Gesellschaft zur Förderung angewandter Informatik e.V. (GFaI)
An- Institut an der Technischen Fachhochschule Berlin und der Fachhochschule für Technik und Wirtschaft Berlin
Mitglied der Arbeitsgemeinschaft industrieller Forschungsvereinigungen 'Otto von Guericke' e.V. (AiF)

Dengel, S., Busch, C., Heinz, G.

Adaptiver Kommunikationsprozessor CP805

Abschlußbericht Projekt CADAS* und Technische Dokumentation

Projektlaufzeit: 1.11.1993 bis 31.3.1995

* Das Projekt 'Entwicklung und Erprobung eines Netzwerkprozessors für massiv parallele Computer mit adaptiv anpaßbarer Schaltungsstruktur (CADAS)'
wurde vom Bundesministerium für Wirtschaft unter Reg. Nr. 37/94 gefördert
Berlin, den 23.6.1995

Projektleitung: Dr. Ing. Gerd Heinz, Rudower Chaussee 5, Geb. 13.7, D-12484 Berlin
Tel. +49 (30) 6392 1624, Fax: -1602
e-mail: heinz@gfai.fta-berlin.de

Inhalt

1. Einführung 4
   1.1. Aufgaben des CP805 5
2. Basis-Algorithmen des CP805 7
   2.1. Zu realisierende Strategien 7
      2.1.1. Deterministischer Algorithmus für Hypercube-Netze 8
      2.1.2. Algorithmus für 2D/3D-Torus-Netze 8
      2.1.3. Adaptiver Algorithmus für Hypercube-Netze 9
   2.2. Kombinierter Algorithmus des CP805 10
3. Funktion des CP805 12
   3.1. Aufgabe und Funktion im Überblick 12
   3.2. Adreßvergleich und Kanalauswahl 13
      3.2.1. Deterministischer Fall 14
      3.2.2. Adaptiver Fall 14
      3.2.3. Adreßvergleich bei Hypercube-Netzen 15
      3.2.4. Adreßvergleich bei 2D/3D-Torusnetzen 16
   3.3. Funktionsprinzip der Kanalauswahl 16
      3.3.1. Router-Baugruppe 17
      3.3.2. Arbiter-Baugruppe 18
      3.3.3. Protokollunabhängige Kanalarbitrierung 20
      3.3.4. Das Single-Stage-Schaltsystem 22
      3.3.5. Protokoll in einer T8xx-Transputerumgebung 23
4. Erprobungsergebnisse 25
   4.1. Timinganalyse der Transputerkarte 25
   4.2. Simulation des CP805 26
      4.2.1. Timing des CP805 28
      4.2.2. Gestaltung einer simulierten VHDL-Testbench 30

5. Vergleich zu Kommunikationsstrategien 31
5.1. Kommunikationsstruktur T805/C004 31
5.2. Kommunikationsstruktur T9000/C104 31
6. Applikative Aspekte 35
6.1. Kommunikation des CP805 mit der Knoten-CPU 35
6.1.1. Portorientierter Zugriff 35
6.1.2. Datenorientierter Zugriff 36
6.1.3. Portorientierter Zugriff über Dual-Kanal-RAM 36
6.1.4. Zugriff auf einen gemeinsamen Speicher 37
6.1.5. Daten im RAM, Sendeaufforderung im FIFO 37
6.2. ATM-Kopplung von Parallelrechnern 38
6.2.1. Zum ATM-Übertragungsverfahren 38
6.2.2. Schaltsysteme für den Anschluß an die ATM-Datenautobahn 40
6.2.3. Chipsätze zum Anschluß an ATM-Highways 42
7. Zusammenfassung 44
8. Literatur 47
9. Anlagen 48

1 Einführung

Auf zahlreichen wissenschaftlichen und technischen Gebieten revolutionieren Computer die Forschung und Entwicklung. Basis dafür ist die rasche Vervollkommnung von Prozessor- und Netzwerktechnologien, welche die Lösung wichtiger Aufgaben ermöglichen. Die Betrachtung kompliziertester Naturphänomene, unter anderem Strömungsanalysen, Wettervorhersage und Crash-Tests von Flugzeugen, ist aus Kostengründen nur mit der Rechnerleistung und Speicherkapazität moderner Supercomputer möglich. In den USA, Japan und Europa sind wichtige Initiativen gestartet worden, welche die momentane Situation beschreiben und die Erfordernisse für die Entwicklung des High Performance Computing (HPC) darstellen.

In Amerika ersetzen verschiedene Anwender ihre Supercomputer durch Cluster von leistungsstarken Workstations. Das erhöht den Geschäftsdruck auf die Hersteller der Supercomputer, die sich nun nach neuen technischen Möglichkeiten umsehen. Die damit verbundene Leistungssteigerung der Computer in den Tera-FLOP-Bereich (Floating Point Operations per second) ist aus technologischen Gründen nur mit Parallelcomputern möglich.

In Europa wurde eine Gruppe von hochrangigen Wissenschaftlern und Industriemanagern unter der Leitung von Professor Carlo Rubbia beauftragt, die wissenschaftlichen und industriellen Erfordernisse und Voraussetzungen für eine europäische HPC-Initiative zu untersuchen. In ihrem 'Report of the EEC Working Group on High Performance Computing' wurde festgestellt:

- Soll Europa eine führende Rolle auf dem Gebiet der Supercomputer einnehmen, ist dafür eine langfristige Strategie zu entwickeln.
- Es sind wissenschaftliche und kommerzielle Anwendungen ausfindig zu machen (ebenfalls ein Problem der amerikanischen HPC-Firmen) und auf Parallelrechnern zu installieren.
- Das Vertrauen in die europäische HPC-Technologie und HPC-Industrie ist herzustellen und zu festigen.
- Die Grundlagenforschung auf dem Gebiet der Parallelrechner muß gefördert werden.
- Wichtig ist die Weiterentwicklung der Netzwerktechnologie.
- Finanziert werden müßte ein solches Projekt mit jährlich etwa 500 Mill. ECU, das entspricht 1 Mrd. DM.

Bei der Entwicklung von modernen massiv-parallelen Computern stehen laut EG-Kommision folgende Fachgebiete im Vordergrund:

- Hochparallele Prozessoren mit schnellen Kommunikationsverbindungen,
- Software und Algorithmen für ihren Betrieb,
- Schnittstellen und Peripherieanlagen,
- Hochleistungfähige Kommunikationsnetzwerke.

Parallelrechner, bestehend aus einem Netz von Parallelprozessoren, ist ein leistungsfähiger Verbund. Normalerweise könnte man denken, je mehr Parallelprozessoren sich eine Aufgabe teilen, um so schneller wird diese Aufgabe bewältigt. Das ist jedoch ein Irrtum, denn je mehr Parallelprozessoren kommunizieren, um so größer wird die Menge der Zusatzinformationen, die benötigt werden, um die Kommunikation selbst zu koordinieren. Es muß also eine günstige Arbeitsweise gefunden werden, die den Kommunikationsoverhead in Grenzen hält.

Gesucht wird eine Lösung, die mit wenig Steuerungsaufwand einen hohen Datendurchsatz ermöglicht. Als günstig ist hier eine dezentrale Steuerung des Netzes anzusehen.

Parallelrechner werden zur Lösung von numerischen Aufgaben, in der Bildverarbeitung und anderen Bereichen herangezogen. Die vorhandenen sequentiellen Algorithmen müssen dem angepaßt oder neue Algorithmen für Parallelrechner entworfen werden. Der Vorgang der Parallelisierung ist notwendig, damit Programme den gewünschten Zeitgewinn durch Verteilung auf die Parallelrechner erbringen.

Man kann davon ausgehen, daß die einzelnen verteilten Programmteile Daten und Ergebnisse untereinander austauschen. Daraus erwächst die Aufgabe, die anfallenden Datenströme so durch das Rechnernetz zu transportieren, daß möglichst unabhängig vom Grad und der Belastung des Netzes keine Störungen im Informationsaustausch auftreten. Der Übertragungsalgorithmus muß garantieren, daß trotz eventuell vorhandener Prioritäten alle Kommunikationsanforderungen nach einer bestimmten Zeit bedient werden können.

Das Ziel des Projektes ist es, einen Netzwerkprozessor für massiv-parallele Rechner zu entwickeln, dessen Funktion eindeutig mit parallelen Automatennetzwerken beschrieben werden kann. Er soll die Kommunikation zwischen den Parallelprozessoren steuern. Die parallelen Automatennetzwerke garantieren eine konstante Zeit für jede Kommunikationsentscheidung. Es soll eine intelligente Einrichtung geschaffen werden, welche die Datenströme unabhängig von Grad und Belastung des Hypercubusnetzes verklemmungsfrei steuert.

1.1 Aufgaben des CP 805

Der Kommunikationsprozessor CP 805 ist ein Router für massiv-parallele Rechner. Mehrere CP 805 bilden zwischen den einzelnen Parallelknoten ein Kommunikationsnetz. Ein Parallelknoten besteht aus einem CP 805 und einem Transputer T 8xx. Benachbarte Knotenprozessoren werden von der Datenweitergabe entlastet. Durch den CP 805 werden 2D-, 3D- und Hypercubus-Netze unterstützt. Die Steuerung des CP 805 erfolgt in Abhängigkeit des Datenstromes dezentral. Die Linkarbitration innerhalb des Kommunikationsnetzes erfolgt wahlweise deterministisch oder adaptiv. Der CP 805 unterstützt das serielle Transputerprotokoll T 8xx. Der Arbitrationslayer des CP 805 ist unabhängig vom Protokoll und kann in unterschiedlichen Protokolltypen eingesetzt werden. Das Routing erfolgt im innerhalb von Nanosekunden. Physisch ist der CP 805 in einem FPGA XC 4010 integriert.

Die im CP 805 verwendeten Algorithmen basieren auf effizienten deterministischen und adaptiven packet-routing Algorithmen (dynamischer Lastausgleich) für 2D-, 3D- und Hypercubus-Netzwerke von Jossifov und Krapp. Sie führen auf den deterministischen 'e-cube-routing' Algorithmus von Hayes und Thuazon und adaptive Strategien nach Jesshope u.a. zurück. Die verwendeten Algorithmen weisen eine vergleichbare Struktur, Konnektivität, ähnliche Kantengewichte und Zustandsfunktionen auf. Sie sind für deterministische und adaptive Netzwerkprozessoren zur verklemmungs- und konfliktfreien Steuerung von 2D-, 3D- und Hypercubus-Netzen untersucht worden.

Eine Analyse zeigte, daß die untersuchten Algorithmen auf eine effiziente Grundkonstruktion hinführbar und damit als Basis für einen Netzwerkprozessor geeignet

sind. Die Umsetzung in einen PLD-Schaltkreis schafft die Möglichkeit, massiv-parallele Rechner in unterschiedlichen Netzwerkstrukturen zu verschalten, in dem der Schaltkreis mehrere Kommunikationsalgorithmen und Netzstrukturen unterstützt. Der CP 805 kann für die Netztypen 2D-, 3D-Netz und nD-Hypercubus eingesetzt werden.

Der Kommunikationsprozessor wird als asynchrones, paralleles Schaltsystem mit synchronem Verhalten realisiert. Der Vorteil besteht in der konstanten Zeit, die für jede beliebige Kommunikationsentscheidung unabhängig vom Grad des Netzwerkes notwendig ist. Sie entspricht der Summe der Zustandssequenzen von zwei Automaten, die die Steuerung des Arbitrationslayers bilden.

Die Innovation des Kommunikationsprozessors CP 805 gegenüber herkömmlichen Lösungen besteht in der reinen, extrem schnellen Hardwarelösung ohne Mikroprogramm usw., auf Basis von Automatengraphen. Der Arbitrationslayer des CP 805 ist vollständig in synthetisierbarem VHDL-Quelltext beschrieben.

Die Kanalarbitration erfolgt für alle Links zeitgleich und ist in Abhängigkeit der Anzahl der Links skalierbar. Werden bestimmte Schnittstellenbedingungen eingehalten, kann die Kanalarbitration für andere Protokolle eingesetzt werden. Die Kanalarbitration ist mit synthetisierbarem VHDL beschrieben. Die erreichte Verzögerungszeit reicht für ein Adressfeld nicht über 40 Gate-Delays, das entspricht einem Byte, hinaus. Die Latenz bei einer geschalteten Verbindung entspricht ca. 5 Gatterlaufzeiten.

Ein n-dimensionaler Hypercubus besteht aus 2^n Knoten, die als n-dimensionale Konstruktion so zusammengeschaltet sind, daß jeder Knoten n Nachbarn besitzt. Die Nummer jedes Knoten kann als n-bit Binärstring dargestellt werden. Die Knotennummern benachbarter Knoten unterscheidet sich in nur einer Bitposition, die Hammingdistanz ist also eins.

Ein dreidimensionales Netzwerk besteht aus (x^n*x^n*x^n) Knoten, die als dreidimensionales Feld verbunden sind. Jede Achsrichtung besitzt n Knoten, die fortlaufend mit einer Integerzahl für jede Achsrichtung versehen ist.

Es werden starre Netzwerk-Topologie (z.B. INMOS C004) überwunden. Die Verzweigung der Kommunikationsflüsse erfolgt ereignisgesteuert. Mit dem Kommunikationsprozessor ist dynamische Clusterbildung und -veränderung möglich. Die Cluster sind online strukturierbar. Zwischen Transputer und CP 805 existiert ein Sende- und ein Empfangskanal.

Trotz unterschiedlicher Adreßdarstellung (integer bzw. binär) erfolgt die Adreßrechnung in 2D/3D-Netzen und Hypercubus nach den gleichen Prinzipien. Die verwendeten Algorithmen gestatten deterministisches ebenso wie adaptives Routing. Die Wahl des Algorithmus und des Netztyps erfolgt online und ist programmierbar.

Die statische Leitungsbelegung bei stationärer Verbindung kann zu Dead Locks führen. Als Ausweg wurde Paketbetrieb mit Längenbegrenzung der Botschaften gewählt. Er hat die Eigenschaft, daß Standleitungen aufgebaut werden. Das aufwendige softwaremäßige sortieren von Paketen unterbleibt. Das Virtual Channel Prinzip, es können beliebig viele logischen Kanälen parallel benutzt werden, ist durch das Paketverfahren dem CP 805 eigen. Bedingt durch die dezentrale Steuerung des CP 805 werden hot spot's im Routing nicht beachtet. Die Zusammenschaltung der CP 805 erfolgt mittels bidirektionaler serieller Doppelleitung.

Der Prototyp des CP 805 arbeitet mit einer Taktfrequenz von 20 MHz. Die Datenrate je Link ist 20 MBit/s entsprechend dem unterstützten T8xx-Protokoll.

2 Basis-Algorithmen des CP805

Ausgangspunkt der im CP805 implementierten Router-Struktur sind die in [Joss91], [Krapp91-1] und [Krapp91-2] beschriebenen Algorithmen deterministischer und adaptiver Art für n-dimensionale Hypercube-(HC-) Netze bzw. für 2- und 3-dimensionale Torus-Netz

2.1 Zu realisierende Strategien

Kernstück dieser Algorithmen ist ein Nummerierungsschema, das dem lokal entscheidenden Kommunikationsprozessor (CP) erlaubt, lokal durch einfache Vergleichsoperationen mit der eigenen Knotennummer oder durch relative Reduzierung einer n-dimensional mit der Nachricht mitkommenden Adresse entsprechend der Richtung (Dimension) das Ausgabepin festzulegen, von dem aus ein weiterzureichendes Datenpaket sein Ziel erreichen kann.

Im Kommunikationsprozessor sind mehrere Pfade parallel zueinander, d.h. zeitgleich durchschaltbar. Entsprechend dem verwendeten Kommunikationsprotokoll vom INMOS-T800-Transputertyp besteht ein Pfad aus zwei Leitungen. Diese dienen der Datenübermittlung vom Quell- zum Zielknoten und der Statusübermittlung vom Ziel- zum Quellknoten. Die Verbindung zwischen den einzelnen Netzwerkknoten erfolgt bitseriell.

Der Pfad wird dynamisch zur Übertragungszeit aufgebaut und während der Übertragung gehalten. Zunächst wird der gesamte Datenpfad schrittweise beginnend vom Quellknoten aufgebaut. Dazu wird ein Header der Botschaft vorausgeschickt. Dabei fügt jeder Knoten nach erfolgter Arbitration ein Pfadsegment hinzu.

Die verschiedenen Arbitrationsstrategien werden im einzelnen nachfolgend beschrieben. Falls mehrere Eingänge simultan einen Ausgang benötigen, hat der Kommunikationsprozessor hat eine korrekte Konfliktlösung zu sichern.

Mit dem Empfang des Headers im Zielknoten startet der Bestätigungsprozeß. Eine Bereit-Botschaft (RDY) wird zum Quellknoten zurück gesendet. Sie folgt exakt dem Pfad in entgegengesetzter Richtung. Erreicht das Bereit-Signal den Quellknoten, startet dieser die Datenübertragung.

Der Pfad wird abgebaut, wenn das Ende der Botschaft, gebildet durch ein Ende der Nachricht-Wort (EOM), den Kommunikationsprozessor passiert. Es weist den Kommunikationsprozessor an, den Kanal zum folgenden Knoten freizugeben. Die Kanäle lösen 'on the fly' aus. Im Zielknoten wird das EOM-Wort aus der Botschaft entfernt, da es nicht Teil der Originalbotschaft ist. In der Ziel-CPU terminiert der Übertragungsprozeß.

