Konzept eines verteilten Multiprozessorsystems

Clemens Kordecki Universität Karlsruhe

Kurzfassung: In diesem Bericht wird ein hardwarerealisiertes Konzept zur Kommunikation und Synchronisation von Prozessen in einem verteilten System vorgestellt. In [10] wird dargestellt wie der Aufwand zur Kommunikation und Synchronisation verringert wird, wenn die Kommunikation durch implizite Prozesse kontrollierbar wird. Ein solcher Kommunikationskanal ist asynchron. Synchrone Kommunikation ist in diesem Sinne ein Spezialfall der asynchronen Kommunikation. Die impliziten Kommunikationsprozesse, die durch Hardware realisiert werden, bilden die Basis eines verteilten Multiprozessorsystems.

Schlüsselwörter: Multiprozessorsysteme, Prozeßkommunikation, Synchronisation

1. EINFÜHRUNG

Die Trennung von funktionalen Einheiten (Modulen) hat sich in der Programmierung seit langem als vorteilhaft erwiesen. Durch das Definieren von Schnittstellen werden diese Einheiten separat test- und verifizierbar.

Als Konsequenz dieser guten Erfahrungen soll diese Trennung auch auf Betriebssystemteile und verteilte Applikationen angewandt werden. Der Bereich, der hier angesprochen wird, ist die Kommunikation und Synchronisation von verteilten Prozessen.
Ziel ist es, die Kommunikation vor dem Anwendungsprogrammierer zu verstecken und die Synchronisation implizit an die Kommunikationsobjekte zu knüpfen. In der Softwaresicht werden die Kommunikationsprozesse als Objekte modelliert, deren Benutzung eine spezielle Disziplin voraussetzt [11]. Die für die Kommunikation benutzten Datenobjekte werden Kanäle genannt. Sie können als separate Prozesse aufgefaßt werden, die für eine konkrete Kommunikation durch ihre Installationsparameter auf die jeweiligen speziellen Anforderungen abgestimmt werden. Auf Grund ihres einfachen Aufbaus realisieren wir diese Prozesse direkt durch Hardware, einem Kommunikationsspeicher innerhalb jedes Multiprozessorknotens, mit festen Zugriffsoperationen. Die Realisierung durch Hardware reduziert den Aufwand des Kommunikationsprotokolls auf einen Speicherzugriff.

Die möglichen Kommunikationsdisziplinen lassen sich durch die folgenden Pfadausdrücke beschreiben:

```
(w r )*
(w*r )*
(w r*)*
(w*r*)*
```

Prozesse dieses verteilten Multiprozessors sind Aggregationen von lokalen Daten, den Zugriffsoperationen auf diese Daten und sequentielle Aktionen. Diese Prozesse beschreiben modular und anonym zueinander die eigentliche Applikation. Ein wesentlicher Gesichtspunkt an dieser Stelle sind die lokal deklarierten Eingabe- und Ausgabe- Ports. Diese Ports stellen die Kommunikationskanäle zu anderen Prozessen dar. Eine virtuelle Verbindung dieser Ports wird während der Laufzeit eingerichtet und nach Termination der beteiligten Prozesse wieder abgebaut.

Im Gegensatz zu diesen Anwender- Prozessen sind die Kommunikationskanäle als implizite Prozesse im System integriert. Sie sind ebenso wie die Applikationsprozesse strukturiert. Lokale Daten mit den Zugriffen put und get und eigene Aktivitäten wie Init, Veto und Interrupt characterisieren diese Prozesse. Im nachfolgenden Kapitel wird auf diese Prozesse näher eingegangen.

