# Interface

herausgegeben von der
Abt. Hybridrechenanlage (Simulationsrechenanlagen)
des EDV-Zentrums der
Technischen Universität Wien

Nummer 25
März 1990


Simulation eines Fußgängerunfalls

# Inhaltsverzeichnis




Redaktion: Irmgard Husinsky

Abt. Hybridrechenanlage (Simulationsrechenanlagen) des EDV-Zentrums der Technischen Universität Wien, A-1040 Wien, Wiedner Hauptstraße 8-10.

Herausgeber, Verleger, Hersteller: EDV-Zentrum der Technischen Universität Wien, Abteilung Hybridrechenanlage (Simulationsrechenanlagen), Leitung: Dipl. Ing. Dr. W. Kleinert, A-1040 Wien, Wiedner Hauptstraße 8-10. Tel.: (0222) 58801 5481. Tlx: (232) 3222467=TUW, Fax: (0222) 5871020. Druck: Hochschülerschaft Technik, A-1040 Wien, Wiedner Hauptstraße 8-10.

*Liebe Leser!*

*Das vorliegende INTERFACE setzt die Tradition unserer anwenderorientierten Zeitschrift seit dem Jahre 1974 fort, ist eine Jubiläumsnummer und zugleich ein Abschluß.*

*Um den im Laufe der letzten Jahre geänderten Anforderungen zu entsprechen, die an ein zentrales Rechenzentrum einer Technischen Universität gestellt werden, wird zur Zeit an einer umfassenden Reorganisation des EDV-Zentrums gearbeitet. Die seit der Gründung des EDV-Zentrums bestehenden drei Abteilungen (Digitalrechenanlage, Prozeßrechenanlage, Hybridrechenanlage) werden aufgelöst und es wird ein einheitliches Rechenzentrum entstehen, das den geänderten Anforderungen der Benützer sowie den neuen Entwicklungen der Technologie Rechnung tragen wird.*

*Unsere Abteilung hat jahrelang den Namen Hybridrechenanlage getragen. An unserer Abteilung wurden bahnbrechende Entwicklungen für die Automatisierung der Hybridrechentechnik durchgeführt. Heute steht allerdings der Betrieb von Hybridrechnern nicht mehr im Mittelpunkt unserer Aktivitäten. Wir geben in diesem INTERFACE einen kurzen geschichtlichen Überblick über die Einsatzzeit des EAI PACER 600A AutoPATCH Hybridsystems. Ein Artikel beschreibt eine Anwendung aus der Medizin, die anfangs am PACER, später dann am SIMSTAR durchgeführt wurde.*

*Aus dem umfangreichen Gebiet der Simulation, das wir in den letzten Jahren betreut haben, sind interessante Simulationen aus der Mechanik und der Physik beschrieben. Eine große Simulationstagung wird im September an der Technischen Universität Wien stattfinden.*

*Den neuen Technologien Rechnung tragen die Beiträge über das superskalare RISC-Prozessor-System und das Multi-Transputer System. Im Mai veranstalten wir ein Workshop für Transputer-Systemspezialisten. Den Schwerpunkt, der in nächster Zeit auf das Parallelrechnen gelegt werden soll, beschreiben die Berichte über die Ziele des Austrian Center for Parallel Computation und über das Programmierwerkzeug Linda.*

*Wie anfangs gesagt, bildet dieses INTERFACE einen Abschluß in der bisherigen Tradition dieser Zeitschrift. In Zukunft werden neue Publikationsorgane die Aktivitäten eines einheitlichen, reorganisierten EDV-Zentrums der Technischen Universität an die Öffentlichkeit bringen.*

*Irmgard Husinsky*
*Redaktion*

# BIOMEC - Eine ACSL Anwendung in der Unfallbiomechanik

Dipl.-Ing. H. Ecker
Dipl.-Ing. W. Hanreich
Institut für Maschinendynamik und Meßtechnik
Technische Universität Wien

Dipl.-Ing. H. Schrefl
Voith AG, St Pölten

## 1. Einleitung

Während der Begriff der Biomechanik zwar ein umfassendes, aber doch relativ klar abzugrenzendes Forschungsgebiet umschreibt, wird das Teilgebiet "Unfallbiomechanik" in Zukunft noch genauer zu definieren sein. Dieser Beitrag befaßt sich mit den Bewegungen und Belastungen des Menschen und seiner Körperteile in Unfallsituationen, wobei hier speziell Verkehrsunfälle diskutiert werden.

Wie in vielen anderen Wissenschaftsdisziplinen kommt auch in der Biomechanik verstärkt die Simulationstechnik zum Einsatz. Im Bereich der Unfallbiomechanik besteht seit der Verfügbarkeit leistungsfähiger Rechner die Möglichkeit, aufwendige und teure Versuche durch Simulationsrechnungen mittels Computer zu ersetzen oder Versuche zu simulieren, die praktisch gar nicht ausführbar sind. Dies ist besonders in der Fahrzeugtechnik von Bedeutung und deshalb stammen die meisten Anwendungen auch aus diesem Bereich. Die Automobil-Hersteller versuchen, die kostspieligen Crash-Versuche mit Dummies zu reduzieren und die Fahrzeugdeformationen, die Innenraumgestaltung oder Rückhaltesysteme rechnerisch zu optimieren. Aber auch Fußgängerunfälle und Kollisionen PKW-Motorrad wurden bereits rechnerisch simuliert, nicht zuletzt weil in diesen Fällen die Versuchstechnik erhebliche Probleme aufwirft.

Es existieren daher einige, größtenteils sehr umfangreiche und rechenzeitintensive Simulationsprogramme [1,2]. Auf der Basis allgemein verwendeter Simulationssprachen ist den Autoren aber keine Anwendung bekannt. Deshalb wurde der Versuch unternommen, mit ACSL und unter Ausnützung der damit verbundenen Vorteile gegenüber einer klassischen Programmiersprache ein Programm für unfallbiomechanische Simulationen zu entwickeln.

## 2. Mechanische Modelle des menschlichen Körpers

Aufgrund der unterschiedlichen Aufgabenstellungen bei Unfallsimulationen wurden die verschiedensten mechanischen Ersatzmodelle vorgeschlagen und verwendet. Gemeinsames Charakteristikum fast aller Modelle ist aber der Versuch, den Körper durch starre, gelenkig miteinander verbundene Stangen darzustellen. Aus wievielen Elementen und Gelenken ein solches "Gelenkstangenmodell" besteht, hängt von der angestrebten Annäherung an die Wirklichkeit ab.

Ein wesentlicher Unterschied besteht darin, ob das Modell nur ebene oder auch räumliche Bewegungen ausführen kann. Setzt man baumartige Strukturen und Kugelgelenke voraus, so ergeben sich bei n gelenkig verbundenen Teilkörpern $f_{2D} = n+2$ Freiheitsgrade im Falle der ebenen (2D) Modellierung, jedoch $f_{3D} = 3n+3$ Freiheitsgrade für räumliche (3D) Modelle. Jeder Freiheitsgrad bedeutet

hier eine nichtlineare Differentialgleichung 2.Ordnung und man sieht, daß der Aufwand für eine dreidimensionale Modellbildung überproportional anwächst. Von den 3D-Modellen ist bekannt, daß sie selbst auf sehr leistungsfähigen Rechenanlagen hohe Rechenzeiten erfordern.

Da es im Rahmen dieses Projektes das Ziel war, mit einem Rechner von der Leistungsfähigkeit einer Workstation auszukommen und ACSL außerdem zwar verschiedene Integrationsalgorithmen zur Verfügung stellt, aber sonst wenig Rücksicht auf speziell strukturierte Probleme nimmt, wurde auf eine 3D-Modellierung von vorneherein verzichtet.

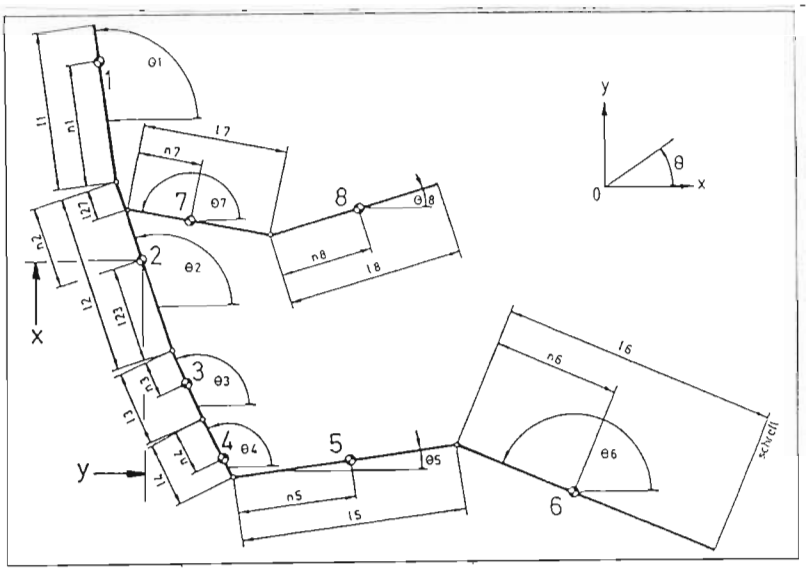



Abb.1: Gelenkstangenmodell für den Menschen

Das verwendete ebene, mechanische Ersatzmodell ist in Abb.1 dargestellt [3]. Es besteht aus dem Kopf (1), dem oberen, mittleren und unteren Torso (2-4), dem Ober- und Unterschenkel (5-6) sowie dem Ober- und Unterarm (7-8). Arme und Beine sind zufolge der ebenen Modellbildung nur einfach vorhanden. Alle Körperelemente sind durch Scharniergelenke miteinander verbunden. In jedem Gelenk sind nichtlineare Drehfedern und Dämpfer angebracht, die den Bewegungsbereich der Gelenke begrenzen und so die Gelenkhalte darstellen, siehe Abb.2. Es können aber auch zusätzliche zeitabhängige Momente in den Gelenken aufgebracht werden, um willentliche Muskelanspannungen zu simulieren. Darin besteht einer der wesentlichen Vorteile der Simulation gegenüber dem Versuch.

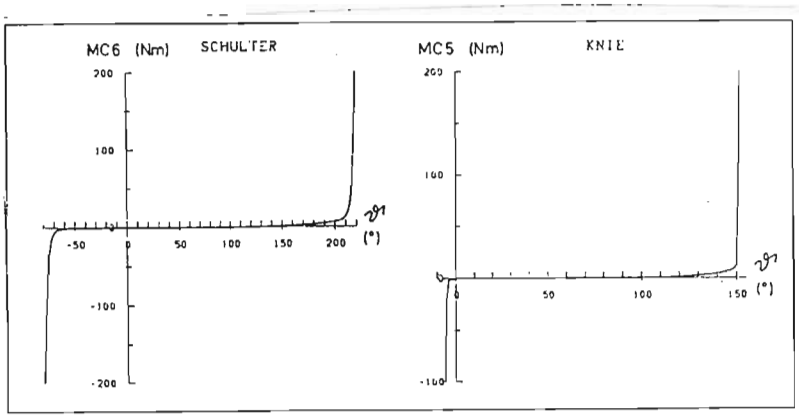



Abb.2: Gelenkskennlinie für Schulter und Knie

Die nichtlinearen Bewegungsgleichungen dieses Modells haben die allgemeine Form

$$\underline{M}(x)\ddot{x} + K(x,\dot{x}) = Q(x,\dot{x},t) \qquad (1)$$

mit dem Lagevektor x und seinen zeitlichen Ableitungen, der lageabhängigen Massenmatrix $\underline{M}$, dem Vektor der verallgemeinerten Kreiselkräfte K und dem Vektor der verallgemeinerten eingeprägten Kräfte Q. Sie sind hochgradig nichtlinear, weil große Bewegungen zugelassen werden müssen und daher die zahlreichen Ausdrücke mit Winkelfunktionen nicht linearisiert werden können. Für das Aufstellen der Gleichungen in analytischer Form wurde (neben der Handrechnung) das Programmpaket NEWEUL verwendet.

Diese Gleichungen gelten (im Rahmen der Modellbildung) gleichermaßen für einen Menschen wie für einen Dummy. Während die Abmessungen und Massen bzw. Trägheitsmomente der Dummy-Körperteile weitgehend mit denen des "Durchschnitts"-Menschen übereinstimmen, bestehen in den nichtlinearen Gelenkskennlinien häufig wesentliche Unterschiede. Dies ist auf die Schwierigkeiten bei der Herstellung von menschenähnlichen Gelenken in Dummies zurückzuführen. Simulationsstudien können deshalb Versuche wesentlich ergänzen.

## 3. Berechnung von Kontaktkräften

Die Interaktion zwischen dem Modell und einer Umgebung ist der wesentlichste, aber auch diffizilste Bestandteil einer Unfallsimulation. Dazu wird das Gelenkstangenmodell mit einem Umrißmodell versehen und ein Kontaktpartner (Fahrzeuginnenraum, PKW-Front,...) definiert. Sehr häufig werden den Körperteilen des Modells Ellipsen umschrieben. Diese Art der Umrißgenerierung hat sich als effektiv erwiesen und wurde auch hier angewendet. Der geometrische Umriß des Kontaktpartners wird in diesem Modell durch eine entsprechende Anzahl von Polygonzügen approximiert, siehe Abb.3.

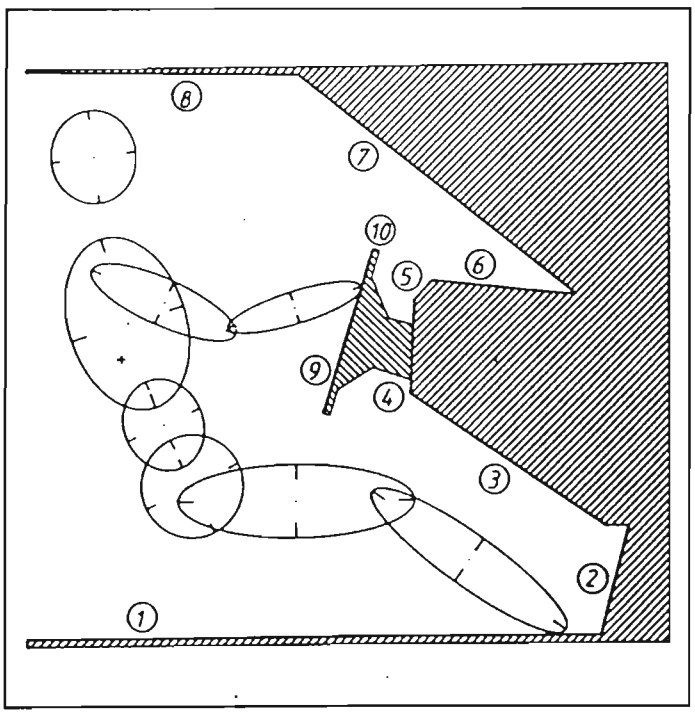



Abb.3: Umrißmodell und Fahrzeuginnenkontur

Treten im Zuge der simulierten Bewegung des Modells Durchdringungen von Umrißlinien auf, so müssen die Kontaktkräfte berechnet werden, welche auf das Modell einwirken.

Von Danforth und Randall wurde ein sehr allgemeines Verfahren für die Berechnung des Kraft-Deformationsverlaufes entwickelt, mit dem auch ein sehr komplexes nichtlineares Materialverhalten simuliert werden kann [4]. Die Möglichkeiten reichen vom einfachen linear-elastischen Verhalten über die Einführung einer Streckgrenze bis hin zu unterschiedlichen Wiederbelastungsverläufen nach einer plastischen Verformung. Zwar zeigen die in der Literatur dokumentierten Versuche an Menschen und bei extremen Belastungen an Leichen und Tierkadavern, daß der Zusammenhang zwischen Kraft und Deformationsweg bzw. -geschwindigkeit sehr komplex ist, aber mangels ausreichender Versuchsdaten (man denke nur an die zahlreichen unterschiedlichen Körperbereiche) wird das Verfahren nach Danforth und Randall in seiner allgemeinen Form praktisch kaum angewendet.

In dem hier erstellten Modell werden die Kontaktkräfte nach nichtlinearen, quasistatischen Be- und Entlastungskennlinien berechnet, denen ein geschwindigkeitsproportionaler Anteil überlagert wird, siehe Abb.4. Zusätzlich werden Reibungskräfte in Abhängigkeit von der relativen Gleitgeschwindigkeit berechnet. Für die Kontaktbereiche am Fahrzeug müssen selbstverständlich ebenfalls Kraft-Deformationskennlinien berücksichtigt werden. Durch die Hintereinanderschaltung beider Verformungselemente entsteht die Berechnungsvorschrift für eine Kontaktzone.

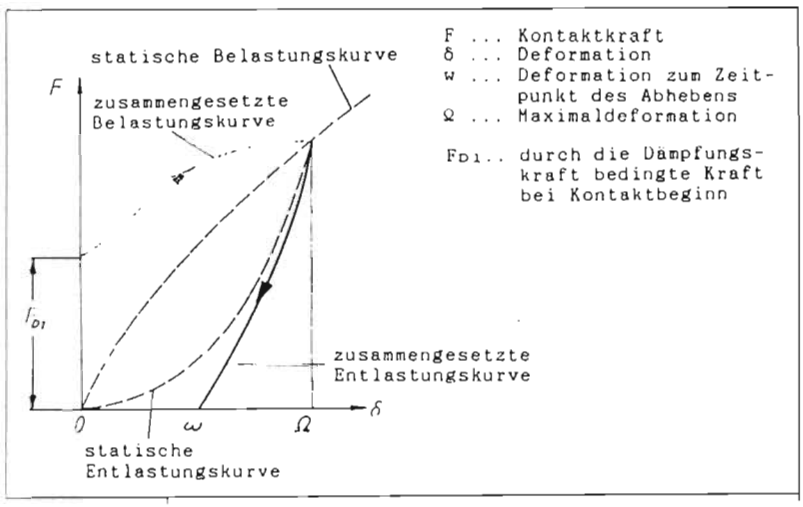



Abb.4: Entstehung der Be- und Entlastungskennlinie für Normalkräfte

Dabei geht man aus von der sog. "Eindringtiefe und -geschwindigkeit", mit der das nichtlineare Kraftgesetz ausgewertet wird. Während die Berechnung letzterer zumindest als Mittelwert noch einigermaßen problemlos möglich ist, kann eine "maßgebliche Eindringtiefe" nur in geometrisch einfachen Fällen eindeutig angegeben werden. Auch die Verwendung einer resultierenden Kontaktkraft statt ortsabhängiger Druck- und Schubspannungsverteilungen zwingt zu Hypothesen bezüglich des Angriffspunktes und der Kraftrichtung. In [5] wird darauf ausführlich eingegangen. Eine Abkehr von diesen Vereinfachungen ist nur um den Preis von drastisch erhöhter Rechenzeit denkbar und würde bedeuten, daß man ein Kontaktmodell z.B. nach der Methode der Finiten Elemente erstellt. Ob dann allerdings ACSL noch die geeignete Programmierumgebung darstellt, wäre noch zu diskutieren.

## 4. Implementierung in ACSL

Wie schon erwähnt, wurde das vorgestellte Modell in der Simulationssprache ACSL formuliert, weil diese an verschiedenen Rechenanlagen der TU-Wien installiert ist. Die Möglichkeit, auch FORTRAN-Unterprogramme einbinden zu können, und die interaktiven Graphik-Möglichkeiten waren weitere Argumente.

Bei den Bewegungsgleichungen (Gl.1) fällt als Besonderheit auf, daß diese nicht explizit nach den höchsten Ableitungen aufgelöst werden können, wie dies für ACSL innerhalb der DERIVATIVE-Section erforderlich ist. Zusätzlich sind die Koeffizienten der Massenmatrix Funktionen des Lagevektors, sodaß bei jedem Aufruf zuerst die aktuelle Massenmatrix berechnet werden muß. Danach wird die rechte Seite des Differentialgleichungssystems aus K und Q berechnet und es ergibt sich damit ein lineares Gleichungssystem in den unbekannten Beschleunigungen, das mit einer IMSL-Routine gelöst wird. Dann erst kann der INTVC-Befehl verwendet werden, mit dem der Beschleunigungsvektor integriert wird.

Die Berechnung der Kontaktkräfte wurde durch eine Reihe von PROCEDURAL-Blöcken strukturiert. Dies beginnt bei der Abfrage, ob zwischen zwei Umrißelementen Kontakt vorhanden ist oder nicht. In einer LOGICAL-Matrix, in der die Relationen aller Umrißelemente untereinander eingetragen sind, können mögliche und unmögliche Kontakte definiert und so die Anzahl der geometrischen Abfragen wesentlich verringert werden. Die Benützung des SCHEDULE-Befehls zur genauen Iteration von Kontaktbeginn bzw. -ende wurde zwar versucht, hat sich aber als nicht durchführbar erwiesen. Im vorliegenden Fall wären etwa 640 verschiedene, mögliche Nulldurchgänge durch entsprechende SCHEDULE-Anweisungen zu verwalten, eine Anzahl, mit der ACSL hoffnungslos überfordert ist. Deshalb wurde auf die Kontaktiteration verzichtet und durch einen Integrationsalgorithmus niedriger Ordnung bei entsprechend kleiner Schrittweite die gewünschte und im Rahmen der Modellbildung sinnvolle Genauigkeit erzielt.