Dieser Typ des Routing ist eine Form des Worm-Hole-Routing mit dem Unterschied, daß unabhängig von der Botschaftslänge die Botschaft übertragen wird, nach dem der Pfad errichtet wurde. Damit kann das System vollständig synchron arbeiten, ohne die Botschaft im vermittelnden Kommunikationsprozessor zwischenzuspeichern. Vorteil dieses Algorithmus ist, daß der laufende CPU-Prozeß im vermittelnden Knoten nicht unterbrochen wird.

2.1.1 Deterministischer Algorithmus für Hypercube-Netze

Der Routingprozeß basiert auf dem e-cube Routingalgorithmus [Lang], [Sull], der ein verklemmungsfreies (deadlock-free) Netzwerk garantieren soll.

Die Knoten des Hypercubus bekommen eine einheitliche Adresse zugewiesen, die sich bei benachbarten Knoten um ein digit unterscheidet. Die Kanalverbindungen der Nachbarn sind durch ihre korrespondierende Dimension festgelegt. Sie wird durch ein XOR der anliegenden Knotennummern ermittelt.

Beispiel:

Kanal 1 verbindet Knoten 5 und Knoten 7

$111 \otimes 101 = 010$

Die 1 an der Bitposition eins im Ergebnisstring bezeichnet Kanal 1.

Um das garantierte deadlock freie Verhalten zu garantieren, darf die Botschaft nur in aufsteigender Dimension der Kanalverbindungen bis zum Zielknoten geroutet werden. Routing zu niedriger numerierten Kanälen sind nicht erlaubt. Der Header enthält die Adreßinformation die benötigt wird, um den Pfad den die Nachricht nehmen soll, zu errichten. Der Header bildet sich aus der XOR-Verbindung vom Ziel- und Quellknotenadresse.

Beispiel:

Nachricht vom Knoten 2 zum Knoten 4

$100 \otimes 010 = 110$

Jedes Bit im Ergebnisstring 110 ist einem Kanal zugewiesen. Die von rechts beginnend erste 1 kennzeichnet Kanal 1 des Quellknoten (Knoten 2) als Teil des Pfades. Bevor der Header zum nachfolgenden Knoten gesendet wird, ist der Header an der Bitposition des korrespondierenden Kanals auf 0, durch die Verknüpfung 110 ^101 = 100, zu maskieren. Das Resultat, der neue Header 100, wird über Kanal 1 zum Knoten 0 gesendet. Hier definiert der Header Kanal 2. Der Header wird mit 100^011=000 maskiert, so daß der Header 000 den Knoten 4 erreicht. Der Header hat das Ziel erreicht, wenn alle Bits im Header auf Null gesetzt sind.

2.1.2 Algorithmus für 2D/3D-Torus-Netze

Der vorgestellte Algorithmus versucht, die die Vorteile des 'Wormhole Routing' und die adaptive Strategie des 'Mad Postman'-Typs miteinander zu kombinieren.

Die Kanalverbindungen der Nachbarn sind durch ihre virtuelle Lage (up, down, north, east, south, west) festgelegt. Der Header enthält die Adreßinformation des Zielknoten. Im Kommunikationsprozessor wird aus dem Header die relative Lage des Ziels ermittelt. Die Vergleichsergebnisse werden in folgende Klassen eingeteilt:

1	xy-base, up	x = c	y = c	+z
2	xy-base,down	x = c	y =c	-z
3	east, north	+x	-y	z = c
4	east, south	+x	+y	z = c
5	west, south	-x	+y	z = c
6	west, north	-x	-y	z = c
7	processor	x = 0	y = 0	z = 0

Tab. 1 Klassen der Richtungsanforderungen im 3D-Netz

Vor der Vermittlung in die angegebene Richtung wird die Adresse absolut dekrementiert. Der Header hat das Ziel erreicht, wenn alle Bits im Header auf Null gesetzt sind. Die wirkliche Richtung ist abhängig von der Arbeitslast des physischen Kanals, auf den die virtuelle Klasse gemappt wird, sowie von der Priorität

P(x,y,z): u then d then n then e then s then w then p.

2.1.3 Adaptiver Algorithmus für Hypercube-Netze

Die Knoten des Hypercubus (HC) bekommen eine einheitliche Adresse zugewiesen, die sich bei benachbarten Knoten um ein digit unterscheidet. Die Kanalverbindungen der Nachbarn sind durch ihre korrespondierende Dimension festgelegt. Sie wird durch ein XOR der anliegenden Knotennummern ermittelt.

Beispiel:

Kanal 1 verbindet Knoten 5 und Knoten 7

$111 \otimes 101 = 010$

Die 1 an der Bitposition eins im Ergebnisstring bezeichnet Kanal 1.

Die Botschaft kann in beliebiger Dimension der Kanalverbindungen bis zum Zielknoten geroutet werden. Der Header enthält die Adreßinformation, die benötigt wird um den Pfad den die Nachricht nehmen kann, zu errichten. Der Header bildet sich aus der XOR-Verbindung vom Ziel- und Quellknotenadresse.

Beispiel:

Nachricht vom Knoten 2 zum Knoten 4

$100 \otimes 010 = 110$

Jedes Bit im Ergebnisstring 110 ist einem Kanal zugewiesen. Im Knoten 2 sind also die Kanäle 1 und 2 als Ausgänge möglich. Im Beispiel soll Kanal 1 gewählt sein. Bevor der Header gesendet wird, ist dieser an der Bitposition des korrespondierenden Kanals auf 0 durch die Verknüpfung 110 ^101 = 100 zu maskieren. Das Resultat ist ein neuer Header 100. Er wird über Kanal 1 zum Knoten 0 gesendet. Im Knoten 0 definiert er Kanal 2. Der Header wird in Knoten 0 mit 100^011=000 maskiert, so daß der Header 000 den Knoten 4 erreicht. Der Header hat das Ziel erreicht, wenn alle Bits im Header auf Null gesetzt sind.

Die Vermittlung erfolgt in eine der angegebenen Richtungen. Die wirkliche Richtung ist abhängig von der Arbeitslast des physischen Kanals und von der Dimension der möglichen Kanalverbindungen. Es gilt die Priorität P(Dimension) => dim then dim-1 then dim-2 ... then 0 then p.

2.2 Kombinierter Algorithmus des CP 805

Der Routingprozeß basiert auf einer Zusammenfassung der Algorithmen für deterministischen und adaptiven Hypercubus und dem 3D-Algorithmus.

Im Sendeprozeß sind das Verfahren und der Netzwerktyp mit dem die Botschaft transportiert wird einstellbar. Dem Header wird diese Information zusätzlich zur Zieladresse mitgegeben. Jeder Kommunikationsprozessor ließt diese Informationen und interpretiert den Header in deren Abhängigkeit.

Die Knoten des CP 805 bekommen eine einheitliche Adresse zugewiesen, die sich bei benachbarten Knoten um ein digit unterscheidet. Im Betrieb eines Knoten des 3D-Netzes wird sie als (x,y,z)-Tupel interpretiert.

Für der Betrieb als Hypercubus sind die Kanalverbindungen der Nachbarn durch ihre korrespondierende Dimension festgelegt. Sie wird durch ein XOR der anliegenden Knotennummern ermittelt.

Beispiel:

Kanal 1 verbindet Knoten 5 und Knoten 7

$111 \otimes 101 = 010$

Die 1 an der Bitposition eins im Ergebnisstring bezeichnet Kanal 1. Für die Maskierung gilt: Es wird die Bitposition im Header wird auf 0 maskiert die der Binärdarstellung des Ausgangskanals entspricht.

Der Header enthält die Adreßinformation die benötigt wird um den Pfad den die Nachricht nehmen soll zu errichten. Der Header bildet sich aus der XOR-Verbindung vom Ziel- und Quellknotenadresse.

Beispiel:

Nachricht vom Knoten 2 zum Knoten 4

$100 \otimes 010 = 110$

Jedes Bit im Ergebnisstring 110 ist einem Kanal zugewiesen.

Während im deterministischen Hypercubus von rechts beginnend erste 1 den Ausgangskanal kennzeichnet, Kanal 1 im Knoten 2, sind im adaptiven Fall alle mit 1 gekennzeichneten Kanäle (im Knoten 2: die Kanäle 1 und 2) als Ausgänge möglich. Im deterministischen Betrieb darf die Botschaft nur in aufsteigender Dimension der Kanalverbindungen bis zum Zielknoten geroutet werden. Routing zu niedriger numerierten Kanälen sind nicht erlaubt.

Im Gegensatz dazu der adaptive Hypercubus. Hier kann die Botschaft in beliebiger Dimension der Kanalverbindungen bis zum Zielknoten geroutet werden. Die wirkliche Richtung ist abhängig von der Arbeitslast des physischen Kanals und von der Dimension der möglichen Kanalverbindungen. Es gilt die Priorität P(Dimension) => dim then dim-1 then dim-2 ... then 0 then p. Für das Beispiel soll Kanal 1 gewählt sein.

In beiden Fällen ist der Header an der Bitposition des korrespondierenden Kanals auf 0 zu maskieren. Das Resultat der neue Header 100. Er wird über Kanal 1 zum Knoten 0 gesendet. Im Knoten 0 definiert er Kanal 2. Der Header wird in Knoten 0 auf 000 maskiert, so daß der Header 000 den Knoten 4 erreicht. Der Header hat das Ziel erreicht, wenn alle Bits im Header auf Null gesetzt sind.

Im 3D-Netz sind die Kanalverbindungen der Nachbarn sind durch ihre virtuelle Lage (up, down, north, east, south, west) festgelegt. Der Header enthält die Adreßinformation des Zielknoten. Er wird als (x,y,z)-Tupel durch den Sendeprozeß gebildet. Im Kommunikationsprozessor wird Adreß- und Headertupel verglichen und die relative Lage des Ziels ermittelt. Die Vergleichsergebnisse werden in folgende Klassen eingeteilt:

1	xy-base, up	x=c	y=c	+z
2	xy-base,down	x=c	y=c	-z
3	east, north	+x	-y	z=c
4	east, south	+x	+y	z=c
5	west, south	-x	+y	z=c
6	west, north	-x	-y	z=c
7	processor	x=0	y=0	z=0

Tab. 2 Klassen der Richtungsanforderungen im 3D-Netztyp des CP 805

Die Vermittlung erfolgt in eine der angegebenen Richtungen. Sind Knoten- und Headeradresse gleich, ist der Zielknoten erreicht. Die wirkliche Richtung ist abhängig von der Arbeitslast des physischen Kanals auf den die virtuelle Klasse gemappt wird, und von der Priorität

P(x,y,z): u then d then n then e then s then w then p.

3 Funktion des CP 805

3.1 Aufgabe und Funktion im Überblick

Der über die serielle DATAin-Leitung kommende Header wird in der Baugruppe Link synchronisiert und in einer Registerpipe parallel gespeichert. Dem Header werden die Informationen über Kommunikationsverfahren (deterministisch/adaptiv), Netztyp (HC/2D/3D) und das Adreßwort an den entsprechenden Bitpositionen entnommen. Sie werden der Compare-Baugruppe übergeben. Hier wird je nach Verfahren und Netztyp Adreßbits und Knotennummer verglichen und das Vergleichsergebnis (request-Vektor) der Kanalarbitrierung zugeführt. Sind Adreßbits und Knotennummer gleich, ist der Zielknoten erreicht. Die Kanalarbitrierung wählt in Kenntnis der Belegungszustände aller Links einen Ziellink aus. Die MUX-Baugruppe bewirkt die bidirektionale Zusammenschaltung von Quell- und Zielkanal. Anschließend wird der Header zum folgenden Kommunikationsprozessor (DATAout) gesendet. Mit dem EOM-Signal, der CP 805 erzeugt es durch zählen der Datenwörter, löst die Verbindung aus.

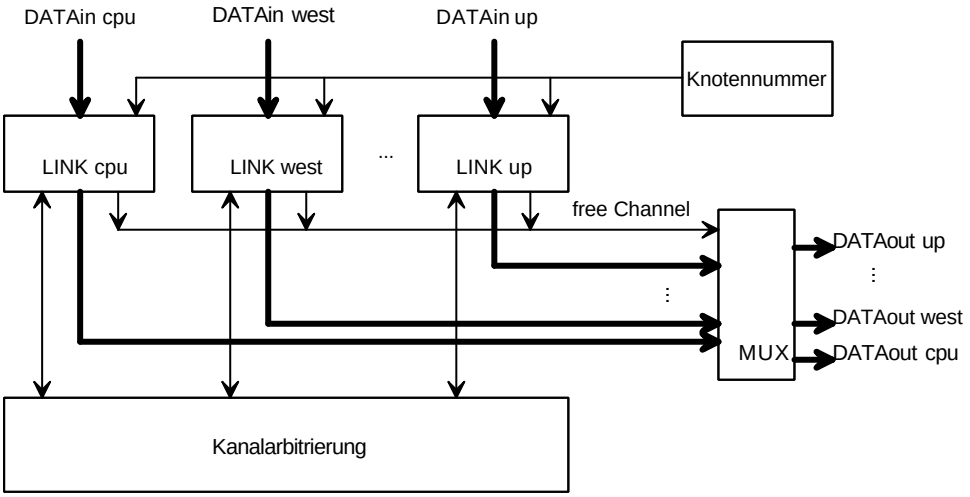
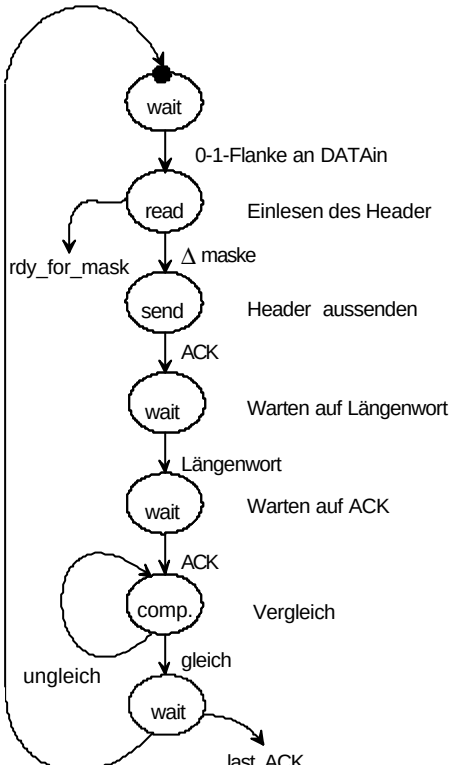
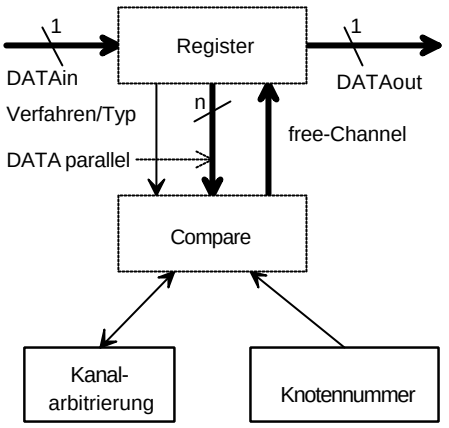


Abb. 3 Blockschaltbild Kommunikationsprozessor CP 805

Im Gegensatz zum adaptiven Verfahren ist der Weg einer Nachricht durch das Kommunikationsnetz beim deterministischen Verfahren vorbestimmt.





Blockschaltung eines Ports

Abb. 4 Automatengraph für den Verbindungsaufbau. Unabhängig von der Art des zu routenden Netzes (2D/3D-Torus, nD-HC) wird ein vergleichbarer Automatengraph abgearbeitet.

Mit der 0-1-Flanke an DATAin beginnt das Einlesen des Headers. Die Bitsynchronisation und Bitzählung übernimmt die Baugruppe selbst. Nach dem Einlesen erscheint das 'rdy_for_mask'-Signal, welches der Compare-Baugruppe zugeführt wird. Zeitgleich stehen an p(7:0) die parallelgewandelten Header-Daten zur Verfügung. Die Baugruppe wartet auf einen High-Impuls, der nach erfolgter Kanalarbitration auf einer der Leitungen maske(6:0) erscheint. Er bewirkt das Herausschieben des Headers sowie einen Bypass vom Eingang zum arbitrierten Ausgang. Die Bypass-Schaltung beschleunigt den Datentransport. Die Register- Baugruppe wartet aus des ACK vom Zielprozessor. Mit seinem Empfang kann das Längenwort durch alle am Verbindungsweg liegenden CP's gelesen werden. Damit ist der Verbindungszustand erreicht und der Datenverkehr findet ausschließlich zwischen den beteiligten Prozessoren statt. Alle am Übertragungsweg liegenden CP's zählen die übermittelten ACK und vergleichen mit dem Längenwort. Mit dem letzten ACK wird das last_ackn-Signal gebildet und der Kanalarbitration zugeführt. Diese löst daraufhin aus.

3.2 Adreßvergleich und Kanalauswahl

Mit dem CP 805 werden 2D/3D- und HC-Netze in einer Schaltung unter Zusammenführung unterschiedlicher Adreßverfahren vereinigt.

Zur Adreßbildung wird ein mit dem Header transportiertes Adreßwort und eine jedem Netzknoten zugeordnete Knotennummer benutzt. Die Adreßdarstellung im 2D/3D-Netz erfolgt mittels binär kodierten Integerwerten. Im Hypercubus-Netz wird die Adresse mittels eines Binärstrings dargestellt.

Durch eine geeignete Interpretation von Adreßwort und Knotennummer können prinzipiell unterschiedliche Adreßformen in einer Schaltung so ausgewertet werden, daß ein einheitliches Format zur Kanalauswahl (-arbitrierung) entsteht, mit dem die Arbitrationsalgorithmen zusammenarbeiten. Grundsätzlich müssen die Adressen (Hypercubus) bzw. Adreßtupelelemente (2D/3D) quantitativ bewertet werden, d.h. die Frage in welcher Richtung das Ziel liegt, und welche Links als Ausgang in Frage kommen, muß beantwortbar sein. Anschließend wird aus der Menge der freien, auf dem Weg zum Ziel liegenden Links genau ein Link ausgewählt. Sind mehrere Links frei, entscheidet eine Priortätsreihenfolge.

