \quad (7.96)$$

Zusätzliche Fläche in der ME

Die ME benötigt zur Bewältigung der Synchronisation eine SB-Schnittstelle sowie eine Erweiterung der Kontrollogik. Letztere wird für die Handhabung des TL Zustands und der zugehörigen Übergänge benötigt.. Die Gatteranzahl der SB-Schnittstelle NG_{SB} kann durch Einsetzen von N_{SB} für die Anzahl der Gleichung 7.6 ermittelt werden. Die Anzahl zusätzlich der für die Kontrolllogik benötigten Gatter ergibt sich durch ein Erhöhen der Anzahl der Zustände und Übergänge in Gleichungen 7.21 bis 7.27 bei der Betrachtung der Komplexität der Kontrollogik der ME. Insgesamt gilt für die Gatterzahl der ME in einem System mit On-Chip Synchronisation:

$$\begin{aligned} NG_M \simeq\ & V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux} + (l_C + l_A + l_K) \cdot NG_{bit} \\ & + (N_{PK} \cdot V_F + 6 + \mathbf{N_{SB}}) \cdot NG_{Tr} + \mathbf{120} NG_{Bit} + l_k + \mathbf{600} \end{aligned} \quad (7.97)$$

Die Fläche der SE

Die Anzahl der Gatter der SE NG_{SE} ergibt sich als Summe der Anzahl der Gatter ihres Caches (NG_C), der SB-Schnittstelle NG_{SB}, der Kontrollogik (NG^S_{CTRL}) und des Zählers (NG_Z).

$$NG_{SE} = NG_C + NG_{SB} + NG_{CTRL} + NG_Z \quad (7.98)$$

Die Gatteranzahl eines Zählers kann aus der Literatur entnommen werden. Sie hängt von der Bitanzahl des maximalen Zählerwertes ab. Da wir von einem 64 Bit Adressraum ausgehen, liegt dieser bei 64 Bit.

Die Gatterzahl der Kontrollogik kann aus den Gleichungen und den Zustandsübergangsdiagrammen in Abbildung 7.24 ermittelt werden. Sie ergibt sich nach großzügiger Aufrundung auf ca. :

$$NG^{SE}_{CTRL} \simeq 1000 NG_{Bit} + 2000 \quad (7.99)$$

Damit gilt für die Gatterzahl der SE:

$$NG_{SE} \simeq 4P \cdot NG_{Ass} + N_{SB} \cdot NG_{Tr} + 1000 NG_{Bit} + 2000 + 500 \quad (7.100)$$

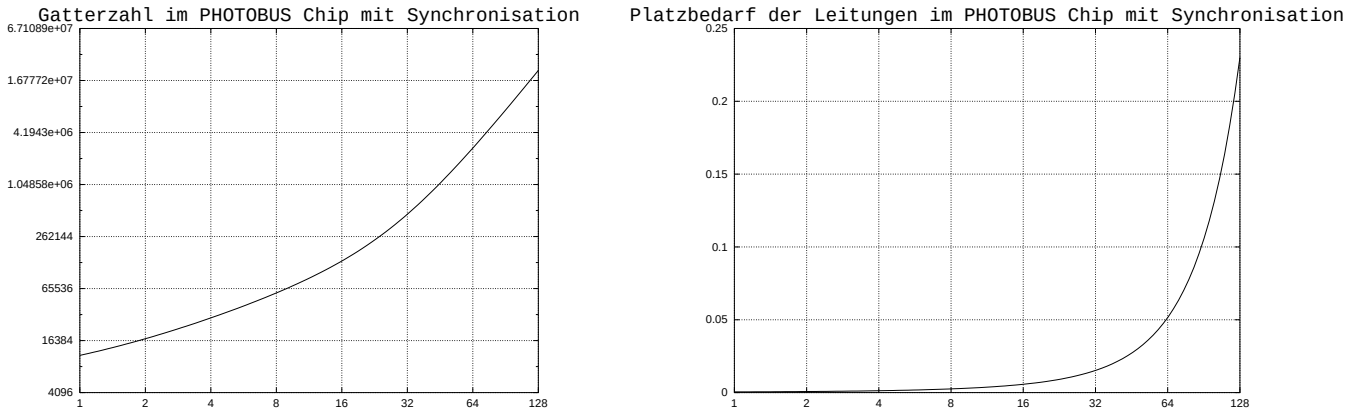


Abbildung 7.28: Das linke Diagramm zeigt die Abhängigkeit der auf dem Bus-Chip mit Synchronisation benötigten Gatter (vertikale Achse) von der Prozessorzahl (horizontale Achse). Das rechte Diagramm stellt den Anstieg der für die Verbindungsstruktur benötigten Fläche in cm^2 (vertikale Achse) mit der Prozessorzahl dar. Beide Diagramme verwenden eine logarithmische Darstellung.

Die Fläche der PESE

Die PESE besteht aus einem FIFO-Zwischenspeicher für ein Befehlswort und eine Adresse, einer SB-Schnittstelle und einer Kontrollogik. Die Gatterzahlen für das FIFO und die SB-Schnittstelle ergeben sich aus Gleichungen 7.7 und 7.55. Die Gatterzahl der Kontrollogik kann aus dem zugehörigen Zustandsübergangsdiagram und Gleichung (7.16) ermittelt werden. Für die Gatteranzahl der PESE (NG_{PS}) gilt dann:

$$NG_{PS} = N_{SB} \cdot NG_{Tr} + (l_k + l_a) \cdot NG_{Bit} + 200NG_{Bit} + 500 \tag{7.101}$$

Zusammenfassung

Die obige Betrachtung hat gezeigt, daß die Komplexität der Erweiterungen einer einzelnen PE und ME nicht von der Prozessor- oder Speicherbankzahl abhängen. Gleichzeitig beinhalten die Gleichungen für die Komplexität der SE und der Arbitrierungslogik nur Terme, die linear von der Prozessoranzahl abhängen bzw. konstant sind. Dies bedeutet, daß die Gesamtanzahl zusätzlich für die Synchronisation benötigter Gatter NG_{Sync} höchstens linear mit der Prozessorzahl steigt. Sie kann als Polynom ersten Grades in P und M geschrieben werden:

$$NG_{Sync} = P \cdot NG^P_{Sync} + M \cdot NG^M_{Sync} + NG^C_{Sync} \tag{7.102}$$

mit

$$NG^C_{Sync} = N_{SB} \cdot NG_{Tr} + 1000NG_{Bit} + 2000 + 500 \tag{7.103}$$

$$NG^P_{Sync} = (2NG_{Tr} + NG_{Bit} + 2) + N_{SB} \cdot NG_{Tr} + (l_k + l_a) \cdot NG_{Bit} + 200NG_{Bit} + 500 \tag{7.104}$$

$$NG^M_{Sync} = 20NG_{Bit} + 600 + N_{SB} \cdot NG_{Tr} \tag{7.105}$$

Obige Werte können benutzt werden, um die Abschätzung eines Zahlenwertes für die Gatteranzahl aus Gleichung 7.77 auf ein System mit On-Chip Synchronisation zu erweitern. Dazu werden lediglich die Werte für die Komplexität einer Speicherstelle des assoziativen Caches und für die Anzahl der Leitungen des SB benötigt. Ersteren schätzen wir der Literatur folgend (siehe z.B. [56]) auf 10 Gatter. Für die Anzahl der Leitungen nehmen wir $N_{SB} = 16$ an. Ein Einsetzen in die obigen Gleichungen, eine großzügige Aufrundung und Addition zu den korrespondierenden Termen aus Gleichung (7.77) liefert:

$$\boxed{NG_S \simeq 2P \cdot M \cdot ld(M) + 50P \cdot M \cdot + 4000P \cdot + 2000M \cdot + 5000} \tag{7.106}$$

Eine ähnliche Betrachtung kann man für die zusätzliche Leitungsfläche A_{Sync} anstellen. Wie Gleichung 7.95 zu entnehmen, müssen dabei M^2, P, $P \cdot M$ sowie konstante Terme berücksichtigt werden. Es gilt:

$$A_{Sync} = M^2 \cdot A^{M2}_{Sync} + P \cdot M \cdot A^{PM}_{Sync} + M \cdot + A^M_{Sync} + P \cdot A^P_{Sync} + A^C_{Sync} \tag{7.107}$$

und

$$A^{C}_{Sync} = 6N_{SB} \quad (7.108)$$
$$A^{P}_{Sync} = N_{SB} \cdot (N_{SB} + N_{PK} \cdot V_F + 7) \quad (7.109)$$
$$A^{M}_{Sync} = N_{SB} \cdot (N_{SB} + N_{PK} \cdot V_F + 3) \quad (7.110)$$
$$A^{PM}_{Sync} = 4N_{SB} \quad (7.111)$$
$$A^{M^2}_{Sync} = 3N_{SB} \quad (7.112)$$

Mit Hilfe obiger Gleichungen ergibt sich für die Abschätzung der Leitungsfläche auf der Basis von Gleichung :

$$\boxed{A_V\,[cm^2] \simeq (12P \cdot M^2 + 100M^2 + 1100P \cdot M \cdot + 42000P \cdot + 12000M + 50000) \cdot 6,25 \cdot 10^{-10}} \quad (7.113)$$

7.9.9 Leistungsbewertung

Die PHOTOBUS-Architektur mit On-Chip Synchronisation wurde, wie auch die ursprüngliche PHOTOBUS-Architektur durch Simulation bewertet. Dazu wird die bereits beschriebene Simulationsumgebung entsprechend erweitert. Sie SE wurde, wie alle anderen Einheiten des Bus-Chips als eigene Klasse modelliert, die über vordefinierte Felder mit den anderen Komponenten kommuniziert. Für die Arbitrierung wurde eine eigene Instanz der Arbiter-Klasse verwendet. Da die verwendeten Modelle die Synchronisation nicht berücksichtigen, wird für die Verifikation kein neues analytisches Modell benötigt. Die Verifizierung der Simulation erfolgt durch den Vergleich mit den in Abschnitt 7.19 beschriebenem theoretischen Modell an Hand der in Abbildung 7.20 dargestellten Ergebnisse.

Durch die Simulation werden zwei Aspekte untersucht. Zum einen wird die Abhängigkeit der Effizienz von der Bandbreite des B-Kanals betrachtet. Zum anderen wird die Auswirkung der Leitungslänge der P- und M-Kanäle auf die Effizienz überprüft, die für die Koppelung von Arbeitsplatzrechnern von Bedeutung ist.

Auswirkung der Bandbreite des B-Kanals

Die Betrachtung in 7.8 hat gezeigt, daß lediglich die Bandbreite des B-Kanals eine Rolle spielen. Im Nachfolgenden wird daher nur ein Wert für die Bandbreite dieser Kanäle, $B_P = B_M = 1$ betrachtet. Alle anderen Parameter sind mit denen in Abschnitt 7.8 identisch. Die Ergebnisse sind in den Abbildungen 7.29 und 7.30 zu sehen. Sie können wie folgt zusammengefaßt werden:

1. Mit Ausnahme des Radix-Benchmarks bei einer Bandbreite von $4Byte/cycle$ kann eine gute Effizienz bis zu 32 Prozessoren erreicht werden.

2. Außer beim Radix-Programm reicht eine Bandbreite von $8Byte/cycle$ für eine hohe Effizienz von bis zu 64 Prozessoren aus.

3. Bei einer Bandbreite von $16Bytes/cycle$ läßt sich für alle Programme eine sehr gute Effizienz bis zu 64 Prozessoren erreichen. Eine weitere Steigerung der Bandbreite auf $32/Bytes/cycle$ bleibt weitgehend ohne Wirkung.

4. Die Ergebnisse stimmen wesentlich besser mit der theoretischen Vorhersage überein, als die Werte für die PHOTOBUS-Architektur ohne integrierte Synchronisation. Da die Auswirkungen der Synchronisation zu den Hauptursachen der Abweichungen gehörten, war dies zu erwarten.

Auswirkung der Leitungslänge

Um die Auswirkung der Leitungslänge auf die Effizienz zu untersuchen, wurde bei der Simulation zu der Latenz der P-, M- und B-Kanäle eine entsprechender Betrag dazu addiert. Dabei wurden eine Zusatzlatenz von 16 und 32 Prozessorzyklen betrachtet. Bei einer Prozessorfrequenz von 500MHz entspricht dies 32 bis 64ns, also je nach Art der verwendeten Lichtleiterfaser einer Entfernung von 5 bis 10m (32ns), bzw. 15 bis 20m (64ns).

Die Ergebnisse für eine B-Kanal Bandbreite von 4 und 16 $Bytes/cycle/$ sind in den Abbildungen 7.31 und 7.32 zu sehen. Dabei wird deutlich, daß:

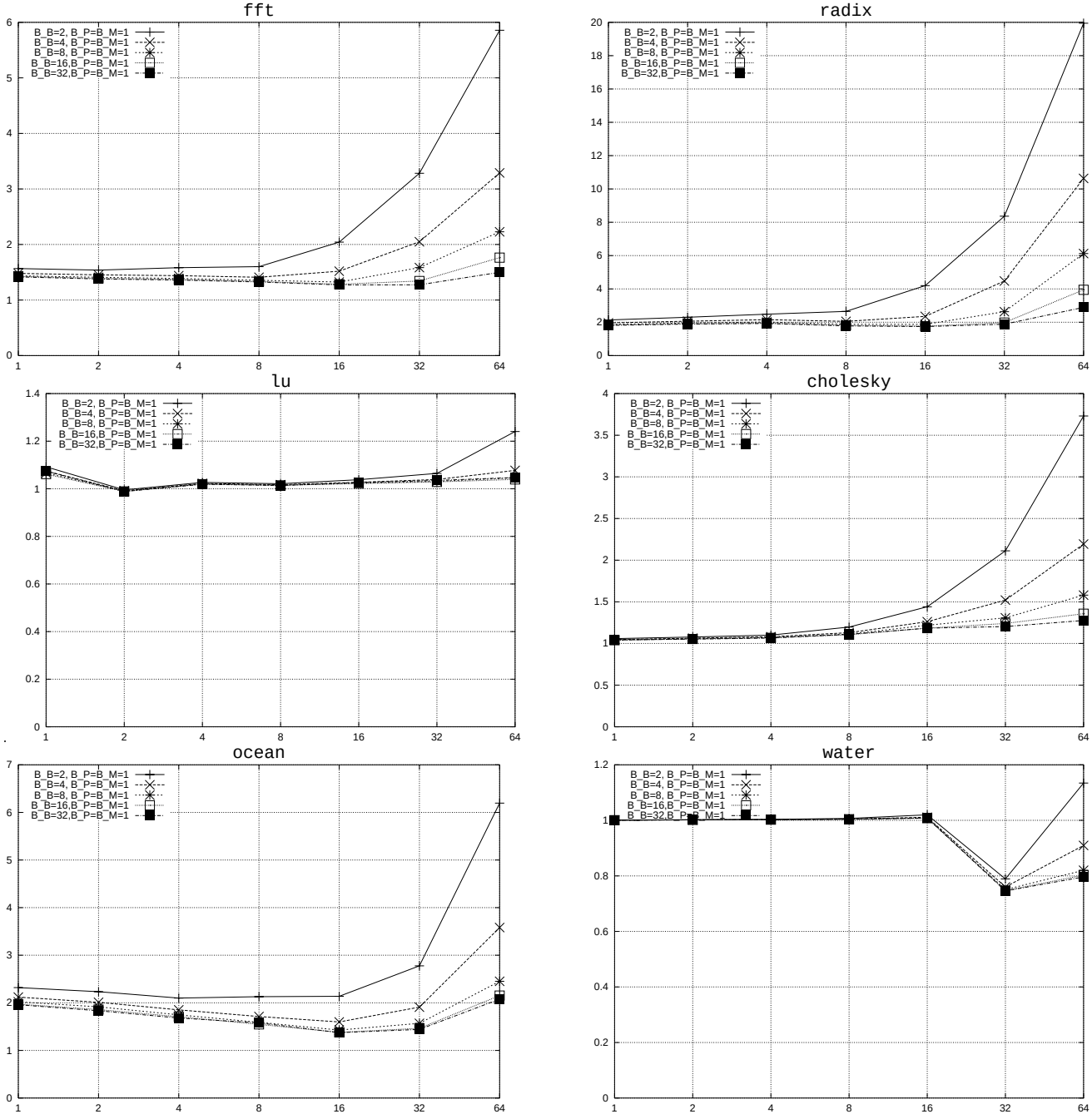


Abbildung 7.29: Die obigen Diagramme zeigen für die PHOTOBUS-Architektur mit On-Chip Synchronisation die durch Simulation ermittelte Ausführungszeit der einzelnen Benchmarks im Vergleich zur Ausführungszeit in einem idealen Speichersystem

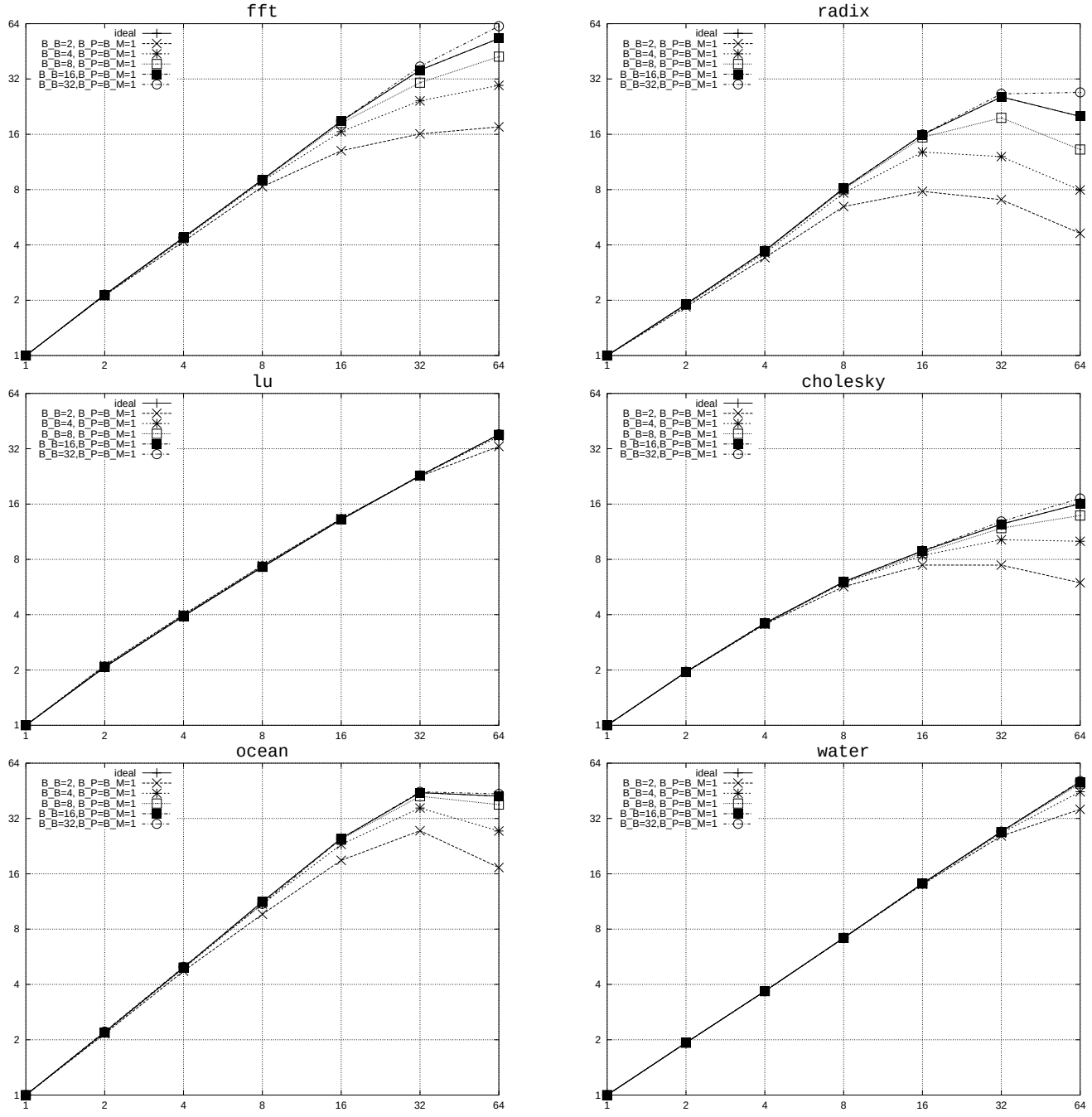


Abbildung 7.30: Die durch Simulation ermittelte Beschleunigung der einzelnen Benchmarkprogramme für die PHOTOBUS-Architektur mit on-Chip Synchronisation.

1. Für wenige Prozessoren die Erhöhung der Latenz eine Verschlechterung der Leistung von
   (a) bis zu 25 %, bei einer Latenz von 16 Zyklen,
   (b) bis zu 50 %, bei einer Latenz von 32 Zyklen,

   verursacht.
2. Bei kleinen Prozessorzahlen die Kurven mit gleicher Latenz unabhängig von der Bandbreite des B-Kanals ähnlich verlaufen.
3. Mit steigender Prozessorzahl die Latenz gegenüber der Bandbreite des B-Kanals an Bedeutung verliert. Ab ca. 8 Prozessoren fangen die Kurven mit gleicher Bandbreite an, einander anzunähern.
4. Bei 32 bis 64 Prozessoren liegen die Kurven gleicher Bandbreite in fast allen Fällen nah beieinander. Die Leistung der Systeme mit einer Latenz von 16 und 2 ist dann meistens identisch.

Insgesamt kann man sagen, daß die Vergrößerung der Latenz um 16 Zyklen grundsätzlich unproblematisch ist. Auch eine Latenz von 32 Zyklen kann in den meisten Fällen hingenommen werden, insbesondere bei höheren Prozessorzahlen.

Fazit

Durch die Simulation wurde gezeigt, daß die PHOTOBUS-Architektur mit On-Chip Synchronisation mit realistischen Werten für die Bandbreite des B-Kanal Transmitters die Realisierung eines effizienten SMP bis zu 64 Prozessoren erlaubt. Dies ist auch bei einer Leitungslänge von bis zu 10m zutreffend, so daß eine effiziente Koppelung von Arbeitsplatzrechnern zu SMP-Architekturen mit bis zu 64 Prozessoren durch die PHOTOBUS-Architektur möglich wird.

7.10 Fazit

Die im vorliegenden Kapitel vorgestellte PHOTOBUS-Architektur realisiert die Prozessor Speicher-Koppelung mit Hilfe eines auf einem SP-Chip integrierten paketvermittelnden Busses. In dem System schicken die Prozessoren ihre Daten zunächst über unidirektionale, direkt in den Chip einkekoppelte optische Faserkanäle an den Chip. Der Chip führt eine Arbitrierung durch und sendet die Daten über einen optischen Rundrufkanal zurück an alle Prozessoren und Speicherbänke.

In dem Kapitel wurde gezeigt, daß die PHOTOBUS-Architektur mit Hilfe sofort bzw. kurzfristig verfügbarer Technologie die Implementierung eines effizienten symmetrischen Multiprozessors mit bis zu 64 Prozessor erlaubt. Es wurde ebenfalls gezeigt, daß die optischen Kanäle eine Leitungslänge von bis zu 10m aufweisen können, ohne daß die Effizienz des Systems relevant verschlechtert wird. Damit ist eine effiziente Koppelung von Arbeitsplatzrechnern zu SMP-Architekturen möglich.

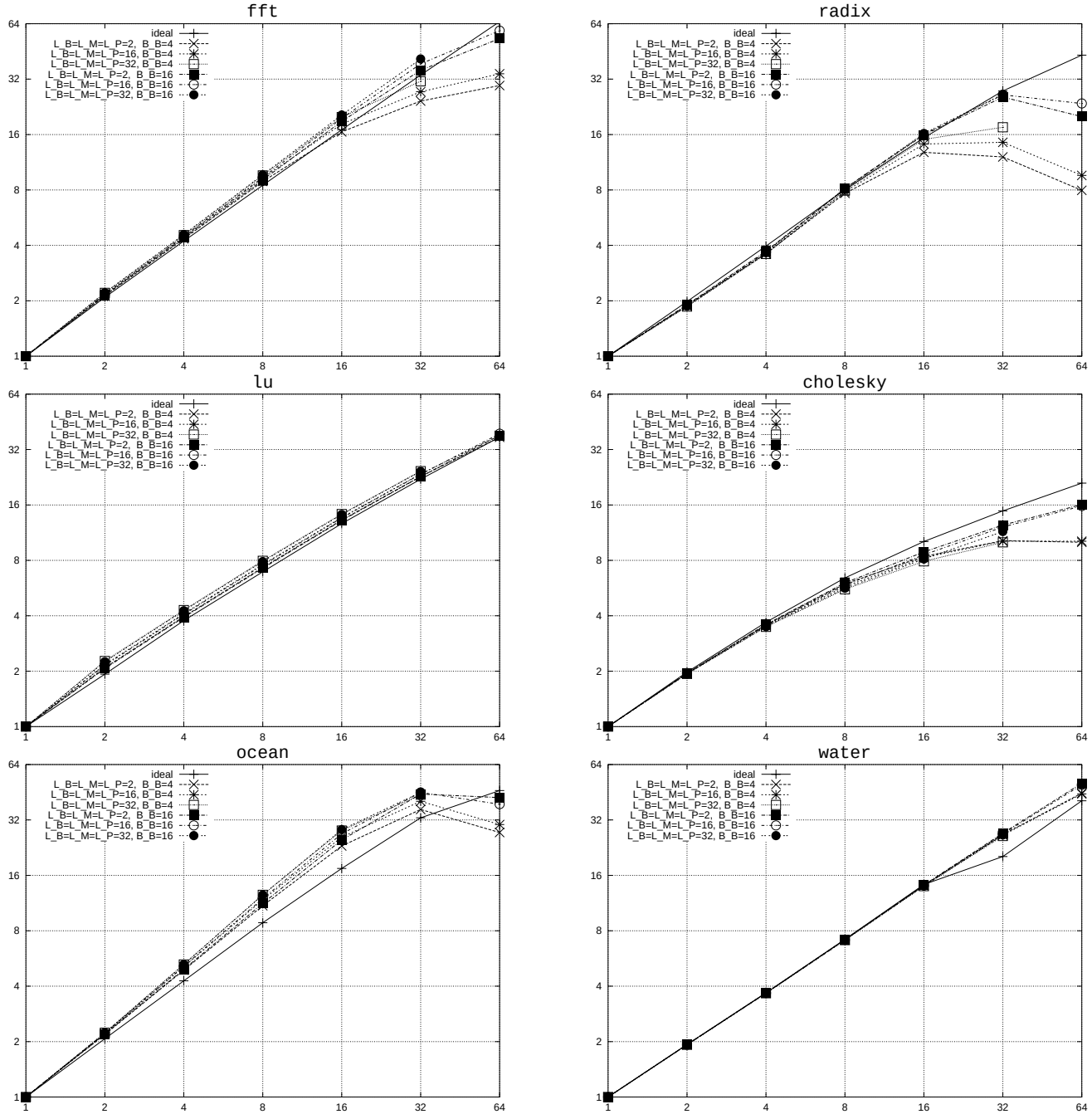


Abbildung 7.31: Der Einfluß der Latenz der optischen Kanäle auf die Ausführungszeit der Benchmarkprogramme für die PHOTOBUS-Architektur mit On-Chip Synchronisation.

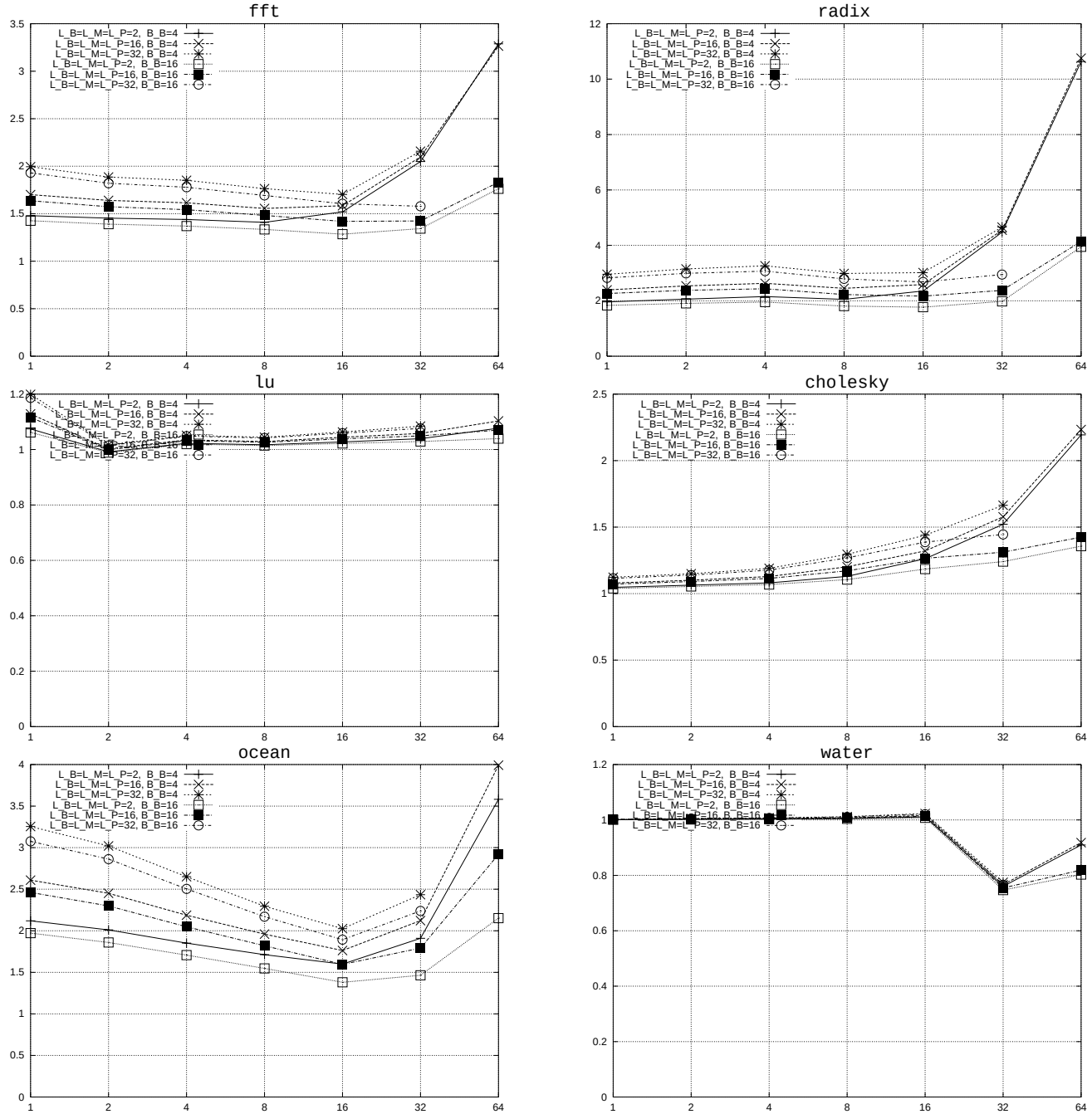


Abbildung 7.32: Der Einfluß der Latenz der optischen Kanäle auf die Beschleunigung der Benchmarkprogramme für die PHOTOBUS-Architektur mit on-Chip Synchronisation.

Kapitel 8

PHOTOBAR

Die Betrachtung des OE-VLSI-Busses hat gezeigt, daß sich mit steigender Prozessorzahl die Bandbreite des Broadcast-Kanals zum Flaschenhals des Systems entwickelt. Die hier vorgeschlagene PHOTOBAR-Architektur umgeht dieses Problem durch die Verwendung bidirektionaler Punkt-zu-Punkt Verbindungen zwischen dem SP-Chip und den Netzwerkknoten. Diese Verbindungen erlauben es, Daten vom SP-Chip unter Umgehung des B-Kanals direkt an einzelne Prozessoren und Speicherbänke zu schicken. Dabei wird der B-Kanal entlastet, so daß bei gleicher Bandbreite eine hohe Effizienz für mehr Prozessoren erreicht werden kann.

Das PHOTOBAR-System ähnelt in vieler Hinsicht dem PHOTOBUS-System. Der vorliegende Abschnitt beschränkt sich auf die Erläuterung der Unterschiede und vermeidet eine wiederholte Beschreibung gleicher Komponenten. Dabei liegen die Beiträge der Arbeit in drei Bereichen:

1. Es wird gezeigt, welche Voraussetzungen erfüllt werden müssen, damit das Berkeley- Protokoll korrekt und effizient realisiert werden kann (Abschnitt 8.1). Dabei geht es vor allem um die Fragen, welche Nachrichten über die Punkt-zu-Punkt Kanäle übertragen werden können, und wo die Gefahr einer Verklemmung besteht.

2. Für alle Komponenten, die sich von den der PHOTOBUS-Architektur unterscheiden, wird die Hardwarearchitektur beschrieben. Dazu gehören die optischen Speicher- und Prozessorschnittstellen (Abschnitte 8.2) so wie die Verbindungsstruktur und die PEs und die MEs des SP-Chips (Abschnitt 8.3). Der Aufbau der Komponenten wird wie im Kapitel 7 durch Strukturdiagramme, ihre Funktionalität durch endliche Automaten spezifiziert.

3. Die Skalierbarkeit und Leistungsfähigkeit der PHOTOBAR-Architektur abgeschätzt. Dazu gehören eine Abschätzung der benötigten Chipfläche in Abhängigkeit von der Anzahl der Prozessoren (Abschnitt 8.4) so wie die Simulation und einfache theoretische Modellierung des Systems (Abschnitt 8.5). Die Vorgehensweise ist dabei die gleiche wie bei der Bewertung der PHOTOBUS-Architektur in Kapitel 7.

8.1 Überblick

Der PHOTOBAR-Architektur liegen die gleichen Überlegungen zugrunde, die zu der Entwicklung gemischter Netzwerke bei kommerziellen SMPs geführt haben. Zum einen muß zur Erhaltung der Cache-Kohärenz nur ein Teil der Kommunikation als Rundruf an alle Prozessoren geschickt werden. Dabei handelt es sich vor allem um kurze, aus einem Befehlswort und einer Adresse bestehende Nachrichten. Die übrigen Nachrichten können über Punkt-zu-Punkt Verbindungen gezielt an einzelne Prozessoren verschickt werden. Zum anderen ist die Realisierung zusätzlicher Punkt-zu-Punkt Verbindungen mit geringer Bandbreite wesentlich einfacher als die Vergrößerung der Bandbreite der Broadcast Verbindung. Dabei werden auch die Schnittstellen der Prozessoren entlastet, so daß auch das Schnittstellenproblem weniger stark zum Tragen kommt.
Die hier vorgeschlagen Realisierung der PHOTOBAR-Architektur basiert auf einer Erweiterung der im vorigen Kapitel beschriebenen PHOTOBUS-Konzepte. Bei einer solchen Erweiterung müssen drei Aspekte berücksichtigt werden:

1. Es muß für die einzelnen Speicheroperationen bestimmt werden, welche Nachrichten über welchen Kanäle verschickt werden müssen, damit eine korrekte und effiziente Implementierung des Berkeley-Protokolls gewährleistet ist.

