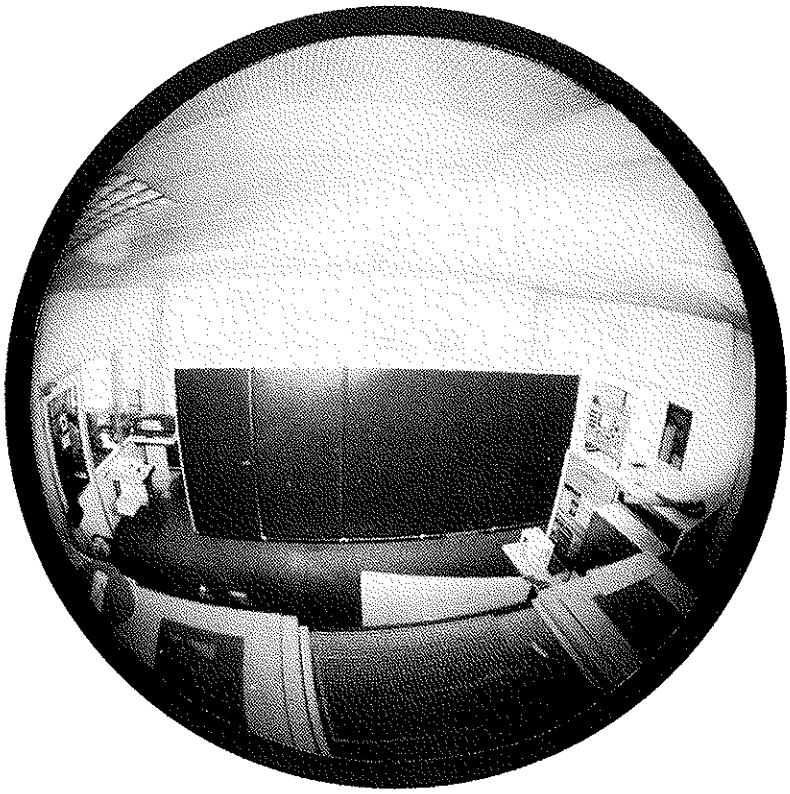
Interface

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

Nummer 22
Juni 1985



Der neue EAI SIMSTAR Multiprozessor
an der Hybridrechenanlage

INHALTSVERZEICHNIS

Redaktion: Irmgard Husinsky
Abt. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien, A-1040 Wien, Gußhausstraße 27-29. Herausgeber, Verleger, Hersteller: EDV-Zentrum der Technischen Universität Wien, Abt. Hybridrechenanlage, Leitung: Dipl.Ing.Dr. W. Kleinert, A-1040 Wien, Gußhausstraße 27-29. Telephon: (0222) 56 01/3706 oder 3669. Ttx: (232) 3222467 = TUW, Modem: (0222) 65 98 17. Druck: Hochschülerschaft Technik, A-1040 Wien, Argentinierstr. 8.

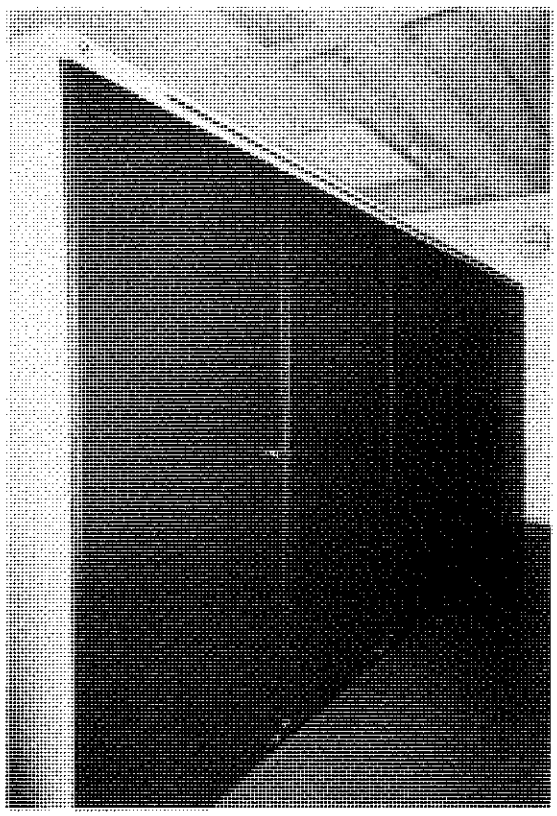
E A I S I M S T A R INSTALLIERT

Nachdem bereits im Dezember 1984 ein erfolgreicher Abnahmetest für das EAI SIMSTAR Simulationssystem im Lieferwerk der Firma Electronic Associates Inc. in West Long Branch, New Jersey, USA, durchgeführt wurde, erfolgte die Lieferung und Installation des SIMSTAR an der Hybridrechenanlage Anfang dieses Jahres.

SIMSTAR ist ein neuartiger, vollständig automatisierter Simulationsrechner, der die neueste analoge VLSI Technologie mit fortgeschrittenem CPU-, Microprocessor- und Controller-Design vereinigt. Er soll den 17 Jahre alten Analogteil der Hybridrechenanlage des EDV-Zentrums an der Technischen Universität Wien ersetzen.

Die bei uns installierte Anlage ist mit den folgenden parallel arbeitenden analogen Rechenmakros bestückt:

28 Integrierer/Track Store/First Order Lag mit variabler Begrenzung
18 Multiplizierer/Dividierer/Quadratwurzel mit variabler Begrenzung
10 Summierer mit 4 Eingängen mit variabler Verstärkung und Begrenzung
18 Zwei-Eingangssummierer mit schaltbaren Eingängen
12 Komparatoren
32 Trunk-Eingänge
64 Trunk-Ausgänge



Am 19. März wurde dann der Leistungstest erfolgreich durchgeführt und die Abnahme vertragsgemäß durch Herrn Ministerialrat Ing. H. Fuchs vorgenommen. Eine Beschreibung der von uns erstellten Leistungstests befindet sich auf Seite 18. Wichtige Aspekte der neuen Hardware werden in dem Artikel "Die SIMSTAR Technologie" auf Seite 5 erläutert.

Nach einer mehrmonatigen Entwicklungs- und Testphase soll die softwaremäßige Integration des SIMSTAR Systems in das Betriebssystem JCS/VS und den hybriden Sprachprozessor HYBSYS bis Ende 1985 abgeschlossen sein. Spätestens Anfang 1986 soll die endgültige Umstellung auf das neue Simulationssystem erfolgen. Die Programmierung wird voll softwarekompatibel von HYBSYS aus erfolgen.

EAI SIMSTAR installed

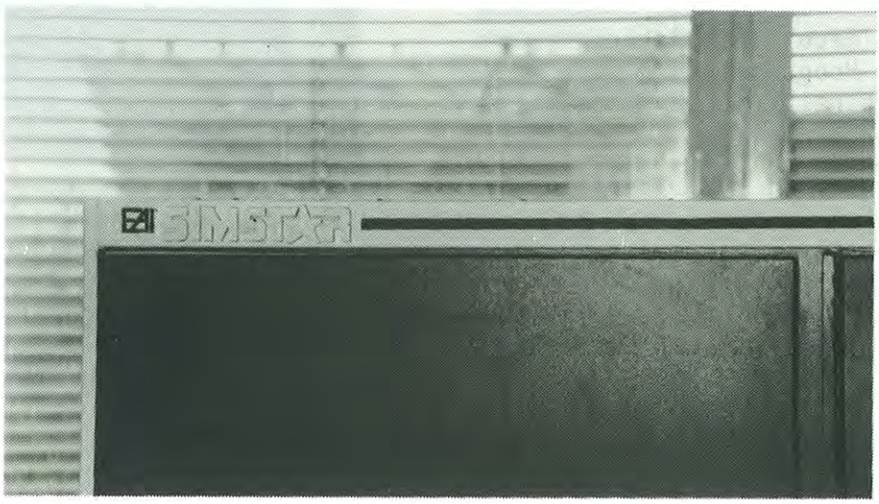
After successful acceptance tests for the EAI SIMSTAR simulation system in December 1984 at EAI's West Long Branch plant, delivery and installation of the SIMSTAR at the Hybrid Computation Centre of the Technical University of Vienna took place at the beginning of this year. The SIMSTAR multiprocessor will be replacing the 17 year-old analog part of the hybrid system.

Our system is equipped with the following parallel analog macros:

28 integrators/track stores/first order lag with variable limit
18 multipliers/dividers/square root with variable limit
10 summers with 4 inputs with variable gain and limit
18 two-input summers with switchable inputs
12 comparators
32 trunk inputs
32 trunk outputs

On March 19, the official acceptance was performed. A description of the benchmarks is given on page 18. Some aspects of the new hardware are described on page 5.

After several months of development and tests, the software integration of the SIMSTAR system into the operating system JCS/VS and the hybrid simulation processor HYBSYS is scheduled to be concluded at the end of 1985. At the beginning of 1986 the new system will be operational for all users. Programming of the new system will be fully software-compatible in HYBSYS.



DIE SIMSTAR TECHNOLOGIE*)

Dieser Artikel beschreibt einige Aspekte der Technologie, die im neuen Simulationsmultiprozessor SIMSTAR von EAI (Electronic Associates Inc.) verwendet wird. Die Implementierung der neuen Entwicklungen auf dem Gebiet der linearen und digitalen integrierten Schaltkreise ermöglichte zusammen mit neuen Techniken beim Entwurf von Subsystem-Schaltungen, der topologischen Anordnung und dem Systemdesign den Bau des SIMSTARs.

Summary

The design of the world's fastest and most advanced simulation computer, SIMSTAR, presented many engineering challenges. This paper describes the new technology employed in EAI's SIMSTAR, an all-new simulation multiprocessor. The latest innovations in implementation of both linear and digital integrated circuits, together with new techniques in subsystem circuit design, packaging, and system design make SIMSTAR possible and practical today.

Die wesentlichen Neuerungen, von denen einige in diesem Artikel vorgestellt werden, sind:

- eine effiziente dreistufige analoge Verbindungsmatrix (CMOS-Implementierung) und ein intelligenter Suchalgorithmus
- eine effiziente Parallel Logic Processing Unit für die schnelle Generierung sequentieller und kombinatorischer Logik
- ein integrierter 32-Bit Digital Arithmetic Processor mit 80 MB Wechselplatte
- ein integrierter 16-Bit Local Control Processor (basierend auf 68000µP) als ein intelligentes Setup und Control Interface
- ein memory-mapped (shared-memory) Interprocessor Digital Interface
- das Konzept der mathematischen Rechenblöcke ermöglicht eine multi-funktionale Verwendung der parallelen Rechenelemente
- systemintegriertes Automated Test Equipment mit automatischer Fehlerdiagnose (sowohl statisch als auch dynamisch) für mathematische Rechenblöcke, Verbindungsmatrix und Parallel Logic Processing Unit
- ein Autobalance System für Zeit- und Temperaturabweichungskontrolle garantiert konsistent hohe Genauigkeit
- ein Auslesesystem für 3000 Punkte (für Diagnostic und Problemlösung) mit einem autoranging ADC und automatischer Offset-Korrektur
- ein aktives Hochqualitäts-Erdungssystem
- bereichserweiterte (pseudo Floating Point) digital setzbare Koeffizienten-einheiten und Multiplizierer
- automatisches Noise/Oscillation Detection System
- systemintegrierte gepufferte Trunks für eine fehlersichere Verbindung mit externen Geräten
- Deglitcher Circuits zur Reduzierung von Transientenfehlern bei schnellen On-line Schaltvorgängen
- aktive feste Funktionsgeneratoren mit großer Bandbreite
- digital setzbare Funktionsgeneratoren mit rascher on-line Funktionsänderung
- Compound Amplifier mit 60MHz Bandbreite und exzellenten DC-Offset- und Drift-Charakteristiken

---

* Überarbeitete Übersetzung von R.W. Embley (EAI West Long Branch): The Technology behind SIMSTAR, an all-new Simulation Multiprocessor. 2. Symposium Simulationstechnik, Wien, September 1984. Informatik-Fachberichte, 85, Springer (1984), 317-327. Bearbeitung: I. Husinsky und W. Kleinert. Fotos von EAI.

Eine effektive dreistufige Verbindungsmatrix (CMOS Implementierung) und ein intelligenter Verbindungssuchalgorithmus

Für den SIMSTAR war ein vollständig automatisiertes Verbindungsmedium erforderlich, um die über 300 parallel arbeitenden linearen Hochleistungs-Rechenelemente und die dazugehörigen Interface-Kanäle zu verbinden. Crosstalk-, Noise- und Phase-Shift-Fehler mußten minimiert werden, um die Verbindungen für das System elektrisch transparent zu gestalten. Dies wurde mit einer elektronischen Schaltmatrix mit hoher Zuverlässigkeit erreicht, wobei ICs aus der Telephon-Schaltnetztechnik verwendet wurden.

Die analoge Verbindungsmatrix ist eine Anordnung von Halbleiterschaltern mit gepufferten Ausgängen, die eine Verbindung von jedem Ausgang eines mathematischen Makros zu jedem Makro-Eingang erlaubt. Die Matrix liefert auch schaltbare analoge Eingangs- und Ausgangssignale zur Verbindung mit anderen SIMSTAR-Konsolen und externen peripheren Geräten. Die analoge Verbindungsmatrix ist mittels eines dreistufigen Clos-Netzwerkes implementiert, das die Anzahl der in einer einstufigen Matrix benötigten Schalter stark reduziert (Bild 1).

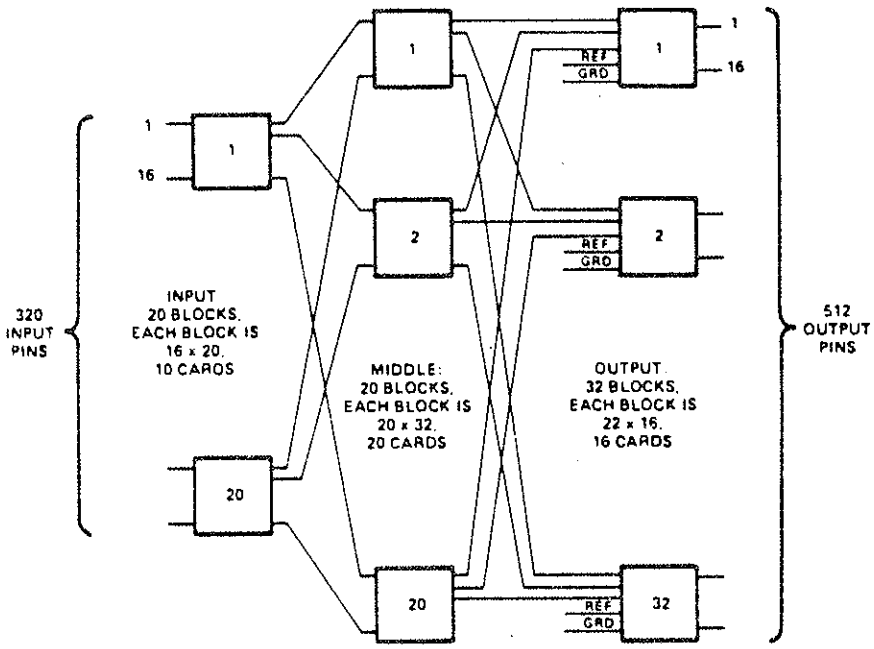


Bild 1
Die dreistufige SIMSTAR Verbindungsmatrix

Dies ist eine 320 x 512 Matrix, die - falls sie als ein einstufiger Crossbar-Switch implementiert werden würde - 163840 Schalter benötigen würde. Durch die Verwendung des dreistufigen Clos-Netzwerkes wird die Anzahl der Schalter auf 29440 reduziert. Diese Implementierung resultiert in einer 320 x 512 analogen Verbindungsmatrix, die, auf 46 Printplatten gepackt, in zwei SIMSTAR Card Files paßt. Auf diesen Karten befindet sich auch ein Auslesesystem für 832 Testpunkte und 320 Diagnostik-Schalter (Bild 2) zur Unterstützung einer automatisierten Fehlerdiagnose. (In dem am Hybridrechenzentrum installierten SIMSTAR ist die Schaltmatrix nur zur Hälfte ausgebaut und erlaubt 160 x 256 Schaltverbindungen.)

Das Herz der analogen Verbindungsmatrix sind hochintegrierte CMOS Schaltkreise. Das verwendete RCA 22100 LSI/CMOS Schalter-Chip vereinigt ein 4x4 Feld von Schaltungspunkten (Transmission Gates) mit einem 4 zu 16 Dekodierer und 16 Latches als Kontrollspeicher. Die CMOS FET Transmission Gates haben eine großzügige Geometrie und 75 Ohm Impedanz, woraus ein geringes Übersprechen zwischen den Switches und eine geringe Phasenverschiebung durch die drei Stufen der Schaltmatrix resultieren.

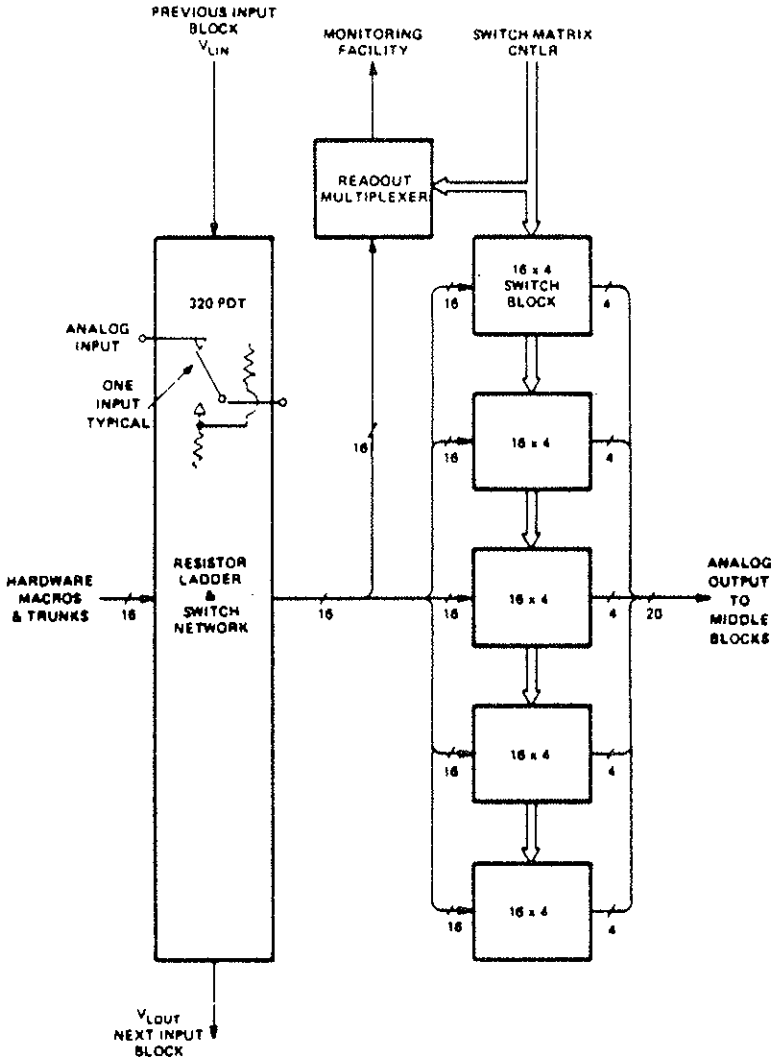


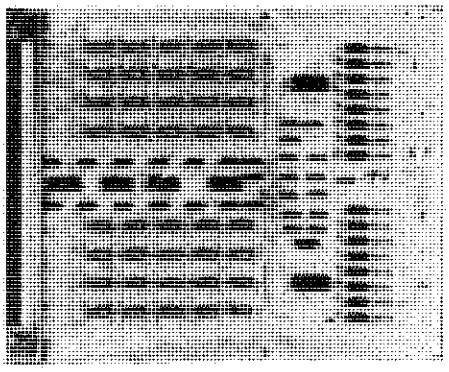
Bild 2
Analoge Verbindungsmatrix mit
integrierten Testspannungen

Von EAI wurde ein intelligenter Suchalgorithmus entwickelt und durch Simulation getestet, der eine optimale Wahl der Mittelblöcke ermöglicht, um eine gewünschte Menge von Eingangs-Ausgangs-Verbindungen ohne Signalblockierung zu realisieren. Dieses sogenannte "Routing"-Problem wurde analysiert und wie folgt gelöst:

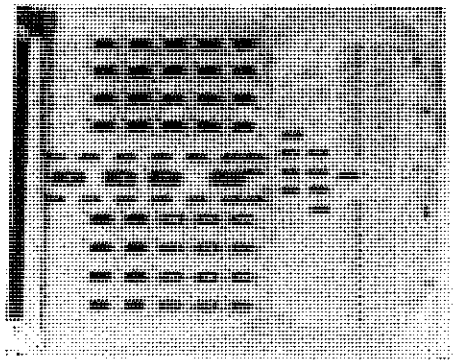
Gegeben sei eine Menge von Verbindungen, die durch die Matrix hergestellt werden sollen. Wie können nun parallele Wege für alle Verbindungen ohne Blockierung (Kurzschlüsse) gefunden werden? Die Lösung für den Fall, bei dem jeder Eingang mit nur einem Ausgang verbunden ist, ist bekannt /1/. Aber für den Fall mit Fanout war kein effektiver Algorithmus bekannt /2,3/. Das Problem gehört zur Klasse, die als NP bekannt ist: Bei einer gegebenen Menge von Wegen ist es leicht, die Blockierungsfreiheit zu überprüfen, aber die große Anzahl der möglichen Wege macht eine erschöpfende Suche unmögich /4/. Für SIMSTAR ist die Anzahl der möglichen Wege $20^{512} \sim 1.3 \times 10^{666}$. Der Exponent zeigt, daß das Problem durch reine Suchverfahren nicht lösbar ist.