Als kleine Besonderheit ist noch zu erwähnen, daß die TERMINAL-Section dazu benützt wird, unmittelbar nach der Integration den Bewegungsverlauf des Modells graphisch auszugeben. Dies wird durch die Verwendung des ACSL-internen ZZDRAW-Befehls in zusätzlichen FORTRAN-Routinen erreicht, die in der TERMINAL-Section aufgerufen werden*. Dadurch sind den graphischen Ausgabemöglichkeiten praktisch keine Grenzen mehr gesetzt, wie die folgenden, innerhalb von ACSL geplotteten Beispiele zeigen.

## 5. Simulationsergebnisse

Abb.5 zeigt die Simulation eines nicht angegurteten PKW-Insassen bei einem Frontalaufprall. Die Fahrzeuginnenkontur ist dabei in Ruhe, der Insasse besitzt eine horizontale Anfangsgeschwindigkeit von 36km/h. Nicht dargestellt sind die Konturen der Sitzfläche. Die Phasen der Bewegung sind in Abständen von 0.01 Sekunden dargestellt. Das Insassenmodell gleitet zunächst im Sitz nach vor, um mit dem Knie in den unteren Bereich des Armaturenbretts einzudringen. Der Kontakt des Armes mit dem Lenkrad wurde deaktiviert, weil die Praxis zeigt, daß in einer solchen Unfallsituation die Hände seitlich vom Lenkrad abrutschen. Mit dem Aufprall auf die Windschutzscheibe endet die Vorwärtsbewegung

* Für die tatkräftige Unterstützung sei Herrn Dipl.Ing. Gräff vom Hybridrechenzentrum der TU nochmals herzlich gedankt.

des Kopfes, während die Beine und das Becken bereits reflektiert wurden. Natürlich können auch die zeitlichen Verläufe sämtlicher Kontaktkräfte ausgegeben werden und mit den entsprechenden Bewertungskriterien [6] hinsichtlich der Verletzungsschwere ausgewertet werden.

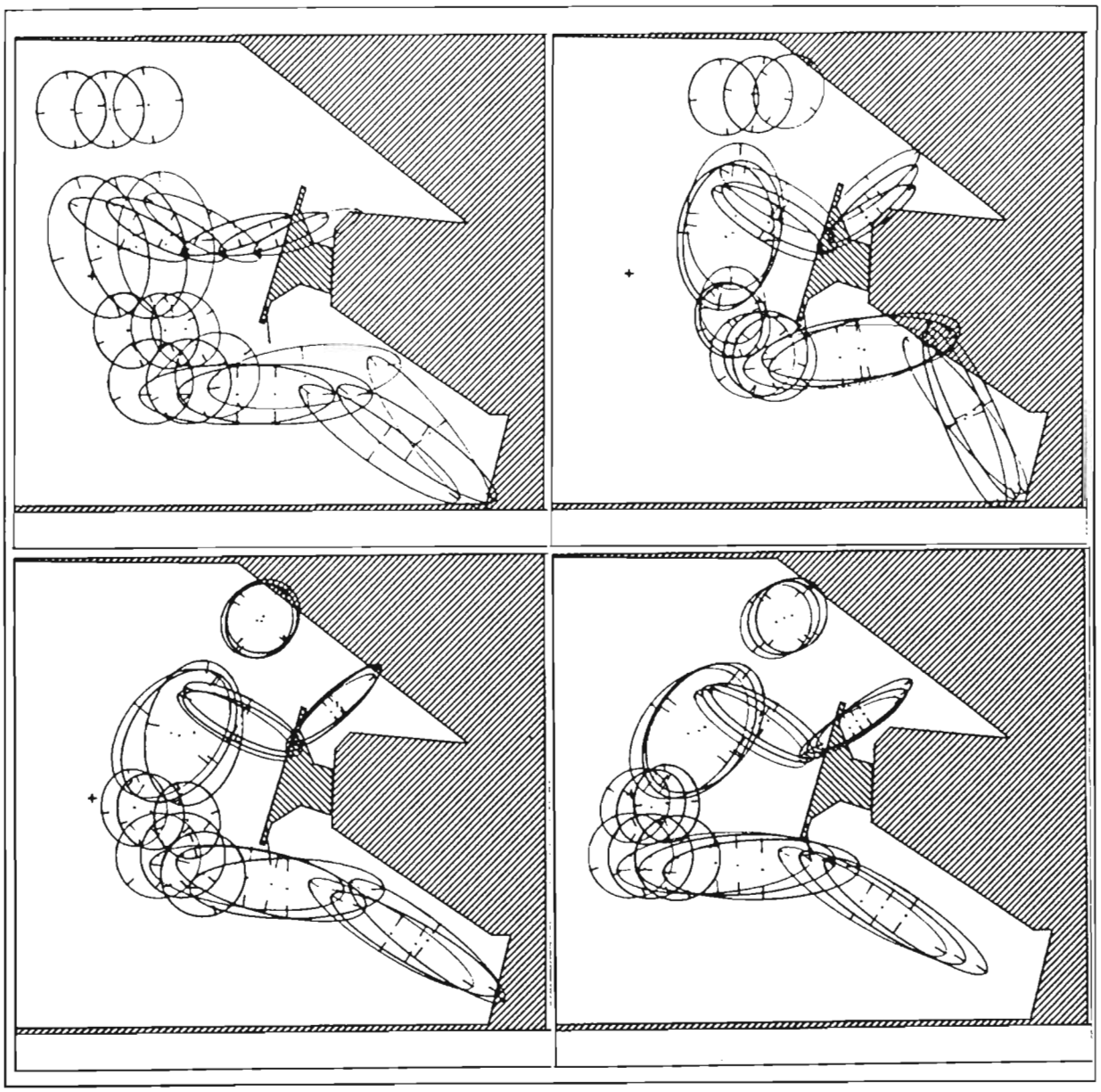


Abb.5: Simulation der Bewegung eines nicht angegurteten Insassen bei einem Frontalcrash mit 36 km/h. Bewegungsphasen in Intervallen von 0.01 s.

Abb.6 zeigt einen Fußgängerunfall. Dabei wird eine stehende Person von einer PKW-Front, die sich unverzögert mit 36 km/h bewegt, gerammt. Der Primäranstoß erfolgt im Kniebereich und leitet eine Rotation des Fußgängers ein. Es folgen der Aufprall des Körpers auf dem Motorraumdeckel und ein besonders gefährlicher Kopfaufschlag im Bereich unterhalb der Windschutzscheibe. Da die nachfolgende Bewegung in der Realität meist in eine räumliche übergeht, wurde hier die Simulation abgebrochen, weil die Grenzen einer ebenen Modellbildung erreicht sind.

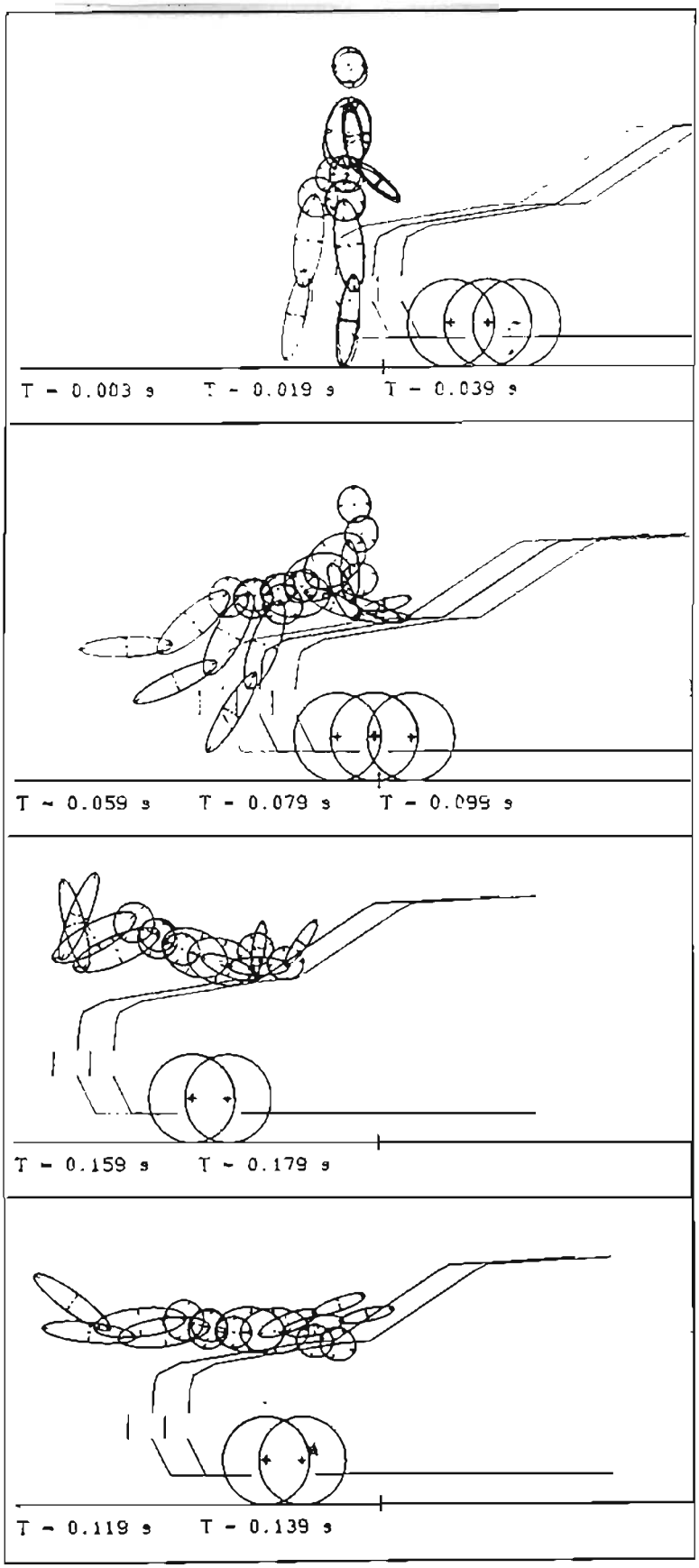



Abb.6: Simulation eines Fußgängerunfalls bei einer Fahrzeuggeschwindigkeit von 36 km/h. Bewegungsphasen in Intervallen von 0.02 s.

## 6. Zusammenfassung

Die gewonnenen Erfahrungen haben gezeigt, daß sich ACSL im Rahmen des hier vorgestellten Modells sehr gut als Programmierumgebung für unfallbiomechanische Simulationen eignet. Der Programmieraufwand konnte, verglichen mit einem FORTRAN-Programm ähnlichen Leistungsumfangs, wesentlich reduziert werden. In der derzeitigen Ausbaustufe hat das Programm BIOMEC eine Länge von ca. 2400 Zeilen incl. aller Datensätze und Kommentare.

Die Probleme bei Unfallsimulationen bestehen vor allem in der Kontaktkraftberechnung. Einerseits erfordern die Algorithmen weitere Verbesserungen, andererseits herrscht bei den Materialdaten relativ große Unsicherheit. Während diese die "großen Bewegungen" (z.B. im Fall des gezeigten Fußgängerunfalls) nicht wesentlich beeinflußt, soferne nur die integralen Werte der Kontaktkräfte realistisch sind, betrifft diese Problematik in erster Linie die berechneten örtlichen Belastungsspitzen. Ohne eine Modellvalidierung durch entsprechende Versuche müssen bei diesbezüglichen Ergebnissen größere Ungenauigkeiten erwartet werden.

## Literatur


[1] Schmid, W.: The Daimler-Benz Mathematical Isohumane Simulation Model, SAE-Paper 840856.

[2] Wismans J., Griffioen J.A., Nieboer J.J.: Use of MADYMO in general impact biomechanics, ASME-Symposium on Computational Methods in Bioengineering, Chicago 1988.

[3] Schrefl H.: Simulation von Fahrzeuginsassenbewegungen, Diplomarbeit TU-Wien 1987.

[4] Danforth J.P., Randall D.: Treatment of Single and Multiple Loading and Unloading of Contact Surfaces in Vehicle Interiors, General Motors Publ.Nr.1254, 1972.

[5] Hanreich W.: Rechnerische Simulation der Bewegung und Belastung des Menschen bei Verkehrsunfällen, Diplomarbeit TU-Wien 1989.

[6] Hering W.E., Patrick L.M.: Response Comparison of the Human Cadaver Knee and a Part 572 Dummy Knee to Impacts by Crushable Materials, 21th Stapp Car Crash Conference, New Orleans 1977.

# ACSL Users Group Österreich

Irmgard Husinsky
EDV-Zentrum, Technische Universität Wien

Die digitale Simulationssprache ACSL (Advanced Continuous Simulation Language) ist zur Simulation kontinuierlicher Modelle sehr weit verbreitet. Wir betreuen die ACSL-Installationen an der TU (ACSL auf der CYBER CDC 860 der Abt. Digitalrechenanlage und NAS AS/9160 des IEZ). Großen Anklang hat in den letzten Jahren die PC-Version gefunden. Institute der TU können bei uns eine Lizenz dafür erwerben. Derzeit ist die Version 9 für PC erhältlich. Wegen geringen Bedarfs wird ACSL auf der NAS AS/9160 abgezogen. Dafür wird die neue ACSL Version 9 auf einer DECstation 3100 installiert und getestet. Die Performance ist außerordentlich gut. Ein erstes Testbeispiel war fast doppelt so schnell wie auf der CYBER. Die DECstation ist mit einer Multi-User-Lizenz ausgestattet und ist als zentraler ACSL Server vorgesehen.

Angesichts der steigenden Zahl von ACSL-Benützern, die über die gesamte TU verstreut mit ACSL arbeiten, haben wir uns im Herbst 1988 entschlossen, eine ACSL Users Group Österreich zu gründen. Die Organisation der ACSL Users Group Österreich wird an der Technischen Universität Wien in Zusammenarbeit der Abt. Regelungsmathematik, Hybridrechen- und Simulationstechnik (Dr. F. Breitenecker) und dem EDV-Zentrum (Frau I. Husinsky) durchgeführt.

Ziele:

- Weitergabe aller Informationen der Herstellerfirma an die Benützer: z.B. ACSL User Reports, Neuankündigungen
- Besorgung von Updates, Testen von neuen Versionen, Vergabe von Unterlizenzen, Dokumentation
- Organisieren von Spezialkursen
- Förderung der Kontakte zwischen den Benützern: Austausch von Informationen über die Anwendungen der einzelnen Benützer, Austausch von Erfahrungen und Problemen mit ACSL

Zu diesem Zweck werden in unregelmäßigen Abständen Treffen organisiert und Mitteilungen ausgesandt. Folgende Treffen haben bereits stattgefunden:

- November 1988, Institut für Technische Mathematik, Gründungstreffen
- April 1989, Institut für Technische Mathematik, Vortrag Dr. Breitenecker: Frequenzbereichsanalyse in ACSL
- November 1989, Institut für Maschinendynamik und Meßtechnik, Dipl.Ing. Ecker: ACSL-Anwendungen in der Maschinen- und Fahrdynamik (KFZ-Simulationen, Unfallbiomechanik, Simulation und Chaos)

Wir planen eine Erweiterung unserer Users Group auf den gesamten deutschsprachigen Raum. Das erste diesbezügliche Treffen ist im Zusammenhang mit dem 6. Symposium Simulationstechnik geplant, der Jahrestagung der ASIM (deutschsprachige Simulationsvereinigung), am 24. September 1990 an der TU Wien. Zu diesem Treffen erwarten wir auch Joseph Gauthier aus den USA, den Mitbegründer von ACSL (Tagungsankündigung siehe auch Seite 49).

# Psychoakustische Untersuchungen zur Frequenzunterscheidung kurzer sinusförmiger Signale

Frank Rattay
Abt. für Regelungsmathematik, Hybridrechen- und Simulationstechnik,
Technische Universität Wien
Hermann Mark
Facharzt für Hals-, Nasen- und Ohrenkrankheiten, Wien

Der Mensch ist imstande, bereits akustische Signale wahrzunehmen, die so schwach sind, daß die Auslenkungen schwingender Teile im Ohr nur in der Größenordnung von Atomdurchmessern liegen. Das Ohr ist damit das empfindlichste Sinnesorgan überhaupt. Leider reagiert ein derart sensibles Organ wie das Innenohr auch sehr empfindlich auf Eingriffe bei der Beobachtung der Übertragungsmechanismen, und dadurch werden viele Messungen erheblich verfälscht. Obwohl die Erforschung der Hörvorgänge schon seit langem mit großem wissenschaftlichen Aufwand betrieben wird und mit ihr die Namen bedeutender Gelehrter wie Helmholtz, Davis oder von Bekesy verbunden sind, sind leider noch immer viele physiologische Zusammenhänge unbekannt.

An der Abteilung für Regelungsmathematik, Hybridrechen- und Simulationstechnik wird derzeit in zwei Richtungen in der Hörtheorie geforscht, mit dem Ziel, die Zusammenhänge zwischen dem akustischen Signal und dem dadurch erzeugten Reizmuster im Hörnerven besser zu verstehen. Dies erfolgt einerseits durch psychoakustische Experimente und andererseits durch Simulation der mechanischen Schwingungsvorgänge im Innenohr und der nachträglichen Umsetzung ins neurale Signal. Dieser Artikel beschreibt die psychoakustischen Experimente - erste Ergebnisse werden präsentiert.

Die erste der durchgeführten Versuchsreihen befaßt sich mit der Frequenzunterscheidung sehr kurzer Signale, die lediglich aus einer, zwei oder drei Sinusschwingungen bestehen. Derartige akustische Signale werden als kurzes Knacksen wahrgenommen und werden 'Klick' genannt. Wird einer Versuchsperson erstmals so ein Klick angeboten, so glaubt sie keinesfalls, daß darin auch eine Frequenz-Information enthalten sei. Erst durch das Anbieten eines Vergleichtones von anderer Frequenz wird festgestellt, daß sich die Wahrnehmungen unterscheiden und durch die Eigenschaft höher - oder tiefer - geordnet werden können.

Mit Hilfe des Hybridrechners konnten die notwendigen kurzen Sinusschwingungen erzeugt werden. Die vom Analogteil generierten Signale wurden über geschirmte Leitungen direkt in einen schallarmen Raum geführt, verstärkt, kontrolliert und über Kopfhörer (AGM bzw. SENNHEISER UNIPOLAR 2002) der Versuchsperson in einer Lautsträke angeboten, die einem Dauerton von 60 dB über dem Schwellwert entspricht. Die ersten dieser umfangreichen Versuchsreihen wurden am EAI-680-PACER durchgeführt, die weiteren am SIMSTAR.

Nach einer kurzen Trainingsphase begann der protokollierte Versuchsablauf. In einem von der Testperson als angenehm empfundenen Intervall von beispielsweise 300 msek wurden zwei kurze Signale unterschiedlicher Frequenz angeboten, und es mußte die Zuordnung hoch-tief oder tief-hoch getroffen werden. Nach 20 solchen Versuchen wurde der Frequenzunterschied beider Töne verkleinert und wieder eine Zwanziger-Reihe ausgeführt, solange bis die Auswertung auf 50% richtige Antworten herunterging, also keine Diskriminierung der Signale mehr möglich war. Solche Versuchsreihen wurden jeweils für Töne mit Frequenzen von 256, 512, 1024 und 2048 Hz, bestehend aus einer, zwei bzw. drei Sinusschwingungen an fünf Versuchspersonen durchgeführt.

Abbildung 1: Die Autoren bei der Arbeit. Die ersten Versuche wurden im Terminalraum des Hybridrechenzentrums durchgeführt. Durch Einlöten von Vorwiderständen konnten die 10 Volt Ausgänge des Analogrechners EAI 680 direkt mit der 'REVOX' verstärkt werden. Über die Funktionstasten des Terminals wurde jeweils ein Tonpaar abgerufen. Leider gab es hier doch einige "Betriebsgeräusche", die mit Handtüchern gedämpft wurden.

Die überraschenden Ergebnisse dieser Untersuchungen sind der Abbildung 2 zu entnehmen. Sie sind insofern überraschend als damit nachgewiesen wird, daß - im günstigsten dargestellten Fall von 1024 Hz - bereits "Töne", die bloß aus einer Einzelschwingung bestehen, unterschieden werden können, wenn sie bloß um einen Halbton verändert sind. Mit anderen Worten, eine einzelne Schwingung von 1 msek erzeugt bereits ein markant verändertes neurales Muster gegenüber einer um 25 Mikrosekunden länger oder kürzer dauernden Schwingung.

Mit derselben experimentellen Ausstattung wurde auch noch eine zweite Versuchsreihe durchgeführt. Hier ging es um die Frage, wieviele Schwingungen notwendig sind, um bei einem kurzen akustischen Signal zu einer Tonempfindung zu kommen.

Erhöht man die Anzahl der angebotenen Sinusschwingungen, so nimmt man zunächst einen Klick, dann einen Klick mit beginnendem tonalen Charakter wahr, und plötzlich klingt es wie ein "wirklicher Ton", d.h. das akustische Signal erreicht bezüglich der Tonhöhe die Qualität eines Tones derselben Frequenz, aber von längerer Dauer. Bei der schrittweisen Erhöhung der Schwingungszahl N kommt man also bezüglich der Tonhöhenempfindung zu zwei Grenzstellen, nämlich dem 'click-pitch' und dem 'tone-pitch'. Diese Grenzen sind in Abbildung 3 dargestellt. Musikalisch begabte Testpersonen bezeichneten die 'tone-pitch'-Schwelle als jene Dauer, die notwendig ist, um den entsprechenden Ton nachsingen zu können.