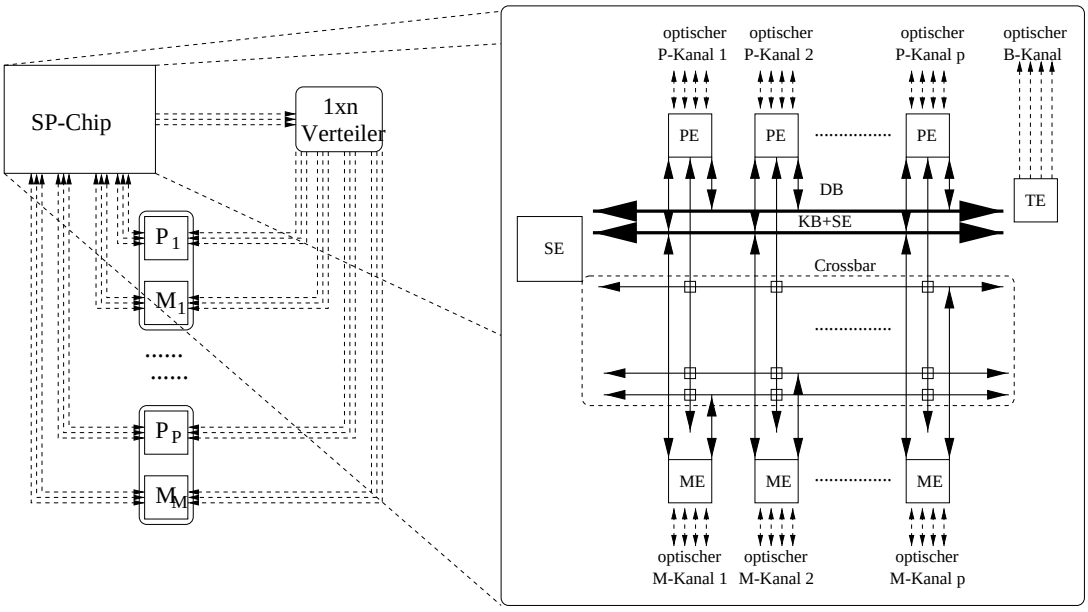


Abbildung 8.1: Das Prinzip der PHOTOBAR-Architektur und der Aufbau des zugehörigen SP-Chips.

2. Es muß sichergestellt werden, daß durch die zusätzlichen Punkt-zu-Punkt Verbindungen keine Probleme mit der Flußkontrolle entstehen. Dabei sind vor allem zwei Dinge von Bedeutung;

   (a) In der PHOTOBAR-Architektur empfangen alle Prozessoren und Speicher Daten auf zwei Kanälen: dem B-Kanal und dem P-, bzw. M-Kanal. Dies macht einen Kontrollmechanismus erforderlich, der bei gleichzeitigem Eintreffen von Nachrichten auf beiden Kanäle ihre Bearbeitung koordiniert. Dieser Mechanismus muß die Tatsache berücksichtigen, daß unterschiedliche Arten von Nachrichten unterschiedliche Prioritäten haben.

   (b) Im PHOTOBUS-System konnten die Prozessoren durch Mithören am B-Kanal mitkriegen, wann ihre Anfragen bearbeitet wurden, und der SP-Chip für weitere Anfragen frei wurde. Da in der PHOTOBAR-Architektur nicht jede Nachricht für alle sichtbar auf dem B-Kanal übertragen wird, muß das System durch gezielt verschickte Nachrichten sicherstellen, daß die Prozessoren benachrichtigt werden, sobald der SP-Chip ihre Anfrage bearbeitet hat.

3. Damit das System die Vorteile der zusätzlichen Kanäle nutzen kann, benötigt der SP-Chip ein zusätzliches On-Chip Netzwerk. Das Netzwerk muß die Datenübertragung zwischen den Ein-/ und Ausgängen der P- und M- Kanäle sicherstellen und miteinander verbinden. Ohne ein solches Netzwerk würde der On-Chip Bus gleichermaßen zum Flaschenhals des Systems werden, wie beim OE-VLSI-Bus System der B-Kanal.

Im Nachfolgenden wird ein Überblick über die Ansätze gegeben, die bei einem Entwurf der PHOTOBAR-Architektur in Bezug auf die obigen Aspekte verfolgt wurden.

8.1.1 Ablauf der Transaktionen

Der Ablauf der einzelnen Speicheroperationen auf dem OE-VLSI-Schalter ist in Abbildung 8.2 und 8.3 gezeigt. Wie ein Vergleich mit Abbildung 7.2 zeigt, besteht ein Unterschied zum OE-VLSI-Bus nur bei der Behandlung der Antworten auf Fetch-Operationen, der WriteBack-Operationen und der Synchronisationsoperationen.

Fetch-Operationen

Bei einer Fetch-Operation schickt der Prozessor die Adresse und das Befehlswort über seinen P-Kanal an den Chip. Dieser wartet, bis die benötigte Speicherbank frei ist und leitet die Anfrage über den B-Kanal weiter. Sie wird dann von der betroffenen Speicherbank und gegebenenfalls von dem Besitzer der Cache-Zeile übernommen und bearbeitet. Wenn die Bearbeitung abgeschlossen ist, wird die Antwort über den M- bzw. P-Kanal zurück

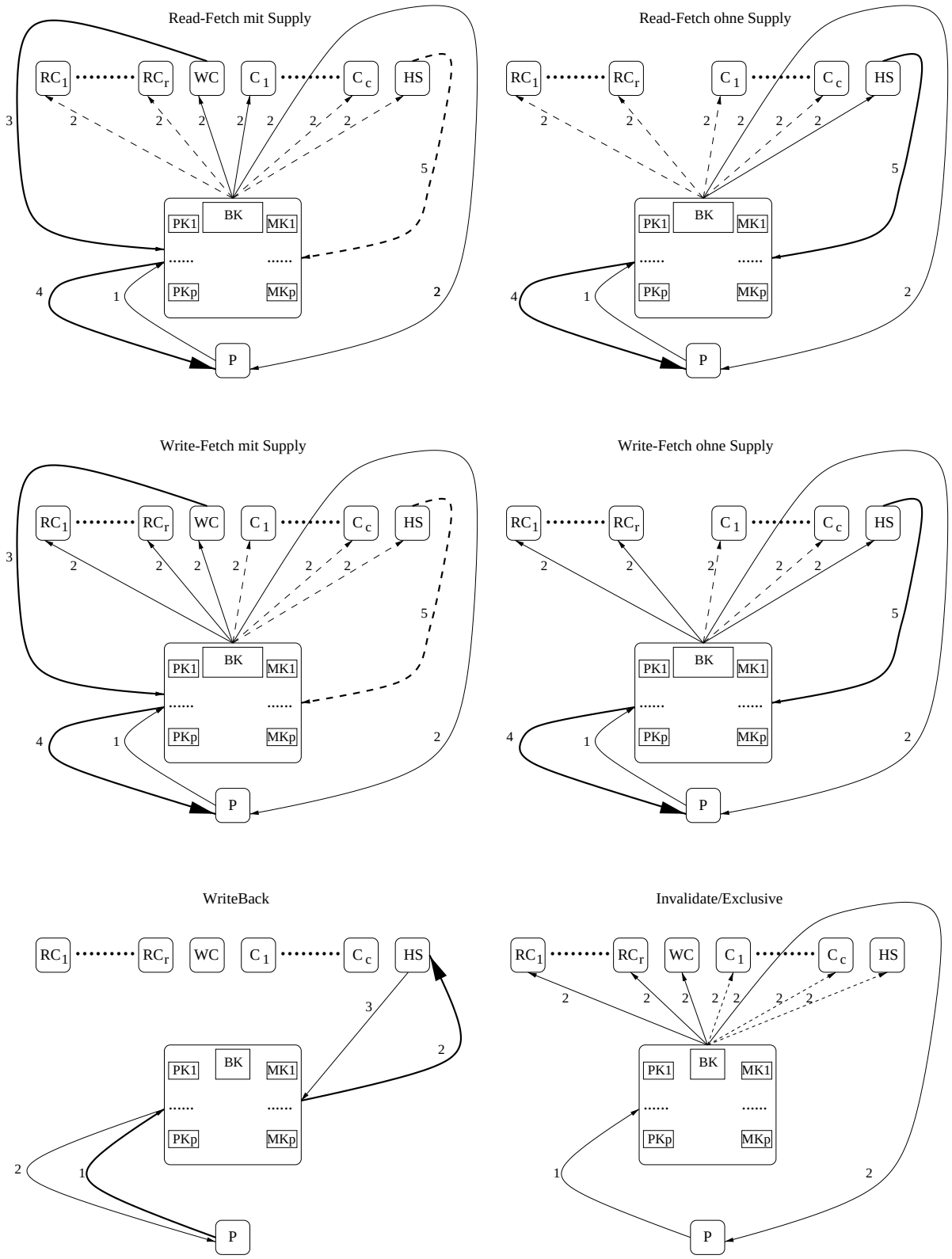


Abbildung 8.2: Der Ablauf der Speicheroperationen des Berkeley-Protokolls in der PHOTOBAR-Architektur. Die gestrichelte Linien bedeuten Kommunikationsschritte, die nicht unbedingt notwendig sind. Durch dicke und dünne Linien wird zwischen Nachrichten unterschieden, die aus einer ganzen Cache-Zeile oder nur aus einem Befehl und einer Adresse bestehen.

Abbildung 8.3: Der Ablauf der Synchronisationsoperationen des in der PHOTOBAR-Architektur.

an den Chip geschickt. Bis zu diesem Zeitpunkt läuft die Operationen genauso wie beim OE-VLSI Bus ab. Der Unterschied besteht darin, daß die Daten nun nicht über den B-Kanal, sondern über den P-Kanal zurück an den Urheber der Anfrage geschickt werden.

WriteBack

Eine WriteBack-Operationen betrifft nur die Speicherbank, zu der die Daten zurückgeschrieben werden. Ein Broadcast ist also nicht nötig. Der Prozessor schickt die betroffene Cache-Zeile über seinen P-Kanal zum OE-VLSI-Chip. Dieser wartet, bis die benötigte Speicherbank frei ist, und gibt die Anfrage über den M-Kanal an sie weiter. Gleichzeitig teilt er dem Prozessor über den P-Kanal mit, daß die Operation beendet ist. Sobald die Speicherbank mit der Schreiboperation fertig ist, schickt sie über ihren M-Kanal eine Bestätigung an den Bus-Chip, der die Bank wieder für andere Anfragen freigibt.

Synchronisation

Unter der Annahme, daß die Synchronisation auf dem OE-VLSI-Chip abgewickelt wird, benötigt sie gar keine Broadcast-Operationen. Die Vorgehensweise ist dabei mit der beim PHOTOBUS bis auf die Tatsache identisch, daß die Rückmeldung an den Prozessor nicht über den B-, sondern über den entsprechenden P-Kanal geschickt wird. Falls die Synchronisation über den Speicher abgewickelt werden muß, so wird die Speicheranfrage über den M- anstelle des B-Kanals verschickt. Damit wird für die Synchronisation der B-Kanal überhaupt nicht mehr benötigt.

8.1.2 Flußkontrolle

Wie am Anfang des Abschnittes 8.1 angesprochen, müssen bei der Flußkontrolle vor allem zwei Aspekte berücksichtigt werden: die Weiterleitung von Fertig-Meldungen des SP-Chips zur Prozessorschnittstelle und die Koordinierung des Datenflusses der beiden Eingänge an den Prozessor und die Speicherschnittstellen.

Rückmeldungen

Bei WriteBack-, Supply-Operationen werden die Daten vom SP-Chip direkt über einen M- bzw. P-Kanal an den Empfänger geleitet. Eine Übertragung über den B-Kanal findet also nicht statt. Dies bedeutet, daß der Urheber der Operation nicht automatisch über ihre Durchführung informiert wird. Er weiß daher nicht, wenn der SP-Chip bereit ist, weitere Anfragen zu empfangen. Aus diesem Grund muß der SP-Chip nach jeder WriteBack- oder Supply-Operation eine Fertig-Meldung an den Urheber-Prozessor über den zugehörigen P-Kanal schicken.

Koordinierung des Datenflusses

Die Handhabung der beiden Eingabekanäle ist bei der Prozessor- und Speicherschnittstelle unterschiedlich. Bei der Speicherschnittstelle wird sie dadurch vereinfacht, daß der Zugang zu den Speicherbänken auf dem SP arbitriert wird. Dabei wird immer nur eine Anfrage an eine bestimmte Speicherbank geschickt. Eine Speicherbank muß also niemals gleichzeitig an sie gerichtete Anfragen auf dem M- und P-Kanal erwarten. Sie braucht sich also um die Koordinierung der beiden Datenströme gar nicht zu kümmern.

Bei der Prozessorschnittstelle ist es hingegen durchaus möglich, daß gleichzeitig auf beiden Kanälen relevante Daten übertragen werden. Dies liegt daran, daß die beiden Eingabekanäle unterschiedliche Funktionen erfüllen. Über den P-Kanal erreichen den Prozessor Antworten auf Anfragen und Fertig-Meldungen. Auf dem B-Kanal kommen dagegen Supply-Anforderungen und Invalidate- bzw. Exclusiv-Operationen. So ist es möglich, daß während der Ankunft von einer Antwort auf dem P-Kanal gleichzeitig eine Invalidate-Operation auf dem B-Kanal übertragen wird.

Bei der Bearbeitung gleichzeitig ankommender Daten muß man bei der Prozessorschnittstelle berücksichtigen, daß sie unterschiedliche Prioritäten haben. Die Anfrage auf dem B-Kanal müssen grundsätzlich sofort bearbeitet werden. Dies liegt zum einen daran, daß insbesondere für Supply-Anfragen strenge Timing-Anforderungen erfüllt werden müssen. Zum anderen könnte eine Verzögerung zu Problemen mit Bufferüberläufen führen. Dies liegt daran, daß die Daten auf dem B-Kanal mit einer hohen Datenraten gesendet werden und es zwischen den einzelnen Prozessoren und dem OE-VLSI-Chip keine Flußkontrolle gibt. Der SP-Chip sendet auf dem B-Kanal, ohne zu überprüfen, ob alle Prozessoren die vorherigen Nachrichten verarbeitet haben. Auf dem P-Kanal hingegen werden neue Daten erst dann an den Prozessor geschickt, wenn dieser eine neue Anfrage gestellt hat.

Die Schwierigkeit der Bearbeitung gleichzeitig ankommender Daten besteht darin, daß sowohl beim Empfang von Antworten auf dem P-Kanal als auch bei der Verarbeitung der über den B-Kanal ankommenden Anfragen ein Zugriff auf den Cache notwendig ist. Bei Antworten auf Anfragen müssen die Daten im Cache abgelegt werden. Bei Invalidate-, Exclusiv- und Fetch-Operationen auf dem B-Kanal muß der Prozessor prüfen, ob das betreffende Datum in seinem Cache vorhanden ist. Trifft also eine Nachricht auf dem B-Kanal an, während auf dem P-Kanal eine Antwort empfangen wird, dann kommt es zu einem Konflikt beim Cache-Zugriff. Damit eine sofortige Bearbeitung der Daten vom B-Kanal möglich ist, muß entweder die Übertragung der Antwort in den Cache unterbrochen werden, oder der Cache muß über zwei Zugriffsports verfügen. Bei der nachfolgenden Beschreibung der PHOTOBAR-Architektur gehen wir von der zweiten Möglichkeit aus. Dies vereinfacht die Betrachtung, ohne daß es sich dabei um eine unrealistische Annahme handelt.

8.1.3 On-Chip Netzwerk

Mit Ausnahme von Synchronisationsoperationen und Supply-Benachrichtigungen wurden im PHOTOBUS-System Daten auf dem SP-Chip ediglich zwischen den PE bzw. MEs und dem B-Kanal Transmitter übertragen. In der PHOTOBAR-Architektur muß der SP-Chip hingegen eine direkte Datenübertragung zwischen einzelnen PEs und MEs ermöglichen. Dies ist notwendig, damit Daten, die über einen P- bzw. M-Kanal im Chip in einer PE bzw. ME ankommen, diesen über den P- bzw. M-Kanal einer anderen PE bzw. ME verlassen können. Für die Realisierung der Speicheroperationen nach dem im Abschnitt 8.1.1 beschrieben Prinzip werden im einzelnen die folgenden Kommunikationsoperationen benötigt:

1. Für WriteBack-Anfragen und über den Speicher ablaufende Synchronisationsoperationen muß eine PE eine Nachrichten an eine ME schicken können.
2. Für vom Speicher kommende Antworten auf Fetch-Operationen muß eine ME Daten an eine PE schicken können.
3. Bei Supply-Operationen muß eine PE über das Crossbar-Netzwerk Daten an eine andere PE versenden können.

Damit das System die Vorteile der bidirektionalen P- und M-Kanäle nutzen kann, muß es die parallele Durchführung mehrerer solcher Operationen zulassen. Dies bedeutet, daß der SP-Chip neben dem, an den B-Kanal angeschlossenen Datenbus ein weiteres On-Chip Netzwerk benötigt. Ohne ein solches Netzwerk würde der On-Chip Bus gleichermaßen zum Flaschenhals des Systems werden, wie beim PHOTOBUS-System der B-Kanal. Er würde eine Sequentialisierung aller Kommunikationsoperation erzwingen und den Vorteil der bidirektionalen P- und M-Kanäle damit zunichte machen.

Die Anforderungen an das zusätzliche On-Chip Netzwerk können wie folgt zusammengefaßt werden:

1. Das Netzwerk sollte nicht-blockierend sein, um die parallele Verarbeitung mehrerer Anfragen zu ermöglichen.
2. Es sollte eine möglichst geringe Latenz, vorzugsweise im Bereich von 1 bis 2 Prozessorzyklen besitzen.
3. Es muß biderektional sein, um sowohl den Datenfluß von den PEs zu den MEs als auch umgekehrt zu erlauben.

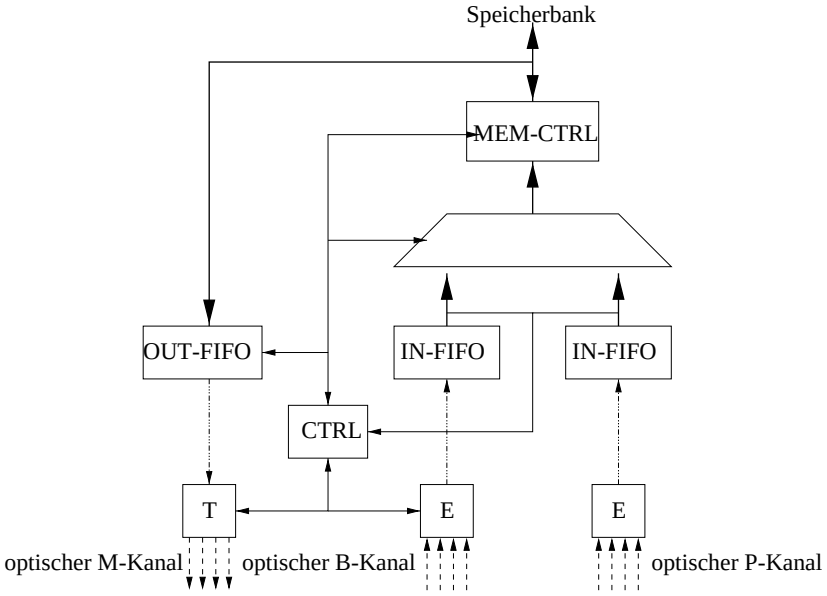


Abbildung 8.4: Aufbau der Speicherschnittstelle in der PHOTOBAR-Architektur.

4. Zusätzlich zur Verbindung zwischen den PEs und den MEs muß es den Datenaustausch zwischen einzelnen PEs zulassen.

5. Obige Eigenschaften müssen mit möglichst wenig Hardwareaufwand realisiert werden, damit möglichst viele PEs und MEs auf dem Chip untergebracht werden können.

Um diese Anforderungen zu erfüllen, wurde als Netzwerk für die PHOTOBAR-Architektur ein leitungsvermittelndes, bidirektionales Crossbar ausgewählt. Das Prinzip eines solchen Netzwerks wurde in Abschnitt 3.4.4 erläutert. Es verbindet zwei Gruppen von Knoten, indem sie jedem Knoten einen eigenen Datenkanal zuordnet. Dabei ist der Datenkanal jedes Knotens der ersten Gruppe über einen Schalter an die Datenkanäle aller Knoten der zweiten Gruppe angeschlossen.

In der PHOTOBAR-Architektur besteht die eine Knotengruppe aus den PEs, während die MEs die zweite Gruppe bilden. Die Wahl einer solchen Crossbar-Architektur bedeutet, daß das Netzwerk eine nicht-blockierende Verbindung zwischen beliebigen PEs und MEs erlaubt. Da zur Herstellung einer Verbindung lediglich die Aktivierung eines Schalters notwendig ist, hat es eine sehr geringe Latenz. Unter der Voraussetzung, daß die Schalter bidirektional sind, können sowohl Daten von der PE zur ME, als auch umgekehrt übertragen werden. Auch die Datenübertragung zwischen zwei PEs kann in einer solchen Anordnung realisiert werden. Dazu müssen die beiden PEs gleichzeitig mit den Leitungen der gleichen ME verbunden sein.

Ein großer Nachteil eines Crossbar-Netzwerks besteht darin, daß die Schalteranzahl proportional zum Produkt der Anzahl der Knoten in den beiden Gruppen ist. Aus diesen Grund werden solche Netzwerke nicht für große Parallelrechner benutzt. In der PHOTOBAR-Architektur spielt der starke Anstieg der Schalterzahl jedoch nur eine geringe Rolle. Dies liegt daran, daß in einer VLSI-Implementierung ein Schalter mit wenigen Transistoren pro Leitung implementiert werden kann. Bei ca. 10^7 Transistoren, die auf modernen VLSI-Bausteinen untergebracht werden können, fallen die Crossbar-Schalter selbst bei hohen Prozessorzahlen daher nicht ins Gewicht.

8.2 Architektur der Schnittstellen

Die Prozessor- und Speicherschnittstellen unterscheiden sich von den Schnittstellen des einfachen Busses darin, daß sie jeweils zwei optische Eingänge besitzen. Wie in Abschnitt 8.1.2 erläutert, werden weder an der Speicher- noch an der Prozessorschnittstellen besondere Vorkehrungen zur Koordinierung des Datenflusses an den beiden Schnittstellen notwendig. Dies bedeutet, daß die Erweiterungen lediglich aus entsprechenden Empfängerschaltkreisen und Buffern bestehen. Sie sehen im einzelnen wie folgt aus:

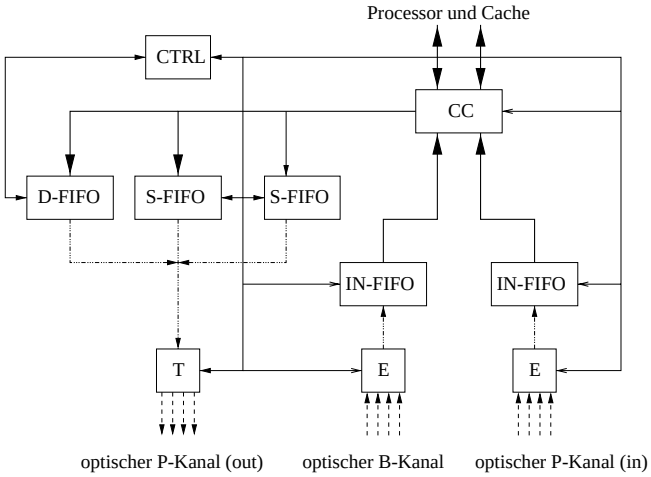


Abbildung 8.5: Die Prozessorschnittstelle der PHOTOBAR-Architektur.

8.2.1 Speicherschnittstelle

Die Architektur der Speicherschnittstelle ist in Abbildung 8.4 zu sehen. Dabei sind die Empfänger der beiden Kanäle über jeweils einen FIFO-Zwischenspeicher und einen Multiplexer an den Speicherkontroller angeschlossen. Die Kontrolleinheit hört an dem Datenstrom der beiden Kanäle mit und wartet auf Anfragen, die für die eigene Speicherbank bestimmt sind. Sobald eine solche Anfrage kommt, schaltet diese den Multiplexer so um, daß die Daten des entsprechenden Kanals an den Speicherkontroller geleitet werden. Die Funktionsweise der Speicherschnittstelle ist damit mit der der PHOTOBUS-Architektur so gut wie identisch. Sie bedarf daher keiner weiteren Erläuterung.

8.2.2 Prozessorschnittstelle

Die Prozessorschnittstelle des PHOTOBAR-Systems unterscheidet sich von der im vorigen Kapitel beschriebenen Schnittstelle der PHOTOBUS-Architektur nur in einem Punkt: Sie besitzt einen zusätzlichen Empfänger mit angeschlossenem FIFO-Zwischenspeicher. Dies ist in Abbildung 8.5 zu sehen. Bei der Beschreibung der Funktionsweise der Schnittstelle muß man berücksichtigen, daß die beiden Empfänger parallel arbeiten. Dies bedeutet, daß anstelle des einen Empfangsprozesses der PHOTOBUS-Architektur nun zwei betrachtet werden müssen. Bei der Spezifikation der Funktionsweise der Schnittstelle mit Hilfe von endlichen Automaten werden daher zwei Übergangsdiagramme benötigt. Sie sind in Abbildung 8.6 zusammen mit dem, gegenüber der PHOTOBUS-Architektur unveränderten Diagramm für den Empfangsprozeß dargestellt. Wie in der Abbildung zu sehen, ist das Diagram des Empfangsprozesses des B-Kanals des PHOTOBUS bis auf das Fehlen der Kohärenz-Operationen identisch. Das Diagramm des Empfangsprozesses des B-Kanals beinhaltet dagegen nur die Kohärenz-Operationen sowie den WA-Zustand für das Warten auf Antworten vom Bus-Chip. Er stellt einen Ausschnitt aus dem Originaldiagramm dar. Da die Zustände und die Übergänge die gleichen Bedeutung wie bei der PHOTOBUS-Architektur besitzen, müssen sie hier nicht nochmals erläutert werden. Eine Beschreibung ist in 7.4 zu finden.

8.3 Chiparchitektur

Die Aufgabe des SP-Chips in der PHOTOBAR-Architektur wurde bereits im vorangegangenen Abschnitt deutlich. Der Unterschied zur PHOTOBUS-Architektur besteht darin, daß nicht alle Nachrichten sequentiell über den B-Kanal verschickt werden. Statt dessen werden WriteBack Anfragen, Antworten auf Nachfragen und Synchronisationsoperationen über die bidirektionalen P- und M-Kanäle direkt an die Empfänger verschickt. Die Architektur des PHOTOBAR-Chips, die hierzu notwendig ist, ist in Abbildung 8.1 skizziert. Sie besitzt zusätzlich zu den von der PHOTOBUS-Architektur bekannten Grundkomponenten (PEs, MEs, TE, SE und den drei Busse n DB, KB und SB) ein bidirektionales, leitungsvermittelndes Crossbar-Netzwerk, das die PEs mit den MEs verbindet. Der Aufbau und die Funktionalität der TE, der SE und der drei Busse sind mit der PHOTOBUS-Architektur identisch. Sie werden hier daher nicht weiter betrachtet. Die PEs und MEs unterscheiden sich von der Beschreibung

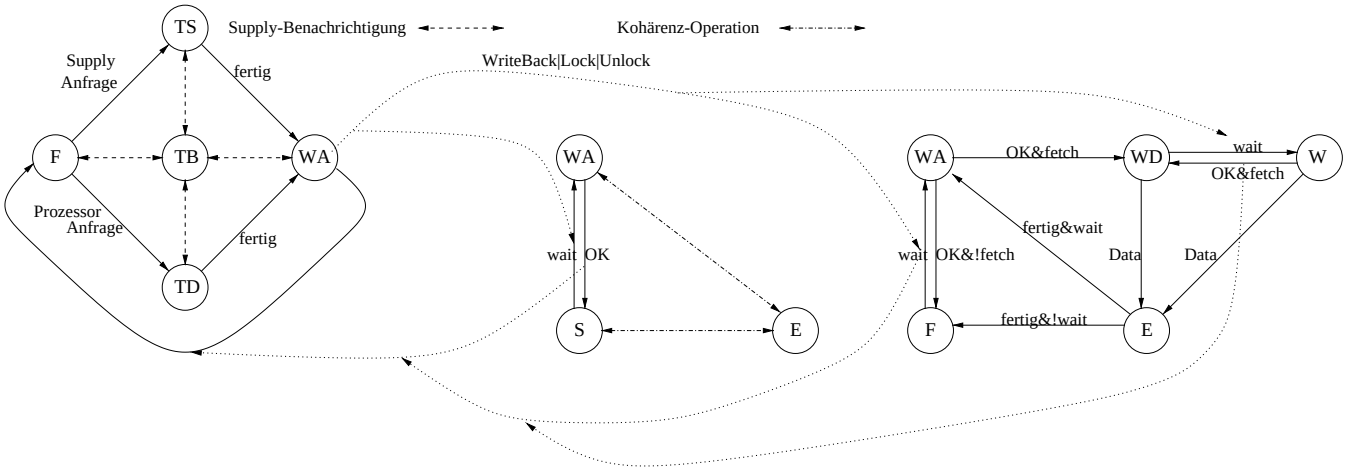


Abbildung 8.6: Das Zustandsübergangsdiagramm der Prozesse, die die Funktion der Prozessorschnittstelle in der PHOTOBAR-Architektur beschreiben. Die gestrichelten Linien zeigen die Interaktion der Prozesse an.

in Kapitel 7 dadurch, daß

1. sie nicht nur optische Empfänger sondern auch Transmitter besitzen.

2. sie an das Crossbar-Netzwerk angeschlossen sind.

3. die ME nicht an den DB angeschlossen sind.

Der Ablauf der Operationen auf dem Chip ist weitgehend durch die Beschreibung des Ablaufs der Speicheroperationen im PHOTOBAR-System (Abschnitt 8.1.1) gegeben. Alle Nachrichten, die über den B-Kanal zu verschicken sind, müssen über den DB an die TE gehen. Nachrichten, die direkt über die P- und M-Kanäle versendet werden, müssen auf dem Chip durch das Crossbar-Netzwerk zwischen den PEs und MEs transportiert werden. Das System muß außerdem sicherstellen, daß, wie in 8.1.1 beschrieben, eine PE Supply-Operationen durchführen kann, während sie auf das Ergebnis ihrer Anfrage wartet. Dies ist wie bereits erläutert für die Verklemmungsfreiheit notwendig. Das Problem besteht dabei daran, daß sowohl das Ergebnis als auch die Supply-Operation über das Crossbar-Netzwerk durchgeführt werden. Dabei kann es zu Zugriffskonflikten kommen, die das Crossbar-Zugriffsprotokoll auflösen muß.

Insgesamt unterscheidet sich der PHOTOBAR-Chip vor allem dadurch, daß er eine Vielzahl parallel ablaufender Aktivitäten abwickeln muß. Bei der Abwicklung dieser Aktivitäten kommt es zu Zugriffskonflikten, insbesondere bei Zugriffen auf die Crossbar-Kanäle. Diese Konflikte müssen effizient und verklemmungsfrei gelöst werden. Dies ist Aufgabe des Crossbar-Zugriffsprotokolls. Dieses Protokoll wird im Nachfolgenden als erstes beschrieben. Danach wird der Aufbau der Crossbar-Hardware erläutert. Abschließend werden die Architektur und die Arbeitsweise der MEs und PEs beschrieben.

8.3.1 Crossbar-Protokoll

Beim Entwurf des Zugriffsprotokolls für das Crossbar-Netzwerk müssen zwei Dinge bedacht werden:

1. Mit dem Zugriff einer PE auf eine ME ist immer die Nutzung des zugehörigen Crossbar-Kanals verbunden. Dies bedeutet, daß die Vergabe des Zugangs zu den Crossbar-Kanälen der MEs mit der Arbitrierung der Speicherbänke koordiniert werden muß. Anderenfalls könnte es zu Verklemmungen kommen.

2. Die Crossbar-Kanäle werden für verschiedene Aufgaben benötigt, die zum Teil parallel ablaufen. Dabei ist vor allem die Tatsache von Bedeutung, daß eine PE, die auf das Ergebnis einer Speicheranfrage wartet, bei Bedarf Supply-Operationen durchführen muß. Da sowohl die Supply-Operation als auch das erwartete Ergebnis über den Crossbar-Kanal der betroffenen PE-verschickt werden, kann es hierdurch zu Zugriffskonflikten kommen.

Im Nachfolgenden wird zunächst die Problematik der Zugriffskonflikte genauer erörtert.. Danach werden die Ideen erläutert, die dem im Rahmen der Arbeit konzipierten Zugriffsprotokoll zugrunde liegen. Abschließend wird der Ablauf der einzelnen Speicheroperationen beschrieben.

Zugriffskonflikte

Wie in Abschnitt 8.1.3 beschrieben, gehen wir davon aus, daß das Crossbar-Netzwerk leitungsvermittelnd und bidirektional ist. Dies bedeutet, daß für die Übertragung einer Nachricht eine direkte Verbindung zwischen dem Sender und dem Empfänger aufgebaut wird. Diese Verbindung kann in beide Richtungen betrieben werden, wobei allerdings jeweils nur eine Instanz senden und eine empfangen muß. Neben einer einfachen Verbindung zwischen einem Sender und einem Empfänger wird außerdem die Möglichkeit benötigt, zwei PEs mit einer und derselben ME zu verbinden. Dadurch wird eine Datenübertragung zwischen den PEs möglich, die zur Durchführung einer Supply-Operation notwendig ist. Auch hier gilt, daß die Verbindung zwar keine Vorzugsrichtung besitzt, aber keine gleichzeitige Datenübertragung in beide Richtungen zuläßt. .
Zugriffskonflikte können sich bei obigen Konstellation in zwei Fällen ergeben:

1. Wenn gleichzeitig eine ME das Ergebnis einer Speicheroperation an eine PE übertragen soll, während diese aber zur gleichen Zeit eine Supply-Operation durchführen muß. In diesem Fall versuchen sowohl die PE als auch die ME Daten über den Crossbar-Kanal der PE zu senden.

2. Wenn das Ergebnis einer Speicheranfrage durch eine Supply-Operation an eine PE geschickt werden soll, während diese selbst eine Supply-Operation durchzuführen hat. In diesem Fall versuchen zwei PEs, die über einen ME-Crossbar-Kanal verbunden sind, gleichzeitig Daten auf diesem Kanal zu übertragen.

Die erste Art von Konflikten kann verhältnismäßig einfach gelöst werden. Dazu muß zum einen sowohl die PE als auch die ME vor einer Datenübertragung prüfen, ob der Crossbar-Kanal nicht bereits benutzt wird. Außerdem muß man eine eindeutige Priorität zwischen der Übertragung von Ergebnissen und Supply-Operationen festlegt. Ersteres verhindert, daß jemand Daten auf dem Kanal sendet, während eine andere Übertragung im Gange ist. Letzteres stellt sicher, daß es keine Verklemmung gibt, falls die PE und die ME zur gleichen Zeit eine Übertragung starten wollen.

