Max-Planck-Institut für Plasmaphysik

Garching bei München

Abt. Informatik, Prof. Dr. F. Hertweck

Multiprozessor mit dynamisch variabler Topologie (Kurzfassung)

H. Richter

IPP R/42 (Deutsch) 13.6.1988

Zusammenfassung

MULTITOP ist ein Multiprozessorsystem, bei dem die Interprozessor-Kommunikation nicht über einen gemeinsamen Speicher sondern über schaltbare Punkt-zu-Punkt-Verbindungen abläuft. Die Verbindungen können während des Programmablaufs umgeschaltet werden, so daß ihre Topologie der Topologie des Programms optimal angepaßt werden kann. Dies wird durch ein für alle Permutationen blockierungsfreies Koppelnetz erreicht, dessen modularer Aufbau eine beliebige Erweiterung des Systems gestattet. Am Beispiel der schnellen Fourier-Transformation und einer Spline-Approximation wird gezeigt, daß der Wirkungsgrad der Programmausführung sich gegenüber festen Verbindungen deutlich erhöht. Als Prozessoren werden Transputer verwendet. Die Programmiersprache ist OCCAM.

1. Einleitung

Diese Arbeit fügt sich ein in den Rahmen der Datenerfassung für das Großexperiment ASDEX-Upgrade des Max-Planck-Instituts für Plasmaphysik. Dort werden nach Inbetriebnahme des Experiments periodisch im Minutenabstand Millionen von Meßwerten anfallen, die selektiert und vorverarbeitet werden müssen. Dazu ist eine hohe Rechenleistung im experimentnahen Bereich notwendig, die kostengünstig nur mit Hilfe paralleler Architekturen erzielt werden kann. Weiterhin sind die Vorverarbeitungs-Algorithmen für die Daten in ihrem Rechenzeitbedarf sehr verschieden, so daß ein in seiner Leistung skalierbares Baukastensystem zur Datenverarbeitung vorteilhaft ist.

Die Methode der parallelen Rechnerarchitekturen ist ganz allgemein in den Bereichen der theoretischen und experimentellen Physik anwendbar, in denen einerseits eine hohe (Echtzeit)-Rechenleistung bei niedrigen Kosten realisiert werden soll, und andererseits die numerischen Algorithmen sich gut parallelisieren lassen. Mit der Parallelisierung der Hard- wie der Software sind aber neue Probleme verbunden, die die Anwendung dieser Methode auf breiter Basis bisher erschwert haben. In dieser Arbeit werden diese Probleme diskutiert und ein neues Konzept zu ihrer Lösung vorgestellt.

2. Das Konzept von MULTITOP

Die folgenden drei Probleme erschweren die Anwendung paralleler Architekturen im Rechnerbau:

1. Das Problem der parallelen Programmierung. Das heißt, daß es bislang keine einfache Methode gibt, einen parallelen Algorithmus auf einen Parallelrechner abzubilden. Damit verbunden sind die Fragen nach der optimalen Architektur eines Parallelrechners, nach seinem Kommunikationsmodell sowie der verwendeten Programmiersprache.

2. Das Problem der Effizienz bei paralleler Verarbeitung, die durch die stets notwendige Interprozessor-Kommunikation reduziert wird. Methoden zur Minimierung der Interprozessor-Kommunikation sind deshalb sowohl auf algorithmischer als auch auf architektonischer Seite notwendig.

3. Das Problem der Skalierbarkeit. Darunter versteht man die Möglichkeit, die Rechenleistung eines Systems in weiten Grenzen variieren zu können, um das Angebot an Leistung dem jeweiligen Bedarf anpassen zu können. Die Architekturmerkmale sollen dabei unverändert bleiben.

Das Konzept von MULTITOP ist ein Vorschlag zur Lösung dieser Probleme.

Das Kommunikationsmodell

Das MULTITOP zugrunde liegende Kommunikationsmodell basiert aus der Sicht des Benutzers auf sequentiellen Prozessen, die durch Botschaftenaustausch kommunizieren. Die Implementierung dieses Modells auf der physikalischen Ebene erfolgt durch Prozessoren (Transputer), die über sog. Kanäle miteinander verbunden sind. Als Programmiersprache wird OCCAM verwendet. Die Vorteile dieses Konzepts gegenüber der Kommunikation über einen gemeinsamen Speicher und gemeinsame Variable sind (Bild 2.1):

1. Der Fall, daß mehrere Prozessoren auf dieselbe Variable zugreifen, tritt bei Botschaftenaustausch nicht auf. Dadurch entfällt das Konsistenzproblem dieser Variablen bei schreibendem Mehrfachzugriff.

2. Die Zahl der über Kanäle gekoppelten Prozessoren ist nicht durch die endliche Bandbreite eines gemeinsamen Speichers limitiert.

3. Die Zuverlässigkeit der Interprozessor-Kommunikation ist wegen der Vielzahl verwendeter Kanäle größer als bei einem zentralen Bus.

4. Die Testbarkeit paralleler Programme wird dadurch erhöht, daß die Zahl der Testpunkte größer als bei gemeinsamen Variablen ist, weil jeder Kanal eine Schnittstelle mit definiertem Protokoll darstellt.

Die Nachteile dieses Kommunikationsmodells liegen in:

1. Gemeinsame Daten und Programme müssen mehrfach gehalten werden.
Aufgrund des Preisverfalls bei Speichern ist dieses Gegenargument jedoch relativiert.

2. Der Datenzugriff ist bei gemeinsamem Speicher prinzipiell schneller, da bei einem Kanal die Übertragungszeit zwischen den lokalen Speichern hinzukommt. Für den Fall eines Zugriffskonflikts auf den gemeinsamen Speicher kehren sich allerdings die Zeitverhältnisse um, da dann eine zeitaufwendige Arbitrierung der Zugriffe erforderlich ist.

Schließlich ermöglicht dieses Kommunikationsmodell die neue Eigenschaft eines Multiprozessors - die dynamisch variable Topologie - zu realisieren.