Priorität	n	.....	6	5	4	3	2	1
2D-Torus	-	-	-	-	north	east	south	west
3D-Torus	-	-	up	down	north	east	south	west
n-dim.HC	$Kanal_n$	.....	$Kanal_5$	$Kanal_4$	$Kanal_3$	$Kanal_2$	$Kanal_1$	$Kanal_0$

Tab. 5 Prioritätsreihenfolge bei der Abarbeitung der Kanäle

3.2.1 Deterministischer Fall

Soll die Botschaft einen bestimmten Weg nehmen, ist der Adreßvergleich so auszuführen, daß nur der entsprechende Ziellink gekennzeichnet wird. Einheitlich gilt nach dem Multiplexer die Zuordnung: $request_i$ fordert $Link_i$ an, $request_0$ fordert $Link_p$ an usw.

Die Compare-Baugruppe übernimmt die Informationen Adreßwort, Verfahren und Netztyp von der Register-Baugruppe. Für jeden Netztyp und jedes Verfahren wird das Adreßwort mit der Knotennummer des Kommunikationsprozessors verglichen. Der nachgeschaltete Multiplexer, der über die Informationen Netztyp und Verfahren gesteuert wird, wählt ein Vergleichsergebnis aus und übergibt es als binären Vektor der Kanalarbitrierung (request).

3.2.2 Adaptiver Fall

Ist mindestens ein Ziellink frei, sendet die Kanalarbitrierung das free-Signal. Es steuert den 1:n-Decoder, der entsprechend der Prioritätsreihenfolge aus den freien Zielkanälen genau einen Zielkanal auswählt (free Channel).

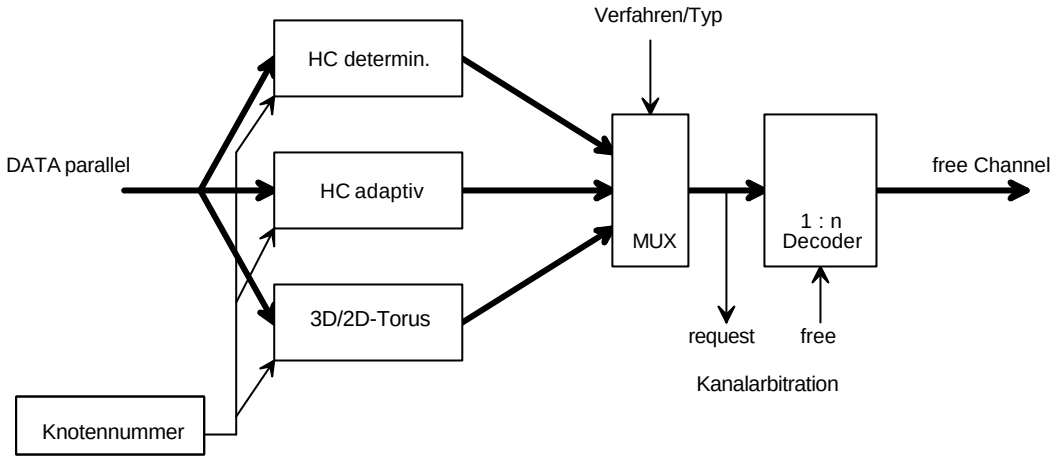


Abb. 6 Blockschaltbild eines Adreßvergleichers (Compare-Block)

3.2.3 Adreßvergleich bei Hypercube-Netzen

Im Hypercube-Netz werden die Links mit einer Linknummer versehen. Sie bildet sich aus einer XOR der anliegenden binären Knotennummern. Es wird die Eigenschaft der Hypercube ausgenutzt, daß benachbarte Knoten sich in nur einer Bitposition unterscheiden. Das Bit i in der Knotennummer entspricht dem Link i.

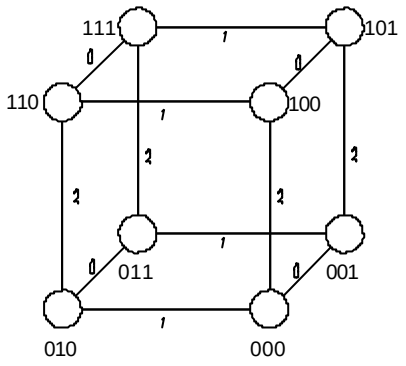
Beispiel:

Linknummer zwischen Knoten 0 und Knoten 1:

$001 := 000 \otimes 001$

Im Ergebnis wird die Position 0 gesetzt, der verbindende Link von Knoten 0 und 1 besitzt die Linknummer 0.

Wird jedem Link ein Bit im Adreßwort und in der Knotennummer zugeordnet, kann ermittelt werden, ob in dieser Linkrichtung das Ziel liegt. Ein XOR von Adreßwort und Knotennummer bildet den HC-Requestvektor. Für ein 3D-HC-Netz ergibt sich:



Beispiel einer Verbindung von Knoten 010 nach 100:

- *im Knoten 010:*
  010 Ä 100 Þ 110, die möglichen Richtungen sind Link 1 oder 2
- *im Knoten 000:*
  000 Ä 100 Þ 100, die mögliche Richtung ist Link 2
- *im Knoten 110:*
  110 Ä 100 Þ 100, die mögliche Richtung ist Link 1
- *im Knoten 010:*
  100 Ä 100 Þ 000, die mögliche Richtung ist Link p

Abb. 7 Zur Adreßbildung beim 3D-Hypercube

Die Baugruppe HC-deterministisch sucht aus dem parallelen Adreßvektor (DATAparallel) das kleinste gesetzte Bit heraus. Die Baugruppe HC-adaptiv gibt die Belegung des Adreßvektors direkt weiter. Ist kein Bit des Adreßvektors gesetzt, ist der Zielknoten erreicht und der Link p ist Ziel. Damit wird jeder Link eindeutig identifiziert. Beim aussenden des Headers zum folgenden CP wird das mit der Linknummer korrespondierende Bit auf Null gesetzt (Bit-Erosion).

Größere Netze sind mit entsprechendem Aufwand ohne Veränderung der Kommunikationsalgorithmen routbar.

3.2.4 Adreßvergleich bei 2D/3D-Torusnetzen

Die Knoten des 2D-Netzes werden mit einem (x,y)-Tupel, 3D-Netze mit einem (x,y,z)-Tupel numeriert. Die Tupelelemente werden durch Integerwerte repräsentiert. Die Vergleichsergebnisse von Adreßwort und Knotennummer entstehen laut der Tabelle:

	Richtung	up	down	north	east	south	west	node-CPU
$X_h > X_k$	+x				1			
$X_h < X_k$	-x						1	
$Y_h > Y_k$	+y					1		
$Y_h < Y_k$	-y			1				
$Z_h > Z_k$	+z	1						
$Z_h < Z_k$	-z		1					
Kanal p	x = y = z = 0							1

Anmerkg.:
X, Y, Z: Richtungen
Index h: Adreßwort
Index k: Knotennummer

Tab. 8 Richtungszuordnung im 3D-Netz

Die OR-Verknüpfung der Richtungssignale bildet den Requestvektor. Die Breite der vergleichenden Baugruppe ist allein von der maximalen Linkanzahl abhängig. Eine 2D-Netzkonfiguration besitzt die Ports *north, east, south, west*. Eine 3D-Netzkonfiguration besitzt die Ports *up, down, north, east, south, west*. Der

n-dimensionale HC besitzt indessen *n* Links zu benachbarten CP, demzufolge ist der Requestvektor *n* Stellen breit.

3.3 Funktionsprinzip der Kanalauswahl

Die Kanalauswahl (Kanalarbitrierung) ist die zentrale Einrichtung des CP 805, die die Suche nach freien Links organisiert. Jeder Link besitzt einen Automaten zur Arbitrationssteuerung (Router) und einen Zustandsspeicher (Arbiter). Der Router stellt bei Linkanforderungen den Belegungszustand der angeforderten Arbiter fest und sperrt diesen gegen Doppelbenuztung. Der Arbiter zeigt den Belegungszustand des zugeordneten Links an.

Der Router besteht aus einzelnen Automaten, die den Ablauf zu einem Arbiter steuern (Routerästen). Jedes Element des Requestvektors ist einem Routerast zugeordnet. Trifft in Auswertung eines Headers ein Requestvektor ein, erkennt der Routerast beim zugeordneten Arbiter, ob der Link frei ist. Sollte das der Fall sein, gibt der Arbiter eine frei-Meldung an den Routerast. Dieser reicht sie zur Compare-Baugruppe (Steuerung des 1:n-Demux) weiter. Mit EOM, dem Botschaftsende, löst die Kanalarbitration aus und gibt den Link frei. Durch Hinzufügen weiterer Arbiter und Routeräste ist die Kanalarbitration skalierbar.

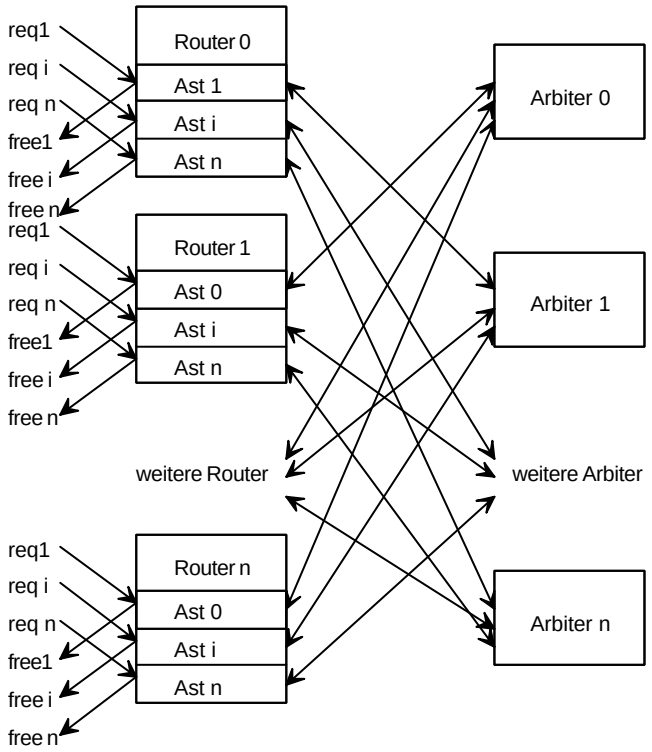
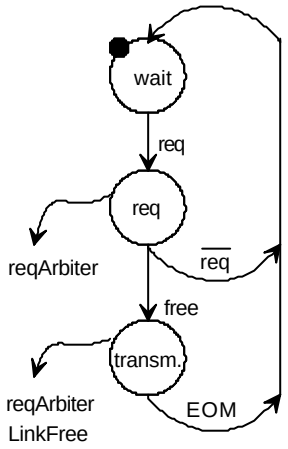


Abb. 9 Vernetzung der Kanalarbiter. Das Konzept des CP805 gestattet, alle Ports gleichzeitig kommunizieren zu lassen. Folglich sind für die gleichzeitige Kommunikation von n Ports n Arbiter vonnöten.

3.3.1 Router-Baugruppe

Der Router besteht aus einzelnen Automaten, die den Ablauf zu einem Arbiter steuern (aus sog. Routerästen). Die Anzahl der Äste entspricht der Anzahl der möglichen Ports (2D-Torus: 4, 3D-Torus: 6, nD-HC: n). Jedes Element des Requestvektors ist einem Routerast zugeordnet. Jede Anforderung (request) wird durch ein Bit im Requestvektor repräsentiert. Trifft in Auswertung eines Headers ein Requestvektor ein, erkennt der Routerast beim zugeordneten Arbiter, ob der Link frei ist. Sollte das der Fall sein, gibt der Arbiter eine frei-Meldung an den Routerast. Im adaptiven Verfahren können mehrere Routeräste gleichzeitig aktiv werden. Sind mehrere Arbiter frei, muß gesichert werden, daß sich eine Kanalanforderung durchsetzt. Der 1:n-Decoder in der Compare-Baugruppe schaltet die überzähligen Anforderungen ab. Diese Bedingung ($\overline{req}$) wird im Router bewertet. Es entsteht folgender Routerast:



Signal:	req	free	EOM	$V1^n$	$V0^n$	$V1^{n+1}$	$V0^{n+1}$
A=>A, /req				0	0	0	0
A=>B, req	1	0		0	0	0	1
B=>B, req, /free	1	0		0	1	0	1
B=>A, /req	0			0	1	0	0
B=>C, free		1		0	1	1	1
C=>C, /EOM		1	0	1	1	1	1
C=>A, EOM			1	1	1	0	0

Es bedeuten:

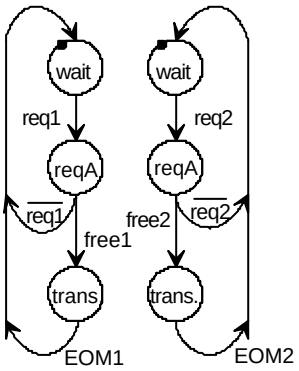
$V0 = \overline{V1}\, req \wedge \overline{V1}\, V0\, free \wedge V1 V0\, \overline{EOM}$

$V1 = \overline{V1}\, V0\, free \wedge V1\, V0\, \overline{EOM}$

$reqArbiter = V0$

$LinkFree = V1V0$

Abb. 10 Routerast für adaptive Verfahren ohne RDY-Bewertung



Dem Wait-Zustand folgen drei sequentielle Zustände:

1. *Link anfordern* (reqArbiter zum Arbiter)
2. *Link ist frei*
3. *Verbindungszustand* (transmit)

Abb. 11 Prinzip des Routers für adaptive Verfahren

Damit entsteht ein einheitlicher Router, dessen Breite nur von der Anzahl der angeschlossenen Links am CP abhängig, und damit skalierbar ist.

3.3.2 Arbiter-Baugruppe

Die in [Joss91], [Krapp91-1] und [Krapp91-2] dargestellten Arbiter unterscheiden sich bezüglich ihrer Kantengewichte (req_{ij}). Diese können mit folgenden Boolschen Formeln beschrieben werden:

Typ:	Bedeutung der Terme:
HC determin. (min=p, 0, ..., i, ..., j, ..., max): $free_{ij} = req_{ij} \bigwedge_{k=i+1}^{\max} \overline{req_{kj}} \bigwedge_{j=j+1}^{\max} \overline{req_{jl}} \wedge \overline{EOM}$	1) Kanal i fordert j an 2) eine Anforderung größer i bis max j liegt nicht vor 3) eine Anforderung j bis max liegt nicht vor
HC adaptiv (min=p, 0, ..., i, ..., j, ..., max): $free_{ij} = req_{ij} \bigwedge_{\substack{k=\min \\ k \neq i,j}}^{\max} \overline{req_{kj}} \bigwedge_{\substack{l=\min \\ l \neq j}}^{\max} \overline{req_{jl}} \wedge \overline{EOM}$	1) Kanal i fordert j an 2) Anforderung min bis max nach j (außer i,j) liegt nicht vor 3) Anforderung j von min bis max (außer j nach j) liegt nicht vor
3D-Torus (min=p, 0, ..., i, ..., j, ..., max): $free_{ij} = req_{ij} \bigwedge_{\substack{k=\min \\ k \neq i,j}}^{\max} \overline{req_{kj}} \bigwedge_{\substack{l=\min \\ l \neq j}}^{\max} \overline{req_{jl}} \wedge \overline{EOM}$	1) Kanal i fordert j an 2) Anforderung min bis max nach j (außer i,j) liegt nicht vor 3) Anforderung j von min bis max (außer j nach j) liegt nicht vor

Tab. 12 Arbiterformeln für verschiedene Netztypen

Reduziert man gedanklich einen Hypercubus auf die Dimension sechs, können 3D-Link und Hypercube-Link entsprechend ihrer Priorität (vgl. Adreßbildung) zugeordnet werden. Es entsteht eine einheitliche Indizierung.

Index ->	6	5	4	3	2	1	P
2D-Torus			north	east	south	west	node-CPU
3D-Torus	up	down	north	east	south	west	node-CPU
6D-HC	6	5	4	3	2	1	node-CPU

Tab. 13 Indizierung der Ports bei verschiedenen Netztypen

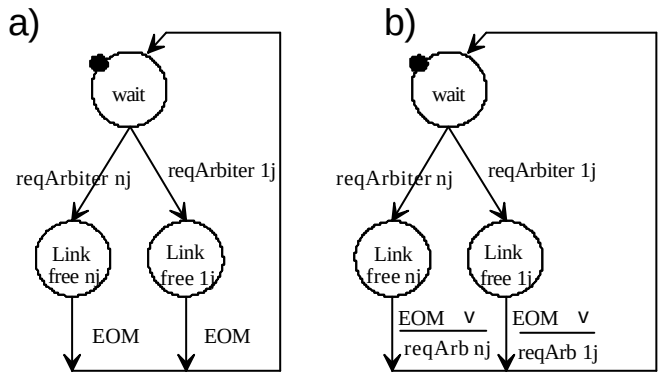
Durch Fortsetzung der Indizierung sind Hypercube höherer Dimension bearbeitbar. Eine einheitliche, formelmäßige Fassung kann gewonnen werden:

$$free_{ij} = req_{ij} \bigwedge_{k=0}^{n} \overline{req_{kj}} \wedge \overline{EOM}$$

1) $free_{ij}$ Freigabe Kanal j für Kanal i
2) req_{ij} Anforderung Kanal j durch Kanal i
3) req_{kj} bestehende Anforderungen Kanal

Tab. 14 Arbitergleichung für n-dimensionales Netzwerk

Bedingt durch das adaptiven Verfahren muß es möglich sein, die Arbiteranfrage des Routers zurückzuziehen. Das ist der Fall, wenn mehrere Links angefordert und mehrere Links frei sind, aber nur ein Link als Ausgang gewählt werden kann.
Die Übergangsbedingung EOM_n wird durch den Term $\overline{reqArbiter_{nj}}$ erweitert.



reqArbiter 1j	Kanal 1 fordert Kanal j an
reqArbiter nj	Kanal n fordert Kanal j an
free 1j	Kanal j ist für Kanal 1 frei
free nj	Kanal j ist für Kanal n frei
EOM	EOM-Signal

Abb. 15 Beispiel: Arbiterautomat für eine Dimension n = 2.
a) Arbiter nach Krapp/Jossifov
b) modifizierter Arbiter

Die Breite des Arbiterautomaten ist abhängig von der Linkanzahl skalierbar.