2. KOMMUNIKATIONSKANÄLE

Grundidee dieses Verfahrens ist die Verlagerung des Kommunikationsprotokolls in einen separaten Prozeß, in dem Maßnahmen zur Synchronisation der beteiligten Prozesse parallel zu den Aktivitäten dieser Prozesse ausgeführt werden. Ergebnis dieser Separation ist eine hochgradig parallele Ausführung der beteiligten Prozesse, weil Synchronisationsmaßnahmen immer zum spätest möglichen Zeitpunkt einsetzen.
Aus der Sicht unserer geplanten Anwendung, der Steuerung und Regelung industrieller Prozesse, besteht ein verteiltes Multiprozessor-System aus einer Vielzahl einzelner Verarbeitungsknoten, die nach den Anforderungen des zu steuernden Prozesses physikalisch verteilt und miteinander vernetzt sind. Da es nicht vorhersehbar ist, welche zusätzlichen Anforderungen während des Betriebs entstehen, die neue Kommunikationswege, aber auch neue Prozesse nach sich ziehen, soll das System maximale Flexibilität bieten. Für unseren Entwurf bedeutet dies beliebige Kommmunikationswege zwischen allen Knoten und die Möglichkeit, Änderungen der Prozeßverteilung am laufenden System durchzuführen.
Jeder Verarbeitungsknoten im System ist ein Multiprozessor, der über ein globales Bussystem mit den anderen Knoten verbunden ist (Abbildung 1).

Ein Multiprozessor besteht aus folgenden Baugruppen:

- Einer Schnittstelle zum globalen Bussystem, das Fehlertoleranz und Übertragungssicherheit für das Netzwerk sichert. Diese Schnittstelle wird durch einen separaten Kommunikationsprozessor realisiert.