Die dynamisch variable Topologie

Warum ist eine dynamisch variable Topologie sinnvoll? Zur Klärung dieser Frage soll stellvertretend für viele Probleme aus der Numerik die Parallelisierung der schnellen Fourier-Transformation (FFT) und einer Spline-Glättung dargestellt werden.

Anhand des Signalflußgraphen der FFT wird ersichtlich (Bild 2.2), daß dieser Algorithmus in Stufen abläuft. Zwischen den Stufen werden Daten ausgetauscht, wobei die Art des Austauschs sich von Stufe zu Stufe ändert (Bild 2.3). Eine bestimmte feste Topologie von Prozessorverbindungen kann deshalb für die verschiedenen Phasen der FFT nicht optimal bezüglich der Interprozessor-Kommunikation sein. Weiterhin ist die Vereinigungsmenge aller Teil-Topologien zu komplex für eine reale Implementierung. Es ist also besser, die Topologien der Verbindungen zwischen den Stufen so zu verändern, daß sie dem Datenfluß der FFT entspricht.

Derselbe Sachverhalt wird beim Algorithmus einer parallelen Spline-Glättung deutlich: Bei dem von mir entwickelten Verfahren der Spline-Approximation wird mit Hilfe von sog. Basis-Splines eine Ausgleichskurve durch eine Folge von Punkten bzw. Meßwerten gelegt, wobei gleichzeitig eine Datenreduktion durch Glättung stattfindet. Dieses Verfahren wurde auch im Hinblick auf gute Parallelisierbarkeit entwickelt. Es muß dabei das lineare Gleichungssystem $\mathbf{A}^T\mathbf{A}\mathbf{p} = \mathbf{A}^T\mathbf{y}$ gelöst werden (Bild 2.4). Dazu sind mehrere Schritte notwendig, die - jeder für sich - eine eigene optimale Topologie von Prozessorverbindungen aufweisen:

1. Die Dateneingabe erfolgt am zweckmäßigsten von einem zentralen Punkt aus mit Hilfe einer sternförmigen Topologie zur Verteilung der Daten.

Kommunikationskonzept: aus logischer Sicht: Botschaften aus physikalischer: Kanäle	
Vorteile:	Nachteile:
o keine Synchronisation bei Mehrfachzugriff o Zahl der Prozessoren nicht begrenzt o erhöhte Zuverlässigkeit o bessere Testbarkeit	o gemeinsame Daten mehrfach o Kanäle langsamer als Speicher

Bild 2.1: Kommunikationskonzept

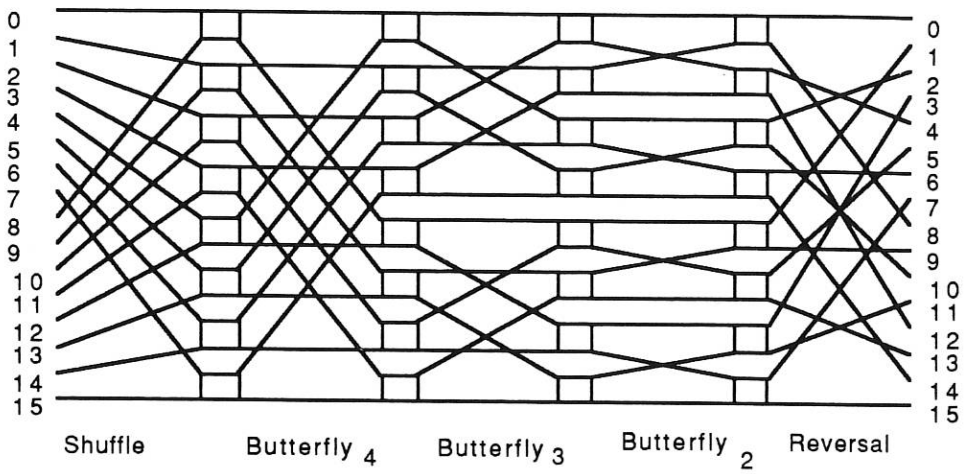


Bild 2.2: Signalflußgraph der schnellen Fourier-Transformation

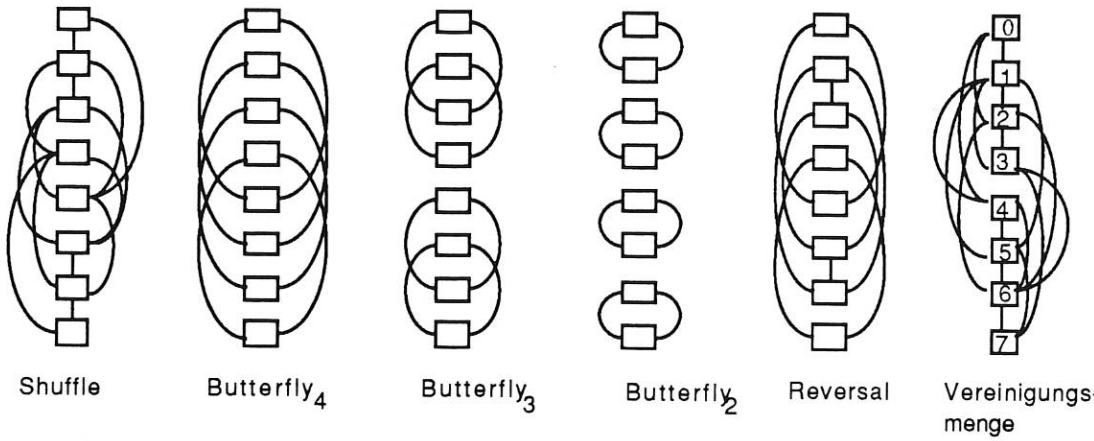


Bild 2.3: Sequenz von Topologien

$$A^T A p = A^T y$$

1. Daten eingeben: Stern
2. Matrix a aufstellen: Gitter
3. A transponieren: Shuffle
4. Matrix-Vektor-Mult.: Kette
5. lineare Gleich. lösen: Gitter
6. p ausgeben: Stern

Bild 2.4: parallele Spline-Glättung

2. Das Aufstellen der Matrix **A** kann dadurch erfolgen, daß jeder Prozessor eine Submatrix innerhalb einer gitterförmigen Topologie erzeugt.