Diese Arbeit ergänzt eine Reihe von Interface-Berichten (siehe Referenzen), die sich auf neurale Zusammenhänge des Innenohrs beziehen, wodurch die Vielseitigkeit der Anwendungsmöglichkeiten von Hybridrechnern auch auf diesem Gebiet unterstrichen wird.

Die Autoren danken Maria Adler, Maria Resch, Martin Angebrandt und Dr. Georg Karner für ihre Geduld und Ausdauer als Versuchspersonen, sowie Hermann Stallbaumer, DI Martin Gräff und DI Heinz Silberbauer für ihre Hilfe, technische Probleme aus der Welt zu schaffen.

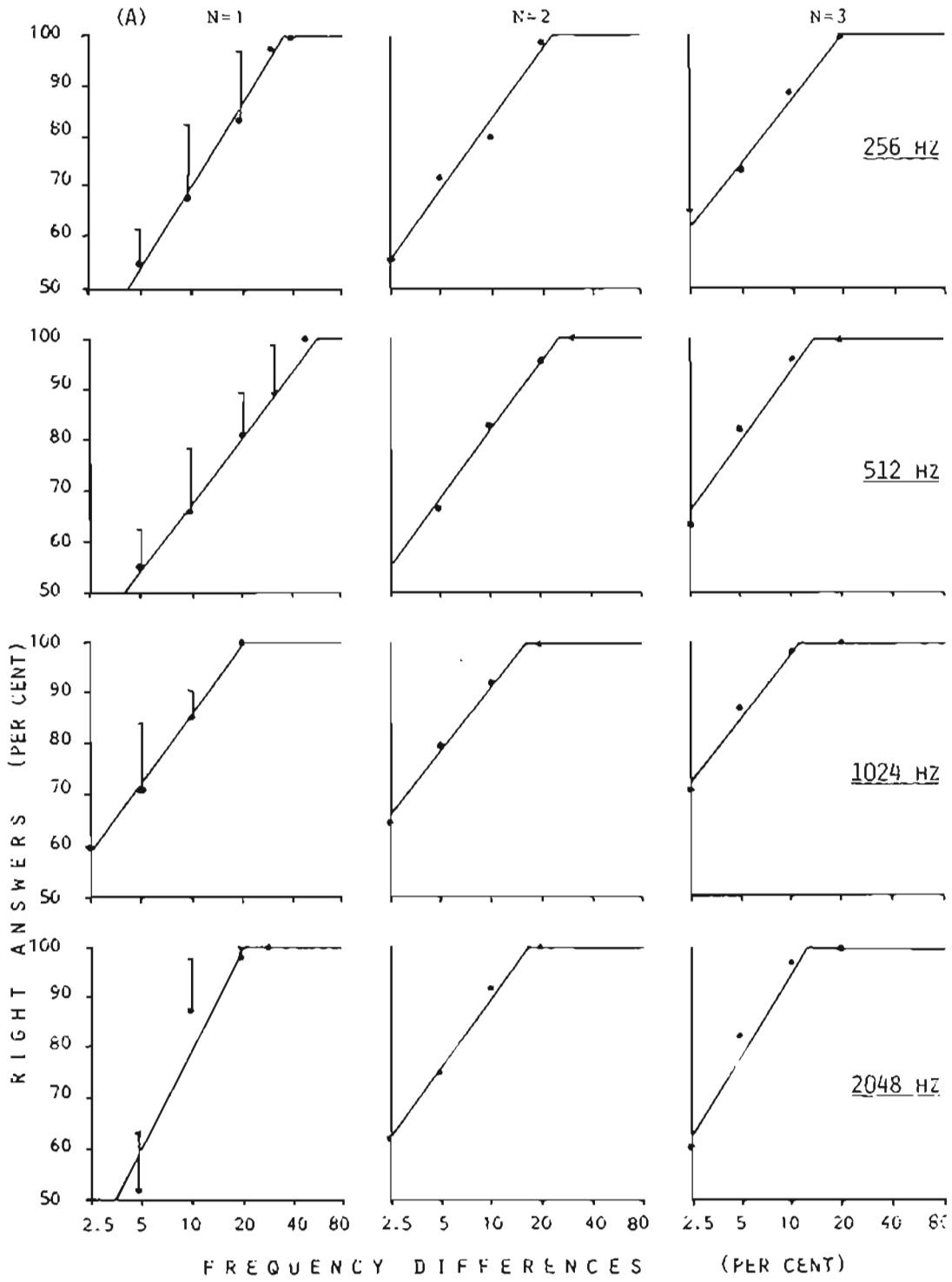



Abbildung 2: Unterscheidungsvermögen von akustischen Signalen, die aus N Sinusschwingungen bestehen. Für vier verschiedene Frequenzen werden die Häufigkeiten der richtigen Antworten in Abhängigkeit des Frequenzunterschiedes von zwei Vergleichstönen angegeben. Die gemittelten Ergebnisse von 5 Versuchspersonen sind jeweils durch Punkte gekennzeichnet. Interessanterweise läßt die logarithmische Skala für die Frequenzunterschiede der angebotenen Tonpaare einen linearen Ausgleich der Versuchsergebnisse zu.

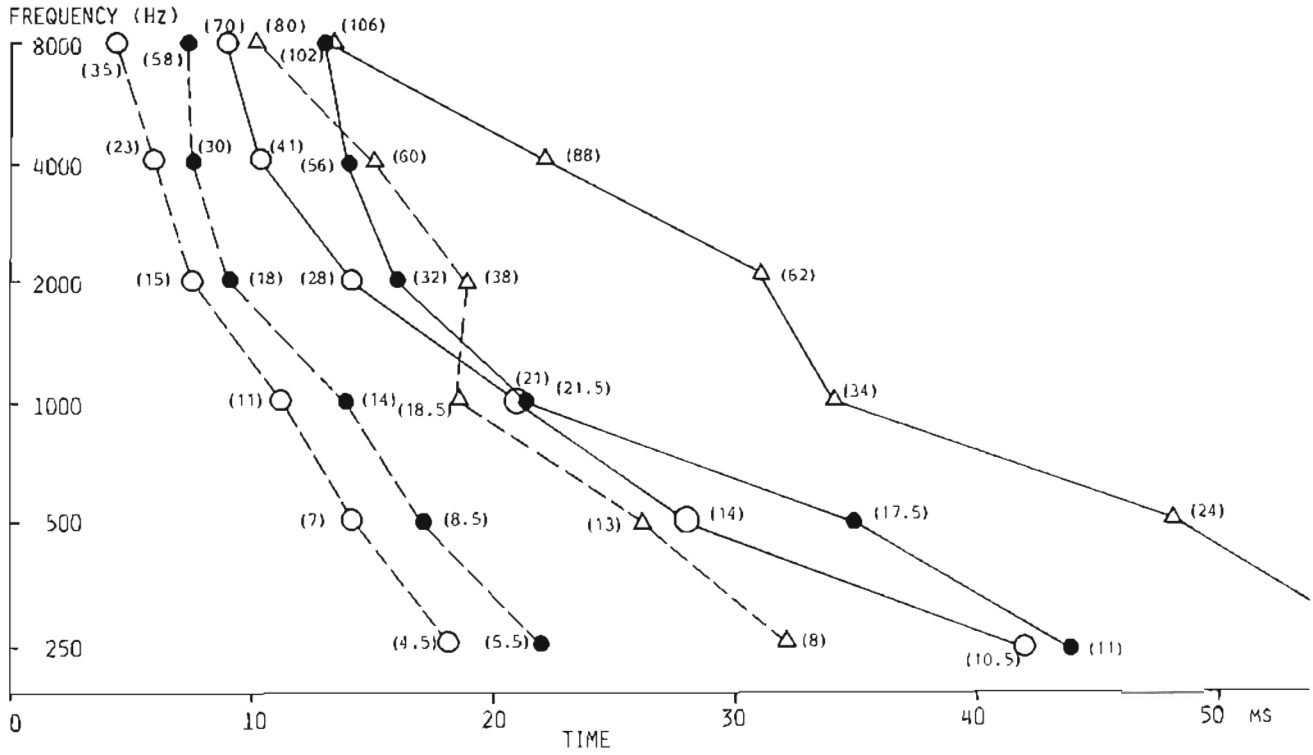



Abbildung 3: Entwicklung vom Klick zum Ton für Frequenzen von 250 - 8000 Hz. Die Ergebnisse sind für drei Versuchspersonen (○,●,△) dargestellt. Strichlierte Linien geben die 'click-pitch'- Grenze an, volle Linien zeigen die 'tone-pitch'-Schwelle. Die Zahlen in den Klammern geben die Anzahl der dabei angebotenen Schwingungen an.

## Literatur


[1] F. Rattay: Simulation der Reizentwicklung im Gehörnerv durch elektrische Anregung. Interface 20, 46-48, 1983

[2] F. Rattay und H. Mark: Stimulation des Gleichgewichtssinnes mittels Analogrechner. Interface 23, 34-36, 1986

[3] F. Rattay: Vokalsynthese. Interface 23, 15-19, 1986

[4] F. Rattay: Die Simulation elektrisch angeregter Nervenfasern. Interface 24, 23-29, 1988

# Simulation der verzögerten Alkalidesorption durch F-Zentren-Kolloide mittels HYBSYS

Ch. Polster, G. Betz, W. Husinsky
Institut für Allgemeine Physik
I. Husinsky
EDV-Zentrum
Technische Universität Wien

Elektronen-stimulierte Desorption (ESD) von Alkali-Halogeniden wird seit ca. 20 Jahren untersucht. Die Hauptmechanismen, die dabei zur Desorption von Halogenatomen führen sowie zum damit verbundenen Abdampfen von Alkalimetallen, werden dabei bereits gut verstanden; viele experimentell beobachtete Details sind jedoch noch nicht erklärt [1 - 4]. Die praktische Bedeutung dieser Untersuchungen liegt unter anderem darin, ein besseres Verständnis über das Verhalten dieser Materialien zu erhalten, wenn diese energetischer Strahlung ausgesetzt sind. Viele dieser Materialien werden z.B. für UV-Laseroptiken verwendet, wo sie hohen Strahlungsdichten ausgesetzt sind. Die "Widerstandsfähigkeit" dieser Materialien unter solchen Bedingungen ist noch weitgehend unbekannt.

Untersucht wurde an LiF-Proben das Desorptionsverhalten von neutralem Lithium bei Elektronenenergien bis zu 400eV und Kristalltemperaturen zwischen 300K und 800K. Eine der auffälligsten Effekte dabei unter den oben genannten Bedingungen ist eine Desorption des Alkalimetalls auch nach Beendigung des Elektronenbeschusses [5, 6]. Dies wird einer verzögerten Diffusion von F-Zentren zur Oberfläche und der anschließenden Verdampfung von neutralem Lithium zugeschrieben, wobei jedoch diese verzögerten F-Zentren offensichtlich erst nach dem Beschuß aus anderen während des Beschusses erzeugten Fehlstellen gebildet werden müssen. Man kann nun diese verzögerte Emission bis einige Sekunden nach dem Elektronenbeschuß beobachten (Abb.1). Dabei ist weiters augenfällig, daß abhängig von den Beschußbedingungen, diese verzögerte Desorption ein Maximum aufweist. Dieses Maximum scheint gut geeignet, Modelle zur Kinetik der Defektbildung zu überprüfen, indem man diese mit einem mathematischen Modell simuliert.

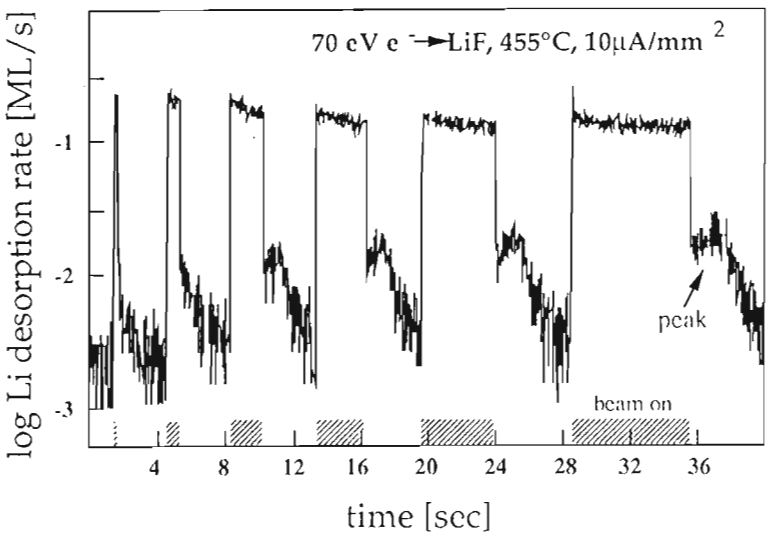



Abb. 1: Verzögerte Emission von neutralem Litium aus Litiumfluorid nach Elektronenbeschuß

Ein einfaches Modell geht davon aus, daß sich F-Zentren während des Beschusses zu M-Zentren (F2) und R-Zentren (F3) oder auch S-Zentren (F4) verbinden (Aggregation, Kolloidbildung) und in der Folge nach dem Beschuß wieder in F-Zentren zerfallen. Das Modell basiert auf folgenden Ratengleichungen:

$$\frac{dR}{dt} = k_r MF - rR$$

$$\frac{dM}{dt} = rR - mM + k_f FF - k_r MF$$

$$\frac{dF}{dt} = rR + 2mM - k_r MF - 2k_f FF - fF + P$$

In diesen Gleichungen sind F,M und R die Konzentrationen von F,M und R-Zentren. P ist der Produktionsterm für F-Zentren während des Elektronenbeschusses und r,m,f,kr,kf sind die Ratenkonstanten für verschiedene Prozesse, wie der Zerfall von R und M-Zentren sowie deren Bildung; f ist der Verlustterm der F-Zentren, wenn diese die Oberfläche erreichen und zur Desorption von Li führen. Da die F-Zentrendiffusion sehr rasch erfolgt, und alle Prozesse innerhalb 10nm von der Oberfläche stattfinden, wird in diesem Modell die Tiefenabhängigkeit nicht berücksichtigt.

Unter Verwendung des Simulationssystems HYBSYS wurden einige Ergebnisse für Modelle mit F,M und R-Zentren erstellt, die als Grundlage für Berechnungen mit höheren Aggregaten (S-Zentren) dienen können. In diesem speziellen Fall wurde das Verhalten nach Abschalten des Elektronenstrahls simuliert, d.h. der Produktionsterm P wurde Null gesetzt und eine entsprechende Anfangskonzentration für die F-Zentren angegeben Dieser Anfangswert hängt von der Beschußzeit ab und wurde aus einem anderen Set von Simulationen als Funktion der Beschußzeit ermittelt.

Das Simulationssystem HYBSYS ist ein digitales Modellentwicklungs- und Simulationswerkzeug zur Simulation kontinuierlicher Systeme auf AT PCs [7 - 9]. Zur Zeit ist eine Beta Release erhältlich. Zu den besonderen Eigenschaften von HYBSYS gehört die komfortable interaktive Experimentiermöglichkeit. Die Modellbeschreibung erfolgt blockorientiert mit speziellen HYBSYS-Operatoren. Im Deklarationsteil stehen die Parameter mit ihren Startwerten, die Variablen und die Gleichungen.

HYBSYS Befehle zur Modelldeklaration:

```
PAR
   FS = 10,MS = 1.E4,RS = 100
   FF = 500,MM = 1.E-1,RR = 10,KF = 10,KR = 1.E-1
   ZMM,ZKF
   END
VAR
   R*"R-ZENTREN",M*"M-ZENTREN",F*"F-ZENTREN"
   MTF,FTF
   END
```

```
EQU
   ZMM = MULT(2,MM)
   ZKF = MULT(2,KF)
   MTF = MULT(M,F)
   FTF = SQR(F)
   R = INTEG(RS,-RR*R,KR*MTF)
   M = INTEG(MS,RR*R,-MM*M,KF*FTF,-KR*MTF)
   F = INTEG(FS,RR*R,ZMM*M,-KR*MTF,-ZKF*FTF,-FF*F)
   END
```

Das Experimentieren erfolgt interaktiv am Terminal mit graphischer oder numerischer Ausgabe der Ergebnisse.

Experimentierbefehle, z.B.:

| | |
|---|---|
| TEND = 20 | Definition des Simulationsintervalls |
| NDT = 480 | Anzahl der Kommunikationsintervalle |
| RUN.MTD = 7 | Festlegen des Integrationsverfahrens (Runge Kutta Fehlberg-Algorithmus mit variabler Schrittweite) |
| F,M,R | Simulation und Display der Variablen F,M,R |

Die Ergebnisse wurden abgespeichert und auf einen Mac II Cx übertragen und dort mit dem Software Packet Passage II im logarithmischen Maßstab dargestellt.

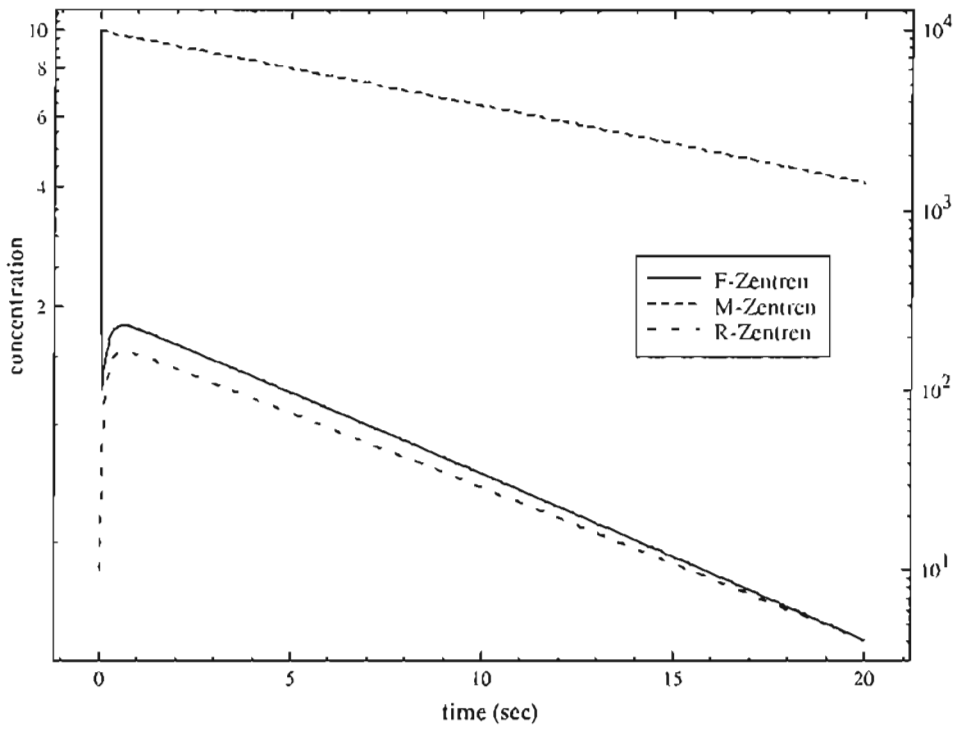



Abb. 2: Simulierte Desorption von neutralem Litium unter der Voraussetzung, daß mehr M-Zentren Agglomerate vorliegen als R-Zentren Agglomerate.

Die Simulation zeigt, daß durch geeignete Wahl der verschiedenen Ratenkonstanten (diese sind weitgehend unbekannt und müssen daher über weite Bereiche variiert werden) die experimentellen Ergebnisse zumindest qualitativ reproduziert werden können. Die Simulation ergibt für spezielle Kombinationen der Ratenkonstanten die charakteristischen Eigenschaften der Messungen:

- Nach Ende des Elektronenbeschusses sinkt die F-Zentren-Konzentration an der Oberfläche und damit verbunden die Alkalidesorption stark ab (bis zu 2 Zehnerpotenzen).
- Die Desorption (F-Zentren-Konzentration) dauert jedoch etliche Sekunden an, da F-Zentren aus dem Zerfall der Kolloide nachgeliefert werden.
- Ein Maximum der Desorption wird bei speziellen Ratenkonstanten beobachtet.

Die qualitative Beschreibung durch das Modell ist gut gelungen. Das quantitative Verhalten (die richtigen Verhältnisse zwischen Maximum und Abklingzeit) muß durch Experimentieren mit den Ratenkonstanten bzw. durch Berücksichtigung von höherwertigen Kolloiden im Simulationsmodell noch verbessert werden.