Die Lösung erforderte eine Kombination von mathematischen Verfahren und Ingenieurtechniken. Zuerst wurde ein rekursiver Suchalgorithmus mit Baumstruktur entwickelt, der erschöpfend alle Pfade untersuchte. Das funktionierte gut für kleine Matrizen (z.B.: 20x32), erforderte jedoch sehr viel Rechenzeit für größere Systeme. In der "Ingenieurphase" wurden heuristische Methoden entwickelt, um große Blöcke von Lösungen auf einmal zu eliminieren /5/. Diese wurden an Hand von zufällig generierten, extrem schlechten Fällen getestet. Die Originalversion des Suchalgorithmus mit Baumstruktur fand dierichtigen Wege nur

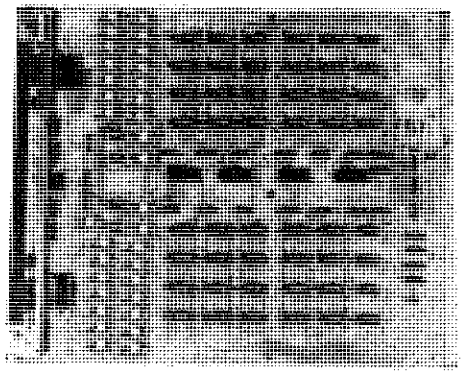
in 50% der Testfälle, selbst wenn sie über zwei Stunden lief. Die Endversion mit komplizierter Heuristik war in allen Fällen erfolgreich. Die mittlere Suchzeit war drei Minuten. Da dies künstlich gewählte, extrem schlechte Fälle waren, kann angenommen werden, daß für reale Anwendungen wesentlich weniger Zeit benötigt wird.



Matrix Input Card



Matrix Middle Card



Matrix Output Card

Eine effektive Parallel Logic Unit (PLU)

Es wurde eine Parallel Logic Unit entwickelt, die die händisch gesteckte parallele Logik der klassischen Hybridsysteme ersetzt. Die PLU ist ein Prozessor, der die Beobachtung und Kontrolle von Echtzeitvorgängen ermöglicht, eine sequentielle und kombinatorische Logik beinhaltet, sowie den Anschluß von externen Signalen erlaubt und in einer höheren Programmiersprache programmierbar ist. In 1µs können bis zu 160 Eingangssignale parallel verarbeitet und die Ergebnisse zu maximal 320 Ausgängen geleitet werden.

Dieses extrem schnelle Ein-Bit-Prozessor-Design verwendet eine Time-Slice-Methode, die die Anzahl der intern benötigten ICs stark reduziert. 80 ICs machen die Arbeit von 1120 im Boolean Function Generator (BFG). 320 Bool'sche Funktionen von vier Variablen werden in jedem Time-Slice generiert. Das resultiert in einem kostensparenden Entwurf, der vollständig in einem SIMSTAR Multiprocessor Card File untergebracht werden kann.

Die Komponenten des Boolean Function Generators (BFG) der PLU sind im Blockdiagramm, Bild 3, dargestellt. Das Memory und die 2:1 MUX-Register bilden eine Schleife, in der Teile des Memorys während jeder der bis zu 14 internen Zeitstufen untersucht werden. Diese Schleife wird iteriert, um die für die Ausgänge notwendigen Signalverzweigungen und die Generierung von Bool'schen Funktionen durchzuführen. Die maximal 14 internen Stufen (jeweils weniger als 100ns) beinhalten eine Stufe zur Synchronisation der Eingänge und eine für die Anfangswerterzeugung und bilden einen kompletten Zyklus. Normalerweise werden die ersten vier Stufen dazu verwendet, die Eingänge zur fünften Stufe zu schalten und die kombinatorische Logik wird in den Stufen fünf bis elf durchgeführt. Die tatsächliche Anzahl der Stufen hängt von der Komplexität der durchzuführenden Logik ab. Die letzten Stufen werden dann dazu verwendet, die BFG-Ausgänge zu den richtigen PLU-Ausgängen zu schalten. Logische Operationen können, für den Fall von Blockierungen oder falls eine erweiterte Funktion benötigt wird, in jeder der Stufen durchgeführt werden. Normalerweise werden alle logischen Funktionen in den Stufen fünf bis sieben erzeugt, wobei die Zykluszeit auf 1.2 Microsekunden reduziert wird.

Das resultierende PLU-Subsystem bringt dem SIMSTAR Multiprocessor System beinahe unerschöpfliche Logikkapazitäten. Falls es in einer Simulationsapplikation benötigt wird, kann das Äquivalent von über 4000 Vier-Input-Gatter zur Verfügung gestellt und in weniger als 2 Microsekunden upgedatet werden.

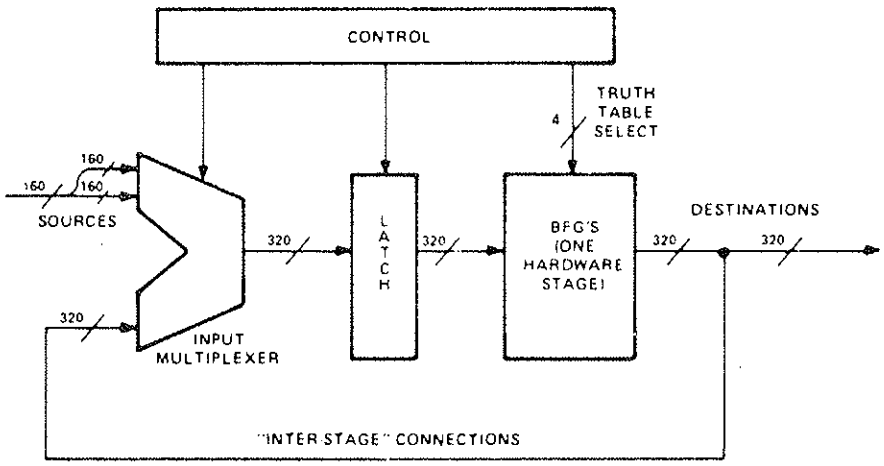
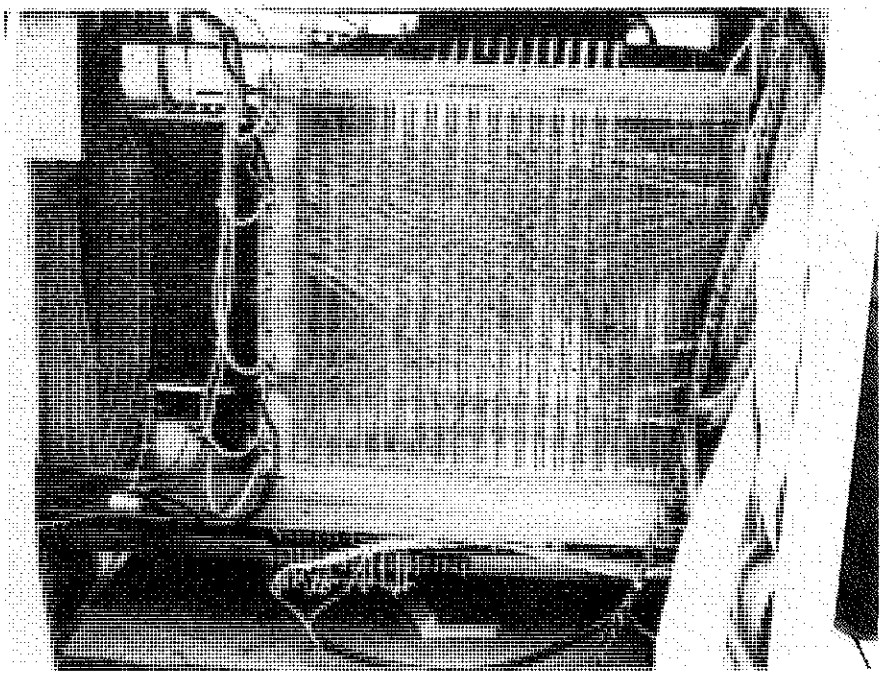


Bild 3
Vielstufiges BFG-Netzwerk mit reduzierter Hardware



Parallel Logic Unit, Backplane

Ein systemintegrierter Digital Arithmetic Processor (DAP)

In den SIMSTAR wurde ein dem neuesten Stand der Technik entsprechender DAP integriert, der eine hocheffiziente ökonomische Simulation der langsameren Prozesse einer Simulation und ein memory-mapped Setzen und Kontrollieren des schnellen Parallel Simulation Processors (PSP)ermöglicht.

Das DAP-Design beinhaltet die folgenden Eigenschaften:

- echte 32-Bit Bit-Slice CPU (Gould 32/27)
- Firmware Single und Double Floating Point Arithmetik
- optionaler Floating Point Accelerator (1.6 - 2.2µs FP)
- bis zu 2 Megabytes Hauptspeicher

- Vector-Interrupts
- Shared Memory mit dem Local Control Processor und dem Host Processor
- Memory Management für Multiprogramming
- 80 MByte Wechselplatte

Ein systemintegrierter Local Control Processor (LCP)

Der LCP dient für Setup, Kontrolle und Wartung des PSP. Der LCP basiert auf einem OMNIBYTE Single Board Computer mit 68000 µP und hat einen 16-Bit Datenbus und einen 24-Bit Adressbus. Ein 256 KB lokales Memory speichert alle Daten, die zum PSP gesendet werden. Komplexe Datenübertragungen vom DAP oder Host-Rechner zum PSP werden vom LCP gesteuert.

Der LCP führt folgende PSP-Funktionen durch:

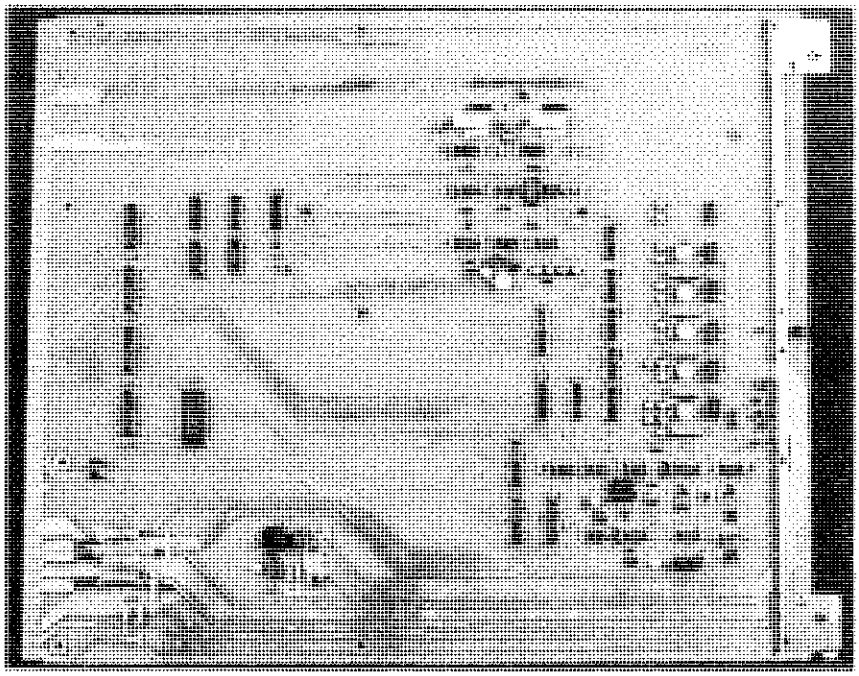
- Initialisierung
- Makro Inventarisierung
- Datenformatumwandlung
- Wartung:
  automatische Diagnose, Autobalance, Messung von Temperatur und Strom, Erkennen von Rauschen und Oszillationen
- Auslesen von Problemlösungen: Auswählen der Multiplexer-Adresse und ADC Gain Ranging

Systemintegrierte automatische Testeinrichtung (ATE)

Aufgrund des einzigartigen Mehrfach-Element-Parallelismus des SIMSTAR war eine ausgeklügelte Testmethode notwendig. Es wurde ein automatisches Testsystem entwickelt, das nicht nur eine einfache Wartung ermöglicht sondern auch das Funktionieren aller wichtigen Hardwarespezifikationen im PSP garantiert. Es werden sowohl Software- als auch Hardwarefehler wie z.B. extrem driftende Operationsverstärker erkannt und der fehlerhafte Teil kann aus dem Inventar der verfügbaren Rechenkomponenten genommen werden, bevor sich Fehler in die Berechnung einschleichen können. Der fehlerhafte Teil wird dann vor dem nächsten Problem Setup automatisch und elektronisch durch einen anderen desselben Typs ersetzt. Es wurden effektive Diagnose-Algorithmen entwickelt, die jeden einzelnen Schaltfehler in der Verbindungsmatrix oder ein einzelnes schlechtes Bit in einem PLU Ram entdecken können. Für den Wartungstechniker werden das Board und der Chip identifiziert. Für eine einfache Wartung und rasche Erkennung bei der Reparatur gibt es LED-Fehlerindikatoren auf allen Macro Computing Boards.

Die in den SIMSTAR integrierten Testeinrichtungen bestehen aus:

- Autoranging ADC (26 Bit Auflösung),
- Auslesesystem mit Multiplexern für 3000 Testpunkte,
- Precision Programmable Gain Device,
- Precision Error Detection Amplifier (EDA),
- Precision Sign Changing Amplifier,
- Peak Error Detector Amplifier,
- programmierbarem frequenz- und amplitudenstabilisierten Oszillator,
- Ein- und Ausgängen der Verbindungsmatrix zur Verschaltung der Unit Under Test (UUT) in der spezifischen Testkonfiguration,
- programmierbaren Präzisionsspannungsquellen (16 Bit),
- Meßeinrichtungen für Betriebstemperatur und Spannungsversorgung,
- Automatic System Power Shutdown Circuit,
- 320 als Eingänge der Verbindungsmatrix schaltbaren Testkonstanten.



Diagnostic Test Unit

Diese Testeinrichtungen ergeben zusammen mit umfangreichen Diagnoseroutinen ein automatisiertes Testsystem. Bild 4 zeigt ein Blockdiagramm des automatisierten Testsystems. Hier ist zum Beispiel ein paralleles Rechenelement (UUT) mit der Diagnostik-Testeinheit (DTU) über die analoge Verbindungsmatrix verbunden. Die im AC-Mode gezeigte DTU liefert ein sinusförmiges Testsignal (programmierbare Frequenz) an den UUT Input und einen Eingang des Fehlererkennungsverstärkers (EDA). Der andere Input des EDA wird über den Precision Sign Changing Amplifier und die Gain Device zum UUT über die Verbindungsmatrix verbunden. Dann wird der Ausgang der UUT mit ihrem Eingang verglichen, wobei eventuelle Vorzeichen- oder Verstärkungsänderungen durch die DTU korrigiert werden. Am Ausgang des EDA wird ein Fehlersignal erzeugt, das direkt proportional zum Fehler in der UUT ist. Der Peak Detector erfaßt den Total Instantaneous Dynamic Error (TIDE), die Vektorsumme eines Phasenverschiebungsfehlers und des Amplitudenfehlers. Das Ausgangssignal des Peak Detectors wird durch den ADC digitalisiert, zum LCP gesendet, wo es mit vordefinierten Spezifikationsgrenzen verglichen wird, und die Ergebnisse werden in den Wartungsfehlerfile eingetragen.

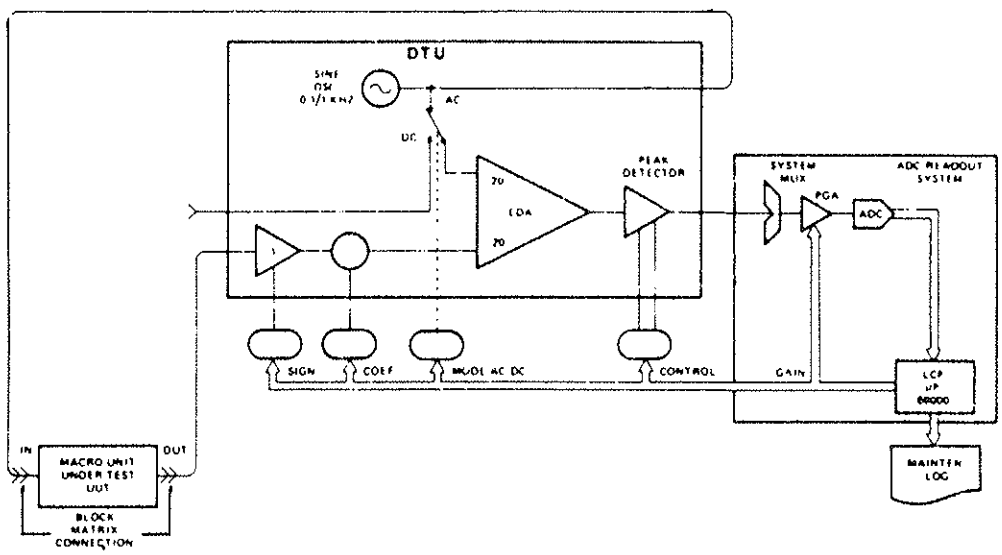


Bild 4
Blockdiagramm des automatisierten Testsystems
(im Macro AC Test Mode (0.1/1 KHz TIDE))

Ein Autobalance System für die Korrektur von Zeit- und Temperaturabweichungen

Für den SIMSTAR wurde ein automatisches Hilfsmittel zur elektronischen Nullabgleichung des eventuell in kritischen Operationsverstärkerschaltkreisen vorhandenen Offset-Fehlers errichtet. Dieses Autobalance System eliminiert komplett die zeitraubende Testzeit des manuellen Nullabgleichs und die Simulationsungenauigkeiten, die bei den Vorgängersystemen durch driftende Operationsverstärker verursacht wurden. Es werden nicht nur alle kritischen Verstärker nach einer bestimmten Zeit und nach Temperaturveränderungen abgeglichen (auf Null ±5 Microvolt), sondern es wird auch ein Protokoll über das Ausmaß des Abgleichs für einen gegebenen Verstärker gemacht und, falls vorgegebene Grenzen überschritten werden, wird der Verstärker als ein "exzessiver Abweicher" gekennzeichnet, der dann ausgetauscht werden kann, bevor er die Genauigkeit der Simulation beeinträchtigt.