Leider ist eine solche einfache Lösung im zweiten Fall nicht möglich, da sie zu Verklemmungen führen könnte. Das Problem ergibt sich, wenn zwei PEs gleichzeitig versuchen, sich gegenseitig Daten durch Supply-Operationen zu schicken. Egal, ob man der Übertragung von Ergebnissen oder Supply-Operationen Vorrang gibt, kommt es in einem solchen fall zu Probelmen. Wenn man die Übertragung von Ergebnissen bevorzugt behandelt wird, dann setzen beide PEs die Durchführung der Supply-Operation aus und warten auf Daten. Anderenfalls würden beide gleichzeitig ihre Daten übertragen, was nicht zulässig ist.

Grundidee

Die Arbitrierung der Netzwerkkanäle wird mit der Vergabe der Speicherbänke zusammengelegt. Mit dem Zugriff auf eine Speicherbank bekommt eine PE also automatisch auch den Zugriff auf den Crossbar-Kanal der zugehörigen ME. Sie baut dann sofort eine Crossbar-Verbindung zur betroffenen ME auf. Der Crossbar-Kanal wird freigegeben, sobald die Speicheroperation beendet ist und das Ergebnis an die PE übermittelt wurde. Durch das obige Vorgehen wird die Gefahr von Verklemmungen gebannt, die eine getrennte Vergabe der Speicherbänke und der Crossbar-Kanäle mit sich bringen würde.

1. Bei Zugriffskonflikten zwischen PEs und MEs wird Supply-Operationen der Vorrang vor der Ergebnisübertragung eingeräumt. Eine ME muß also mit der Datenübertragung warten, falls die PE eine Supply-Operation durchführen möchte.

2. Um Zugriffskonflikte zwischen zwei PEs bei Supply-Operationen zu vermeiden, wird wie folgt vorgegangen:

   (a) Falls eine PE eine Supply-Operation durchführt während sie auf das Ergebnis einer Speicheroperation wartet, dann trennt sie für die Dauer der Supply-Operation die bestehende Verbindung zur ME. Dadurch werden Zugriffskonflikte durch Supply-Operationen vermieden. Während der Zeit, in der die Verbindung getrennt ist, bleibt die PE aber im Besitz des Kanals und der ME. Die Verbindung wird am Ende der Supply-Operation wieder aufgebaut, ohne daß eine erneute Arbitrierung notwendig ist.

(b) Falls bei einer Supply-Operation die PE, für die die Daten bestimmt sind, nicht mit dem ME-Kanal verbunden ist (weil sie gerade selbst eine Supply-Operation durchführt), dann werden die ankommenden Daten von der ME empfangen. Die ME speichert die Daten zwischen und leitet sie an die Ziel-PE weiter, sobald diese die getrennte Crossbar-Verbindung wieder aufgebaut hat.

Dies bedeutet, daß sich die PE, die eine Supply-Operation durchführt, nicht um die Konfliktauflösung kümmern muß. Sie stellt einfach die Verbindung mit dem Crossbar-Kanal der entsprechenden ME her und sendet die Daten. Die PE, für die die Daten bestimmt ist, ist entweder bereit, sie zu empfangen, oder sie ist gar nicht mit dem Crossbar-Kanal verbunden. Im letzten Fall ist die ME empfangsbereit, so daß die Daten auf jeden Fall übertragen werden.

Durchführung einzelner Speicheroperationen

Wie bereits beschrieben, laufen Invalidate- und Exclusive-Operationen im PHOTOBAR-System genauso wie im PHOTOBUS-System ab. Gleiches gilt für Synchronisationsoperationen, die vollständig auf dem Chip durch die SE und den SB abgewickelt werden. Die anderen Operationen werden wie folgt ausgeführt:

WriteBack: Bei einer WriteBack-Operation fordert die PE zunächst den Zugang zur entsprechenden ME. Sobald ihr dieser gewährt wird, stellt sie die Crossbar-Verbindung her und schickt die Daten an die ME. Sobald die Übertragung beendet ist, wird die Verbindung wieder getrennt und die ME freigegeben.

FetchRead bzw. FetchWrite ohne Supply: Bei einer Fetch-Operation fordert die PE zunächst den Zugang zur Speicherbank und zum B-Kanal. Sie überträgt dann die Anfrage auf dem B-Kanal, stellt die Crossbar-Verbindung zur ME her, schaltet ihre Crossbar-Schnittstelle auf Empfang und wartet auf das Ergebnis. Während der Wartezeit werden bei Bedarf Supply-Operationen durchgeführt. Während dieser Operationen wird die Verbindung zur ME getrennt. Falls die Daten von der Speicherbank in der ME ankommen während die Verbindung getrennt ist, werden sie dort zwischengespeichert. Sobald die Verbindung wieder hergestellt ist, schickt die ME das Ergebnis an die PE. Daraufhin trennt die PE die Verbindung, und die ME samt Crossbar-Kanal wird für andere Operationen freigegeben.

FetchRead bzw- FetchWrite mit Supply: Am Anfang einer Fetch-Operation ist nicht bekannt, ob ein Supply durchgeführt wird oder nicht. Die Operation verläuft daher bis zu dem Zeitpunkt wie oben beschrieben, an dem eine andere PE eine Supply Benachrichtigung auf ihrem Kanal erhält. Diese PE (im nachfolgenden als Supply-PE bezeichnet) teilt dies zunächst wie beim OE-VLSI-Bus über die entsprechende F-Leitung der ME mit. Die ME weiß damit, daß sie keine Daten von der Speicherbank weiterleiten darf. Allerdings geht sie anders als beim PHOTOBUS nicht sofort in den Frei-Zustand über, kann also nicht sofort an eine andere PE vergeben werden. Dies ist notwendig, da bei der Arbitrierung der Speicherbänke automatisch auch der Zugriff auf die Crossbar-Kanäle der MEs vergeben wird. Da der Kanal für die Übertragung der Supply Daten benötigt wird, kann aber zu diesem Zeitpunkt noch nicht an eine andere PE weitergegeben werden. Außerdem muß die ME bei Bedarf die Supply-Operation empfangen und zwischenspeichern. Wenn die Supply-PE die die benötigten Daten von ihrem Prozessor erhält, stellt sie eine Verbindung zum Crossbar-Kanal der ME her und führt die Supply-Operation durch. Falls zu diesem Zeitpunkt die PE mit dem ME-Kanal verbunden ist, dann empfängt sie die Daten selbst. Ansonsten werden die Daten von der ME empfangen, gespeichert und später an die PE weitergeleitet. Sobald die Daten von der PE empfangen wurden, wird die Crossbar-Verbindung getrennt und die ME freigegeben. Die Fetch-Operation ist damit beendet.

Lock über den Speicher: Eine Lock-Operation, die an den Speicher weitergegeben werden muß, verläuft fast genauso wie eine Fetch-Operation ohne Supply. Der einzige Unterschied besteht darin, daß die Anfrage an die Speicherbank nicht über den B-Kanal, sondern über das Crossbar-Netzwerk an die ME und dann über den M-Kanal an die Speicherbank geleitet wird.

Unlock über den Speicher: Eine Unlock-Operation, die an eine Speicherbank weitergeleitet werden muß, wird genauso abgewickelt wie eine WriteBack-Operation.

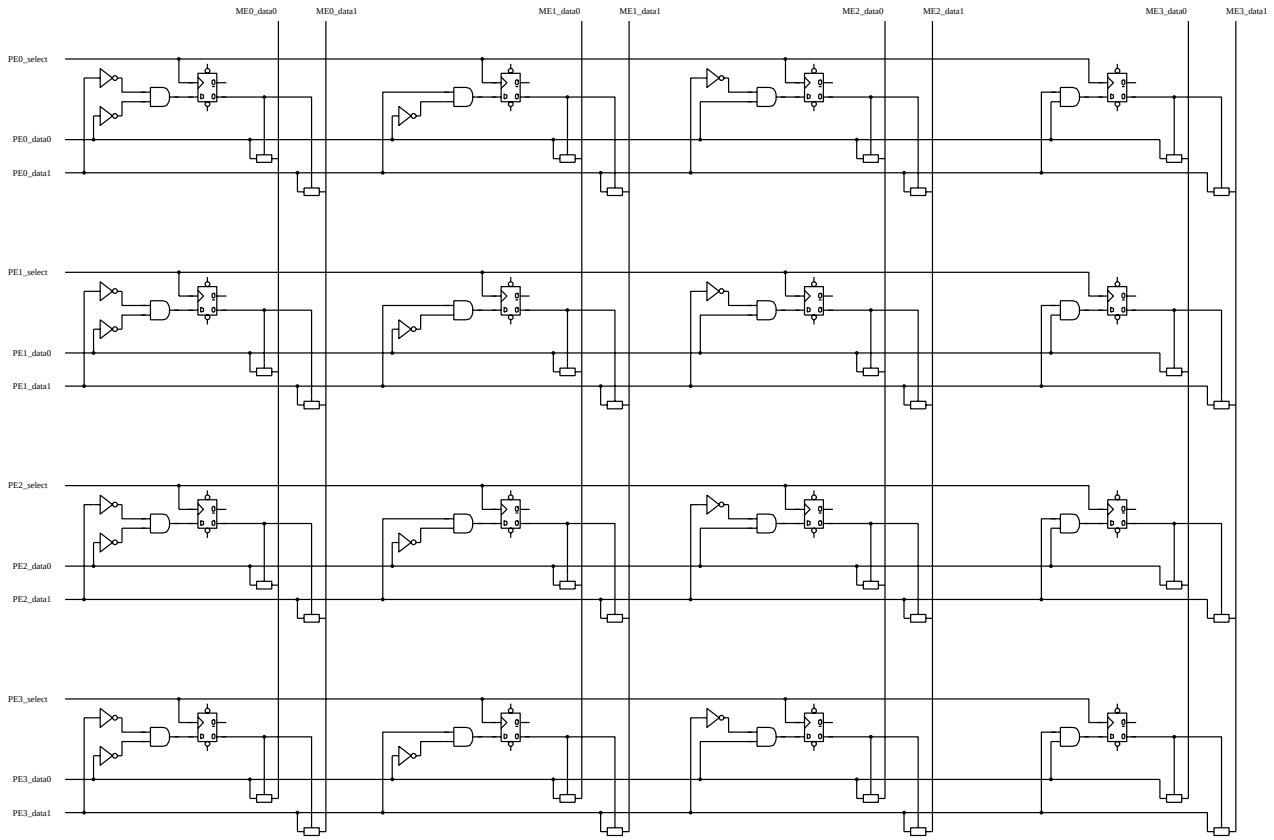


Abbildung 8.7: Der Aufbau der Crossbar-Hardware am Beispiel eines Systems mit 4 PEs, 4MEs und zwei Datenleitungen.

8.3.2 Das Crossbar-Netzwerk

Wie bereits beschrieben, besitzt bei einer Crossbar-Verbindung jeder PE und jeder ME einen eigenen Datenkanal. Wir gehen im Nachfolgenden davon aus, daß der Datenkanal aus b_c Datenleitungen, einer STROBE und einer CLOCK-Leitung besteht. Der Datenkanal einer jeden PE ist über einen Schalter mit den Datenkanälen aller MEs verbunden. Der Schalter kann eine direkte, bidirektionale Verbindung zwischen den korrespondierenden Leitungen der beiden Datenkanäle herstellen. Um eine Verbindung mit einer bestimmten ME herzustellen, muß die PE also lediglich den geeigneten Schalter aktivieren.

Die wichtigsten Aspekte der obigen Netzwerkarchitektur sind der Aufbau der Schalter und die Realisierung des Auswahlmechanismus für die Aktivierung der Schalter. Hinzu kommt die Frage, woran eine ME merkt, ob die PE mit ihr verbunden und empfangsbereit ist, oder ob sie gerade die Verbindung zur Durchführung einer Supply-Operation getrennt hat. In dem PHOTOBAR-Chip werden dabei folgende Ansätze verfolgt:

Schalteraufbau: Da die Schalter schnell, kompakt und bidirektional sein müssen, werden sie mit Hilfe von Transmission Gates (TG) realisiert.

Auswahlmechanismus: Für die Auswahl des zu aktivierenden Schalters werden die Datenleitungen zusammen mit einer zusätzlichen SELECT Leitung benutzt. Die Verbindung zwischen einer bestimmten PE und ME wird aufgebaut, indem die PE die Nummer der ME auf die Datenleitungen und danach die SELECT-Leitung auf 1 schaltet. Die Trennung der Verbindung erfolgt mit Hilfe einer READY-Leitung.

Erkennung der Empfangsbereitschaft: Um der ME ihre Empfangsbereitschaft zu signalisieren, bekommt jede PE eine ONLINE-Leitung. Während die PE mit der ME verbunden ist und bereit ist, das Ergebnis zu empfangen, wird diese Leitung auf 1 gesetzt. Ist die Verbindung getrennt, so liegt an der Leitung keine Spannung und die ME weiß, daß die PE nicht empfangsbereit ist.

Das Prinzip der Harwarerealisierung ist in Abbildung 8.7 am Beispiel eines System mit 4 PEs, 4 MEs und zwei Datenleitungen pro Datenkanal dargestellt. Jeder besteht aus einer Kontrollschaltung sowie einer TG für jede

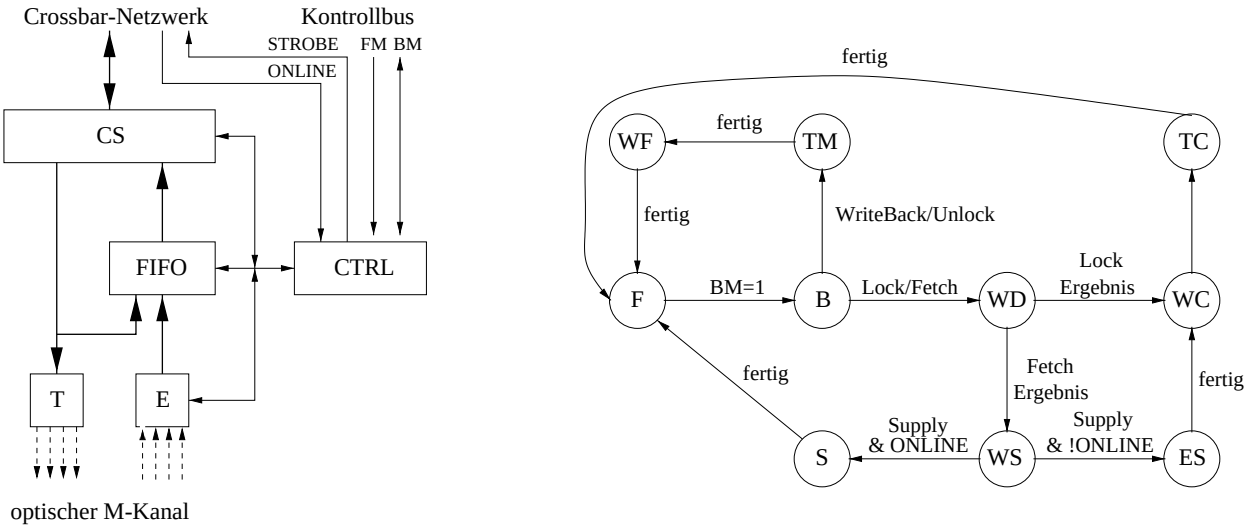


Abbildung 8.8: Die Abbildung zeigt den Aufbau (links) und Zustandsübergangsdiagramm (rechts) einer ME im OE-VLSI-Schalter.

Datenleitung eines Kanals. Die Kontrollschaltung ist an die Datenleitungen sowie an die SELECT- und READY-Leitung angeschlossen. Die Kontrollschaltung aktivieren die TGs, sobald an der SELECT-Leitung der PE eine 1 anliegt und gleichzeitig die Signale an den Datenleitungen der PE mit der Nummer der ME übereinstimmen. Sie deaktivieren sie, wenn die Ready-Leitung der PE oder ME auf 1 geht. Um diese Funktionalität mit möglichst wenig Hardwareaufwand zu realisieren, besitzt jede Kontrollschaltung einen $ld(M)$-Bit Vergleicher und einen Flip-Flop. Der Ausgang des Flip-Flops ist an die Kontrolleingänge der TGs angeschlossen. Der Zustand des Flip-Flops bestimmt also, ob die TGs auf Durchlaß geschaltet oder geschlossen sind. Der Vergleicher prüft, ob die an den Datenleitungen anliegenden Signale und die ME-Nummer übereinstimmen. Sein Ausgang ist mit dem Dateneingang eines Flip-Flops verbunden, dessen CLOCK-Eingang an die SLECT-Leitung angeschlossen ist. Je nachdem, ob der Vergleich positiv oder negativ ausfällt, wird das Flip-Flop also mit steigender Flanke der SELECT-Leitung entweder gesetzt oder gelöscht. Darüberhinaus kann das Flip-Flop auch über die READY-Leitung gelöscht werden, die an seinen RESET-Eingang angeschlossen ist.

8.3.3 Die ME

Die ME ist im PHOTOBAR-System für drei Aufgaben zuständig:

1. Die Übertragung von WriteBack-Daten und Synchronisationsanfragen zur Speicherbank,
2. Die Weitergabe von Ergebnissen von der Speicherbank an die PE,
3. Das Zwischenspeichern von Supply-Operationen, die während der Zeit ankommen, in der die PE von dem ME-Kanal getrennt ist.

Der hierfür notwendige Aufbau der ME und ihre Funktionsweise können wie folgt zusammengefaßt werden:

Aufbau der ME

Die ME besteht aus einem optischen Empfänger (E), einem optischen Sender (T), einem FIFO-Speicher, einer Crossbar-Schnittstelle (CS) und einer Kontrolleinheit (CTRL). Um das Zwischenspeichern von Supply-Operationen zu ermöglichen, kann der FIFO-Speicher sowohl von dem optischen Empfänger als auch von der CS Daten empfangen. Für das Senden von Daten auf dem M-Kanal wird kein zusätzlicher Zwischenspeicher benötigt, da ankommende Daten sofort gesendet werden können. Der Aufbau ist in Abbildung 8.8 links zu sehen.

Der Empfänger wandelt, den hochfrequenten optischen in einen niederfrequenten elektronischen Datenstrom um und leitet ihn an die Crossbar-Schnittstelle weiter. Der Sender erfüllt die umgekehrte Funktion: er generiert aus dem niederfrequenten elektrischen Datenstrom der Crossbar-Schnittstelle das hochfrequente optische Signal auf dem M-Kanal. Die Schnittstelle selbst besteht aus $b_c + 2$ bidirektionalen Treibern für die Crossbar-Leitungen

und $b_c + 2$ TG-Multiplexern. Die Multiplexer verbinden die Treiber je nach Datenrichtung entweder mit dem Sender oder mit dem Empfänger. Die Kontrolleinheit ist an die FM_i- und BM_i-Leitungen des KB, und an die die STROBE- und ONLINE-Leitungen des korrespondierenden Crossbar-Kanals angeschlossen.

Die Funktionsweise der ME

Die ME tritt in Aktion, sobald eine PE ihre BM-Leitung auf 1 setzt. Als erstes schaltet sie ihre Crossbar-Schnittstelle auf Empfang. Falls innerhalb des nächsten Zyklus Daten von der PE ankommen, werden sie über den Transmitter an die Speicherbank weitergegeben. Handelt es sich bei der Operation um ein WriteBack, dann ist der Zugriff damit beendet und die ME betrachtet sich als frei. Anderenfalls geht die ME in Wartestellung, bis entweder die Antwort von der Speicherbank auf dem M-Kanal ankommt, oder eine Supply-Operation durch eine 1 auf der FM-Leitung angezeigt wird. Im ersten Fall leitet die Crossbar-Schnittstelle die ankommenden Daten auf dem Crossbar-Kanal weiter. Im zweiten Fall läßt die ME ihre Crossbar-Schnittstelle auf Empfang und wartet, bis die Supply-Daten übertragen wurden. Dabei überwacht sie auch ständig die ONLINE-Leitung. Wenn die Supply-Operationen durchgeführt wird, während auf der ONLINE-Leitung keine 1 anliegt, dann speichert die ME die Daten in ihrem FIFO zwischen und leitet sie später an die PE weiter. Sonst ist der Zugriff mit der Supply-Operation beendet und die ME ist frei für weitere Anfragen.

Die Funktionsweise der ME kann mit Hilfe eines endlichen Automaten mit 8 Zuständen beschrieben werden. Dazu gehören ein Frei-Zustand, (F) ein Besetzt-Übergangszustand (B), zwei Zustände für die Abwicklung von WriteBack-Operationen und 4 Zustände für Fetch-Anfragen. Die Zustandsübergänge des Automaten werden durch die BM- und FM-Leitungen des KBs, die S-Leitung des Crossbar-Netzwerks und die Supply-Leitung des M-Kanals gesteuert.

Frei (F): Die ME befindet sich im F-Zustand, wenn keine Anfragen an ihre Speicherbank vorliegen. Sie geht in den B-Zustand über, sobald ihre BM-Leitung auf 1 gesetzt wurde.

Besetzt (B) : Der B-Zustand ist ein Übergangszustand, in dem die ME zwar als 'besetzt' markiert ist, aber noch nicht weiß, welche Art von Operation sie durchführen soll. Die ME verbleibt im B-Zustand genau einen Taktzyklus lang. Falls während dieser Zeit die S-Leitung des Crossbar-Netzwerks auf 1 geht (also Daten über das Netzwerk übertragen werden) geht sie in den WB-Zustand und führt eine WriteBack-Operation durch. Sonst folgt der WD-Zustand.

WriteBack (TM): Im TM-Zustand wird eine WriteBack oder Unlock Anfrage über den M-Kanal an die Speicherbank übertragen. Sobald die Übertragung beendet ist, geht die ME in den WF-Zustand über.

Warten auf Fertigmeldung (WF): Eine ME im WF-Zustand wartet darauf, daß die Speicherbank das Ende einer WriteBack- oder Unlock-Operation meldet. Sobald dies passiert ist, geht sie in den F-Zustand zurück.

Warten auf Daten (WD): Der WD-Zustand markiert den Anfang einer Fetch- bzw. Lock-Operation. Die ME wartet darauf, daß entweder die Daten von der ME ankommen, oder eine Supply-Operation durch eine 1 auf der FM-Leitung angezeigt wird. Ersteres hat einen Übergang in den WC-Zustand zur Folge. Letzteres bewirkt einen Wechsel in den WS-Zustand.

Warten auf Crossbar (WC): Eine ME im WC-Zustand wartet darauf, daß die PE ihre Bereitschaft zum Empfangen von Daten auf dem Crossbar-Kanals signalisiert.

Datenübertragung über Crossbar (TC): Im TB-Zustand werden Daten über das Crossbar-Netzwerk an die PE übertragen. Die ME verläßt diesen Zustand, sobald die Übertragung beendet ist und kehrt in den F-Zustand zurück.

Warten auf Supply (WS): Im WS-Zustand wartet die ME darauf, daß die Supply-Daten über ihren Crossbar-Kanal übertragen werden. Sobald die Übertragung anfängt, geht sie in den S-Zustand.

Supply (S): Der S-Zustand bedeutet, daß Supply-Daten über den Crossbar-Kanal der ME übertragen und von der PE empfangen werden. Die ME verläßt den S-Zustand, wenn die Übertragung beendet ist und geht in den F-Zustand zurück.

Empfange Supply (ES): im ES-Zustand werden Supply-Daten über den Crossbar-Kanal der ME übertragen, während die PE nicht mit dem Crossbar-Kanal verbunden ist, sie also nicht empfangen kann. Die Daten werden in diesem Fall von der ME in ihrem FIFO zwischengespeichert. Sobald die Übertragung beendet ist, geht die ME in den WC-Zustand über.

8.3.4 Die PE

Im Vergleich zur PHOTOBUSArchitektur unterschiedet sich die Aufgabe der PE in der PHOTOBARArchitektur in drei Punkten:

1. WriteBack- und Synchronisationsanfragen an die Speicherbank werden über das Crossbar-Netzwerk an die MEs übertragen. Dies bedeutet, daß:
   (a) die PE für diese Anfragen nicht auf die Verfügbarkeit des B-Kanals warten muß und
   (b) nachdem die Anfrage über den Crossbar-Kanal geschickt wurde, eine Bestätigung über den P-Kanal an den Prozessor geschickt werden muß. Dadurch wird der Prozessor benachrichtigt, daß die PE nun frei ist und neue Daten empfangen kann.
2. Die Antwort auf FetchRead und FetchWrite Anfragen wird über den Crossbar-Kanal an die PE übertragen, die sie über den P-Kanal an die PE weiterleiten muß. Dies gilt unabhängig davon, ob die Daten von der Speicherbank oder von einem anderen Prozessor durch eine Supply-Operation zur Verfügung gestellt werden.
3. Wenn die PE selbst eine Supply-Operation durchführt, so muß sie diese über den Crossbar-Kanal abwickeln. Dabei sind zwei Dinge zu beachten:
   (a) Falls die PE gleichzeitig auf die Antwort und auf eine Fetch-Anfrage wartet, muß dabei sichergestellt werden, daß die Übertragung nicht mit ankommenden Daten kollidiert.
   (b) Nach Abschluß der Supply-Operation muß die PE über den P-Kanal eine Erfolgsmeldung an den Prozessor schicken. Dieser weiß dann, daß er weitere Anfragen an den SP-Chip schicken kann.
   (c) Falls die PE gleichzeitig eine Synchronisationsoperation durchführt, dann kann es sowohl beim Zugriff auf die Crossbar-Schnittstelle als auch bei der Übertragung der Erfolgsmeldung über den P-Kanal zu Zugriffskonflikten kommen.

Bei der Beschreibung der PE wird im Nachfolgenden zunächst die Problematik der Zugriffskonflikte erörtert. Daraufhin werden der Aufbau der PE und ihre Funktionsweise betrachtet. Dabei gehen wir wie beim PHOTOBUS mit On-Chip Synchronisation davon aus, daß die Synchronisation von einer Untereinheit der PE, der PESE erledigt wird. Diese arbeitet bis auf die Zugriffe auf den Crossbar-Kanal und den P-Kanal Transmitter unabhängig von den übrigen PE-Komponenten. Sie wird daher getrennt am Ende des Abschnittes beschrieben.

Zugriffskonflikte und ihre Auflösung

In der PE der PHOTOBARArchitektur können Supply-Anfrage parallel mit Synchronisationsanfragen und mit dem Warten auf das Ergebnis einer Fetch-Operation durchgeführt werden. Dabei können sich drei Arten von Zugriffskonflikten ergeben:

1. ein Konflikt beim Zugriff auf den Crossbar-Kanal, wenn eine Supply-Operation ausgeführt werden soll, wenn gleichzeitig das Ergebnis einer Fetch-Operation auf dem Crossbar-Kanal eintrifft,
2. ein Konflikt beim Zugriff auf den Crossbar-Kanal, wenn eine Supply-Operation ausgeführt werden soll, wenn gleichzeitig Daten das Ergebnis einer über den Speicher ablaufenden Lock-Operation auf dem Crossbar-Kanal eintrifft,
3. ein Konflikt beim Zugriff auf den Crossbar-Kanal, wenn gleichzeitig eine Synchronisationsoperation und eine Supply-Operation über den Crossbar-Kanal geschickt werden sollen und
4. ein Konflikt beim Zugriff auf den P-Kanal Transmitter wenn gleichzeitig das Ergebnis einer Synchronisationsoperation sowie die Fertig-Meldung nach einer durchgeführten Supply-Operation an den Prozessor geschickt werden sollen.

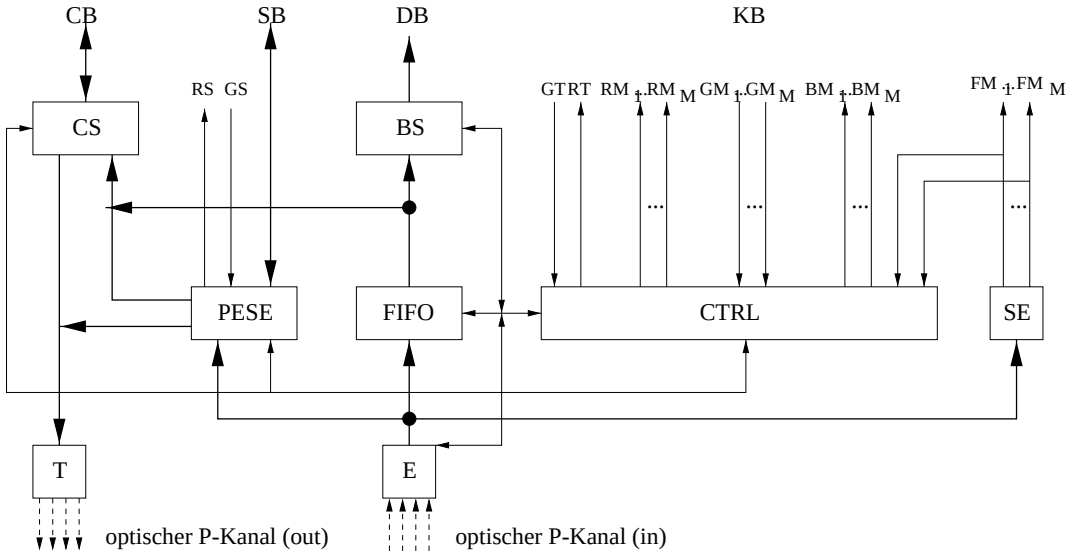


Abbildung 8.9: Aufbau der PE der PHOTOBAR-Architektur

Die ersten beiden Konflikte und deren Auflösung wurden bereits im Zusammenhang mit dem Crossbar-Protokoll (Abschnitt 8.3.1) erläutert. Sie bedürfen daher keiner weiteren Erörterung. Die letzten beiden sind eine PHOTOBAR-spezifische Abwandlung des Konflikts, der bei der PHOTOBUS-Architektur beim Zugriff auf dem B-Kanal auftritt. Wie in Abschnitt 7.9.2, wartet der Prozessor nach einer Synchronisationsoperation nicht mit dem Verschicken von Supply-Operationen auf eine Fertig-Meldung vom Chip. Dies ist möglich, da die Synchronisationsoperationen in der PESE zwischengespeichert werden. Allerdings führt dies dazu daß sich beliebige Phasen der Synchronisation mit beliebigen Phasen der Supply-Operationen überlappen. Dadurch kann es beim Verschicken von Anfragen an den Speicher und bei der Übertragung von Fertig-Meldungen zu Zugriffskonflikten kommen. Da beim PHOTOBUS alle Nachrichten über den B-Kanal verschickt werden, treten dort alle Konflikte bei DB Zugriffen auf. Bei der PHOTOBAR-Architektur verteilen sie sich hingegen auf den Crossbar-Kanal und den P-Kanal Transmitter. Diese Auftrennung ändert jedoch nichts an dem Verfahren für die Auflösung der Konflikte. Auch beim PHOTOBAR-System findet bei gleichzeitigen Zugriffen eine faire Arbitrierung durch die Kontrolleinheit der PE statt. Allerdings werden hier zwei Arbitrierungsschaltkreise benötigt: einer für den Crossbar-Kanal und einer für den P-Kanal Transmitter.

Aufbau der PE

Die Architektur den PE, die sich aus den obigen Überlegungen ergibt, ist in Abbildung 8.9 zu sehen. Sie ist bis auf den P-Kanal Transmitter (T) und die Schnittstelle zum Crossbar-Netzwerk (CS) mit der PE der PHOTOBUS-Architektur identisch. Der Transmitter und die CS gleichen denen der ME (siehe Abschnitt 8.3.3). Da sowohl Synchronisationsanfragen als auch WriteBack und Supply-Anfragen über das Crossbar-Netzwerk übertragen werden, ist der Eingang der CS sowohl an das FIFO als auch an die PESE angeschlossen. Der Ausgang der CS ist direkt mit dem Transmitter des P-Kanals verbunden. Ein FIFO für die Zwischenspeicherung von Daten ist hier nicht notwendig, da der P-Kanal Transmitter immer frei ist, wenn über die CS Daten für den P-Kanal ankommen.

Funktionsweise der PE

Eine formale Beschreibung durch das Zustandsübergang eines endlichen Automaten ist in Abbildung 8.10 zu sehen. Das Diagramm unterscheidet sich in zwei Punkte von dem Automaten, der die PE der PHOTOBUS-Architektur beschreibt:

1. Er besitzt vier zusätzliche Zustände. Zwei davon sind für die Handhabung der Datenübertragung über das Crossbar-Netzwerk, zwei für die Übertragung über den Transmitter des P-Kanals gewidmet. Zur ersten Gruppe gehören der WC-Zustand (Warten auf Crossbar) und der TC-Zustand (Datenübertragung auf dem Crossbar). Die zweite beinhaltet den WP- (Warten auf den Transmitter) und den TP-Zustand (Senden auf dem P-Kanal).

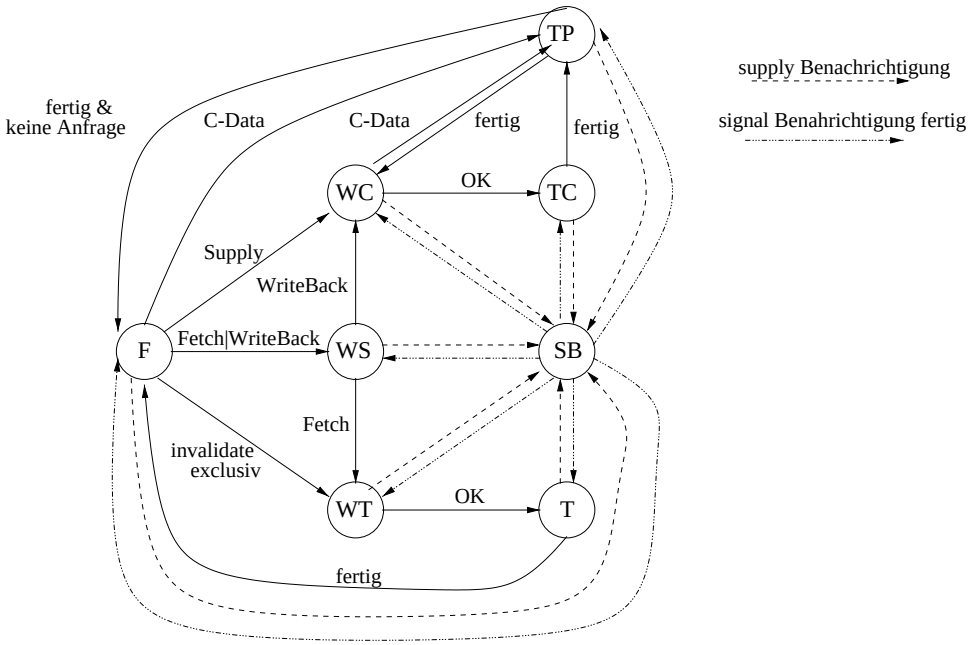


Abbildung 8.10: Das Zustandsübergangsdiagram der PE in der PHOTOBAR-Architektur.