## Literatur:


[1] R.T. Williams: Rad. Eff. a. Def. in Solids **109** (1989) 175

[2] N. Itoh: Nucl. Instr. Meth. **132** (1976) 201

[3] P. Wurz, E. Wolfrum, W. Husinsky, G. Betz, L. Hudson and N. Tolk: Rad. Eff. a. Def. in Solids **109** (1989) 203

[4] G. Betz, J. Sarnthein, P. Wurz and W. Husinsky: Nucl. Instr. Meth. B (1990) in print

[5] T.A. Green, G.M. Loubriel, P.M. Richards, N.H. Tolk and R.F. Haglund Jr.: Phys. Rev. **B35** (1987) 781

[6] A.B. Lidiard: Comments Solid State Phys. **8** (1978) 73

[7] D. Solar and F. Breitenecker: HYBSYS User Manual, Beta Release, January 1990, TU Wien

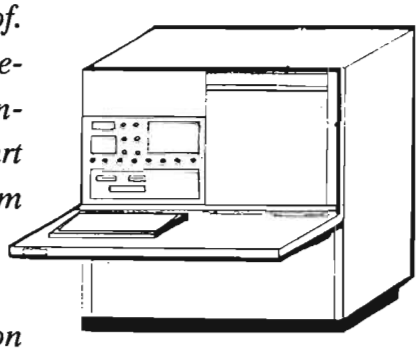
[8] D. Solar and F. Breitenecker: Das Simulationssystem HYBSYS. INTERFACE **24** (1988) 13

[9] F. Breitenecker, D. Solar and I. Husinsky: HYBSYS - A new Simulation System. Proceedings of the 3rd European Simulation Congress, Edinburgh (1989) 275

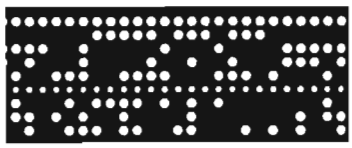
# Geschichte des EAI PACER 600A AutoPATCH Systems an der Technischen Universität Wien

Irmgard Husinsky
EDV-Zentrum, Technische Universität Wien

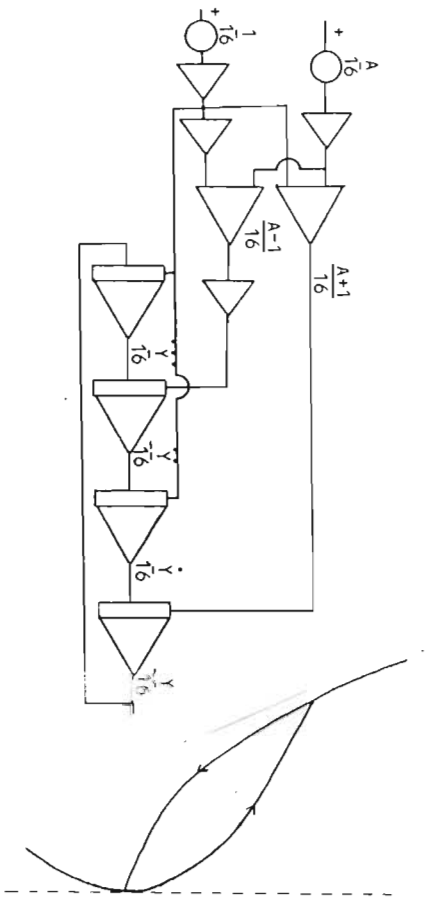
*Im Jahre 1969 wurde am damaligen I. Mathematischen Institut (Prof. Bukovics) ein EAI 680 Analogrechner installiert und im Praktikumsbetrieb sowie für wissenschaftliche Arbeiten eingesetzt. Die Analogrechentechnik wurde von diesem Institut in Vorlesungen und Praktika gelehrt und praktiziert. Dr. Kleinert war damals Assistent am Institut und nahm sich dieses Rechners besonders an.*



*Die Installation eines Digitalrechners EAI 640 bildete in Kombination mit dem Analogrechner das erste Hybridsystem. Als Speichermedium für digitale Programme stand anfangs nur eine Paper Tape Reader/Punch Station zur Verfügung. Sukzessive wurde die Peripherie des Systems erweitert.*



*Der erste "Rechenraum" war ein Raum im 2. Stock des Hauptgebäudes der Technischen Universität am Karlsplatz. Da dieser Raum nicht klimatisiert war, mußte an heißen Sommertagen der Betrieb eingestellt werden. Dieser Raum beherbergte auch die Arbeitsplätze der ersten Mitarbeiter der Abteilung Hybridrechenanlage des EDV-Zentrums, das gerade in Gründung war.*

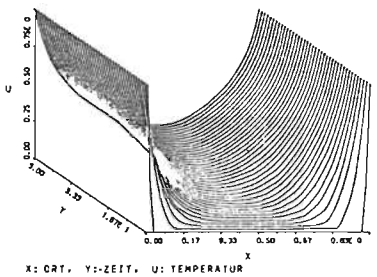


*Anfang 1973 erfolgte die lang erwartete Übersiedlung in das neue Institutsgebäude in der Gußhausstraße 27-29, wo für die Hybridrechenanlage für damalige Verhältnisse ausreichend Raum zur Verfügung stand. Es gab einen klimatisierten Rechenraum, Zimmer für die Mitarbeiter und einen eigenen Raum für Benützer mit eigens angefertigten Arbeitsplätzen zur Vorbereitung der Steckbretter.*

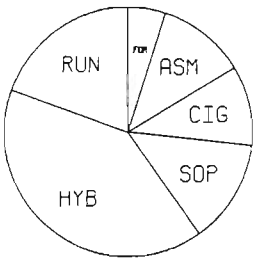
*Zur Eingabe eines Problems in den Analogrechner mußten alle notwendigen Verbindungen zwischen den analogen Rechenelementen mit Kabeln auf einem Steckbrett händisch gesteckt werden. Die Benützung des Analogrechners war dann nach Reservierungslisten jeweils nur für einen Benützer möglich, der dann sein Steckbrett auf den Rechner montierte.*

*Im Hybridsystem hatte zur Zeit der Übersiedlung ein besserer Digitalrechner, ein EAI PACER 100, den alten Digitalteil ersetzt. Das gesamte Hybridsystem nannte sich PACER 600.*

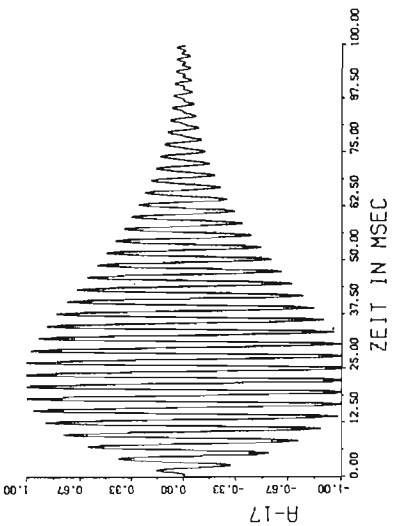
*Die Software, die die Herstellerfirma zur Unterstützung des hybriden Rechnens anbieten konnte, genügte nicht den Anforderungen der Benützer. Auch das Betriebssystem ließ zu wünschen übrig. Bald wurde mit der Entwicklung eines speziell auf den EAI PACER zugeschnittenen Betriebssystems begonnen (A. Blauensteiner). Die erste komplett unabhängige Release war 1974 das System JCS/MP 6, ein Job Control System mit Multiprogramming-Möglichkeiten, das den Batch Betrieb der Digitaljobs unterstützte. Eine digitale Programmbibliothek wurde aufgebaut. Das Vorhandensein eines Plotters sowie komfortable Ausgabesoftware brachte viele Benützer rein wegen des Plottens von Daten an die Hybridrechenanlage. 1974 war die erste Übertragung von Daten von der CYBER 74 zum PACER möglich - über Binärkarten!*




*Die Forderung nach besserer Programmunterstützung der Kommunikation Analogrechner-Digitalrechner brachte weitere Softwareeigenentwicklungen hervor. Begonnen wurde mit dem Übertragen und Zeichnen von Analogsignalen am Dataplotter, dann folgten Programme zur digitalen Unterstützung von Simulationen am Analogrechner, wie statischer Test, automatische Skalierung von Analogschaltungen (1976), FORTRAN-Routinen zur Programmierung der Synchronisation und des Datentransfers zwischen den beiden Rechnern (D. Solar). In weiterer Folge entstand daraus das Simulationssystem HYBSYS, das eine komfortable Benützerumgebung für Simulationen bietet und auch auf andere Rechensysteme übertragen wurde.*

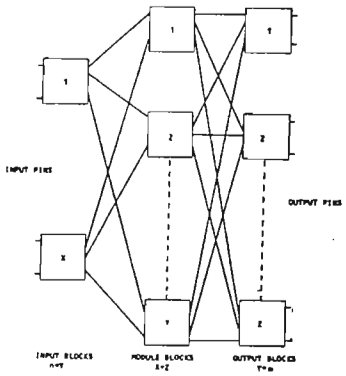



*In die Softwareentwicklungen gingen stets die Rückkopplungen der Benützeranforderungen ein, sodaß die entwickelte Software laufend den Sonderwünschen der einzelnen Benützer angepaßt wurde und dadurch sehr komfortable Produkte entstanden. Nach demselben Prinzip durchgeführte Hardwareeigenentwicklungen verbesserten die Leistungsfähigkeit des vorhandenen Systems.*

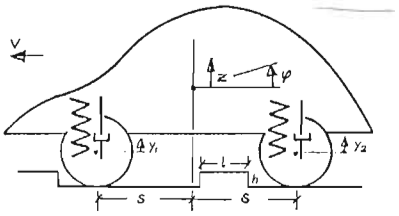



*Das selbstentwickelte Betriebssystem erfuhr in weiterer Folge die Version JCS/TS 7 (mit Time Sharing und flexiblem Massenspeicherkonzept) und JCS/VS 8 (ein virtuelles System mit Real Time Features und hybridem Time Sharing). Das Betriebssystem nutzte stets optimal die Eigenschaf-*

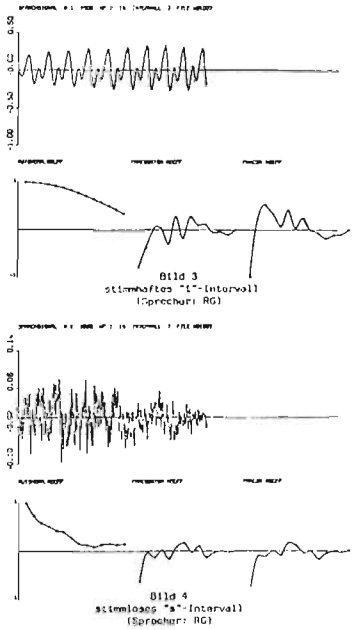
*ten der Hardware aus. Andererseits wurden auch spezielle Hardware-Anpassungen für das System gemacht.*



*Als logische Entwicklung für das Analogrechnen kam eine Switch Matrix auf den Markt, die das manuelle Stecken überflüssig machte und die Verbindungen der Analogrechenelemente automatisch vornimmt. Ein Prototyp wurde 1980 installiert. Mit der entsprechenden Unterstützung durch die Software (Betriebssystem JCS und Simulationsumgebung HYBSYS) wurde daraus das EAI PACER 600A AutoPATCH System, das in seiner Art einzigartig war. Durch die Anschaffung von mehreren Terminals war das Rechnen am Analogrechner nicht mehr an den Rechenraum gebunden. Durch das hybride Time-Sharing System MACHYS konnten sich mehrere Benützer den Analogrechner teilen.*



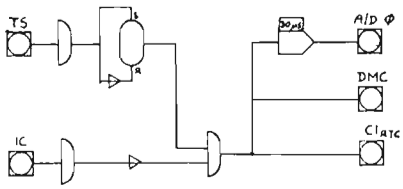
*Zufriedene Benützer, internationale Veranstaltungen und Publikationen verschafften dem PACER System viel Ansehen. Eine große Anzahl interessanter Anwendungen des Systems aus verschiedenen Wissenschaftsgebieten wurden seit 1974 im INTERFACE publiziert.*




*Die Anschaffung des neuen Multiprozessors SIMSTAR (1985) leitete die Ablöse des PACER Systems ein. 1987 übersiedelte die Hybridrechenanlage wieder in ein neues Institutsgebäude, diesmal auf der Wiedner Hauptstraße, wo wiederum größere Räume für die aus den Nähten platzende Abteilung zur Verfügung standen.*

*Immer größer werdende Fehleranfälligkeit reduzierte laufend die Einsatzfähigkeit des EAI PACER 600A AutoPATCH Systems. Alle Anwendungen wurden auf andere Systeme übertragen.*

*Durch die rasante Entwicklung der Digitalrechner und neue Technologien wurde die klassische Analog- und Hybridrechentechnik an der Technischen Universität stark zurückgedrängt. Für Simulationen stehen andere, zum Teil bessere Werkzeuge zur Verfügung. Auch die Studenten werden nicht mehr in der Analogrechentechnik ausgebildet.*




*Das Arbeiten mit dem hybriden PACER System war für alle eine interessante und wertvolle Erfahrung.*

# Das XL-8032 System

Diplomarbeit am Institut für Technische Informatik (Betreuer: Dr. W. Kleinert)
Josef Fritscher

## Einleitung

Die folgende Beschreibung beschäftigt sich mit dem dem Bau eines Computers mit dem Prozessor XL-8032 von Weitek. Die Firma Weitek ist vor allem mit ihren Floating Point Co-Prozessoren bekannt geworden. Als Beispiel sei der WTL 1167 erwähnt, der statt des 80387 in PC kompatiblen Computern mit dem Intel Prozessor 80386 eingesetzt werden kann.

Mit den XL-80xx Prozessoren versuchte Weitek nun ein Allround- Chip-Set zu entwickeln. Wie zu erwarten kam dabei die Floating Point Arithmetik nicht zu kurz. Die XL-80xx Familie besteht aus drei Chip-Sets: dem XL-8000 (ein 32 Bit Integer Prozessor), dem XL-8032 (ein XL-8000 mit zusätzlicher 32 Bit Floating Point Arithmetik) und dem XL-8064 (mit 64 Bit Floating Point Arithmetik).

Vom Konzept her suchte man den Mittelweg zwischen herkömmlichen Mikroprozessoren (z.B.: iAPX 80386, MC 68020) und den mikroprogrammierbaren Bit-Slice Prozessoren (z.B.: Am 2900). Dies gelang vor allem durch den Einsatz von Compilern, die optimierten Mikrocode erzeugen und dabei die parallelen Verarbeitungspfade des Prozessors ausnützen können.

Sowohl der Speicher, als auch das I/O System sollten die hohe Rechengeschwindigkeit des Prozessors nicht beeinträchtigen. Daher wurde ein schneller statischer RAM Speicher eingesetzt, der ohne Wartezyklen auskommt. Als Alternative dazu wäre auch eine Lösung mit Cache Speichern und großem dynamischen Arbeitsspeicher in Frage gekommen. Um jedoch innerhalb des Rahmens einer Diplomarbeit zu bleiben, wurde auf letzteres jedoch, ebenso wie auf ein vollständiges I/O System, verzichtet. Statt dessen übernimmt ein PC-AT kompatibler Computer sämtliche I/O Transfers. Die beiden Computer sind zum Datenaustausch mit einem 8 Bit parallel Port verbunden. Diese Lösung stellt einen Kompromiß zwischen einem seriellen Interface, welches komplizierter im Aufbau und langsamer gewesen wäre, und einem direkten Busanschluß, der eine zu starke Abhängigkeit vom Hostsystem bedeutet hätte, dar.

## Eckdaten des Systems:

- 32 Bit Floating Point Arithmetik mit 8 MIPS und 16 MFLOPS Peak Performance.
- Harvard Architektur, d.h. getrennter Programm- bzw. Datenspeicher. Auf der Speicherkarte befinden sich 128 KByte Programmspeicher und 256 KByte Datenspeicher. Dabei werden statische Speicherchips verwendet, deshalb müssen keine Wartezyklen eingefügt werden.
- Superskalare RISC Architektur
- Entwicklungssystem mit optimierendem C Compiler.

## Mehr Details über das System

Der eingesetzte Prozessor ist Weitek's XL-8032 mit einer 32 Bit Integer ALU, einer Shift/Merge Unit, 64 Bit Codebus, 32 Bit Datenbus, 32 general purpose Registern, einem Stack mit einer Tiefe von 32 Wörtern on-the-Chip, mit je einer Wortbreite von 32 Bit. Die Floating Point Unit enthält zusätzliche 32 Register, mit ebenfalls 32 Bit Wortbreite, eine 32 Bit Floating Point Arithmetik Unit mit vierstufiger Pipeline und einer on-the-Chip lookup Tabelle für Divisionen. Dabei wird der IEEE Floating Point Standard unterstützt. Die stromsparende High Speed CMOS Technologie gewährleistet dennoch eine hohe Arbeitsleistung von 8 MIPS und 16 MFLOPS als Spitzenwert. Die Harvard Architektur des Systems ermöglicht einen hohen Datendurchsatz. Jeweils ein Datenbus und ein Adressbus für Programmcode und Daten werden eingesetzt. Der Datenbus für den Programmcode ist 64 Bit breit, während die anderen Busse 32 Bit breit sind.

Zur einfachen Speichererweiterung befindet sich der Arbeitsspeicher auf einer eigenen Karte, getrennt von der CPU. Auf der Speicherkarte befinden sich 384 KByte (eine Erweiterung auf 1.5 MByte ist vorgesehen). Davon sind 128 KByte Programmspeicher und 256 KByte Datenspeicher. Eine Bestückung mit zusätzlichen Speicherkarten ist nach dem Hinzufügen einer Motherboard möglich. Für den Speicher selbst werden besonders schnelle statische RAM Chips verwendet, damit der Prozessor Speichertransfers, ohne warten zu müssen, sofort durchführen kann (zero wait states).

Gesteuert wird das XL-8032 System von einem Host Rechner, z.B. einem PC-AT kompatiblen Computer. Auf dem PC befindet sich auch die Software-Entwicklungsumgebung. Unter dem Betriebssystem XENIX sind diverse Development Tools verfügbar, so z.B. ein Software Simulator. Des weiteren sind ein C Compiler, der dazugehörende Parallelizer, sowie ein Assembler, ein Disassembler und ein Linker vorhanden. Den Datentransfer zwischen dem XL-8032 System und dem Host Computer übernimmt ein Monitor im XL Sytem, der mit einem Shadow Process über einen XENIX Device Driver im PC kommuniziert. Fertige Programme können mit entsprechender Treibersoftware auch unter DOS laufen. Die Verbindung zwischen den beiden Computern wird mit einem 8 Bit Parallel Port (Centronics Standard) realisiert. Abb. 1 zeigt die beiden miteinander gekoppelten Computer, während Abb. 2 einen Blick in das XL-8032 System wirft. Ein vereinfachtes Blockschaltbild stellt Abb. 3 dar.

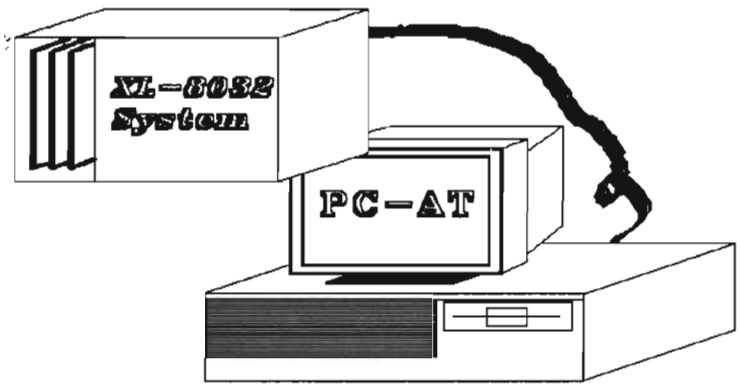



Abbildung 1

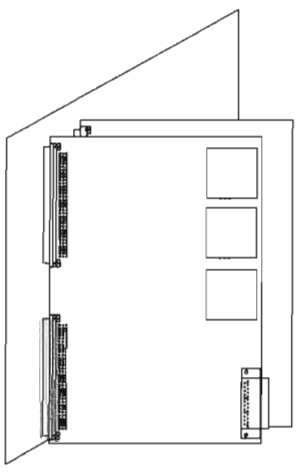


Abbildung 2

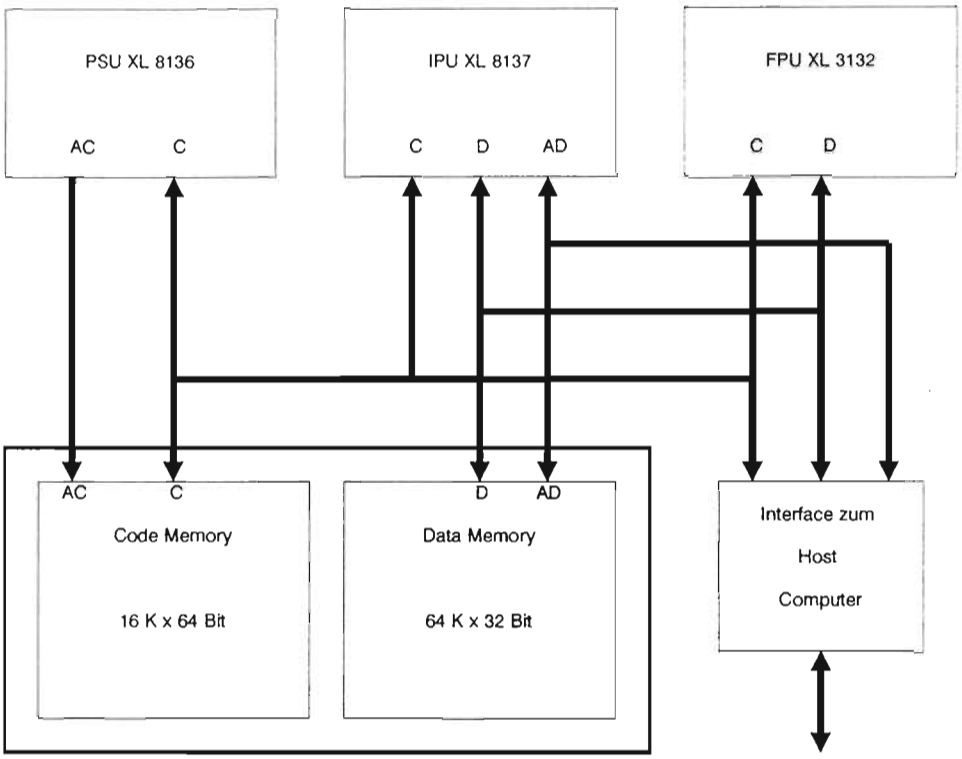



Abb. 3: Ein grobes Blockschaltbild des XL-8032 Systems, mit Program Sequencing Unit (PSU), Integer Processing Unit (IPU) und Floating Point Unit (FPU)

## Ein Überblick über das System

Abb. 4 zeigt ein Blockschaltbild der zwei Karten, aus denen der Computer besteht. Es sind dies die Prozessorkarte und die Speicherkarte. Auf der erstgenannten befindet sich die Weitek CPU XL-8032, die sich aus der Program Sequencing Unit (PSU) XL 8136, der Integer Processing Unit (IPU) XL 8137 und der Floating Point Unit (FPU) XL 3132 zusammensetzt. Weiters sind der Taktgenerator, die OP-Bus Dekodierung, das 'Code Address Latch', das 'Data Address Register', ein 24 Bit Adresszähler, das System State Register (SSR), der Centronics Interface Block und die notwendigen Treiber auf der Prozessorkarte untergebracht. Die Speicherkarte besteht aus zwei Speicherbaugruppen (dem Codespeicher und dem Datenspeicher), sowie jeweils einer Adressdekodierung.