3.3.3 Protokollunabhängige Kanalarbitrierung

Die schnelle Kanalarbitration arbeitet unabhängig vom Übertragungsprotokoll. Sie kann somit für andere Protokolle und Übertragungsarten verwendet werden. Durch die

Beschreibung in synthetisierbarem VHDL besteht die Möglichkeit, mittels eines Steuerparameters eine skalierbare Kanalarbitration zu schaffen. Dazu kann die 'generic'-Anweisung verwendet werden. Das verwendete VHDL-Synthese-Subset unterstützt diese Möglichkeit nicht. Hier muß auf spätere VHDL-Synthesizer gewartet bzw. ein anderes Entwurfstool verwendet werden.

Im Protokoll muß in geeigneter Form die Zieladresse, den Netztyp und das zu verwendende Verfahren mitführen. Die Linkebene extrahiert diese Daten und übergibt sie (adress-Signalbus) an das Protokollinterface der Kanalarbitration. Als Startkennzeichen ist die Meldung 'HeaderIn' an das Protokollinterface zu geben. Sie übernimmt die Daten, führt die Kanalarbitration aus und signalisiert mit 'send', daß die Botschaft auf der Linkebene zum nachfolgenden CP gesendet werden kann. Am Botschaftsende ist das Signal 'EOM' an das Protokollinterface zu geben.

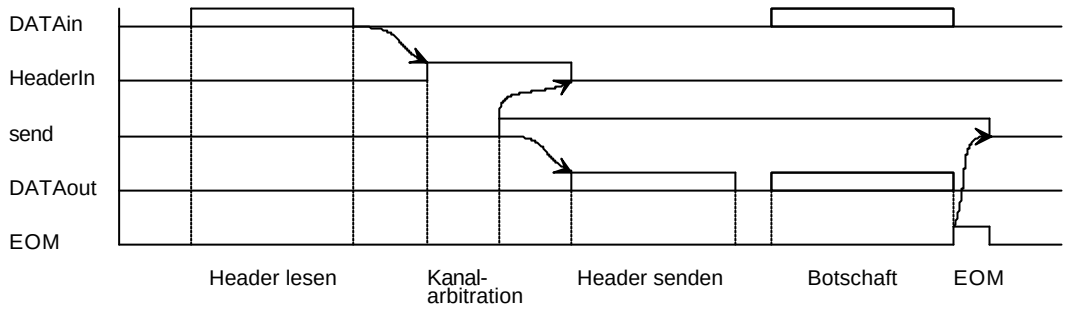


Abb. 16 Zeitablauf bei Arbitrierung unabhängig vom Protokoll

Allgemein gelten folgende Bedingungen:

1) Die Adreßbits liegen als paralleler Vektor vor. Der Header enthält die Informationen über Netztyp und Verfahren.
2) Die Größe der Knotennummer entspricht dem Adreßvektors. Die Vergleichsbaugruppen sind der Breite der Vektoren angepaßt.
3) Nachdem der Adreßvektor stabil ist, startet ein Signal die Kanalarbitration.
4) Steht der Ziellink fest und ist die Zusammenschaltung der Links über einen Multiplexer erfolgt, erscheint das Signal 'send'.
5) Nach der Botschaft ist an die Kanalarbitration ein EOM-Signal zu geben.

Zu den protokollunabhängigen Einstellungen gehört die Übergabe der Knotennummer. Die Signale xyDATA und xyACK sind die Daten-bzw. Acknowledgeleitungen. Jede Leitung des Signalpaares wird monodirektional durch den CP geschaltet. Dabei gilt: Das kommende Protokoll geht an **s**yDATA in den CP und verläßt den CP an **e**zDATA. Das zugehörige ACK erscheint an **e**zACK und verläßt den CP an **s**yACK. y und z stehen für die Integerzahl des Link.

Die Möglichkeit, breite Adressvektoren zu verwenden, resultiert aus der synthetisierbaren VHDL-Beschreibung. Nur mit einem Texteditor sind die Skalierungen einfach realisierbar. Folgender VHDL-Ausschnitt zeigt die Schnittstelle zum Transputerprotokoll.

```
ENTITY protokoll IS
 PORT ( adress1, adress2, adress3,adress4, adress5 : in BIT_VECTOR (7
DOWNTO 0);
        RDY1, EOM1, HeaderIn1, RDY2, EOM2, HeaderIn2,
        RDY3, EOM3, HeaderIn3, RDY4, EOM4, HeaderIn4,
        RDY5, EOM5, HeaderIn5                        : IN BIT;
        send1, send2, send3, send4,send5             : OUT BIT;
        s1DATA, s2DATA, s3DATA, s4DATA, s5DATA       : IN BIT;
        s1ACK , s2ACK , s3ACK , s4ACK , s5ACK        : OUT BIT;
        e1ACK , e2ACK , e3ACK , e4ACK , e5ACK        : IN BIT;
        e1DATA, e2DATA, e3DATA, e4DATA, e5DATA       : OUT BIT;
```

Tab. 17 Transputerprotokoll-Schnittstelle als VHDL-Script

```
Loading Technology
-- /disk2/mentor/synthese_libs/gen_lib *** gen_lib Library - Version 1.2 -
07Jul93
// Optimizing Netlist
Netlist Statistics
========================================
        type  count cost  subtotal
========================================
GFL_LIB: AND   322  5/pin 4135
GFL_LIB: DFF    80  10     800
GFL_LIB: INV   344
GFL_LIB:  OR   173  5/pin 2065
GFL_LIB: XOR    14 10/pin  280
----------------------------------------
Totals:        933   0.00 7280
```

Tab. 18 Auszug: Steuerung der VHDL-Synthese

3.3.4 Das Single-Stage-Schaltsystem

Zur Kanalkopplung wird ein einstufiger Multiplexer verwendet. Die entstehende Latenz ist lediglich von der Technologie abhängig. Sie beträgt wenige Gatterlaufzeiten. Die Größe des Multiplexers hängt von der Anzahl der Link's ab. In einem 2D-Netz werden fünf, im 3D-Netz sieben, im HC-Netz der Dimension n werden n+1 Links verknüpft.

Falls sich benachbarte Knoten zeitgleich einen Header zusenden, hat jeder Kommunikationsprozessor dezentral zu entscheiden, welcher der Knoten als erster die Leitung benutzen darf.

Für ein 3D-Netz sind die Gewinnkriterien fest vorgegeben: Ist der Link in einem Kommunikationsprozessor höher priorisiert, muß er im gegenüber liegenden

Kommunikationsprozessor niedriger priorisiert sein. Es gewinnt der Kommunikationsprozessor mit dem höher priorisierten Link.

Da sich beim Hypercubus benachbarte Knoten in einer Bitstelle unterscheiden (Hammingdistanz von Eins), und zwar genau an der Bitstelle des gemeinsamen Kanals, kann die Bitstelle der Knotennummer K[j] zur Entscheidung herangezogen werden. Aus Sicht des vermittelten Headers kann die Verbindung regulär erfolgen, wenn die eigene Knotennummer größer als die des Zielknotens ist. Bei größerer Knotennummer ist dieses Bit Eins.

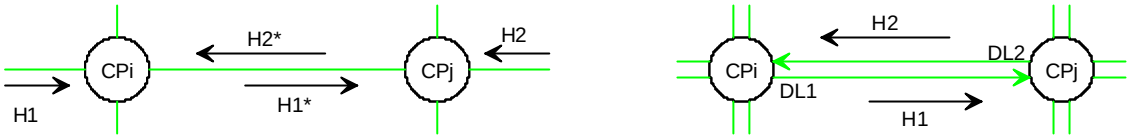


Abb. 19 Zeitgleiches Senden über eine bzw. zwei Doppelleitungen

Die schaltungstechnischen Lösungen verkomplizieren das Verständnis des Kommunikationsprozessors. Deshalb wurden im CP 805 zwei Doppelleitungen zwischen benachbarten CP's instanziiert, die aus je einer Daten- und einer ACK-Leitung besteht. Während eine Doppelleitung Senderichtung des Kommunikationsprozessors ist, wird die andere Doppelleitung für die Empfangsrichtung verwendet.

3.3.5 Protokoll in einer T8xx-Transputerumgebung

Das Transputerprotokoll sieht vor, daß jedes gesendete Datenwort mit einem Bestätigungssignal (Acknowledge) beantwortet wird. Nachfolgende Abbildung verdeutlicht den Ablauf. Das Datenwort des Transputers ist elf Bit lang. Den beiden Startbits folgen acht Datenbits und ein Stoppbit. Das ACK kann entweder unmittelbar nach den beiden Startbits oder nach dem Stopbit vom Ziel gesendet werden.

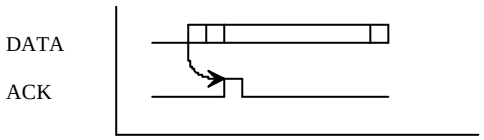


Abb. 20 Transputerprotokoll der Serien INMOS T2xx...T4xx...T8xx

Für den CP 805 wurde das T8xx-Transputerprotokoll beibehalten. Das zwischengeschaltete Netz 'sehen' die Transputer nicht. Beim Verbindungsaufbau zwischen den CP 805 wird der Header unbestätigt weitergegeben. Ist der Header beim Zieltransputer angekommen, gibt dieser das ACK.

Eine Botschaft besteht aus Header, Längenwort und Daten. Der Header enthält die Zieladresse, das Längenwort die Anzahl der Worte in der Botschaft. Ein Wort entspricht einem Byte.

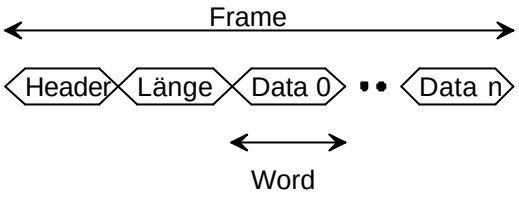


Abb. 21 Frame-Protokoll des CP 805

Die Bit acht und sieben des Headers geben den Netzwerktyp und das Verfahren an. Mit ihnen wird die Bewertung des Headers gesteuert. Für den Netztyp und das Verfahren gilt folgende Bitbelegung:

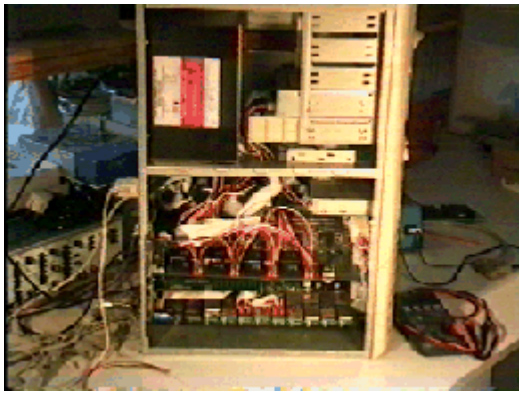
Header								Erläuterung
Netztyp	Verfahren	Adresse						
Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	
0	0	y2	y1	y0	x2	x1	x0	2D
0	1							z.Z. unbelegt (3D)
1	0			Link 4	Link 3	Link 2	Link 1	HC det.
1	1			Link 4	Link 3	Link 2	Link 1	HC ad.

Tab. 22 Bitbelegung der Header bei T8xx-Protokoll

Der Paketbetrieb schafft Abhilfe gegen Dead Lock's. Da Standleitungen aufgebaut und die Pakete nicht im Netz zwischengespeichert werden, entfällt das Problem von Paketreihenfolgen. Das Konzept der virtuellen Kanäle ist dem CP 805 eigen.

4 Erprobungsergebnisse

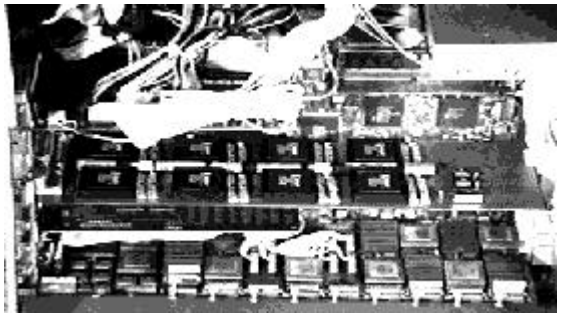
Der entwickelte Kommunikationsprozessor wurde in einem 8-Knoten Transputersystem (IMS-T805 mit 4MB RAM pro Knoten) erprobt.



Kommunikationsprozessoren (8 Stck.)

Transputer-Board mit 8 Transputern

Abb. 23 Versuchsaufbau. Ein Pentium i586 dient als Träger für 8 Kommunikationsprozessoren CP805, die 8 Transputer- CPU's TRAM T805 miteinander verbinden



Kommunikationsprozessoren

Transputer-Board

Abb. 24 Die Kommunikationsprozessoren sind im Pentium auf einem ISA-Slot über dem Transputerboard untergebracht

4.1 Timinganalyse der Transputerkarte

Vor der Simulation mit dem CP 805 wurde die Transputerkarte (MSC 9s-16l) einer Timinganalyse unterzogen. Benachbarte Transputer, verbunden über einen C004, tauschen eine Nachricht aus. Es wurden folgende Werte ermittelt

Aktion	t/ns
1.Byte ↑	0
1.ACK ↑	1.000
1.ACK ↓	1.056
2.Byte ↑	80.056
2.ACK ↑	81.056
1.ACK ↓	81.112
3.Byte ↑	92.112

Meßbedingungen:
Linkrate 20 MBit/s
Links am SUB-D-Stecker verbunden
Testprogramm: 'CP1'
Logic Analyzer HP 1662 A
Auflösung des Logic Analyzers: 8ns

Tab. 25 Ermittlung der Dauer des Verbindungsaufbaus (Latency) anhand von Meßwerten im erprobten Transputersystem

Es zeigt sich eine zeitliche Verschiebung zwischen ersten Acknowledge und zweitem Datenwort. Um eine Vorstellung über die Verhaltensweise der Testumgebung zu erhalten, wurden in einer Transputerpipe Datenpakete unterschiedlicher Größe und über unterschiedliche Entfernungen verschickt. Folgende Werte in Sekunden wurden dabei erreicht:

Datenmenge (Byte)	Transputeranzahl						
	1	2	3	4	5	6	7
100*120	1s	2s	2s	2s	3s	3s	3s
200*120	3s	3s	4s	5s	5s	6s	7s
300*120	4s	5s	6s	7s	8s	9s	10s
400*120	6s	6s	8s	9s	11s	11s	13s
500*120	7s	9s	10s	13s	14s	15s	16s
600*120	9s	10s	13s	15s	16s	18s	19s
700*120	10s	13s	14s	17s	18s	21s	22s
800*120	12s	13s	16s	18s	21s	24s	26s
900*120	13s	15s	18s	20s	23s	27s	29s
1000*120	14s	17s	21s	23s	26s	30s	32s

Programm CP2

Option /nv
no virtuell channel

Tab. 26 Zeitabschätzung an einer Transputer-Kette

4.2 Simulation des CP 805

Als zusätzliche Verifikation wird der CP nach der Logikgenerierung als FPGA und der Backannotation simuliert.

HC und 2D/3D	t/ns	Takte	Vorgang
t_{01}	653,4	12	Einlesen des Header
t_{12}	219,8	5	Arbitrierung des Zielkanals
t_{23}	113,3	3	interne Verarbeitung
$\sum$	986,3	20	
Wortverzögerung im CP	38,4	1	

t_0 Startflanke des Headers
t_1 Start der Arbitration
t_2 Arbitration beendet
t_3 Header am Ausgang des CP's

Meßbedingungen:
- Zeitbasis 50 ns = 20 MHz
- Routing: deterministisch, 2D-Feld
- Technologie: FPGA XC4010

Tab. 27 Ergebnisse der FPGA-Simulation

Die Schaltungssynthese unter Mentor-Autologic ist so eingerichtet, daß zwischen den getakteten Register-Flipflops typisch drei (Xilinx-) Gatterlaufzeiten liegen. Über die Simulation kann die Funktion des Netzwerkprozessors bestätigt werden. Aufgrund eines gleichen Globalalgorithmus benötigen die unterschiedlichen Verfahren zum Arbitrieren und zur Kanalschaltung die gleiche Zeit. Die Verzögerung der Botschaft bei bestehender Verbindung ist gering. Der Ablauf eines Verbindungsaufbaus ist in folgender Abbildung verdeutlicht.