3. Die Transposition von **A** kann in logN Schritten mit Hilfe eines Shuffle-Netzwerks realisiert werden.

4. Die Multiplikation von $\mathbf{A}^T$ mit **y** bzw. mit **A** erfolgt in einer Kette bzw. einem Gitter von Prozessoren am einfachsten.

5. Das Lösen des linearen Gleichungssystems kann mit Hilfe des Gauss-Jordan-Verfahrens auf einem Gitter optimal ablaufen.

6. Die Ausgabe der Unbekannten **p** erfolgt wiederum über den Stern.

In beiden Fällen - FFT und Spline-Glättung - ist eine dynamisch variable Topologie also sinnvoll. Für viele andere parallele Algorithmen gilt ebenso, daß sie phasenweise ablaufen, wobei für jede Phase ein optimale Topologie existiert. Ein Umkonfigurieren der Verbindungen zwischen den Prozessoren zur Laufzeit eines Programms ermöglicht deshalb, die Topologie des Rechners ständig an die Topologie des Problems anzupassen. Voraussetzung dafür ist, daß die Umschaltung schnell genug erfolgt. Bei MULTITOP wird eine zur Übersetzungszeit bekannte Topologie in ca. 10 µs eingestellt, so daß ein schneller Topologie-Wechsel möglich ist. Durch diese neue Eigenschaft können folgende Verbesserungen erzielt werden:

1. Paralleles Programmieren wird in der Hinsicht vereinfacht, daß sich der Rechner an das Programm anpaßt und nicht umgekehrt.

2. Die stets notwendige Interprozessor-Kommunikation wird dadurch minimiert, daß die Erzeuger von Daten direkt, d.h. ohne Einschalten von Zwischenstufen, mit den Verbrauchern verbunden sind. Dies wird durch die schaltbaren Punkt-zu-Punkt-Verbindungen der dynamisch variablen Topologie erreicht. Somit kann die Effizienz bei der Ausführung paralleler Programme gesteigert werden.

Aufgrund dieser Überlegungen ergibt sich eine bestimmte Architektur für das MULTITOP System:

Die Architektur

Ausgangspunkt der Darstellung der Architektur von MULTITOP ist die Idee, Prozessoren mit lokalen Speichern über Kanäle zu koppeln. Die Forderung der variablen Topologie erzwingt als Verbindungsnetzwerk entweder einen vollständig vermaschten Graphen (Bild 2.5) oder einen Kreuzschienenverteiler (Bild 2.6). Denn der vollständig vermaschte Graph enthält alle möglichen Topologien als Teilgraphen, bzw. diese lassen sich bei Bedarf mit Hilfe des Kreuzschienen-verteilers einstellen.