Autobalance wird immer nach dem Laden eines Problems durchgeführt, nach einer Änderung in der Makro-Konfiguration oder auf Anforderung des DAP. Autobalance wird (falls nicht vom Benützer ausgeschaltet) nach dem Abspeichern eines Problems durchgeführt, falls nach dem letzten Autobalance die Temperaturdifferrenz 1°C überschreitet (die Temperatur wird mit vier Präzisions-Halbleiter-Sensoren alle 10 Minuten gemessen), und falls das letzte Autobalance vor mehr als 8 Stunden stattgefunden hat. Autobalance wird nie während des Run-Modes von SIMSTAR durchgeführt.

Die Autobalance Hardware besteht aus über 600 8-Bit Korrektur-DACs. Jeder von ihnen ist mit kritischen Operationsverstärkerschaltkreisen verbunden (siehe auch Bild 5). Vereinfachend kann der Autobalance-Algorithmus wie folgt verstanden werden:

1. Anwählen des UUT Outputs über das Readout-System.
2. Nullsetzen aller Outputs der Verbindungsmatrix.
3. Setzen des Korrektur-DAC auf Null-Output.
4. LCP empfängt und zeichnet den Output-Offset auf.
5. Laden des Korrektur DACs mit dem entsprechend abgeglichenen Wert.
6. Wiederholung der Schritte 4 und 5.

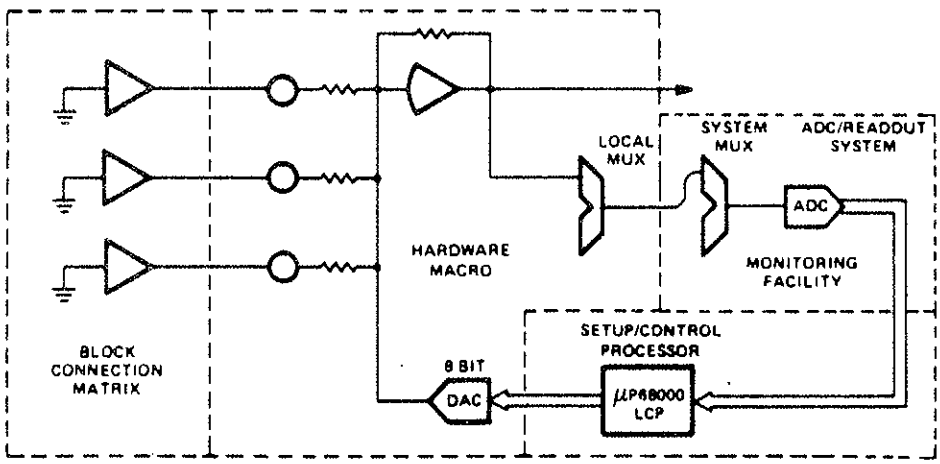


Bild 5
Systemblockdiagramm des Autobalance Systems

Ein aktives Erdungssystem mit allerhöchster Präzision

Es wurde ein Erdungssystem von hoher Qualität für den SIMSTAR benötigt, das in Hinblick auf den zentralen System-Erdungspunkt extrem kleine Potentialdifferenzen (Mikrovolt) zwischen allen 200 analogen Rechenmakros erhält. Bild 6 zeigt ein Diagramm des "Star Point"-Erdungssystems im SIMSTAR, das Erdungsschleifen eliminiert. Aber IR-Abfälle in den verschiedenen Erdverteilungskabeln produzieren immer noch nicht-tolerierbare DC Offsets. Im SIMSTAR wurde eine einmalige Lösung für dieses Problem implementiert. Es wurde ein aktives System entwickelt, das den Erdspannungsfluß, der von jedem Makro fließt, um 4 Größenordnungen reduziert, wobei der proportionale IR-Abfall in den Verteilungsdrähten im wesentlichen eliminiert wird (siehe Bild 7a). Dadurch werden definitiv Fehlerverbreitungen durch die Erdung minimiert.

Der aktive Erdungsschaltkreis (AGC, siehe Bild 7b) besteht aus einem Operationsverstärker mit sehr kleinem Drift, der als Spannungsfolger beschaltet ist, um eine Spannungsquelle mit geringer Impedanz zu produzieren, deren Ausgangs-Potential auf Nullspannung gehalten wird (virtuelle Erde). Daraus folgt, daß der normalerweise von und zum Erdungspunkt eines Rechenelements fließende Strom in der Größenordnung von zig Milliampere zu den unempfindlichen ±15V Bussen geleitet wird, und die Hochqualitätserde nicht betroffen wird.

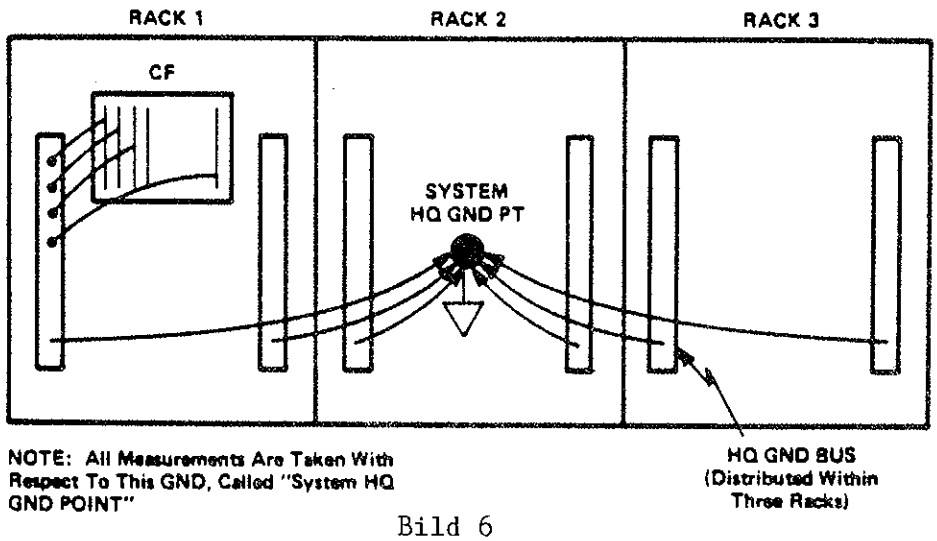


Bild 6
Star Point-Erdungssystem

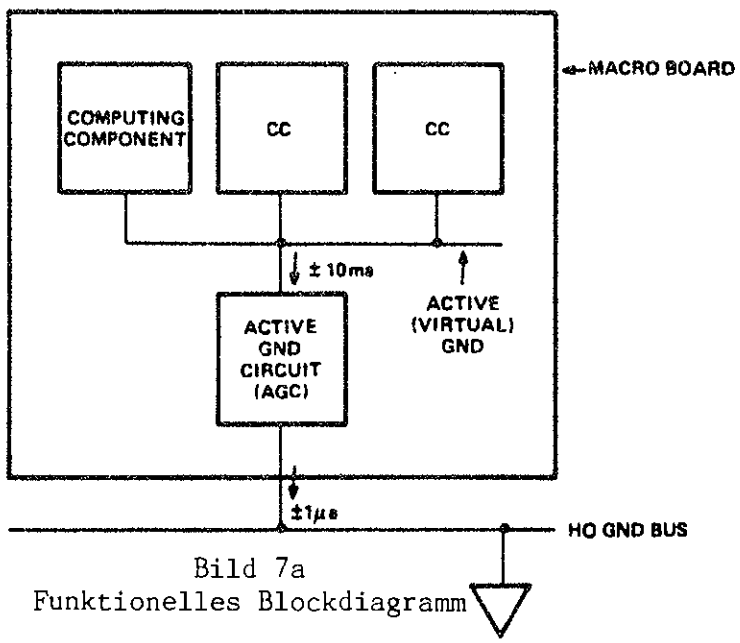


Bild 7a
Funktionelles Blockdiagramm

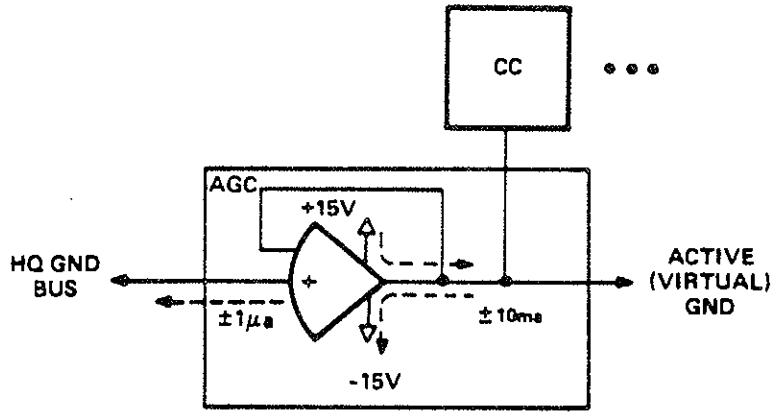


Bild 7b
Vereinfachtes Schema des aktiven Erdungsschaltkreises

Bild 7
Aktives Hochqualitäts-Erdungssystem

Bereichserweiterte (pseudo Floating Point) digital setzbare Koeffizienteneinheiten und Multiplikationselemente

Gegenüber den Vorgängersystemen wurde beim SIMSTAR eine Verbesserung um einen Faktor 10 im Verwendungsbereich realisiert, indem Schaltungen zur automatischen Verstärkungsänderung (lokale Reskalierung) innerhalb jeder digital setzbaren Koeffizienteneinheit (DSCU) und der analogen Multiplikationselemente eingebaut wurden.

Das bereichserweiterte DSCU-Design (siehe Bild 8) bringt sowohl Verbesserungen in der Genauigkeit als auch in der Auflösung für kleine Koeffizientenwerte (kleiner als 1/4). Die DSCU-Funktionen verwenden, zusammen mit dem LCP, die signifikanten Bits des MDAC, sogar für das Setzen von kleinen Werten. Diese Autoranging- oder Autoscaling-Technik resultiert in einem effektiven Gesamtbereich von 18 Bit (+ Vorzeichen) mit 16 und 18 Bit Auflösung bei Werten unter 1/4 bzw. 1/16. Die Genauigkeit wird um einen Faktor 4 bzw. 16 verbessert, was tendentiell zu einem konstanten relativen (statt wie bisher absoluten) Ausgangsfehler über den gesamten Wertebereich führt.

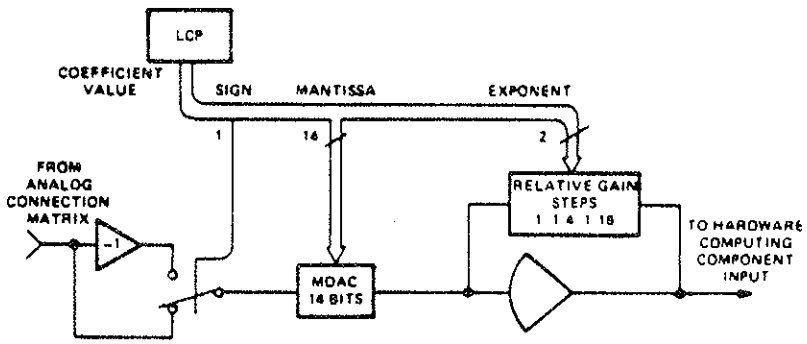


Bild 8
Vereinfachtes Blockschema einer bereichserweiterten digital setzbaren Koeffizienteneinheit (mit konstantem relativem Ausgangsfehler)

Durchführung des Autoranging (LCP-Funktionen mit DSCU):

Ist ein spezieller DSCU-Koeffizientenwert kleiner als 1/4, aber größer als 1/16, wird das Koeffizientenwort um zwei Bit nach links geschoben und der Exponent wird auf ein 1/4 verändert, wobei der Koeffizientenwert beibehalten wird. Fehler am Ausgang werden jedoch um das Vierfache reduziert. Eine ähnliche Umwandlung wird vorgenommen, falls der Koeffizientenwert kleiner als 1/16 ist.

Das Design der bereichserweiterten Multiplizierer bietet, ähnlich wie die DSCU, große Verbesserungen bei der Genauigkeit bei kleinen Ausgangssignalen. Da $(1/4)^2$ Multiplizierer immer einen absoluten Fehler haben, wird der relative Fehler für kleine Ausgänge sehr groß. Dieses Multiplizierer-Design reduziert hingegen diesen inhärenten Nachteil, indem automatisch das interne Multiplizierersignal umgeschaltet oder reskaliert wird, sodaß ein konstanter relativer Fehler angestrebt wird. Fensterkomparatoren innerhalb des Multiplizierers entdecken, wann entweder X oder Y oder X und Y unter 1/4 der Referenz absinken. Sobald dies passiert, werden die X- und Y-Signale auf vollen Bereich verstärkt und der Multiplizierausgang wird um ein 1/4 bzw. 1/16 verringert, was den Fehler um ebensoviel vermindert. Bild 9a zeigt ein Blockdiagramm und Bild 9b die Operationsgleichungen eines bereichserweiterten Multiplizierers. Bild 10 zeigt die relative Fehlerreduktion in der X-Y-Ebene bei konventionellen Multiplizierern. Bild 11 zeigt das verbesserte Fehlerverhalten gegenüber konventionellen Multiplizierern und Bild 12 die Verbesserung beim Quadrieren.

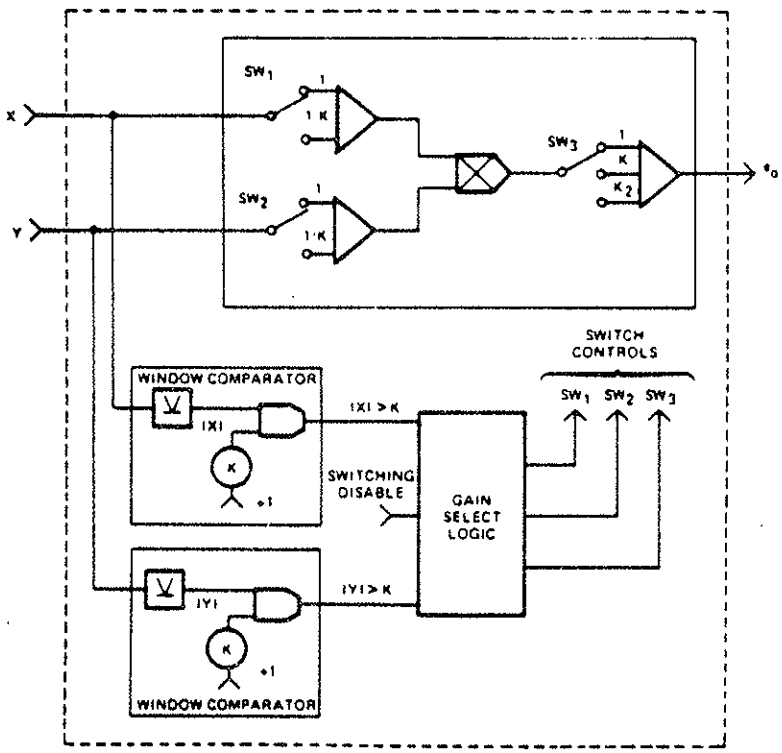


Bild 9a
Blockdiagramm des
bereichserweiterten Multiplizierers

for:

$\|X\|$ AND $\|Y\| > K$	$e_o = XY \pm \epsilon$
$\|X\|$ OR $\|Y\| \leq K$	$e_o = Y\left[\frac{X}{K}\right]K \pm K\epsilon$
	OR
	$e_o = X\left[\frac{Y}{K}\right]K \pm K\epsilon$
$\|X\|$ AND $\|Y\| \leq K$	$e_o = \left[\frac{X}{K}\right]\left[\frac{Y}{K}\right]K^2 \pm K^2\epsilon$

$0 < K \leq 1$, $K = 1/4$ in Simstar design

Bild 9b
Operationsgleichungen des
bereichserweiterten Multiplizierers

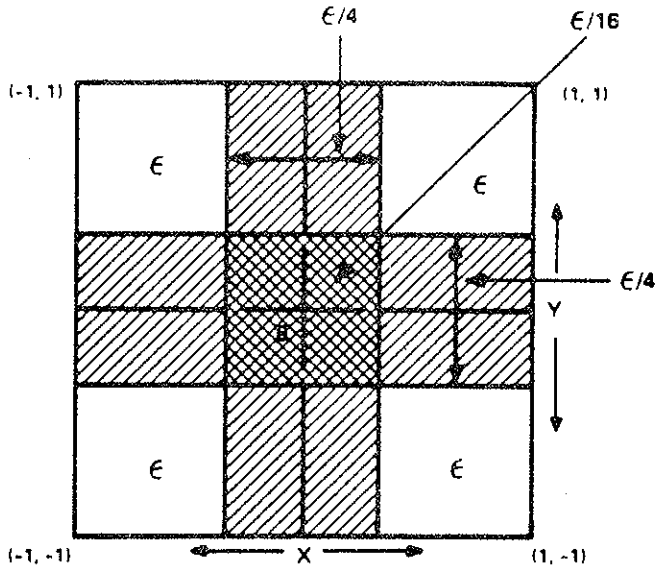


Bild 10
Fehlerreduktion des bereichserweiterten Multiplizierers

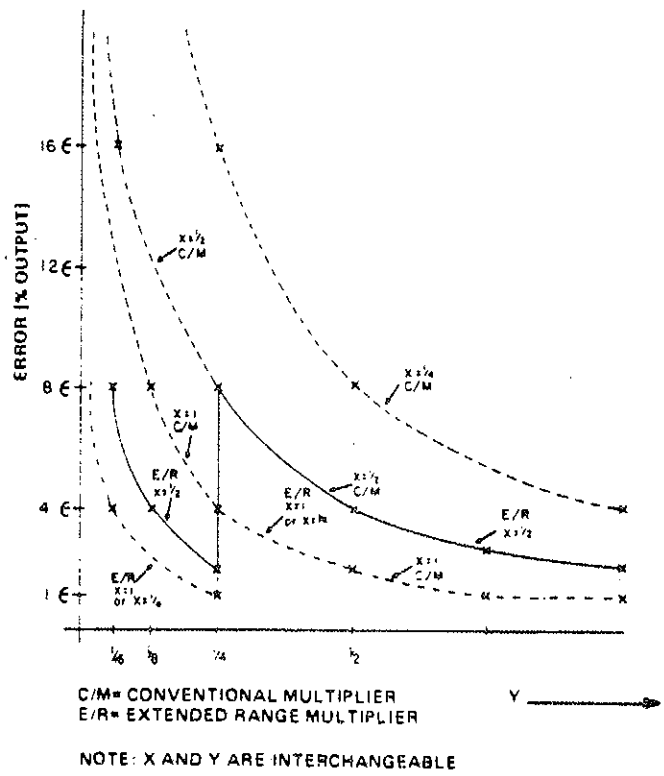


Bild 11
Fehlerverhalten des bereichserweiterten Multiplizierers

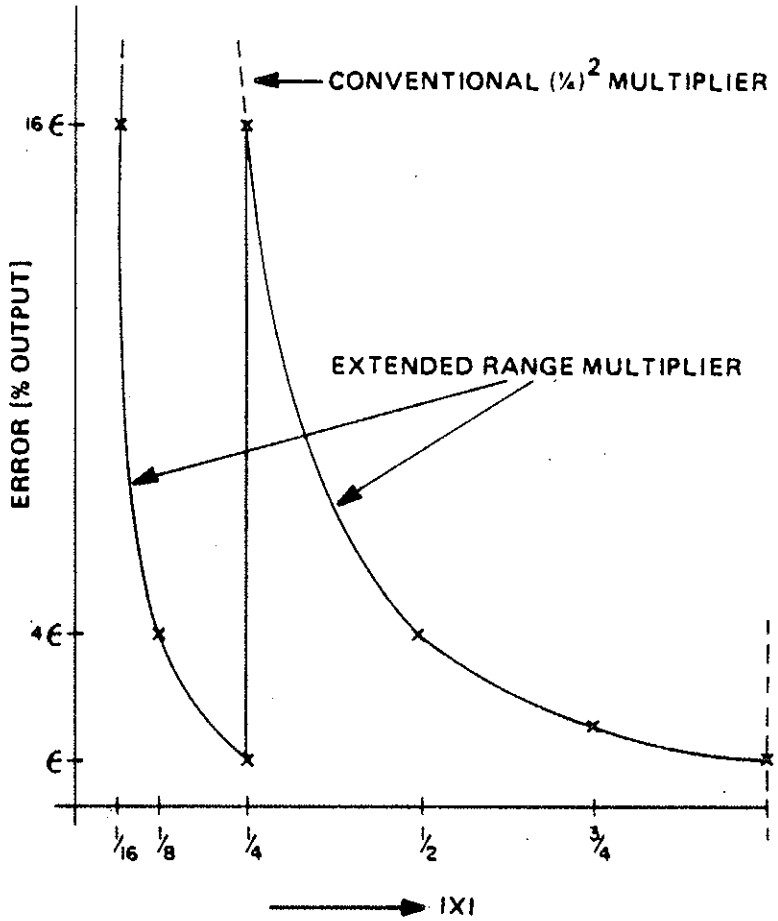


Bild 12
Fehlerverhalten des bereichserweiterten Multiplizierers
im Quadriermode

Zusammenfassung

Acht der grundlegend neuen technologischen Entwicklungen für EAIs neuen SIMSTAR Multiprozessor wurden präsentiert. Jede einzelne kann als eigenständige technische Neuerung gesehen werden. Diese und die anderen am Anfang erwähnten, aber nicht näher diskutierten Neuentwicklungen bilden die Basis für ein State-of-the-Art Simulationssystem, das von vielen Forschungsstellen in der ganzen Welt verwendet werden kann.