2. Die Abwicklung von Supply- und WriteBack-Operationen findet nicht über den WT- und T-, sondern über die neuen Zustände statt, da diese Operationen über das Crossbar-Netzwerk ablaufen.

Das Zustandsübergangsdiagramm des resultierenden endlichen Automaten ist in Abbildung 8.10 zu sehen. Darin ist zu erkennen, daß die übrigen Zustände und Übergänge mit denen der PHOTOBUS-Architektur identisch sind. Dies gilt auch für den SB-Zustand, der der Durchführung von Supply-Benachrichtigungen dient. Ein Wechsel in diesen Zustand ist auch bei der PHOTOBAR-Architektur aus jedem Zustand, inklusive der neuen Zustände möglich.
Die Bedeutung der neuen Zustände kann wie folgt zusammengefaßt werden:

Warten auf Crossbar (WC): Die PE kann aus dem F- oder aus dem WS-Zustand in den WC-Zustand gelangen. Ersteres passiert, wenn im F-Zustand eine Supply-Operation vom Prozessor ankommt. Letzteres erfolgt, nachdem der PE bei einer WriteBack-Operation der Zugriff auf die Speicherbank gewährt wurde. Im WC-Zustand wartet die PE, bis der Crossbar-Kanal für eine Datenübertragung frei wird. Er kann wie beschrieben durch ankommende Daten oder durch die Übertragung einer an den Speicher gerichteten Synchronisationsoperation belegt sein. Sobald der Crossbar Kanal frei wird, geht sie für die Datenübertragung in den TC Zustand über.

Übertragung auf dem Crossbar (TC): Dieser Zustand dient der Übertragung von Daten über das Crossbar-Netzwerk (TC-Zustand). Die PE verläßt diesen Zustand, sobald die Übertragung beendet ist und geht in den WP-Zustand.

Warten auf den P-Kanal-Transmitter (WP): Im WP-Zustand wartet die PE darauf, daß der P-Kanal Transmitter verfügbar wird. Er kann im ungünstigen Falle durch die Übertragung des Ergebnisses einer auf dem Chip abgewickelten Synchronisationsoperation besetzt sein. Sobald der Transmitter wieder frei wird, geht die PE in den TP-Zustand und beginnt mit der Datenübertragung.

Datenübertragung auf dem P-Kanal (TP): Der TP-Zustand dient der Datenübertragung auf dem optischen P-Kanal. Er kann entweder aus dem WP-Zustand oder aus dem F-Zustand erreicht werden. Im ersten Fall wird eine Fertig-Meldung an den Prozessor übertragen. Im zweiten wird eine von der CS kommende Antwort auf eine Speicheranfrage an den Prozessor weitergeleitet. Die Umgehung des Wartezustands (WP) und ein direkter Übergang in den TP-Zustand sind bei der Übertragung einer Speicherantwort möglich, da ein Konflikt beim Zugriff auf den Transmitter mit Sicherheit ausgeschlossen werden kann. Die PE verläßt den TP-Zustand am Ende der Transmission und geht in den F-Zustand zurück.

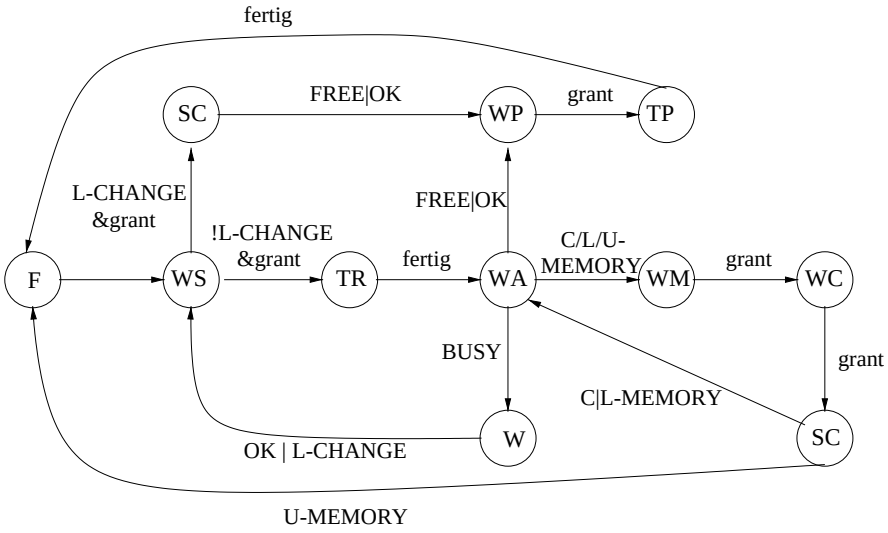


Abbildung 8.11: DasZustandsübergangsdiagramm der PESE der PHOTOBAR-Architektur.

Modifikationen der PESE

Die Aufgabe der PESE der PHOTOBAR-Architektur unterscheidet sich von der PHOTOBUS-Architektur in zwei Punkten:

1. Wenn Synchronisationsanfragen an den Speicher weitergereicht werden müssen, dann geschieht dies über das Crossbar-Netzwerk und den M-Kanal, nicht über den DB und den B-Kanal. Daraus folgt, daß die PESE die Speicherbank, nicht aber den B-Kanal anfordern muß. Allerdings muß sie unter Umständen auf die Verfügbarkeit des Crossbar-Kanals warten.

2. Die Erfolgsmeldungen an den Prozessor werden nicht über den DB und den B-Kanal sondern über den P-Kanal auf den Prozessor übertragen. Auch hier entfällt das Anfordern des B-Kanals. Dafür muß die PESE sich aber um den Zugang zum P-Kanal bemühen. Dieser kann, wie bereits erklärt, durch die Übertragung einer Fertig-Meldung nach einer Supply-Operation besetzt sein.

Obiges bedeutet, daß die PESE an die CS und an den P-Kanal Transmitter anstatt an die BS angeschlossen sein muß (siehe Abbildung 8.9). Was den eigentlichen Aufbau der PESE betrifft, so ergeben sich keine Veränderungen. Lediglich die Kontrollogik muß an die veränderte Funktionalität angepaßt werden. Diese Anpassung kann mit Hilfe des Zustandsübergangsdiagramms in Abbildung 8.11 formal beschrieben werden. Dieses Diagramm unterscheidet sich von dem entsprechenden Diagramm der PHOTOBUS- Architektur an folgenden Stellen:

1. Der WB (Warten auf B-Kanal) -Zustand worde durch einen WC (Warten auf Corssbar-Kanal) -Zustand ersetzt. Im WC-Zustand fordert die PESE bei der Kontrollogik der PE den Zugang zum Crossbar-Kanal und wartet darauf, daß ihr dieser gewährt wird.

2. Die beiden für die Datenübertragung auf dem B-Kanal verantwortlichen Zustände (BE und BR) wurden durch Übertragungszustände für den P-Kanal (TP) und das Corssbar-Netzwerk (TC) ersetzt. Im TP-Zustand wird die Erfolgsmeldung an den Prozessor verschickt. Im TC-Zustand wird die Synchronisationsanfrage an die zuständige Speicherbank weitergeleitet.

3. Es wurde ein zusätzlicher Zustand für das Warten auf die Verfügbarkeit des P-Kanals (WP) hinzugefügt. In diesem Zustand fordert die PESE von der Kontrollogik der PE den Zugang zum P-Kanal Transmitter und wartet, bis er ihr zugeteilt wird.

8.4 Chipfläche

Der Flächenbedarf des PHOTOBAR-Chips unterscheidet sich von dem des PHOTOBUS-Chips in folgenden Bereichen:

1. Es wird zusätzliche Fläche für die Leitungen und die Logik des Crossbar-Netzwerks benötigt.
2. Die PEs und MEs benötigen zusätzliche Gatter für die Implementierung der Crossbar-Schnittstellen und der P- bzw. M-Kanal Transmitter
3. Die Kontrolllogik der PEs und der MEs muß eine komplexere Aufgabe erfüllen, so daß auch hier zusätzliche Gatter benötigt werden.
4. Die MEs müssen nicht auf den B-Kanal und damit auf den DB zugreifen. Dies bedeutet zum einen, daß die DB-Schnittstellen der MEs und die dazugehörenden Anschlußleitungen an den DB nicht benötigt werden. Zum anderen entfällt beim KB die die Arbitirerungslogik samt Anschlußleitungen.

Bei der Betrachtung des Flächenbedarfs stützen wir uns auf die Betrachtung der PHOTOBUS-Architektur im vorigen Kapitel.

8.4.1 Crossbar-Fläche

Das Crossbar-Netzwerk trägt in zwei Bereichen zur Chip-Fläche bei: es vergrößert die für die Leitungen benötigte Fläche A_v um die Fläche der Crossbar-Leitungen A_{CB} und erhöht die Anzahl der benötigten Gatter NG_S um die Gatteranzahl der Corssbar-Schalter NG_{CB}.

Leitungsfläche

Bei der Abschätzung der Leitungsfläche gehen wir davon aus, daß die MEs und PEs wie beim PHOTOBUS gegenüber einander angebracht sind. Die Kanäle der MEs verlaufen dann wie der DB in der Mitte zwischen den PEs und den MEs, die Kanäle der PEs sowie die Anschlußleitungen der MEs verlaufen dann senkrecht dazu. Damit verlaufen parallel zum DB $M \cdot (b_c + 3)$ einzelne Leitungen während senkrecht dazu $(P + M) \cdot (b_c + 3)$ Leitungen benötigt werden (PE-Kanäle und die ME-Anschlüsse). Da die die gesamte Leitungsfläche aus dem Produkt der Breite der parallel verlaufenden und der senkrecht verlaufenden Leitungen ermittelt werden kann, gilt mit Gleichung 7.55 (für die Breite eines Leitungsbündels):

$$A_{CB} = (P + M) \cdot M \cdot N_{CB}^2 \cdot \left(\frac{W_L + D_L \Leftrightarrow 1}{N_{Lag}} \cdot \lambda \right)^2 \tag{8.1}$$

Kontrollschaltungen

Wie bereits beschrieben, wird für jede PE ein Schalter pro ME benötigt, so daß das Crossbar-Netzwerk insgesamt aus $P \cdot M$ Schaltern besteht. Wie in Abbildung 8.7 zu sehen, besitzt jeder Schalter eine TG pro Leitung, eine Flip-Flop sowie einen Vergleicher für $ld(M)$ Bits. Damit ergibt sich für die gesamte Gatteranzahl der Crossbar-Schalter:

$$NG_{CB} = P \cdot M(N_{CB} \cdot NG_{TG} + NG_{Bit} + ld(M)) \tag{8.2}$$

8.4.2 Neue Komponenten

Neu in der PHOTOBAR-Architektur sind im Vergleich zur PHOTOBUS-Architektur die Crossbar-Schnittstellen und die P- bzw. M-Kanal Transmitter. Die ersteren benötigen genauso wie die Busschnittstellen jeweils einen bidirektionalen Treiber für jede Leitung. Ihre Gatteranzahl NG_{CS} ist also bis auf die Anzahl der Leitungen mit der der Busschnittstellen identisch:

$$NG_{CS} = N_{CB} \cdot NG_{Tr} \tag{8.3}$$

Wir gehen im Nachfolgenden daher davon aus, daß die Gatteranzahl der Transmitter NG_T durch die der Empfänger angenähert werden kann:

$$NG_T \simeq NG_E \tag{8.4}$$

8.4.3 Auswirkung auf die einzelnen Komponenten

Unterschiede zum PHOTOBUStreten, wie beschrieben, bei der Arbitrierungslogik, bei der ME und bei der PE auf. Sie können wie folgt zusammengefaßt werden:

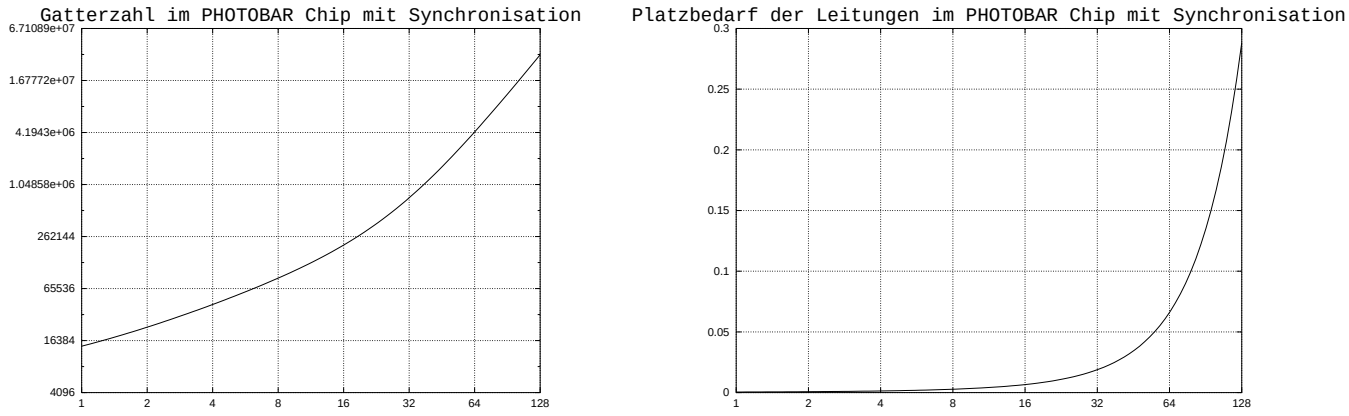


Abbildung 8.12: Das linke Diagramm zeigt die Abhängigkeit der auf dem PHOTOBAR Chips mit Synchronisation benötigten Gatter (vertikale Achse) von der Prozessorzahl (horizontale Achse). Das rechte Diagramm stellt den Anstieg der für die Verbindungsstruktur benötigten Fläche in cm^2 (vertikale Achse) mit der Prozessorzahl. Beide Diagramme verwenden eine logarithmische Darstellung.

Arbitrierungslogik

Da die MEs in der PHOTOBAR-Architektur nicht an den DB angeschlossen sind, entfallen die entsprechenden Arbitrierungsschaltkreise am KB. Dadurch wird aus Gleichung 7.54:

$$NG_{KB} = (P + PM) \cdot (2NG_{Tr} + NG_{Bit} + 2) \tag{8.5}$$

Veränderung des Flächenbedarfs der ME

Die Fläche der ME wird durch die Ersetzung der Busschnittstelle durch die Crossbarschnittstelle, die Hinzunahme des Transmitters sowie die notwendigen Erweiterungen der Kontrolleinheit verändert. Die Fläche der Kontrolleinheit kann, wie im Abschnitt 7.6.1 erklärt, durch Abzählen der Zustände und der Übergänge im Zustandsdiagramm (Abbildung 8.8) und ein Einsetzen in Gleichung (7.16) ermittelt werden. Durch geeignetes Aufrunden ergibt sich dann zusammen mit Gleichung (8.3) und (8.4) und (7.97) für die Fläche der ME im PHOTOBAR-System:

$$\begin{aligned} NG_M \simeq\ & \mathbf{2}(V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux}) + (l_C + l_A + l_K) \cdot NG_{bit} \\ & + (\mathbf{N_{CB}} + N_{SB}) \cdot NG_{Tr} + \mathbf{200}NG_{Bit} + l_k + \mathbf{1000} \end{aligned} \tag{8.6}$$

Veränderung des Flächenbedarfs der PE

Bei der PE werden sowohl die Gatteranzahl der PESE als auch die Gatteranzahl der eigentlichen PE verändert. Bei der PESE ist lediglich die Kontrolllogik betroffen. Für sie ergibt sich aus Abbildung 8.11 und Gleichung (7.16):

$$NG_{PS} = N_{SB} \cdot NG_{Tr} + (l_k + l_a) \cdot NG_{Bit} + 200NG_{Bit} + 500 \tag{8.7}$$

Bei der eigentlichen PE liegen die Veränderungen in der Kontrollogik, sowie der Crossbar-Schnittstelle und des P-Kanal Transmitters. Die Komplexität der Kontrollogik kann wie gehabt aus dem Zustandsübergangsdiagramm (Abbildung 8.10) und Gleichung (7.16) ermittelt werden. Ein Einsetzen in Gleichung 7.45 und großzügiges Aufrunden ergibt dann für die Gatteranzahl der PE der PHOTOBAR-Architektur:

$$\begin{aligned} NG_P \simeq\ & \mathbf{2}(V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux}) + (l_C + l_A + l_K) \cdot NG_{bit} \\ & + (N_{PK} \cdot V_F + 2 + 4M + 2 + \mathbf{N_{CB}}) \cdot NG_{Tr} + ld(M) \cdot M + l_k \\ & + \mathbf{400}NG_{Bit} + 10M + \mathbf{1000} + ldM \cdot M \end{aligned} \tag{8.8}$$

Zusammenfassung

Die obige Betrachtung hat gezeigt, daß sich die Gatterzahl einer einzelnen PE bzw. ME beim Übergang von der PHOTOBUS-Architektur zur PHOTOBAR-Architektur nur um jeweils einen konstanten Term erhöht, so daß

insgesamt eine zur P bzw. M proportionale Steigerung der Gatterzahl zu erwarten ist. Hinzu kommen die für die Realisierung der Schalter des Crossbars notwendigen Gatter, deren Zahl proportional ist zum Produkt von P und M sowie zum Term $P \cdot M \cdot ld(M)$ ist. Demgegenüber steht eine Abnahme der Gatterzahl der Arbitrierschaltung um einen in M linearen Faktor. Dies bedeutet, daß die Gesamtanzahl bei der PHOTOBAR-Architektur zusätzlich benötigter Gatter NG_{CB} als Polynom der folgenden Form geschrieben werden kann:

$$NG_{CB} = P \cdot M \cdot ld(M) \cdot NG_{CB}^{PMldM} + P \cdot M \cdot NG_{CB}^{PM} + P \cdot NG_{CB}^{P} + M \cdot NG_{CB}^{M} \tag{8.9}$$

mit

$$NG_{CB}^{PMldM} = 1 \tag{8.10}$$
$$NG_{CB}^{PM} = N_{CB} \cdot NG_{TG} + NG_{Bit} \tag{8.11}$$
$$NG_{CB}^{P} = (V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux}) + N_{CB} \cdot NG_{Tr} + 400NG_{Bit} + 1000 \tag{8.12}$$
$$NG_{CB}^{M} = (V_F \cdot (2N_{PK} + 3) \cdot NG_{Bit} + (V_F + 2) \cdot NG_{Mux}) + (N_{CB} \Leftrightarrow N_{PK} \cdot V_F) \cdot NG_{Tr} + 80NG_{Bit} \tag{8.13}$$

Obige Werte können benutzt werden, um die Abschätzung eines Zahlenwertes für die Gatteranzahl aus Gleichung 7.106 auf ein PHOTOBARSystem zu erweitern. Dazu wird lediglich ein Wert für die Anzahl der Leitungen des Crossbar-Netzwerks N_{CB} benötigt, den wir hier mit

$$N_{CB} = b_c + 4 = 20 \tag{8.14}$$

abschätzen. Damit ergibt sich mit eingen weiteren Aufrundungen:

$$\boxed{NG_S \simeq 3P \cdot M \cdot ld(M) + 100P \cdot M \cdot +6000P \cdot +3000M \cdot +5000} \tag{8.15}$$

Bei der Betrachtung der Fläche der Verbindungsstruktur muß lediglich Gleichung (8.1) berücksichtigt werden. Mit der Schätzung für die Anzahl der Leitungen der Crossbar-Kanäle (Gleichung (8.14))ergibt sich für die Abschätzung der Leitungsfläche auf der Basis von Gleichung 9.9:

$$\boxed{A_V[cm^2] \simeq (12P \cdot M^2 + 500M^2 + 1500P \cdot M \cdot +42000P \cdot +12000M + 50000) \cdot 6,25 \cdot 10^{-10}} \tag{8.16}$$

Die obigen Abschätzungen wurden in Abbildung 8.12 graphisch dargestellt. Dabei wurde, wie auch bei der Abschätzung für das PHOTOBUS-System, angenommen, daß es im System für jeden Prozessor eine Speicherbank gibt, es gilt also $P = M$. Aus den Graphen ist ersichtlich, daß bei 64 Prozessoren in etwa $4 \cdot 10^6$ Transistoren für die Schaltungen und ca. $0.8cm^2$ für die Verbindungsfläche benötigt werden. Bei 128 Prozessoren steigt die benötigte Gatteranzahl auf über 10^7, und die Verbindungsfläche auf $0.3cm^2$. Unter Berücksichtigung des Standes der VLSi-Technologie und der Vorhersagen für ihre Entwicklung in naher Zukunft (siehe [13]) ist damit die folgende Behauptung berechtigt:

Mit heutiger VLSI Technologie kann ein PHOTOBAR-System für bis zu 64 Prozessoren problemlos implementiert werden. In naher Zukunft sollte Dank der zu erwartenden Fortschritte in der VLSI-Technologie auch ein System für 128-Prozessoren möglich sein.

8.5 Leistungsbewertung

Im vorliegende Abschnitt wird die Leistung der PHOTOBAR-Architektur in Abhängigkeit von der Bandbreite der P-, M- und B-Kanäle untersucht. Im Nachfolgenden wird zunächst das Vorgehen bei der Simulation beschrieben. Danach werden die Ergebnisse präsentiert und ihre Bedeutung besprochen. Abschließend werden die Simulationsergebnisse mit den Vorhersagen eines einfachen schlangentheoretischen Modells der PHOTOBAR-Architektur verglichen, um ihre Plausibilität zu überprüfen.

8.5.1 Vorgehensweise

Für die Untersuchung werden eine Erweiterung des in Kapitel 5 beschriebenen Simulationssystems und dieselben SPLASH-2 Benchmarks eingesetzt, die beim PHOTOBUS System benutzt wurden. Die Vorgehensweise bei der Erweiterung der Simulationsumgebung und der Wahl der Parameter kann wie folgt zusammengefaßt werden.

Erweiterung der Simulation

Zur Simulation der PHOTOBAR-Architektur wurde das Simulationssystem so erweitert, daß es die bidirektionalen P- und M-Kanäle sowie das auf dem Chip integrierte Crossbar-Netzwerk modellieren kann. Dabei wurde nach dem gleichen Prinzip wie bei den anderen Teilen der Simulation vorgegangen: es wurden die einzelnen Teilkomponenten als unabhängige Objekte realisiert, die durch entsprechende Felder und Methoden den Datenfluß durch das System realitätsnah nachbilden. Damit kann die Simulation auch zur Verifikation der Protokolle herangezogen werden. Allerdings wurde die Nachbildung der Chiparchitektur um des Berechnungsaufwandes willen weniger genau als beim PHOTOBUS modelliert.

Die einzelnen Bearbeitungsschritte innerhalb des Chips ähneln bei der PHOTOBAR-Architektur stark den Bearbeitungsschritten auf dem PHOTOBUS-Chip. Aus diesem Grund wurde für die Bearbeitungszeiten der Nachrichten in den einzelnen Chip-Komponenten die Zeiten aus der Betrachtung in Abschnitt 7.7 übernommen. Dabei wurde angenommen, daß die Latenz der Crossbar-Übertragung mit der Latenz des DB gleich ist.

Bandbreite und Latenz der optischen Kanäle

Wie in Kapitel 6 erläutert, zielt die PHOTOBAR-Architektur auf kurz bis mittelfristig verfügbare Systeme ab. Die Schranken für Werte für die Latenz und die Bandbreite der optischen Kanäle können aus den entsprechenden Spalten der Tabelle 6.3 abgelesen werden. So liegt die maximale Bandbreite für die P- und M-Kanäle zwischen 4 und 64, für die B-Kanäle zwischen 4 und 128 Bytes/Cycle. Für die minimale Latenz wurde für die P- und M-Kanäle $2 + d/c$, für den B-Kanal $6 + d/c$ ermittelt.

Im Rahmen der Arbeit wurden Experimente mit einer Vielzahl von Parametern unternommen. Im Nachfolgenden werden die Ergebnisse für

$$1Bytes/cycle \leq B_B \leq 32Bytes/Cycle, \quad 1Bytes/Cycle \leq B_P = B_M \leq 4Bytes/Cycle, \tag{8.17}$$

dargestellt und besprochen. Die Wahl dieser Parameter ist durch zwei Faktoren bestimmt. In Bezug auf den B-Kanal hat sich, wie im Weiteren genauer erläutert wird, gezeigt, daß im Bereich von bis zu 128 Prozessoren eine Bandbreite von wenigen Bytes/Cycle völlig ausreichend ist. Die Wahl der P- und M-Kanal Bandbreite ist durch die Überlegung bestimmt, daß die Anzahl der optischen Ein-/Ausgabeleitungen, die an den Chip angekoppelt werden können, zu den kritischen Aspekten der Implementierung zählt (siehe Kapitel 6). Durch die Verwendung bidirektionaler Kanäle kommt das Problem bei der PHOTOBAR-Architektur stärker als beim PHOTOBUS zu tragen. Aus diesem Grund macht es wenig Sinn, hohe Bandbreitenwerte für die P-und M-Kanäle zu betrachten, da diese nur mit einer großen Anzahl einzelner optischer Datenleitungen pro Kanal erreicht werden kann.

Bei der Latenz gehen wir, so wie auch bei dem PHOTOBUS-System, von einem kompakten System aus, bei dem die Signalausbreitung zwischen den Prozessoren und dem Chip maximal 2 Prozssorzyklen dauert. Dies entspricht in etwa einem Systemradius von 1m, also einem großen Servergehäuse, oder einer Reihe eng nebeneinander angeordneter Arbeitsplatzrechner.

8.5.2 Ergebnisse

Die Ergebnisse der Simulation sind in den Abbildungen 8.13 und 8.14 zusammengefaßt. Dabei wurden zunächst die Laufzeiten der einzelnen Programme als Funktion der Prozessorzahl im Vergleich zur Laufzeit ein einem Parallelrechner mit eine idealen Speichersystem mit Cache dargestellt. In der Abbildung 8.14 ist dann die Beschleunigung für verschiedene Prozessorzahlen zu sehen.

Die wichtigsten Schlußfolgerungen aus den Ergebnissen können wie folgt zusammengefaßt werden:

1. Die Bandbreite des B-Kanals hat bei bis zu 128 Prozessoren nur eine geringe Auswirkung auf die Effizienz. Eine Saturierung des B-Kanals macht sich erst bei einer Bandbreite von 1Byte/Cycle ab ca 32 bis 64 Prozessoren bemerkbar.

2. Die Bandbreite der P- und M-Kanäle hat einen verhältnismäßig großen Einfluß auf die Ausführungszeit. In Anbetracht der Tatsache, daß viele lange Nachrichten über diese Kanäle übertragen werden, ist dies verständlich. Für die einzelnen untersuchten Banbreitenwerte gilt:

   (a) Die Bandbreite von 1Byte/cycle ist für Anwendungen mit einer hohen Anzahl von Speicherzugriffen (ocean, radix) nicht geeignet. Sie führt gegenüber dem idealen Speichersystem zu einer Verlangsamung um einen Faktor von bis zu 3.

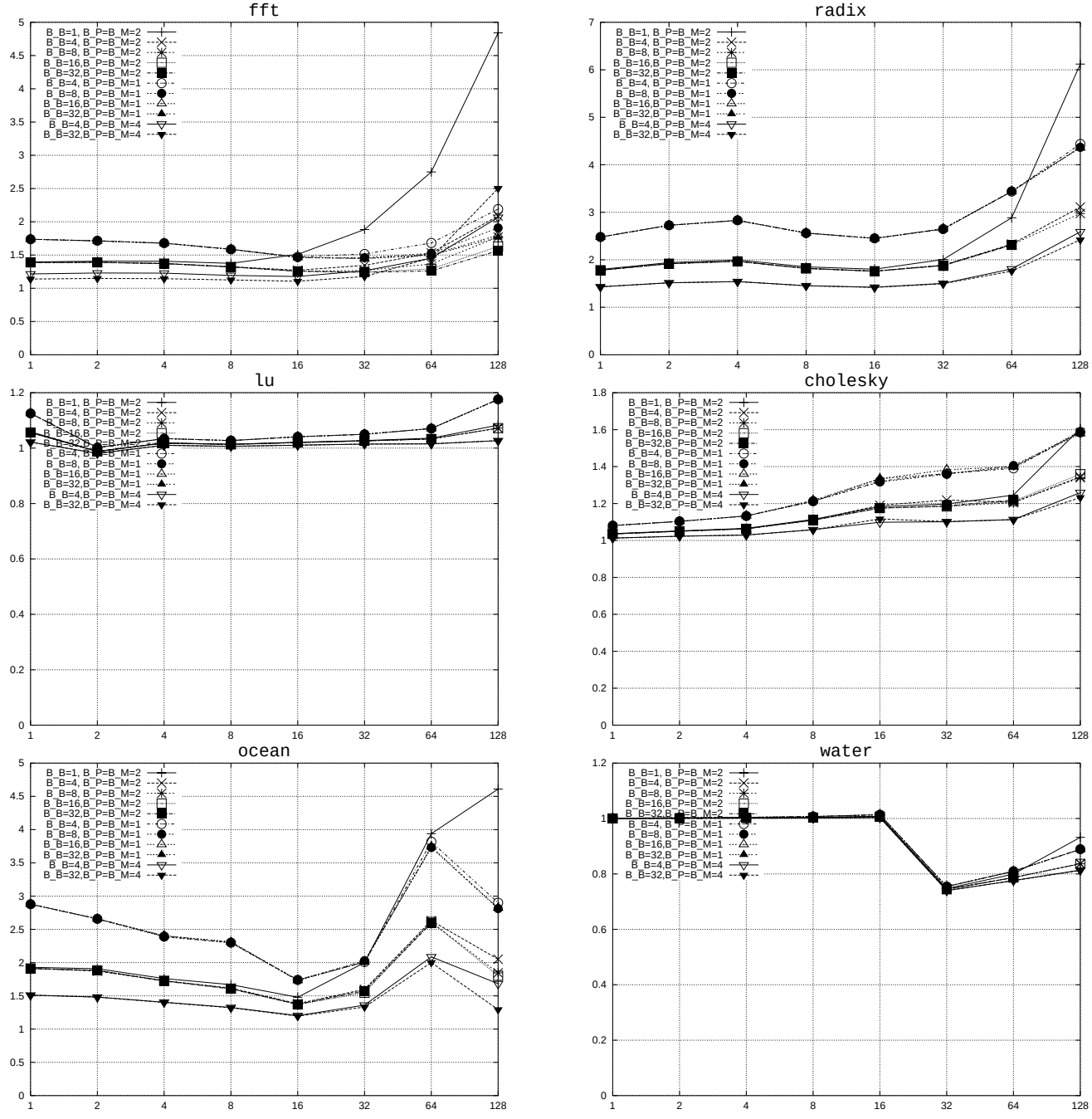


Abbildung 8.13: Die obigen Diagramme zeigen die Ausführungszeit der einzelnen Benchmarks im PHOTOBAR-System im Vergleich zur Ausführungszeit mit einem idealen Speichersystem

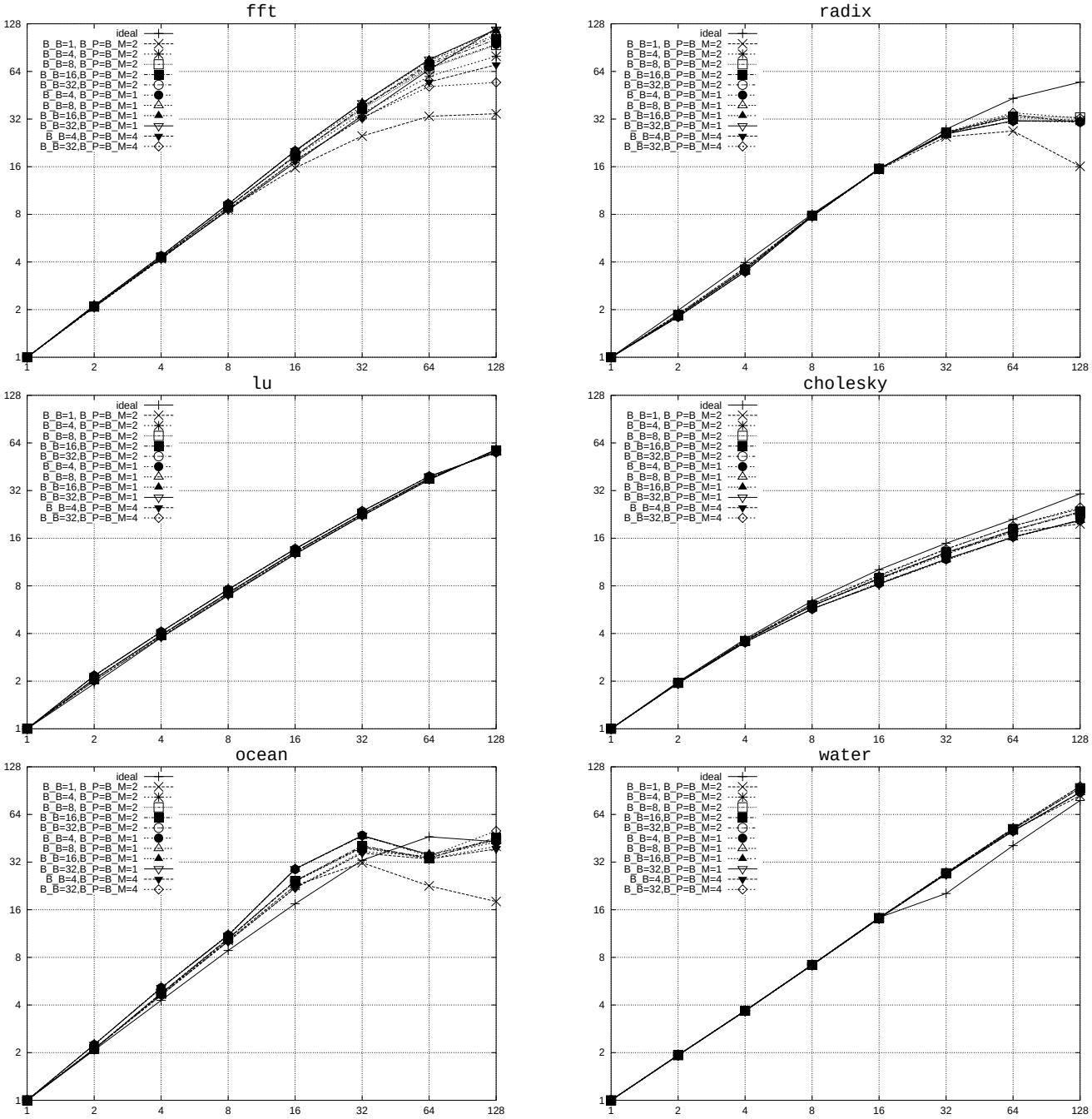


Abbildung 8.14: Die durch Simulation ermittelte Beschleunigung der einzelnen Benchmarkprogramme im PHOTOBARSystem.