Während der Programmlaufzeit erzeugt die PSU die Codeadressen (AC Bus), die nach einer Zwischenspeicherung im 'Code Address Latch' und nach der Adressdekodierung dem Codespeicher zugeführt werden. Der Codespeicher versorgt PSU, IPU und FPU mit den auszuführenden Instruktionen über den Codebus (C Bus). Die IPU erzeugt die Datenadressen (AD Bus) zum Speichern und Laden von Daten zum und vom Datenspeicher (mittels D Bus). Die Speichersteuerung (Speichern, Laden und Übernehmen der Datenadresse) erfolgt über den von der PSU generierten OP-Bus (5 Bit breit), der in der OP-Bus Dekodierung verarbeitet wird. Die Datentransfers mit den Externen Registern (System State Register und Interface Machine) werden über den AD Bus vorgenommen, wobei die Registeradresse im OP-Bus kodiert ist.

Mit dem System State Register (SSR) kann die CPU von außen gesteuert werden. Es liefert die Signale RESET-, EXT1- (ein externer Interrupt) und STALL- bzw. ABORT-. Die beiden zuletztgenannten Leitungen werden zum Anhalten der CPU verwendet, damit der Host Computer die Steuerung übernehmen kann.

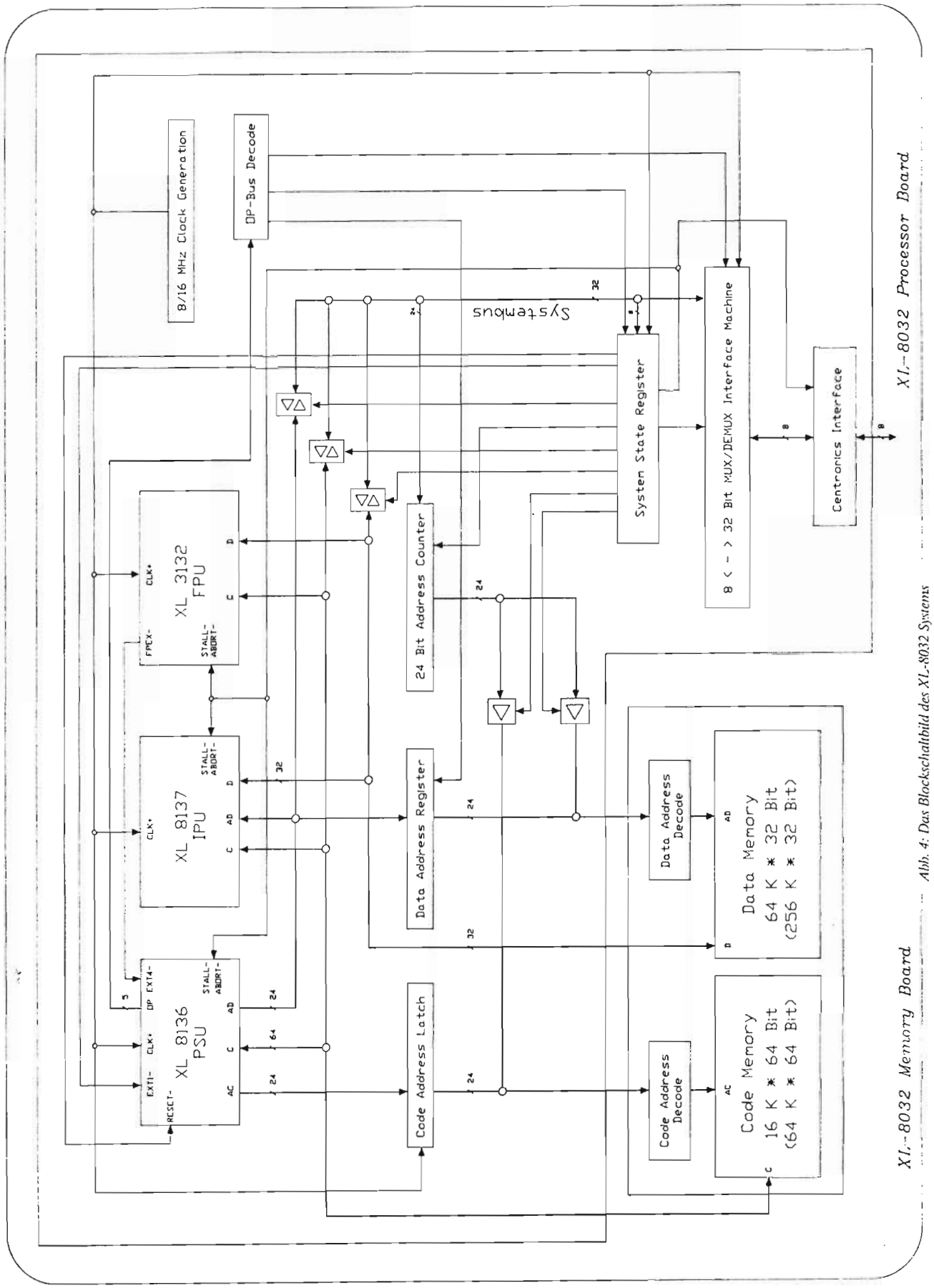



*Abb. 4: Das Blockschaltbild des XL-8032 Systems*

**Abbildung 4: Das Blockschaltbild des XL-8032 Systems**

Zum 'Downloaden' von Programmen und Daten werden das Centronics Interface, das System State Register und der Adresszähler benötigt. Dazu wird der Code- bzw. der Datenbus mit dem Systembus verbunden, über den dann die übertragenen Worte vom Centronics Interface in den Code- bzw. Datenspeicher gelangen. Der Datentransfer in umgekehrter Richtung ist ebenso möglich. Die aufeinanderfolgenden Adressen werden dabei vom Adresszähler erzeugt, der mit jedem übertragenen Wort inkrementiert wird. Mit dem System Status Register kann der Adresszähler vor dem Datentransfer mit der Startadresse geladen werden. Das Zusammensetzen der Bytes zu Wörtern mit einer Breite von 32 Bit aus dem Centronics Interface übernimmt die Interface Machine. In der umgekehrten Richtung werden die Wörter von ihr in Bytes zerlegt. Mit dem System State Register, in das der Host Computer über das Centronics Interface schreiben kann, erfolgt die Steuerung der hier beschriebenen Funktionen.

Die zum System gehörende Software unterstützt das 'Downloaden' und Ausführen der, mit dem Cross Development System erstellten, Programme.

**Erste Erfahrungen**

Nach der Fertigstellung des XL-8032 Systems stellte sich sofort die Frage: Wie schnell ist der Computer im Vergleich zur Konkurrenz? Dazu wurde ein kleines C Programm entwickelt (siehe Listing). Die Laufzeitmessungen ergaben eine ungefähr gleiche Ausführungsgeschwindigkeit von Cyber (mit maximaler Optimierung) und XL System (ca. 38 Sekunden).

```
#include <stdio.h>
#include <math.h>

/*              #define LOOP 100000
                #define COUNT 100

mit XL handgestoppt: ca. 38 s          */

#define LOOP 100000
#define COUNT 100

float avals[COUNT];
float bvals[COUNT];
char wert[20];

prep(ival,stream)
int ival; char stream[];
{
  stream[0]= ival / 100 + '0';
  ival%= 100;
  stream[1]= ival /10 + '0';
  stream[2]= ival % 10 + '0';
  stream[3]= 0xa;
  stream[4]= 0;
}
```

```
main()
{
  int i, isum;
  long j;
  float sum, diff;

  sum= 0;
  diff= 0;
  for (i=0;i<COUNT;i++)
        avals[i]= (float) rand();
  for (i=0;i<COUNT;i++)
        bvals[i]= (float) rand();
  for (j=0;j<LOOP;j++)
        for (i=0;i<COUNT;i++)
                {
                  sum= sum + avals[i] * bvals[i];
                  diff= diff - avals[i] / bvals[i];
                }
  isum= (int) sum;
  isum%= 1000;
  prep(abs(isum),wert);
  write(1,wert,4);
}
```

**Anwendungsmöglichkeiten**

Das XL-8032 System ist mit dem XENIX PC 8 an der Abt. Simulationsrechenanlagen des EDV-Zentrums der Technischen Universität Wien gekoppelt. Die dort entwickelten Programme (C oder Assembler) können mit dem Runtime-Monitor 'xrun' in das XL System geladen und ausgeführt werden. Das genaue Kommando lautet:

**xrun filename**

Beim Linken von Objektfiles mit dem Linker 'aln' ist zu beachten, daß das Datensegment bei Adresse 0 (wird mittels -D Option bestimmt) beginnt.

# Multi-Transputer System

A. Blauensteiner
EDV-Zentrum, Technische Universität Wien

## Einleitung

Neben den konventionellen Mikroprozessoren, welche in Rechnerarchitekturen wie den PCs oder den Minis ihre Bedeutung haben, spielt ein spezieller Typus von Mikroprozessoren, der sogenannte Transputer, eine immer wichtigere Rolle. Der Transputer ist ein unabhängiges Rechenelement, das einen eigenen Befehlssatz besitzt. Seine spezielle Eigenschaft liegt darin, daß jeder einzelne Transputer 4 Kommunikationspfade besitzt, über die er mit anderen Transputern zusammengeschlossen werden kann. Dadurch lassen sich mehrere Transputer netzförmig miteinander verbinden. Durch die Vernetzung einerseits und die darüber laufende Kommunikation andererseits lassen sich also zahlreiche Transputer in einer parallel rechnenden Systemarchitektur miteinander verbinden, wodurch die Rechnerleistung, die aus dem vernetzten System gewonnen werden kann, durchaus in den Bereich der konventionellen Mainframe-Systeme kommt.

Der besondere Einsatz der Transputersysteme besteht daher darin, die spezifischen Probleme und Anwendungen durch eine geeignete Topologie mit vernetzten Transputern anzupassen. Dadurch kann bereits vor der Programmierung ein hoher Grad an natürlicher Parallelisierung der Probleme vorbereitet werden. So findet sich auch der hauptsächliche Einsatz der Transputersysteme zum Beispiel in Problemen der Kernphysik, der Molekulardynamik sowie der Bildverarbeitung.

## Das System

### Hardware

Anfang April 1989 konnte nach umfangreichen Vorbereitungen ein Transputer-Multiprozessorsystem IMPULS 2400 im Rahmen eines Projektes des Fonds zur Förderung der wissenschaftlichen Forschung bestellt werden. Projektträger sind das Institut für Kernphysik an der Technischen Universität Wien sowie die Abteilung Simulationsrechenanlagen des EDV-Zentrums. Das Transputersystem wurde bald darauf von der Firma IMPULS aus Wien installiert und in Betrieb genommen. Es besteht aus 64+2 Transputern T800 mit 17 MHz Taktfrequenz und jeweils 2 Megabyte RAM. Die Transputer sind nicht alle direkt miteinander nach einer festen Topologie verbunden, sondern über 3 sogenannte Link-Switches verschaltet, sodaß über diese Link-Switches programmgesteuert Verbindungen hergestellt werden können. Das System IMPULS 2400 hat zweimal 640 Megabyte Plattenspeicher und ein 2.3 Gigabyte Streamertape für die Datensicherung.

Das System wird durch 2 Frontends kontrolliert: einen ATARI, der über die zwei zusätzlichen Transputer ein Filesystem kontrolliert, über das die zwei Festplatten angesprochen werden können, und einen PC/XT mit einer 80 Megabyte-Platte und einem Transputer-Board für die Initialisierung des Systems sowie dem Netzanschluß für die Benutzung - einen ETHERNET-Adapter.

Nimmt man die VAX11/780 mit FPA als Maßeinheit, so erreicht man bereits mit einem einzelnen T800 Transputer die 3 bis 4-fache Skalarperformance. Das 64-Transputersystem mit 128 Megabyte bringt

bei der parallelisierten Anwendung etwa 200-fache Skalarperformance. Das entspricht etwa einer 6-Prozessor IBM 3090-600 mit 128 Megabyte User-Space bzw. einer 5-Prozessor CRAY X-MP/516. Daran erkennt man das besondere Preis/Leistungs-Verhältnis von derartigen Transputersystemen. Neben dem günstigen Anschaffungspreis - unter 2 Millionen Schilling - fallen kaum Betriebs- oder Wartungskosten an.

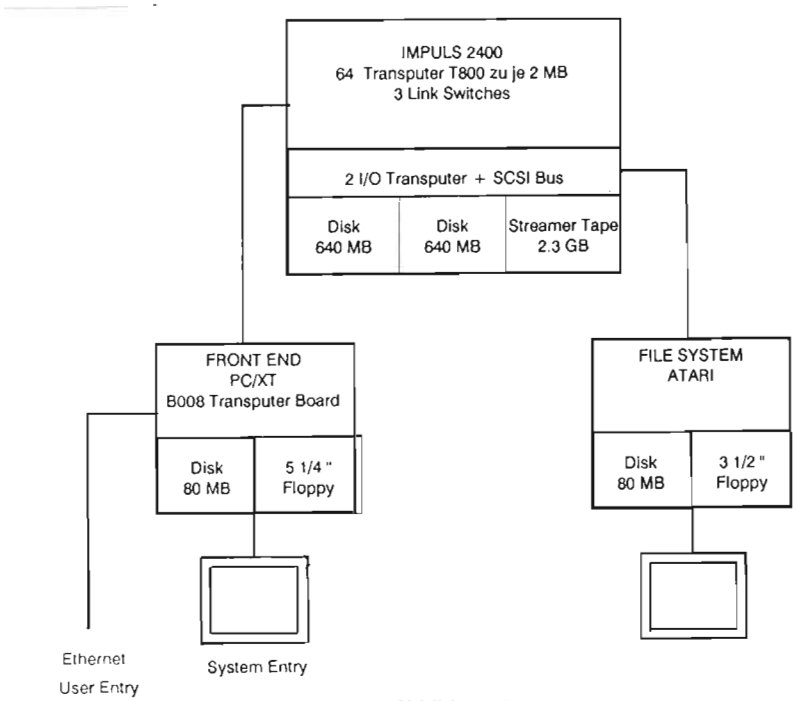



Abbildung 1
Hardwarekonfiguration

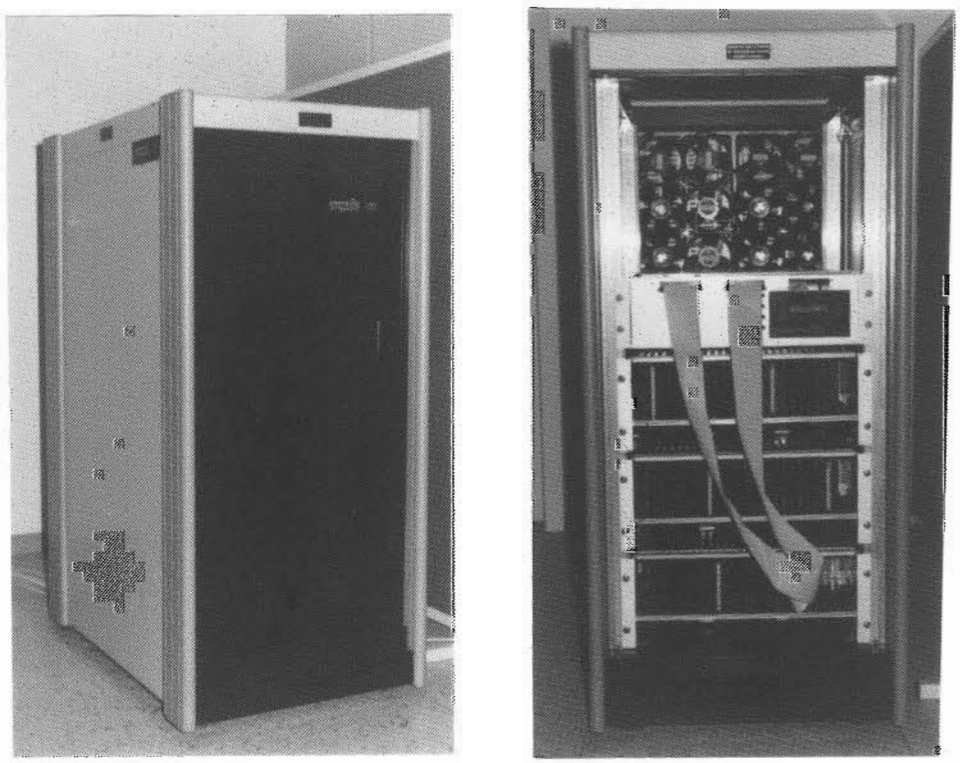


Abbildung 2
Das Transputer System IMPULS 2400

**Software**

Die Softwareunterstützung für das IMPULS 2400 System beginnt beim TDS, Transputer Development System, einem Unterstützungspaket für den Frontend-PC, um das gesamte System zu laden und zu testen. Zur Programmierung stehen 3 Compiler zur Verfügung: Parallel-FORTRAN, Parallel-C und OCCAM.

Während FORTRAN und C gewohnte Programmierumgebungen bieten, wird durch die Programmiersprache OCCAM voll auf die Multitransputerstruktur des Systems bzw. auf die Kommunikationsmechanismen eingegangen. OCCAM erfordert nur eine kurze Programmierumstellung, liefert aber den weitaus besten Code und damit wesentlich schnellere Exekutionszeiten als C bzw. FORTRAN. OCCAM ermöglicht Low-Level-Programmierung wie z.B. die Zuweisung von Variablen zu speziellen Memoryadressen, Zugriff zu Timern oder die Einbettung von Maschinencode. Außerdem ist OCCAM am besten zu debuggen. Um ein Multitransputerprogramm zu erstellen, muß man die einzelnen Prozesse den entsprechenden Transputern zuordnen. In OCCAM wird diese Verteilung mit der sogenannten Konfigurationsbeschreibung durchgeführt. Diese Beschreibung wird von einem eigenen Konfigurer bewerkstelligt. Dieser Konfigurer erstellt alle notwendigen Bootstrap- und Rootinginformationen, um das gesamte Transputersystem in seiner entsprechenden Vernetzung zu laden und speichert diese Informationen zusammen mit dem compilierten Code in einem Programmteil ab.

Es besteht die Möglichkeit, einzelne Objects mit verschiedenen Compilern zu erstellen und zusammenzubinden. Außerdem gibt es eine Reihe von Unterprogrammpaketen mit den nötigen Funktionen, mathematischen Funktionen, sowie Routinen, um die Link-Switches zu bedienen. Mit Hilfe dieser Link-Switches kann das gesamte Transputernetzwerk dynamisch vom Zeitpunkt des Programmeinsatzes dahingehend wirken, die Topologie bzw. einzelne Verbindungen zu ändern. Für unterschiedliche Applikationen können verschiedenen Topologien softwaregesteuert konfiguriert werden.