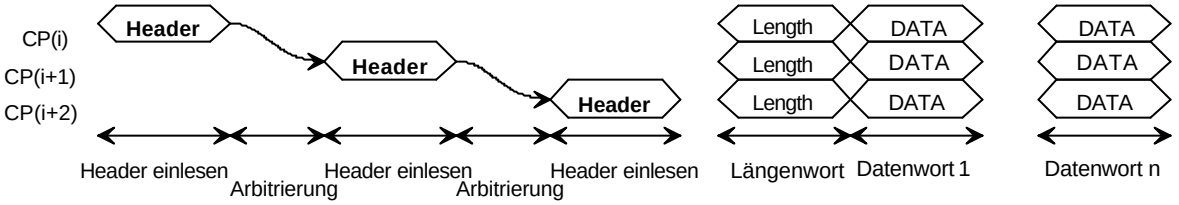


Abb. 28 Verzögerung einer Nachricht beim Verbindungsaufbau in drei in Kette geschalteteen CP 805

Messung des Datendurchsatzes:

Datenrate	20MBit = 2,5 MByte
max. Paketlänge	256 Byte
Overhead des CP	2 Byte (Header und Länge)
Transputer	2,5 MByte / 256 Byte/Paket = 9766 Pakete
CP	2,5 MByte / 258 Byte/Paket = 9690 Pakete
Differenz	76 Pakete
Zeitbedarf der Vermittlung von 9690 Paketen	9690 Pakete/s*(20*20/MHz+1/20MHz) = 3,9 Pakete
Transputer mit 9766 Paketen/s	9766Pakete/s*256Byte/Paket

CP mit drei Verbindungen	3*9686Pakete/s = 29058Pakete/s = 29058Pakete/s*256Byte/Paket = 7438848 Byte/s

Während der Transputer 9766 Pakete a 256 Byte überträgt, schafft der CP lediglich 9686, also 80 Pakete weniger. Wird angenommen, daß eine optimale Linkauslastung erfolgt, kann für einen CP mit sieben Links, das sind drei mögliche Verbindungen, ein *Datendurchsatz von 7,44 MByte/s pro CP* angegeben werden.

4.2.1 Timing des CP805

An Einzelbeispielen soll die Funktion des CP demonstriert werden. Zunächst wurde ein 2D-Netz mit acht Transputern aufgebaut. Die Bilder stammen aus dem Screen-Print des verwendeten Logic-Analyzers HP1662A.

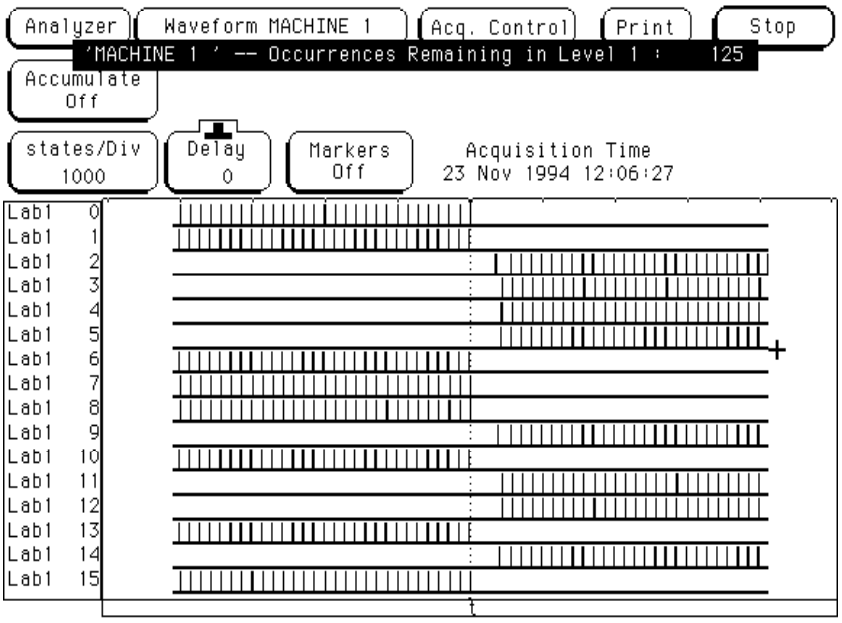
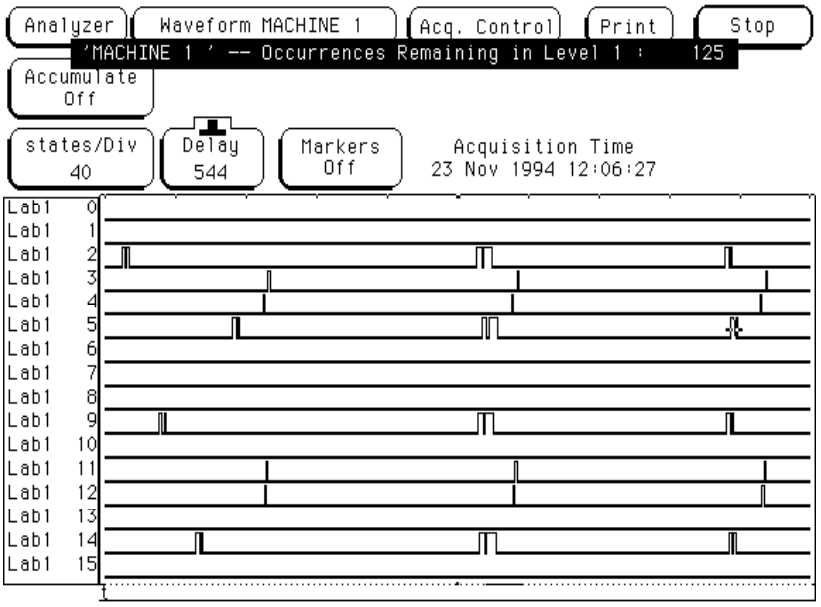


Abb. 29 Screen Print vom Logikanalysator zur Verbindung zweier Transputer über drei CP`s (Pipe). Transputer "2" sendet 125 Byte an Transputer "0", anschließend umgekehrt.

Weg der Daten		Weg des Acknowledge	
Ausgang	ScreenPrint	Ausgang	ScreenPrint
Transputer 2	Lab 6	Transputer 0	Lab 0
CP 2	Lab 10	CP 0	Lab 15
CP1	Lab 13	CP 1	Lab 8
CO 0	Lab 1	CP 2	Lab 7

Abb. 30 Versuchsaufbau der Transputer-Pipe zur Gewinnung des Screenprints

Die folgende Abbildung zeigt die ersten drei Byte der Datenübertragung von Transputer "0" an Transputer "2". Beim Verbindungsaufbau schreitet der Header zunächst unbestätigt zum Ziel. Die große Zeitspanne zwischen Empfang des ACK und dem Aussenden des folgenden Datenwortes ist deutlich.



		Takt	Zustand
Lab2	Header vom Quelltransputer trifft am ersten CP ein	0	Verbindungsaufbau
Lab 9	Header am Ausgang des ersten CP	21	t_{summe}= 83 States
Lab 14	Header am Ausgang des zweiten CP	42	
Lab 5	Header am Ausgang des dritten CP	63	
Lab 4	ACK vom Zieltransputer	80	
Lab 12	ACK nach CP 3	81	
Lab 11	ACK nach CP 2	82	
Lab 3	ACK nach CP 1	83	
Lab2	Längenwort vom Quelltransputer	203	Übertragung des Längenwortes
Lab 9		204	t_{summe}= 226-203= 23 States
Lab 14		205	
Lab 5		206	
Lab 12	ACK vom Zieltransputer	223	
Lab 11	ACK nach CP 3	224	
Lab 4	ACK nach CP 2	225	
Lab 3	ACK nach CP 1	226	
	Datenwort	355	Übertragung eines Datenenwortes

Abb. 31 Verbindungsaufbau über drei CP's, T0 sendet an T2

Die ersten drei Byte der Datenübertragung von Transputer "0" an Transputer "2"

Der Verbindungsaufbau über drei CP's ist nach 83 Zustandsübergängen abgeschlossen. Pro CP werden ca. 20 States benötigt. Nach dem die Verbindung aufgebaut ist, beträgt die Verzögerung nur noch 23 States. Bemerkenswert ist die Tatsache, daß der sendende Transputer sehr viel Zeit benötigt um das nächste Datenwort zu senden. Zur Effizienzerhöhung muß also die Reaktionszeit der Software minimiert werden.

4.2.2 Gestaltung einer simulierten VHDL-Testbench

Das VHDL-Modell des CP805 ist zusätzlich in einem Simulationsmodell einer VHDL-Testbench simuliert worden. Das Ziel der VHDL-Beschreibung, anderen Anwendern ein einfaches Modell des CP 805 in die Hand zu geben, wurde erreicht. Die Kanalarbitration ist in der Entwurfsumgebung Mentor Graphics voll synthetisierbar. Auf einen synthesefähigen Linkanschluß wurde verzichtet. Die graphische Darstellung der VHDL-Testbench spiegelt auch den inneren Aufbau des CP 805 wieder.

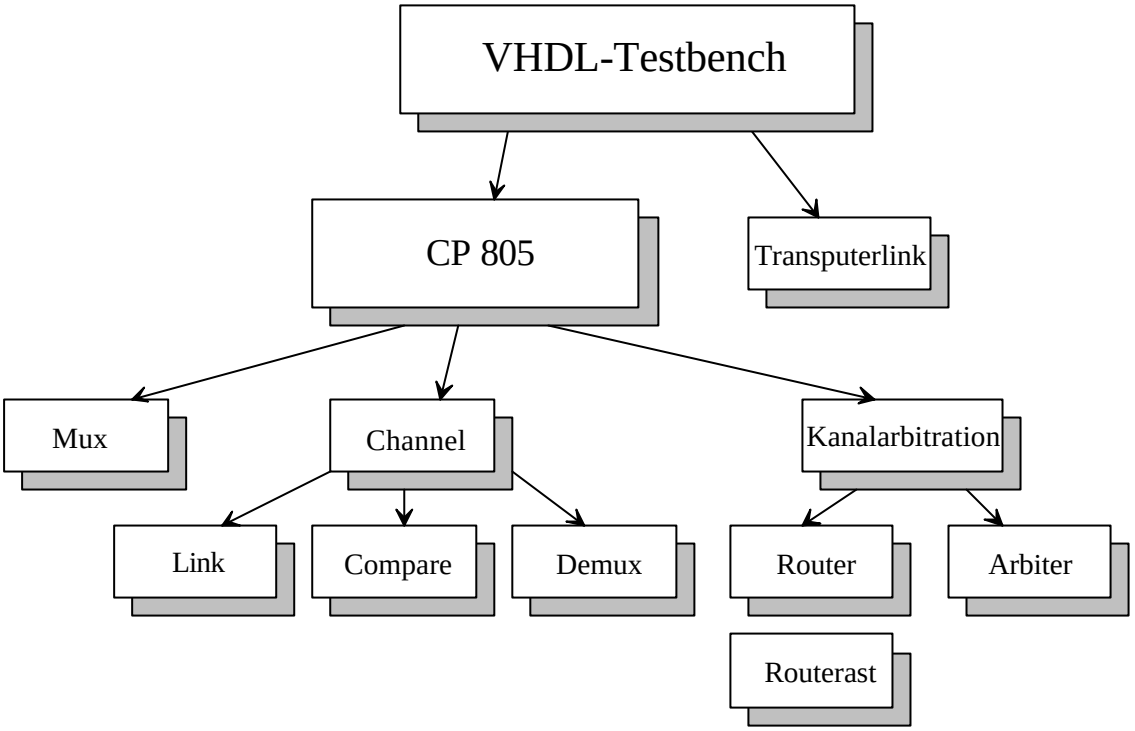


Abb. 32 Baugruppenübersicht der VHDL-Testbench

Die VHDL-Testbench ist so aufgebaut, daß sich im Mittelpunkt ein CP befindet, an dem fünf Transputerlinks angeschlossen sind. Durch zwei Parameter kann jedes Link gesteuert werden. Die Simulationsergebnisse sind im Anhang C angefügt.

5 Vergleich zu Kommunikationsstrategien

5.1 Kommunikationsstruktur T805/C004

Der Transputer T805 ist in einem 84-poligen PinGrid-Gehäuse eingebaut. Er arbeitet intern mit einer Taktfrequenz von 30 MHz. Die Datenrate der seriellen Links beträgt 20MBit/s, womit ein Nettodurchsatz von 800KByte/s erreicht wird. Bei der byteweisen Übertragung der Botschaft wird jedes Byte vom Empfänger quittiert. Der Transputer T805 besitzt folgende Leistungsmerkmale:

1) 30 MIPS (Million Instruktionen per Second)
2) 3,3 MFLOP (Million Floating Point Operation per Second)
3) 30 MHz intern.

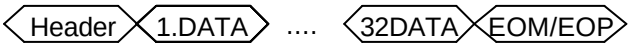
Netzstrukturen können über den Kreuzschienenverteiler-Schaltkreis C004 aufgebaut werden. Der C004 wird dabei mit einer entsprechenden Konfiguration geladen und behält diese bis zur Neukonfiguration bei. Kommunikation ist mit dem C004 an zentrales Laden der Kommunikationsstruktur gebunden, prozeßabhängige Kommunikation ist nur über den Umweg einer Mitteilung an den zentralen Server gangbar, demzufolge ist diese extrem schwerfällig.

5.2 Kommunikationsstruktur T9000/C104

Der Transputer T9000 besitzt folgende Leistungsmerkmale:

1) 200 MIPS
2) 25 MFLOP
3) 50 MHz intern.

Der T9000 besitzt einen internen 'Virtual Channel Processor' VCP. Er steuert die Kommunikation zwischen T9000 und Außenwelt. Das Link-Interface besteht aus vier seriellen Links mit einer Linkgeschwindigkeit von 100MBit/s und erreicht einen Nettodurchsatz 80 MByte/s pro Link. Es wird ein Verfahren mit Data-Strobe-Link eingesetzt, d.h. pro Link existieren die Leitungen DATA und Strobe. Ein Datenbit wird erkannt, falls auf einer der Leitungen das Potential wechselt. Bedingt durch die angewandte VLSI-Technik sind mehr Links nicht möglich. Der T9000 unterstützt das Konzept virtuellen Kanäle, d.h. mehrerer sende- bzw. empfangswillige Prozesse teilen sich einen physischen Link. Ein sendewilliger Prozeß reiht seine Botschaft in eine Warteschlange ein. Die Botschaft wird in bis zu 32 Byte lange Pakete verpackt und verschickt. Ein Paket hat folgendes Aussehen:



EOM-End Of Message
EOP- End Of Packet

Abb. 33 Protokoll des T9000

Mit dem Eintreffen des ersten Bytes wird jedes empfangene Paket quittiert. Für den Fall, daß der Empfangsprozess nicht bereit ist, besitzt der T9000-Chip einen internen Puffer für ein Paket.

Der *C104*, die Weiterentwicklung des C004, ist ein für das T9000-System entwickelter Paket-Routing-Chip. Intern können seine 32 Links frei miteinander über einen 32*32 Kreuzschienenverteiler verbunden werden. Der C104 vermittelt die vom T9000 kommenden Pakete und unterstützt damit das Konzept der virtuelle Kanäle.

Folgende Schaltbeispiele sollen dessen Leistungsfähigkeit zeigen:

1) Ein Clos-Netz (bekannt als Spiegel-Verfahren aus der Nachrichtentechnik) mit 512 Knoten bei dem die Botschaft maximal drei Routing-Chips passiert, weist eine Latenzzeit von 1µs und einer Bandbreite 5 GByte/s, verteilt auf vier Links und bei Niedriglast, auf.
2) Mit einem 16*16 2D-Array mit 256 Knoten, bestehend aus je einem T9000 und C104, kann eine Latenzzeit 10 µs und eine Gesamtbandbreite von 1,3 GByte/s bei Niedriglast erreicht werden.
3) Ein Crossbar-Netzwerk bestehend aus 32 Transputern mit je einem Link am C104 bei dem 16 Paare bidirektional kommunizieren erzielt eine Bandbreite von 320 MByte/s mit 0,5 µs Latenz.
4) Die Spitzenbandbreite von 10 GByte/s erreicht ein Hypercubus mit 1024 Knoten bei einem maximalen Weg der Botschaft über 5 Routerknoten mit einer Latenz 1,5 µs je Knoten.

Der C104 verwendet den Algorithmus des Intervall-Labeling. Jeder Transputer erhält eine Nummer (Label). Beim Botschaftstransport wird dieses Label als Zieladresse angegeben. In einem C104 existiert für einen bestimmten Adreßbereich (mögliche Headerwerte - ein Intervall) ein Ausgangslink. Das Ziel befindet sich in der Richtung des Links. Botschaften, deren Header in dieses Intervall fällt, werden zu diesem Ausgangslink vermittelt. Im nächsten Knoten wird der Bereich weiter eingeengt bis schließlich der Zielknoten erreicht ist. Die Intervallwerte müssen in den C104 geladen werden.

Der C104 ist für die Zusammenschaltung kleinerer Transputerzahlen (bis 32 Links - 8 Transputer) konzipiert. Erst mit einer Umprogrammierung auf einen Zweibyte-Header, gestattet der C104 die Bildung von großen Netzwerken. Mit diesem Header können bis zu 64k Knoten adressiert werden. Dabei wird ein C104 mit n Links am Netzwerk, die restlichen Links sind mit lokalen Transputern verbunden.

Der *C100* ist ein Link-Protokoll-Konverter zwischen T8xx und T9000. Er unterstützt die Zusammenarbeit von Mischsystemen und enthält daher Leistungsmerkmale beider bereits beschriebener Systeme.