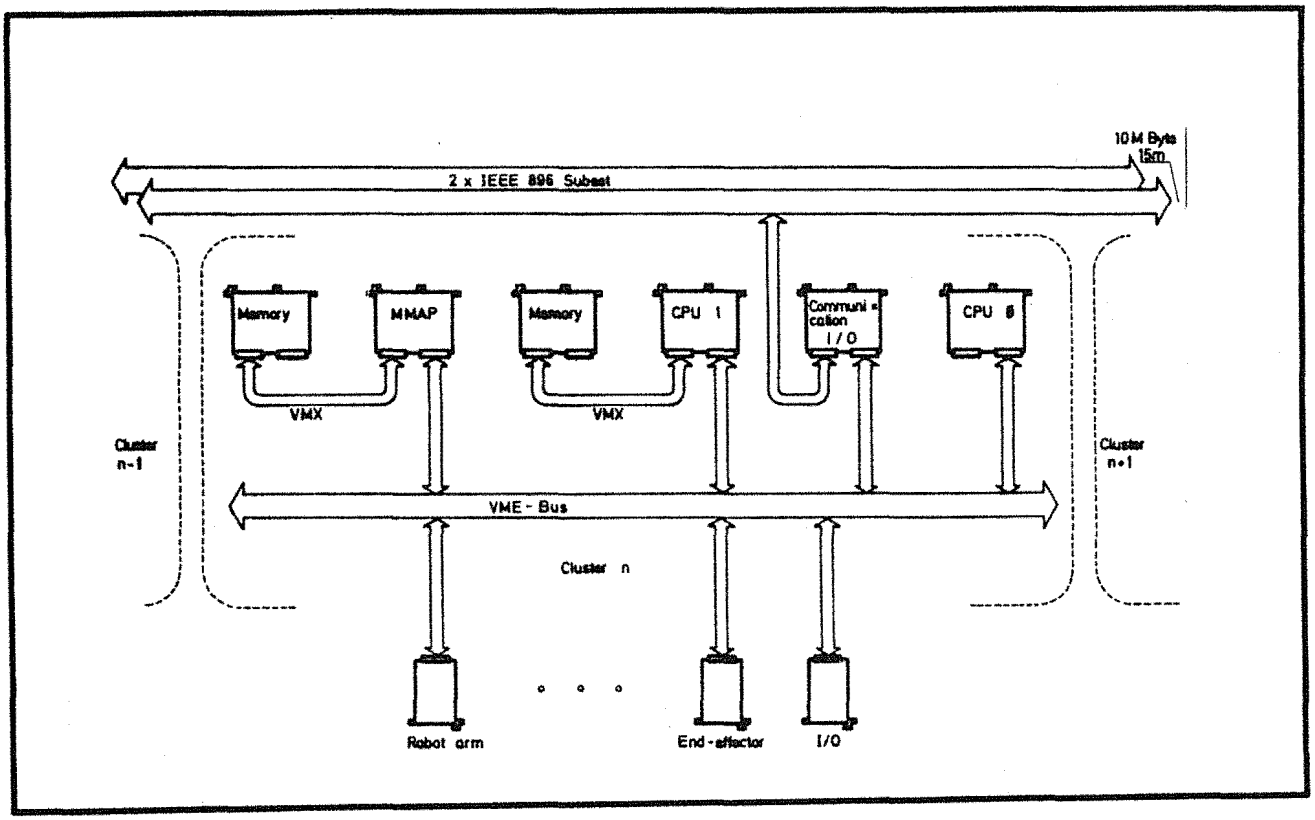

ABBILDUNG 1

- Einem Kommunikationsspeicher (MMAP) mit einer neuartigen Speicherverwaltung (MMAP Memory Map Access and Protection Unit)
- Einem Prozessor, der die Kommunikationskanäle installiert und Aktivitäten des globalen Betriebssystems durchführt.
- Mindestens einem weiteren Prozessor mit lokalem Speicher (Akteur), der die geforderte Rechenleistung erbringt. Auf allen Akteuren läuft ein lokales Betriebssystem, das um einige Systemprimitiven erweitert wurde.
- Peripherie-Anschlüssen, entsprechend den jeweiligen Knotenfunktionen, Parallele Ein- Ausgabe, A/D-Wandler, Hintergrundspeicher usw.

Die Forderungen an Kommunikationskanäle lassen sich in der folgenden Weise beschreiben:

- Es soll Speicher bereitgestellt werden, um die Kommunikation schnell, d.h. ohne Wartezyklen wie sie bei synchroner Kommunikation auftreten, in asynchronerweise abwickeln zu können.
- Es sollen unzulässige Zugriffe auf den Speicher erkannt und gemeldet werden.
- Der Kommunikationskanal soll eine Pufferfunktion ausüben, um die einzelnen Prozesse zu entkoppeln.
- Der Kommunikationskanal muß unter Beibehaltung der Applikationssicht auf die dezentralen Systeme verteilt werden können.
- Es muß die gesamte Synchronisation der beiden beteiligten Applikationsprozesse durch diesen Kommunikationsprozess überwacht werden. Ggfs. muß ein Applikationsprozess verzögert werden können bis ein Zustand eintritt, der den Zugriff ermöglicht. Die Überwachung der Synchronisation erfordert aber eine Definition der Zugriffsdisziplin, z.B. alternatives Lesen und Schreiben, für einen Kanal.

Für einen solchen Kanal lassen sich eine Reihe von Aussagen treffen:

- Dieser Kanalprozess ist der häufigste aller Prozesse.
- Es ist immer der gleiche Prozeß, nur mit unterschiedlichem Typ des Kommunikationsobjektes.
- Der Prozeß besitzt einen Zustand durch den letzten Zugriff auf das Kommunikationsobjekt.
- Er kann die Zugriffssequenz auf ein Kommunikationsobjekt überwachen und muß bei Verletzungen der Zugriffsdisziplin handeln.
- Er ist unabhängig von allen Applikationsprozessen.

3. KOMMUNIKATIONSSPEICHER UND VETO

3.1 Zugriffsdisziplinen

Der Kommunikationsspeicher ist ein gemeinsamer Speicher innerhalb eines Multiprozessorknotens. Der Zugriff auf einen Kanal dieses Speichers wird durch einen Deskriptor geschützt, dessen Inhalt Auskunft über den jeweiligen Zustand gibt. Abbildung 2 zeigt die Informationen, die der Deskriptor enthält:

B (Belegt) Der betreffende Deskriptor ist initialisiert und damit einem Kommunikationsobjekt zugeordnet.