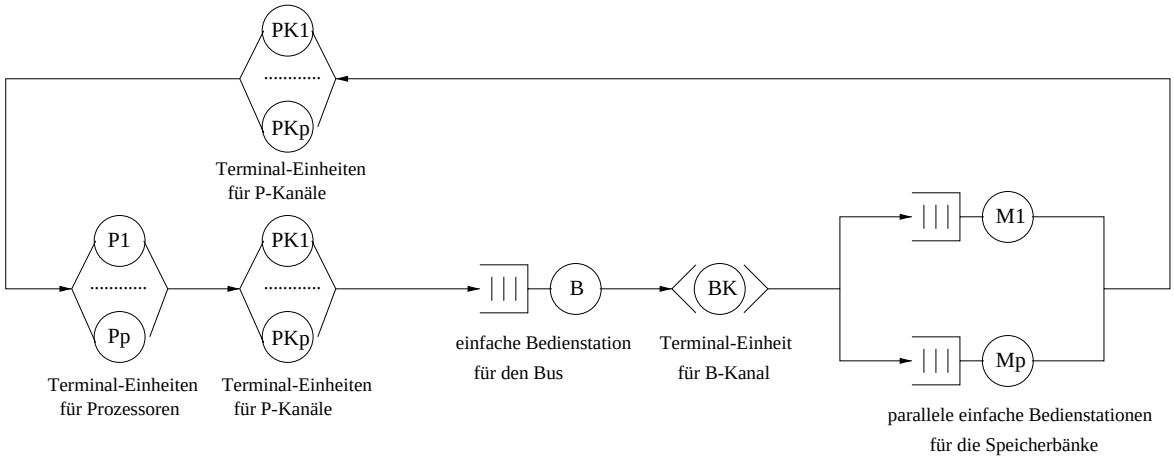


Abbildung 8.15: Das für die theoretische Analyse verwendete Model der PHOTOBUS-Architektur.

(b) Bei der Bandbreite von 2Bytes/cycle zeigen alle Programme eine zufriedenstellende Effizienz. Dabei ist bei den speicherintensiven Anwendungen die Verschlechterung gegenüber dem idealen System in etwa halbiert. Insbesondere für kurzfristig verfügbare, große Systeme, bei denen die Anzahl der Datenleitungen am Chip ein Problem darstellt, erscheint dieser Wert sinnvoll.

(c) Eine weitere Vergrößerung der Bandbreite auf 4Bytes/Cycle führt zu einer Verringerung des Unterschiedes zum idealen System um ein Viertel bis einhalb. Diese Bandbreite ist vor allem für kleinere bzw. mittelfristig verfügbare Systeme geeignet, bei denen die Anzahl der Datenleitungen am Chip nicht so relevant ist.

Als Fazit kann man die folgende Behauptung aufstellen:

Die PHOTOBUS-Architektur erlaubt unter realistischen Annahmen über die Leistungsfähigkeit opto-elektronischer Verbindungen die Realisierung effizienter symmetrischer Multiprozessoren mit bis zu 128 Prozessoren.

8.5.3 Überprüfung durch theoretische Modellierung

Wie auch bei der Betrachtung der PHOTOBUS-Architektur, basiert die theoretische Analyse auf dem in Abschnitt 5.5.2 beschriebenen einfachen, schlangentheoretischen Modell eines paketvermittelnden Bussystems. Zur Modellierung der Last werden wie auch beim PHOTOBUS die in Tabelle 5.3 zusammengefaßten Speicherzugriffshäufigkeiten aus[187] verwendet.

Das Model der PHOTOBAR-Architektur

Wie in Kapitel 5 erläutert, beziehen sich die Werte aus Tabelle 5.3 lediglich auf das Lesen und Schreiben von Daten in den Speicher. Dabei werden weder Synchronisationen, noch WriteBack oder Invalidate-Operationen berücksichtigt. Dies bedeutet, daß bei der Lastmodellierung nur Fetch-Operationen betrachtet werden. Bei solchen Operationen wird zuerst die Adresse über den P- und dann über den B-Kanal übertragen. Danach wird die gesamte Cache-Zeile zunächst über den M- und danach über den B-Kanal übertragen.

Ein Modell, daß einen solchen Nachrichtenfluß nachbilden kann, ist in Abbildung 8.15 zu sehen. Es besitzt Terminal Einheiten zur Darstellung der Prozessoren, der P-Kanäle und der Latenz des B-Kanals. An diesen Komponenten können sich keine Warteschlangen bilden, so daß nur eine konstante Verzögerung stattfindet. Hingegen muß an den Speicherbänken sowie am Transmitter des B-Kanals sehr wohl mit Warteschlangen gerechnet werden. Diese werden daher durch einfache Bedienstationen modelliert. Die Speicherbänke sind in dem Modell der PHOTOBAR-Architektur über P-Kanal Terminal Knoten mit den Prozessorknoten verbunden. Anders als bei der PHOTOBUS-Modellierung gibt es keinen Pfad von den Speicherbänken zurück zum B-Kanal, so daß die Ergebnisse auf dem Rückweg nur eine konstante Verzögerung erleiden. Dadurch wird der Tatsache Rechnung

getragen, daß die Ergebnisse nicht über den gemeinsamen B-Kanal, sondern über die P-Kanäle an die Prozessoren verschickt werden.

Vergleich mit der Simulation

Die Ergebnisse der theoretischen Modellierung sind in Abbildung 8.16 zusammengefaßt. Dabei wurde für einige der bei der Simulation verwendeten Bandbreitenwerte das theoretisch ermittelte Verhältnis der Ausführungszeit zur Ausführungszeit auf einem idealen Speichersystem mit Cache als Funktion der Prozessorzahl aufgetragen. Ein Vergleich mit den Ergebnissen der Simulation in Abbildung 8.13 zeigt, daß:

1. die Form der Kurven in beiden Abbildungen weitgehend übereinstimmen. Insbesondere die starke Abhängigkeit der Effizienz von der Bandbreite der P- und M-Kanäle bei speicherintensiven Anwendungen wird durch die analytische Modellierung bestätigt. Gleiches gilt für die geringe Bedeutung der Bandbreite des B-Kanals.

2. die theoretische Analyse zwar eine höhere Effizienz voraussagt, aber die Größenordnung weitgehend mit der Simulation übereinstimmt.

Damit kann man sagen, daß die theoretische Betrachtung im Rahmen der verwendeten Näherung die Ergebnisse der Simulation untermauert.

8.6 Fazit

Die im vorliegenden Kapitel vorgestellte PHOTOBAR-Architektur nutzt bidirektinonale P- und M-Kanäle zusammen mit einem auf dem SP-Chip integrierten Crossbar-Netzwerk um die Skalierbarkeit und Effizienz des Systems gegenüber den PHOTOBUS-Architektur zu verbessern. Sie ermöglicht dadurch die Implementierung eines effizienten symmetrischen Multiprozessors mit bis zu 128 Prozessoren. Dabei kann sie mit Hilfe kurzfristig bzw. mittelfristig verfügbarer Technologie realisiert werden. Da für die Kommunikation über lange Strecken wie auch bei der PHOTOBUS-Architektur optische Faser benutzt werden, läßt auch die PHOTOBARArchitektur eine effiziente Koppelung von Arbeitsplatzrechnern zu SMP-Architekturen zu.

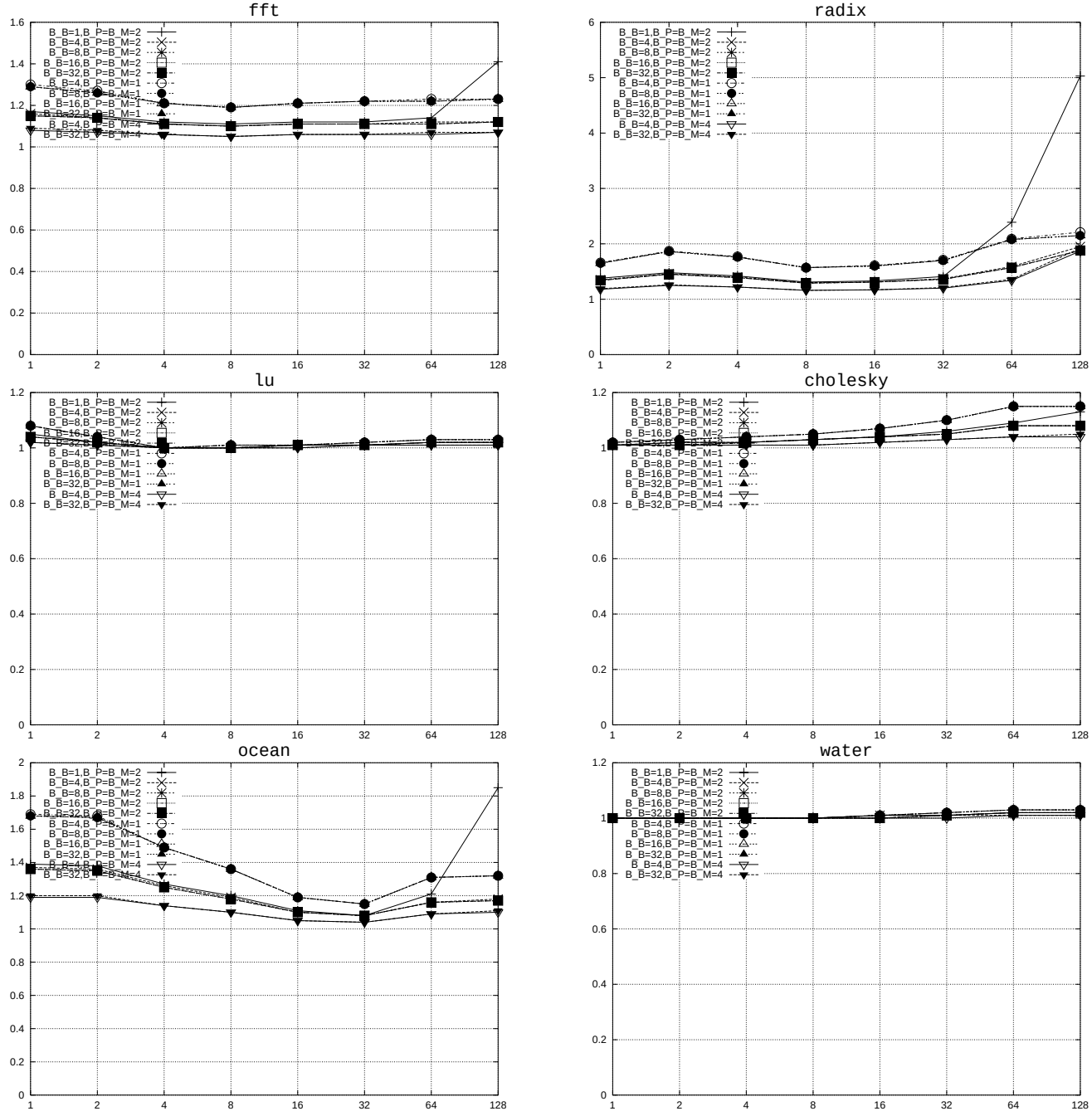


Abbildung 8.16: Die obigen Diagramme zeigen die durch theoretische Modellierung ermittelte Ausführungszeit der einzelnen Benchmarks im PHOTOBAR-System im Vergleich zur Ausführungszeit mit einem idealen Speichersystem.

Kapitel 9

PHOTON

In diesem Kapitel wird gezeigt, wie die drei Haupthindernisse für die Skalierbarkeit der PHOTOBUS und PHOTOBAR-Architekturen beseitigt werden können: der starke Anstieg der Fläche der SP-Chips mit der Prozessorzahl, die Saturierung des B-Kanals und das Cache Update Pressure Problem. Dazu wird ein Architekturkonzept vorgeschlagen, in dem mehrerer PHOTOBUS- bzw. PHOTOBAR-Systeme parallel betrieben und über SP-Bausteine an die Prozessoren angeschlossen werden.

Den wichtigsten Beitrag dieses Kapitels stellt das Verfahren zur Lösung des Cache-Update-Pressure Problems für hohe Prozessorzahlen dar. Das Verfahren basiert auf einer Parallelisierung des Snoop-Vorgangs, die durch eine besondere Verteilung von Adressen auf die Speicherbänke und die Stufen Cache-Herarchie ermöglicht wird. Dem Autor ist kein anderes Verfahren bekannt, das dieses leisten kann. Das Cache-Update Pressure Problem wird vielmehr in der Literatur als eines des Hauptargumente gegen die Skalierbarkeit von Snoopy-Cache-Kohärenz Protokollen angeführt.

Das Grundkonzept (Abschnitt 9.1) und das Parallelisierungsverfahren für den Snoop-Vorgang sind auf eine Vielzahl von Architekturen anwendbar, die auf den PHOTOBUS und PHOTOBAR-Architekturen aufbauen. Wir fassen alle diese Architekturen unter dem Namen PHOTON zusammen. Eine genaue Untersuchung verschiedener, konkreter PHOTON-Architekturen nach dem Muster der beiden vorangegangenen Kapiteln würde den Rahmen dieser Arbeit sprengen. Den Schwerpunkt des Kapitels stellt daher die Erläuterung des Parallelisierungsverfahrens dar (Abschnitt 9.2). Die Beschreibung konkreter Komponenten beschränkt sich auf die für die Parallelisierung des Snoop-Vorgangs notwendigen Voraussetzungen. Wegen der allgemeinen, auf keine konkrete PHOTONVariante spezialisierten Hardwarebeschreibung ist auch keine ausgiebige Leistungsbewertung möglich. Sie ist daher auf zwei Bereiche beschränkt. Zum einen wird die Auswirkung der Nutzung mehrerer SP-Chips auf die benötigte PHOTOBUS- und PHOTOBAR-Chipfläche untersucht. Dabei wird auf der Betrachtung aus den Abschnitten 7.9.8 und 9.3.1 in den vorigen Kapiteln aufgebaut. Zum anderen werden einige Simulationsergebnisse für ein einfaches PHOTOBUS-basiertes System präsentiert (Abschnitt 9.3.2).

Die Untersuchung und Bewertung verschiedener PHOTON-Architekturvarianten ist ein Thema für zukünftige Forschung. Eine bestimmte Variante wurde vom Autor bereits genauer studiert und die Ergebnisse der Studie veröffentlicht [101].

9.1 Überblick

) Weder die PHOTOBUS noch die PHOTOBAR Architektur kann unter realistischen Annahmen bezüglich der Leistungsparmeter der Systemkomponenten mehr als 128-Prozessoren unterstützen. Dies hat drei Gründe:

1. Mit steigender Prozessorzahl wird der B-Kanal zum Flaschenhals. Bei der PHOTOBUS-Architektur stellt dabei vor allem die Bandbreite ein Problem dar. Bei der PHOTOBAR-Architektur ist die Bandbreite weniger kritisch, da lediglich kurze Nachrichten auf dem B-Kanal übertragen werden. Allerdings kann wegen der Latenz des Transmitters unabhängig von der Bandbreite nicht mehr als eine Nachricht pro Taktzyklus übertragen werden. Da bei hohen Prozessorzahlen im Durchschnitt deutlich mehr als eine Nachricht pro Taktzyklus anfallen kann, führt diese Beschränkung zur Saturierung und hohen Wartezeiten.

2. Selbst unter Berücksichtigung der zu erwartenden Fortschritte der VLSI-Technologie scheint es fraglich, ob

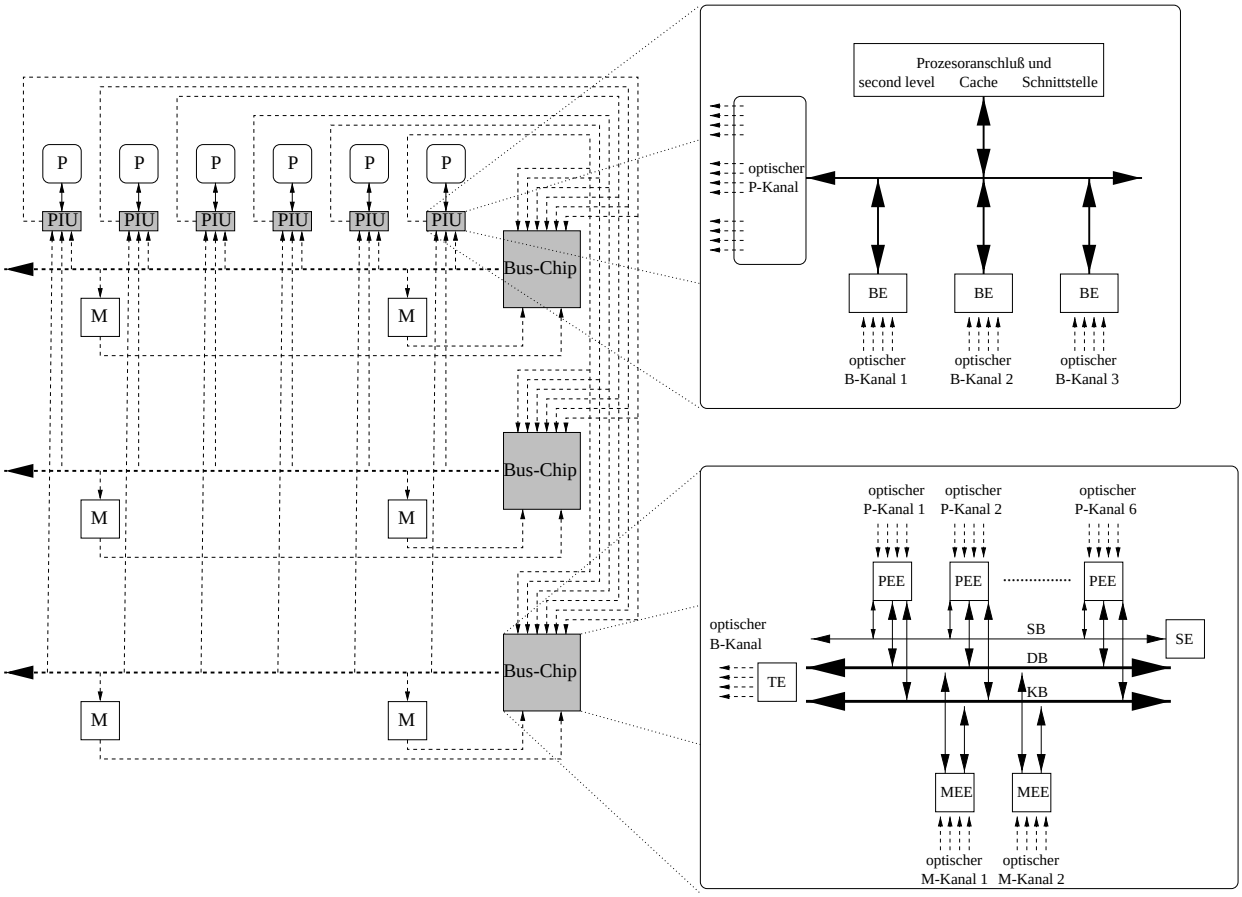


Abbildung 9.1: Das PHOTON-Konzept am Beispiel eines Systems mit 6 Prozessoren, 6 Speicherbänken und 3 PHOTOBUS-Chips.

ein PHOTOBUS- bzw. PHOTOBAR-Chip für mehr als 128 Prozessoren realisiert werden kann. Dies liegt daran, daß die benötigte Fläche wie $P \cdot M^2$ ansteigt.

3. Weder die PHOTOBUS noch die PHOTOBAR-Architektur setzt sich mit dem Problem des Cache Update Pressure auseinander. Wie in Kapitel 3 erläutert, ergibt sich das Problem aus der beschränkten Zugriffsgeschwindigkeit des Caches. Diese Geschwindigkeit bestimmt die maximale Anzahl von Invalidierungen bzw. Supply-Operationen, die ein Prozessor pro Zeiteinheit durchführen kann.

Um die obigen Probleme zu lösen, werden in der PHOTONArchitektur mehrerer PHOTOBUS bzw. PHOTOBAR-Systeme parallel betrieben und über SP-Bausteine an die Prozessoren angeschlossen. Der Ansatz hat eine gewisse Ähnlichkeit mit Mehrbussystemen (Abschnitt 3.4.2), die in verschiedenen kommerziellen Architekturen eingesetzt werden. Der entscheidender Unterschied besteht darin, daß die PHOTON-Architektur die Lösung des Cache-Update Pressure Problems erlaubt. Konventionelle Mehrbusarchitekturen können dies hingegen nicht leisten. Im vorliegenden Abschnitt wird zunächst der Aufbau der PHOTON -Architektur beschrieben. Danach wird die Grundidee vorgestellt, die die Lösung des Cache-Update Pressure Problems ermöglicht.

9.1.1 Aufbau der PHOTON Architektur

Der Aufbau der PHOTONArchitektur ist in Abbildung 9.1 dargestellt. Sie unterscheidet sich von den PHOTOBUS- und PHOTOBAR-Architekturen dadurch, daß sie nicht nur einen, sondern mehrere (N_{BC}) SP-Netzwerkchips besitzt. Die M Speicherbänke werden gleichmäßig auf diese Chips verteilt. Jeder Chip ist also mit einer Untermenge von M/N_{BC} Speicherbänken und allen P Prozessoren verbunden. Abgesehen von der geringeren Anzahl der Speicherbänke ist der Aufbau der Netzwerkchips wie beim PHOTOBUS- bzw. PHOTOBAR-System. Für

jeden Prozessor und jede der M/N_{BC} Speicherbänke besitzt der Chip einen optischen Eingang, der an den entsprechenden P- bzw. M-Kanal angeschlossen ist. Um Nachrichten weiterzuleiten, hat der Chip einen Broadcast-Kanal (B-Kanal) mit dem Fanout $P + M/N_{BC}$. Der B-Kanal ist nur mit den Speicherbänken verbunden, die ihm zugeordnet sind. Die Prozessoren sind an die B-Kanäle über SP-Bausteine, die PIUs (für processor interface unit) angeschlossen. Jede PIU besitzt einen separaten optischen Eingang für den B-Kanal jedes der N_{BC} Bus-Chips, sowie einen optischen Ausgang für den P-Kanal. Der Aufbau der PIU ist dem eines Netzwerkchips sehr ähnlich. Jeder optischer Eingang ist mit einer eigenen Eingabeeinheit versehen, die die Daten von dem zugehörigen B-Kanal verarbeitet. Diese Logik ist über ein elektronisches On-Chip Netzwerk mit dem P-Kanal, einer Prozessorschnittstelle, sowie einer Schnittstelle zum Second Level Cache verbunden.

9.1.2 Lösung des Update Pressure Problems

Da in einer PIU jedem B-Kanal eine eigene Eingabeeinheit zugeordnet ist, kann die PIU alle B-Kanäle gleichzeitig überwachen. Um die Cache-Kohärenz zu gewährleisten, müssen die auf den B-Kanälen übertragenen Daten aber nicht nur überwacht werden. Es ist vielmehr notwendig, bei jeder Schreiboperation nachzuprüfen, ob sich die zugehörige Adresse im eigenen Cache befindet um bei Bedarf den Zustand der entsprechenden Cache-Zeile zu modifizieren. Gleiches gilt für Supply-Anforderungen. Die Einheiten müssen also die Adressen aus dem Cache- bzw. Attributspeicher lesen, und das Ergebnis (im Falle eines Snoop-Hits) wieder dort hineinschreiben. Um das Problem der Cache-Update Pressures zu lösen, muß man sicherstellen, daß diese Cache-Zugriffe nicht zum Flaschenhals werden. Dies ist nur dann gewährleistet, wenn alle Eingabeeinheiten parallel auf den Cache zugreifen können. Das Problem hierbei liegt darin, daß bei einem Multiport-Speicher die Speicherdichte quadratisch mit der Anzahl der Ports abnimmt. Somit liegt die maximale realisierbare Anzahl paralleler Zugriffsports bei 2 bis 4.

Um das Problem zu umgehen, nutzt die PHOTON-Architektur die Tatsache aus, daß jede Adresse eindeutig einer Speicherbank und damit auch einem bestimmten Bus-Chip zugeordnet ist. Dies ergibt sich aus der Verteilung der Speicherbänke auf die Bus-Chips. Es bedeutet, daß Operationen, die ein bestimmtes im Cache befindliches Datum betreffen, immer auf dem gleichen B-Kanal übertragen werden. Insbesondere werden immer Invalidate-Operationen und Supply-Anfragen bezüglich einer in einem Cache befindliche Speicheradresse die PIU über die gleiche Eingabeeinheit erreichen, über die die Daten von der Adresse in den Cache gelesen wurden. D.h., daß die einzelnen Eingabeeinheiten für die Kohärenz disjunkter Untermengen der im Cache befindlichen Daten sorgen müssen. Jede Eingabeeinheit kann sich also in einem privaten Attributspeicher merken, welche Daten über sie in den Cache gekommen sind, und in welchem Zustand sie sich befinden. Damit ist die Voraussetzung für die Parallelisierung des Snoopings gegeben.

9.1.3 Varianten der PHOTON-Architektur

Durch das PHOTON-Konzept werden lediglich einige Eckpunkte der Architektur festgelegt, die die Parallelisierung des Snoop-Vorgangs so wie die Reduzierung der Chipfläche und die Vermeidung der Saturierung des B-Kanals zum Ziel haben. Am wichtigsten sind dabei die folgenden Punkte

1. die Verteilung der Speicherbänke auf mehrere SP-Chips,
2. die Austattung jedes Chips mit seinem eigenen B-Kanal, der an alle Prozessoren angeschlossen ist,
3. die Verwendung von SP-Bausteinen als Prozessorschnitstellen und
4. der Anschluß eines jeden Prozessors an alle SP-Bausteine

Über die obigen Punkte hinaus ist eine Vielzahl von Architekturvarianten möglich. Die wichtigsten unterschiede zwischen diesen Varianten stellen die Art der SP-Chips und die Realisierung der P-Kanäle dar. Bei den SP-Chips kann es sich zum einen um die in der Arbeit beschriebenen PHOTOBUS- und -PHOTOBARChips handeln. Diese können fast unverändert in ein PHOTON-System übernommen werden. Darüberhinaus ist aber noch eine Menge anderer Netzwerkchips denkbar, auf die hier nicht weiter eingegangen wird.

Varianten der P-Kanal Realisierung

Bei den P-Kanälen stellt sich die Frage, wie die Adressierung der einzelnen SP-Chips durchgeführt wird. Eine Möglichkeit besteht darin, die P-Kanäle als Broadcast-and-Select Kanäle auszulegen. D.h. jeder Prozessor hat

nur einen Sender, an den alle Sp-Chips angeschlossen sind. Jeder Chip empfängt alle Nachricht und bestimmen auf der Basis der Adresse welche Nachricht für ihn bestimmt ist. Für Systeme mit wenigen SP-Chips ist dies zweifelsohne die beste Lösung. Bei größeren Systemen ist diese Lösung weniger sinnvoll. Zum einen erfordert ein Rundruf mit hohem Fanout wie in Kapitel 4 und 7 erläutert eine hohe optische Leistung und ist daher mit großen Implementierungsaufwand verbunden. Dieser Aufwand ist nur dann gerechtfertigt, wenn ein Rundruf tatsächlich benötigt wird (wie z.B. beim B-Kanal). Zum anderen kann es bei vielen SP-Chips für die Durchführung von Supply-Operationen sinnvoll sein, gleichzeitig verschiedene Nachrichten an mehrere SP-Chips zu schicken (siehe Abschnitt 9.2.4). Es bietet sich daher an an Stelle eines Rundrufkanals, mehrere Sender zu benutzen, von den jeder für eine Untermenge von SP-Chips verantwortlich ist. Dies wird in Abbildung 9.3 angedeutet, indem die Prozessorschnittstelle mit drei Transmittern dargestellt wurde.

9.2 Parallelisierung der Cache-Kohärenz Operationen

Das oben beschriebene Prinzip der Parallelisierung des Snoop-Vorgangs ist denkbar einfach. Seine Verwirklichung in einem realistischen Cache-System ist dies jedoch nicht. Sie muß verschiedene Aspekte der Cache-Architektur, sowie die Anforderungen des Cache-Kohärenz Protokolls berücksichtigen.

1. Die in den privaten Attributspeichern der Eingabeeinheiten gespeicherte Information ist nur dann nützlich, wenn sie in die Cache-Verwaltung integriert wird. Dies bedeutet zum einen, daß bei jedem Zugriff auf eine Cache-Zeile der Zustand der Daten in dem Attributspeicher der zugehörigen Eingabeeinheit überprüft werden muß. Zum andren muß die Eingabeeinheit benachrichtigt werden, wenn der Cache-Kontroller ein Datum aus dem Cache entfernt.

2. In den meisten Systemen werden, wie in Kapitel 3 beschrieben, mehrere Cache-Stufen verwendet. Bei der Implementierung des Cache-Kohärenz Protokolls müssen alle diese Stufen berücksichtigt werden. Das Problem besteht hierbei daran, daß die die Anforderungen an die Zugriffslatenz von Stufe zu Stufe zunehmen. Die erste Cache-Stufe zeichnet sich in der Regel dadurch aus, daß sie mit Prozessorgeschwindigkeit arbeitet und auf dem Prozessorchip integriert ist. Jeder Zugriff hat eine gravierende Verlangsamung der meisten Anwendungen zur Folge. Dies bedeutet, daß der Prozessor nicht bei jedem Zugriff auf die erste Cache-Stufe in den Eingabeeinheiten der PIU den Zustand der Daten abfragen kann. Die benötigten Invalidierungen müssen also bis zu der ersten Cache-Stufe propagiert werden.

3. Bei der Parallelisierung des Snoop-Vorgangs muß nicht nur die Invalidierung, sondern auch die Durchführung von Supply-Benachrichtigungen und -Operationen berücksichtigt werden. Die strengen Timing-Anforderungen an die Benachrichtigungen wurden bereits mehrmals im Verlauf der Arbeit diskutiert. Die Tatsache, daß Supply-Anforderungen gleichzeitig in mehreren Eingabeeinheiten eintreffen können, wirft die Frage auf, ob und wie diese Anforderungen eingehalten werden können.

Die Lösung der obigen Probleme ist eng mit dem genauem Aufbau des Cache-Systems verbunden, das in das parallel Snooping-Verfahren eingebaut werden soll. Wir werden daher in der nachfolgenden Beschreibung des in der Arbeit entwickelten Verfahrens zur Parallelisierung des Snoop-Vorgangs zuerst einige Annahmen über die Architektur des Cache-Systems zusammenfassen. Danach wird der Aufbau der Prozessorschnittstelle (PIU) erläutert, die für die parallele Durchführung des Snoop-Vorgangs zuständig ist. In Abschnitt 9.2.3 wird dann gezeigt, wie das parallele Snooping-Verfahren auch auf die im Prozessor integrierte höchste Stufe der Cache-Hierarchie erweitert werden kann. Abschließend wird der Ablauf einzelner Speicher und Cache-Operationen skizziert. Dabei wird auch die Frage des Timings der Supply-Benachrichtigungen und -Operationen erörtert.

9.2.1 Annahmen bezüglich des Cache-Systems

Bei der Frage nach der Cache-Architektur müssen drei Faktoren berücksichtigt werden: das verwendetet Cache-Kohärenz Protokoll, die verschiedenen in Kapitel 3.2 beschrieben Aspekte der Cache-Verwaltung sowie der Hardware-Aufbau des Systems. Der erste Punkt ist bereits geklärt worden: es wird, wie auch bei den PHOTOBUS- und PHOTOBAR-Architekturen das Berkeley-Protokoll betrachtet. Bezüglich der Verwaltungsstrategie werden die folgenden Annahmen gemacht:

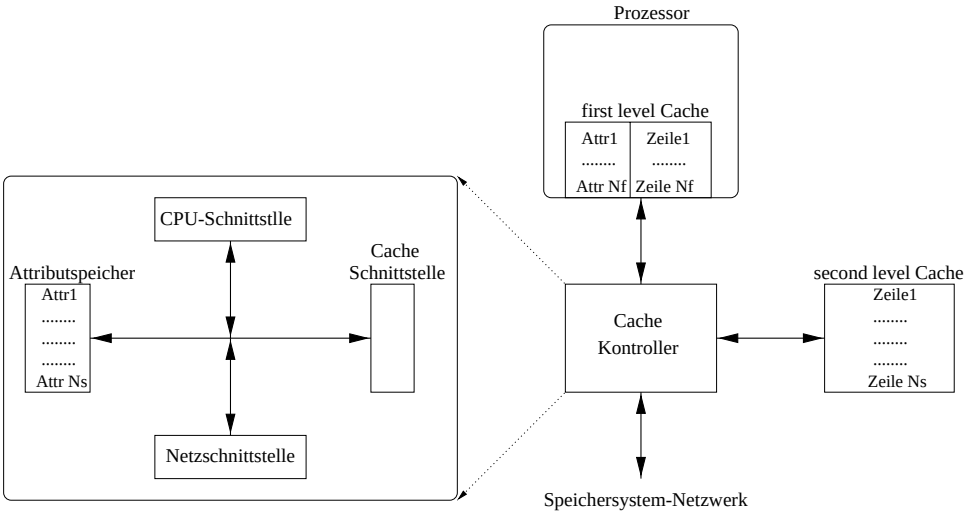


Abbildung 9.2: Der Aufbau eines Cachsystems, der den Ausführungen im vorliegenden Kapitel zugrunde liegt.

1. Es wird ein System mit zwei Cache-Stufen angenommen. Die erste Stufe, First Level Cache (FC), arbeitet mit Prozessorgeschwindigkeit und ist 64 bis 128KByte groß. Die zweite Stufe, Second Level Cache (SC), hat eine Zugriffszeit von einigen wenigen Prozessorzyklen und kann zwischen 0.5 und 4MB Daten fassen.

2. Wir gehen davon aus, daß beide Caches set assoziativ sind. Wie in Kapitel 3 erklärt, bedeutet dies, daß die Cache-Zeilen in Gruppen unterteilt sind, die jeweils bestimmten Bereichen des Adreßraumes zugeordnet sind.

3. Es wird angenommen, daß der SC und der FC dieselbe Cache-Organisation haben und der SC inklusiv ist. Es sind also alle im FC-enthaltenen Daten auch im SC vorhanden

Der betrachtete Hardwereaufbau ist in Abbildung 9.2 schematisch dargestellt. Der FC ist vollständig im Prozessor integriert. Dies bedeutet, daß sowohl die Cache-Zeilen als auch die Attribute und die Kontrollogik Bestandteil des Prozessors sind. Der SC ist vom Prozessor unabhängig. Er wird von einem, ebenfalls vom Prozessor unabhängigen, Cache-Kontroller (CC) verwaltet. Der CC bildet die Schnittstelle zwischen dem Prozessor, dem SC und dem Hauptspeichersystem. Bei der hier präsentierten Betrachtung gehen wird davon aus, daß alle Zugriffe auf den SC über den CC gehen. Aus diesem Grund wird die Information über den Zustand der Cache-Zeilen (die Attribute der Zeilen) im Cache-Kontroller gespeichert. In den meisten heutigen Systemen befinden sich der Prozessor, der SC und der CC auf unterschiedlichen VLSI-Bausteinen. Sie sind über kurze, bis zu 256 Bit Breite elektronische Leitungen auf der Platine oder auf einem Multichipmodul verbunden.