Die Bandbreite des CP 805 und des C004 ist wegen des Protokolls gleich und liegt mit 20MBit/s unter der des C104 mit 100 MBit/s. Das Übertragungsverfahren des CP 805 ist im Gegensatz zum C104 asynchron. Dieses dürfte für weit verteilte Netze interessant sein. Hier stellt sich die Frage, warum das DATA-Strobe-Verfahren mit zwei aktiven Leitungen gewählt wurde. Sinnvoll ist ein kreisendes Token um Synchronisation in diesem Geschwindigkeitsbereich zu erreichen. Die frei skalierbare Linkanzahl beträgt im CP 805-Prototyp vier. Sie unterstützt damit direkt ein 2D-Netz. Die 32 Links des C004 und C104 resultieren aus dem Ansatz Subnetze über einen einzigen Schaltkreis verschalten zu können.

Der CP 805 unterstützt mehrere Routingstrategien und ist somit ohne Änderung und nur mit Kenntnis seiner eigenen Knotennummer, sie kann z.B. über Jumper fest eingestellt werden, in verschiedenen Netztypen einsetzbar. Für 2D/3D-Arrays und die HC-Variante adaptiv ist das Intervall-Labeling, für die HC-Variante deterministisch die Bit-Erosion gewählt. Der CP 805 gestattet adaptives und deterministisches Routing, was insbesondere für Anwendung interessant sein wird, die einen bestimmten Weg des Paketes durch das Netz voraussetzen.

Merkmal	**T805 mit CP805**	**T805 mit C004**	**T9000 mit C104**
Link-Bandbreite	20 MBit/s	20 MBit/s	100 MBit/s
Verfahren	asynchron	asynchron	synchron
Linkanzahl	4	32	32
Routingstrategie	• Wormhole Routing • Bit Erosion • Intervall Routing	starr, Crosspoint-Switch	Intervall Labeling
unterstützte Topologien	• 2D Array • HC • 3D Array	alle statisch Verbindbaren, kein dynamisches Nachladen	• 2D Array • HC • Crossbar • Clos-Netz
virtuelle Kanäle	ja	nein	ja
Routingzeit	1 Byte		1 bzw. 2 Byte
Routing aller Link's erfolgt:	gleichzeitig	kein Routing, starre Topologie	nur eine Link kann geroutet werden
max. Knotenanzahl	64k	skalierbar	64k
Headerlänge	3 Byte (im Prototyp ein Byte)	ohne	1 bzw. 2 Byte
Botschaftslänge	2..256 Byte	1 Byte	1..32 Byte
Overhead	3:2 bis 3:256	ohne	2:2 bis 1:32
Quittung	Wortquittung	Wortquittung	Paketquittung
Latenz/Bandbreite			
2D-Array	1µs / 20 MBit/s		10 µs / 1,3 GByte/s
HC	1µs / 20 MBit/s		1,5 µs / 10 GByte/s
je Link	800 kByte/s	800 kByte/s	80 MByte/s
Technologie			
Schaltkreis	FPGA XILINX XC4010...25 (8500...20000 Gates)	ASIC (1,2 Mill. Transistorfunktionen)	ASIC

Tab. 34 Vergleich verschiedener, serieller Kommunikations-Strategien (CP 805, C104, C004)

Aus Sicht eines Transputers arbeiten CP 805 und C104 nach den Worm-Hole-Algorithmus. Beide unterstützen das Virtual-Channel- Konzept. Die Routingzeit des CP 805 liegt etwa im Bereich eines Bytes, die des C104 darüber, wobei der CP 805 zeitgleich mehrere Routingvorgänge vollziehen kann. Die Maximale Knotenanzahl ist bei beiden Systemen 64k.

Die Headerlänge des Prototyps CP 805 ist ein Byte. Wegen der größeren Wortlänge ist der Overhead des CP 805 geringer. Der wortweise Quittungsbetrieb des CP 805 resultiert aus dem Protokoll des Transputer T8xx. Für die eigentliche Arbeit benötigt der CP 805 keine

Bestätigung. Mit einer Lösung als FPGA dürfte bei weiter fallenden Hardware-Preisen der CP 805 durchaus interessant sein. Der als Prototyp eingesetzte FPGA (XILINX XC 4010) kostet derzeit ca. 250 DM. Allerdings ließen sich auf diesem FPGA nur vier Links unterbringen (3-dim. HC plus CPU-Link). Für die Unterbringung eines 3D-Torus/6D-HC (6+1 = 7 Links) wären etwa 20000 Gates erforderlich. Ein entsprechend leistungsfähiges FPGA (XC4025) kostete derzeit noch etwa 1700 DM.

Eine Leistungsbewertung verschiedener Systeme im Vergleich ist kompliziert. Zur Entwicklung der Tabelle werden verschiedene Leistungsparameter aus den Datenblättern herangezogen.

6 Applikative Aspekte

Für kleine Netzwerke geometrischer Längen bis zu etwa einem Viertel der Link-Taktrate (Beispiel 100 MHz-Link: 2.5*30 cm = 75 cm) sind synchrone, parallele Kommunikationsverfahren aufgrund ihres minimalen Aufwandes wohl optimal geeignet. Asynchrone Techniken erlauben größere Netzausdehnungen. Jedoch erschweren EMV-Probleme, verursacht durch das parallele Schalten aller Kommunikationsbusse, zunehmend deren Einsatz. Abhilfe schafft bei großen Netzwerken eine die Strombilanz ausgleichende, serielle Übertragung. Ab einer bestimmten Netzwerkgröße sind Preisvorteile und Gewichtsersparnisse zu erwarten. Der Toleranzbereich, innerhalb dessen serielle und parallele Lösungen nebeneinander koexistieren, ist dennoch breit. So hat sich im PC-Sektor neben der seriellen Schnittstelle im selben Geschwindigkeitsbereich offenbar gleichwertig die parallele Schnittstelle etabliert.

Der CP 805 stellt eine serielle Lösung für eine große Anzahl von Prozessorknoten dar.

Mittels optischer Verbindung und Vermittlung kann die Übertragungsgeschwindigkeit erhöht werden. Unterbrochene Links können damit auf technisch einfache Weise aus dem Routing ausgenommen werden. Der Einsatz serieller Vergleichsbaugruppen kann den Durchsatz des CP's erhöhen. Damit können die Pakete 'on the fly' durch den CP gelenkt werden.

Intelligent routende, serielle Lösungen (CP805, C104) zeigen in Parallelrechnersystemen ein Problem bei gehäufter Übertragung kleiner Datenpakete. Die Reaktionszeit der Software ist im Verhältnis zur Übertragungsdauer unverhältnismäßig hoch, da im Vergleich zum Crosspoint-Switch-Übertragungssystem (seriell: INMOS-C004, parallel: GMD-MANNA) zusätzliche Prozesse zur Prüfung einkommender Datenpakete, wie zur Vorbereitung zu versendender Datenpakete zu erwarten sind.

Ein wesentlicher Vorzug routender, serieller Verfahren liegt in der relativ unaufwendig realisierbaren Möglichkeit, Netzabschnitte dadurch zu verlängern, daß Hochgeschwindigkeits-Übertragungsnetzwerke (ATM, FDDI) zwischen Links geschaltet werden können. So ist es bei Bedarf denkbar, im Territorium verteilt angeordnete Plattformen zur Lösung komplexer Aufgaben zu verbinden. Korrespondierende Management- und Softwareprobleme verhindern bislang die Anwendung dieser Möglichkeit.

6.1 Kommunikation des CP805 mit der Knoten-CPU

Grundsätzlich gibt es verschiedene Möglichkeiten, mit externen Geräten (CP805) bidirektional zu kommunizieren.

6.1.1 Portorientierter Zugriff

Beim portorientierten Gerätezugriff sind Basisadresse sowie ein Offset bekannt. Die Basisadresse führt zu den Steuer- bzw. Datenregistern. Beim Empfang vom Gerät kann ein Hard- oder Softwareinterrupt einen Treiber aktivieren. Folgend findet Polling ab einer bekannten Adresse statt. Der Vorteil dieser Variante besteht in der relativ einfachen

Anpassung an verschiedene Umgebungen mittels Treibersoftware. Dem steht der Nachteil gegenüber, daß Senden und Empfang mit CPU-Unterstützung laufen. Damit belastet der Austausch winziger Kommunikationspakete den Zeitfonds der Knoten-CPU extrem negativ. Besonders ungünstig ist die quasiserielle, wortweise Übertragung von einer CPU in Datenregister.

6.1.2 Datenorientierter Zugriff

Bei einer datenorientierten Steuerung besitzen CPU und Gerät gemeinsame Speicherzelle(n) und können wahlfrei darauf zugreifen. Ein permanent unterladener Prozeß (Semaphore) überwacht den exklusiven Zugriff. Der Vorteil ist die hohe Geschwindigkeit, die durch den parallelen Speicherzugriff entsteht.

Aus Sicht des Betriebssystems ist an dieser Stelle zwischen Prozeß und Thread zu unterscheiden. Während ein Thread den gesamten Speicherraum erreicht, gehören Prozeß, virtuelle Maschine und prozeßeigene Daten zusammen. Bei einem Message Passing System werden Objekte zwischen den Prozessen ausgetauscht, daß heißt der Betriebssystemkern verschiebt Speicherinhalte.

Im folgenden Abschnitt werden weitere Möglichkeiten der Kopplung von CPU und CP805 betrachtet.

6.1.3 Portorientierter Zugriff über Dual-Kanal-RAM

Am Markt werden 8k*8 Bit Dual-Kanal-RAM's mit einer Zugriffszeit von 15 ns (ca. 65 MHz) angeboten. Durch Parallelschaltung von z.B. acht Speichern, können 64 Bit Busbreite aufgefangen werden. Legt man eine lineare Skalierung der Routingzeit (20 MHz - 350 ns; 200 MHz - 35 ns) und einem Sende-Empfangsverhältnis der CPU von 1:1, ergibt sich 17,5 ns Zugriffszyklus.

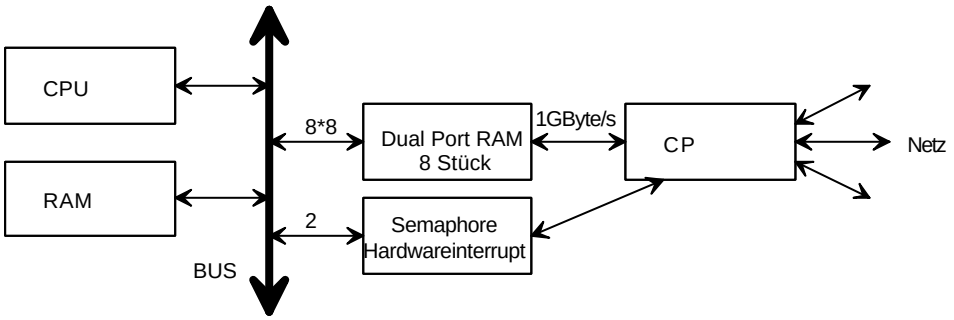


Abb. 35 Portorientierter Zugriff über Dual-Kanal-RAM

Dem Vorteil, daß die CPU nach dem Füllen der Dual Kanal RAM's sofort weiter arbeiten kann, stehen die Nachteile der a priori begrenzte Paketgröße (64 Bit) sowie das Einblenden der Dual-Kanal-RAM's in den Adreßraum gegenüber.

6.1.4 Zugriff auf einen gemeinsamen Speicher

Die Ablage der Nachrichten im Speicher in der Form: 'Zieladresse - Länge - Zeiger-auf-die-Daten' unter CPU-Kontrolle kommt den Kommunikationsanforderungen und dem Message Passing Konzept am nächsten. Die Paketgröße ist praktisch unbegrenzt. Die CPU siht das externe Gerät nicht. Nachteil dieser Lösung ist der konkurrierende BUS-Zugriff von CPU und CP sowie das Überwachen von Speicherzellen durch den CP.

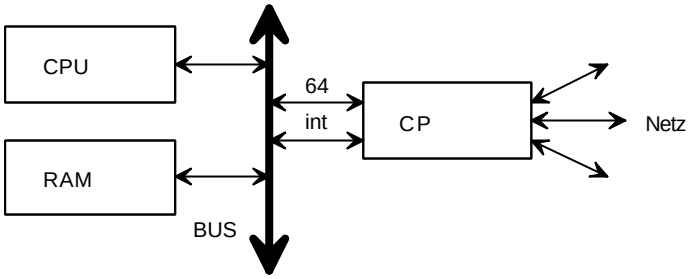


Abb. 36 Speicherorientierter Zugriff von CPU und CP über einen gemeinsamen RAM

6.1.5 Daten im RAM, Sendeaufforderung im FIFO

Eine Mischform ist die Kopplung von Datenablage im Speicher und Headerinformationen in einem FIFO. Die Sendeaufrufe können günstig als FIFO organisiert werden, der von der CPU gefüllt und vom CP geleert wird. Es werden Ziel, Länge und ein 'Zeiger-auf-die-Daten' im FIFO abgelegt. Der CP errechnet aus der Längenangabe die Anzahl der zu sendenden Pakete, holt die Daten aus dem Quellbereich, fügt einen Header hinzu und schickt das Paket ab. Dieser Vorgang wiederholt sich, bis die Länge erreicht ist. Die Adreßrechnung (Zeiger := Zeiger +1) erfolgt im CP.

Beim Empfang können zwei Fälle auftreten:

1) Ein Prozeß wartet auf die Daten. In diesem Fall sind RAM-Adresse und Länge bekannt und die Daten können gleich in den Speicher geschickt werden.
2) Wartet noch kein Prozeß, so ist die ankommende Nachricht in einen Kommunikationsbereich des RAM zu legen und in einer Empfangswarteschlange zu vermerken.

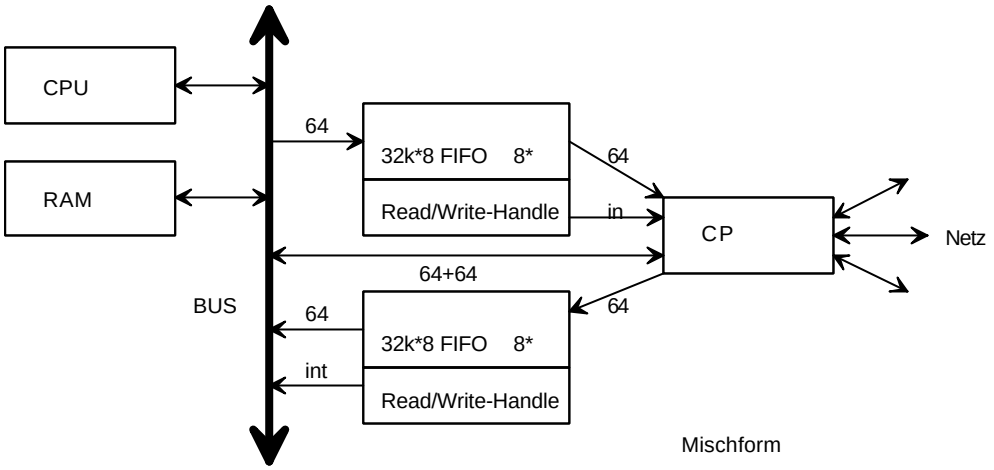


Abb. 37 Portorientierter CP-Anschluß über FIFO

6.2 ATM-Kopplung von Parallelrechnern

Zur Zeit werden in der Fachwelt verschiedene Ansätze zu einer prozessor- und plattformunabhängigen Parallelverarbeitung diskutiert. Dazu zählen PVM (Parallel Virtuell Memory) auf Workstations-Cluster. Diese Ansätze können effektiv nur für grobgranulare Aufgaben verwendet werden, da sonst der Kommunikationsoverhead noch größer wird als bei Transputerlösungen. Ursachen sind im Anschluß an das relativ langsame Ethernet (max. 10 MHz) sowie in der softwaremäßigen Plattformkonvertierung zu finden. Zum Anschluß von Parallelrechnern an Hochgeschwindigkeitsnetze stellt ATM (Asynchron Transfer Mode) eine günstige Alternative dar. Die ATM-Übertragungsebene stellt protokollmäßig Plattformunabhängigkeit her. Durch die Integrationsfähigkeit in höhere Geschwindigkeitsbereiche (Skalierbarkeit von ATM) kann ein WAN-Anschluß für massiv-parallele Rechner konzipiert werden. Viele Chip-Hersteller erkannten den Markt der sich mit der Bereitstellung von High-Speed-Chipsätzen eröffnen wird.

6.2.1 Zum ATM-Übertragungsverfahren

Übertragungsverfahren werden durch das verwendete Multiplexverfahren, die Vermittlungstechnik und den Verbindungsdienst unterschieden.

Alle Multiplexverfahren haben die Aufgabe, unabhängige Datenströme über ein Medium zu übertragen.

Beim synchronen Zeit-Multiplexing besteht der Übertragungsrahmen aus einer bestimmten Anzahl von Zeitschlitzen fester Größe. Jeder Benutzer bekommt einen bestimmten Zeitschlitz zugeordnet. Der Übertragungskanal kann durch die synchrone Lage des Zeitschlitzes im Übertragungsrahmen eindeutig identifiziert werden.

Im asynchronen Zeit-Multiplexing werden die Datenströme in Informationseinheiten fester oder variabler Länge umgewandelt und anschließend asynchron übertragen. Jedes Datenpaket wird mit einer Kanal-Identifikations-Nummer versehen. Ist die Länge der

Informationseinheit variabel wird das als Paketvermittlung, im Gegensatz zur festen Länge, der Zellenvermittlung, bezeichnet.

Vermittlung ist die Art mit der ein Übertragungspfad zwischen Sender und Empfänger ermittelt wird. Grundsätzlich können Leitungs- und Paketvermittlung unterschieden werden.