A (Aktiv) Die aktuelle Adresse ist Null, d.h. ein Datenblock wurde vollständig übertragen oder ausgelesen.

Z (Zustand)

00 : unbenutzt, dies ist der Zustand nach der Initialisierung
01 : gelesen, der letzte Zugriff war eine Leseoperation
10 : beschrieben, der letzte Zugriff war eine Schreiboperation
11 : geschlossen, das Kommunikationsobjekt darf nicht mehr benutzt werden.

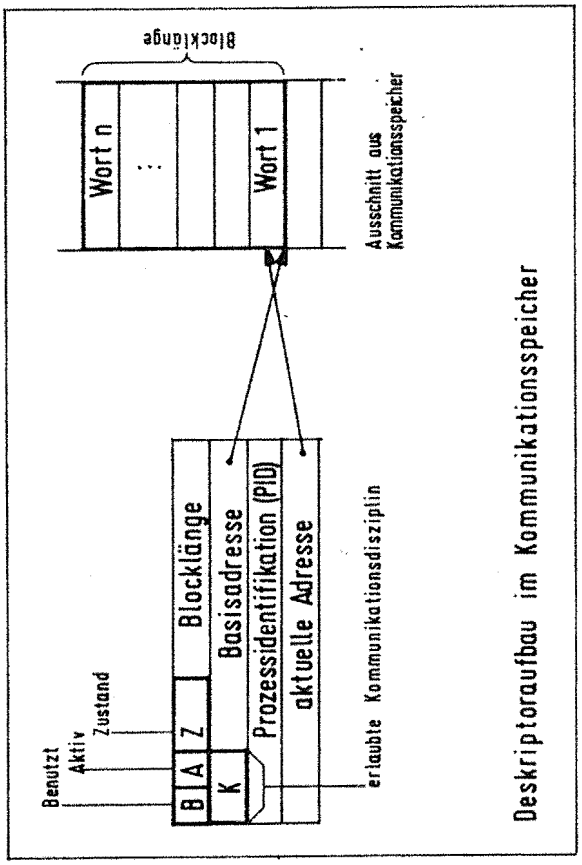

ABBILDUNG 2

K beschreibt die von den Prozessen definierte Kommunikationsdisziplin.

11 : (wr)*
01 : (w*r)*
10 : (wr*)*
00 : (w*r*)*

Basisadresse
verweist auf den Speicherbereich, der dem jeweiligen Kanal zugeordnet ist. Dieser Speicher ist von außen (vom Bus aus) nicht adressierbar.

aktuelle Adresse
verweist innerhalb des Speicherbereichs auf den Inhalt, der während eines Lese- oder Schreibvorganges als nächstes gelesen oder geschrieben wird.

PID ist die Prozeßidentifikation des Prozeßes, der aufgrund des Zustandes des Kanals unterbrochen wurde und bei einer Änderung des Zustandes erneut aktiviert wird.

Die Kopplung der beiden Prozesse ist festlegbar. Es gibt vier Abstufungen zwischen vollständig synchronisiert und vollständig unsynchronisiert. Festgelegt wird dies durch die K-bits im Deskriptor. Die vier Möglichkeiten lassen sich wie folgt beschreiben:

1. (wr)*
   Beide Prozesse sind vollständig synchronisiert. Diese Zugriffsart zwingt den Produzenten - Prozeß mit der Übertragung des Wertes zu warten, bis der Konsument den vorhergehenden Wert des Objekts vollständig gelesen hat. Genauso muß der Konsument nach dem Lesen eines Werts warten, bis der Produzenten - Prozeß wieder in einen neuen Wert übertragen hat.
2. (w*r)*
   Der Produzent darf bei dieser Zugriffsart einen neuen Wert in das Kommunikationsobjekt eintragen, auch wenn der vorher übertragene Wert noch nicht gelesen wurde.
3. (wr*)*
   Der Wert des Objekts darf vom Konsument beliebig häufig gelesen werden.
4. (w*r*)*
   Die Kopplung ist in dieser Zugriffsart am schwächsten. Die beteiligten Prozesse greifen völlig unsynchronisiert auf das Kommunikationsobjekt zu.