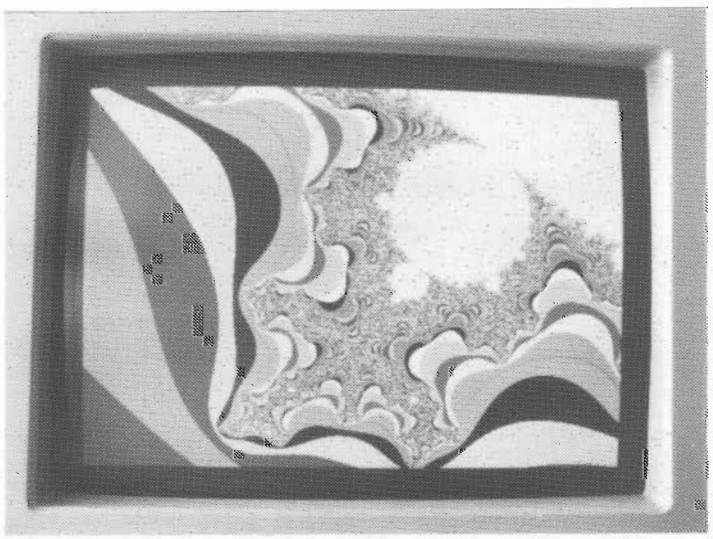


Abbildung 3
Mandelbrotmenge, am IMPULS 2400 System mit 63 Transputern gerechnet,
ca. 2000 mal so schnell wie auf einem 286-PC

## Topologien

Wie bereits erwähnt, wurde das Transputersystem IMPULS 2400 mit 3 Link-Switches konfiguriert. Jeder einzelne dieser 3 Link-Switches ermöglicht 32 Anschlüsse an beliebige Transputer, sodaß 32 bidirektionale Kommunikationspfade pro Link-Switch verschaltet werden können. Da die einzelnen Schaltungen nur pro Link-Switch programmiert werden können, war eine fixe Zuordnung von Transputerlinks zu den Link-Switches notwendig. Diese Zuordnung wurde so gewählt, daß eine möglichst große Flexibilität für Topologien besteht.

Mit der gewählten Zuordnung können sowohl ein geschlossener 8 x 8 Torus konfiguriert werden, als auch verschiedene Größen von diagonal verstrebten Ringen, als auch Standardtopologien, wie ein vollständiger binärer Baum. Außerdem läßt sich in Hinblick auf das Projektproblem ein 2 x 2 Torus konfigurieren, um grundsätzliche Funktionen bzw. Tests auszuführen. Die Links stehen auch miteinander in Verbindung, sodaß auch ein mehrstufiges Rooting möglich ist. Ebenso besteht ein Anschluß an die beiden Transputer, welche für das Filesystem zuständig sind, um den Datentranfer zu ermöglichen. Die Link-Switches werden durch den Frontend-PC programmiert bzw. in einem Fall indirekt über eine Zwischenschaltung erreicht.

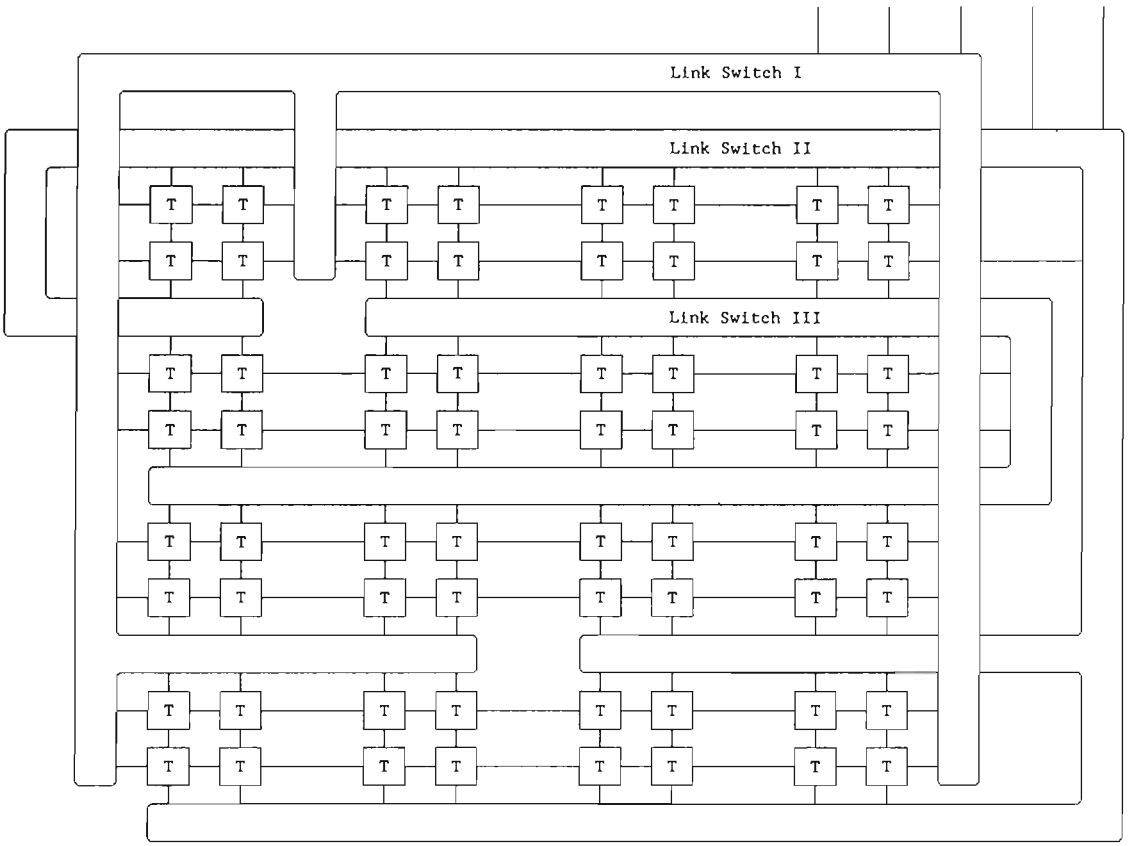
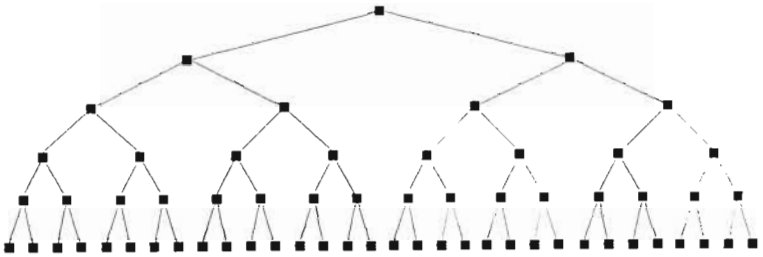
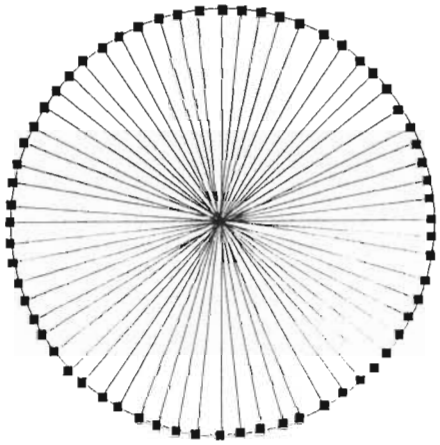


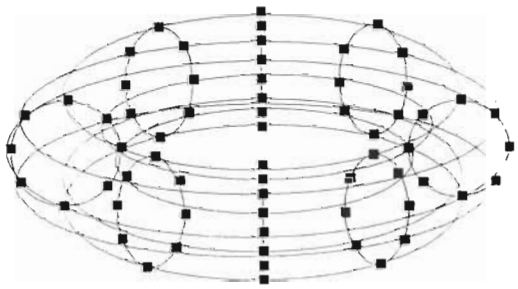

Abbildung 4
Transputernetzwerk des Systems IMPULS 2400



Binärer Baum



Diagonal verstrebter Ring



Geschlossener Torus

Abbildung 5
Beipiele für Topologien

## Das Projekt

Bei dem Projekt des Forschungsförderungsfonds handelt es sich um ein wichtiges Problem der Gitterquantenchromodynamik, nähmlich die Simulation von Quarkeinschluß in den Hadronen. Diese Problem ist hervorragend für die Parallelisierung auf gitterartig vernetzten Multiprozessorsystemen geeignet. Es ist ein schönes Beispiel dafür, wie - ausgehend von der algorithmischen Struktur eines technischen Problems - das Transputersystem und dessen Kommunikationsnetzwerk ausgewählt werden und dabei eine hervorragende Rechenleistung durch die natürliche Parallelität erreicht werden kann.

Bei dem Projekt werden bereits existierende Fortranprogramme auf das Multitransputersystem in OCCAM übertragen. Von der Abteilung Simulationsrechenanlagen des EDV-Zentrums wurde das Filesystem auf dem ATARI-Frontend entwickelt, um eine komfortable Ausgabe der erheblich großen Datenmengen zu erreichen. Diese Daten werden über das lokale TUNET zur Weiterverarbeitung an den Institutsrechner des Instituts für Kernphysik übertragen. Trotz der enormen Beschleunigung der Exekutionszeiten wird das Transputersystem für das eben erwähnte Problem 24 Stunden am Tag im Einsatz stehen.

# Linda: Konzepte und Implementierungen

M. Gräff
Technische Universität Wien
EDV-Zentrum

## Einleitung

Der Einzug paralleler Rechnerarchitekturen in die Landschaft der allgemeinen Datenverarbeitung scheint unaufhaltbar. Damit kommen auf die Informatik große Aufgaben zu, soll diese Entwicklung nicht ein evolutionärer Rückfall in jene Zeiten bedeuten, in denen Programme jeweils für eine dezidierte Hardware geschrieben wurden.

Die Werkzeuge für parallele Programmierung lassen sich in vier Klassen einteilen: Auto-parallelisierende Compiler, deren Hauptaufgabe in der Übertragung bestehender Fortranprogramme (*dusty decks*) auf neue Rechner besteht, interaktive Werkzeuge zur Formulierung paralleler Programme, Übersetzer von Problembeschreibungen auf hohem, anwendungsnahen Niveau in parallele Programme und parallele Programmiersprachen.

Die zwei wichtigsten Parallelisierungsprinzipien für algorithmische Programmierung sind *funktionale Parallelisierung* und *Datenpartitionierung*. Bei der funktionalen Parallelisierung besteht das Programm aus inhomogenen, parallel ablaufenden Tasks. Die Aufgabe, bestehende serielle Programme automatisch funktional zu parallelisieren, scheint heute kaum allgemein lösbar zu sein.

Das Prinzip der Datenpartitionierung ist für Programme anwendbar, bei denen eine auf einer Datenmenge definierte Aufgabe durch Zuordnung von Teilmengen zu Prozessen parallelisiert werden kann. In diesem Bereich ist hingegen breiter Raum für die Entwicklung halbautomatischer und automatischer Werkzeuge. Im numerischen Bereich sind auto-parallelisierende/vektorisierende Compiler [12] bereits zu Produkten gereift, die geschachtelte `DO`-Schleifen auf mehrere Prozessoren verteilen. Bei den interaktiven Werkzeugen zur Erstellung paralleler Programme ist SUPERB [11] bis zum kommerziellen Einsatz gelangt.

Einen vielversprechenen Ansatz der vierten Klasse stellt Linda dar, das von Forschern an der Yale University entwickelt worden ist. Linda ist weniger eine neue Programmiersprache als vielmehr ein Programmiermodell, das bestehende sequentielle Programmiersprachen in orthogonaler Weise ergänzt. Linda stellt einfache, leicht anwendbare, trotzdem aber sehr mächtige Kommunikationsmechanismen zur Verfügung. Neben dem anschaulichen Programmiermodell muß bei den Vorteilen von Linda die leichte Übertragbarkeit der Programme erwähnt werden, da Linda unabhängig von Architektur und Topologie ist.

Nach der Vorstellung der Konzepte von Linda soll dieser Beitrag am Beispiel realer Implementierungen zeigen, wie solche Systeme effizient aufgebaut werden.

## Programmierung von Parallelrechnern

Mit den heutzutage verfügbaren billigen und leistungsfähigen Mikroprozessoren können Parallelrechner gebaut werden, deren Leistungsfähigkeit die von *traditionellen Supercomputern* (Vektorrechnern) erreicht oder sogar übertrifft. Bei den herkömmlichen Architekturen (z.B. IBM 3090, Cray X-MP, Y-MP) ist eine Leistungssteigerung hauptsächlich durch die Entwicklung in Richtung von Parallel-Vektorverarbeitung zu erzielen. Die Anwendbarkeit der neuen *parallelen Supercomputer* (z.B. Intel iPSC, NCube, Connection Machine) wurde vielfach unter Beweis gestellt [10], doch haben Programme auf diesen Rechnern strukturell wenig mit sequentiellen Programmen gemeinsam; die Anwendbarkeit wird daher immer wieder heftig diskutiert.

Während bei Supercomputern nicht die leichte Programmierbarkeit sondern die Lösbarkeit von neuen Aufgaben ausschlaggebend ist, muß für die Akzeptanz von Parallelrechnern kleinerer Leistungsfähigkeit

eine Lösung des Programmierproblems gefunden werden, da ansonsten die Verwendung paralleler Maschinen an Stelle von Mini-Supercomputern oder High-End-Workstations nicht attraktiv ist.

Parallelrechner mit mehreren Instruktionsströmen werden für gewöhnlich in zwei Klassen unterteilt: In *Multiprozessorsystemen mit gemeinsamem Speicher* (MGS) kann jeder Prozessor direkt auf einen globalen Speicher zugreifen, wohingegen in *Systemen mit verteiltem Speicher* (MVS) jeder Prozessor nur seinen lokalen Speicher direkt ansprechen kann; der Zugriff auf die lokalen Daten anderer Prozessoren ist nur durch Kommunikationsmechanismen möglich und ist um Größenordnungen langsamer.

Typische Programmiertechniken auf MGS sind beispielsweise Monitore, Semaphore und Locks, wohingegen die Kommunikation in MVS meist über Nachrichten oder Kanäle abgewickelt wird. Die Portierung von Programmen zwischen verschiedenen Architekturen ist mit hohem Aufwand verbunden, da nicht nur syntaktische Änderungen notwendig sind, sondern die Programmiermodelle vollkommen verschieden sind. Eine wichtige Forderung an ein Konzept für explizite parallele Programmierung ist daher Architekturunabhängigkeit. Die Erstellung von effizienten Programmen ist sogar auf Vektorrechnern heute noch immer eine maschinenspezifische Aufgabe [6]. Ein Werkzeug für die explizite parallele Programmierung soll daher ein anschauliches, einfaches Programmiermodell effizient auf verschiedene Architekturen umsetzen können.

# Linda

Linda wurde ursprünglich von David Gelernter in Yale [1, 9] entwickelt. Die erste Implementierung wurde von Nicolas Carriero [5] auf der AT&T S/Net, einem MVS, durchgeführt. Mittlerweile wurden Linda-Systeme auf MGS (z.B. Encore Multimax, Sequent Balance und Symmetrie, Apollo DN10000, Silicon Graphics), MVS (Intel Hypercube, Cogent XTM) und auf Rechnernetzwerken (Vax) implementiert.

In Linda kommunizieren Prozesse über einen logischen gemeinsamen Speicher, der als *Tuple Space* (TS) bezeichnet wird. Der TS kann als assoziativer Speicher angesehen werden.

Auf dem TS sind drei Operationen erklärt:

**out** stellt ein Tupel in den TS

**in** sucht ein passendes Tupel im TS und löscht es dort

**rd** sucht ein passendes Tupel im TS und kopiert es

Ein *Tupel* ist ein geordnete Sammlung von Daten, z.B. das Tupel

```
("hello", 5, true)
```

enthält drei Einträge: einen String, eine Zahl und einen logischen Wert.

Die Operation

```
out("hello", 5, true)
```

stellt das Tupel in den TS. Die out-Operation blockiert nie, es können mehrere gleiche Objekte im TS sein.

Die Suche nach einem Tupel im TS (*Matching*) erfolgt durch Vergleich von Struktur und Werten des Tupels:

```
int     i;
boolean b;
in("hello", ?i, ?b);
```

liefert (irgend)ein Tupel im TS, dessen erstes Feld aus dem String "hello" besteht, dessen zweites Feld eine Zahl und dessen drittes Feld ein logischer Wert ist. Bei diesem Beispiel ist "hello" der Schlüssel, zu dem ein passendes Tupel gesucht wird. Wenn ein passendes Tupel gefunden wird, dann werden den Variablen i und b die entsprechenden Werte aus diesem Tupel zugewiesen und das Tupel wird aus dem TS entfernt. Die Felder von Tupeln sind nicht ausgezeichnet, Felder können bei manchen Operationen Schlüssel und bei anderen Wertfelder sein:

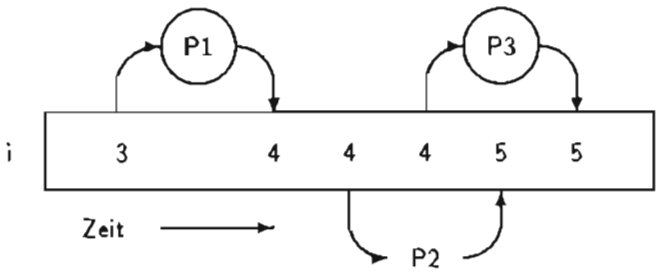
```
int     i;
boolean b;
char    *s;
in("hello", ?i, ?b);
in(?s, 3, true);
```

zeigt, daß Schlüssel und Werte ihre Bedeutung vertauschen können.

Die Operation rd liefert dasselbe Ergebnis wie in, das Tupel verbleibt aber im TS. Sowohl in- als auch rd-Operationen blockieren bis ein passendes Tupel im TS steht. Es gibt auf den meisten Linda-Systemen auch nicht-blockierende Versionen inp und rdp.

## Synchronisation

Die Synchronisation von Datenzugriffen soll am Beispiel eines gemeinsamen Zählers veranschaulicht werden, den mehrere Prozesse inkrementieren. Auf einem MGS kann die folgende Situation auftreten, wenn der gemeinsame Zähler i nicht geschützt wird:



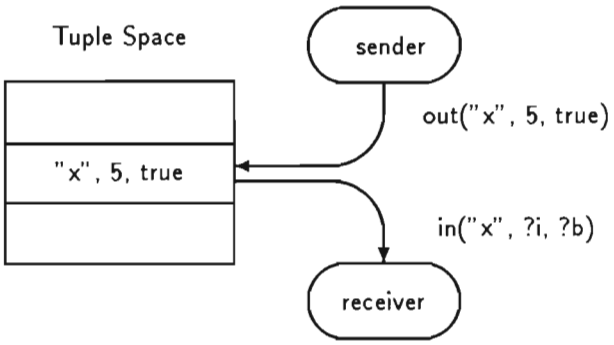

In einem Linda-Programm

```
in("i", ?i);
out("i", i+1);
```

entfernt die in-Operation den Zähler aus dem TS; Prozesse, die auf den Zähler zugreifen wollen, werden bis zum Ende der out-Operation blockiert.

## Entkopplung von Prozessen

Die Übermittlung einer Nachricht von einem Prozeß sender an einen Prozeß receiver entspricht einer out- und einer in-Operation:

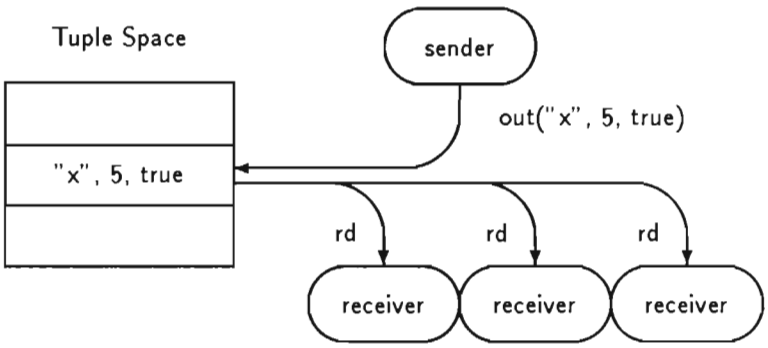



Der Sender kann schneller arbeiten als der Empfänger, da `out`-Operationen nicht blockieren. Bei asynchroner Programmierung ist jedoch eine Flußkontrolle vorteilhaft, falls der Sender wesentlich schneller ist. Wenn der Empfänger seine Arbeit vor dem Sender beendet hat, wartet er an der nächsten `in`-Operation.

Linda entkoppelt den sendenden und empfangenden Prozeß sowohl in Zeit als auch Ort. Die Entkopplung in der Zeit ist gegeben, da die `out`-Operationen asynchron ausgeführt werden und weil Nachrichten über den TS weitergereicht werden. Unter Linda können sogar Programme kommunizieren, die nicht gleichzeitig ausgeführt werden, da Nachrichten im TS warten können. Diese Eigenschaft ist besonders für ein Betriebssystem wichtig, da es Dienste für meist noch nicht existierende Programme bereitstellen muß.