Bei der Leitungsvermittlung wird zwischen Sender und Empfänger vor der Datenübertragung eine physikalische Verbindung errichtet. Einer Verbindung kann eine bestimmte Bandbreite fest zugeordnet werden. Die paketorientierte Vermittlung wandelt die Datenströme in Datenpakete um. Die Bandbreite wird bedarfsorientiert angefordert, kann ggf. anderen Verbindungen zugeordnet werden. Es besteht die Gefahr von Paketverlusten und Paketreihenfolge ist nicht garantiert.

Die Verbindungsdienste können in die grundlegenden Verfahren verbindungsorientiert und verbindungslos eingeteilt werden.

Die verbindungsorientierte Kommunikation errichtet zunächst eine (virtuelle) Verbindung zwischen den Teilnehmern. Anschließend erfolgt die Datenübertragung. Der Vorteil dieses Verfahrens ist die eindeutige Paketreihenfolge sowie die Tatsache, daß bei Problemen während der Übertragung ist sofortige Reaktion möglich ist. Verbindungsorientierte Kommunikation impliziert Fehlerkontrolle sowie Sende- und Empfangsbestätigung im Verfahren.

Bei der verbindungslosen Kommunikation werden die Datenpakete ohne Empfangsbestätigung übertragen. Jedes Datenpaket enthält die komplette Zieladresse und wird unabhängig von den anderen Paketen durch das Netz transportiert. Die Paketreihenfolge ist nicht garantiert. Wegen des geringeren Verwaltungsaufwandes kann ein höherer Datendurchsatz als bei verbindungsorientierer Kommunikation erzielt werden.

Die Übertragungsverfahren können weiterhin in

1) Synchroner Transfer Mode (STM),
2) Paket Transfer Mode (PTM) und
3) Asynchroner Transfer Mode (ATM)

eingeteilt werden.

Das Übertragungsverfahren von ATM ist asynchrones Zeit-Multiplexing. Der Datenstrom wird in Datenpakete fester Länge, den ATM-Zellen, geteilt. Eine solche Zelle ist 53 Byte lang. Die ersten fünf Byte repräsentieren einen Header, der die Kanal- und Pfadadressierung enthält.

Es werden Zellenheader am User-to-Network-Interface (UNI) und am Network-to-Network-Interface (NNI) unterschieden.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Oktett
Generic Flow Controll				Virtual Path Identifier				1
Virtual Path Identifier				Virtual Channel Identifier				2
Virtual Channel Identifier								3
Virtual Channel Identifier				Payload Type			Priority	4
Headerr Error Controll								5

Tab. 38 Zellenheader am User-to-Network-Interface (UNI)

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Oktett
Virtual Path Identifier								1
Virtual Path Identifier				Virtual Channel Identifier				2
Virtual Channel Identifier								3
Virtual Channel Identifier				Payload Type			Priority	4
Headerr Error Controll								5

Tab. 39 Zellenheader am Network-to-Network-Interface (NNI)

Die Verbindung erfolgt über ein oder mehrere ATM-Schalteinheiten. Netzteilnehmer sind direkt mit der Schalteinheit verbunden. Sie teilen sich nicht ein gemeinsames Medium (wie z.B. im Ethernet-LAN), sondern übergeben ihre Zellen an den ATM-Schalteinheit ohne Rücksicht auf Zugriffsalgorithmen. Die benötigte Bandbreite wird von der Schalteinheit zur Verfügung gestellt. Damit können Dienste mit stark unterschiedlichem Bedarf gedeckt werden

6.2.2 Schaltsysteme für den Anschluß an die ATM-Datenautobahn

Ein ATM-Pfad setzt sich aus ATM-Kanälen zusammen. ATM-Schalteinheiten werden grundsätzlich als ATM-Switch oder als ATM-Cross-Connect ausgeführt. Die ATM-Pfadvermittlung (Cross-Connect) beendet die ankommende Pfade, einschließlich aller Kanäle, und leitet sie in einen anderen abgehenden Pfad um. Die ATM-Kanalvermittlung (ATM-Switch) beendet eingehende Pfade sowie Kanäle und leitet diese in abgehende Pfade bzw. Kanäle um. Die Kanalvermittlung impliziert das Schalten von Pfaden. In beiden Schaltvarianten bleiben die einzelnen ATM-Kanäle unberührt, d.h. die ATM-Zellen werden nicht aufgelöst.

Die Grundfunktionen der ATM-Schalteinheiten bestehen im identifizieren und auswerten der Kanal- bzw. Pfadidentifikation (Header) der ATM-Zelle sowie dem Transport der ATM-Zelle von einem Input-Kanal zu einem Output-Kanal der Schalteinheit. Das eigentliche Schaltwerk, die Switching-Fabric, hat die Aufgaben Input- und Output-Ports zur Verfügung zu stellen und dynamisch Übertragungswege konfliktfrei zu schalten. Die Switching-Fabric wird aus kleinen Schaltelementen, meist mit 8 bzw. 16 Input-Ports, aufgebaut. Die Schaltelemente bestehen aus einem Interconnection-Netzwerk, daß die Übertragungswege für die ATM-Zellen zur Verfügung stellt. Man unterscheidet grundsätzlich die zwei Arten: Matixstruktur-Netzwerk und Time-Division-Multiplexing-Netzwerk.

Ein Matrixstruktur-Netzwerk ist durch volle Erreichbarkeit, zeitgleiche Verbindung aller Zellen an den Input- zu den Output-Controllern und synchronem zentralem Takt gekennzeichnet. Die Ein- und Ausgänge werden durch ein Netz von Übertragungspfaden zusammengeschaltet. Mit Pufferspeichern an den Input- und Output-Ports können Blockierungen vermieden werden die durch den konkurrenten Zugriff zweier Ein- auf einen Ausgang entstehen.

Beim Time-Division-Multiplexing-Netzwerk wird eine gemeinsame Ressource, Bus bzw. Ring oder Speicher, genutzt.

Die Zellen werden auf eine gemeinsame Bus- bzw. Ringstruktur übertragen und darüber vermittelt. Die Bus-Topologie besitzt ein Interconnection-Netzwerk über einen 16- bzw. 32-Bit breiten Hochgeschwindigkeitsbus. Seine Übertragungskapazität muß größer als die Summe die der Input-Ports sein. Da die Geschwindigkeit des Busses größer als die des Output-Controller ist und ein Output-Controller bereits besetzt sein kann, sind Output-Kanal-Pufferspeicher nötig. In der Ring-Topologie werden alle Input- und Output-Controller über einen Ring miteinander verbunden. Der Vorteil gegenüber der Bus-Topologie ist, daß der Time-Slot während eines Ringumlaufes von mehreren Input-Controllern genutzt werden kann. Nachteilig wirkt sich der zusätzlicher Overhead zur Steuerung des Rings aus.

Ein andere Weg die Switching-Fabric zu realisieren besteht darin, die Zellen durch einen Input-Controller in einen gemeinsamen Speicher zu schreiben und von einem Output-Controller wieder ausgelesen. Dieses Verfahren wird als Central-Memory-Switching bezeichnet. Nachteilig ist die seriell-parallel-Wandlung des Zellenstromes. Durch die effiziente Speicherplatznutzung ist diese Variante zum Aufbau großer Systeme geeignet.

Schaltnetzwerke können entsprechend folgender Grafik eingeteilt werden:

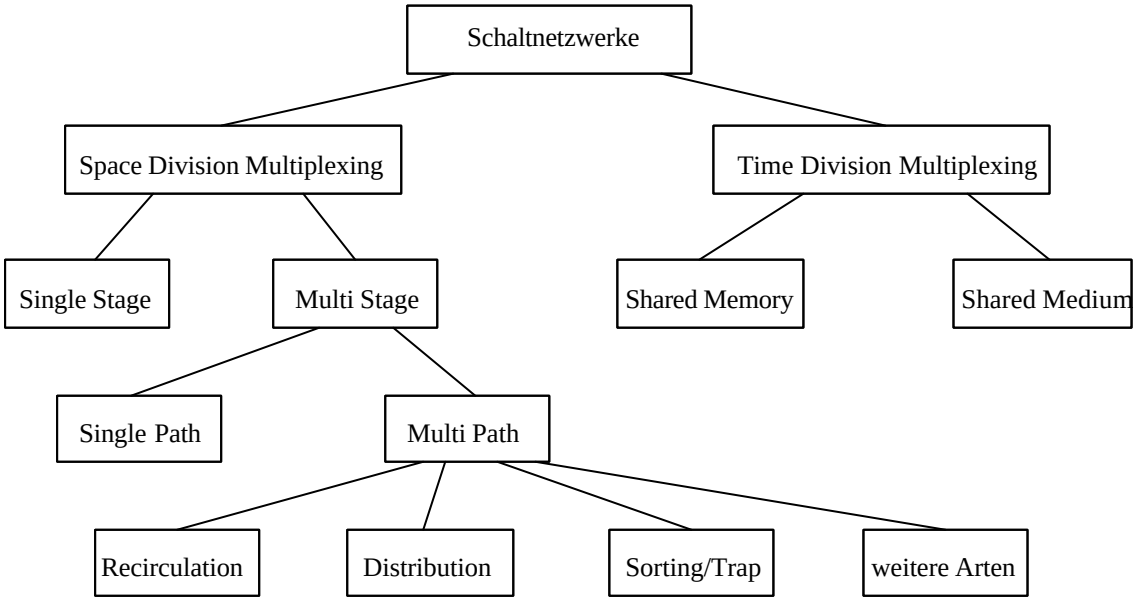


Abb. 40 Interconnection-Netzwerke nach [Kyas]

Single Stage-Netzwerke benötigen ein Schaltschritt vom Input zum Output. Sie sind zum Aufbau kleinerer Schalteinheiten geeignet. In bestimmten Arten dieser Netzwerkklasse (Shuffle Exchange), sind nicht alle Ausgänge erreichbar. Hier muß ein zusätzliches Feedback für einen erneuten Durchlauf der Zelle eingebaut werden muß.

Multi Stage Netzwerke eignen sich zum Aufbau größerer Schalteinheiten. Der Verbindungspfad wird in mehreren Schaltschritten errichtet. Beim Single Path Netzwerk (Banyan) ist nur ein sehr einfach zu adressierender Weg zulässig, allerdings sind hier Blockierungen möglich. Dagegen gestatten Multi-Path-Netzwerke mehrere Wege, wodurch

die Blockierungsgefahr verringert wird. Distribution-Netzwerke zielen auf möglichst gleichmäßiges verteilen des Eingangszellenstromes auf das Schaltnetzwerk, um eine hohe Auslastung und geringe Verklemmungsgefahr zu erreichen.

Die Zellenvermittlung kann in Eigenvermittlung und gesteuerte Vermittlung unterteilt werden.

Bei der Eigenvermittlung (Self-Routing) wird den zu vermittelnden Zellen ein zusätzlicher, schaltelementspezifischer Header vorangestellt. Das Schaltnetzwerk ist aus n Stufen aufgebaut. Der Header enthält n Teilfelder, die die Angaben über den einzuschlagenden Weg in jede Schaltstufe beinhalten.

In der Verbindungsaufbauphase wird zur gesteuerten Vermittlung (Table-Controlled-Routing) eine Tabelle für jedes Schaltelement initialisiert. Auf sie wird zur Vermittlungszeit zugegriffen. Die Kanal- bzw. Pfadidentifikation der Zellen wird in einen vermittlungsspezifischen Wert übersetzt mit dem der Output-Kanal identifiziert wird.

Als Auswahlkriterien für die Netzwerkarchitektur gelten folgende Eckpunkte:

1) Anzahl der IO-Ports,
2) Verlustrate,
3) Art und Größe des Pufferspeichers, die Größe wird von der Verlustwahrscheinlichkeit bestimmt und
4) Integrationsdichte und Anzahl der Chipbausteine.

Die Fachwelt wertet die Benes-Multipath-Architektur in großen Netzen, die Knockout-Architektur für kleine IO-Zahlen in Single-stage-Architekturen als günstig.

6.2.3 Chipsätze zum Anschluß an ATM-Highways

Die Firma Motorola dringt mit ihrem 'Autobahn Spanceiver' auf den Markt. Er gestattet 200 MByte/s auf der High-Speed-Seite. Mittels einer neuartigen Synchronisationsarchitektur wird lediglich ein Synchronisationsbit pro Datenbyte benötigt.

InterfaceTransferrate	Frequenz auf dem seriellen BUS
50MByte/s	225MHz
100MByte/s	450MHz
200MByte/s	900MHz

Tab. 41 Transferraten des Autobahn Spanceiver

Der bidirektionale Autobahnanschluß erinnert an die Ethernet-Topologie. Zusätzlich zur eigentlichen Autobahn wird ein traditioneller Parallelbus zum Transport von Synchronisationssignalen benötigt. Den Kommunikationsablauf legen die Kommunikationspartner über den Parallelbus fest. Nach der Synchronisation der Partner erfolgt der Datenaustausch und abschließend der Verbindungsabbau. Der Anschluß an einen Computer erfolgt über einen 16-/32-Bit Daten-Bus. Der Spanceiver erscheint als eine Ein-Gehäuse-Variante.

Das Produkt SMC91C100 - FEAST ist ein Fast Ethernet Controller für 10 und 100 Mbps nach dem CSMA/CD-Verfahren. Computerseitig wird ein 32 Bit breiter Datenpfad sowie synchroner, asynchroner und Burst DMA-Modus unterstützt. Zur Datenpufferung kann ein 128 KByte großer externer RAM eingesetzt werden. Der SMC91C100 ist insbesondere zum Einsatz in 16- bzw. 32-Bit INTEL x86 basierenden Umgebungen geeignet.

Relativ neu am Markt sind Lösungen des Workstations-Anbieters SUN. Der Sun ATM Adapter ist eine Einsteckkarte in den S-Bus, mit der die Kopplung an ein ATM-Netz mit der Spezifikation SONET/SDH möglich wird. Die z.Z. angebotene Bandbreite liegt bei 155Mbps. TCP/IP Applikationen werden in ATM-Zellen gekapselt und in bis zu 256 Kanälen übertragen. Voraussetzungen sind Solaris 2.4 computerseitig, sowie Multimodefiber bzw. UTP netzseitig.

Hewlett Packard bietet mit dem HDMP-1000 Chip-Satz die z.Z. schnellste Lösung für Peer-to-Peer Verbindungen an. Der computerseitige Anschluß erfolgt mit 16- bzw. 20-Bit. Hier liegt die maximale Datenrate bei 62,5 MWord/s, das entspricht 1250 MBit/s auf der seriellen Seite. Dieser Chip-Satz ist als Bus-Ersatz für größere Entfernungen (100m) geeignet. Ein ATM Einsatz ist nicht vorgesehen.

Die Produkte der Firma Wellfleet sind im WAN-Bereich angesiedelt. Die Lösungen basieren auf Standards und ermöglichen so die Zusammenarbeit verschiedener Plattformen. Durch Multiprotokoll Routing und Data Link Switching, unter anderem ist TCP/IP Encapsulation integriert, werden diese Produkte für PVM interessant. Zu den unterstützten Anschlußprotokollen zählt: TCP/IP, IBM DLSw, DECnet, AppleTalk, IPX, X.25, ATM, T1, T1/E1 und HSSI.

Die Firma TranSwitch stellt in ihrer ATM-Produktlinie eine Fülle von ATM-Chips zur Verfügung. Der Chip-Satz wurde für ATM-Konzentratoren, ATM-Hub's, lokale und Campus-ATM Switches sowie PC- und Workstations-Netzwerk Interface entwickelt. Der 'ATM Line Interface Device for 25 Mbit/s Operation TXC-07025' Chip-Satz ist als Kopplungsbaustein von ATM-Netz und Terminal konzipiert. Physisch werden STP und UTP bis zu einer Entfernung von 100m unterstützt. Aus den Bausteinen TXC-27325 (Kopplung an das Netzwerk), ALI-25T (Transceiver), ALI-25C (Controller) sowie SARA-S und SARA-R (terminalseitige Kopplung, Layer Protokoll Chip's) kann ein ATM-Anschluß realisiert werden. Der 25 Mbit/s-Anschluß kann über andere Chip's in ein größeres ATM-Netz eingekoppelt werden. Wegen der komplexen Produktlinie der Firma TranSwitch ist der ALI-25 Chip-Satz günstig geeignet, ATM-Schnittstellen für PC bzw. Workstations oder massiv-parallele Rechner zu realisieren.

Es kann festgestellt werden, daß mit dem Einsatz des ALI-25 Chip-Satzes ein ATM-Anschluß des parallelen Prozessorknoten möglich wird.

7 Zusammenfassung

Der Kommunikationsprozessor CP805 ist ein Router für massiv-parallele Rechner. Mehrere CP805 bilden zwischen den einzelnen Parallelknoten ein Kommunikationsnetz. Ein Parallelknoten besteht aus einem CP805 und einem Transputer T8xx. Benachbarte Knotenprozessoren werden von der Datenweitergabe entlastet. Durch den CP805 werden 2D-, 3D- und Hypercubus-Netze unterstützt. Die Steuerung des CP805 erfolgt in Abhängigkeit des Datenstromes dezentral. Die Linkarbitration innerhalb des Kommunikationsnetzes erfolgt wahlweise deterministisch oder adaptiv. Der CP805 unterstützt das serielle Transputerprotokoll der T8oo-Reihe. Der Arbitrationslayer des CP805 ist unabhängig vom Protokoll und kann in unterschiedlichen Protokolltypen eingesetzt werden. Das Routing erfolgt im innerhalb von Nanosekunden. Physisch ist der CP805 mit vier Links derzeit auf einem XILINX-FPGA XC4010 realisiert. Die Mehrzahl der Gatter werden für Synchronisation und Pufferung an jeder Link verbraucht. Entsprechend größere FPGA vorausgesetzt, ist das Konzept durch Vervielfachung der Links erweiterbar.