3.2 Der VETO Mechanismus

Veto erzwingt als Signal vom Speicher zum Busmaster das Blockieren das Prozesses. Wie aus Abbildung 2 ersichtlich, enthält der Deskriptor eine Beschreibung der Zugriffsrechte und der Zugriffssequenzen, die auf ein in diesem Speicherbereich abgelegtes Kommunikationsobjekt zulässig sind, sowie eine Identifikation des unterbrochenen Prozesses. Für jedes, zusammen mit dem Deskriptor gespeicherte Objekt, wird nun eine bestimmte Zugriffsfolge zugelassen. Die Speicherverwaltung (MMAP) überprüft während der Laufzeit die Speicherzugriffsfolge bei jedem einzelnen Zugriff. Die Folge wurde bei der Definition des Prozesses und des Kommunikationsobjektes festgelegt.

Bei einem Datentransfer zum Kommunikationsspeicher ist der am zentralen Bus aktive Prozessor der Busmaster, der die Kommunikation mit Hilfe seines lokalen Betriebssystems abwickelt. Nur er kann zu diesem Zeitpunkt Adressen anlegen und einen Datenzyklus durchführen. Alle Zugriffe auf Kommunikationskanäle werden über den Knotenbus und die MMAP abgewickelt.

Stimmt das Zugriffsrecht des Busmasters (read oder write) auf den Kanal nicht mit dessen Zustand und der vereinbarten Kommunikationsdisziplin überein, so darf der Datentransfer auf dem Bus nicht mit einer Bestätigung beantwortet und abgeschlossen werden. Vielmehr wird, initiiert vom Kommunikationsspeicher jetzt ein VETO-Signal über den Bus ausgegeben.

Dieses Veto-Signal ist ungerichtet, geht also an alle Prozessoren. Der Speicherverwaltung braucht nicht bekannt zu sein, an welchen Prozessor sich das Veto richtet, weil der jeweils aktive Busmaster das Veto automatisch auf sich bezieht und seinen gerade aktiven Prozeß blockiert.

Da die Datenleitungen zum Zeitpunkt des Vetos nicht benutzt wurden, ist der Datenbus frei. Auf den Bus wird jetzt vom aktuellen Busmaster die Prozeßidentifikation (PID) gelegt, die den am Bus aktiven Prozeß eindeutig kennzeichnet. Zeitlich etwas verzögert sendet dann der aktive Busmaster ein Veto-Acknowledge auf den Bus, das die Gültigkeit der PID-Daten anzeigt. Diese zeitliche Verzögerung beinhaltet den Übernahme-Strobe, mit dem die Prozeßidentifikation in den Deskriptorspeicher geladen wird. Damit ist dem Deskriptor des aktuellen Kanals, dessen Adresse immer noch am Bus an-

liegt, die Prozeßidentifikation des blockierten Prozesses zugeordnet, und der Datenzyklus kann abgeschlossen werden. Dieser Zyklus kann weder durch Interrupts noch durch direkte Speicherzugriffe unterbrochen werden.

Infolge anderer Prozesse oder Kommunikationsvorgänge wird das Zugriffsrecht auf den Kanal zu einem späteren Zeitpunkt verändert: entweder ist in den Kommunikationskanal inzwischen ein gültiger Wert eingetragen worden oder der Kanal ist bereit, einen neuen Wert aufzunehmen.

Dabei wird geprüft, ob der zugeordnete Deskriptor in seinem PID - Teil einen Prozeß bezeichnet (PID$\neq$0), der auf den Kommunikationskanal wartet.