9.2.2 Die PIU

Für die parallele Durchführung des Snoop-Vorgangs ist die PIU zuständig, die im System anstelle des Cache-Kontrollers eingesetzt wird. Sie muß die folgenden Aufgaben des Cache-Kontrollers und der Prozessorschnittstelle eines PHOTOBUS bzw. PHOTOBARSystems gleichzeitig erfüllen:

1. Bei Speicheranfragen des Prozessors muß geprüft werden, ob sie aus dem SC befriedigt werden können. Ist dies der Fall, so müssen die benötigten Daten aus dem SC geholt und an den Prozessor zurückgegeben werden. Anderenfalls muß über den P-Kanal eine Anfrage an den entsprechenden SP-Chip geschickt werden.

2. Wenn das Ergebnis einer Prozessoranfrage über einen der B-Kanäle eintrifft, dann muß es an den Prozessor weitergeleitet werden. Außerdem muß die entsprechende Zeile bei Bedarf im SC aktualisiert werden. Bei Fetch-Operationen müssen dazu die Daten im SC gespeichert werden. Bei Exclusiv-Operationen muß der Zustand der Zeile aktualisiert werden.

3. Bei Invalidate- bzw. Exclusiv-, und FetchWrite- Operationen, die über irgendeinen der B-Kanäle ankommen, muß geprüft werden,ob sich das betreffenden Datum um Cache befindet. Ist dies der Fall so muß es als ungültig markiert werden. Dabei müssen sowohl der SC als auch der FC berücksichtigt werden.

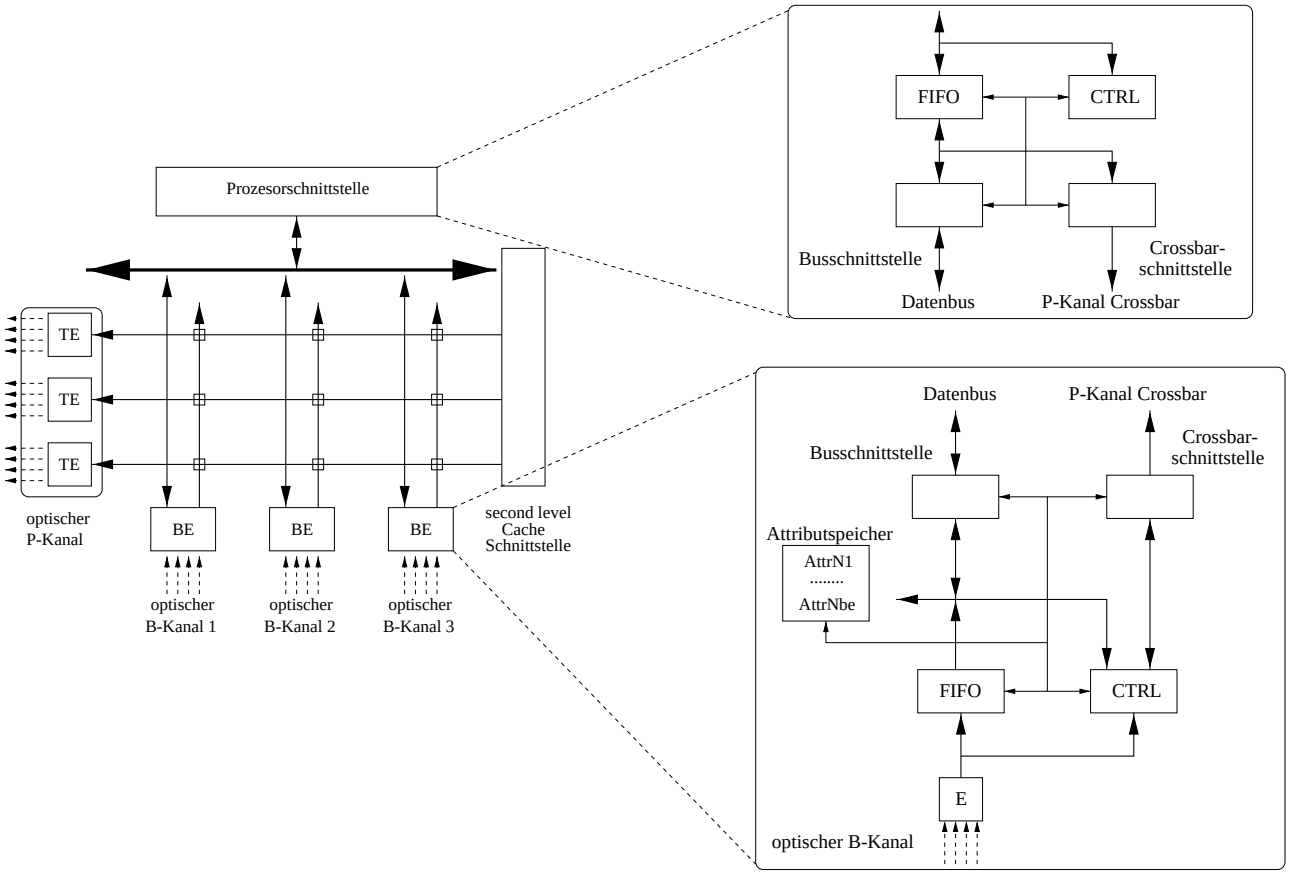


Abbildung 9.3: Der Aufbau der PIU und seiner beiden wichtigsten Komponenten der B-Kanal Eingabeeinheit (BE) und der Prozessorschnittstelle.

4. Wird auf irgendeinem B-Kanal eine Fetch-Operation übertragen, so muß überprüft werden, ob sich die Daten im Exclusiv-Zustand im eigenen Cache befinden. Ist dies der Fall, so muß sofort eine Supply-Benachrichtigung über den P-Kanal an den zuständigen SP-Chip geschickt werden. Später müssen die Daten aus dem SC geholt und ebenfalls an den Chip übertragen werden.

Grobaufbau

Die PIU ist genauso wie der Cache-Kontroller an den Prozessor, an den SC und an das Speichersystem angeschlossen. Dabei besteht der Anschluß an das Speichersystem aus den Eingängen der B-Kanäle und dem Ausgang des P-Kanals (siehe Abbildung 9.1). Der letzere besteht in der Regel aus mehreren Transmittern, von den jeder per Rundruf eine bestimmte Untermenge der SP-Chips ansprechen kann. Der wichtigste Unterschied zum konventionellen Cache-Kontroller besteht darin, daß der Attribut-Speicher nicht als eigenständige Einheit vorhanden ist. Er ist statt dessen in N_{BC} Teile aufgeteilt, von denen jeder der Eingabeeinheit eines anderen B-Kanals zugeordnet ist.

Die Struktur der PIU ist in Abbildung 9.3 für ein System mit drei B-Kanälen und drei getrennten P-Kanal Trannsmittern zu sehen. Sie besitzt eine Prozessorschnittstelle, eine Schnittstelle zum SC, jeweils eine Eingabeeinheit für jeden optischen B-Kanal (nachfolgend als BE bezeichnet) und die Transmitter für den P-Kanal. Die BEs sind mit den beiden Schnittstellen wie beim konventionellen Cache-Kontroller über eine Busleitung verbunden. Hinzu kommt ein Crossbar-Netzwerk, über das die BEs und die Cache-Schnittstelle Daten an die einzelnen Transmitter des P-Kanals schicken können. Dieses Netzwerk ist notwendig, damit gleichzeitg Daten an verschiedene Transmitter geschickt werden können. Dies ist wie in Abschnitt 9.2.4 erläutert wird für die Durchführung von Supply-Benachrichtigungen notwendig.

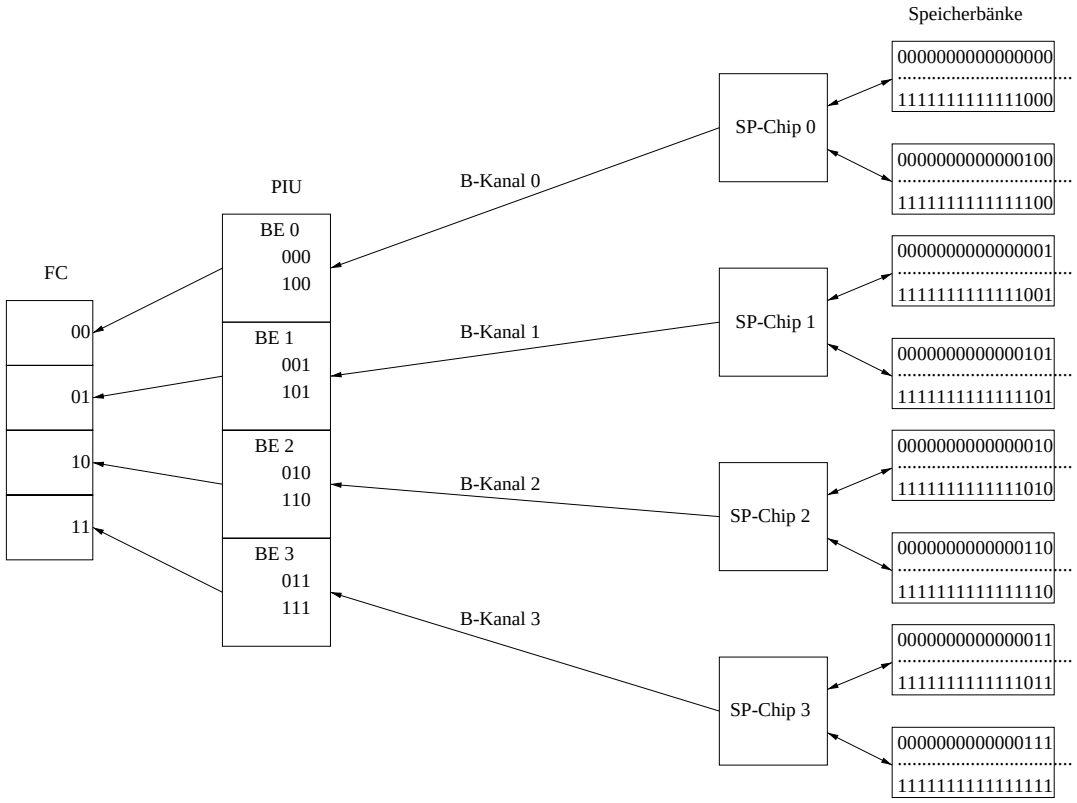


Abbildung 9.4: Beispiel für eine Zuordnung von Adressen zu Speicherbänken, SC- und FC-Cacheline Gruppen, die die Parallelisierung des Snoop-Vorgangs ermöglicht. Die Abbildung geht von 4 FC-Cacheline Gruppen ($z_F = 2$), 8 SC Gruppen ($z_S = 3$) und vier SP-Chips (also vier B-Kanälen und damit $N_{BC} = 4$). Es werden also vier FC-Bereiche und 4 BEs benötigt. Jeder FC-Bereich beinhaltet lediglich eine Cacheline-Gruppe, während jede BE die Attribute von zwei SC Cahcheline-Gruppen verwalten muß. Die Zahlen in den FC-Bereichen geben die Werte der beiden untersten Bits der Adressen an, die zugehörigen Cacheline-Gruppen zugeordnet sind. Analog sind durch die Zahlen in den BEs die Werte der 3 niederwertigsten Bits angegeben der Adressen angegeben, die in die zugehörigen SC-Cacheline-Gruppen geladen werden können.

Die BE

Die BEs bestehen, wie in Abbildung 9.3 zu sehen, aus einem Empfänger, der die vom B-Kanal ankommenden Daten an einen FIFO leitet, einer Kontrolleinhiet, den Schnittstellen für die beiden On-Chip-Netzwerke und dem privaten Attributspeicher. Dieser ist als Speichermodul mit zwei Zugriffsports angelegt. Der eine ist mit dem FIFO, der anderen mit dem On-Chip Bus verbunden. Über den, mit der FIFO-verbundenen Port, kann die BE die Snoop-Operationen auf ihrem privaten Attributspeicher durchführen. Somit können die BEs parallel die benötigten Operationen auf den ihnen zugeteilten Teilen des Attributspeichers durchführen. Gleichzeitig kann der Prozessor mit Hilfe des zweiten Zugriffsports über den Bus auf den Attributspeicher jeder BE zugreifen. Er kann also wie bei einem konventionellen Cache-Kontroller jeder Zeit beliebige Attribute lesen und schreiben.

Verteilung des Attributspeichers auf die BEs

Die Aufteilung des Attributspeichers auf die BEs wirft die Frage der für die einzelnen Teile benötigten Speicherkapazität auf. Die Verteilung der Speicherbänke auf die Netzchips sorgt dafür, daß alle eine bestimmte Adresse betreffenden Nachrichten in der PIU in der gleichen BE eintreffen. Eine einfache Verteilung hat aber keinen Einfluß darauf, wieviele Daten über welche BE in den Cache gelangen. So könnte es theoretisch sein, daß alle zu einem bestimmten Zeitpunkt im Cache vorhandenen Daten über den gleiche BE gekommen sind. Dies bedeutet, daß jede BE nicht einen Teil des Attributspeichers, sonder einen privaten Attributspeicher der vollen Größe benötigen würden. Bei vielen B-Kanälen und großen Caches würde dies zu Platzproblemen auf dem PIU-Chip führen. In der vorliegenden Arbeit wurde ein Konzept erarbeitet, das dieses Problem durch eine geschickte Zuordnung von Speicheradressen zu Speicherbänken löst. Diese Zuordnung stellt sicher, daß Daten, die die PIU über

einen bestimmten B-Kanal erreichen, immer nur in einer bestimmten Untermenge von Cache-Zeilen gespeichert werden. Daraus folgt, daß jede BE nur für eine Untermenge von Cache-Zeilen zuständig ist. Sie braucht somit nur eine Untermenge von Attributen zu speichern. Sie benötigt also, wie gefordert, nur einen Teil, und keine vollständige Kopie des Attributpeichers.

Das Verfahren nutzt die Tatsache aus, daß bei einem set assoziativen Cache bestimmte Adressen nur in bestimmten Cache-Zeilen gespeichert werden können. Wie in Abschnitt 3.2.1 erklärt, werden in einem set assoziativen Cache die Cache-Zeilen in Gruppen eingeteilt. Jeder dieser Gruppen ist eindeutig einem bestimmten Speicherbereich zugeordnet. Daten, die aus einem Speicherbereich stammen, werden immer in einer Zeile der diesem Speicherbereich zugeordneten Gruppe geladen. Für die Zuordnung werden die z niederwertigen Bits der Speicheradresse benutzt wobei 2^z der Anzahl der Cache-Zeilen-Gruppen entspricht. In die Cache-Zeilen-Gruppe i können demnach Adressen geladen werden, deren z niederwertigen Bits als Binärzahl den Wert i haben. Aus der obigen Zuordnung von Adreßbereichen zu Cache-Zeilen kann leicht eine Zuordnung von B-Kanälen zu Cache-Zeilen abgeleitet werden. Dazu muß man lediglich alle, einer bestimmten Cache-Zeilen-Gruppe zugeordneten Speicheradressen in den Speicherbänken desselben B-Kanals unterbringen (siehe Abbildung 9.4). Unter der Annahmen, daß für die Zuordnung der Adressen zu Cache-Zeilen-Gruppen z Bits benutzt werden, bedeutet dies, daß Adressen mit gleichen z niederwertigen Bits alle den Speicherbänken des gleichen B-Kanals zugewiesen werden müssen. Unter der Annahme, daß die Anzahl der B-Kanäle N_{BC} eine Zweierpotenz und kleiner als als 2^z ist, können die Adressen dabei trivialer Weise gleichmäßig auf die B-Kanäle verteilt werden. Dadurch wird sichergestellt, daß jeder BE gleich viele Cache-Zeilen-Gruppen zugeordnet werden.

Die oben skizzierte Aufteilung des Attributspeichers auf die BEs und die dazu notwendige Adreßverteilung auf die Speicherbänke der B-Kanäle ist in Abbildung 9.4 veranschaulicht. Eine formale Spezifikation sieht wie folgt aus:

2^{z_S}: die Anzahl der Cache-Zeilen-Gruppen im SC,

g_k^S: die k-te Cache-Zeilen-Gruppe des SC,

$\mathcal{G}_k^S$: die Menge aller Adressen, die in die Cache-Zeilen-Gruppe g_k des SC geladen werden können,

$\mathcal{B}_i$: die Menge aller Adressen, die sich in den, dem B-Kanal i zugeordneten Speicherbänken befinden und

$\mathcal{BE}_i$: die Menge aller Cache-Zeilen-Gruppen des SC, dessen Attribute in dem privaten Attributspeicher der BE i gespeichert sind.

Damit muß gelten:

$$\forall adr \in \mathcal{G}_k^S : \quad adr\&(2^{z_s+1} \Leftrightarrow 1) = k \tag{9.1}$$

$$\forall adr_1, adr_2 \in \mathcal{G}_k^S : \quad adr_1 \in \mathcal{B}_i \Leftrightarrow adr_2 \in \mathcal{B}_i \tag{9.2}$$

$$\forall adr, k, i : \quad adr \in \mathcal{G}_k^S \wedge adr \in \mathcal{B}_i \Rightarrow g_k^S \in \mathcal{BE}_i \tag{9.3}$$

Das ' &' Symbol in der ersten Gleichung bedeutet eine bitweise UND-Operation.
Eine wichtige Eigenschaft der obigen Spezifikation ist, daß sie lediglich einige Eigenschaften der Verteilung von Cache-Zeilen-Gruppen und Adressen auf die B-Kanäle bestimmt. Diese Eigenschaften können von einer Vielzahl verschiedener Zuordnungen erfüllt werden. Dies ist, wie später beschrieben wird, für die Handhabung des FC von Bedeutung.

Abwicklung der Prozessoranfragen

Bei Ankunft einer Prozessoranfrage überprüft die Kontrollogik der Prozessorschnittstelle zunächst, um was für eine Art von Anfrage es sich handelt. Synchronisations-, Exlusiv- und Invalidate-Anfragen werden über das Crossbar zum entsprechenden P-Kanal-Transmitter und von dort an den zuständigen SP-Chip weitergeleitet. Bei anderen Anfragen wird die Adresse zunächst über den Datenbus an die für diese Adresse zuständige BEs gegeben. Die BE überprüft den Zustand des Datums im SC und teilt das Ergebnis der Prozessorschnittstelle mit. Diese leitet die Anfrage dann entweder über den Datenbus an die SC weiter oder übergibt sie über das Crossbar-Netzwerk an den zuständigen P-Kanal-Transmitter.

Sobald das Ergebnis einer Operation über den B-Kanal ankommt, wird es von der zuständigen BE über den Datenbus an die Prozessorschnittstelle weitergegeben, die sie an den Prozessor leitet. Bei einer Fetch-Operationen

werden die Daten außerdem in den SC gespeichert. Dabei wird die Nummer der Cache-Zeile, in die die Daten hineingeschrieben wurden, der BE über den Bus mitgeteilt. Gleiches gilt für die Nummer der FC-Cache-Zeile, in die die Daten eingelesen wurden. Somit kann die BE ihren privaten Attributspeicher aktuell halten.

Der Parallele Snoop-Vorgang und die Verwaltung des SC

Unabhängig von der oben beschriebenen Bearbeitung der Prozessoranfrage müssen die BEs permanent die B-Kanäle überwachen. Stellt die BE an einem Kanal eine Ivalidate-, Exclusiv-Operation fest, so überprüft sie zunächst in ihrem Teil des Attributspeichers, ob die betroffene Adresse dort eingetragen ist. Ist dies der Fall, so wird der im entsprechenden Attribut gespeicherte Zustand modifiziert. Diese Veränderung muß bei Bedarf auch zum FC propagiert werden. Wie dies gelöst werden kann, wird im nächsten Abschnitt beschrieben.

Wird über den B-Kanal eine Fetch-Anfrage übertragen, so wird zunächst ebenfalls im privaten Attributspeicher überprüft, ob, und in welchem Zustand sich die gewünschten Daten im SC befinden. Falls sich die Daten im SC im Exclusiv-Zustand befinden, dann wird im Attributspeicher vermerkt, daß eine Supply-Operation stattfinden muß. Der Ablauf einer solchen Operation wird in Abschnitt 9.2.4 beschrieben. Falls sich die Daten im Speicher in einem anderen Zustand befinden, so wird nur dann etwas unternommen, wenn es sich bei der Fetch-Operation um ein FetchWrite handelt. In diesem Fall wird genauso wie bei Invalidate- und Exlusiv-Operationen vorgegangen.

9.2.3 Handhabung des FC

Wie am Anfang des vorliegenden Abschnittes angesprochen, besteht das Problem bei der Handhabung des FC darin, daß dieser mit dem Prozessor integriert ist und mit Prozessorgeschwindigkeit betrieben wird. Daraus folgt, daß der Zustand des benötigten Datums nicht vor dem Zugriff im Attributspeicher der PIU überprüft werden kann. Dies würde den Geschwindigkeitsvorteil zunichte machen, der durch die Integration mit dem Prozessor erreicht wird. Es muß daher sichergestellt werden, daß die im FC enthaltenen Attribute immer aktuell sind. Dazu müssen alle Invalidierungen von der PIU an den FC weitergegeben und in dem Attributspeicher des FC vermerkt werden. Die Schwierigkeit hierbei besteht darin, daß die Bandbreite der Verbindung zwischen der PIU und dem Prozessor in der Regel deutlich geringer ist, als die kommulierte Bandbreite der B-Kanäle. Gleiches gilt für die Bandbreite, mit der der Attributspeicher des FC beschrieben werden kann. Das Cache-Update Pressure Problem, das für den SC durch die Parallelisierung des Snoop-Vorgans gelöst wurde, taucht also an dieser Stelle im Zusammenhang mit dem FC wieder auf.

In der Arbeit wurde ein Verfahren entwickelt, daß auch diese Version des Cache-Update Pressure Problems durch eine Parallelisierung löst. Es basiert auf zwei Beobachtungen:

1. Der FC kann genauso wie der SC in disjunkte Bereiche eingeteilt werden, in denen sich nur Daten befinden, die jeweils über einen bestimmten B-Kanal in den Cache gelangt sind. Die BEs können dadurch unabhängig voneinander und parallel die Invalidierungen an die entsprechenden Teile des FC weiterleiten. Dazu benötigen die einzelnen Bereiche lediglich eigenständige, mit den BEs verbundene Zugriffsports. Dabei wird das gleiche Prinzip angewendet, wie bei der Verteilung des Attributspeichers des SC auf die BEs.

2. Wenn sich die BEs merken, welche Daten in welcher Cache-Zeile des FC gespeichert werden, dann können bei Invalidierungen die FC-Cache-Zeilennummer anstelle von Speicheradressen an den FC geschickt werden. Dabei werden nicht mal die absoluten Zeilennummern benötigt. Es reicht vielmehr, wenn die Position des Datums innerhalb des, einer BE zugeordneten Bereiches angegeben wird. Dies verringert die für die Übertragung der Invalidierungen zwischen der PIU und dem Prozessor benötigte Bandbreite um einen Faktor von 4 bis 8.

Bei der nachfolgenden Beschreibung des Verfahrens wird zunächst erläutert, wie sich die obigen Beobachtungen aus den in 9.2.1 beschriebenen Annahmen bezüglich des Cache-Systems ergeben. Danach wird die parallele Durchführung der Invalidierungen beschrieben.

Zuordnung von FC Bereichen zu B-Kanälen

Wie in 9.2.1 beschrieben, gehen wir von der Annahme aus, daß der FC genauso wie der SC set assoziativ und im SC enthalten ist. Wegen der set Assoziativität sind die Cache-Zeilen des FC wie beim SC in Gruppen eingeteilt, die bestimmten disjunkten Adreßbereichen zugeordnet sind. Die Zuordnung erfolgt wie gehabt auf der

Basis der niederwertigen Bits der Adresse, wobei die Anzahl der für die Zuordnung verwendeten Bits von dem Binärlogarithmus der Anzahl der Gruppen abhängt. Die gewünschte Beziehung zwischen Bereichen des FC und den B-Kanälen kann also durch eine entsprechende Zuweisung von Adressen zu Speicherbänken hergestellt werden. Die Vorgehensweise ist dabei die gleiche wie beim SC. Allerdings müssen zwei zusätzliche Randbedingungen berücksichtigt werden.

1. Die Bereiche, in die der FC eingeteilt wird, sollen zusammenhängend sein. D.h., die Cache-Zeilen-Gruppen, die einem bestimmten B-Kanal zugeordnet werden, sollen fortlaufende Nummern haben. Es ist also, anders als bei der Verteilung der Attribute auf die BEs, nicht egal, welche Cache-Zeilen-Gruppe welchem B-Kanal zugeordnet wird. Diese Bedingung ergibt sich daraus, daß der FC anders als die Attribute des SC nicht physikalisch aufgeteilt wird. Er wird statt dessen in Bereiche eingeteilt, von denen jeder einen eigenen Zugriffsport bekommt. Die Zugriffsports der einzelnen Bereiche sollen parallel zu dem Zugriffsport existieren, über den der Prozessor auf den gesamten FC zugreift. Wären die Bereiche nicht physikalisch zusammenhängend, so würde die Realisierung der zusätzlichen Zugriffsports eine komplexe Verbindungsstruktur mit sich bringen, die die Dichte (und damit die Größe) des Caches reduzieren würde. Würde man die Cache-Zeilen-Gruppen dagegen umsortieren, so würde der Zugriffsmechanismus des Prozessors komplexer und damit langsamer werden. Beides würde die Leistung des Prozessors erheblich beeinträchtigen.

2. Durch die Verteilung der Attribute des SC auf die BEs sind bereits, wie in Abschnitt 9.2.2, bestimmte Anforderungen an die Zuordnung von Adressen zu Speicherbänken gegeben. Die Adreßverteilung, die sich aus der Zuordnung von FC-Bereichen zu B-Kanälen ergibt, muß mit diesen Anforderungen kompatibel sein.

Die erste Randbedingung allein kann sehr einfach befriedigt werden. Man muß lediglich sicherstellen, daß die einem B-Kanal zugeordnete Adressen fortlaufende Werte der relevanten niederwertigen Bits besitzen. Die zweite Forderung entpuppt sich bei näheren Hinschauen als trivial. Sie ist allein durch die Tatsache erfüllt, daß der FC kleiner als der SC ist. Dies kann wie folgt begründet werden:

1. Damit eine Verteilung der Attribute des SC auf die BEs möglich ist, ist lediglich eins notwendig: es müssen alle zu einer Cache-Zeilen-Gruppe des SC gehörenden Adressen demselben M-Kanal zugeordnet werden. Dies bedeutet, wie beschrieben, daß alle Adressen, bei denen die niederwertigen z_S Bits einen bestimmten Wert haben, demselben B-Kanal zugeordnet werden.

2. Für die Unterteilung des FC ist es, wie oben erläutert, notwendig, daß sich alle einer Cache-Zeilen-Gruppe des FC zugeordneten Adressen in den Speicherbänken eines B-Kanals befinden. Unter der Annahme, daß der FC 2^{z_F} Zeilen besitzt, heißt das, daß die Adressen bei denen die z_F niederwertigen Bits einen bestimmten Wert haben, alle dem gleichen B-Kanal zugeordnet werden.

3. Eine Inkompatibilität wäre dann gegeben, wenn aus Punkt 2 folgen würde, daß Punkt 1 nicht erfüllt werden kann. Es müßte also aus der Zuordnung der fortlaufender Cache-Zeilen-Gruppen des SC zu den B-Kanälen folgen, daß die zu einer und derselben SC Cache-Zeilen-Gruppe gehörenden Adressen auf zwei verschiedene B-Kanäle verteilt werden. Die Festlegung von Adressen zu B-Kanälen aufgrund der z_F niederwertigen Bits müßte also dazu führen, daß die Adressen mit einem bestimmten Wert der z_S niederwertige Bits zwangsläufig über mehrere B-Kanäle verteilt werden.

4. Da der FC kleiner als der SC ist, gilt: $z_F < z_S$. Legt man also durch die unteren z_F Bits der Adressen die Zuordnung von FC Cache-Zeilen-Gruppen zu B-Kanälen fest, so legt man auch automatisch die Zuordnung einer ganzen Menge von SC Cache-Zeilen-Gruppen fest. Um welche Zeilen es sich handelt, bestimmen die Bits z_F bis $z_S \Leftrightarrow 1$. Das Wichtige ist, daß alle Adressen dieser Cache-Zeilen-Gruppen des SC dann diesem einen B-Kanal zugeordnet werden, da sie alle die gleichen untersten z_F Bits haben. Aus einer bestimmten Zuordnung der Cache-Zeilen-Gruppen des FC kann also unter keinen Umständen folgen, daß Adressen einer SC Cache-Zeilen-Gruppe zwei verschiedenen B-Kanäle zugeordnet werden.

Die Unterteilung des FC in Bereiche, die einzelnen B-Kanäle zugeordnet sind, ist in Abbildung 9.4 zu sehen. Dabei wird auch der zugehörige Zusammenhang zwischen den Cache-Zeilen-Gruppen des FC und SC verdeutlicht.

Für FC-Invalidierungen benötigte Bandbreite

Zur Übertragung einer Hauptspeicheradresse werden in der Regel 64 Bit benötigt. Die Länge des zur Angabe der Position eines Datum innerhalb eines Bereiches des FC benötigten Adreßwortes l_{BE} hängt von zwei Faktoren ab:

der Anzahl der Zeilen im FC (N_f) und der Anzahl der B-Kanäle N_{BC}. Es gilt:

$$l_{BE} = ld(N_f/N_{BC}) \tag{9.4}$$

Gehen wir von einem 64KByte großen FC mit 64Byte langen Zeilen aus, so beträgt die Anzahl der Zeilen im FC:

$$N_f = \frac{64 \cdot 1024}{64} = 1024 \tag{9.5}$$

In einem System mit 8 B-Kanälen müssen also lediglich $ld(128) = 7$, bei 32 B-Kanälen nur $ld(32) = 5$ Bits übertragen werden.

Parallelisierung der Invalidierungen im FC

Um die Parallelisierung der FC-Invalidierungen zu realisieren, sind in dem Cache-System folgende Erweiterungen nötig:

1. In dem auf die BEs verteilten Attributspeicher des SC muß dem Eintrag einer jeden Cache-Zeile ein zusätzliches Feld hinzugefügt werden. In diesem Feld wird vermerkt, ob sich der Inhalt dieser Zeile im FC befindet und wenn ja, in welcher Zeile des entsprechenden Bereiches des FC er zu finden ist.

2. Die Datenleitungen der Verbindung zwischen der PIU in dem Prozessor müssen in N_{BC} Gruppen eingeteilt werden. Jede dieser Gruppe dient der Übertragung von Invalidierungen zwischen einer bestimmten BE und dem ihr zugeordneten Teil des FC.

3. Die Verbindung zwischen der PIU und dem Prozessor benötigt eine zusätzliche Kontrolleitung, die die Übertragung von Invalidierungen zwischen den BE und dem FC anzeigt.

4. Jeder Bereiche des FC muß mit einem zusätzlichen Eingang versehen werden, der Zugriff auf die Attribute der einzelnen Cache-Zeilen erlaubt. Dieser wird über die entsprechende Leitungsgruppe der dem Bereich zugeordneten BE verbunden.

Jedes Mal, wenn ein Datum in den FC geladen wird, speichert die FC in dem zusätzlichen Feld des Attributspeichers die Nummer der Zielzeile in dem ihr zugeordneten Bereich des FC. Kommt nun über den B-Kanal eine Invalidierung, die dieses Datum betrifft, so wird diese über die der BE zugeordneten Leitungsgruppe an den Eingang des FC-Bereiches geschickt. Da jede BE über eine eigene Leitungsgruppe verfügt und mit einem anderen Bereich des FC verbunden ist, können bei Bedarf alle BEs gleichzeitig Invalidierungen an den FC schicken. Sollte zu dem Zeitpunkt, als eine die Invalidierung in der BE eintrifft, bereits eine andere Datenübertragung (z.B. die Übertragung des Ergebnisses einer Fetch-Operation) stattfinden, so kann die BE diese Übertragung durch Setzen der INV-Leitung vorübergehend für die Dauer der Invalidierung unterbrechen.

9.2.4 Supply Operationen

Die Frage der Supply-Operationen wurde bereits ausgiebig im Zusammenhang mit den PHOTOBUS und PHOTOBAR-Architekturen erläutert. Das Problem bei der Durchführung solcher Operationen besteht darin, daß die SP-Chips Fetch-Anfragen verschicken, ohne darauf zu achten, ob die Prozessoren genug Zeit hatten, um die notwendigen Supply-Benachrichtigungen und -Operationen durchzuführen. Hinzu kommt, daß an die Ankunft der Benachrichtigungen strenge Timing Anforderungen gestellt werden. Sie müssen vor den Antworten der Speicherbänke bei den SP-Chips eintreffen.

Speichern ausstehender Supply-Operationen

Das Speichern der Anfragen stellt in einem PHOTON-System insofern kein Problem dar, als dafür die Attributspeicher benutzt werden können. Eine Supply-Anforderung kann in einer BE nur für ein Datum ankommen, daß in seinem privaten Attributspeicher vermerkt ist. Dies bedeutet, daß zum Speichern ausstehender Anfragen kein zusätzlicher Buffer notwendig ist. Es reicht, wenn den Attributen ein 1-Bit Feld hinzugefügt wird, daß eine ausstehende Supply-Operation anzeigt.

Timing der Supply-Benachrichtigungen

Bei den PHOTOBUS- und PHOTOBAR-Systemen basierte die Lösung des Timing Problems darauf, daß die Supply-Benachritigungen wesentlich kürzer als die Fetch-Operationen sind. Eine Fetch-Operation besteht aus einer Adresse und einem Befehlswort und ist damit ca. 80 Bit Lang. Eine Supply-Benachrichtigung beinhaltet hingegen nur einer Speicherbanknummer und ein Befehlsbit. Sie ist damit ca. 10 Bit lang. Daraus folgt, daß Supply-Benachrichtigungen trotzt der geringeren Bandbreite des P-Kanals schneller oder genauso schnell verschickt werden konnten, wie über den B-Kanal neue Anfragen ankommen konnten. Damit kommen sie auf jedem Fall bei den SP-Chips an, bevor die langsamen Speicherbänke antworten können.

Für das PHOTON-Konzept wird eine ähnliche Lösung vorgeschlagen. Bei wenigen B-Kanälen (2 bis 8) mit mäßiger Bandbreite (4 bis 8 Bytes/Cycle) reicht es hierfür die bandbreite des P-Kanals entsprechend zu erhöhen (auf 2 bis 6 Bytes/Cycle). Mit Steigender Anzahl der B-Kanäle ist es hingegen notwendig mehrere, unabhängige P-Kanal Transmitter zu benutzen. Jeder dieser Transmitter ist dann für das Verschicken von daten zu einer bestimmten Gruppe von SP-Chips verantwortlich. Dies bedeutet, daß die Anzahl der Supply-Anforderungen, die ein solcher Transmitter pro Zeiteinheit übertragen können muß, nicht zu Gesamtanzahl der B-Kanäle sonder zur zahl der Ihm zugeordneten SP-Chips proportional ist. Damit benötigt jeder Transmitter nur eine verhältnismäßig geringe Bandbreite.