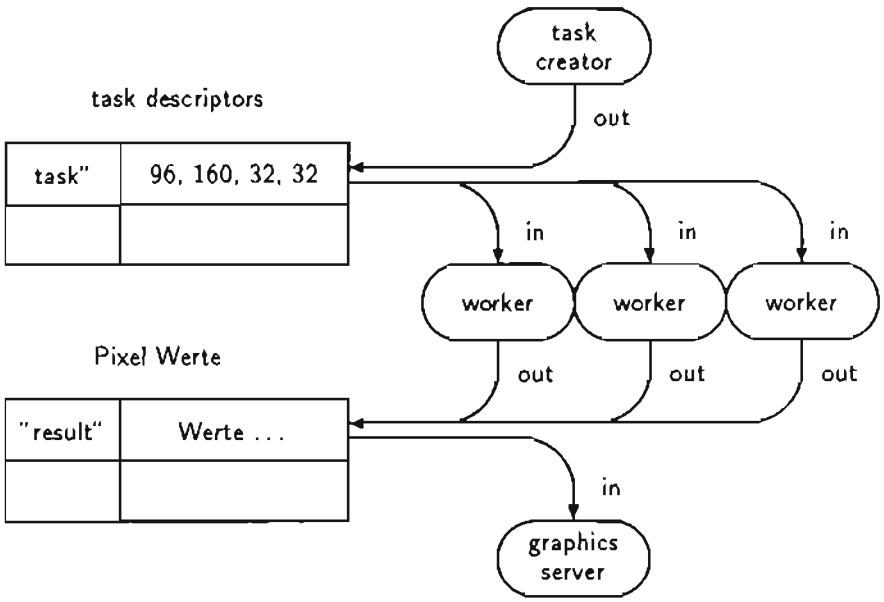
Die Nachrichten sind auch im Ort entkoppelt, weil die Identifizierung einer Nachricht durch ihren Inhalt erfolgt. Der Sender einer Nachricht muß weder den Empfänger noch seine Lokalisierung kennen, und der Empfänger benötigt kein Wissen um die Identität oder die Lokalisierung des Senders. Kommunikationsstrukturen in Linda können daher sehr dynamisch sein. Insbesondere können Sender und Empfänger von Nachrichten auf beliebigen Prozessoren angeordnet sein, ihre Zuordnung zu Prozessoren kann sogar dynamisch während der Laufzeit veränderlich sein. Das ist eine sehr wichtige Eigenschaft von Linda und erleichtert das Schreiben von parallelen Programmen wesentlich.

Es ist in Linda auch möglich, eine Nachricht mehreren Empfängern bekanntzugeben. In diesem Fall führt der Sender eine gewöhnliche `out`-Operation aus, während jeder Empfänger ein `rd` durchführt. Da Nachrichten durch ihren Inhalt identifiziert werden, muß der Sender die Anzahl der Empfänger nicht kennen.




## Prozessorfarmen

In einer Prozessorfarm führen mehrere Arbeiter gleichartige Aufgaben durch, die ihnen von einem Master zugeteilt werden. In diesem Beispiel nehmen wir an, daß die durchzuführende Arbeit aus einer Graphikanwendung wie Ray-Tracing, Bildverarbeitung, oder Berechnungen von Mandelbrotmengen kommt. Eine naheliegende Parallelisierung ist die Zuteilung von Teilgebieten des Bildes an verschiedene Arbeiter. Die Arbeitsbeschreibungen (task descriptors) bestimmen jeweils die Fläche des zu berechnenden Bildes, z.B. definiert `(96,160,32,32)` ein Rechteck des Bildes an der Stelle (96, 160) und mit Breite und Höhe gleich 32 Pixels.




Der Master stellt die Arbeitsbeschreibungen in den TS:

```
for (x = 0; x < maxx; x += 32)
for (y = 0; y < maxy; y += 32)
    out("task", x, y, 32, 32);
```

Jeder Arbeiter liest eine Arbeitsbeschreibung:

```
in("task", ?x, ?y, ?width, ?height);
```

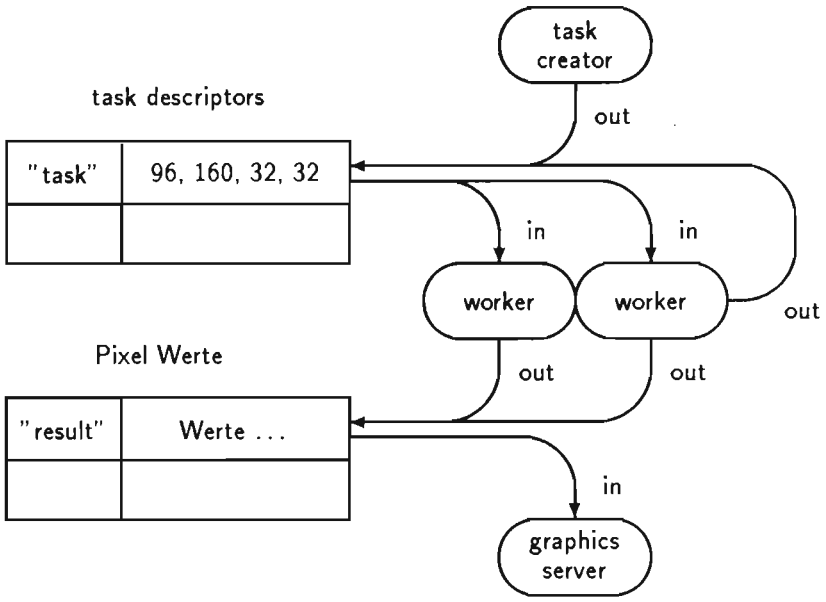
Danach berechnet er den entsprechenden Teil des Bildes, stellt das Ergebnis in den TS, und anschließend holt er die nächste Arbeitsbeschreibung.

Die Nachrichtenübermittlung über den TS hat zur Folge, daß der Master unabhängig von der Anzahl und Plazierung der Arbeiter ist. Die Anzahl der Arbeiter kann sogar dynamisch verändert werden.

## Dynamische Lastverteilung

Will man mit einer Prozessorfarm eine gute Prozessorauslastung erreichen, dann müssen die Arbeiter ungefähr gleich viel Arbeit verrichten. Oft ist die Arbeit aber sehr ungleichmäßig verteilt und ein oder zwei Arbeiter benötigen wesentlich länger als der Rest, während die anderen ungenutzt warten. Solche Situationen sind besonders bei Graphikanwendungen typisch, wo manche Aufgaben um Größenordnungen komplexer als andere sein können.

Mit Linda können solche Aufgaben leicht gelöst werden. Wenn ein Arbeiter entdeckt, daß er ein besonders schwieriges Teilgebiet zu bearbeiten hat, so teilt er es in kleinere Stücke und gibt diese Teilgebiete an andere Arbeiter ab.




Weil Nachrichten im TS zwischengespeichert werden, bemerken die Arbeiter nicht welcher Prozeß ihre Arbeitsbeschreibungen generiert hat.

## Effizienz

Es steht außer Frage, daß Linda ein mächtiges Werkzeug für die parallele Programmierung ist, doch wie groß ist der Preis dafür? Der erste Eindruck, daß jede `in`- oder `rd`-Operation eine Suche im TS erfordert, läßt Linda untragbar langsam erscheinen. Es gibt jedoch eine Anzahl von Techniken um den Suchprozeß zu beschleunigen oder zu eliminieren.

Mit Linda ist wie mit jedem anderen Werkzeug mit hoher Abstraktionsstufe ein Mehraufwand verknüpft, der hier aber nicht allzu groß ist [2, 4]. Die Situation ist etwa mit Hochsprachen vergleichbar, bei denen der Mehraufwand durch den Vorteil leichterer Programmierbarkeit und Übertragbarkeit mehr als ausgeglichen wird. Dynamische Lastverteilung, wie im obigen Beispiel, kann mit Linda sehr einfach implementiert werden; Linda erleichtert auch das Experimentieren mit der parallelen Struktur eines Programms, um dadurch eine bessere Rechnerauslastung zu erzielen.

Der verbreitetste Weg, die Effizienz von Linda-Programmen zu verbessern, ist die Verwendung eines Präprozessors [2], der die zur Laufzeit notwendigen Suchvorgänge verringert oder überhaupt eliminiert. Die durch die Analyse der Operationen auf dem TS gewonnenen Daten werden dazu benutzt, um die Kommunikation in optimaler Weise auf die jeweilige Zielhardware abzubilden, sodaß im laufenden Programm anstatt Linda-Operationen die auf der Maschine verfügbaren Mechanismen (Semaphore, Nachrichten) verwendet werden.

Die Verwendung eines Präprozessors ist nun aber nicht immer möglich. Soll Linda beispielsweise zusammen mit interpretierenden Sprachen wie Lisp, PostScript oder Prolog, oder für die Kommunikation von Programmen, die nicht zur gleichen Zeit übersetzt werden, z.B. Betriebssystem und Anwendungsprogramm, benutzt werden, so kann die Verwendung des TS nicht bei der Übersetzung analysiert werden.

Eine andere Methode effiziente Linda-Systeme zu realisieren besteht in der Einschränkung der Operationen. Die Beschränkung auf einfache Schlüssel erlaubt, durch die Verwendung von Hashing gute Performance zu erzielen ohne dabei die Programme einer statischen Analyse zu unterziehen.

# Yale Linda

Die Linda-Implementierungen der Yale Universität und die kommerziell vertriebenen Derivate der Firma Scientific Computing Associates weisen als wesentliches Element einen Präprozessor auf. Linda war zunächst als eigene Sprache mit starker Verwandschaft zu C konzipiert, wohingegen heute unter Linda eine orthogonale Ergänzung bestehender Sprachen verstanden wird (C-Linda, Fortran-Linda in Yale, Modula-2-Linda von Siemens [3, 4]).

Die Aufgabe des Präprozessors soll zuerst an einem einfachen Beispiel motiviert werden: Betrachten wir dazu ein Linda-Programm mit zwei Prozessen, die `out("x", i)` bzw. `in("x", ?j)` eines Tupels durchführen, wobei i und j Integer-Variable sind. Da der Schlüssel `"x"` zur Übersetzungszeit konstant ist, erkennt der Linda-Präprozessor, daß dieses Paar von Linda-Operationen als Queue zwischen den Variablen `i` und `j` implementiert werden kann. Auf einem MGS kann das als gemeinsame Queue mit einem Semaphor realisiert werden, auf einem MVS wird die Realisierung aus dem Abschicken einer asynchronen Nachricht bestehen. In beiden Fällen ist kein Suchen notwendig, und die für die jeweilige Zielhardware optimale Realisierung wird ohne Änderung am Quellprogramm gewählt.

In einer anderen oft auftretenden Situation wird dasselbe Tupel-Feld als Schlüssel vewendet, der zur Laufzeit veränderlich ist. In diesem Fall kann die Suche durch Hashing beschleunigt werden. Der Yale-Präprozessor teilt die Verwendung des TS in fünf Klassen ein,

**Semaphor:** Keine Daten, sondern nur Synchronisationsinformation

**Queue:** Daten werden übermittelt, der Schlüssel ist zur Laufzeit konstant

**Hashtabelle:** Der Schlüssel ist nicht konstant, aber immer an der gleichen Stelle

**private Hashtabelle:** Der Schlüssel ist meist an einer Stelle, mitunter jedoch an einer anderen; die Suche nach dem ersten Schlüssel erfolgt durch Hashing, nach den anderen Schlüsseln wird linear gesucht

**„Mess“:** Volle lineare Suche im TS

und führt die Realisierung von jeder dieser Klassen durch geeignete Abbildung auf Mechanismen, die die Zielmaschine zur Verfügung stellt, durch. Die bisherigen Erfahrungen mit Linda zeigen, daß in den meisten Fällen die aufwendige Laufzeitsuche eliminiert werden kann.

# Kernel Linda

Kernel Linda [7, 8], die Linda-Implementierung von Cogent Research, ist ein dynamisches Linda-System, d.h. Applikationen müssen nicht durch einen Präprozessor analysiert werden. Aus Effizienzgründen sind Operationen auf einfache Schlüssel beschränkt, Tupel können daher als Schlüssel-Wert-Paare aufgefaßt werden. Die Performance des Systems ist verhältnismäßig gut, da der Aufwand der Berechnung der Hashfunktion verglichen mit den Kosten der Nachrichtenübertragung gering ist. Die Beschränkung auf einen einzigen Schlüssel ist zwar eine Einschränkung von Linda, die Hauptvorteile von Linda, Übertragbarkeit und die Entkopplung in Zeit und Ort, bleiben aber erhalten.

Als Ausgleich für die Einschränkung auf einfache Schlüssel wurden mehrere (benannte) TS, die als *Dictionaries* bezeichnet werden, eingeführt. In Yale-Linda werden mehrfache Schlüssel meist zur Einteilung der Kommunikation in verschiedene Klassen verwendet; diese Aufgabe kann mit mehrfachen TS effizienter durchgeführt werden. Die Unterteilung des TS in kleine Stücke ist außerdem für die Performance von Suchen und die Größe der Hashtabellen vorteilhaft. Mehrfache TS haben auch andere Vorteile für parallele Programmierung: Wenn nur ein einziger globaler TS verwendet wird, so können Namenskonflikte zwischen Programmen auftreten, die denselben Schlüssel zur Identifikation verwenden; mehrere TS erlauben hingegen eine bessere Strukturierung der Kommunikation.

Ein dynamisches Linda-System hat gegenüber statischen Systemen den Vorteil, daß die Bindung an interpretierende Sprachen möglich ist. Das sprachunabhängige Design von Kernel Linda ermöglicht es, daß Programme in verschiedenen Programmiersprachen durch Linda-Operationen kommunizieren. Durch

die Einbindung von Kernel Linda als IPC-Mechanismus in das Betriebssystem der Cogent XTM ist es außerdem möglich, daß Anwendungsprogramme mittels Linda-Operationen auf Betriebssystemdienste zugreifen.

## Zusammenfassung

Das Programmiermodell von Linda ermöglicht eine einfache und anschauliche Formulierung explizit parallelisierter Programme auf maschinenunabhängigem Niveau. Verschiedene Realisierungen dieser Konzepte haben zu sehr unterschiedlichen Lösungen geführt. Ausser der Klarheit seiner Konzepte hat Linda auch aus folgenden Gründen gute Chancen sich zu einem Quasi-Standard für paralleles Programmieren zu entwicklen:

- Die Verwendung von Linda ist nicht auf numerische Anwendungen beschränkt.
- Linda ist für funktionale Parallelisierung und für Datenpartitionierung geeignet.
- Die Einbettung von Linda in objektorientierte Sprachen (z.B. C++ bei Kernel Linda) erlaubt eine sehr elegante Einbindung der Kommunikation in bestehende Programmiersprachen.
- Das Programmiermodell beruht auf maximaler Entkopplung paralleler Prozesse, Programmieren muß nicht völlig neu erlernt werden.
- Linda bietet sich als maschinenunabhängige Zwischensprache für Werkzeuge zur automatischen Parallelisierung an.

## Literatur


[1] S. Ahuja, N. Carriero & D. Gelernter, *Linda and Friends*, IEEE Computer, Vol. 19(1986), Nr. 8, S. 26–34.

[2] R. Bjornson, N. Carriero, D. Gelernter, J. Leichter, *Linda, The Portable Parallel*, Research Report YALE/DCS/RR–520(Jan. 1988).

[3] L. Borrmann, M. Herdieckerhoff, A. Klein, *Tuple Space Integrated into Modula-2, Implementation of the Linda Concept on a Hierarchical Multiprocessor*, Proc. CONPAR 88(1988), Stream C, S. 92–99.

[4] L. Borrmann, M. Herdieckerhoff, *What is the Price of Elegance? Efficiency of Parallel Programs in a Linda System*, PARS Mitteilungen, Nr. 6(Juni 1989), S. 163–176.

[5] N. Carriero, *Implementing tuple space machines*, Doctoral Dissertation, Yale University, 1987.

[6] Hui Cheng, *Vector Pipelining, Chaining, and Speed on the IBM 3090 and Cray X-MP*, IEEE Computer, Vol. 22(Sept. 1989), S. 31–46.

[7] Cogent Research, *Kernel Linda Specification*, Cogent Research Technical Note 89.17.

[8] Cogent Research, *Linda meets UNIX*, Cogent Research Technical Note.

[9] D. Gelernter, *Getting the Job Done*, Byte Magazine, Nov. 1988, S. 301–308.

[10] J. Gustafson, *"Are Parallel Computers Special Purpose"?*, Supercomputing, Dec. 1989, S. 30–31.

[11] H. P. Zima, H.-J. Bast, M. Gerndt, *SUPERB - A Tool for Semiautomatic MIMD/SIMD Parallelization*, Parallel Computing 6(1988)

[12] H. P. Zima, B. M. Chapman, *Supercompilers for Parallel and Vector Computers*, erscheint in acm Press Frontier Series, Addison-Wesley.

# Austrian Center for Parallel Computation: Zielsetzung und Aktionsplan

o.Univ. Prof. Dr. Bruno Buchberger, Universität Linz
o.Univ. Prof. Dr. Günter Haring, Universität Wien
Dr. Wolfgang Kleinert, Technische Universität Wien
Doz. Dr. Christoph Überhuber, Technische Universität Wien
o.Univ. Prof. Dr. Hans Zima, Universität Wien
o.Univ. Prof. Dr. Peter Zinterhof, Universität Salzburg

## Zusammenfassung

Das *Austrian Center for Parallel Computation* **(ACPC)** wurde von Bruno Buchberger, Günter Haring, Wolfgang Kleinert, Christoph Überhuber, Hans Zima und Peter Zinterhof mit dem Ziel gegründet, das zukunftsweisende Gebiet der Parallelverarbeitung in Österreich gemeinsam zu fördern.

Das ACPC hat sich zum Ziel gesetzt,

- die *österreichische Forschung* im Bereich der Software für parallele Systeme zu forcieren und schwerpunktartig zu betreiben,
- durch koordinierte universitäre Ausbildung (Diplom- und Doktoratsstudien) *in Österreich einen Strom von hochqualifizierten jungen Wissenschaftlern und Entwicklern heranzubilden* und
- durch Kooperation mit Industriepartnern einen *Technologietransfer* herbeizuführen und hierdurch einen starken *innovativen Impuls für die österreichische Industrie* (insbesondere die Software-Industrie) zu erzeugen.

Das ACPC wird kollegial von seinen Mitgliedern geleitet. Sprecher ist Hans Zima, Universität Wien.

## Zielsetzung

Wissenschaft, Technik und Industrie stellen immer höhere Anforderungen an die Leistungsfähigkeit von Computern. Probleme wie die Berechnung der Masse subatomarer Elementarteilchen, die computerbasierte Synthese neuer Werkstoffe, die Bestimmung der atomaren Struktur von Viren oder die Abschätzung der Auswirkungen von genetischen Defekten auf die Entstehung von Erbkrankheiten übersteigen bei weitem die Möglichkeiten konventioneller Rechner.

Während sich die Rechengeschwindigkeit einzelner Prozessoren in Zukunft nur noch innerhalb relativ enger (durch physikalische Gesetzmäßigkeiten vorgegebene) Grenzen erhöhen läßt, bietet *Parallelität* die Möglichkeit der faktisch unbegrenzten Leistungssteigerung. Das Prinzip besteht darin, daß mehrere Prozesse, die in mehreren Bearbeitungselementen (Prozessoren, Computern) gleichzeitig ablaufen und miteinander kommunizieren können, im Hinblick auf die Erfüllung einer bestimmten Aufgabenstellung kooperieren.

*Parallelverarbeitung (Parallel Processing, Parallel Computation)* ist jenes Gebiet der Informatik, welches sich mit Problemlösungsmethoden (Algorithmen) und systemtechnischen Realisierungen auf der Basis paralleler Prozesse beschäftigt. Fortschritte auf diesem Gebiet spielen eine *entscheidende Rolle für die Effizienzsteigerung computerbasierter Problemlösungen.* Manche Problemstellungen sind erst durch Parallelisierung zu bewältigen. Dementsprechend vielfältig sind die Anwendungen der Parallelverarbeitung. Wichtige Beispiele sind:

- Numerische Simulation
- Bildverarbeitung und Bilderkennung,
- Symbolisches Rechnen (analytisches, algebraisches, geometrisches und logisches Rechnen)
- VLSI Entwurf
- Softautomation (CAD/CAM, Roboter-Software)
- Funktionales und logisches Programmieren
- Expertensysteme und wissensbasierte Systeme
- Konnektionismus, neuronale Netzwerke
- Computerbasierte Synthese neuer Werkstoffe

Das Gebiet der *Parallelverarbeitung entwickelt sich weltweit immer mehr zu einer Schlüsseltechnologie.* Während bei der Lösung der Hardwareprobleme in den letzten Jahren große Fortschritte gemacht wurden – bereits heute sind ausgereifte Systeme mit Tausenden von Prozessoren kommerziell verfügbar – ist die Entwicklung von Software und die Erforschung ihrer Methodik sowie der theoretischen Grundlagen weit zurückgeblieben. Dies ist der Hauptgrund dafür, daß parallele Rechner ihr Potential als hochwertige Werkzeuge für Wissenschaft, Forschung und Industrie heute noch nicht adäquat erfüllen können.

Die *große Herausforderung für die Zukunft besteht darin,*

- für ein breites Spektrum von Anwendungsbereichen *parallele Algorithmen* zu entwickeln,
- neue *Programmiersprachen* zu entwerfen, in denen Parallelverarbeitung leicht und überschaubar beschrieben werden kann und in denen gleichzeitig die volle Potenz der verfügbaren parallelen Architekturen ausgenutzt werden kann,
- *Programmierumgebungen* zu schaffen, die effiziente Software-Werkzeuge für die Entwicklung paralleler Programme zur Verfügung stellen. Hierzu gehören insbesondere automatische Systeme zur Parallelisierung sequentieller Programme.