Da die Identität des unterbrochenen Prozesses durch die im Deskriptor stehende Prozeßidentifikation bekannt ist, kann dem Betriebssystem des Knotens per Interrupt-Serviceroutine dieser PID mitgeteilt werden. Dieses überführt daraufhin den wartenden Prozeß in den "ready-to-run" -Zustand. Der Prozeß wird fortgesetzt, sobald der Prozessor bereit ist, auf dem er seine Aktivität begonnen hatte.

Aus den Statusinformationen des Deskriptors lassen sich folgende Fälle ableiten, die den normalen Datentransfer unterbrechen:

1. Der Deskriptor ist aktuell keinem Kommunikationskanal zugeordnet. In diesem Fall wird ein Interrupt ausgelöst, der dem Knotenbetriebssystem einen System- oder Programmfehler anzeigt.
2. Der Deskriptor zeigt den Zustand "geschlossen", was eine Prozeßtermination, zumindest aber die Auslösung eines "Exceptions", bewirkt. Es wird der Prozeß angesprochen, der den Zugriff versucht hat.
3. Der Deskriptor ist dem entsprechenden Kanal zugeordnet, die Kommunikationsdisziplin entspricht aber nicht dem aktuellen Zugriff. In diesem Fall wird das oben beschriebene VETO ausgelöst.

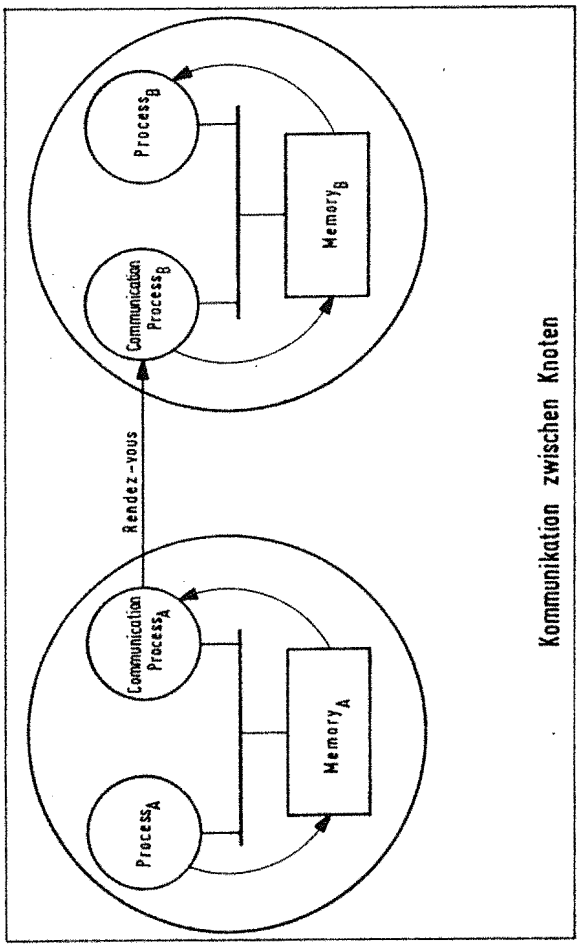

ABBILDUNG 3

Der bisherige Aufbau setzt noch einelementige Kanäle voraus. Um auch Kommunikationen mit Kanalkapazitäten von k=o oder k>1 zu unterstützen, muß das Konzept erweitert werden.

Zur Emulation synchroner Kommunikation mit diesem Mechanismus muß der sendende Prozeß sich blockieren bis die Nachricht abgeholt wurde. Diese Betriebsart kann durch ein Statusbit im Deskriptor gefordert werden. Es wird dann durch den Veto-Mechanismus der PID eingelesen und der sendende Prozeß blockiert. Nach Verbrauch des Objekts durch den empfangenden Prozeß wird der blockierte Prozeß wieder weitergeführt. Es wird also der Empfang der Daten durch den empfangenden Prozeß implizit dadurch bestätigt, daß der sendende Prozeß fortgesetzt wird. Erreicht der empfangende Prozeß als erster den Synchronisationspunkt, so wird er wie bei der asynchronen Kommunikation blockiert. Zeitüberwachungen der Verweildauer eines Wertes im Kommunikationsspeicher kann auf Störungen im Prozeßgeschehen hinweisen.