Ablauf der Supply-Operationen

Wird auf dem B-Kanal eine Fetch-Operation übertragen, so prüft die BE in ihrem privaten Attributspeicher, ob sich die betroffene Adresse im SC im Exclusiv-Zustand befindet. Ist dies der Fall, so wird ebenfalls im privaten Attributspeicher bei der entsprechenden Zeile vermerkt, daß eine Supply-Operation notwendig ist. Danach wird sofort eine Supply-Benachrichtigung über das Crossbar-Netzwerk und den entsprechenden P-Kanal-Transmitter an den zuständigen SP-Chip geschickt. Wegen der strengen Timing-Anforderungen an die Supply-Benachrichtigungen hat diese bei der Übertragung absolute Priorität. Sollten also gerade andere Daten über den Transmitter übertragen werden, so wird diese Übertragung für die Dauer der Benachrichtigung unterbrochen. Eine Wartezeit ist nur dann notwendig, wenn gerade eine andere Supply-Benachrichtigungen gesendet wird. Wie oben beschrieben, gehen wird davon aus, daß der P-Kanal durch mehrere unabhängige Transmitter so viel Bandbreite zur Verfügung stellt, daß Verzögerungen durch Konflikte zwischen Supply-Benachrichtigungen kein Problem darstellen.

9.3 Bewertung

Neben der Lösung des Cache-Kohärenz Problems besteht die Motivation des PHOTON-Konzeptes darin, dem starken Anstieg der Fläche der Netzwerkschips und der Saturierung des B-Kanals bei hohen Prozessorzahlen entgegenzuwirken. Der vorliegende Abschnitt beschäftigt sich mit der Bewertung des PHOTON-Konzepts im Hinblick auf diese beiden Aspekte.

9.3.1 Chipfläche

Das Problem bei der Chipfläche besteht darin, daß die Gatteranzahl der Schaltungen wie $P \cdot M \cdot ld(M)$ und die Fläche der Verbindungsstruktur wie $P \cdot M^2$ ansteigt. Dies führt dazu, daß weder die PHOTOBUS noch die PHOTOBARArchitektur für mehr als 128 Prozessoren praktibel ist. Da bei der PHOTON-Architektur die Speicherbänke auf N_{BC} SP-Chips verteilt sind, werden an jedem Chip nur $1/N_{BC}$ Speicherbänke benötigt. Die Anzahl der Speicherbänke M kommt in den entsprechenden Gleichungen als $\frac{M}{N_{BC}}$ vor. Mit einem angestrebten Wert von N_{BC} von bis zu 32, führt dies zu einer dramatischen Reduktion des Flächenbedarfs. Um dies zu verdeutlichen betrachten wir die Schätzungen für die Gatterzahl und Verbindungsfläche der PHOTOBUS- (Gleichungen 7.106 und 9.9) und der PHOTOBAR-Architektur (Gleichungen 8.15 und 8.16). Durch Ersetzen von M durch $\frac{M}{N_{BC}}$ ergibt sich für den PHOTOBUS:

$$NG_P \simeq 2P \cdot \frac{M}{N_{BC}} \cdot ld\left(\frac{M}{N_{BC}}\right) + 50P \cdot \frac{M}{N_{BC}} + 4000P + 2000\frac{M}{N_{BC}} + 5000 \tag{9.6}$$

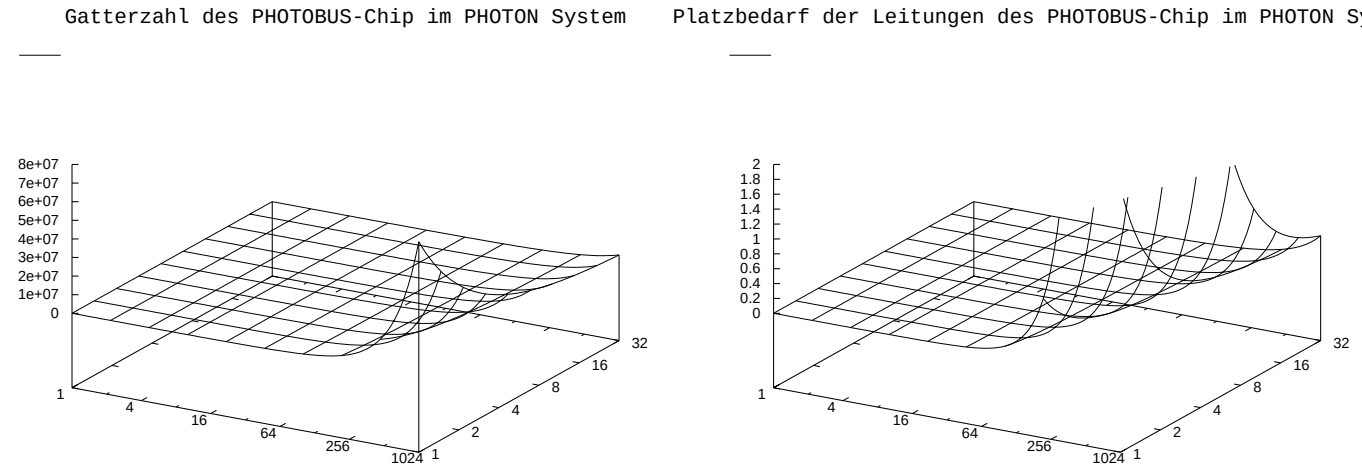


Abbildung 9.5: Das linke Diagramm zeigt die Abhängigkeit der auf einem PHOTOBUS-Chip im PHOTON-System benötigten Gatter (vertikale Achse) von der Prozessorzahl (x Achse) und der Anzahl der Buschips (y-Achse). Das rechte Diagramm stellt die Abhängigkeit der für die Verbindungsstruktur benötigten Fläche in cm^2 (vertikale Achse) von der der Prozessorzahl und der Anzahl der Chips dar. Beide Diagramme verwenden für die Prozessor- und Chipzahl eine logarithmische Darstellung.

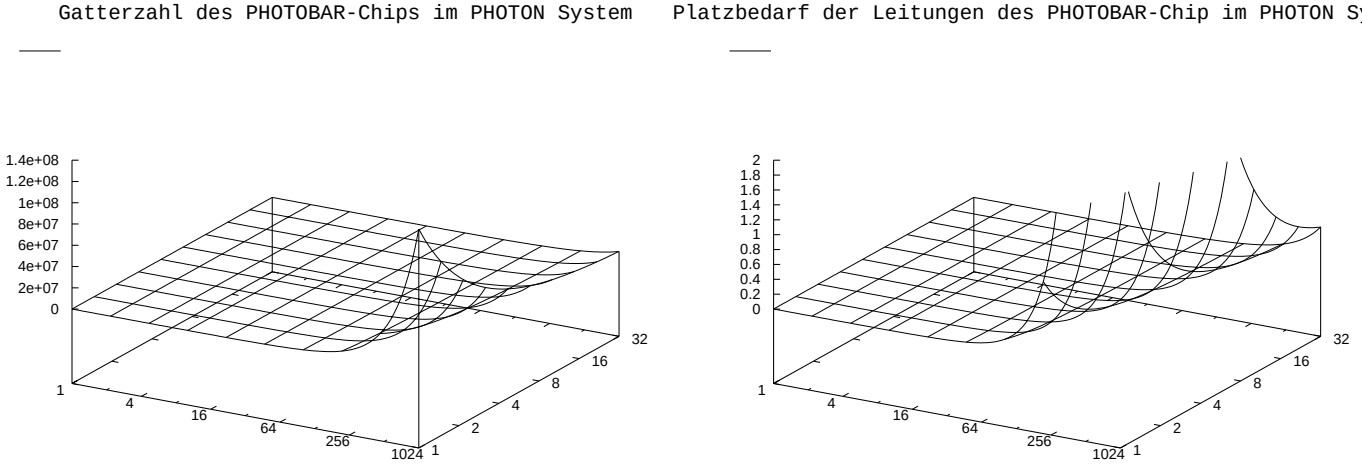


Abbildung 9.6: Das linke Diagramm zeigt die Abhängigkeit der auf einem PHOTOBAR-Chip im PHOTON-System benötigten Gatter (vertikale Achse) von der Prozessorzahl (x Achse) und der Anzahl der Netzchips(y-Achse). Das rechte Diagramm stellt die Abhängigkeit der für die Verbindungsstruktur benötigten Fläche in cm^2 (vertikale Achse) von der der Prozessorzahl und der Anzahl der Chips dar. Beide Diagramme verwenden für die Prozessor- und Chipzahl eine logarithmische Darstellung.

und

$$\boxed{\begin{array}{rcl} A_V[cm^2] & \simeq & (12P \cdot \left(\frac{M}{N_{BC}}\right)^2 + 100\left(\frac{M}{N_{BC}}\right)^2 + 1100P \cdot \frac{M}{N_{BC}} \\ & & +42000P \cdot +12000\frac{M}{N_{BC}} + 50000) \cdot 6,25 \cdot 10^{-10} \end{array}} \tag{9.7}$$

Für den PHOTOBARChip gilt analog:

$$\boxed{NG_S \simeq 3P \cdot \frac{M}{N_{BC}} \cdot ld(\frac{M}{N_{BC}}) + 100P \cdot \frac{M}{N_{BC}} \cdot +6000P \cdot +3000\frac{M}{N_{BC}} \cdot +5000} \tag{9.8}$$

und

$$\boxed{A_V[cm^2] \simeq (12P \cdot \left(\frac{M}{N_{BC}}\right)^2 + 500\left(\frac{M}{N_{BC}}\right)^2 + 1500P \cdot \frac{M}{N_{BC}} \cdot +42000P \cdot +12000\frac{M}{N_{BC}} + 50000) \cdot 6,25 \cdot 10^{-10}} \tag{9.9}$$

Diese Ausdrucke wurden in Abbildungen 9.5 und 9.6 für Prozessorzahlen von 1 bis 1024, und 1 bis 32 Netzwerkchips dargestellt. Bei der Darstellung wurde, wie auch bei der Betrachtung in den Kapiteln 8 und 9 angenommen, daß die Gesamtanzahl der Speicherbänke gleich der Prozessorzahl ist. Ein Vergleich mit Abbildungen 7.28 und 8.12 zeigt, daß durch die Verteilung der Speicherbänke auf 16 bis 32 B-Kanälen der Ansteig der Prozessorzahl auf 512 bis 1024 kompensiert wird. Somit ist der folgende Schluß gerechtfertigt:

Durch die Verwendung des PHOTON-Konzeptes stellt der Flächenbedarf der SP-Chips kein Hindernis für die Realisierung von Rechnern mit 512 bis 1024 Prozessor dar.

9.3.2 Effizienz

Die Frage der Effizienz der PHOTON-Architektur ist insofern schwer exakt zu beantworten, als PHOTON wie beschrieben ein Konzept für eine ganze Klasse von Architekturen darstellt. Hinzu kommt, daß die verwendetet Benchmark-Implementierungen und Datensätze für 64 bis 128 Prozessoren gedacht sind. Dies bedeutet, daß bei größeren Prozessorzahlen, wie sie für das PHOTON-Konzept von Bedeutung sind, nur bedingt sinnvolle Ergebnisse zu erwarten sind. Die Verwendung größere Probleme ist nicht ohne weiteres möglich, da sie einen zu hohen Simulationsaufwand mitsich bringen würde. Um größere Systeme zu bewerten ist es daher notwendig, Teilläufe von Programmen zu betrachten. Dazu müssen allerdings zunächst sinnvolle Teile bestimmt werden und Vergleichswerte für das idealle System ermittelt werden. Eine genaue Bewertung größere Konfigurationen konkreter PHOTON-Architekturen muß daher als eine Aufgabe für künftige Arbeiten betrachtet werden.

Als Indiz für das Leistungspotential des Konzeptes werden im Nachfolgenden die Ergebnisse einiger Simulationen eines PHOTOBUS-basierten PHOTON-Systems präsentiert. Dabei wurde die Simulationsumgebung für die PHOTOBUS-Architektur mit on-Chip Synchronisation so erweitert, daß sie mehrere Bus-Chips zuläßt. Für die Bearbeitungszeiten im Buschip und in den Schnittstellen wurden die gleichen Werte wie in Kapitel 7 verwendet. Da man davon ausgehen muß, daß ein PHOTON-System mit vielen Prozessoren viel Platz einnehmen wird, wurde für die Signallatenz ein Wert von $L_B = L_P = L_M = 8$ verwendet. Für die Bandbreite der P und M-Kanäle wurde $B_P = B_M = 1$ angenommen. Dieser Annahme liegt die Überlegung zur Grunde, daß bei einem PHOTON-System die Anzahl der Fasern, die an die SP-Chips angekoppelt werden einen kritischen Faktor darstellt, so daß pro Kanal möglichst wenige fasern (und damit eine möglichst geringe Bandbreite) benutzt werden sollten.

Die Ergebnisse der Simulation sind in Abbildungen 9.7 und 9.8 zu sehen. Die Simulation wurde wegen der oben erwähnten Probleme mit der Größe der Datensätze nur bis 256 Prozessoren durchgeführt. Die Graphen zeigen wie auch bei der Bewertung der PHOTOBUS- und PHOTOBAR-Architekturen auf der vertikalen die Ausführungszeit im Vergleich zum ideale Speichersystem mit Cache so wie die parallelen Beschleunigung. Auf der horizontalen Achse ist die Anzahl der Prozessoren aufgetragen. Die einzelnen Kurven unterscheiden sich in der Bandbreite des B-Kanals (als B_B gekennzeichnet) und der Anzahl der Buschips (als R gekennzeichnet).

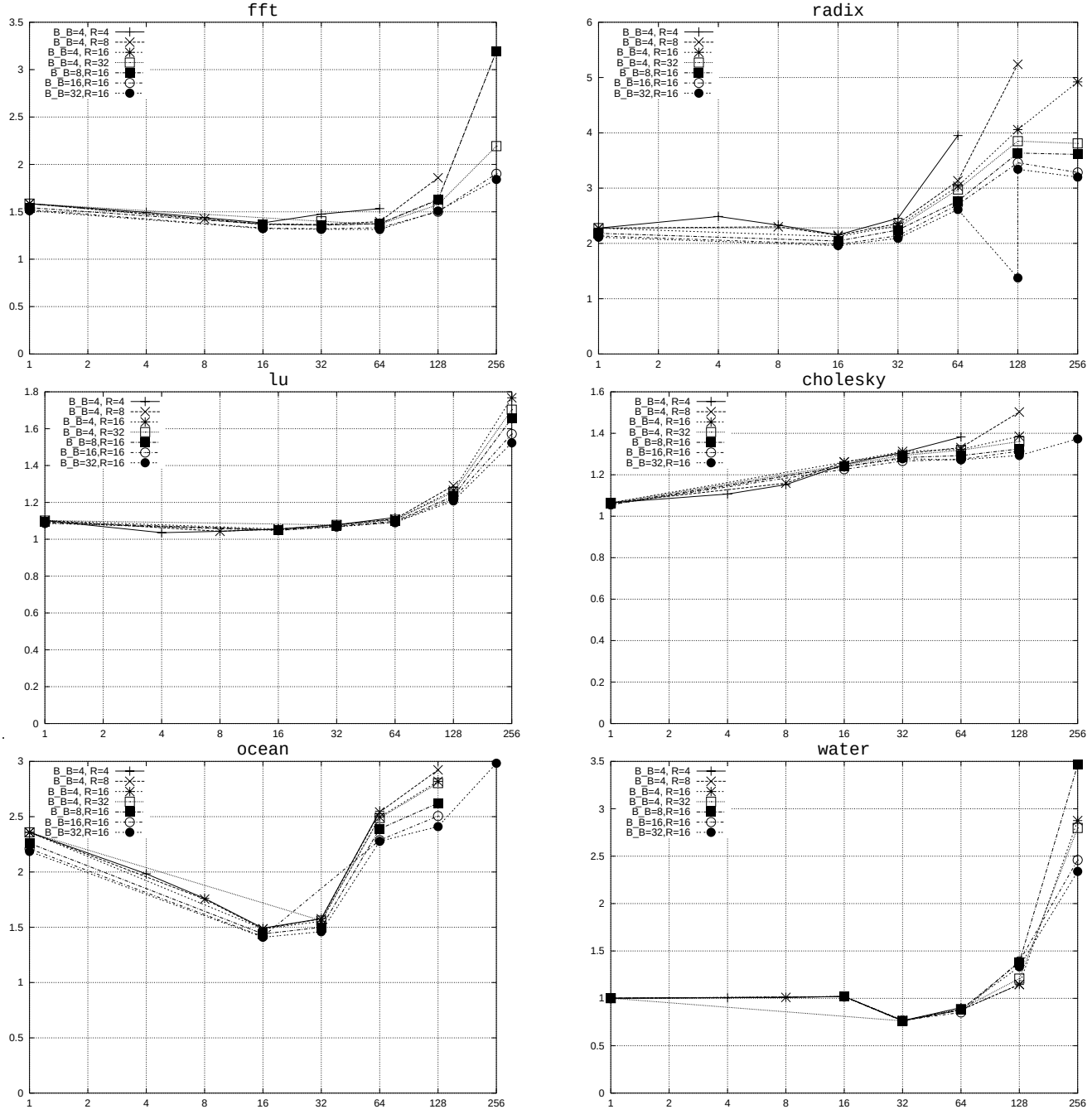


Abbildung 9.7: Die obigen Diagramme zeigen für die PHOTOBUS-Architektur mit on-Chip Synchronisation die durch Simulation ermittelte Ausführungszeit der einzelnen Benchmarks im Vergleich zur Ausführungszeit mit einem idealen Speichersystem

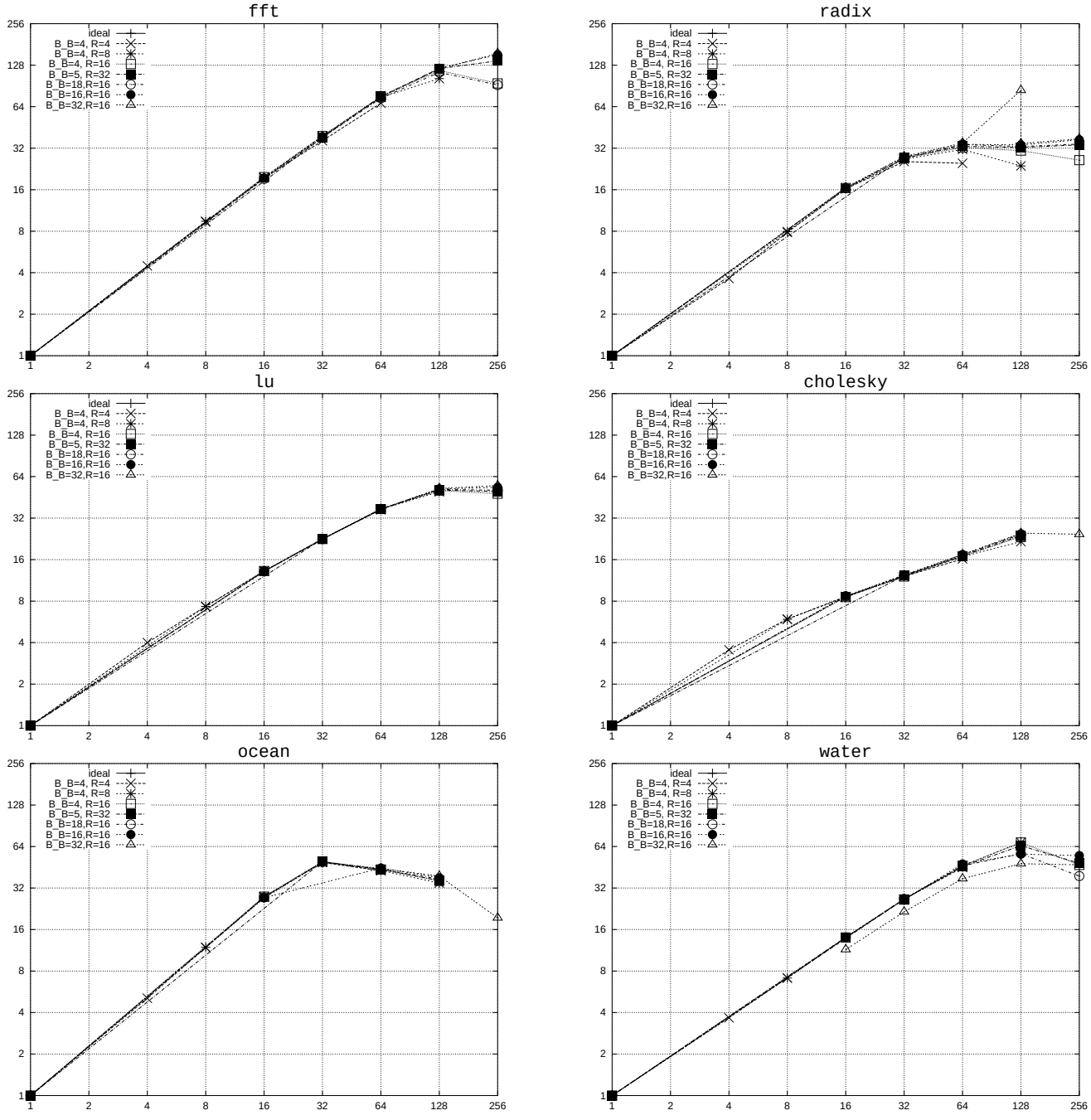


Abbildung 9.8: Die durch Simulation ermittelte Beschleunigung der einzelnen Benchmarkprogramme für die PHOTOBUS-Architektur mit on-Chip Synchronisation.

Kapitel 10

Zusammenfassung und Ausblick

10.1 Zusammenfassung der Ergebnisse

Wie in Kapitel 1 beschrieben verfolgt die Arbeit vor allem zwei Ziele:

1. zu zeigen, daß die Argumente für die Beschränkung des SMP Speichermodells auf Rechner mit wenigen Prozessor und gegen seine Anwendung bei der Vernetzung von Arbeitsplatzrechnern nicht für Rechner mit opto-elektronische Prozessor-Speicher Koppelung nicht gelten und

2. eine Grundlage für die tatsächliche Implementierung solcher opto-elektronischer SMPs zu schaffen.

Die im Bezug auf diese beiden Ziele in der Arbeit erzielten Ergebnisse können wie folgt zusammengefaßt werden.

Widerlegung der Argumente für die Beschränkung der SMPs

Es wurde an Hand einer Literaturstudie gezeigt, daß optische Verbindungen mit Hunderten bis Tausenden von hochfrequenten Kanälen und einem Fanout von bis zu 1000 kurz bis mittelfristig realisiert werden können. Solche Verbindungen können direkt zwischen VLSI-Bausteinen implementiert werden, so daß die Latenz trotz des hohen Fanouts im Bereich der Signallaufzeit liegt. Daraufhin wurden mit dem PHOTOBUSund der PHOTOBAR zwei Architekturen vorgestellt, die solche Verbindungen effektiv zu Realisierung eines symmetrischen Multiprozessors nutzen. Für diese Architekturen wurde unter realistischen Annahmen bezüglich der Technologieparameter durch Simulation gezeigt, daß eine hohe Effizienz für bis zu 128 Prozessoren erreicht werden kann. Abschließend wurde durch das PHOTON-Konzept gezeigt, wie die die drei Haupthindernisse für die Skalierbarkeit PHOTOBUS- und PHOTOBAR-Architekturen bis zu 256 oder gar 512 Prozessoren beseitigt werden können. Dazu wurde beschreiben wie durch den parallelen Betrieb mehrerer PHOTOBUS- oder PHOTOBAR-Systeme der starke Anstieg der Fläche der SP-Chips mit der Prozessorzahl, die Saturierung des B-Kanals und das Cache Update Pressure Problem gelöst werden können. Insbesondere die Lösung des Cache-Update-Pressure Problems stellt einen sehr wichtigen Beitrag dar, da es bisher allgemein als nicht praktikabel angesehen wurde.

Schaffung der Grundlagen für eine Implementierung

Um der Weg für die tatsächliche Realisierung opto-elektronischer symmetrischer Prozessoren zu ebnen, wurde zunächst in Zusammenarbeit mit Wissenschaftlern aus den Bereichen der Optischen Nachrichtentechnik der Fernuniversität Hagen, und der Mikromechanik am IMM Mainz ein Konzept für ein modulares opto-elektronisches Baukastensystem entwickelt. Das Konzept des Systems wurde durch die Herstellung einfacher Komponenten verifiziert. Es wurde auch gezeigt, wie es zur Realisierung der in der Arbeit vorgeschlagenen Architekturen benutzt werden kann. Als nächstes wurde für die PHOTOBUSund die PHOTOBAR-Architektur beschrieben, welche Voraussetzungen erfüllt werden müssen, damit das Berkeley- Protokoll korrekt und effizient realisiert werden kann. Ausgehend von diesen Voraussetzungen wurde für alle Komponenten die benötigte Hardwarearchitektur beschrieben. Dabei wurde der Aufbau der Komponenten durch Strukturdiagramme, ihre Funktionalität durch endliche Automaten exakt spezifiziert.

10.2 Ausblick

Ein nur auf Papier existierender und lediglich durch Simulation untersuchter Rechner ist nur ein 'Papiertieger'. Die wichtigste nachfolgende Forschungsaufgabe stellt daher die tatsächliche Implementierung der in der Arbeit vorgestellten Rechnerarchitekturen dar. Darüber hinaus gibt es eine Reihe möglicher Verbesserungen und Weiterentwicklungen der PHOTOBUS- und PHOTOBAR-Architekturen, die im Rahmen der Arbeit nicht mehr untersucht werden konnten. Hinzu kommt die Notwendigkeit des Entwurfs konkreter PHOTON-Architekturen. Im Hinblick auf langfristige Entwicklungen stellt sich die Frage, wie das Potential der Freiraumoptik besser genutzt werden kann.

Implementierung

An der Implementierung eines Beispielsystems, das auf den in der Arbeiten vorgestellten Konzepten basiert, wird zur Zeit im Rahmen eines Projektes der Deutschen Forschungsgemeinsachaft gearbeitet. Das Projekt wird vom Institut für Promgrammstrukturen und Datenorganisation der Universität Karlsruhe zusammen mit dem Institut für Optische Nachrichtentechnik der Fernuniversität Hagen durchgeführt. In der ersten Stufe des Projektes wird der Prozessor-Speicher-Bus eines älteren SUN-Arbeitsplatzrechners durch eine optische Verbindung ersetzt. Die Verbindung basiert auf parallelen Fasersystemen, die an einen in planarer Freiraumoptik realisierten Sternkoppler angeschlossen sind. Dieses einfache System stellt eine Testplattform für die neunen Komponenten dar, die für die Realisierung der PHOTOBUS- und PHOTOBAR- Architekturen notwendig sind. Dabei geht es vor allem um die Ankoppelung der parallele Fasersysteme an die planare Freirraumoptik und die Schnittstellen zwischen dem konventionellen elektrischen Speicherbus und dem optischen Übertragungssystem. Aufbauend auf dem Testsystem soll dann eine erste, einfache Version der PHOTOBUS-Architektur realisiert werden.

Weiterentwicklung der PHOTOBUS und PHOTOBAR Architekturen

Ein großer Vorteil der Verwendung von opto-elektronischen, SP-Chip basierte Systemen besteht darin, daß auf dem Chip alle zu einem bestimmten Zeitpunkt im System vorhanden Speicheranfragen bekannt sind. Dies verspricht die Möglichkeit, verschiedene Optimierungen zu realisieren, die in anderen Systemen nicht praktikabel sind. In der Arbeit wurde nur eine solche Optimierung beschrieben: die effiziente on-Chip Synchronisation. Um das Potential der hier beschriebenen Architektur voll nutzen zu können, ist es notwendig, weitere solche Optimierungsverfahren zu entwickeln. Besonderes interessant erscheint dabei die Verwendung von Cache-Speichern auf den SP Bausteinen. Ein weiteres wichtiges Forschungsthema ist die Frage, ob man die Effizienz von PHOTOBUS- und PHOTOBAR-Systemen durch die Verwendung anderer Cache-Kohärenz Protokolle verbessern kann. Dabei sollte man auch die Möglichkeit in Betracht ziehen, neue, speziell auf diese Architekturen zugeschnittene Protokolle zu entwerfen.

Konkrete PHOTON-Architekturen

Wie im Kapitel 9 erläutert stellt das PHOTON-Konzept lediglich einen Rahmen für mögliche Rechnerarchitekturen dar, der die Lösung des Cache Update Pressure Problems ermöglicht. Um die Implementierung von großen symmetrischen Multiprozessoren mit mehreren Hundert Prozessoren möglich zu machen, müssen konkrete Architekturen entworfen und untersucht werden.

Bessere der Nutzung des Potentials der Freiraumoptik

Eine der großen Stärken des in der Arbeit vorgeschlagenen Bauskastensystems für opto-elektronische Verbindungen besteht darin, daß es die Vorteile von Fasersystemen mit den von hochintegrierten optischen Freiraumsystemen vereinigt. Die letzeren werden in den untersuchten Architekturen allerdings nur wenig genutzt. Die Freiraumoptik wird lediglich zur Abbildung der Faser auf die optischen Fenster der SP-Bausteine und zur Realisierung des Rundrufs verwendet.

Wie in Kapitel 4 erläutert erlaubt die Freiraumoptik die Implementierung sehr komplexer Verbindungsstrukturen auf einem engen Raum. So kann eine Verbindungsstruktur mit einer Bisektionsbreite von B, die in planarer VLSI-Technologie eine Fläche von $O(B^2)$ benötigt, auf einer Grundfläche von $O(B)$ realisiert werden. Der Autor hat ein Konzept zur Nutzung solcher Verbindungsstrukturen entwickelt, daß die Implementierung opto-elektronische Speichermodule mit einem echt parallelen Zugriff für hohe Prozessorzahlen genutzt

werden kann. Verschiedene Aspekte des System wurde in bereits in mehreren Publikationen veröffentlicht [106, 107, 108, 102, 103]. Es basiert auf dem gleichen Prinzip, das es einer Gruppe von Menschen erlaubt, gleichzeitig eine auf einen Schirm projizierte Folie zu lesen. Analog können mehrere Menschen gleichzeitig mit Hilfe von z.B. Laserzeigern Informationen auf einen Schirm 'schreiben'.

Mit Hilfe solchen parallele Speichermodule wäre es möglich, mehrere Prozessoren an einen gemeinsamen Cache anzuschließen, ohne daß die Cachelatenz durch Sequentialisierung vergrößert wird. Dies würde das Hauptspeichersystem entlasten und dadurch effizientere SMPs mit mehr Prozessoren erlauben. Allerdings erfordert die Implementierung eines solchen Systems noch eine Menge Forschungsarbeit, sowohl im Bereich der Technologie als auch in der Frage der Cache-Zugriffsprotokolle.

Literaturverzeichnis

[1] Microelectronics advanced research initiatives mel- ari opto technology roadmap - june 1998. http://www.cordis.lu/esprit/src/melop-rm.htm, 1998.

[2] Tarek S. Abdelrahman. Latency hiding on COMA multiprocessors. *The Journal of Supercomputing*, 10(3):225–242, 1996.

[3] Ferri Abolhassan, Reinhard Drefenstedt, Jörg Keller, Wolfgang J. Paul, and Dieter Scheerer. On the physical design of PRAMs. *Computer Journal*, 36(8):756–762, December 1993.

[4] F.A.P. Tooley et al A.C. Walker, M.P.Y. Desmulliez. Construction of an optoelectronic bitonic sorter based on cmos/ingaas smart pixel technology. In *Proc. of the 2cnd International Conference on Massively Parallel Processing Using Optical Interconnections*, page 180. IEEE Computer Society Press, 1995.

[5] Sarita V. Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. *Computer*, 29(12):66, December 1996.

[6] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz. An evaluation of directory schemes for cache coherence. *Proc. 15th Int. Symp. Comput. Architecture*, pages 280–289, 1988.

[7] Anant Agarwal, Ricardo Bianchini, David Chaiken, Kirk L. Johnson, David Kranz, John Kubiatowicz, Beng-Jong Lim, Kenneth Mackenzie, and Donald Yeung. The MIT alewife machine: Architecture and performance. In *Proceedings of the 22th Annual International Symposium on Computer Architecture*, 1995.

[8] R. A. Alfieri. An efficient kernel-based implementation of POSIX threads. In *Proceedings of the Summer 1994 USENIX Technical Conference and Exhibition*, pages 59–72, June 1994.

[9] Robert Alverson, David Callahan, Daniel Cummings, Brian Koblenz, Allan Porterfield, and Burton Smith. The Tera computer system. In *Proceedings of the 1990 ACM International Conference on Supercomputing*, pages 1–6, 1990.

[10] A. Rogner J. Goettert J. Mohr A.Mueller, J. Hehmann. Hybrid optical tranceiver module with a micro optical LIGA-bench. In *Proc. 21st Eur. Conf. on Opt. Comm. (ECOC'95-Brussels)*, page 465, 1995.

[11] A. J. De Groot an R. J. Deri, R. E. Haigh, F. G. Patterson, , and S. P. DiJaili. High-performance parallel processors based on star-coupled wdm optical interconnects. In *Proc. of the Third International Conference on Massively Parallel Processing Using Optical Interconnections*, page 62. IEEE Computer Society Press, 1996.

[12] Archibald and Baer. Cache coherence protocols: An evaluation of cache coherence solutions in shared-bus multiprocessors. *ACM Transactions on Computer Systems*, pages 273–298, November 1986.

[13] Semiconductor Industry Association. The national technology roadmap for semiconductors. Technical report, 1997.

[14] Hagit Attiya, Soma Chaudhuri, Roy Friedman, and Jennifer L. Welch. Shared memory consistency conditions for nonsequential execution: Definitions and programming strategies. *SIAM Journal on Computing*, 27(1):65–89, February 1998.

[15] Friedhelm Meyer auf der Heide and Hieu Thien Pham. On the performance of networks with multiple busses. In *Proceedings of the 9th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 97–103, 1992.

[16] D.A.B. Miller A.V. Krishnamoorthy. Scaling optoelectronic-vlsi circuits int the 21st century: A technology roadmap. *IEEE Journal Of Selected Topics in Quantum Electronics*, 2(1), April 1996.

[17] Utpal Banerjee. *Loop transformations for restructuring compilers : the foundations.* Kluwer, 1993.

[18] William R. Bryg, Kenneth K. Chan, and Nicholas S. Fiduccia. A high-performance, low-cost multiprocessor bus for workstations and midrange servers. *Hewlett-Packard Journal: technical information from the laboratories of Hewlett-Packard Company*, 47(1):18–24, February 1996.

[19] David R. Butenhof. *Programming with POSIX threads.* Addison-Wesley, Reading, MA, USA, 1997.

[20] R. Jger M. Grabherr F. Eberhard RMichalzik K.J. Ebeling C. Jung, R. King. Highly efficient oxide confined vcsel arrays for parallel optical interconnects. In *Proc. Optics in Computing OC'98 Brugge*, pages 2–5, 1998.

[21] J. W. Kim R. V. Stone P. S. Guilfoyle F. E. Kiamiliev C. M. Travers, J. M. Hessenbruch. Vlsi photonic smart pixel array for i/o system architectures. In *Proc. SPIE Photonics West Conference*, January 1998.

[22] L. M. Censier and P. Feautrier. A new solution to the coherence problems in multicache systems. *IEEE Transactions on Computers*, C-27(12):1112–1118, December 1978.