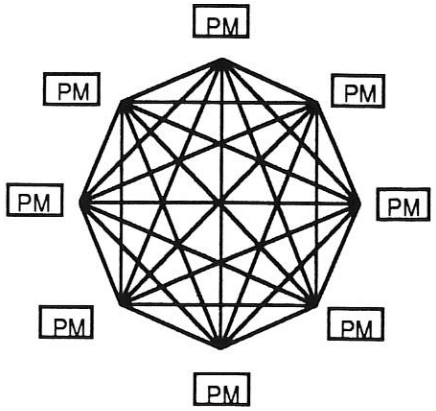


Bild 2.5: vollständig vermaschter Graph

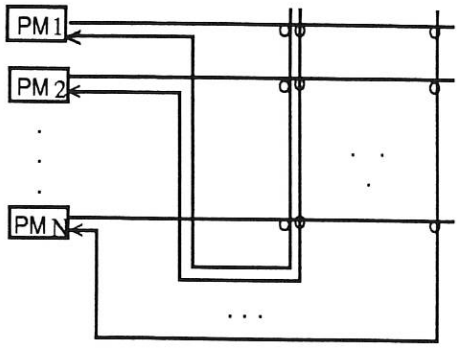


Bild 2.6: Kreuzschienenverteiler

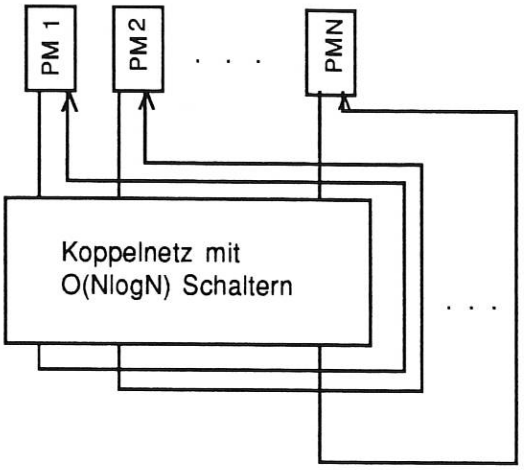


Bild 2.7: Koppelnetz

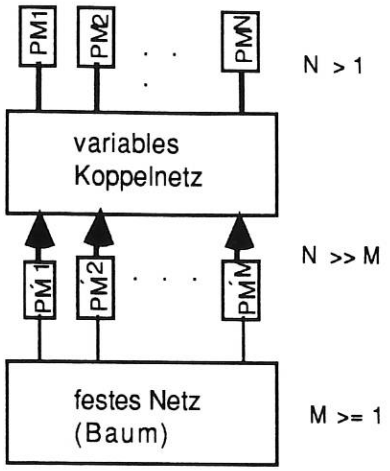


Bild 2.8: paralleles Setzen

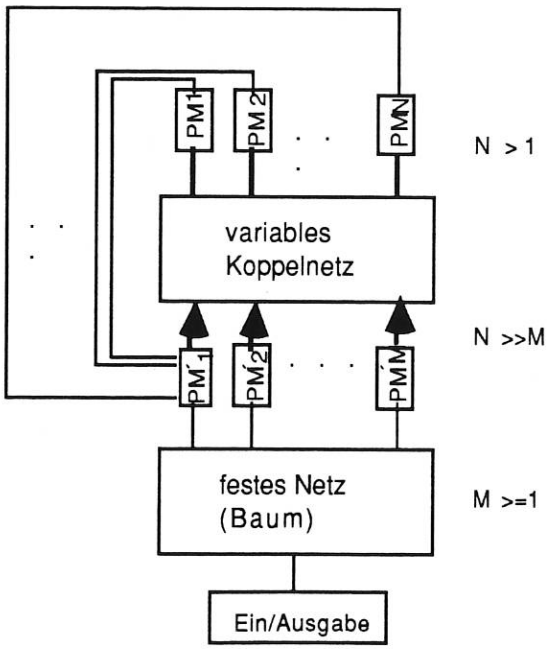


Bild 2.9: Stern zur Datenein/-ausgabe. PM1 ist Zentrum des Sterns

Beide Lösungen sind aber für die Kopplung vieler Prozessoren unwirtschaftlich, da sie $O(N^2)$ Kanäle bzw. Schalter erfordern. Die Lösung dieses Problems liegt in der Einführung von schaltbaren Verbindungen, die mit Hilfe eines Koppelnetzes, das aus O(NlogN) Schaltern besteht, realisiert werden (Bild 2.7).

Allerdings ist der mit der Durchschaltung der Verbindungen verbundene Zeitverlust, der durch das Setzen der Schalter des Netzes entsteht, so klein zu halten, daß er bei der Programmausführung nicht wesentlich ins Gewicht fällt. Dies führt zu der Idee, die Schalter des Netzes *parallel*, d.h. von mehreren Prozessoren gleichzeitig, setzen zu lassen. Im Prinzip könnten dazu dieselben Prozessoren PM1 - PMN verwendet werden, die auch dem Benutzerprogramm zur Verfügung stehen. Dieses würde dann allerdings in der Zeit, in der die Berechnung der Schalterstellungen vorgenommen wird, nicht ausgeführt werden können. Um diesen Zeitverlust bei jeder neuen Verbindung zu vermeiden, können zusätzliche Prozessor-Speicher-Module (PM´1 - PM´M) eingesetzt werden, deren Zahl M wesentlich kleiner als N sein kann (Bild 2.8). Die M Module müssen gemäß dem Algorithmus des Schalter-Berechnens miteinander verbunden sein. Diese Art der Verbindung kann fest sein, da stets dasselbe Programm ausgeführt wird. Geeignet ist eine Baum- oder Gitter-Topologie.

Bei den bisherigen Betrachtungen wurde die Datenein/-ausgabe nicht berücksichtigt. Eine schnelle Verarbeitung ist allerdings ohne eine ebensolche Datenein/-ausgabe wirkungslos. Bei Rechnern erfolgt sie zumeist von einer zentralen Stelle aus - der Schnittstelle zur Außenwelt. An diesem Punkt werden Eingabedaten eingespeist, die danach auf alle Prozessoren verteilt werden müssen. Nach der Verarbeitungsphase werden Ergebniswerte an dieser Stelle gesammelt und von dort ausgegeben. Aus Gründen der Auslastung ist es zweckmäßig, die zentrale Datenein/-ausgabe mit Hilfe derselben Prozessor-Speicher-Moduln vorzunehmen, die auch für das Koppelnetz zuständig sind, da diese nur während der Verarbeitungsphase mit dem Herstellen neuer Topologien beschäftigt sind. Dazu müssen sie in einer bestimmten, festen Topologie mit den Benutzer-Moduln verbunden sein. Für das Verteilen und Sammeln von Daten eignet sich eine sternförmige Topologie besonders, im Zentrum des Sterns steht die Schnittstelle zur Außenwelt. Somit ist dem Aufbau von MULTITOP noch ein Stern von festen Verbindungen überlagert (Bild 2.9).

3. Das MULTITOP-Koppelnetz

Das modulare Koppelnetz des MULTITOP-Rechners verbindet die N Eingänge des Netzes mit den N Ausgängen auf N! verschiedene Arten (Bild 3.1). Aus diesem Grunde ist das Netz für alle Permutationen von Punkt-zu-Punkt-Verbindungen blockierungsfrei. Dabei besteht es, wie das bereits bekannte Benes-Netz [1], aus nur (N/2)(2logN - 1) Schaltern, die aufgrund ihrer möglichen Stellungen - parallel oder gekreuzt- als Kreuzschalter bezeichnet werden. Aufgrund der nicht quadratisch anwachsenden Zahl von Kreuzschaltern ergibt sich für große N eine erhebliche Einsparung an Schaltern verglichen mit einem Kreuzschienenverteiler. So müssen z.B. für N = 1000 Eingänge beim MULTITOP-Netz ca. 10 000 Schalter aufgewendet werden, während der Kreuzschienenverteiler bereits 1 Mio. Schalter erfordert.