Zur Realisierung von Kanälen mit Kapazitäten größer 1 müssen im Kommunikationsspeicher zusätzliche Zeiger auf den Speicher verwaltet werden. Die Stellung dieser Zeiger reguliert die Zugriffsdisziplin. Prinzipiell sollte diese Verwaltung natürlich mit Software realisiert werden. Dies setzt allerdings einen zusätzlichen Prozessor mit der alleinigen Aufgabe der Speicherverwaltung voraus und dürfte den bisherigen Vorteil kürzerer Kommunikationszeiten stark beeinträchtigen.

Abbildung 3 zeigt die Kommunikation über Knotengrenzen hinweg. Ein bisher nicht gelöstes Problem ist die Kopplung von Prozessen, die auf unterschiedlichen Knoten des Systems ausgeführt werden. In unseren Überlegungen wird die Verteilung auch durch die Verteilung des Kommunikationskanals unterstützt. Dies führt zum Aufbau wie in Abb. 3, wobei zur Kommunikation die Speicher in beiden Knoten beteiligt werden.

4. ZUSAMMENFASSUNG UND AUSBLICK

Das vorgeschlagene Konzept der hierarchischen Funktionsverteilung des Betriebssystems und der impliziten Synchronisation von Applikationsprozessen weist, gegenüber den auf dem Markt befindlichen Produkten, einige Verbesserungen und Erleichterungen im Einsatz auf:

- es erhöht durch die implizite Synchronisation und das asynchrone Kommunikationsverhalten den Grad der Parallelität und läßt eine geringere Anzahl der Prozeßwechsel erwarten.
- es reduziert den Aufwand zur Beschreibung von Kommunikation in den Applikationsprozessen erheblich, wobei gleichzeitig die Belastung des Systems gesenkt wird.
- es ermöglicht eine hierarchische, auch heterogene Erweiterung, ohne Veränderung der Applikationsprozesse.

- es verbindet die Vorteile von fester Kopplung und gemeinsamen Speicherbereichen (mit der damit verbundenen schnelleren Kommunikation) mit den Vorteilen der losen Kopplung, dynamischer Rekonfigurierbarkeit, Prozeßverteilung und schnellerer Reaktion auf Eingaben.

Als Nachteile sind derzeit die nicht standardisierte Speicherverwaltung einerseits und die hinzukommenden Vetoleitungen andererseits zu nennen.

Ein oben beschriebener Aufbau wurde im September 85 fertiggestellt und erstmalig getestet. Dabei erwiesen sich die gemachten Angaben als richtig:

- die Kommunikation zwischen den Prozessen ist äußerst effizient. Sie ist nicht vom Kommunikationsprotokoll, sondern lediglich von der Zugriffszeit auf den verwendeten Bus und der Zykluszeit des Speichers abhängig. Die Busbelastung durch ein **VETO** liegt im Fehlerfall bei 700 ns.
- die Kommunikationsdisziplin führt zur Synchronisation der beteiligten Prozesse. Der Veto Mechanismus und das erneute Starten eines Prozesses nach einer Änderung des Status eines Kanals führen zur ordnungsgemäßen Synchronisation.