Fortschritte auf diesen Gebieten sind nur möglich, wenn außerdem

- eine *Wissenschaft der parallelen Programmierung*, das heißt ein solider theoretischer und formaler Rahmen für paralleles Programmieren geschaffen wird.

Wir fassen diese Gebiete unter dem Titel **Software für parallele Systeme** zusammen. Sie stellen die zentrale Thematik für das ACPC dar.

Die im ACPC erarbeiteten Forschungsresultate werden in großem Umfang in die konkrete Entwicklung von Software für parallele Maschinen umgesetzt werden können. Dieser **Technologietransfer** wird völlig neue Märkte in der Software-Industrie eröffnen, in denen mit *jährlichen Wachstumsraten von 25% und mehr* gerechnet werden kann, weil Parallelverarbeitung faktisch den gesamten Anwendungsbereich von Computern durchzieht und den Prozeß der Programmentwicklung softwaretechnisch grundlegend verändern wird.

Die Software-Industrie stellt für Österreich – angesichts seiner exportorientierten Wirtschaft – eines der **zukunftsträchtigsten industriellen Potentiale** dar. Der Bereich der *Parallverarbeitung eröffnet hier besondere Möglichkeiten*, weil

- *weltweit ein großes Defizit* auf diesem Gebiet herrscht und die Chancen denjenigen gehören, die jetzt die Weichen für die künftige Entwicklung stellen,
- wir am *Beginn eines zu erwartenden Booms stehen*, bei dem Österreich weit vorne dabei sein kann,
- eine wesentliche Ressource in der formal mathematisch-logischen Ausbildung junger Forscher und Entwickler liegt und hier *Österreich durch sein überlegenes Ausbildungssystem besonders begünstigt ist*,
- die *notwendigen Investitionen für Hardware relativ gering* sind, da parallele Systeme moderater Größe für die Entwicklung und den Test von Programmen in der Regel ausreichen,
- die Anwendungsmöglichkeiten so vielfältig sind, daß sich auch für kleinere Firmen *zahlreiche Chancen zur Entfaltung und internationalen Etablierung* ergeben,
- im Rahmen der *europäischen Integration* vielfältige Aktivitäten auf dem Gebiet der Parallelverarbeitung initiiert wurden und auch künftig zu erwarten sind.

Um dieses *Entwicklungspotential für Österreich voll zu entfalten*, ist es jedoch dringend erforderlich

- die vorhandenen Forschungs- und Ausbildungsansätze in Österreich zu *koordinieren*, zu konzentrieren und dadurch wirksamer zu gestalten, und
- *signifikante Investitionen* zu tätigen, um rasch eine adäquate Forschungs- und Ausbildungsumgebung für Studenten und junge Wissenschaftler zu schaffen.

Bei dieser Gesamtstrategie ist *internationale Verflechtung* ein entscheidender Faktor:

- Die österreichische Forschungslandschaft im Gebiet der Parallelverarbeitung muß so attraktiv sein, daß *ausländisches Know-How* in Form von Gastforschern, Gastprofessoren, gemeinsamen Projekten, und in Österrreich stattfindenden internationalen Workshops *ständig nach Österreich fließt.*

- *Österreich muß als Studienort für qualifizierte ausländische Studenten* im Bereich Parallelverarbeitung einen internationlen Ruf gewinnen, denn nur so kann wichtiges zusätzliches Forschungs- und Entwicklungspotential für die österreichische Industrie angezogen werden.
- Die österreichische Forschung und Entwicklung auf dem Gebiet der Parallelverarbeitung muß *nach außen ihren hohen Rang dokumentieren*, um internationale industrielle Kooperationen zu ermöglichen und für einschlägige österreichische Produkte (Software, Systeme) Marktchancen zu eröffnen.

## Der bisherige Beitrag Österreichs

Österreichische Forscher haben im In- und Ausland seit langer Zeit **international beachtete Forschungs- und Entwicklungsbeiträge** auf dem Gebiet der parallelen Systeme geleistet. In diesem Zusammenhang seien unter anderem erwähnt

- die Entwicklung einer der ersten *höheren parallelen Programmiersprachen*, der Realzeitsprache "PROGRESS" für die Flugsicherung, in den Jahren 1971-1973 unter der Leitung von H.Zima.
- Entwurf und Entwicklung der parallelen "L-Maschine" und der parallelen "L-Sprache" in den Jahren 1977 bis 1984 am RISC-Linz unter der Leitung von B.Buchberger. Dieses Projekt ist ein Vorläufer des heute weltweit erfolgreichen Transputers und der OCCAM-Sprache. Sowohl die L-Maschine als auch die L-Sprache stellen ein viel allgemeineres Konzept dar als der Transputer und OCCAM.
- Mitarbeit an der Entwicklung der *parallelen Programmiersprache ADA* (Clausen)
- Die Entwicklung von *formalen Methoden zur Beschreibung der Eigenschaften und der Leistung paralleler Programme* (Haring)
- Entwicklung von MACHYS, dem *ersten Time-Sharing System für Analog/Hybrid-Rechner* (Kleinert)
- Mitarbeit an dem multi-nationalen CREST-Projekt AIPS auf dem Gebiet der *Entwicklung von Parallelrechnern* (Kleinert)
- Die Entwicklung des weltweit ersten *Parallelisierungssystems*, "SUPERB", das automatisch sequentielle numerische Algorithmen in parallele Algorithmen für eine Maschine mit verteiltem Speicher transferiert. Dieses System wurde von 1985 bis 1989 im Rahmen des deutschen Superrechnerprojekts SUPRENUM unter der Leitung von H.Zima entwickelt.

Das ACPC strebt an, ein **Forschungspotential zu schaffen, das es erlaubt, auf dem Gebiet der Software für parallele Systeme international konkurrenzfähig an der Spitze mitzuarbeiten.**

## Partner

Um das in Österreich auf dem Gebiet der Parallelverarbeitung vorhandene Potential in Forschung, Lehre und industrieller Kooperation zu verstärken und eine Basis für die dringend notwendige Weiterentwicklung zu etablieren, *haben die folgenden fünf universitären Einrichtungen beschlossen, ihre Aktivitäten zu koordinieren*, einen gemeinsamen Aktionsplan für die Zukunft zu entwerfen und ihn durch das Instrument des ACPC wirksam umzusetzen:

- **RISC-LINZ (Research Institute for Symbolic Computation an der Universität Linz; Leiter: o.Univ. Prof. Dr. Bruno Buchberger).**

  Das Institut arbeitet seit 1978 im Bereich der Parallelverarbeitung.

  - *Arbeitsvorhaben*: Parallele Sprachen für symbolisches Rechnen; Parallelisierung von Algorithmen des Symbolischen Rechnens; parallele Algorithmen für Graphik und CAD/CAM.

- **Universität Wien, Lehrstuhl für Angewandte Informatik (o.Univ. Prof. Dr. Günter Haring).**

  - *Arbeitsvorhaben*: Leistungsbeurteilung von Multiprozessorsystemen; Programmierumgebungen für parallele Systeme; Neuronale Netzwerke.

- **Technische Universität Wien (Dr. Wolfgang Kleinert, Doz. Dr. Christoph Überhuber).**

  - *Arbeitsvorhaben*: Algorithmen und Software für das Gebiet der Numerischen Quadratur und für die Lösung von steifen Anfangswertproblemen gewöhnlicher Differentialgleichungen; Applikationen auf dem Gebiet der Quantenchromodynamik; Neuronale Netze.

- **Universität Wien, Lehrstuhl für Angewandte und Praktische Informatik (o.Univ. Prof. Dr. Hans Zima).**

  Die Gruppe arbeitet seit 1970 im Bereich der Parallelverarbeitung.

  - *Arbeitsvorhaben*: Höhere Programmiersprachen für Numerik; automatische Parallelisierung; Programmierumgebungen für parallele Systeme; Expertensysteme für Programmtransformationen; Neuronale Netze; Supercomputerplanung.

- **Forschungsinstitut für Software-Technologie an der Universität Salzburg (Leiter: o.Univ. Prof. Dr. Peter Zinterhof).**

  - *Arbeitsvorhaben*: Parallelisierung hochdimensionaler Numerik, parallele Simulation technischer Prozesse, Software-Werkzeuge für parallele Numerik.

Das ACPC soll im Stil eines **verteilten Campus** funktionieren. Das heißt, daß es nach außen und in seiner synergetischen Ausrichtung in Forschung und Lehre als *eine Einheit* auftritt, aber geographisch verteilt realisiert ist. Dies erscheint angesichts der guten Verkehrsverbindungen zwischen den Standorten und der Möglichkeiten der Telekommunikation durchaus realistisch. Hier kann man sich *die Erfahrungen amerikanischer Universitäten*, die oft auf sehr viele Einzelstandorte verteilt sind, zunutze machen.

Die intensive Zusammenarbeit der beteiligten Gruppen hat sich bei der Beantragung des Forschungsschwerpunkts "Software für Parallele Systeme" beim FWF, der Definition und Durchführung weiterer gemeinsamer Forschungsprojekte, der Konzeption eines Curriculums für Parallelverarbeitung und bei der Koordination der apparativen Erstausstattung bereits konkret bewährt. Gleichzeitig wurde ein vielfältiges Netz an Kontakten mit wissenschaftlichen Institutionen im Ausland und industriellen Partnern geknüpft.

## Aktionsplan

### Forschung

- Durchführung gemeinsamer Forschungsprojekte (insbesondere Einrichtung eines Forschungsschwerpunkts "Software für Parallele Systeme" beim FWF),
- Einbindung der Aktivitäten von ACPC in europäische (z.B. ESPRIT, EUREKA) und außereuropäische Forschungsprogramme (z.B. NSF Software and Technology Centers, ICOT Institute for 5th Generation Computing, Tokio, International Computer Science Institute, Berkeley),
- Herausgabe einer gemeinsamen Serie technischer Berichte ("ACPC Reports") (Herausgeber: die kollegiale Leitung) und einer Buchreihe,
- Organisation gemeinsamer Workshops, Seminare und Vortragsreihen
- gemeinsame Einladung und Austausch von Gastforschern und Gastvortragenden,
- gegenseitige Beratung, Koordinierung und Unterstützung bei der apparativen Ausstattung,
- Schwerpunktsetzung in der apparativen Ausstattung für die Forschung, um die Duplizierung von Anschaffungen zu vermeiden und Mittel optimal auszunützen.

### Lehre

- gemeinsame Ausarbeitung eines Curriculums,
- Austausch von Lehrveranstaltungen und Skripten,
- Abstimmung bei der Grundausstattung der einzelnen Gruppen für die Lehre,
- Schaffung von Möglichkeiten für Diplom- und Doktoratsstudenten, im Laufe des Studiums die Einrichtungen am Ort der einzelnen Gruppen flexibel zu nutzen.

Das ACPC beabsichtigt nicht die Einrichtung eines eigenen Studienzweiges oder einer eigenen Studienrichtung, sondern bietet vielmehr die Möglichkeit an, *Parallelverarbeitung als Studienschwerpunkt (Diplom, Doktorat) innerhalb verschiedener bestehender Studienrichtungen an den verschiedenen Standorten* zu studieren.

### Industrielle Kooperation

- Stimulierung und Durchführung kooperativer Forschungs- und Entwicklungsprojekte mit der Industrie,
- Beratung und Schulung für außeruniversitäre Einrichtungen,
- Stimulierung von Firmenansiedlungen und -gründungen im Bereich paralleler Systeme (Kooperation mit lokalen Technologiezentren),
- Förderung des Austausches von qualifizierten Forschern und Entwicklern zwischen ACPC und Industrie (Kooperation mit der lokalen Arbeitsmarktverwaltung).

## Zusammenfassung

Durch die bisherigen Leistungen der Partnerinstitutionen des ACPC besitzt Österreich eine gute Ausgangsposition auf dem Gebiet der Parallelverarbeitung. Die Konstituierung des ACPC und der vorgeschlagene Aktions- und Investitionsplan bietet unserer Meinung nach die beste Gewähr dafür, daß Österreich international in diesem wichtigen Bereich langfristig eine gewichtige Rolle spielen wird. Wir sind überdies davon überzeugt, daß die für Österreich *neue Idee des verteilten Campus* dazu geeignet ist, unter sparsamem Einsatz der Mittel das angestrebte Ziel rasch und effizient zu erreichen.

Wir erwarten als Resultat

- eine Stärkung der *internationalen Etablierung der österreichischen Forschung* auf dem Gebiet der Parallelverarbeitung,
- den zügigen Aufbau eines innerösterreichisch und international *attraktiven neuen Schwerpunkt-Curriculums*,
- und einen *gewaltigen Impuls für die österreichische Industrie.*

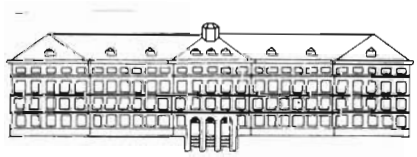
**1. Ankündigung**
**und**
**Einladung zur Vortragsanmeldung**

# 6. Symposium Simulationstechnik

## Technische Universität Wien
## 25. - 27. September 1990

**und**

**User Groups**
**"Simulationssoft- und Hardware"**
**24. September 1990**



WISSENSCHAFTLICHE LANDESAKADEMIE
FÜR NIEDERÖSTERREICH

## Tagungsziel

Ziel des 6. Symposiums SIMULATIONSTECHNIK ist es, den Austausch von Ideen und Erfahrungen von Fachleuten und Interessenten zu fördern, die auf dem Gebiet der Modellbildung und Simulation in Theorie und Praxis tätig sind.

## Veranstalter

ASIM - Fachausschuß 4.5 "Simulation" der Gesellschaft für Informatik (GI),
Technische Universität Wien,
Wissenschaftliche Landesakademie für Niederösterreich

ASIM ist eine Vereinigung, die sich der Förderung und der Weiterentwicklung der Simulation in allen Fachrichtungen widmet.

Mitveranstalter sind IMACS und SCS.

## Programmkomitee

W. Ameling (Aachen), I. Bausch-Gall (München) F. Breitenecker (Wien), H. Fuß (Bonn), H.J. Halin (Zürich), K.H. Heyl (Berlin), G. Kampe (Esslingen), W. Kleinert (Wien), P. Kopacek (Linz), J. Krauth (Stuttgart), A. Kuhn (Dortmund), D. Möller (Lübeck), I. Troch (Wien)

## Tagungsorganisation

Prof.Dr.Felix Breitenecker, Prof.Dr.Inge Troch,
Prof.Dr. Peter Kopacek

## Tagungssekretariat

Irmgard Husinsky, Gabriele Schuster

## Adresse:

Technische Universität Wien
Abt. für Regelungsmathematik, Hybridrechentechnik
und Simulationstechnik
Wiedner Hauptstraße 8-10
A- 1040 Wien
Tel.: A - (0) 222 58801 5387, 5484 oder 5374
Fax: A - (0) 222 587 10 20

---

Ich melde mich zum "6. Symposium SIMULATIONSTECHNIK" an:

Name _____ Vorname _____ Titel _____

Institut/Firma _____

Stadt _____ Postleitzahl _____ Straße,Nr. _____

Telefon _____ Telefax _____

Ich bin Mitglied folgender Organisation (ASIM, GI, IMACS, SCS): Unterschrift:

_____ _____

## Vorläufiges Programm

### Montag 24. September 1990: TU Wien

| | |
|---|---|
| Ab 14 Uhr | User Groups<br>Ausstellung |
| Am Abend | Eröffnungscocktail |

### Dienstag, 25. September 1990: TU Wien

Eröffnung
Hauptvortrag und Vorträge
Ausstellung
ASIM Mitgliederversammlung

### Mittwoch, 26. September 1990: Landesakademie Krems, Wachau

Bustransfer nach Krems
Hauptvortrag und Vorträge
Besichtigung von Krems
Heurigenabend in der Wachau
Bustransfer nach Wien

### Donnerstag, 27. September 1990: TU Wien

Hauptvortrag und Vorträge
Ausstellung

## Tagungsbeitrag

Bei Bezahlung vor dem 1. August 1990:
S 2800.- bzw. DM 400.- (inkl. Tagungsband, gesellschaftliches Programm).
ASIM-, GI-, IMACS- und/oder SCS-Mitglieder S 2450.- bzw. DM 350.- .
Nach dem 1. August 1990 erhöht sich der Betrag um jeweils S 420.- bzw. DM 60.-.
Studenten: S 350.- bzw. DM 50.- (ohne Tagungsband).
Begleitpersonen: S 560.- bzw. DM 80.-.

## Ausstellungen

Während der Tagung wird eine Ausstellung simulationsbezogener Hard- und Software stattfinden.

## Thematische Gliederung

A) MODELLBILDUNG und SIMULATIONSMETHODIK

Modellbeschreibung * Mathematische Verfahren * Modellvalidierung * Optimierungsverfahren * Systemidentifikation * Parallele Algorithmen * Künstliche Intelligenz und Expertensysteme

B) SIMULATIONSWERKZEUGE

Digitale Simulation diskreter, kontinuierlicher und kombinierter Systeme * Simulationsumgebung, Simulationssprachen, Softwareunterstützung, Datenverwaltung * Rechnersysteme, Rechnerarchitekturen * Simulatoren * Simulation auf Analog- und Hybridrechnern

C) ANWENDUNGEN

Ingenieurwissenschaften, einschließlich Robotik, Fahrzeug- und Flugkörperdynamik, Logik- und Schaltkreissimulation, Fertigungstechnik u.a. * Künstliche Intelligenz * Mathematik * Physik * Chemie * Medizin, Biologie, Ökologie, Meteorologie * Verwaltung, Planung * Operations Research, Wirtschafts- und Sozialwissenschaften * Ausbildung

## Treffen von User Groups

finden am 24. September statt,

für ACSL, MATRIX$_X$, XANALOG, MATLAB.

## Termine

Vortragsanmeldung mit 1-2seitiger Zusammenfassung in dreifacher Ausführung bis 15. April 1990.

Mitteilung über Annahme oder Ablehnung des Vortrags durch das Programmkomitee (2 Begutachter) bis 15. Mai 1990.

Endgültige Version der Beiträge (camera ready) bis 30. Juni 1990.

Die Tagungsbeiträge werden in einem Tagungsband veröffentlicht, der zu Beginn der Tagung vorliegen wird.

**Tagungssprache ist Deutsch**

---

Prof.Dr.F. Breitenecker
Abt. Simulationstechnik - E1145
Technische Universität Wien
Wiedner Hauptstraße 8-10
A - 1040 Wien

Ich bin damit einverstanden, daß die umseitig angegebenen Personendaten in der Teilnehmerliste verwendet werden:

☐ ja ☐ Nein

Ich beabsichtige, einen Vortrag zu halten:

ja ☐ Nein ☐

Mein Vortrag fällt in die Gruppe (A, B, C)

Ich nehme am User Group Treffen für _____ teil.

# 1. Transputertag

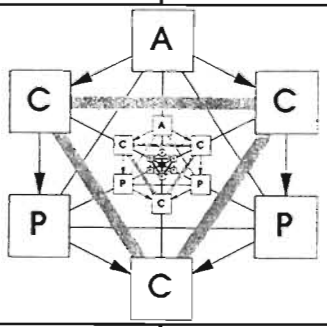
## Workshop über Programmierwerkzeuge für Transputersysteme

**14. Mai 1990**
**10.00 - 18.00 Uhr**

**Technische Universität Wien**
**Gußhausstraße 27-29, 1040 Wien**
**Kontaktraum, 6. Stock**

**Einladung**

Veranstalter:
ACPC (Austrian Center for Parallel Computation)
und
EDV-Zentrum, Technische Universität Wien
Dr. Wolfgang Kleinert




Der Transputertag ist der erste in einer Reihe von Veranstaltungen auf dem Gebiet des Parallelrechnens, die in Österreich in nächster Zeit geplant sind.

Ziel des Workshops ist ein Erfahrungsaustausch der Systemspezialisten aller Gruppen aus Wissenschaft, Forschung und Industrie, die bereits mit Transputern arbeiten. Vor allem die Erfahrungen mit Programmierwerkzeugen wie OCCAM, Helios, Parallel C, Parallel FORTRAN und TDS sollen besprochen werden.

Für Vorträge sind je 30 Minuten mit anschließenden 15 Minuten Diskussion vorgesehen. Anmeldungen werden bis Ostern erbeten.

Anmeldungen und Auskünfte bei:
Dr. Wolfgang Kleinert oder Irmgard Husinsky
EDV-Zentrum, Technische Universität Wien, Wiedner Hauptstraße 8-10, 1040 Wien
Tel.: 0222 58801 5480 oder 5484, Fax: 0222 587 10 20

Bitte abtrennen und einsenden an:

---

Irmgard Husinsky
EDV-Zentrum
Technische Universität Wien
Wiedner Hauptstraße 8-10
A - 1040 Wien

1. Transputertag
Workshop über Programmierwerkzeuge für Transputersysteme

Ich nehme am 14. Mai 1990 am Transputertag teil.

Titel, Vorname, Zuname: _____
Firma, Institut: _____
_____
Adresse: _____
_____
Telefon: _____

Ich möchte einen Vortrag mit folgendem Titel halten:

_____

Datum, Unterschrift: _____

INTERFACE 25 März 1990