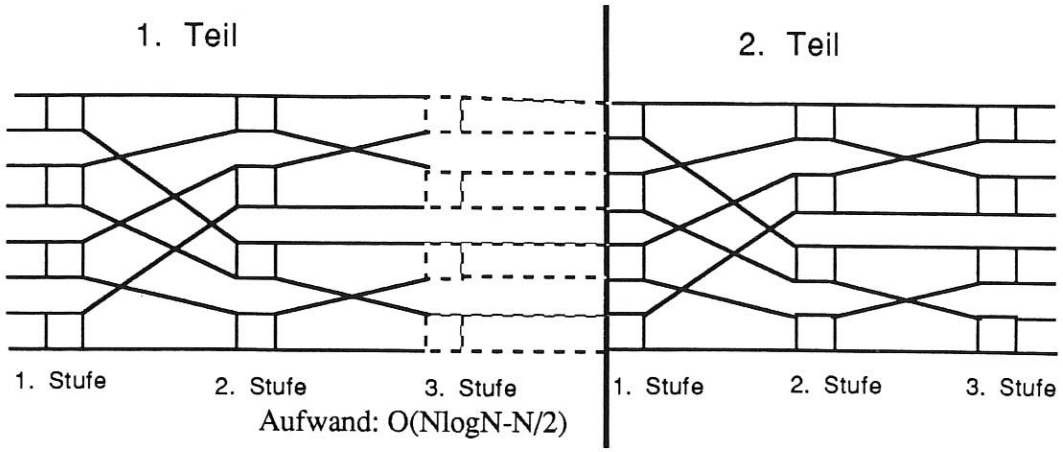


Bild 3.1: MULTITOP-Netz

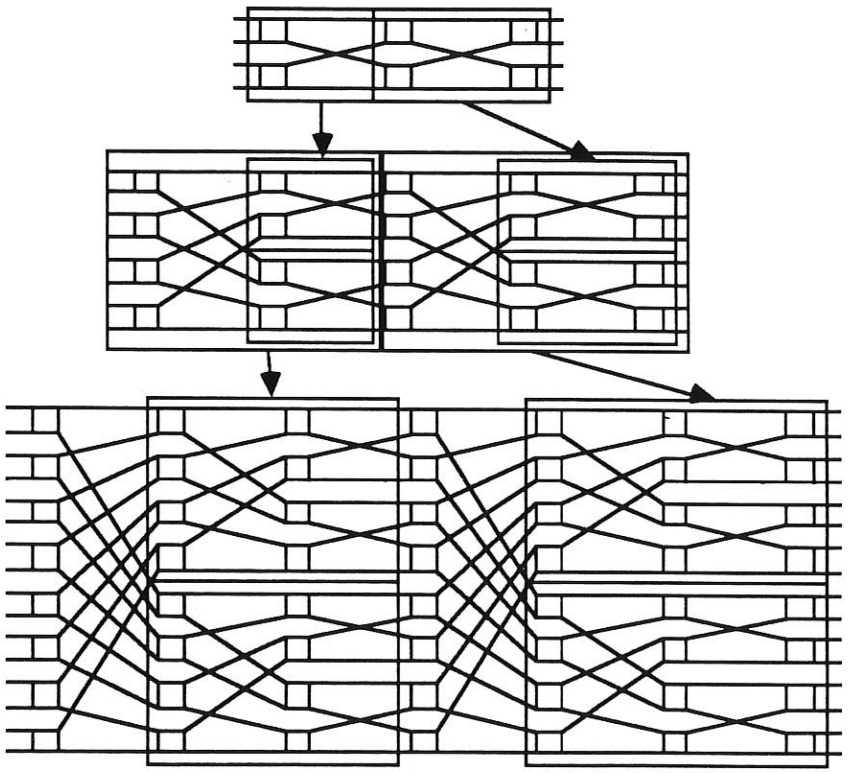


Bild 3.2: Modulare Erweiterung

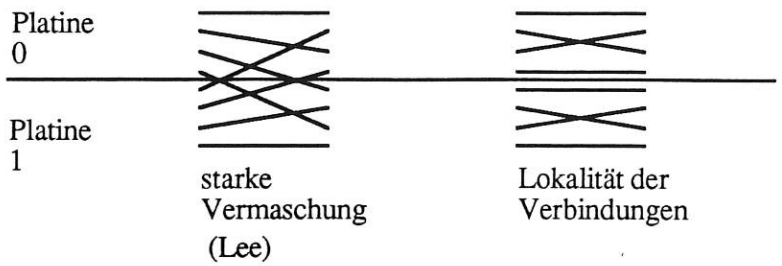


Bild 3.3: Geringe Vermaschung

Der technische Vorteil des MULTITOP-Netzes gegenüber dem Benes-Netz liegt darin, daß es aus zwei baugleichen Moduln besteht. Dadurch senken sich die Produktionskosten, weil die Stückzahl der Module sich verdoppelt. Ebenso wird ihr Test und ihre Entwicklung vereinfacht, weil nur ein Typ statt zweien existiert. Weiterhin weist das MULTITOP- Netz eine Selbstähnlichkeit in seiner Topologie auf, die bewirkt, daß die beiden Moduln eines bestehenden Netzes rekursiv in einem Netz mit der doppelten Zahl von Eingängen weiterverwendet werden können (Bild 3.2). Bei einem Netz mit vier Eingängen z.B. können der linke und der rechte Teil dieses Netzes unverändert in einem Netz mit 8 Eingängen weiterverwendet werden. Dieser Sachverhalt gilt genauso für ein Netz mit 16 Eingängen u.s.f..

Die Selbstähnlichkeit der Topologie erlaubt weiterhin, ein bestehendes Netz modular zu erweitern, wobei mit jeder Verdopplung der Zahl der Eingänge sich auch die Kommunikations- Bandbreite verdoppelt. Diese modulare Erweiterbarkeit kann z.B. dazu genutzt werden, die Netto-Rechenleistung eines Multiprozessor-Systems nach dem Baukastenprinzip zu erhöhen.

Ein weiterer Vorteil ist beim Vergleich mit dem ebenfalls bekannten Lee-Netz [2] zu erkennen (Bild 3.3): Da die Vermaschung im Netz von Stufe zu Stufe abnimmt, ist die Verdrahtung insgesamt wesentlich lokaler. Dies bedeutet für ein Netz, das auf mehreren Platinen realisiert ist, eine weitere Kostenersparnis, da weniger Steckverbinder und Kabel zwischen den Platinen notwendig sind. Das MULTITOP-Netz kann so auch in der Telefon-Vermittlungstechnik in den Fällen, in denen ein Lee- oder Benes-Netz Verwendung findet, eingesetzt werden.

Schließlich kann man zeigen, daß bei Verwendung von Kreuzschaltern mit zusätzlicher Broadcast-Funktion das ganze Netz auch als Broadcast-Netz verwendet werden kann.