Als Erweiterung des vorgestellten Konzepts und der derzeitigen Realisierung ist eine Zeitüberwachung für die einzelnen Objekte geplant. Diese ermöglicht dann eine schnellere Transaktionsüberwachung. Der bereits oben beschriebene Ansatz stellt ein Prozeßsystem als eine Baumstruktur dar. Die Kontrolle dieser Struktur erfolgt durch das (globale) Betriebssystem, dessen Komponenten auf Knoten verteilt werden. Aufgaben dieses Betriebssystems in Bezug auf die Verwaltung der beteiligten Prozesse sind u.a.:

- Basisdienste bereitzustellen; z.B. Compiler und Debugger,
- die Bereitstellung der Kommunikationsprimitiven, eine Firmware integrierte Zwischensprache,
- die initiale Konfigurierung des Prozeßsystems inklusive des Ladens vom Entwicklungssystem,
- die Rekonfiguration des Systems aufgrund äußerer Einflüsse (z.B. einer Änderung des zu regelnden Systems). Es soll möglich sein, zu jeder Zeit den Zustand des Systems zu erfragen und zu ändern, also das Kommunikationsnetz abzuändern,
- die Rekonfiguration des Systems aufgrund ungünstiger Lastverteilungen. Hierzu gibt es keine bekannten Verfahren.

Die Zuordnung von Prozessen und Prozessoren unterliegt u.a. folgenden Einflüssen:

- den vom Prozeß benötigten Resourcen,
- den von den Prozessoren bereitgestellten Resourcen,
- dem Grad der Kopplung zwischen Prozessen,
- der Auslastung von Prozessoren, Bus und Peripherie,
- der topologischen Anordnung technischer Prozesse und den sie steuernden (regelnden) Prozessen,
- den Anforderungen des technischen Prozesses (z.B. an die Regelzeit).

Unsere Vorstellung der Zuordnungsstrategie ähnelt der Traversierung eines Baumes, wobei eine feste Strategie zugrunde liegt, die aber von jedem Cluster beeinflußt werden kann. Dieser Konfigurationsprozeß beginnt initial an der Wurzel des Systembaumes, bei jedem Start eines Prozesses im entsprechenden Cluster.

Anmerkungen:

Das Hardwarekonzept entstand durch intensive Diskussionen mit Dr.S.Jähnichen, GMD Forschungsstelle Karlsruhe und durch viele Gespräche mit Kollegen zum Entwurf einer Rechnerarchitektur für intelligente, sensorgeführte Robotersysteme. Die Arbeiten zur Realisierung dieser Kommunikationskonzepte wurden im Institut für Informatik III, Lehrstuhl Prof. U. Rembold durchgeführt und durch die DFG unter der Kennziffer Re 489/2 gefördert.

Literaturverzeichnis

1 Behr,P.'Entwurf eines verteilten Multicomputer-Systems, TU Berlin Dissertation 1983

2 Bowen,B.A., 'The logical design of Multiple- Microcomputer Systems', Prentice-Hall Inc. 1980

3 Böhler,E. 'Entwurf und Aufbau einer Speicherverwaltungseinheit' Diplomarbeit Uni.Karlsruhe 85

4 Brinch Hansen,P.'Operating System Principles', Prentice Hall,82

5 Dijkstra, E.W.D., 'Cooperating Sequentiell Processes', Programming Languages, Academic Press 68

6 Färber, 'Bussysteme',Oldenburg 84

7 Giloi,W. 'Rechnerarchitektur'Springer-Verlag Berlin 1981

8 Hoare, C.A.R., 'Monitors: An Operating System Structuring Concept', CACM, Vol.17, #10,74

9 Iliffe,J.K. 'Basic Machine Principles' American Elsevier 1972

10 Jähnichen ,S., 'Kommunikation und Synchronisation in verteilten Multiprozessorsystemen' Interner Bericht , GMD Forschungsstelle Karlsruhe, 1984

11 Jähnichen, S., Kordecki, C.,'Communication and Synchronization in Distributed Systems', to be published

12 Spaniol,O. Konzepte und Bewertungsmethoden für lokale Rechnernetze, Informatik-Spektrum 5(1982)

Anschrift des Autors

C.Kordecki
Institut für Informatik III
Prof.Rembold
Universität Karlsruhe
Postfach 6380
7500 Karlsruhe