[23] D. Chaiken, J. Kubiatowics, and A. Agarwal. LimilLESS directories: A scalable cache coherence scheme. In *Proc. Fourth International Conf. on Architectural Support for Programming Languages and Operating Systems, SIGOPS Operating Systems Review Special Issue*, volume 25, page 224, Santa Clara, CA, April 1991.

[24] Clare and Keith Jenkins. Shared-memory optical/electronic computer: Architecture and control. *Applied Optics*, 33(8):1559, January 1994.

[25] Cray Research, Inc. *CRAY T3D System Architecture Overview*, hr-04033 edition, September 1993.

[26] Fredrik Dahlgren, Michel Dubois, and Per Stenstrom. Sequential hardware prefetching in shared-memory multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 6(7):733–746, July 1995.

[27] Anoop Gupta David E. Culler, Jaswinder Pal Singh. *Modern parallel computer architecture : a hardware/software approach.* Morgan Kaufmann, 1999.

[28] Jr. Martin H. Davis and Umakishore Ramachandran. A distributed hardware barrier in an optical bus-based distributed shared memory multiprocessor. In *Proceedings of the 1992 International Conference on Parallel Processing*, volume I, Architecture, pages I:228–231, Boca Raton, Florida, August 1992. CRC Press.

[29] M. H. Davis, Jr. and U. Ramachandran. Optical bus protocol for a distributed shared memory multiprocessor. In *Proc. of the SPIE—The Int'l Society for Optical Engineering*, pages 176–187, July 1991.

[30] L. Dekker, E.E.E. Frietman, and J.C. Zuidervaart W. Smit. Optical link in the delft parallel processor: An example of momi-connection in an mimd supercomputer. In *Proceedings of the International Conference on Frontiers in Computing, Amsterdam*, page 141, 1987.

[31] E. Desurvivre. Erbium-doped fiber amplifiers fpr new generation of optical communication systems. *Optics & Photonics News*, 2(1):6–11, 1991.

[32] Reinhard Drefenstedt and Dietmar Schmidt. On the physical design of butterfly networks for PRAMs. In *Proc. Frontiers '92: Fourth Symp. on Massively Parallel Computation*, pages 202–209, McLean, VA, October 1992. IEEE.

[33] G. S. Graham K. C. Sevcik E.D. Lazowska, J. Zahorjan. *Quantitative System Perfomance.* Prentice-Hall, 1984.

[34] Frietman E.E.E. and W. Smit. Massively parallel processing: Optical interconnects according to a system to device approach. In *Proc. of the First Workshop on Massively Parallel Processing Using Optical Interconnections*, page 94. IEEE Computer Society Press, 1994.

[35] W. Ehrfeld and D. Muenchmeyer. Three dimensional microfabrication using synchrotron radiation. *Nuclear Instruments and Methods in Physics Research*, A 303:523, 1991.

[36] G. Eisenstein. Semiconductor optical amplifiers. *IEEE Circuits and Devices Magazine*, 5(4):25–30, 1989.

[37] D.R. Engebertsen, D.M. Kuchta, R.C. Booth, J.D. Crow, and W.G. Nation. Parallel fiber-optic sci links. *IEEE Micro*, 16(1):20–26, January 1996.

[38] A. V. Krishnamoorthy et al. Vertical cavity surface emitting lasers flip-chip bonded to gigabit/s cmos circuits. In *Post Deadline Paper, Optics in Computing OC'98 Brugge*, June 1998.

[39] F.B. MCormic et al. Five-stage free-space optical switching network with field-effect transistor delf-electro-optic-effect-device smart-pxiel arrays. *Applied Optics*, 33(8):1601, March 1994.

[40] K. W. Goossen et al. Gaas mqw modulators integrated with silicon cmos. *IEEE Photonics Technology Letters*, 7(5), April 1995.

[41] P. S. Guilfoyle et. al. High-speed, low energy digital optical processors. *Optical Engineering*, 35(2), February 1996.

[42] Chih fang Wang and Sartaj Sahni. Otis optoelectronic computers. In Keqin Li, Yi Pan, and Si Qing Zheng, editors, *Parallel Computing Using Optical Interconnections*, pages 99–115. Kluwer Academic Publishers, 1998.

[43] M.R. Feldman, T.J. Drabik, S.C. Esener, and C.C. Guest. Comparison between optical and electrical interconnects for fine grain processor arrays based on interconnect density capabilities. *Applied Optics*, 28(18):3820–3829, September 1989.

[44] W. Erhardand D. Fey. *Parallele Digitale Optische recheneinheiten*. B.G. Teubner Stuttgart, 1994.

[45] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Computers*, C-21(9):948–960, September 1972.

[46] Michael J. Flynn. *Computer architecture : pipelined and parallel processor design*. Jones & Bartlett, 1995.

[47] S. Fortune and J. Wyllie. Parallelism in random access machines. In *ACM Symposium on Theory of Computation*, pages 114–118, May 1-3 1978.

[48] Ricardo Bianchini John Kubiatowicz Frederic T. Chong, Beng-Hong Lim and Anant Agarwal. Application performance on the mit alewife machine. *IEEE Computer*, 29:57–64, December 1996.

[49] E.E. Frietman, W. van Nifterick, L. Dekker, and T.J.M. Jongeling. Parallel optical interconnects: Implementation of optoelectronics in multiprocessor architectures. *Applied Optics*, 29(8):1160–1177, March 1990.

[50] E.E.E. Frietman. Opto-electronic processing & networking in massively parallel processing systems. In *Proc. of the Second International Conference on Massively Parallel Processing Using Optical Interconnections*, page 355. IEEE Computer Society Press, 1995.

[51] Institut fur Mikrotechnik Mainz. Internet produktinformationen. http://www.imm-mainz.de/content.html.

[52] A. M. Fox G. D. Boyd and D. A. B. Miller. 33 ps optical switching of symmetric self-electro-optic effect devices. *Applied Physics Letters*, (57):1843–1845, 1990.

[53] Mike Galles and Eric Williams. Performance optimizations, implementation, and verification of the sgi challenge multiprocessor. Silicon Graphics Report http://www.sgi.com/Technology/challenge_paper.html.

[54] A Geist, A Beguelin, J Dongarra, W Jiang, R Manchek, and V Sunderam. *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.* Scientific and Engineering Computation. MIT Pres, 94.

[55] K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. Hennessy. Memory consistency and event ordering in scalable shared-memory multiprocessors. In *Proceedings of the 17th Annual Symposium on Computer Architecture*, pages 15–26, May 1990.

[56] W.K. Giloi. *Rechnerarchitektur.* Springer-Verlag, 1991.

[57] C. Gimkiewicz and J. Jahns. Thermal mamagement in planar optical systems with active components. In *Proc. SPIE Symp. Microelectronic Structures and MEMS for Optical Processing III, Austin*, September 1997.

[58] D. B. Glasco, B. A. Delagi, and M. J. Flynn. Update-based cache coherence protocols for scalable shared-memory multiprocessors. In Trevor N. Mudge and Bruce D. Shriver, editors, *Proceedings of the 27th Hawaii International Conference on System Sciences. Volume 1 : Architecture*, pages 534–545, Los Alamitos, CA, USA, January 1994. IEEE Computer Society Press.

[59] Allen Gottlieb, Ralph Grishman, Clyde P. Krukal, Kevin P. McAuliffe, Larry Rudolph, and Marc Snir. The NYU ultracomputer – designing an MIMD shared memory parallel computer. *IEEE Transactions on Computers*, C-32(2):175–190, February 1983.

[60] N.J. Gunther. *The Practical Perfomance Analyst.* McGraw Hill.

[61] David B. Gustavson. The Scalable Coherent Interface and related standards projects. *IEEE Micro*, 12(1):10–22, February 1992.

[62] M. Honsberg J. Kropp J. Wieland M. Blaser H. Karstensen, Ch. Hanke. Paroli - high performance optical bus with integrated components. In *Topical Meeting on 'Integrated Optoelectronics', Lake Tahoe*, June 1994.

[63] Joon-Ho Ha and Timothy Mark Pinkston. SPEED DMON: Cache coherence on an optical multichannel interconnect architecture. *Journal of Parallel and Distributed Computing*, 41(1):78–91, February 1997.

[64] S. P. Harbison. *Modula-3.* Prentice Hall, Englewood Cliffs, 1992.

[65] S. Hariri, J. B. Park, F.-K. Yu, M. Parashar, and G. C. Fox. A message passing interface for parallel and distributed computing. In IEEE, editor, *Proceedings of the 2nd International Symposium on High Performance Distributed Computing, July 20–23, 1993, Spokane, Washington, Cavanaugh's Inn at the Park*, pages 84–91, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1993. IEEE Computer Society Press.

[66] E. Hecht. *Optics.* Addison-Wesley, Reading, Mass., 1987.

[67] J. Hecht. *Understanding Fiber Optics.* SAMS Publishing, 1993.

[68] John Heinlein, Kourosh Gharachorloo, Scott A. Dresser, and Anoop Gupta. Integration of message passing and shared memory in the Stanford FLASH multiprocessor. pages 38–50, San Jose, California, 1994.

[69] David C. Hoffmeister, John Chu, James A. Perreault, and Patrick Dowd. Lightning network and systems architecture. In Keqin Li, Yi Pan, and Si Qing Zheng, editors, *Parallel Computing Using Optical Interconnections*, pages 4–21. Kluwer Academic Publishers, 1998.

[70] N. Yoshida Y. Igasaki T.Hara K. K. Japan; N. McArdle M.Naruse M. Ishikawa H.Toyoda, Y. Kobayashi. Compact optical interconnection module for ocular-ii: a pipelined parallel processor. In *Proc. Optics in Computing OC'98 Brugge*, page 204, June 1998.

[71] C. Dragone T. Koch U. Koren A. A. M. Saleh A. J. Kirby C. M. Ozveren B. Schofield R. E. Thomas R. A. Barry D. M. Castagnozzi V. W. S. Chan B. R. Hemenway Jr. D. Marquis S. A. Parikh ¡salil@ll.mit.edu¿ M. L. Stevens E. A. Swanson S. G. Finn I. P. Kaminow, C. R. Doerr and R. G. Gallager. A wideband all-optical wdm network. *IEEE Journal on Selected Areas in Communications*, 14(5), 1996.

[72] Henry Burkhardt III, Steven Frank, Bruce Knobe, and James Rothnie. Overview of the KSR1 computer system. Technical Report KSR-TR-9202001, Kendall Square Research, February 1992.

[73] Th. Ungerer J. Silc, B. Robic. *Processor Architecture From Dataflow to Superscalar and Beyond.* Springer-Verlag, Berlin, Heidelberg, 1999.

[74] J. Jahns. Integrated free space interconnects for chip-to-chip communications. In *Proc. of the MPPOI'98, Las Vegas.* IEEE Computer Society Press, June 1998.

[75] J. Jahns and A. Huang. Planar integration of free space optical components. *Applied Optics*, 28:1602–1605, 1989.

[76] David V. James, Anthony T. Laundrie, Stein Gjessing, and Gurindar S. Sohi. Scalable coherent interface. *IEEE Computer*, 23(6):74–77, June 1990.

[77] A.Huang J.Jahns. Planar integration of free space optical components. *Applied Optics*, 28:4251, May 1990.

[78] J.Jewell, A. Scherer, S.L. McCall, Y.H. Lee, J.P. Harbison, and L.T. Florez. Low-treshhold electrically pumped vertical-cavity surface-emitting microlasers. *Electronis Letters*, 25(17):1123–1124, 1989.

[79] C.T. Retter J.M. Feldman. *Computer Architecture.* McGraw Hill, 1994.

[80] R. Katz, S. Eggers, D. A. Wood, C. Perkins, and R. G. Sheldon. Implementing a cache consistency protocol. *Proceedings of the 12th International Symposium on Computer Architecture*, pages 276–283, June 1985.

[81] Tareef Kawaf, D. John Shakshober, and David C. Stanley. Performance analysis using very large memory on the 64-bit AlphaServer system. *Digital Technical Journal of Digital Equipment Corporation*, 8(3):58–65, December 1996.

[82] K. Thelen M. Moser D. Leipold J. Epler H.P. Schweizer E. Greger P. Riel K.G. Golden, D. Ruffieux. 16 x 16 individually addressable top emitting vcsel array with high uniformity and low threshold voltages. In *6th Topical Meeting of the European Optical Society, Mulhouse.* IEEE Computer Society Press, October 1995.

[83] F. E. Kiamilev, J. S. Lambirth, , and R. G. Rozier. Design of a 64-bit microprocessor core ic for hybrid cmos-seed technology. In *Proc. of the Third International Conference on Massively Parallel Processing Using Optical Interconnections, Maui, Hawai.* IEEE Computer Society Press, October 1996.

[84] David Kranz, Kirk Johnson, Anant Agarwal, John Kubiatowicz, and Beng-Hong Lim. Integrating message-passing and shared-memory: Early experiences. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, pages 54–63, May 1993.

[85] A. V. Krishnamoorthy, J. E. Ford, K. W. Goossen, J. A. Walker, S. P. Hui, J. E. Cunningham, W. Y. Jan, T. K. Woodward, M. C. Nuss, R. G. Rozier, D. A. B. Miller, and F. E. Kiamilev. The amoeba chip: an opto-electronic switch for multiprocessor networking using dense-wdm. In *Proc. of the Third International Conference on Massively Parallel Processing Using Optical Interconnections, Maui, Hawai.* IEEE Computer Society Press, October 1996.

[86] A. V. Krishnamoorthy and K. W. Goossen. Optoelectronic-vlsi: Photonics integrated with electronic circuits (review paper). *IEEE J. Special Topics in Quant. Electr.*, November 1998.

[87] Maria Kufner and Stefan Kufner. *Microoptics and Lithography.* VUBPRESS Brussels, 1997.

[88] S. S. Lee L. Fan and M. C. Wu. Single chip 8x8 optical interconnect using micromachined free-space micro-optical bench technology. In *Proc. of the MPPOI'96, Maui, Hawai.* IEEE Computer Society Press, October 1996.

[89] S. S. Lee L. Y. Lin, J. L. Shen and M. C. Wu. Realization of novel monolithic free-space optical disk pickup heads by surface micromachining. *Optics Letters*, 21(2), January 1996.

[90] Y.A. Akulova J. Ko E.M. Strzelecka S.Y. Hu L.A. Colden, E.R. Hegblom. Vertical cavity lasers for parallel optical interconnects. In *Proc. of the MPPOI'98, Las Vegas.* IEEE Computer Society Press, June 1998.

[91] Produktinformation Laser2000 Gmbh. http://www.laser2000.co.uk/lithium.html, 1998.

[92] James Laudon and Daniel Lenoski. The SGI Origin: A ccNUMA highly scalable server. In *Proceedings of the 24rd Annual International Symposium on Computer Architecture*, pages 241–251, Denver, Colorado, June 2–4, 1997. ACM SIGARCH and IEEE Computer Society TCCA.

[93] Lawrence Livermore National Lb. Product information. http://www.llnl.gov/eng/ee /erd/oedmg/intoptmodulators.html, 1998.

[94] Daniel Lenoski, James Laudon, Kourosh Gharachorloo, Wolf-Dietrich Weber, Anoop Gupta, John Hennessy, Mark Horrowitz, and Monica S. Lam. The Stanford DASH multiporcessor. *IEEE Computer*, 25(3):63–79, March 1992.

[95] Daniel E. Lenoski and Wolf-Dietrich Weber. *Scalable Shared Memory Multiprocessing.* Morgan Kaufmann Publishers, 1995.

[96] Yue Liu. Smart pixel array module based on vcsel to si asic integration,. In *Proc. 1999 OSA Topical Meeting on Spatial Light Modulators, Aspen Colorado*, April 1999.

[97] L.Lamport. How to make a multiprocessor computer that correctly executes multiprocess progra ms. *IEEE Transactions on Computers*, 690, 1979.

[98] L.M.F. Chirovsky and et. al. A high speed optoelectronic chip with 4352 optical inputs/outputs for a 256x256 ATM switching fabric. In *Proc. OSA Topical Meeting on Optical Computing, Sendai.* OSA, April 1996.

[99] Ahmed Louri and Brent Weech. Scalable optical interconnection networks for large-scale parallel computers. In Keqin Li, Yi Pan, and Si Qing Zheng, editors, *Parallel Computing Using Optical Interconnections*, pages 48–74. Kluwer Academic Publishers, 1998.

[100] Bell Labs Lucent Technology. Oevlsi-homepage.

[101] P. Lukowicz. Design of an efficient shared memory architecture using hybrid opto-electronic vlsi circuits and space invariant optical busses. In *Proc. of the Third International Conference on Massively Parallel Processing Using Optical Interconnections, Maui, Hawai*, pages 231–238. IEEE Computer Society Press, October 1996.

[102] P. Lukowicz. Optoelectronic concurrent access memory. *Optical Processing & Computing, Technical Working Group Newsletter*, 8(1), 1997.

[103] P. Lukowicz. High density concurrent access opto-electronic vlsi memory. In *Proc. Optics in Compitng OC'99 Snowmass, Colorado*, page 41. OSA, 1999.

[104] P. Lukowicz, S. Sinzinger, K. Dunkel, and H.-D. Bauer. Design of an opto-electronic VLSI/parallel fiber bus. In *Proc. Optics in Compitng OC'98, Brugge*, pages 289–292. SPIE Vol. 3490, 1998.

[105] P. Lukowicz, S. Sinzinger, K. Dunkel, and H.-D. Bauer. Design of an opto-electronic VLSI/parallel fiber bus. *Journal of the European Optical Society, Part-A: Pure and Applied Optics*, pages 367–370, May 1999.

[106] P. Lukowicz and W. F. Tichy. On the feasibility of a scalable opto-electronic CRCW shared memory. In *Proc. of the Int. Conf. on Algorithms And Architectures for Parallel Processing (ICA3PP-95),Brisbanne.* IEEE, 1995.

[107] P. Lukowicz and W. F. Tichy. Taming massive parallelism: The prospects of opto- electronic CRCW shared memory. In *Proc. PetaFLOPS Frontier Workshop, Frontiers'95*, McLean, VA, February 6–9, 1995. CEDIS Technical Report Series TR-95-161.

[108] P. Lukowicz and W.F. Tichy. A concept for an opto electronic smart pixel based concurrent read concurrent write shared memory. In *Proc. Topical Meeting on Optical Computing, Sendai Japan.* JSAP, April 1996.

[109] Dal Cinand M., Hohland W., T. Dalibor, S.and Eckert, A. Hessenauer Grygier, H., Hildebrand U., J. Hnig, F. Hofmann, C.-U. Linster, E. Pataricza Michel, A. Sieh, V. Thiel, and T. Turowski. Architecture and realization of the modular expandable multiprocessor system memsy. In *Proc. First Intl. Conf. on Massively Parallel Computing Systems (MPCS'94)*, pages 7–15. IEEE, 1994.

[110] A. Neyer M. Johnck, B. Wittmann. 64 channel two dimensional pof based optical array inter chip interconnect. In *Proc. Optics in Computing OC'98 Brugge*, page 285, June 1998.

[111] J. Jahns M. Testorf. Paraxial theory of planar integrated systems. *J. Opt. Soc. Am. A*, 14:1569–1575, 1997.

[112] Barry A. Maskas, Stephen F. Shirron, and Nicholas A. Warchol. Design and performance of the DEC 4000 AXP departmental server computing systems. *Digital Technical Journal of Digital Equipment Corporation*, 4(4):82–99, Fall 1992.

[113] Thomas M. Warschko Matthias M. Mller and Walter F. Tichy. Prefetching on the cray-t3e. In *In Proceedings of the 12th ACM International Conference on Supercomputing*, pages 368–375, June 1998.

[114] A.D. McAulay. *Optical Computer Architectures*. John Willye & Sons, Inc., 1991.

[115] A. McCarthy. Fibre ribbon optical interconnects - the star project. In *Presentation Rank Prize Fund's Mini-Symposium on Ultrafast Photonic Processing and Networks, Grasmere.*

[116] F.B. McCormic, F.A.P. Tooley, J.M.Sasian, J.L.Brubaker, A.L. Lentine, T.J. Cloonan, R.L. Morrison, S.L. Walker, and R.J. Crisci. Parallel interconnection of two 64x32 symmetric selfeelectro-optic effect device arrays. *Electronis Letters*, 27(20):1869–1870, September 1991.

[117] F.B. McCormick, F.A.P. Tooley, T.J.Cloonan, J.L. Brubaker, A.L. Lentine R.L. Morrison, S.J. Hinterlong, M.J. Herron, S.L. Walker, and J.M. Sassian. Experimental investegation of a free-space optical switching network by using symmetric self-electro-optic-effect devices. *Applied Optics*, 31(26):7470–7492, September 1992.

[118] E. McCreight. The Dragon computer system: An early overview. *Xerox Tech. Rep.*, September 1984.

[119] Message-Passing Interface Forum. *MPI-2.0: Extensions to the Message-Passing Interface*, chapter 9. MPI Forum, June 1997.

[120] P. Khurana F. Lacroix A.G. Kirk F.A.P. Toolel D.V. Plant M.H. Ayliffe, D. Kabal. Optomechanical, electrical and thermal packaging of large 2d optoelectronic device arrays for free-space optical interconnects. In *Proc. Optics in Computing OC'98 Brugge*, pages 502–505, June 1998.

[121] Sun Microsystem. Gigaplane-xb: Extending the ultra enterprise family. Whitepaper, Sun Microsystems (http://www.sun.com/datacenter/whitepapers, 1997.

[122] Sun Microsystem. The ultraenterprise 10000 server. Whitepaper, Sun Microsystems (http://www.sun.com/datacenter/whitepapers, 1997.

[123] D. A. B. Miller. Physical reasons for optical interconnection. *Special Issue on Smart Pixels, Int'l J. Optoelectronics*, 3(11):155–168, 1997.

[124] D. A. B. Miller. Dense two-dimensional integration of optoelectronics and electronics for interconnections. In Anis Husain andMahmoud Fallahi, editor, *Heterogeneous Integration: Systems on a Chip*, pages 80–109. SPIE Critical Reviews of Optical Engineering, Vol. CR70 (SPIE, Bellingham, 1998), 1998.

[125] D.A.B. Miller. Quantum wells for optical information processing. *Optical Engineering*, 26(5):369–373, May 1987.

[126] D.A.B. Miller. Hybrid-seed-massively parallel optical interconnections for silicon ics. In *Proc. of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, page 151. IEEE Computer Society Press, 1995.

[127] G.D. Boydand D.A.B. Miller, D.S. Chemia, S.L. McCall, A.C. Gossard, and J.H. English. Multiple quantum well reflection modulator. *Appl. Phys. Lett*, 50(17):1119–854, May 1987.

[128] Motorola Inc. Product Information, 1996.

[129] H. L. Muller, P. W. A. Stallard, and D. H. D. Warren. Implementing the data diffusion machine using crossbar routers. In *Proc. of the 10th Int'l Parallel Processing Symp. (IPPS'96)*, pages 152–158, April 1996.

[130] John Norwood and Shankar Vaidyanathan. Symmetric multiprocessing for PCs. *Dr. Dobb's Journal of Software Tools*, 19(1):80, 82–85, 98–99, January 1994.

[131] A. Nowatzyk, M. Monger, M. Parkin, E. Kelly, M. Browne, G. Aybay, and D. Lee. The S3.mp architecture: A local area multiprocessor. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 140–141, Velen, Germany, June 30–July 2, 1993. SIGACT and SIGARCH.

[132] A.G. Nowatzyk and P.R. Pruncal. Are crossbars really dead ? the cose for optical multiprocessor interconnect systems. In *Proc. 22t International Symposium on Computer Architecture, Gold Coast, Santa Margherita, Italy.* ACM, 1995.

[133] Andreas G. Nowatzyk and Paul R. Prucnal. Are crossbars really dead? the case for optical multiprocessor interconnect systems. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 106–115, Santa Margherita Ligure, Italy, June 22–24, 1995. ACM SIGARCH and IEEE Computer Society TCCA.

[134] University of Glasgow. Internet Project Overview http://eeapp.elec.gla.ac.edu, 1997.

[135] J.M. Hessenbruch R. Kaminski P. S. Guilfoyle, R.V. Stone. Photonic switching demonstration focused towards hpoc module implementation. In *Proceedings of the SPIE, 2400-03, Optoelectronic Interconnects III; San Jose, CA*, February 1995.

[136] R. V. Stone P. S. Guilfoyle, J. M. Hessenbruch. Free-space interconnects for high-performance optoelectronic switching. *IEEE Computer*, February 1998.

[137] Daniel J. Palermo, Eugene W. Hodges IV, and Prithviraj Banerjee. Dynamic data partitioning for distributed-memory multicomputers. *Journal of Parallel and Distributed Computing*, 38(2):158–175, November 1996.

[138] Mark S. Papamarcos and Janak H. Patel. A low-overhead coherence solution for multiprocessors with private cache memories. In *Proceedings of the 11th Annual International Symposium on Computer Architecture*, pages 348–354, Ann Arbor, Michigan, June 5–7, 1984. IEEE Computer Society and ACM SIGARCH.

[139] M. Philippsen, E.A. Heinz, and P Lukowicz. Compiling machine-independent parallel programs. *SIGPLAN Notices*, 31, August 1993.

[140] T. M. Pinkston. The GLORI strategy for multiprocessors: Integrating optics into the interconnect architecture. Technical Report CSL-TR-92-552, Stanford University, Department of Computer Science, December 1992.

[141] T. M. Pinkston and J-H. Ha. The SPEED cache coherence protocol for an optical multi-access interconnect architecture (OMIA). In *Proc. of the Fifth Workshop on Scalable Shared Memory Multiprocessors*, June 1995.

[142] T.M. Pinkston. Design considerations for optical interconnects in parallel computers. In *Proc. of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, page 151. IEEE Computer Society Press, 1994.

[143] F. Pong, A. Nowatzyk, G. Aybay, and M. Dubois. Verifying distributed directory-based cache coherence protocols: S3.mp, a case study. *Lecture Notes in Computer Science*, 966:287–??, 1995.

[144] Jeremy Ekman Fouad Kiamilev Premanand Chandramani, James Rieve. Test chip for flip-chip integration with vcsel and photodetector arrays. In *Proc. 1999 OSA Topical Meeting on Spatial Light Modulators,Aspen Colorado*, April 1999.

[145] K.A. Aly P.W. Dowd, K. Bogineni and J. Perreault. Hierarchical scalable photonic architectures for high-performance processor interconnection. *IEEE Transactions on Computers*, 42:6920, September 1993.

[146] P.W.Dowd and J.Chu. Photonic architectures for distributed shared memory multiprocessors. In *Proc. of the First International Workshop on Massively Parallel Processing Using Optical Interconnections*, page 151. IEEE Computer Society Press, 1994.

[147] Donald Chiarulli Rami Melhem, Greg Gravenstreter and Steven Levitan. The communication capabilities of partitioned optical passive stars networks. In Keqin Li, Yi Pan, and Si Qing Zheng, editors, *Parallel Computing Using Optical Interconnections*, pages 77–97. Kluwer Academic Publishers, 1998.

[148] Abhiram G. Ranade. How to emulate shared memory. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 185–194. IEEE, 1987. Yale University.

[149] Lawrence Rauchwerger, Nancy M. Amato, and David A. Padua. A scalable method for run-time loop parallelization. *International Journal of Parallel Programming*, 23(6):537–576, December 1995.

[150] I. Redmond and E. Schenfeld. A distributed, reconfigurable free-space optical interconnection network for massively parallel processing architectures. In *Proc. Optical Computing 1994 OC'94*, page 373, Edinburgh, 1994.

[151] J. Reisner and T. S. Wailes. A cache coherence protocol for optically connected parallel computer systems. In *Proc. of the 2nd IEEE Symp. on High-Performance Computer Architecture (HPCA-2)*, February 1996.

[152] M.C. Geear B. Shaw A. Briggs P.M. Harrison A. Schmid M. Bitter J. Wieland O. Zorba R.G. Peall, H.F. Priddle and R. Harcourt. September 1996.

[153] Jeffrey Kuskinand David Ofeltand Mark Heinrichand John Heinleinand Richard Simoniand Kourosh Gharachorlooand John Chapinand David Nakahiraand Joel Baxterand Mark Horowitzand Anoop Guptaand Mendel Rosenblum and John Hennessy. The stanford flash multiprocessor. In *In Proceedings of the 21st International Symposium on Computer Architecture*, 1994.

[154] J. Jahns S. Sinzinger. Integrated microoptical imaging system with high interconnection capacity fabricated in planar optics. *Applied Optics*, 36:4729–4735, 1997.

[155] J. Jahns S. Sinzinger. *Microoptics*. Wiley-VCH, Weinheim, 1999.

[156] T. Sakano, T. Matsumoto, K. Noguchi, and T. Sawabe. Design and performance of a multiprocessor system employing board-to-board free space optical interconnections: Cosine-1. *Applied Optics*, 30(17):2334–2342, June 1991.

[157] B.E.A. Saleh and M.C. Teich. *Fundamentals of Photonics*. John Wiley & Sons, INC, 1991.

[158] Sequent. The symmetry technical summary. *Sequent Computer Systems, Inc.*, 1988.

[159] Inc Sequent Computer Systems. Sequent's numa-q architecture. http://www.sequent.com/products/highendsrv/archwp1.html, 1993.

[160] D. Shasha and M. Snir. Efficient and correct execution of parallel programs that share memory. *ACM Transactions on Programming Languages and Systems*, 10(2):282–312, April 1988.

[161] N. Sherwani, Q. Yu, and S. Badida. *Introduction to Multichip Modules*. John Wiley & Sons Inc., 1995.

[162] K. M. Sivalingam. Hybrid Media Access Protocols for a DSM System Based on Optical WDM Networks. In *Proc. of the Fourth IEEE Int'l Symp. on High Performance Distributed Computing*, August 1995.

[163] GS Buller MPY Desmulliez SJ Fancey, MR Taghizadeh and AC Walker. Opto-electronic connection (spoec) project. In *Proc. Optics in Computing OC'98 Brugge*, page 370, June 1998.

[164] P. Stenstrom. A survey of cache coherence schemes for multiprozessors. *COMPUTER*, 23(6):12, June 1990.

[165] V. S. Sunderam. PVM: a framework for parallel distributed computing. *Concurrency, practice and experience*, 2(4):315–339, December 1990.

[166] S. Parikh D. Marquis E. A. Swanson. Network management and control in two deployed wdm all-optical network testbeds. In *ICC'97, WDM Network Management and Control Workshop, Montreal*, 1997.

[167] Ted H. Szymanski. Parallel computing with "intelligent optical networks". In Keqin Li, Yi Pan, and Si Qing Zheng, editors, *Parallel Computing Using Optical Interconnections*, pages 26–44. Kluwer Academic Publishers, 1998.

[168] T.H. Szymanski and H.S. Hinton. Reconfigurable intelligent optical backplane for parallel computing and communications. *Applied Optics*, pages 1253–1268, March 1996.

[169] K. W. Goossen J. A. Walker B. T. Tseng S. P. Hui J. Lothian R. E.Leibenguth T. K. Woodward, A. L. Lentine. Demultiplexing 2.48 gb/s optical signals with a cmos receiver array based on clocked-sense-amplifiers. *IEEE Photonics Technology Letters*, (8), August 1997.

[170] J. P. Hayes T. N. Nudge and D. C. Winsor. Multiple bus architectures. *IEEE Computer*, pages 42–48, June 1987.

[171] M. Jurczyk T. Schwederski. *Verbindungsnetze Strukturen und Eigenschaften.* B.G. Teubner Stuttgart, 1996.

[172] Manu Thapar and Bruce Delagi. Distributed-directory scheme: Stanford distributed-directory protocol. *Computer*, 23(6):78–80, June 1990.

[173] A. Thirumalai, J. Ramanujam, and A. Venkatachar. *Communication Generation and Optimization for HPF*, chapter 29. Kluwer Academic Publishers, 1996.

[174] Mongkol Raksapatcharawong Timothy M. Pinkston and Yungho Choi. Warrp core: Optoelectronic implementation of network router deadlock handling mechanisms. *Applied Optics*, 37(2):276–283, January 1998.

[175] Y. Choi. T.M. Pinkston, M. Raksapatcharawong. Warpp ii: an optoelectronic fully adaptive network router chip. In *Proc. Optics in Computing OC'99 Brugge*, June 1998.

[176] M. Tomasevic and V. Milutinovic. A simulation study of snoopy cache coherence protocols. In *Proceedings of the Hawaii International Conference on System Sciences, Koloa, Hawaii*, pages 426–436, January 1992.

[177] F. Tooley. Optical interconnects do not require improved optoelectronic devices. In *Proc. Optics in Computing OC'99 Brugge*, June 1998.

[178] T. Ungerer. *Parallelrechner und parallele Programmierung.* Spektrum Akad. Verl., 1997.

[179] Stanford University. Home page for the stanford parallel applications for shared memory (splash). http://www-flash.stanford.edu/SPLASH/.

[180] J.M. Vaughan. *The Fabry-Perot Interferometer.* Adam Hilger, Bristol, 1989.

[181] D. V. Plantand B. Robertsonand M. H. Ayliffeand G. C. Boissetand D. Kabakand R. Iyerand Y. S. Liuand D. R. Rolstonand M. Vendittiand and T. H. Szymanski W. M. Robertsonand M. R. Taghizadeh. A multistage optical backplane demonstration system. In *Proc. of the MPPOI'96, Maui, Hawai.* IEEE Computer Society Press, October 1996.

[182] S. Sinzinger W. Eckert, J. Jahns. Design and fabrication of a compact planar-integrated optical correlator. In *Proc. IEEE/LEOS Ann. Meet., San Francisco*, November 1997.

[183] G. Li W. Yuen and C. J. Chang-Hasnain. Multiple-wavelength vertical-cavity surface-emitting laser arrays with a record wavelength span. *IEEE Photonics Technology Letters*, 8(1):4, January 1996.

[184] Hu-Jun Wang and Jian Li. Bandwidth analysis for a class of bus-based systems. *The Computer Journal*, 39(4):346–352, ???? 1996.

[185] C. Waterson and K. Jenkins. Shared-memory optical/electronic computer: architecture and control. *Applied Optics*, 33(8):1559, March 1994.

[186] J. Poulton W.J. Dally M.-J.E. Lee F.-T. An and S. Tell. High-performance electrical signaling. In *Proc. of the MPPOI'98, Las Vegas*. IEEE Computer Society Press, June 1998.

[187] S. C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. The SPLASH-2 programms: Characterization and methodological considerations. In *Proceedings of the 22rd Annual International Symposium on Computer Architecture*, pages 24–36. ACM SIGARCH and IEEE Computer Society, June 1995.

[188] E.E.E. Frietman F. Zhao. Optical multi-faceted free-space image distributor for massively parallel processing. In *Proc. of the MPPOI'98, Las Vegas*. IEEE Computer Society Press, June 1998.

[189] Y. Zhou, L. Iftode, K. Li, J. P. Singh, B. R. Toonen, I. Schoinas, M. D. Hill, and D. A. Wood. Relaxed consistency and coherence granularity in DSM systems: A performance evaluation. In *Proc. of the Sixth ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPOPP'97)*, pages 193–205, June 1997.