Literatur

1. Benes, V.E.: Mathematical Theory of Connecting Networks and Telephone Traffic. Academic Press, New York (1965)
2. Hannauer, G.: Stored Program Concept for Analog Computers. Final Report, NASA Project #NAS8-21228 (June 1968)
3. Hannauer, G. and Asthana, A.: Recent Advances in Automatic Patching Software. Proceedings of Special Symposium on Advanced Hybrid Computing, San Francisco (July 1975)
4. Gary, M.R. and Johnson, D.S.: Computers and Intractability. W.H. Freeman and Company, San Francisco (1979)
5. Wirth, N.: Algorithms + Data Structures = Programs. Prentiss-Hall, Englewood Cliffs, New Jersey (1976)

SIMSTAR Leistungstests

W. Kleinert

Als Teil der Abnahmeprozedur des neuen EAI SIMSTAR Multiprozessor Systems wurden im Leistungstest fünf vom Hybridrechenzentrum erstellte Testprogramme am SIMSTAR gerechnet und mit den Ergebnissen am hybriden Time-Sharing System MACHYS sowie mit den theoretischen oder numerischen Lösungen verglichen.

Bei der Auswahl dieser Testprogramme ließen wir uns vor allem von unserer jahrelangen Erfahrung mit spezifischen Problemen des Analogrechnens leiten. Dabei standen artifizielle Fragestellungen mit bekannten mathematischen Lösungen zur Beurteilung der Genauigkeit des neuen Analogrechners im Vordergrund. Nur eines der fünf Testprogramme stellt einen Anwendungsfall eines Benutzers (Simulation) dar, der am MACHYS-System nicht zufriedenstellend gelöst werden konnte.

Im folgenden werden diese Testprogramme kurz beschrieben und die Resultate diskutiert. Da zum Zeitpunkt des Abnahmetests am SIMSTAR noch keine graphische Ausgabe in Form von Plotterzeichnungen zur Verfügung stand, konnten die Ergebnisse nur auf einem 6-Kanal-Schreiber dokumentiert werden. Die bei den Testbeispielen gegenübergestellten MACHYS-Lösungen wurden mit HYBSYS erstellt und werden in einem äquivalenten Maßstab dargestellt.

Testbeispiel 1

Gewöhnliche lineare Differentialgleichung der Ordnung 22 mit konstanten Koeffizienten

$$Y^{(22)} + Y = 0$$

Eine einfache Analyse zeigt, daß bei geeigneten Anfangsbedingungen $Y = \sin(t)$ eine Lösung darstellt. Das für die Modellbeschreibung am Analogrechner erforderliche System von Differentialgleichungen 1. Ordnung lautet:

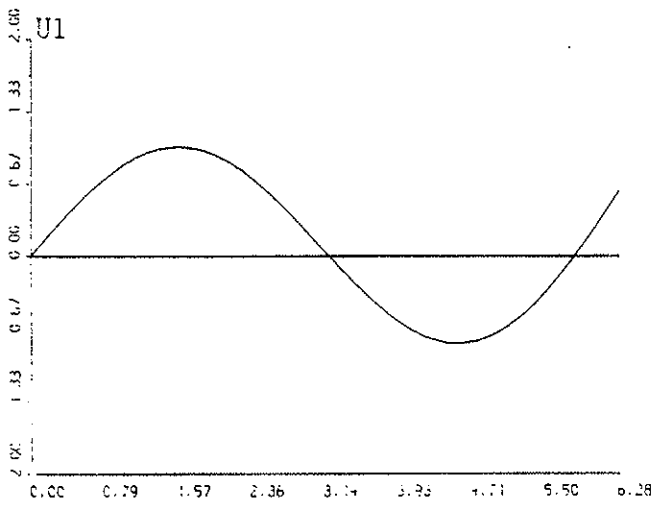
$$U_i' = V_i \qquad i=1,\ldots,11$$

$$V_1' = -U_{11} \qquad V_1(0) = 1$$

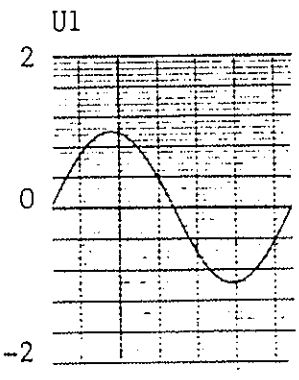
$$V_i' = U_{i-1} \qquad V_i(0) = -1^{(i-1)} \qquad i=2,\ldots,11$$

Bei den gegebenen Anfangsbedingungen sind die $U_i \pm \sin(t)$ die $V_i \pm \cos(t)$. In dieser Notation ist die Bandstruktur des Systems deutlich zu erkennen. Mit diesem Beispiel kann sehr schön der Einfluß der Schaltmatrix auf die Genauigkeit studiert werden.

MACHYS



SIMSTAR



Die SIMSTAR-Lösung gibt den Sinus recht genau wieder, während die Phasenverschiebung beim alten System beträchtlich ist.

Testbeispiel 2

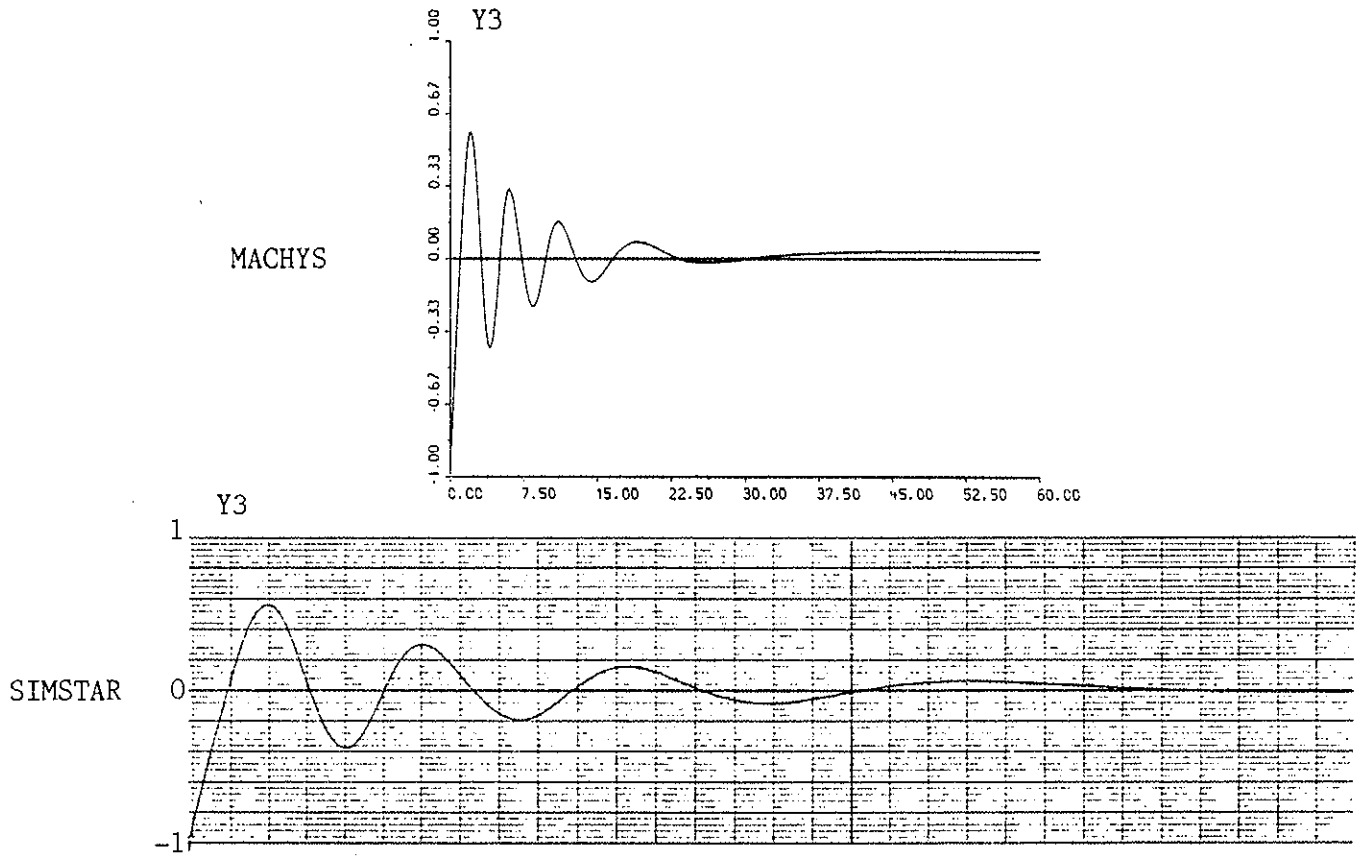
Nichtlineares System der Ordnung 10 mit asymptotisch stabilem Ursprung.

$$Y_1' = -0.1\ Y_1 + Y_{10}\ Y_2 \qquad Y_1(0) = 1$$

$$Y_i' = -0.1\ Y_i + Y_{i-1}\ Y_{i+1} \qquad Y_i(0) = -1 \qquad i=2,\ldots,9$$

$$Y_{10}' = -0.1\ Y_{10} + Y_9\ Y_1 \qquad Y_{10}(0) = -1$$

Dieses Beispiel hat wieder Bandstruktur und testet zusätzlich den Einfluß der Multiplizierer.



Während die SIMSTAR-Lösung richtigerweise asymptotisch gegen die Ruhelage strebt (wobei der Trajektorienverlauf auch sehr gut mit dem Ergebnis einer numerischen Integration mit Hilfe eines Runge-Kutta-Fehlberg-Algorithmus der Ordnung 5 übereinstimmt), schwingt die MACHYS-Lösung auf einen falschen Wert ein.

Testbeispiel 3

Einfache Integratorkette der Länge 22 mit Eingang Null

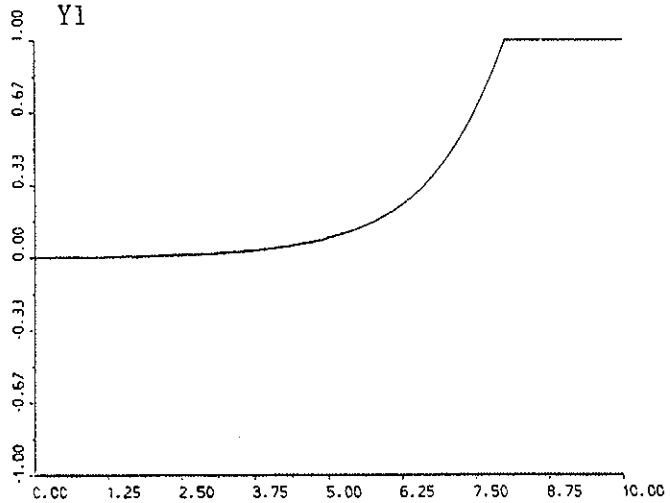
$$Y_i' = Y_{i+1} \qquad i=1,\ldots,21$$

$$Y_{22}' = 0$$

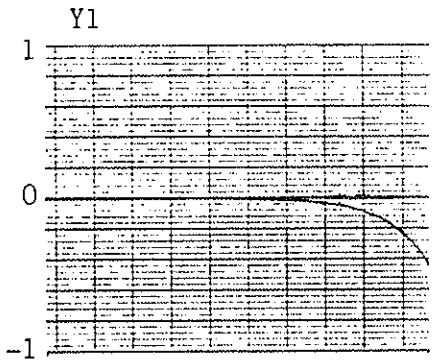
$$Y_i(0) = 0 \qquad i=1,\ldots,22$$

Da die allgemeine Lösung dieser Gleichung ein Polynom vom Grad 21 ist, dessen Koeffizienten bei den gegebenen Anfangsbedingungen aber alle gleich Null sind, ist leicht einzusehen, daß schon kleinste Störungen zu beträchtlichen Abweichungen am Ende der Integriererkette führen müssen. Während bei gleicher Skalierung der letzte Integrierer in MACHYS bereits in die Übersteuerung geht, ist das SIMSTAR-Ergebnis um Größenordnungen besser.

MACHYS



SIMSTAR



Testbeispiel 4

Gewöhnliche linerae Differentialgleichung mit konstanten Koeffizienten der Ordnung 11 mit -1 als elffachen Eigenwert.

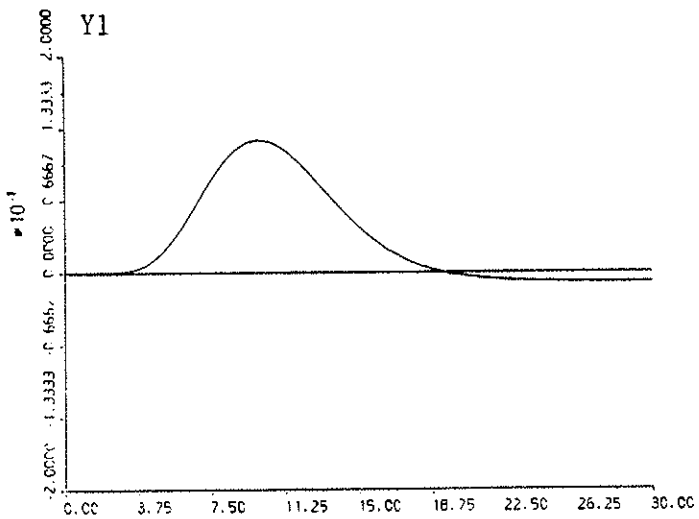
$$(\frac{d}{dt} + 1)^{11} Y = 0$$

$$y^{(i)} = 0 \qquad i=0,\ldots,9$$

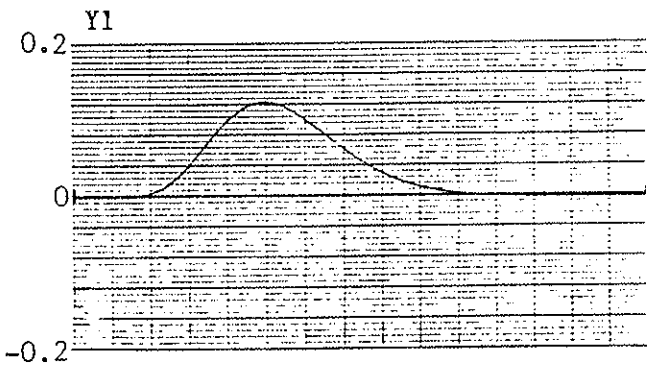
$$y^{(10)} = 1$$

Die Lösung ist proportional $t^{10}e^{-t}$ und sollte für $t \to \infty$ gegen Null streben.

MACHYS



SIMSTAR



Wieder stimmt die SIMSTAR-Lösung weit besser mit der theoretischen Lösung überein.

Testbeispiel 5

Hochoszillatorisches Simulationsproblem aus der quantentheoretischen Beschreibung von Laserphänomenen, das uns freundlicherweise von Herrn Dr. J. Seke, Institut für Theoretische Physik, zur Verfügung gestellt wurde.

$$Y_1' = -WG\ Y_2$$

$$Y_2' = WG\ Y_1 + 4N\ Y_3\ (Y_2 - OMEGA\ Y_1)$$

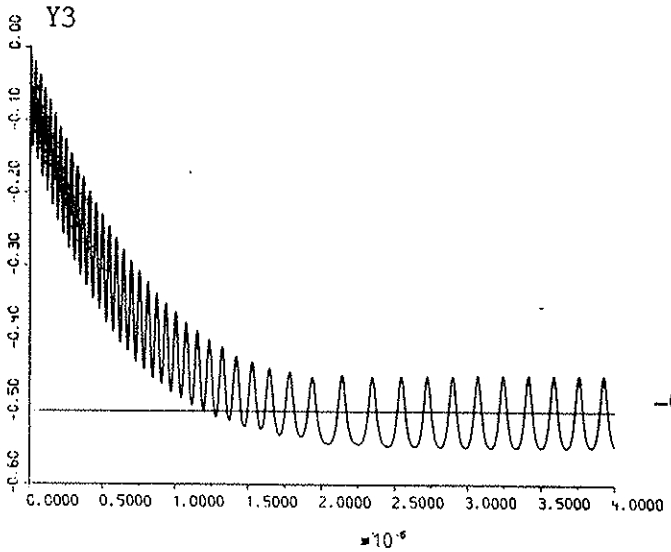
$$Y_3' = -N/2 + 2N\ (Y_3^2 + Y_1^2 - Y_2^2 + OMEGA\ Y_1\ Y_2)$$

$Y_1(0) = 0.5$ $\qquad$ $WG = 10^8$
$Y_2(0) = 0$ $\qquad$ $N = 10^6$
$Y_3(0) = 0$ $\qquad$ $OMEGA = -50$

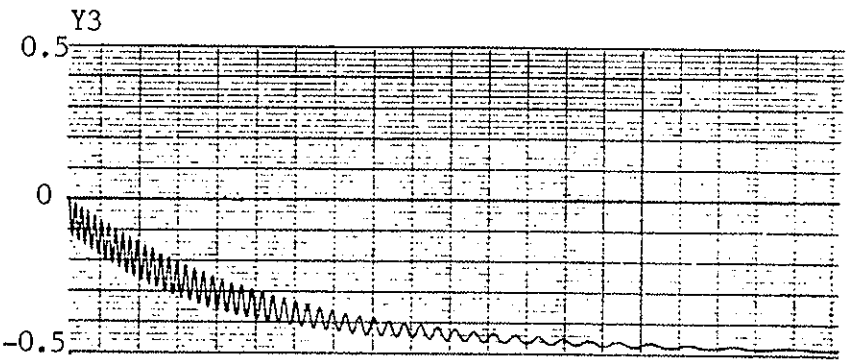
$0 \leq t \leq 4*10^{-6}$ (benötigt Zeittransformation zur Darstellung am Analogrechner, $\beta = 10^8$)

Die Analysis des Problems zeigt, daß (0,0,-0.5) ein stationärer Punkt dieses Systems ist, wobei aber der gegebene Wert von OMEGA = -50 den kritischen Wert für die Stabilität dieses Gleichgewichtspunkts darstellt.

MACHYS



SIMSTAR



Beim SIMSTAR-System wird dieser stationäre Punkt der Theorie entsprechend erreicht, während die MACHYS-Lösung so fehlerbehaftet ist, daß statt des stationären Punktes nur der benachbarte Grenzzyklus erreicht wird.

Das Ergebnis dieses Vergleichs von extremen Testprogrammen zeigt die überlegene Genauigkeit des neuen Systems, während das MACHYS-System sich bezüglich seiner Rechengenauigkeit wie andere klassische Analogrechner verhält.

aktuelle mitteilungen

Simulationssoftware

Das Hybridrechenzentrum ist bemüht, neben der Software zum Betreiben der bei uns installierten Hybridrechner auch andere Simulationswerkzeuge auf verschiedenen Rechenanlagen des EDV-Zentrums den Benützern zur Verfügung zu stellen.