Die im CP805 verwendeten Algorithmen basieren auf effizienten deterministischen und adaptiven packet-routing Algorithmen (dynamischer Lastausgleich) für 2D-, 3D- und Hypercubus-Netzwerke von Jossifov und Krapp. Sie führen auf den deterministischen 'e-cube-routing' Algorithmus von Hayes und Thuazon und adaptive Strategien nach Jesshope u.a. zurück. Die verwendeten Algorithmen weisen eine vergleichbare Struktur, Konnektivität, ähnliche Kantengewichte und Zustandsfunktionen auf. Sie sind für deterministische und adaptive Netzwerkprozessoren zur verklemmungs- und konfliktfreien Steuerung von 2D-, 3D- und Hypercubus-Netzen untersucht worden.

Eine Analyse zeigte, daß die untersuchten Algorithmen auf eine effiziente Grundkonstruktion hinführbar und damit als Basis für einen Netzwerkprozessor geeignet sind. Die Umsetzung in einen PLD-Schaltkreis schafft die Möglichkeit, massiv-parallele Rechner in unterschiedlichen Netzwerkstrukturen zu verschalten, in dem der Schaltkreis mehrere Kommunikationsalgorithmen und Netzstrukturen unterstützt. Der CP805 kann für die Netztypen 2D-, 3D-Netz und nD-Hypercubus eingesetzt werden.

Der Kommunikationsprozessor wird als asynchrones, paralleles Schaltsystem mit synchronem Verhalten realisiert. Der Vorteil besteht in der konstanten Zeit, die für jede beliebige Kommunikationsentscheidung unabhängig vom Grad des Netzwerkes notwendig ist. Sie entspricht der Summe der Zustandssequenzen von zwei Automaten, die die Steuerung des Arbitrationslayers bilden.

Die Innovation des Kommunikationsprozessors CP805 gegenüber herkömmlichen Lösungen besteht in der reinen, extrem schnellen Hardwarelösung ohne Mikroprogramm auf Basis von Automatengraphen. Der Arbitrationslayer des CP805 ist vollständig in synthetisierbarem VHDL-Quelltext beschrieben.

Die Kanalarbitration erfolgt für alle Links zeitgleich und ist in Abhängigkeit der Anzahl der Links skalierbar. Werden bestimmte Schnittstellenbedingungen eingehalten, kann die Kanalarbitration für andere Protokolle eingesetzt werden. Die Kanalarbitration ist mit synthetisierbarem VHDL beschrieben. Die erreichte Verzögerungszeit reicht für ein

Adressfeld nicht über 40 Gate-Delays, das entspricht einem Byte, hinaus. Die Latenz bei einer geschalteten Verbindung entspricht ca. 5 Gatterlaufzeiten.

Ein n-dimensionaler Hypercubus besteht aus 2^n Knoten, die als n-dimensionale Konstruktion so zusammengeschaltet sind, daß jeder Knoten n Nachbarn besitzt. Die Nummer jedes Knoten kann als n-bit Binärstring dargestellt werden. Die Knotennummern benachbarter Knoten unterscheidet sich in nur einer Bitposition, die Hammingdistanz ist also eins.

Ein dreidimensionales Netzwerk besteht aus (x^n*x^n*x^n) Knoten, die als dreidimensionales Feld verbunden sind. Jede Achsrichtung besitzt n Knoten, die fortlaufend mit einer Integerzahl für jede Achsrichtung versehen ist.

Die Verzweigung der Kommunikationsflüsse erfolgt ereignisgesteuert. Mit dem Kommunikationsprozessor ist dynamische Clusterbildung und -veränderung möglich. Die Cluster sind online strukturierbar. Zwischen Transputer und CP805 existiert ein Sende- und ein Empfangskanal.

Trotz unterschiedlicher Adreßdarstellung (integer bzw. binär) erfolgt die Adreßrechnung in 2D/3D-Netzen und Hypercubus nach den gleichen Prinzipien. Die verwendeten Algorithmen gestatten deterministisches ebenso wie adaptives Routing. Die Wahl des Algorithmus und des Netztyps erfolgt online und ist programmierbar.

Die statische Leitungsbelegung bei stationärer Verbindung kann zu Dead Locks führen. Als Ausweg wurde Paketbetrieb mit Längenbegrenzung der Botschaften gewählt. Er hat die Eigenschaft, daß Standleitungen aufgebaut werden. Das aufwendige softwaremäßige sortieren von Paketen unterbleibt. Das Virtual Channel Prinzip, es können beliebig viele logischen Kanälen parallel benutzt werden, ist durch das Paketverfahren dem CP805 eigen. Bedingt durch die dezentrale Steuerung des CP805 werden hot spot's im Routing nicht beachtet.

Die Zusammenschaltung der CP805 erfolgt mittels bidirektionaler serieller Doppelleitung.

Der Prototyp des CP805 arbeitet mit einer Taktfrequenz von 20 MHz. Die Datenrate je Link ist 20 MBit/s entsprechend dem unterstützten T8xx-Protokoll.

Im Kommunikationsprozessor sind mehrere Pfade zeitgleich routbar. Ein Pfad besteht aus zwei Leitungen (DATA vom Quell- zum Zielknoten und Status vom Ziel- zum Quellknoten). Die Verbindung zwischen den einzelnen Netzwerkknoten erfolgt bitseriell.

Der Pfad wird dynamisch zur Übertragungszeit aufgebaut und während der Übertragung gehalten. Der Datenpfad wird schrittweise beginnend vom Quellknoten aufgebaut. Dazu wird ein Header der Botschaft vorausgeschickt. Jeder Knoten fügt nach erfolgter Arbitration ein Pfadsegment hinzu.

Verschiedene Arbitrationsstrategien werden unterstützt. Falls mehrere Eingänge simultan einen Ausgang benötigen, hat der Kommunikationsprozessor hat eine korrekte Konfliktlösung zu sichern.

Mit dem Empfang des Headers im Zielknoten startet der Bestätigungsprozeß. Eine RDY-Botschaft wird zum Quellknoten zurück gesendet. Sie folgt exakt dem Pfad in entgegengesetzter Richtung. Erreicht das RDY-Signal den Quellknoten, startet dieser die Datenübertragung.

Der Pfad wird abgebaut, wenn das Ende der Botschaft, gebildet durch ein EOM-Wort, den Kommunikationsprozessor passiert. Es weist den Kommunikationsprozessor an, den Kanal zum folgenden Knoten freizugeben. Die Kanäle lösen 'on the fly' aus. Im Zielknoten wird das

EOM-Wort aus der Botschaft entfernt, da es nicht Teil der Originalbotschaft ist. In der Ziel-CPU terminiert der Übertragungsprozeß.

Dieser Typ des Routing ist eine Form des Worm-Hole-Routing mit dem Unterschied, daß die Botschaft übertragen wird, nachdem der Pfad errichtet wurde unabhängig von der Botschaftslänge. Damit kann das System vollständig synchron arbeiten, ohne die Botschaft im vermittelnden Kommunikationsprozessor zwischenzuspeichern. Vorteil dieses Algorithmus ist, daß der laufende CPU-Prozeß im vermittelnden Knoten nicht unterbrochen wird.

8 Literatur

[Carp93] Hans-J. Carper: Kettenrechner. Elektronik Praxis 19/93, S. 30-32

[Deng93-1] Dengel, S.: Entwurf und Simulation eines Netzwerkprozessors für massiv-parallele Computer. Diplomarbeit. Fachhochschule für Technik und Wirtschaft Berlin, Fachber. Technische Informatik, 18.6.1993, 84 S.

[Deng93-2] Dengel, S., Heinz, G., Jossifov, V.: Entwurf und Simulation eines Netzwerkprozessors für massiv parallele Computer. Zwischenbericht Projekt NP-COMPAS, GFaI-Report 30.6.1993.

[Deng93-4] Dengel, S., Heinz, G., Jossifov, W.: Entwurf und Simulation eines Netzwerkprozessors für massiv parallele Computer. BMWi- Zwischenbericht 613/93, GFaI Report vom 30.6.1993.

[Deng93-5] Dengel, S.: CP805 Technische Dokumentation. Schaltungsdokumentation, Simulationsergebnisse, VHDL-Dokumentation. GFaI, 29.11.1993

[Deng94-1] Dengel, S., Heinz, G., Jossifov, V.: Entwurf und Simulation eines Kommunikationsprozessors für Massiv Parallele Computer. Projekt-Abschlußbericht NP-COMPAS, GFaI, 30.01.1994

[Deng94-2] Dengel, S., Busch, C., Heinz, G.: Ultraschneller, parallel routender Kommunikations- prozessor CP805 für 2D-, 3D-Torus- und Hypercube-Netze und Implementierung als FPGA für ein T805-Transputer-Cluster. Vortrag Workshop 'Hochgeschwindigkeitskommunikation in (parallelen) Rechnernetzen', GMD-FIRST/GFaI Berlin, 25.11.94

[Deng94-3] Dengel, S.: Kommunikationsprozessor CP805. Vortrag auf dem Transputer Anwender Treffen (TAT) 1994, Aachen 21.-22.9.1994

[Deng94-5] Dengel, S., Heinz, G.: Adaptiver Kommunikationsprozessor CP805 - Aufbau und Wirkungsweise. Vortrag in der Firma StageTec, Berlin-Johannisthal, 13.1.1994

[Deng95-1] Dengel, S.: Vortrag Kommunikationsprozessor CP805, Anwendervortrag auf der Veranstaltung 'Mentor Graphics auf einer HP-Plattform', Berlin 22.03.1995

[Getov91] Getov,V., Jesshope,C.R.: Simulation facility of distributed memory system with "mad postmen" communication work, Proc. Sec. Europ. Distrib. Memory Comp. Conference, april 1991, Munich.

[Heidk94] Heiduk, Falk (FHTW Berlin): Entwicklung von VHDL-Modellen eines Kommunikationsprozessors für 3D/2D-Netze und PLD-Synthese der Modelle. GFaI Report vom 16.2.1994, 93 S., Ingenieurpraktikumsarbeit, Betreuer: Dr. G. Heinz, GFaI.

[Heinz93] Heinz, G.: Lehrgang VHDL 1076 mit Mentor Autologic VHDL V.8.2, Fachhochschule für Technik und Wirtschaft, Berlin-Karlshorst, Zwei-Tages-Schulung,12./13.10.1993

[Heinz93] Heinz, G., Dengel, S., Jossifov, V.: Technical Documentation Network Communication Processor CP805. GFaI-Report 1.8.1993

[Heinz-1] Heinz, G.: Kommunikationsprozessor für PowerPC 601 - Aufgaben und Leistungsmerkmale. Vortrag zum GFaI-Workshop im Rahmen des Projekts NP-COMPAS "Konzept eines Kommunikationsprozessors für massiv parallele Rechner", Thesys GmbH Erfurt, 2.12.1993.

[Heinz-2] Heinz, G., Dengel, S.: Effiziente Routing-Verfahren für verteilte Netzwerke - Prinzipfindung für den CP805. Vortrag in der Firma StageTec, Berlin-Johannisthal, 13.1.1994

[Heinz-3] Heinz, G., Dengel, S., Busch, C., Jossivof, V.: Fast, Adaptive Communication Processor for 2D/3D-Torus and nD-Hypercube Nets. Proc. of the VI. International Workshop on Parallel Processing by Cellular Automata and Arrays Parcella '94, Sept. 21-23., University Potsdam

[Heinz-4] Workshop 'Hochgeschwindigkeitskommunikation in (parallelen) Rechnernetzen', GMD-FIRST Berlin, 25.11.94, Org.: GFaI e.V. Berlin, Dr. G. Heinz

[INMOS91] INMOS: The T9000-Transputer. Products Overv. Manual. INMOS/SGS-THOMSON, 1991.

[INMOS93-1] INMOS: Transputer Applications Notebook - Architecture and Software. INMOS/SGS-THOMSON, 1993.

[INMOS93-2] INMOS: Transputer Applications Notebook - Systems and Performance. INMOS/SGS-THOMSON, 1993.

[INMOS93-3] INMOS: The T9000-Transputer. Hardware Ref. Manual. INMOS/SGS-THOMSON, 1993.

[INMOS93-4] INMOS: The T9000-Transputer. Instruction Set Manual. INMOS/SGS-THOMSON, 1993.

[INTEL88] Intel Corp.: iPSC/2 System Manual. 1988

[IPSC/2] Intel Corp.: iPSC/2 System Manual. 1988

[Jessh86] Jesshope,C.R.: Dynamic load balanced active data model of parallel processing for vision, Proc. BCS, Workshop on parallel processing for vision, Oxford University, 1986.

[Jessh87] Jesshope,C.R.: Parallel processing based on active data, Proc. BIRA seminar on Parallel processing and super computing, Antwerpen, Nov.,1987.

[Jess88] Jesshope, C. R., Miller, P., Jantchev, J.: Programming with active data, Workshop PARCELLA'88, Berlin, 1988.

[Jessh90] Jesshope,C.R.: Communications architectures for finegrain parallel processing, Proc. Workshop PARCELLA'90, Berlin, 1990.

[Jessh90] Jesshope, C.R.: High performance Communication architectures, Proc. Workshop WP&DP'90, (Ed.) K.Bojanov, North-Holland, March 1990, Sofia.

[Joss90] Jossifov, V.: Parallel Computer architectures - theory and applications, Habilitation, Institut für Mathematik und Informatik der Bulg. Akad. der Wissenschaften, Dez. 1990, Sofia.

[Joss91] Jossifov, V., Krapp, M.: Adaptive packet routing algorithm for 3D-meshes described with parallel automaton nets, ZKI-Berlin Technical report 6/91.

[Joss91-2] Jossifov, V., Krapp, M.: Adaptive routing algorithm for 3D-mesh computer networks - formal description with automaton nets. 36. Int. Wiss.Kolloquium der TU Ilmenau, 24.-28.10.1991, Ilmenau.

[Joss92] Jossifov, V.: The challenge of the key technologies in High-Performance Computing - Some aspects, Deutsch-Französisches Kolloquium an der FHTW Berlin "Moderne Rechnerarchitekturen", 07-09.12.1992, Berlin.

[Joss93] Jossifov, V., Dengel, S., Heinz, G.: Inhalt und Ergebnisse des Projekts NP-COMPAS. Vortrag zum GFaI-Workshop im Rahmen des Projekts NP-COMPAS "Konzept eines Kommunikationsprozessors für massiv parallele Rechner", Thesys GmbH Erfurt, 2.12.1993.

[Krapp89] Krapp, M., Jossifov, V.: Formal specification of a distributed packet-routing algoritm with automation-nets, Technical report No.5, Bulgarian Academy of Sciences, Center for Informatics and Computer technology, Oct.1989, Sofia,

[Krapp90] Krapp, M., Jossifov, V.: Formal specification of a distributed packet-router for nD-hypercubes. Workshop on parallel and distributed processing, Bulgarian Academy of Sciences, March,1990, Sofia, Elsavier Sci. Publ., 1991, accepted for presentation also on Joint Conference on Vector and Parallel Processing - CONPAR 90, Zurich, Sep, 1990 and at Workshop on Algorithms and Parallel VLSI Architectures, June 1990, Pont-a-Mousson.

[Krapp91] Krapp, M, Jossifov, V.: Formal specification of a dynamic load balancing packet routing algorithm for binary nD-hypercubes, ZKI-Berlin Technical report 6/91.

[Krapp91-2] Krapp, M., Jossifov, V.: Formal specification of a distributed packet-router for nD-hypercubes. Workshop on parallel and distributed processing, Bulgarian Academy of Sciences, March,1990, Sofia, Elsevier Science Publ., 1991

[Kyas93] Othmar Kyas: ATM-Netzwerke: Aufbau, Funktion, Performance. DATACOM-Verlag Bergheim, 1993 (ISBN 3-89238-080-5)

[Lang82] C.R. Lang: The Extension of object - oriented languages to a homogenous, conncurrent architecture, Dept. of Computer Science, California Institute of Technology, Technical Report, 5014, May 1982.

[Menz94] Menzer, Lars (TFH Berlin): Entwicklung von Hypercube- VHDL- Modellen eines Kommunikationsprozessors und PLD- Synthese der Modelle. GFaI Report vom 16.2.1994, 88 S., Ingenieurpraktikumsarbeit, Betreuer: Dr. G. Heinz, GFaI.

[Pich91-1] Herbert Pichlik: Transputer-(r)evolution H1. ct 1/91, Heise Verlag, S.62-69

[Pich91-2] Herbert Pichlik: T9000-Offensive. ct 7/91, Heise Verlag, S.14

[Poun90] Pountain: H1-neue Transputer-Generation. ct 7/90, Heise Verlag, S. 34-43

[Schwa94] Schwabe, Ivo (TFH Berlin): Entwicklung von VHDL-Modellen und PLD-Synthese der Modelle. GFaI Report vom 16.2.1994, 21 S., Ingenieurpraktikumsarbeit, Betreuer: Dr. G. Heinz, GFaI.

[Sull77] H. Sullivan, T.R. Beshkow: A Large Scale Homogenous Machine. Proc. 4-th Annual Symposium on Computer Architecture, 1977, pp.105-124

[Tuaz85] Tuazon, J., Peterson, K., Puiel, K., Lirbermann, D.: Caltech/IPC Marc II Hypercube concurrent processor, Proc of ICPP, 1985, pp.666-673

9 Anlagen

Anlage A	CP805 Innenschaltung
Anlage B	VHDL-Quellen
Anlage C	Simulationsergebnisse der VHDL-Testbench
Anlage D	Synthesebeispiel Kanalarbitrierung
Anlage E	Package Information und Board
Anlage F	C-Quellen für Transputer
Anlage G	Report für Place & Route des FPGA