Wie funktioniert das MULTITOP-Netz?

Eine äquivalente Betrachtungsweise für die Frage der Wegewahl eines Koppelnetzes ist, wenn man das Netz als eine Maschine zu Sortieren von Zahlen ansieht (Bild 3.4). Am Eingang der Sortiermaschine wird eine Permutation von Zahlen eingespeist, am Ausgang erhält man die Zahlen in ihrer natürlichen Reihenfolge. Das Problem, Eingänge mit Ausgängen zu verbinden, ist also eng verknüpft mit der Aufgabe Zahlen zu ordnen. Dies führt zu dem analogen Problem, in einem Speicher Zahlen zu sortieren (Bild 3.5). Die adressierbaren Zellen des Speichers entsprechen dann den verschiedenen Eingängen des Netzes.

Das analoge Problem ist deshalb einfacher zu lösen, weil Aussagen wie " die obere und untere Hälfte der Eingänge " oder " die zwei Eingänge eines Kreuzschalters" leicht durch die höchst- bzw. niederstwertigen Adressbits dargestellt werden können. Damit kann das folgende neue Verfahren zum Sortieren von Zahlen angegeben werden (Bild 3.6):

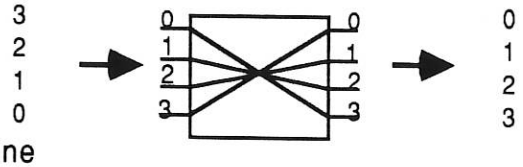


Bild 3.4: Sortiermaschine

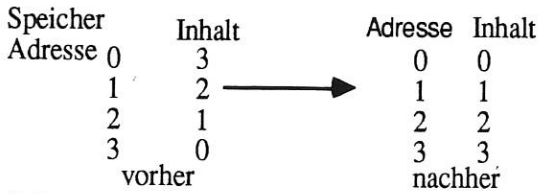


Bild 3.5: Zahlen im Speicher

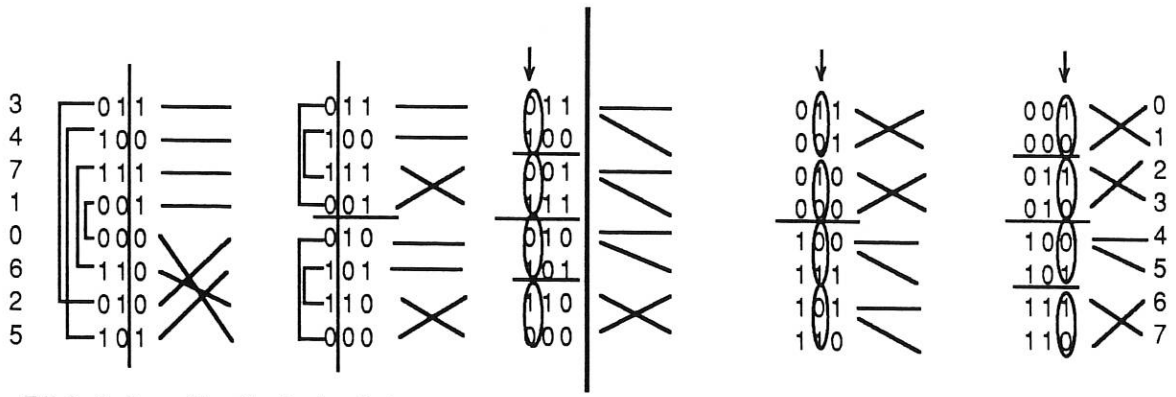


Bild 3.6: Sortierbeispiel

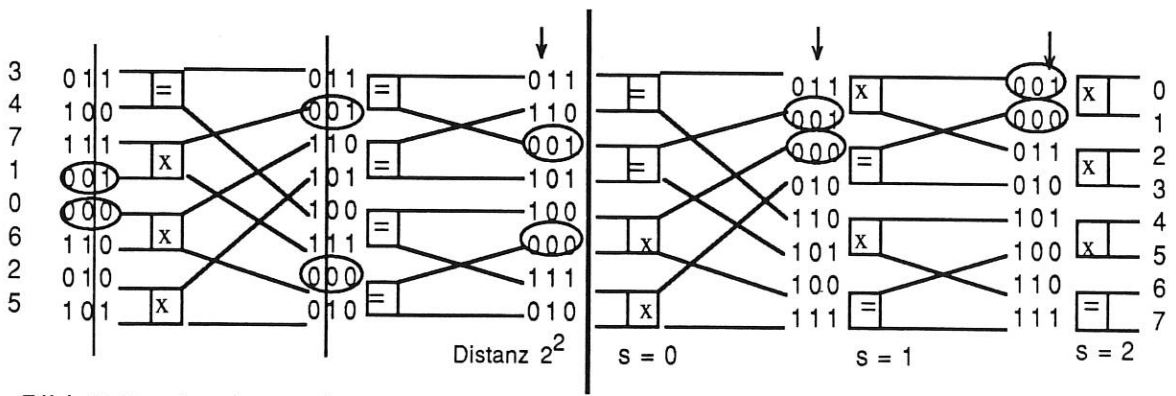
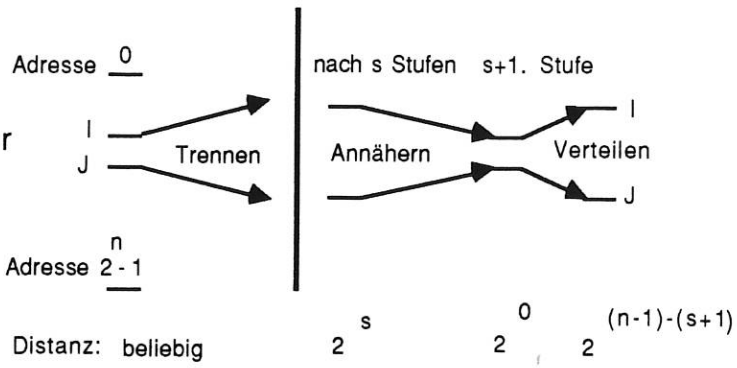


Bild 3.7: Implementierung

Bild 3.8:

Blockierungsfreier Weg



Im ersten Schritt werden aus den N Zahlen N/2 Paare gebildet. In jedem Paar sollen sich die zwei Zahlen nur in ihrem niederwertigsten Bit unterscheiden, d.h. ihre logN - 1 höherwertigen Bits sind gleich. Da es sich um eine Permutation von N Zahlen handelt, und N eine Zweierpotenz ist, gibt es zu jeder Zahl genau einen Partner, der diese Bedingung erfüllt. Die Paarbildung kann deshalb eindeutig und vollständig ablaufen.