Es stehen derzeit folgende Simulationssprachen und -pakete zur Verfügung:

* HYBSYS:
  - interaktive hybride Simulationssprache, entwickelt am Hybridrechenzentrum der Technischen Universität Wien
  - implementiert auf EAI PACER 600 und ab Ende 1985 auf EAI SIMSTAR

* ACSL:
  - Advanced Continuous Simulation Language (Mitchell & Gauthier Inc., Concord, Massachusets, USA)
  - sehr weit verbreitete kommerzielle Simulationssprache zur Simulation kontinuierlicher Systeme mit Möglichkeiten zum Event-Handling
  - implementiert an der CYBER 720 und ab Ende 1985 auch am EAI SIMSTAR

* GASP:
  - General Activity Simulation Program für kontinuierliche und diskrete und "gemischte" Simulation
  - bestehend aus GASP IV, einem Simulationspaket für diskrete Simulation (Pritsker & Ass. Inc., West Lafayette, Indiana, USA) und GASP V, einer Ergänzung und Weiterentwicklung der ETH Zürich (F.E. Cellier, Institut für Automatisierungstechnik) für kontinuierliche Simulation (Sammlung von FORTRAN-Unterprogrammen, Rechnerprogrammen, Postprozessoren, Runtime- Interpreter)
  - implementiert auf VAX 780

* DARE-Interactive:
  - interpreter-orientierte Simulationssprache für kontinuierliche Prozesse (G.A. Korn, University of Arizona, USA)
  - implementiert auf VAX 780

Geplant ist die Beschaffung und Implementierung folgender Software:

* FORSIM:
  - Simulationspaket für kontinuierliche Prozesse, die insbesondere durch partielle Differentialgleichungen beschrieben werden (M.B. Carver, Atomic Energy of Canada Ltd.)
  - Implementation geplant auf CYBER 720

* DYNAMO:
  - Simulationssprache für diskrete Modelle (Gesellschaft für Mathematik und Datenverarbeitung, Bonn), Forresters Weltmodelle werden üblicherweise günstig mit DYNAMO simuliert
  - Implementation geplant auf PC (16 Bit) am Hybridrechenzentrum und am Institut für Technische Mathematik mit MS-DOS

Implementierungsfragen und einzelne Benutzungsbewilligungen kann Herr Hummer (Hybridrechenzentrum, Klappe 3704) klären bzw. erteilen. Prinzipielle Benutzungsfragen (Rechnerzugriff, etc.) kann Dr. Kleinert, der Leiter des Hybridrechenzentrums (Klappe 3702) beantworten. Für Informationen bezüglich Anwendungen steht neben Herrn Dr. Kleinert und Herrn Hummer (und Herrn Dipl.Ing. Solar für HYBSYS, Klappe 3978) auch Herr Doz. Breitenecker von der Abteilung für Regelungsmathematik und Hybridrechen- und Simulationstechnik am Institut für Technische Mathematik (Klappe 5374) zur Verfügung.

Es besteht die Möglichkeit, Implementationsunterstützung bei anderer einschlägiger Simulationssoftware zu organisieren.

Einführung in die digitale Simulationssprache ACSL
Neuerungen und zusätzliche Features von ACSL
28.10. - 29.10. 1985

Anfragen: Doz.Dr.F. Breitenecker (Klappe 5374), H. Hummer (Klappe 3704).

Folgende Kurse werden an der Hybridrechenanlage nach Vereinbarung kurzfristig abgehalten (ab 3 Teilnehmern):

Benützung der Betriebssysteme JCS/VS und MPX

Hinweise für FORTRAN-Programmierer an der Hybridrechenanlage

Einführung in das hybride SIMSTAR-System

Bedienung des hybriden Prozessors HYBSYS zur Simulation dynamischer Systeme am SIMSTAR-System, Teil I

Bedienung des hybriden Prozessors HYBSYS zur Simulation dynamischer Systeme am SIMSTAR-System, Teil II

Programmentwicklung am Terminal

Weiters wird auf die einschlägigen Lehrveranstaltungen der Studienrichtung Technische Mathematik verwiesen.

Nähere Auskünfte und Anmeldungen zu den Kursen telephonisch oder persönlich bei Herrn M. Schandl (1040 Wien, Gußhausstraße 27-29, 4. Stock, Zimmer CD0442, Tel.: 5601 / 3706 oder 3669 DW).

Kurse der Abteilung "Regelungsmathematik und Hybridrechen- und Simulationstechnik" des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik:

Simulationen an den Klein-Analogrechnern EAI 1000
20.1. - 21.1. 1986

Einführung in die hybride Simulationssprache HYBSYS
22.1. - 24.1. 1986

HYBSYS für Fortgeschrittene
27.1. - 28.1. 1986

Einführung in die digitale Simulationssprache ACSL
(zur allgemeinen Verwendung an CYBER implementiert)
29.1. - 31.1. 1986

Auskünfte: Doz.Dr. F. Breitenecker, Dr. F. Rattay (Klappen 5374, 5379, 5387).

Neu im Betriebssystem JCS/VS 8

F. Blöser

System Line Printer

Als Standard Line Printer im Betriebssystem JCS/VS 8 ist jetzt der B600 Drucker von Dataproducts, der schon bisher als zweiter Drucker zur Verfügung stand, angeschlossen. Dieser Drucker kann 132 Spalten drucken und wird mit der FORTRAN-Einheitennummer 15 angesprochen. Die SPOOLING-Option ist für die Ausgabe auf diesem Drucker nicht mehr notwendig.

Unterbrechung von Listausgabe am Terminal

Die Ausgabe von Listen auf einem Terminal kann jetzt durch Eintippen eines beliebigen Zeichens angehalten werden. Die Ausgabe wird nach dem Eintippen eines weiteren Zeichens (ungleich Break) fortgesetzt, durch ein Break jedoch abgebrochen. Dies gilt auch für die Listen von Systemprozessoren (FORTRAN-Compiler, Core Image Generator), die mit der ASSIGN-Option auf ein Terminal verlegt wurden. In jedem Fall muß dazu aber auf der betreffenden /RUN- bzw. Prozessorsteuerline die TERMINAL-Option angeführt sein.

Aufruf des FORTRAN 77-Compilers JCSFTU

Der an der Hybridrechenanlage entwickelte FORTRAN 77-Compiler JCSFTU kann jetzt mit der Steuerline "/FTU" bzw. "/FTU nameob" aufgerufen werden, wobei nameob der gewünschte Name des erzeugten Object Files ist. Eventuelle Options können daran anschließend, durch Beistriche getrennt, angeführt werden.

Erweiterung der virtuellen Character

Die virtuellen Character, die eine komfortable Benützung der verschiedenen Möglichkeiten der an die Hybridrechenanlage angeschlossenen Terminals gestatten, wurden erweitert. Alle Character von '0 bis '237 werden jetzt als virtuelle Character behandelt. Unter den neuen Zeichen befinden sich u.a. deutsche Zeichen (Umlaute, scharfes ß), mathematische Symbole, logische Operatoren, weitere Semigraphik-Zeichen (runde Ecken, Pfeile) und visuelle Attribute (inkl. Farbauswahl) sowie Terminal Control-, Cursor Control- und Edit Control-Zeichen. Außerdem wurde die Anzahl der Zeichen, die in Balkenschrift ausgegeben werden können, auf den gesamten möglichen Bereich von '0 bis '377 ausgedehnt. Es können daher jetzt etwa auch Kleinbuchstaben, Umlaute und Semigraphik-Zeichen in Balkenschrift mit der Routine GIANT ausgegeben werden.

Rechnen mit dem Floating Point Processor (FPP)

Die Run-Time-Library JCSRTL enthält nun die Versionen der Systemroutinen, die ohne FPP rechnen. Beim Binden eines Programms entfällt daher die Loadline für den File NONFPP. Benutzer, die mit dem FPP rechnen wollen, müssen am Beginn ihres Programms die Routine FPP (ohne Parameter) aufrufen, außerdem ist die Loadline "LOAD,D9 FPPRTL" erforderlich. Diese Loadline muß auch beim Binden eines HYBSYS-Overlays verwendet werden, da HYBSYS nun in einer FPP-Version bereitgestellt wird.

Neuer Source Editor JCSSED

Der neue Source Editor JCSSED (siehe Seite 36) ermöglicht das bildschirmorientierte Bearbeiten von Source Files. Da dazu aber bestimmte Edit-Funktionen am Terminal notwendig sind (Character Insert/Delete, Line Insert/Delete, Cursorabfrage und Cursorpositionierung), kann der Editor JCSSED nur auf jenen Terminals, die über diese Möglichkeiten verfügen, gerechnet werden.

Listen von Source Files

Source File-Listen am Line Printer können mit dem Programm LISTEN, das vom District 50 geladen werden muß, erzeugt werden. Der zu listende Source File ist über die INPUT-Option anzugeben (/RUN LISTEN,LOAD=50,INPUT:nameso=dist). Mit der Option FORCE erhält man eine Liste mit Linenummern (wie bei LIST im SOP).

Optimierung in HYBSYS

F. Breitenecker
Institut für Technische Mathematik, TU Wien

Zweck einer Simulation ist es, das Verhalten eines (dynamischen) Systems durch Rechnersimulation eines mathematischen Modells des Systems zu untersuchen. Variation von Modellparametern gibt dabei einen tiefen Einblick in die charakteristischen Phänomene des Systems. Dabei stellt sich dann sofort die Frage nach optimalen Parameterwerten, die bestimmte Systemgrößen (Gütemaße) minimieren.

Die Simulation selbst wird durch Simulationssprachen wesentlich erleichtert. Eine zusätzliche Optimierung wird dabei allerdings nur spärlich unterstützt. Auf der anderen Seite schenkt Optimierungssoftware der auftretenden speziellen und komplizierten Form der Nebenbedingungen (System von Differentialgleichungen) keine Beachtung. Ist man prinzipiell auch am zeitlichen Verhalten des Systems und nicht nur an der Optimierung einzelner Parameter interessiert, so sind Optimierungsmöglichkeiten in Simulationssprachen der alleinigen Verwendung von Optimierungspaketen vorzuziehen.

Wesentlichster Befehl in einer Simulationssprache ist jener, der einen Simulationslauf startet. Er ist üblicherweise der einzige "aktive" Befehl zur Modelluntersuchung. Nach /3/ ist jedoch ein Simulationslauf als eine der Methoden zur Modelluntersuchung anzusehen und damit gleichberechtigt zu anderen Methoden zur Modelluntersuchung wie Eigenwertanalyse, Frequenzanalyse und Optimierung. Solche zusätzlichen Methoden sollen sowohl in der Simulationssprache bereits enthalten als auch vom Benützer in geeigneter Weise definierbar sein. Nach /3/ und /4/ ist diese Forderung am ehesten in interpreter-orientierten Simulationssprachen erfüllbar, während sie in compiler-orientierten Sprachen schwer zu erfüllen ist.

Compiler-orientierte Sprachen müssen alle aktiven Berechnungen (zusätzliche Methoden zur Modelluntersuchung) üblicherweise in ein Programm mit der Modellbeschreibung zusammenfassen, das dann von einem Runtime-Interpreter gestartet wird (/1/). Damit muß z.B. eine Optimierung in der Modellbeschreibung programmiert werden (/4/). Moderne Simulationssprachen wie ACSL bieten hier die Möglichkeit an, das vom Precompiler der Simulationssprache erzeugte FORTRAN-Hauptprogramm in einen selbst programmierbaren Optimierungsrahmen einzubetten (/1/, /2/). Interpreter-orientierte Sprachen dagegen haben hier - neben den bekannten Nachteilen eines Interpreters - den großen Vorteil, daß die Modellbeschreibung in Form einer Modelldatenbasis vorliegt, die von beliebigen Ebenen (Interpretersprache, Modellsprache, Basissprache) aus exekutiert werden und mit beliebigen anderen Berechnungen (Methoden zur Modelluntersuchung) verbunden werden kann (/3/, /4/).

HYBSYS bietet als interpreter-orientierte Simulationssprache nun einen weiteren Vorteil für die Definition zusätzlicher Methoden zur Modelluntersuchung an: jedem HYBSYS-Befehl, auch einem modelldefinierenden, entspricht ein äquivalentes FORTRAN-Unterprogramm, das in einem HYBSYS-Overlay aufgerufen werden kann (/7/). Damit kann in einem Overlay (nach /3/ ein Makro auf der Methodenebene), der ja einen neuen HYBSYS-Befehl darstellt, all das programmiert werden, was in HYBSYS interaktiv getan werden kann: Simulationslauf durchführen, Modell definieren (ändern), Parameter ändern, etc. Zu erwähnen ist, daß einige Basisbefehle von HYBSYS selbst bereits Overlays sind.

Diese Möglichkeiten erlauben nun eine effektive Implementierung der Optimierung in Form einer Methode zur Modelluntersuchung: ein Overlay (=Makro) ruft ein Optimierungsprogramm einer Library auf, das selbst zur Auswertung der Gütefunktionen wiederholt Simulationsläufe durchführt; dabei bildet die HYBSYS-Modelldatenbasis die Schnittstelle für die Übergabe von Parameterwerten zwischen dem Programmteil, der die Auswertung der Gütefunktionen durchführt, und dem, der dann einen Simulationslauf startet. Abbildung 1 zeigt den Ablauf in einem Optimierungs-Makro.

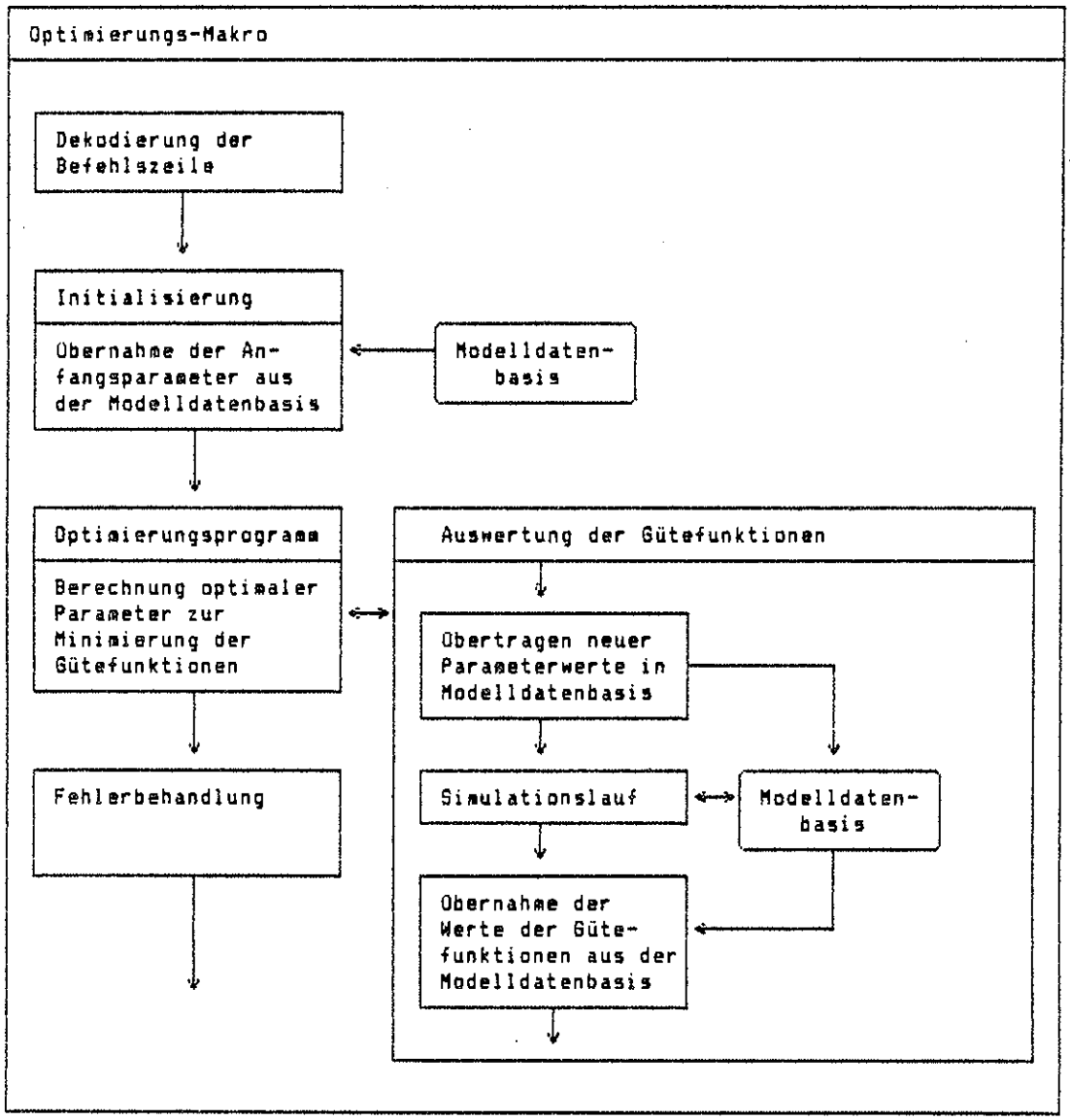


Abb. 1: Ablauf in einem HYBSYS-Makro zur Optimierung

Erwähnt sei, daß HYBSYS auch komfortable Unterstützungssoftware für die Dekodierung der Befehlszeile, die den Makro aufruft, und für den Datentransfer von und zur Modelldatenbasis zur Verfügung stellt. Das Programmieren derartiger Makros basiert derzeit auf FORTRAN 77 und ist eingehend beschrieben in /7/ und /8/.

Standardmäßig bietet HYBSYS dem Benutzer zwei Makros zur Optimierung an, die nach dem erwähnten Schema programmiert sind (/8/):

Der Makro

ZERO f1,...,fn BY p1,...,pn

minimiert die Gütefunktionen f1,...,fn bezüglich der Parameter p1,...,pn durch Ermitteln der Nullstellen von f1,...,fn mit einem Newton-Powell-Algorithmus. Die Parameter p1,...,pn sind dabei "unabhängige" Parameter, die Variablen f1,...,fn, die die Werte der Gütefunktionen darstellen, müssen als ADC-Variable (Endwertfeststellung) definiert werden, da ja dem Optimierungsprogramm der Endwert der Variablen des letzten Simulationslaufes (=Wert der Gütefunktionen) geliefert werden muß.

Der zweite Makro stellt verschiedene Algorithmen zur direkten Optimierung zur Verfügung:

OPTIM,method f1,...,fm BY p1[...],...,pn[...]

Für die Parameter p1,...,pn und die Werte der Gütefunktionen f1,...,fm gelten dieselben Voraussetzungen wie beim Makro (Overlay) ZERO. Der Befehlsparameter "method" gibt die gewünschte Optimierungsmethode an. Zur Wahl stehen die folgenden Methoden (/6/):

CYCVAR ..... zyklische Parametervariation
RANDOM ..... Zufallssuche
ROTCOR ..... Suche nach orthogonalen Richtungen
GRADNT ..... einfaches Gradientenverfahren
CONGRD ..... Verfahren der konjugierten Gradienten
VARMET ..... Variable-Metrik-Verfahren

In den Klammern nach den Parameternamen können Bereiche für den Parameter angegeben werden (CYCVAR, RANDOM, GRADNT) bzw. müssen die in der HYBSYS-Modellbeschreibung definierten Namen für die partiellen Ableitungen der Gütefunktion(en) nach den Parametern angegeben werden.