Danach werden die beiden Zahlen jedes Paares dadurch getrennt, daß sie in die obere und untere Hälfte des Adressraums verteilt werden. Dabei ist es irrelevant, welche Zahl in welche Hälfte kommt, wichtig ist nur, daß sie in verschiedene Hälften transportiert werden.

Der erste Schritt wird nun rekursiv auf die obere und untere Hälfte des Adressraums angewandt, wobei jetzt 2x(N/4) Paare mit logN - 2 gleichen höherwertigen Bits gebildet werden. Der erste Teil des Sortierverfahrens terminiert nach logN - 1 Schritten, da dann keine Zahlen mit gleichen Bits im jeweiligen Adressraum mehr existieren.

Der zweite Teil des Sortierverfahrens ist bereits bekannt. Er wird auch als sukzessive Approximation bezeichnet, da in jedem Schritt die Genauigkeit des Ziels verdoppelt wird. Nach logN Schritten ist das Ziel auf logN Bits genau.

Neu ist, daß in jedem Schritt jeweils ein Paar benachbarter Zahlen in komplementäre Hälften verteilt werden kann, weil deren relevante Bits komplementär sind. Diese neue Eigenschaft wurde durch den 1.Teil des Sortierverfahrens erreicht. Damit verbunden ist aber auch die Möglichkeit, das Sortierverfahren mit Hilfe eines Netzes, das aus Kreuzschaltern und einer sog. Unshuffle-Verdrahtung besteht, zu implementieren (Bild 3.7). In diesem Netz ist die Unshuffle- Verdrahtung dafür zuständig, in verschiedene Hälften zu transportieren, während je nach Stellung des Kreuzschalters entschieden wird, in welche Hälfte transportiert wird.

Für ein blockierungsfreies Arbeiten des rechten Teils des Netzes ist es notwendig, daß an jedem Kreuzschalter zwei Zahlen mit komplementärem Steuerbit anliegen, da diese aufgrund der Unshuffle-Verdrahtung in komplementäre Hälften transportiert werden. Um dies sicherzustellen, ist der erste Teil des Sortierverfahrens bzw. der linke Teil des Netzes notwendig. Die Idee, die dabei zugrunde liegt, entspringt folgender Beobachtung (Bild 3.8):

Wenn sich im rechten Teil des Netzes zwei Zahlen an einem Kreuzschalter der Stufe s treffen, müssen diese Zahlen s gleiche höherwertige Bits aufweisen. Denn beide steuern ihr Ziel nach dem Prinzip der sukzessiven Approximation an, wobei ihre s höherwertigen Bits als Steuerbits verwendet wurden.

Damit der Kreuzschalter, an dem die beiden Zahlen anliegen, konfliktfrei gesetzt werden kann, muß ihr s+1. Steuerbit komplementär sein. Das Prinzip des linken Teils ist es also, Paare mit s gleichen höherwertigen Bits zu bilden, und die beiden Zahlen dann so weit zu trennen, daß sie sich im rechten Teil des Netzes erst nach s Stufen treffen können. Dazu müssen sie am Eingang des

rechten Teils die Distanz 2^s im Adressraum aufweisen. Denn wenn ihre Distanz kleiner als 2^s wäre, würden sie sich bereits vor der Stufe s treffen. Die dann relevanten Steuerbits sind nach Voraussetzung aber nicht komplementär, so daß ihr Kreuzschalter nicht widerspruchsfrei gesetzt werden könnte. Der Algorithmus des 1. Teils des Sortierverfahrens beschreibt also einen Vorgang, der auch als "Konfliktpaartrennung" bezeichnet werden kann.

Entscheidend ist nun, daß ich zeigen konnte [3], daß dieser Algorithmus auf einem Netz derselben Topologie wie das für die sukzessive Approximation zuständige Netz ausgeführt werden kann. Deshalb besteht das MULTITOP-Netz aus zwei gleichen Teilen mit allen daraus resultierenden Vorteilen.

4. Ergebnisse

Es wurde ein Prototyp von MULTITOP mit vier Rechen-Transputern und dynamisch variabler Topologie realisiert (Photos). Die Implementierung der FFT auf diesem Prototyp sowie die Implementierung der Spline-Glättung lieferten bezüglich der drei betrachteten Probleme die folgenden Resultate:

1. Problem der parallelen Programmierung: Die verwendete Programmiersprache OCCAM basiert auf kommunizierenden sequentiellen Prozessen. Es wurde in der Praxis bestätigt, daß die parallele Programmierung dadurch vereinfacht wird, daß die Topologie der Prozesse direkt auf die Topologie der Prozessoren abbildbar ist. Auf maschinengegebene Besonderheiten mußte so keine Rücksicht genommen werden. Das Konzept der dynamisch variablen Topologie erwies sich deshalb als große Hilfe bei der Implementierung der parallelen Programme.