Die ersten drei Verfahren sind reine Suchverfahren, der n-dimensionale Parameterraum wird abgetastet. CYCVAR, ein lineares Suchverfahren, variiert die Parameter "zyklisch", d.h. auf einer Koordinatenachse (des Parameterraumes) wird versucht, eine Verbesserung der Gütefunktion durch geeignete Variation eines Parameters (Verdoppeln und Halbieren der Schrittweite) zu erreichen; im nächsten "Zyklus" wird dasselbe mit der nächsten Achse versucht. RANDOM wählt nach einer Gleichverteilung "zufällige" Parametervektoren des Parameterraumes aus. ROTCOR ist ein wesentlich verbessertes lineares Suchverfahren, das in jedem Iterationsschritt n lineare Suchen in n orthogonalen Richtungen durchführt; je Iteration wird neu orthogonalisiert (Verfahren von Gram-Schmidt), die lineare Suche auf den Achsen beruht auf einer Modifikation des Inter- bzw. Extrapolationsverfahrens von Powell.

Die letzten drei Verfahren sind sogenannte "Abstiegsmethoden", d.h. sie benötigen neben dem Wert der Zielfunktion auch die "Richtung des stärksten Abstiegs der Zielfunktion" (die partiellen Ableitungen der Zielfunktion nach den Parametern = Gradient), um einen neuen Parametervektor verwenden zu können; diese Verfahren arbeiten viel effizienter, benötigen aber die erwähnten partiellen Ableitungen.

GRADNT ist ein einfaches Gradientenverfahren, das die lineare Suche in Richtung des stärksten Abstiegs (negative Gradientenrichtung) durchführt. CONGRD sucht pro Iteration in n "konjugierten" Abstiegsrichtungen, die durch Orthogonalisierung nach der negativen Gradientenrichtung gewonnen werden. VARMET, ein Variable-Metrik-Verfahren, verwendet zusätzlich zur Gradientenrichtung auch

die Hess'sche Matrix, die geeignet approximiert wird und ist damit ein Verfahren zweiter Ordnung.

Gemeinsam ist diesen drei Verfahren, daß der Gradient vom Benutzer zur Verfügung gestellt werden muß, d.h. zusätzlich zur Systemdynamik müssen in HYBSYS auch die den Gradienten bestimmenden Differentialgleichungsysteme programmiert werden. Damit ist die Einsatzmöglichkeit dieser effizient arbeitenden Verfahren einerseits durch die Beschreibungsmöglichkeit des Gradienten durch ein Differentialgleichungssystem (was nur in speziellen Fällen möglich ist) und durch die begrenzte Elementanzahl des Analogrechners (die Gradientenbildung verdoppelt zumindest die Schaltung) beschränkt.

Das folgende Beispiel demonstriert die Problemaufbereitung für die Anwendung von OPTIM. Sollen in der Differentialgleichung (Regelkreis)

$$\ddot{x} + 2\dot{x} + x + (ax+b\dot{x}+c\int x) = 0, \quad x(0)=1, \ \dot{x}(0) = 0$$

die Parameter a, b und c bezüglich der Gütefunktion

$$f(x,t;a,b,c) = \int_0^\infty x^2(t)dt \qquad \rightarrow \ \min$$

optimiert werden, so müssen die partiellen Ableitungen von f nach a,b und c bestimmt werden, wenn mit einem Abstiegsverfahren optimiert werden soll. Diese sind

$$fa = 2\int_0^\infty x.x_a dt, \ fb = 2\int_0^\infty x.x_b dt, \quad fc = 2\int_0^\infty x.x_c dt$$

wobei die partiellen Ableitungen x_a, x_b, x_c den Differentialgleichungen

$$\ddot{x}_a + (2+b)\dot{x}_a + (1+a)x_a + c\int x_a dt + x = 0, \ x_a(0)=0, \ \dot{x}_a(0)=0$$

$$\ddot{x}_b + (2+b)\dot{x}_b + (1+a)x_b + c\int x_b dt + \dot{x} = 0, \ x_b(0)=0, \ \dot{x}_b(0)=0$$

$$\ddot{x}_c + (2+b)\dot{x}_c + (1+a)x_c + c\int x_c dt + \int x dt = 0, \ x_c(0)=0, \ \dot{x}_c(0)=0$$

gehorchen. Diese müssen nun zusätzlich zur Systemdynamik in HYBSYS definiert werden. Die folgende Tabelle zeigt die Ergebnisse der Optimierung mit den verschiedenen Verfahren, wobei die Genauigkeitsschranke für alle mit 0.01 vorgegeben wurde und alle mit denselben Startwerten für die Parameter a,b,c arbeiteten:

Methode	Iterationen	Funktionsauswertungen
CYCVAR	6	132
RANDOM	49	184
ROTCOR	16	118
GRADNT	9	80
CONGRD	6	60
VARMET	6	65

Betrachtet man die Anzahl der Funktionsauswertungen (Anzahl von Simulationsläufen) als Kriterium der Effizienz, so lassen sich folgende Aussagen aus diesem und anderen Testbeispielen treffen:

* Die Abstiegsverfahren GRADNT, CONGRD und VARMET arbeiten wesentlich effizienter als die Suchverfahren.

* Bei den Abstiegsverfahren empfiehlt sich die Verwendung von CONGRD oder VARMET.

* Bei den Suchverfahren empfiehlt sich die Verwendung von ROTCOR. Das Verfahren RANDOM schneidet bei allen Testbeispielen am schlechtesten ab.

Prinzipiell sind bei der Wahl des Optimierungsverfahrens folgende Punkte zu beachten:

* Jedes Verfahren benötigt "gute" Startwerte, die nicht notwendigerweise beim selben Problem bei allen Verfahren gleich sind.

* Bei den Abstiegsverfahren sollte der Schaltaufwand für die Erzeugung der benötigten partiellen Ableitungen und die höhere Effizienz gegenüber den Suchverfahren mit der geringeren Effizienz aber ohne zusätzlichen Aufwand abgewogen werden.

* OPTIM führt, wenn nötig, eine Nachskalierung durch. Zusätzlich empfiehlt es sich, nach einer (gelungenen) Optimierung (optimal) zu skalieren und die Optimierung fortzusetzen, was in den meisten Fällen eine Verbesserung bringt.

Schließlich sei bemerkt, daß die analoge Integration in HYBSYS sich bei Optimierungsaufgaben nicht nur wegen der relativ kurzen und problemunabhängigen Rechenzeit für einen Simulationslauf als vorteilhaft herausstellt: werden Parameter optimiert, die Unstetigkeiten erzeugen und verändern, so ergeben sich für die Simulation wegen der echt parallelen Arbeitsweise keinerlei (numerische) Schwierigkeiten. Im Gegensatz dazu müssen digitale Integrationsalgorithmen erheblichen Aufwand treiben, um derartige zustandsbedingte Unstetigkeiten (state events) mit der Integrationsschrittweite zu synchronisieren (/4/).

Abschließend sei noch auf eine an Bedeutung zunehmende Klasse von Optimierungsproblemen hingewiesen. In letzter Zeit gewinnen auch Probleme der Vektoroptimierung (Polyoptimierung) an Bedeutung (/5/). Dabei treten Gütefunktionen f1,...,fn auf, die nicht vergleichbar und kontradiktorisch sind, d.h. die Verbesserung einer Gütefunktion bringt automatisch die Verschlechterung einer anderen mit sich.

Hier muß man von der üblichen Forderung nach einer Lösung (ein Satz optimaler Parameter) abgehen. Als Lösung wird die sogenannte "effiziente Menge" gesucht, die nach dem Pareto-Prinzip (/5/) gleichwertige Lösungen enthält: eine Lösung ($\vec{f1}$, $\vec{f2}$,..., $\vec{fn}$) ist nur dann besser als die Lösung (f1, f2,...,fn), wenn sie in jedem Gütewert besser ist ($\vec{fi}$ > fi für alle i=1(1)n); andernfalls sind sie gleichwertig; alle möglichen gleichwertigen Sätze von Parameterwerten (und zugehörigen Gütefunktionswerten) bilden die effiziente Menge (/5/).

Die Lösung von Vektoroptimierungsaufgaben (das Bestimmen und Beschreiben der effizienten Menge) erfordert somit eine geeignete Abtastung des Parameterraumes und Vergleich der zugehörigen Gütefunktionswerte nach dem Pareto-Prinzip, wobei eine sehr große Anzahl von Simulationsläufen (selbst bei wenigen Parametern) durchzuführen ist. Hybride Simulation bietet bei dieser Problemklasse wieder den Vorteil der kurzen und problemunabhängigen Rechenzeit für einen Simulationslauf.

Derzeit ist nun ein HYBSYS-Makro zur Vektoroptimierung in Entwicklung. Er hat dieselbe Aufruf-Syntax wie die vorher erwähnten Optimierungs-Makros:

VOPTIM,method f1,...,fm BY p1,...,pn

Dabei haben p1,...,pn und f1,...,fm dieselbe Bedeutung wie beim Makro OPTIM. Der Befehlsparameter "method" bestimmt die Methode, mit der der n-dimensionale Parameterraum abgetastet wird (äquidistante Abtastung, Zufallsgitter nach beliebigen Verteilungen, ...). Das Ergebnis (die effiziente Menge) wird tabellarisch ausgegeben und kann wahlweise auch graphisch dargestellt werden (ebene Schnitte im Parameter- und/oder Gütefunktionswertraum). Der Makro greift auf die übliche Programmierungsweise für HYBSYS-Makros zurück und verwendet HYBSYS-Aktivitäten, die von FORTRAN aufgerufen werden.

Literatur

1. ACSL User Guide/Reference Manual. Mitchell & Gauthier Ass., Concordia, MA.
2. Bausch-Gall I.: Parameteroptimierung bei technischen Modellen mittels einer kontinuierlichen Simulationssprache. Informatik-Fachberichte, 56, Springer (1983)
3. Breitenecker F.: The concept of supermacros in today's and future simulation languages. Mathematics and Computers in Simulation XXV, pp 279. (1984)
4. Breitenecker F.: Optimierung in kontinuierlichen Simulationssprachen - Aspekte bei Modellen technischer Systeme. Informatik-Fachberichte, 85, Springer (1984)
5. Peschel M.: Ingenieurtechnische Entscheidungen - Modellbildung und Steuerung mit Hilfe der Polyoptimierung. VEB Verlag Technik, Berlin. (1978)
6. Prantl R.: Programmpaket zur Lösung von Aufgaben der Parameteroptimierung auf einem Hybridrechner. Diplomarbeit Technische Universität Wien (1981)
7. Solar D., Berger F., Blauensteiner A.: HYBSYS - Interactive simulation software for a hybrid multiple-user system. Informatik-Fachberichte, 56, Springer, pp 257 (1983)
8. Solar D.: HYBSYS User Manual. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien (1984)

DER ALLGEMEINE ASSEMBLER METASM*)

A. Blauensteiner

Zusammenfassung

Metaassembler werden beim Design und bei der Programmierung von Mikrocomputern eingesetzt. Es wird der Metaassembler METASM vorgestellt, der horizontale und vertikale Mikroprogrammierung sowie eine Programmierung auf einer allgemeinen Assembler-Makro-Ebene ermöglicht. Weiters kann METASM als Übersetzer für den Transport von Systemprogrammen verwendet werden. METASM wurde an der Technischen Universität Wien entwickelt und steht dort im praktischen Einsatz.

1. Prinzipielles

Der Metaassembler METASM wurde 1981 entwickelt und seitdem ständig erweitert. Dies sollte vor allem den Bau eines mikroprogrammierbaren Bit-Slice-Rechners an der Technischen Universität Wien unterstützen. Zur Zeit wird METASM auch für Anwendungen auf dem Gebiet der Mikrocomputer-Programmierung und für Makro-Assembling verwendet. Sein allgemeines Konzept erlaubt seine Verwendung beim Transport von Programmen und bei Emulationen.

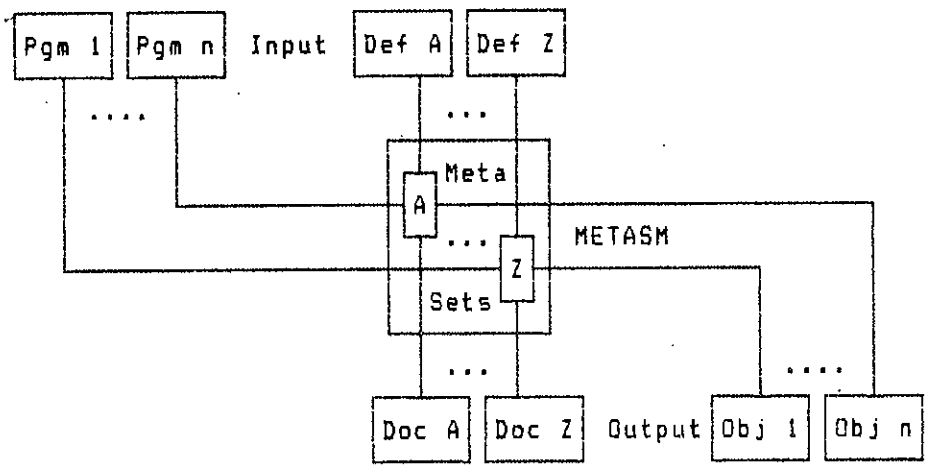


Bild 1
METASM Funktionen

Das METASM-System hat zwei Operations-Modes, die beide durch umfassende Syntax- und Semantiküberprüfung und präzise Fehlerbehandlung kontrolliert werden:

1. Der Definition Mode erlaubt die Definition der elementaren Programmcodes Def X für eine beliebige vorgegebene Architektur M(X) durch eine definierte Syntax. Diese Prozedur generiert ein Meta Set X und liefert die Dokumentation Doc X.

2. Der Assembly Mode bewerkstelligt die Übersetzung von symbolischen Programmeinheiten Pgm i (definiert durch die Syntax im Meta Set X) in ein Objekt Obj i, das zur Programmierung der Maschine M(X) verwendet werden kann.

---

* Übersetzung von A. Blauensteiner: METASM - a general assembler. Microprocessing and Microprogramming, Special Issue 1984, North Holland.

2. Assemblierung

Im Zwei-Pass Assembly Mode führt METASM ähnliche Operationen wie konventionelle Assembler durch. Jeder Befehl besteht zumindest aus dem Code ; dies ist ein Aktionswort des unabhängig definierten Meta Sets. Es gibt keine feste Unterteilung der Befehlselemente, diese werden implizit durch Verwendung von Separatoren erkannt. Diese und andere Kontrollzeichen können vom Benützer definiert werden. Nur wenige Kontrollcodes haben eine fixe Bedeutung wie END für das Ende einer Programmeinheit oder EQU für Tag-Equivalence.

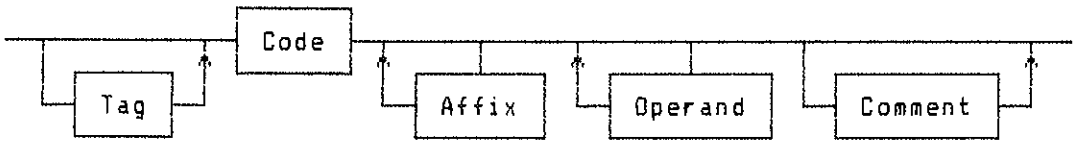


Bild 2
Elemente einer Befehlszeile

Je nach der Code-Definition können optional folgende Elemente zu einem Befehl hinzugefügt werden:

1. Ein Tag (Marke): Tags sind symbolisch kodierte Adressen, die einen bestimmten Befehl markieren.

2. Eine beliebige Anzahl von Affixes : Affixes sind symbolische Namen, die wie Befehle definiert werden, aber nicht als solche auftreten. Ein Affix ermöglicht die Verwendung von bestimmten uneingeschränkten Bitmustern in einem Befehl.

3. Eine bestimmte Anzahl von Operanden : Operanden sind Tags oder arithmetische Ausdrücke von Tags. Diese Tags müssen in derselben Programmeinheit vorkommen oder extern sein. Ausdrücke können aus Tags, dezimalen, oktalen oder hexadezimalen Konstanten, Zeichenketten und den üblichen arithmetischen Operatoren bestehen.

4. Ein beliebiger Kommentar .

3. Definition

Ein separater Definition Pass ermöglicht das Generieren oder Modifizieren eines Meta Sets X, das alle Konstruktionsregeln für die Befehle der Zielmaschine M(X) enthält.

Es muß ein Word Count definiert werden, der aus Paaren von Halbbytes besteht. Das ermöglicht das Referenzieren der Code-Definitionen und eine grundlegende Syntaxüberprüfung.

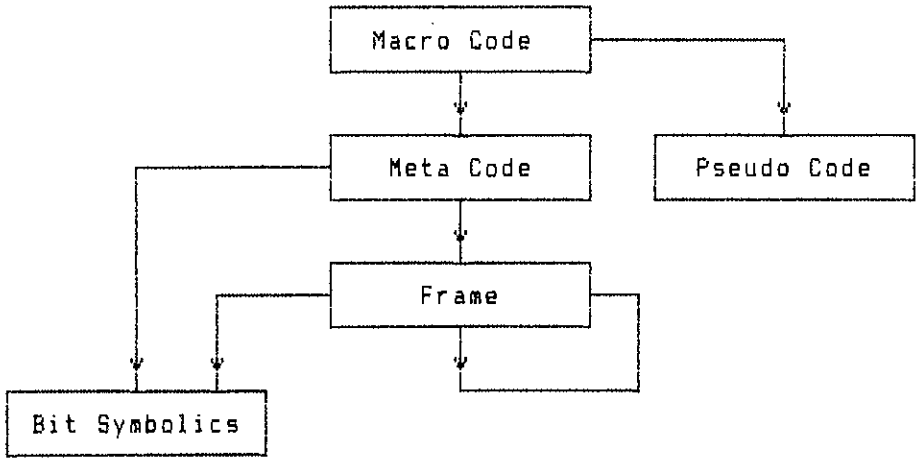


Bild 3
Zusammenhang der Befehlskonstruktion

Jeder definierte Code und jedes Codeelement besteht aus definierten Feldern mit verschieden vielen Bits.

Die Grundelemente der Befehlskonstruktion mit beliebiger Länge sind:

1. Bit Symbolics sind die einzigen bit-orientierten Strukturen (Bild 4). Bit Symbolics definieren die Anzahl der Bits eines bestimmten Field Types. Der Field Type zeigt an: absolute Bit-Verwendung oder absolute oder relative Verwendung eines Operanden oder das Einfügen eines Affixes, d.h. das Programmieren eines Frames, der zur Zeit der Assemblierung bekannt ist.

2. Frames können syntaktisch rekursiv verwendet werden. Jeder Frame wird aus Bit Symbolics und anderen Frames konstruiert, wobei die Anzahl der Operanden spezifiziert werden muß. Jeder Operand kann dabei abgebildet und arithmetisch verknüpft werden, wenn er von anderen Frames verwendet wird. Affixes müssen als Frames definiert werden.

   Eine spezielle Verwendung eines Frames ist der Conditional Frame : Bei dieser Konstruktion wird einer von zwei verschieden definierten Frames zur Zeit der Assemblierung gewählt, abhängig vom Ergebnis eines logischen Ausdrucks von Frames. Durch diesen Mechanismus wird eine dynamische Definition eines allgemeinen Codes, der von bestimmten Bedingungen abhängt, (z.B.: Adressen, Affixes, Operanden) möglich.

L	Low Bits
H	High Bits
X	Undefined Bits
A	Absolute Insertion of Operand
D	Displacement Operand/Address
F	Affix Field

Operands and Affixes are identified by the lower case letter according to the position of coding (i.e. c represents the third Operand or Affix, depending on usage).

Bild 4
Field Types

Jeder Code muß ein Vielfaches des globalen Word Counts sein und kann sein:

1. Ein Meta Code : Meta Codes sind die grundlegenden exekutierbaren Codes. Sie bestehen aus einer beliebigen Anzahl von Worten und werden durch Frames oder Bit Symbolics definiert. Es besteht keine Einschränkung durch Wortgrenzen.

Für jeden Meta Code kann die Anzahl der Operanden anders definiert werden. Die Operanden können abgebildet und arithmetisch verknüpft werden. Die Anzahl der Affixes ist unbeschränkt, das Prozessieren ist abhängig von ihrem Auftreten und ihrer Definition.

2. Ein Macro Code : Macro Codes sind ein frei definierbares Set verschiedener Meta Codes. Ebenso kann für jeden Macro Code die Anzahl der Operanden verschieden sein. Die Operanden können abgebildet und arithmetisch verknüpft werden. Die Anzahl der Affixes ist unbeschränkt. Jedes Affix eines Macro Codes kann in beliebiger Reihenfolge den Meta Codes zugeordnet werden, die Elemente der Macro Codes sind. Sie müssen jedoch nicht zugeordnet werden und es gibt auch die Möglichkeit, Frames direkt als konstante Affixes zu verwenden.

3. Ein Pseudo Code : Pseudo Codes sind spezielle maschinenabhängige Codes, die eine vorgegebene Maximalanzahl von Operanden haben. Ein frei definierter arithmetischer Ausdruck der Operanden bestimmt den Ladepunkt der Fortsetzung nach dem Pseudo Code. Dadurch wird bei Weiterverwendung des assemblierten Objekts das Binden eines exekutablen Programmes ermöglicht. Die Operanden von Pseudo Codes müssen nicht in der laufenden Programmeinheit auftreten, in diesem Fall werden sie Pseudo Tags (das sind Externals) genannt. Ein Pseudo Code kann ohne Einschränkungen Affixes verwenden.

4. Objekts

METASM generiert eine benützerfreundliche Liste des assemblierten Programmes. Die Liste ist dynamisch strukturiert, je nach dem Word Count und der Anzahl und der Dimension der Affixes und Operanden. Die Liste enthält Zeilenkontrolle, den assemblierten Code mit Angabe von Feldüberläufen und Pseudo-Referenzen und eine symbolisch formatierte Wiedergabe des Sourcebefehls. Fehlermeldungen werden in der Zeile des Auftretens des Fehlers eingefügt. Sie erscheinen anstelle der hexadezimalen Code Liste in Form eines Diagnostextes.

Es können zwei verschiedene Arten von Listen angefordert werden:

1. Eine Echo Liste , die die Befehle, wie kodiert, wiedergibt.

2. Eine Splitting List , die einen Macro Code in seine Elemente zerlegt. Da es von der Meta Set-Definition abhängt, ob Macros definiert werden oder nicht, kann der Benützer den METASM dazu bringen, als Übersetzer zu arbeiten.

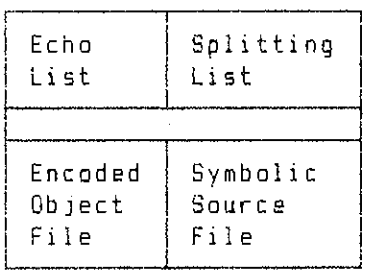


Bild 5
Objekt

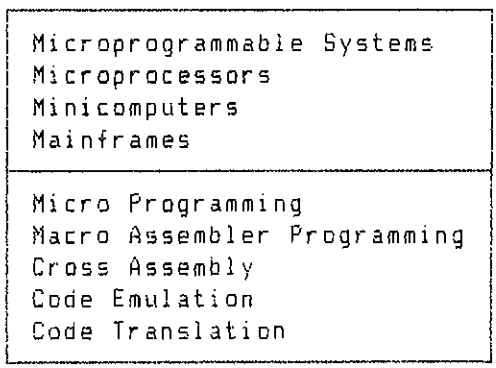


Bild 6
Verwendung

Unabhängig von der Liste kann ein File Output gewählt werden:

1. Der Objekt Output von METASM kann kodiertes Objekt sein, das alle binär kodierte Informationen enthält, die für einen Postprozessor (Loader, Linker, Builder) notwendig sind.

2. Oder es kann ein symbolischer Output sein, der alle Meta- und Macro-Befehle enthält, symbolisch aufgegliedert in ihre Code-Elemente und ihre aktuellen Affixes und Operanden, einschließlich Abbildungen und Ausdrucke, wie definiert. Dieser symbolische Source File kann durch einen anderen Prozessor oder (Meta)- Assembler zur weiteren Verarbeitung prozessiert werden. Auf diese Weise ist es möglich, ein symbolisches Programm in ein anderes zu übersetzen, wobei die Funktionalität unverändert bleibt, was für die Übertragung von Programmen sehr wichtig ist.

Neben der Mikroprogrammierung wurde METASM erfolgreich als Cross-Assembler für PDP 11/23+ und INTEL 8086 Systeme eingesetzt.

Hier sollen einige prinzipielle Verwendungsmöglichkeiten von METASM hervorgehoben werden:

- Ein und dasselbe Source-Programm S kann mit verschiedenen Meta Sets A und B für verschiedene Maschinen M(A) und M(B) assembliert und auf diese Weise virtuell verwendet werden.

- Zwei Source-Programme S und T in verschiedenen Sprachen können mit verschiedenen Meta Sets A (für S) und B (für T) für dieselbe Maschine M übersetzt werden, wodurch der Zugriff zu einer bestimmten Maschine durch verschiedene Sprachen möglich wird.

- Ein Source-Programm T für die Maschine M(B) kann auf der Maschine M(A) simuliert werden, wobei ein anderes Meta Set (A) verwendet wird.

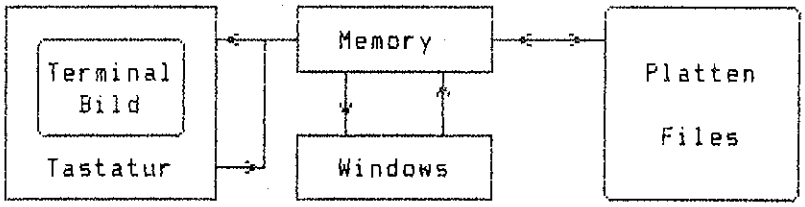
5. Schluß

METASM ist ein universelles Werkzeug zum Einsatz in Microprocessing, (Cross)-Assembling, Code Simulation und Code Translation. Er ermöglicht das Aufstellen komplexer Konstruktionsregeln, auch mit Bedingungen. Der Benutzer kann seine Aufgabe logisch angehen und erhält ausführliche Dokumentation.

Der Source Editor JCSSED

A. Blauensteiner

Im letzten Jahr wurde an der Hybridrechenanlage ein bildschirmorientierter Source Editor JCSSED entwickelt, der in Erweiterung zum systemunterstützten zeilenorientierten Source Processor JCSSOP zum flächenmäßigen Bearbeiten von Programmen, Datenbeständen und Dokumentation eingesetzt wird.



Informationsschema

Es können voneinander unabhängige Source Files errichtet bzw. angesprochen werden, deren Informationen auch untereinander durch Windows umgelagert werden können. Der angesprochene Source File, der die zu bearbeitenden Daten oder Texte enthält, ist an sich plattenresident und wird je nach Art der Operationen auf einen bestimmten Teil im Memory gelagert, von welchem wiederum die sichtbare Information von einstellbar vielen Spalten und Zeilen am Bildschirm dargestellt wird.

Prinzipiell werden zwei Betriebsarten unterschieden: ein Modus, der die allgemeine Eingabe von Steueranweisungen ermöglicht, und ein weiterer Modus, der durch eine bestimmte Steueranweisung ausgelöst wird und die Texteingabe erlaubt. Diese Texteingabe findet mittels des Cursors in dem abgegrenzten Teil des Bildschirms statt, in dem die Informationen sichtbar sind. Die Steueranweisungen aber werden am Beginn der ersten Zeile im Rahmen eines dreizeiligen Status- und Informationsblockes angefordert.

Die Texteingabe ermöglicht das Eintippen von 256 virtuellen Terminal-Funktionen, meist Buchstaben oder Zeichen, aber auch Funktionen wie Hoch- und Tiefstellen von Texten oder visuellen Attributen. Die Eingabe wird durch Ausnützen aller Shift- und Control-Tasten und zusätzlich von Tasten mit vorangehendem Escape erreicht. Die Ausgabe wird durch das virtuelle Terminalsystem des Betriebssystems JCS/VS 8 ermöglicht, durch welches alle virtuellen Terminal-Funktionen dem eigenen Gerätetyp nach Möglichkeit angepaßt werden. Der Cursor kann bei der Texteingabe jederzeit mit den Richtungstasten des Terminals bewegt werden. Zur schnelleren Cursorsteuerung gibt es definierbare Tabulatoren in horizontaler und vertikaler Richtung, jeweils nach oben und nach unten. In gleicher Weise gibt es logische Tabulatoren in allen Richtungen, die die nächstliegende Informationskette, meistens ein Wort, überspringen. Außerdem kann jederzeit nicht nur der vorangehende Buchstabe, sondern ein dem Cursor benachbartes Wort gelöscht werden, sowie der Text der Vorzeile dupliziert werden. An der Cursorposition kann jeweils ein Zeichen in die aktuelle Zeile eingefügt oder weggenommen werden. Ebenso erfolgt ein Einfügen oder Löschen von ganzen Zeilen, wobei der Einbruch bzw. Ausbruch an jeder beliebigen Spalte erfolgen kann. Die Ränder können versetzt werden, sodaß nur in einem optisch abgegrenzten Teil des Bildschirms gearbeitet werden kann. Dabei gilt Wortfortschreibung am rechten Rand, sodaß ein nicht beendetes Wort automatisch in die nächste Zeile versetzt wird. Ebenso rückt das Bild automatisch auf, falls über die letzte Zeile hinausgeschrieben wird. Größere Manipulationen von Textblöcken können über allgemeine Steueranweisungen durchgeführt werden.

Solche Steueranweisungen außerhalb des Textmodus beziehen sich meist auf sogenannte Boxen , das sind rechteckige Bereiche beliebiger Informationen, die durch zwei gegenüberliegende Eckpunkte einer Diagonale identifiziert werden. Dabei sind diese Eckpunkte nach freier Wahl durch den Cursor anzuwählen, wobei sämtliche Cursorbewegungen und Tabulatorsteuerungen verwendet werden können. Boxen können so gelöscht, verschoben oder kopiert, ausgelagert oder eingelagert werden. Sie können vertauscht werden, eventuell auch so, daß dazwischenliegende Informationen verschoben werden müssen, falls die Boxen ungleich groß sind. Der Text von Boxen kann zentriert oder mit Randausgleich sowie Absatzeinrücken versehen werden, wobei auch Abteilungszeichen berücksichtigt werden. Ebenso läßt sich ein bestimmter Text durch einen unterschiedlich langen anderen innerhalb einer Box oder im ganzen File ersetzen. Neben Boxen können auch Bereiche mehrerer Zeilen kopiert oder verschoben werden, indem man wie bei den Boxen den Beginn der relevanten Zeilen durch den Cursor anwählt und bestätigt.

Eine spezielle Eigenschaft ist die Möglichkeit, Kästchen und Linien mit semigrafischen Zeichen darzustellen, hinzuzuzeichnen und wegzulöschen, wobei die entsprechenden Kreuzungspunkte und Anschlußstücke entsprechend der optischen Anforderung ausgewechselt werden. Für die Durchführung einer solchen Zeichnung braucht wiederum nur die entsprechende Box durch den Cursor bestätigt werden.

Um eine entsprechende Beweglichkeit innerhalb eines Files zu erhalten, können die sichtbaren Seiten entweder seitenweise, blockweise oder zeilenweise nach vor bzw. zurück gerollt werden. Weiters kann eine Seite an einer bestimmten Position innerhalb des Files gesucht werden, sei es durch ihre absolute Position nach der Zeilennummer oder durch einen Schlüsseltext, der gesucht wird.

Bearbeitete Files können ganz oder auszugsweise ausgedruckt werden, wobei auch Beschränkungen am Rand gesetzt werden können, Kopf- oder Fußzeilen definiert werden können, sowie Einzelblattzuführung durch ein Anhalten des Druckvorganges ermöglicht wird. Die Seitengröße ist dabei frei definierbar und am Bildschirm durch entsprechende Markierungen im Rand kontrollierbar. Beim Drucken werden alle virtuellen Zeichen und Funktionen entsprechend dem Drucker ausgegeben, sodaß Indizierung, Fettdruck oder mathematische Sonderzeichen dargestellt werden können.

Um die Betriebsicherheit zu erhöhen, kann man den veränderten Memory-Inhalt auf Wunsch sicherheitshalber auf File schreiben oder den Bildschirminhalt nach dem Memory-Inhalt neu darstellen lassen.

Es gibt eine Reihe von jederzeit einstellbaren Options , durch welche der Betriebszustand den unterschiedlichen Anwendungen angepaßt werden kann. So kann der Bildschirminhalt entweder entsprechend der optischen Funktionen dargestellt werden, oder aber so, daß solche Funktionen auch editiert werden können, da sie ja Teil der gespeicherten Information sind. Damit kann etwa eine Subscript-Anweisung durch eine Superscript-Anweisung ersetzt werden. Eine andere Option ermöglicht es, sich ständig im Text-Modus für Zeicheneingabe zu befinden und von dort aus durch Sonderfunktionen allgemeine Steueranweisungen auszulösen.

Umfangreiche Plausibilitätsmechanismen und mehrere Ebenen von internen Speichern bieten den Cursor dem Benützer für seine Anweisungen so an, daß jener möglichst wenig bewegt werden muß.

Eine ausführliche Beschreibung des Source Editors JCSSED ist an der Hybrid-rechenanlage erhältlich.

Anschluss der Hybridrechenanlage an das DATEX-P Netz

Pilotprojekt ACONET Wien*)

A. Blauensteiner

1. Einführung

Das Pilotprojekt ACONET Wien wurde an der Technischen Universität Wien in Koordination mit Forschungsprojekten des Institutes für Informatik der Universität Linz und der Institute für Informationsverarbeitung der Technischen Universität Graz durchgeführt. Das Projekt zeigt Methoden auf, um Einzelrechner und Subnetze verschiedener Universitätsbereiche in ein Akademisches Computernetz (ACONET) zu integrieren. Dazu wurde von Projektteams der beteiligten Universitäten ein Anpassungsrechnerkonzept entwickelt und in einem experimentellen Verbund verifiziert, das die Integration von Systemen beliebiger herkömmlicher, firmenbezogener Kommunikationsarchitektur gestattet.

Am 13. Dezember 1984 wurden die Ergebnisse der Arbeiten von den drei beteiligten Projektgruppen an einem ACONET-Demonstrationstag im Kontaktraum der Technischen Universität Wien präsentiert.

Als Trägernetz für den Verbund wurde das öffentliche Datenpaketvermittlungsnetz der österreichischen Post- und Telegraphenverwaltung DATEX-P verwendet. Die Realisierung der Kommunikationssoftware in den Anpassungsrechnern wurde streng nach den Regeln der Open Systems Interconnection von ISO durchgeführt. Dies stellt die Möglichkeit der problemlosen Integration weiterer Einzelsysteme sowie der Kommunikation mit internationalen Datennetzen sicher.

In dieser Arbeit wurden in den Anpassungsrechnern Netzzugangssoftware für DATEX-P gemäß CCITT X.25 (entspricht funktional den Schichten eins bis drei) und darauf aufbauend die Kommunikationsschicht vier des siebenschichtigen ISO-Schichtmodells realisiert. Diese Schichten ermöglichen die Kooperation zwischen Anwendungsprogrammen in Systemen einzelner Universitätsbereiche. Um allerdings eine Verbunddienstleistung mit allen Funktionen zur Verfügung stellen zu können, müßten auch noch die Schichten fünf bis sieben implementiert werden.

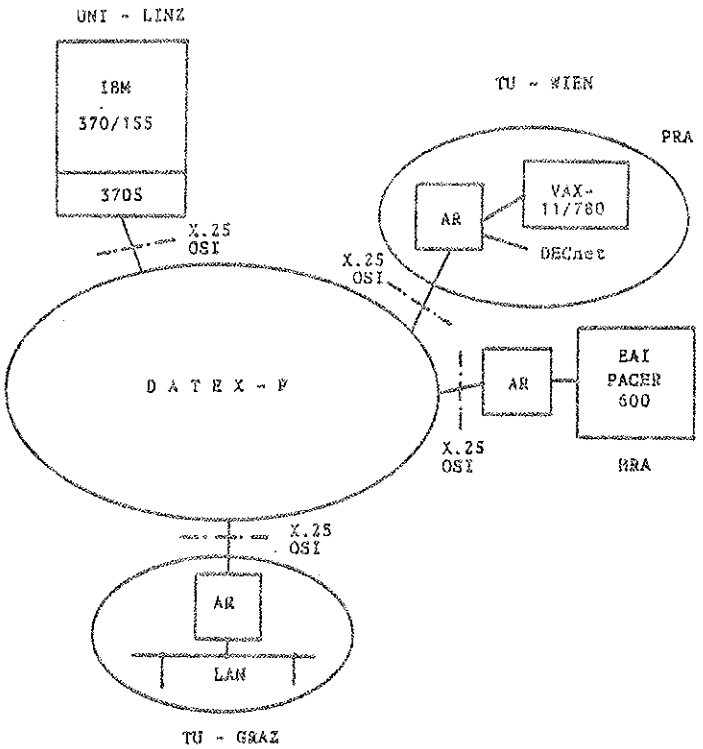


Abb. 1: Aufbau des experimentellen ACONET

AR ... Anpassungsrechner
LAN ... Local Area Network
HRA ... Hybridrechenanlage
OSI ... Open Systems Interconnection
PRA ... Prozeßrechenanlage

* Aus dem Projektbericht zu Zl. 60.017/1-26/82, November 1984, des vom Bundesministerium für Wissenschaft und Forschung, Abteilung II/6 (Dr. Rozsenich) erteilten Forschungsauftrags (Projektleitung: o.Prof.Dr. H. Stimmer).

2. Die Integration des Hybridrechners in den experimentellen Verbund

2.1 Realisierungskonzept

Die Kopplung des digitalen Teils des Hybridrechners (EAI PACER 100) mit dem Anpassungsrechner erfolgte über eine Parallelschnittstelle (siehe Abb. 2). Für diese Schnittstelle wurde ein geeignetes Protokoll ausgearbeitet, das möglichst einfach sein sollte und im wesentlichen eine bloße Durchreichefunktion zwischen Anpassungsrechner und PACER 100 ausführt, so als ob die Schnittstelle zwischen den Schichten 4 und 5 im Anpassungsrechner im PACER 100 selbst bedient werden würde. Spezielle Fehlersicherungen konnten auf ein Minimum reduziert werden, da der Anpassungsrechner unmittelbar neben dem PACER 100 im Rechenraum der Hybridrechenanlage aufgestellt wurde.

Die Hybridrechenanlage verfügt über ein selbstentwickeltes Betriebssystem, sodaß die Implementierung der Kommunikationssoftware für die Integration des EAI PACER 100 in den experimentellen Verbund in optimaler Weise, nämlich direkt in das Betriebssystem, erfolgen konnte.

Das Implementierungskonzept sieht vor, daß eine Transportverbindung vom/zum überregionalen Verbund durch das Programm HYBNET im Anpassungsrechner auf eine sogenannte virtuelle Leitung abgebildet wird, die über die Parallelschnittstelle in den PACER 100 führt und von dessen Betriebssystem im Prinzip wie eine lokale Terminalleitung behandelt wird.

Die Daten werden zwischen Anpassungsrechner und PACER 100 byteweise über die Parallelschnittstelle ausgetauscht. Der Unterschied einer virtuellen Leitung zu einem normalen Terminal besteht in zweierlei Hinsicht: einerseits ist das der virtuellen Leitung entsprechende Terminal nicht physikalisch vorhanden, das heißt, die Daten müssen im Anpassungsrechner entsprechend aufbereitet werden, um sie auf die entsprechenden virtuellen Leitungen zu verteilen. Andererseits müssen über ein einziges physikalisches I/O-Port mehrere logische Anschlüsse durchgeführt werden. Über die Parallelschnittstelle können bis zu vier logische Benutzer ihre Datentransfers im Adreßmultiplexbetrieb durchführen. Im Anpassungsrechner müssen diese Datenströme ebenfalls nach dem jeweiligen logischen Benutzer getrennt behandelt und entsprechenden Transportverbindungen zugeordnet werden. Das Protokoll zwischen Anpassungsrechner und dem PACER 100 muß in der Lage sein, den Adreßmultiplexbetrieb über die Parallelschnittstelle abzuwickeln und dafür die nötigen Kontrollfunktionen bereitzustellen.