2. Problem der effizienten Ausführung: Die zu parallelisierenden Probleme wurden einer genauen Datenflußanalyse zur Minimierung der Interprozessor-Kommunikation unterzogen. Es wurden dabei die vier sequentiell gleichwertigen Formen der FFT, die als Cooley-Tukey- (CT), Gentleman-Sande- (GS), Stockham- (ST) und Pease-FFT (PS) bezeichnet werden, als nicht gleichwertig bezüglich ihrer parallelen Ausführung erkannt, da deren Interprozessor-Kommunikation sich bis zum Faktor drei unterscheidet (Bild 4.1) ! Es wurde daraufhin die Gentleman-Sande-FFT, als die Form mit der kleinsten Interprozessor-Kommunikation implementiert. Dabei zeigte sich ein hoher Wirkungsgrad von ca. 90% bei zwei Prozessoren bzw. 80% bei vieren gegenüber einem Prozessor (Bild 4.2). Um auf eine große Zahl von Prozessoren extrapolieren zu können, wurde ein mathematisches Modell entwickelt, in das die reine Berechnungszeit der FFT, die Interprozessor-Kommunikation, sowie die Topologie der Prozessorverbindungen eingeht (Bild 4.3). Die Topologie drückt sich darin durch die "mittlere Entfernung" zweier Prozessoren aus. Anhand des Modells zeigte es sich, daß das MULTITOP-Konzept mit d = 1 gegenüber dem Hypercube (d -> LogP) und dem Ring (d -> P/4) eine wesentlich größere Effizienz aufweist (Bild 4.4).

$$E_{CT}(n,p) \approx 2^{n-1}(p+4) \qquad E_{CT} = E_{GS} \qquad p = \log P,\ n = \log N$$

$$E_{ST}(n,p) \approx \begin{cases} 2^{n-1}(p+6) \\ 2^{n-1}(p+5) \end{cases} \qquad \frac{E_{PS}}{E_{GS}} = 2{,}75 \quad \text{für } N = 1024,\ P = 16$$

$$E_{CT} = E_{GS} < E_{ST} < E_{PS} \quad \text{für } p + 6 < 2(n+1) \qquad E_{PS}(n,p) \approx 2^{n-1}\,2(n+1)$$

Bild 4.1: Datenflußanalyse von sequentiell gleichwertigen Formen der FFT

	Zahl der Prozessoren		
	1	2	4
Gesamtzeit [ms]	478	276	152
Speedup	-	1,73	3,13
Wirkungsgrad	-	0,87	0,78
Transport/Rechn.	0,44	0,52	0,58

Bild 4.2: Messungen an MULTITOP

Gesamtzeit: $T_{FFT}(n,p,d) = 2^{n-p}(an + bd(1 + (p/4)\ -2^{-p}))$

Daten-Transportzeit: $T_{MOV}(n,p,d) = bd2^{n-p}(1 + (p/4)\ -2^{-p})$

Wirkungsgrad: $e(n,p,d) = \dfrac{n}{n + cd(1 + (p/4)\ -2^{-p})}$

a,b : Konstanten
d: mittlere Entfernung
c = b/a

Bild 4.3: Modell zur Extrapolation auf viele Prozessoren

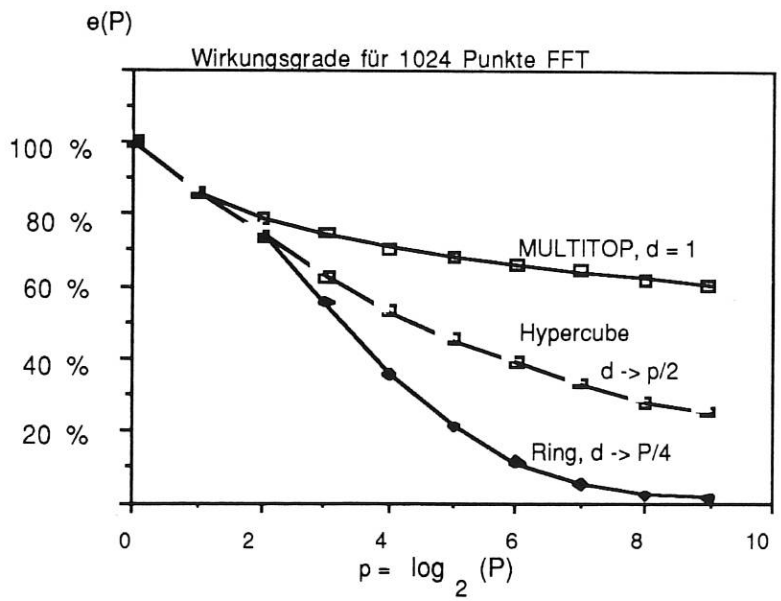


Bild 4.4: Hoher Wirkungsgrad bei MULTITOP

Für die zu parallelisierende Spline-Glättung mit Datenreduktion wurde zur Minimierung der Interprozessor- Kommunikation ein neues Verfahren entwickelt, das sich besonders gut parallelisieren läßt, da es ausschließlich auf Operationen der linearen Algebra, wie Matrix-Matrix-Multiplikationen basiert.

Insgesamt kann gesagt werden, daß eine hohe Effizienz bei paralleler Verarbeitung dadurch erzielt werden kann, daß die Interprozessor-Kommunikation sowohl auf algorithmischer Seite durch eine Datenflußanalyse als auch auf architektonischer Seite durch eine dynamisch variable Topologie verringert wird.

3. Problem der skalierbaren Leistung: Der Prototyp von MULTITOP wurde zwischenzeitlich auf acht Rechen-Transputer erweitert. Die dazu notwendige Verdopplung der Zahl der Eingänge des Koppelnetzes erfolgte unter Verwendung des vorhandenen Netzes. Das neu entwickelte MULTITOP-Koppelnetz erfüllt also die Forderung nach modularer Erweiterbarkeit.

Literaturhinweis:

[1]: V.E.Benes, Math. theory of connecting networks and telphone traffic, Academic Press, 1965.

[2]: K.Y.Lee, IEEE Transactions on computers, Vol. c-34, No.5, May 1985.

[3]: H. Richter, Multiprozessor mit dynamisch variabler Topologie, Doktorarbeit TU München, Fakultät für Elektrotechnik und Informationstechnik, 1988.