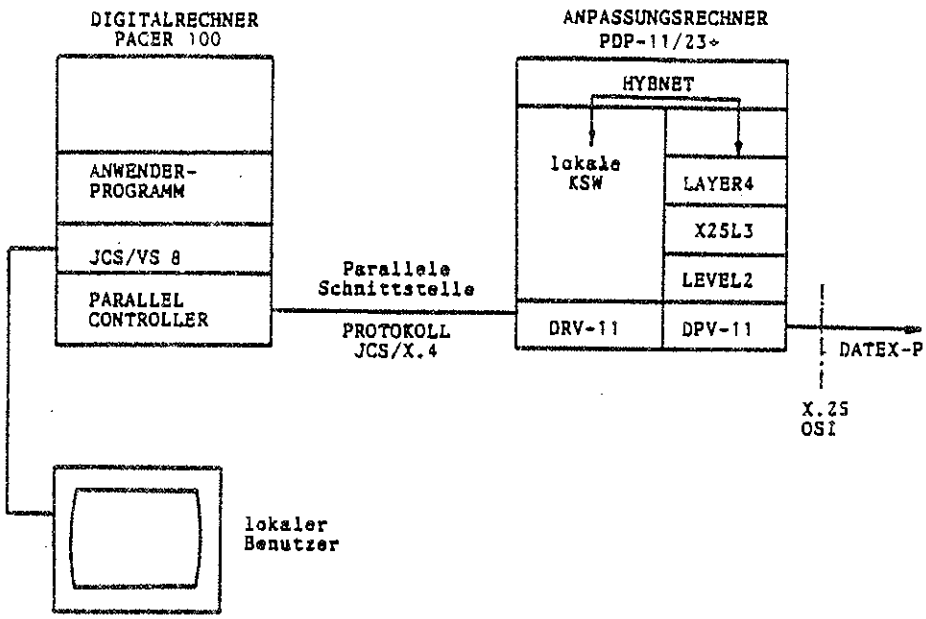


Abb. 2: Integration des Hybridrechners der Hybridrechenanlage in das experimentelle ACONET

Über die physikalische Verbindung zwischen Anpassungsrechner und PACER 100 können 16 Bits parallel übertragen werden. Das Byte mit der geringeren Wertigkeit enthält die Nutzdaten, das mit der höheren Wertigkeit die nötige Kontrollinformation (=Kontrollbyte). Im Kontrollbyte können die Dienstelemente, die an der Schnittstelle 4/5 im Anpassungsrechner zur Verfügung stehen, unterschieden werden. So sind Funktionscodes für den Aufbau einer Transportverbindung, für den Transfer normaler Daten und von Vorrangdaten sowie für den Abbau der Transportverbindung definiert. Darüber hinaus sind zusätzliche Steuerfunktionen enthalten, die es ermöglichen, Restarts des Anpassungsrechners dem PACER 100 anzuzeigen. Es kann zwischen einem kompletten Neustart und dem Neustart für einen bestimmten Benutzer unterschieden werden. Zwei Bits stehen im Kontrollbyte zur Verfügung, um den logischen Benutzer bzw. dessen virtuelle Leitung zu identifizieren.

Da eine Transportverbindung über eine virtuelle Leitung auf einen Terminalanschlußpunkt des Betriebssystems abgebildet wird, stehen dem Benutzer am PACER 100 die normalen Terminal-Ein/Ausgabe-Prozeduren zur Verfügung, um den Datentransfer zu oder von einem überregionalen Bereich zu verwalten. Eine entsprechende logische Gerätenummer bzw. deren Zuordnung zu einer virtuellen Leitung der Parallelschnittstelle zum Anpassungsrechner zeigt dem Betriebssystem an, daß nicht mit einem lokalen Terminal, sondern mit einem entfernten Benutzer der überregionalen Transportdienstleistung kommuniziert wird. Für die Benutzung der Dienstleistungen des Anpassungsrechners sind jedoch noch zusätzliche Funktionen nötig, die beim Betrieb lokaler Terminals wegfallen. So muß zum Beispiel eine Transportverbindung zum gewünschten überregionalen Kommunikationspartner erstellt werden, ehe Daten übertragen werden können. Dafür steht eine eigene Systemprimitive, der sogenannte NET-Trap, zur Verfügung. Dieser ermöglicht dem Systemprogrammierer und über eine entsprechende zusätzliche Prozedur auch dem Anwendungsprogrammierer, die Erstellung bzw. den Abbau von Verbindungen zu entfernten Kommunikationspartnern zu kontrollieren.

2.2 Prozeduren im Betriebssystem JCS/VS 8 am PACER 100 zur Benutzung der Verbundfunktionen

2.2.1 Daten zum Anpassungsrechner

Daten eines Benutzers am PACER 100, welche über die normalen Ausgabeprozeduren zu einem entfernten Kommunikationspartner transportiert werden sollen, werden durch die entsprechende logische Gerätenummer der betreffenden virtuellen Leitung zum Anpassungsrechner und darüber hinaus einer Transportverbindung zum Ziel-Bereich zugeordnet und in isomorpher Weise wie bei den anderen Ausgabeeinheiten in spezielle Puffer gestellt. Auf dieser Ebene kann der Ausgabedatenstrom für die maximal 4 entfernten Kommunikationspartner noch eindeutig unterschieden werden. Die Daten werden gepackt gespeichert. Virtuelle Zeichen werden noch verschlüsselt mitgeführt. Für die Weiterleitung der Daten ist das Interruptsystem verantwortlich, soferne es dazu bereit ist.

2.2.2 Daten vom Anpassungsrechner

Falls das Interruptsystem Daten für den PACER 100 erhält, werden diese in bereits nach logischen Benutzern unterschiedene Puffer eingeordnet. Diese Daten sind bereits von allen Protokollsteuerinformationen befreit und werden als reine Nutzinformation dem Run-Time-System zur Weiterverwendung zur Verfügung gestellt. Der Benutzer greift unter Verwendung der gewohnten Eingabeprozeduren auf den ihm zugeordneten Puffer zu und erhält Daten, soferne welche vorhanden sind. Auch diese Prozeduren gehen völlig äquivalent zu den Eingabeprozeduren anderer Terminals vor sich.

2.2.3 Der NET-Trap

Ein eigener Systemaufruf, NET-Trap genannt, wurde geschaffen, um dem Benutzer den Aufbau einer Verbindung über den Anpassungsrechner zu einem entfernten Kommunikationspartner zu ermöglichen. Der Benutzer fordert durch einen solchen NET-Trap eine virtuelle Leitung zum Anpassungsrechner an und gibt für diese eine logische Gerätenummer bekannt, mit der er in Hinkunft die neue Verbindung ansprechen will. Vom Betriebssytem wird nun diese logische Gerätenummer einer freien virtuellen Leitung zugeordnet, falls noch eine freie vorhanden ist, und für den anfordernden Benutzer reserviert. Diese Zuteilung kann nach Beendigung des Betriebssytemaufrufs auch vom Benutzer abgefragt werden.

Außerdem übergibt der Benutzer über diesen Systemaufruf die notwendigen Parameter zur Erstellung der gewünschten Verbindung, wie zum Beispiel die Zieladresse, dem Betriebssystem. Diese Information wird sodann unter Verwendung des definierten Protokolls zum Anpassungsrechner weitergeleitet. Das Relais-Programm HYBNET im Anpassungsrechner fordert sodann eine Transportverbindung zum entfernten Bereich an, die von den Kommunikations-programmen für den überregionalen Verbund erstellt wird.

2.2.4 Das Interruptsystem

Losgetrennt von den Run-Time-Prozeduren arbeitet asynchron zu diesen ein Interruptsystem, welches die Interrupts der Übertragungseinrichtung für die parallele Datenübertragung zwischen Anpassungsrechner und PACER 100 verarbeitet. Es wird aktiv sowohl, wenn Daten vom Anpassungsrechner für den PACER 100 anliegen, als auch dann, wenn Daten vom PACER 100 zum Anpassungsrechner gesendet werden können, weil dieser empfangsbereit ist. Außerdem wird angezeigt, ob der Anpassungsrechner noch aktiv ist.

Daten vom PACER 100 für den Anpassungsrechner werden in einem zyklischen Verfahren aus den vier Puffern für die virtuellen Leitungen ausgelesen. Die Priorität der Puffer wird laufend verändert, sodaß völlige Gleichberechtigung herrscht. Es wird jedoch nach einer Ausgabe als nächstes so lange der gleiche Benutzer behandelt, bis sein Puffer vorübergehend einmal geleert ist. Dies geschieht deswegen, um möglichst rasch eine vollständige Transportservice-Dateneinheit eines Benutzers zum Anpassungsrechner übermitteln zu können und damit auch einen optimalen Datentransfer über das Datennetz zu erreichen.

Jedes zu sendende Byte wird mit der entsprechenden Kennung des jeweiligen Benutzers versehen. Außerdem wird eine entsprechende Kontrollinformation hinzugefügt, falls dies notwendig ist, und das Wort sodann an das generelle Interrupt-Output-System weitergeleitet. Hier unterscheidet sich die Prozedur wieder nicht mehr von der eines normalen Terminals. Erst zu diesem Zeitpunkt werden eventuelle virtuelle Character in die entsprechende physikalische Characterfolge zerlegt und weiter verarbeitet. Enthält mindestens einer der Ausgabepuffer mindestens ein zu sendendes Byte, so wird der Anpassungsrechner aufgefordert, seine Bereitschaft zum Abholen dieses Bytes zu bekunden. Sind alle Ausgabepuffer leer, wird dieser Teil des Interruptsystems abgeschaltet, bis wieder zu sendende Daten in den Ausgabepuffern einlangen.

Das Interruptsystem ist immer bereit, Daten vom Anpassungsrechner aufzunehmen und an die entsprechenden Benutzer weiterzuleiten. Ankommende Daten werden entsprechend der Benutzerkennung getrennt und in den zugeordneten Puffern abgelegt.

Wird ein Restart angezeigt, so werden sämtliche bisher aktiven Benutzer abgeschaltet und alle virtuellen Leitungen wieder für frei erklärt.

Wird ein zeitweiser Ausfall des Anpassungsrechners angezeigt, so wird so lange pausiert, bis die neuerliche Betriebsbereitschaft des Anpassungsrechners registriert werden kann.

2.3 Das Anpassungsprogramm HYBNET im Anpassungsrechner

Hauptaufgabe des Programmes HYBNET im Anpassungsrechner ist es, die Dienstelemente der Transportschicht im Anpassungsrechner entgegenzunehmen und gemäß diesen über die Parallelschnittstelle mit dem Betriebssystem JCS/VS 8 zu kommunizieren. Nach erfolgtem Verbindungsaufbau soll dies so geschehen, als wäre der Anpassungsrechner ein Terminal des PACER 100. Umgekehrt muß HYBNET die vom PACER 100 eintreffenden Zeichenfolgen interpretieren und in entsprechende Dienstelemente für die Schicht 4 im Anpassungsrechner umwandeln, um auf diese Art und Weise die Verbindungsfunktionen der Transportschicht, die der Anpassungsrechner bietet, in Anspruch nehmen zu können.

Für die Kommunikation zwischen dem Anpassungsrechner und dem PACER 100 wird im PACER 100 ein im Rahmen einer Diplomarbeit am Institut für elektrische Regelungstechnik entwickeltes Parallelinterface eingesetzt. Das Gegenstück im Anpassungsrechner bildet eine Übertragungseinrichtung für parallele Datenübertragung vom Typ DRV-11 (siehe Abb. 2).

Die Kommunikationssoftware im Anpassungsrechner ermöglicht eine wesentliche Erweiterung der Funktionalität des Hybridrechensystems PACER 600. Durch sie wird sowohl der Zugang entfernter Benutzer über den überregionalen Verbund zu den Resourcen der Hybridrechenanlage, zum Beispiel zur Durchführung hybrider Simulationen, als auch der Zugriff lokaler Benutzer zu entfernten Systemen im überregionalen Verbund möglich.

3. Erfahrungen aus Kommunikationsversuchen zwischen Systemen der Hybridrechenanlage und der Prozeßrechenanlage

In den Endsystemen dieser Bereiche wurden geeignete Anwendungsprogramme entwickelt, die es gestatten, von Terminals an Systemen des einen Bereiches auf Systeme des anderen Bereiches zuzugreifen, soweit dies bei Fehlen der Funktionen der Schichten 5 und 6 sinnvoll realisiert werden konnte. Ein entferntes Terminal greift auf ein System eines Bereiches so zu, als wäre es ein lokales Terminal. Kommunikationsversuche zwischen der Hybridrechenanlage und der Prozeßrechenanlage zeigten, daß dieses Prinzip über ein öffentliches Datennetz nicht vollständig realisiert werden kann. Während nämlich das Echo auf Zeichen, die von einem lokalen Terminal an einen Rechner gesendet werden, in der Regel nicht vom Terminal, sondern vom Rechner erzeugt wird, muß das Echo der von einem entfernten Terminal gesendeten Zeichen bereits im Quellbereich generiert werden und nicht erst vom Zielsystem. Dies ist aus zwei Gründen notwendig: Erstens würden die Antwortzeiten, bis ein eingegebenes Zeichen am Terminal auch sichtbar erscheint, für den Benutzer nicht akzeptabel kurz werden, wenn das Echo vom Zielsystem erzeugt und zum Quellterminal über DATEX-P übertragen würde, und zweitens würden die Kosten für die Datenübertragung in DATEX-P beträchtlich steigen.

Manche Anwendungen gehen aber davon aus, daß das Echo vom Zielsystem erzeugt wird, zum Beispiel Programme, die eine automatische Ergänzung teilweise eingegebener Befehle vornehmen usw.. Solche Applikationen können in dieser Form nicht über DATEX-P betrieben werden.

4. Ausblick auf weiterführende Arbeiten

Die Ergebnisse dieses Forschungsprojektes stellen eine erste Basis dar, auf der bei der Realisierung eines ACONET aufgebaut werden kann. Im gegenwärtigen experimentellen Verbund steht lediglich eine normierte Transportdienstleistung zur Verfügung, auf der kommunizierende Programme direkt aufsetzen. Die Realisierung allgemeiner Dateitransfers, eines komfortablen Dialogbetriebes sowie die Erschließung verschiedener Dienste für einen Universitätsbereich erfordern aber eine funktionsmäßig wesentlich breitere Basis, die nur durch die Implementierung der Kommunikationssteuerungs- und der Darstellungsschicht in den Endsystemen bzw. in den Anpassungsrechnern geschaffen werden kann.

Aus diesem Grund mußte die Realisierung vieler Funktionen aufgeschoben werden. Damit z.B. die Resourcen der Hybridrechenanlage von der VAX aus wirklich mit dem nötigen Komfort benutzt werden können, sind noch viele zusätzliche Arbeiten im Bereich der höheren Kommunikationsprotokolle und im Bereich der tieferen Verankerung der nötigen Kommunikationssoftware im Betriebssystem der VAX 11/780 notwendig. An der Hybridrechenanlage konnte die Kommunikationssoftware bereits von Beginn an im Betriebssystem, das autonom von der Abt. Hybridrechenanlage verwaltet wird, integriert werden; eine Implementierung der Funktionen der Schichten 5 und 6 ist aber auch hier nötig, um mit dem gewünschten Komfort auf Resourcen anderer Systeme zugreifen zu können.

Teletex-Fernschreiben

Ab Anfang Juni 1985 tritt durch den Teletex-Dienst der Post eine wesentliche Verbesserung des Telex-Dienstes in Kraft. Für ankommende Fernschreiben ist die bisherige Telex-Nummer der Technischen Universität Wien nicht mehr zu verwenden. Der Telex-Anschluß der Technischen Universität kann von jedem bestehenden Telex- und Teletex-Anschluß aus erreicht werden. Die Telex-Nummer der Technischen Universität Wien ist 232 - 3222 467 = TUW. Dabei ist 232 die Landeskennzahl für Österreich und TUW der Namensteil der Telex-Nummer. Jedes einlangende Fernschreiben sollte neben dem Namen des Empfängers auch eine Institutsangabe enthalten (E0202 = Hybridrechenanlage).

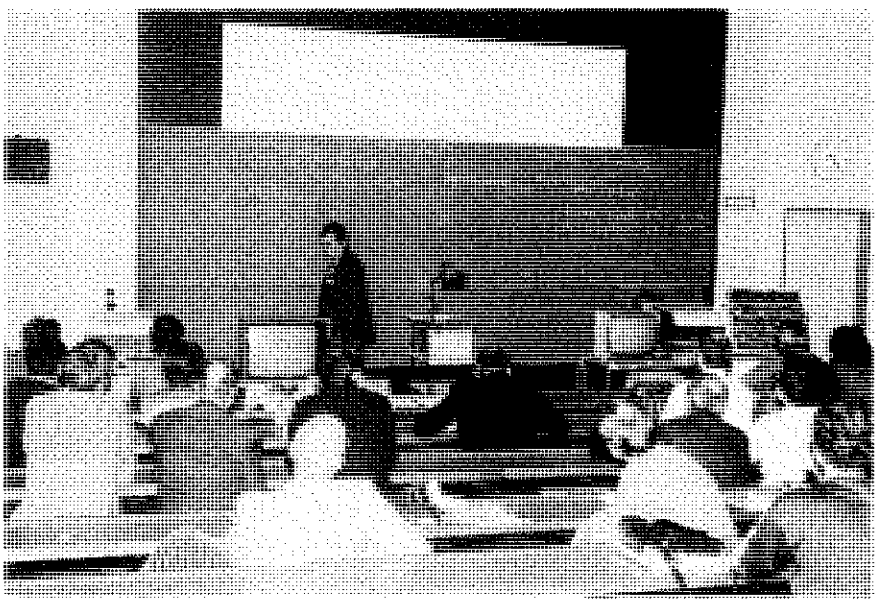
2. Symposium Simulationstechnik

Vom 25. bis 27. September 1984 fand an der Technischen Universität Wien das "2. Symposium Simulationstechnik" statt. Veranstalter waren der ASIM-Fachausschuß 4.5 Simulation in der Gesellschaft für Informatik (GI), die Österreichische Computergesellschaft ÖCG und das Hybridrechenzentrum der Technischen Universität Wien.

Die Tagungsleitung hatten Doz.Dipl.Ing.Dr. F. Breitenecker und Dipl.Ing.Dr. W. Kleinert.

Die Tagung wurde von den Firmen EAI GmbH, Control Data GmbH, SYMBOLICS GmbH, Mitchell & Gauthier Ass. und Mannesmann-Tally gesponsert.

Am Tag vor der Konferenz, am 24. September, wurde ein Tutorium über Hybridrechnen vom Hybridrechenzentrum abgehalten. W. Kleinert berichtete zuerst über das Hybridrechensystem an der Technischen Universität Wien. D. Solar stellte die selbstentwickelte interaktive hybride Simulationssoftware HYBSYS vor. Anschließend erfolgte im Hörsaal eine Demonstration des Hybridrechners durch Vorführung ausgewählter Beispiele. Eine Diskussion über die Zukunft des Hybridrechnens vervollständigte das Programm des Tutoriums.



Ein internationales Programmkomitee wählte 132 Vorträge aus, die über folgende Gebiete der Simulation berichteten:

- Simulation von Rechensystemen
- Schaltkreissimulation
- Simulation in energieerzeugenden und energieverteilenden Systemen
- Simulation in Verfahrenstechnik und -planung
- Simulation in betriebswirtschaftlichen Anwendungen
- Simulationshardware
- Modellbildungs- und Softwaremethodik
- Simulation in Biologie und Medizin
- Simulationssprachen und Simulationssoftware für kontinuierliche und diskrete Systeme
- Fahrzeug- und Flugsimulation
- Simulation in Ökologie
- Simulation in technischen Anwendungen

Ferner gab es eine umfassende Präsentation von Simulationssprachen, wobei zuerst die Sprachen ACSL, GASP, SLAM, GPSS-FORTRAN und LISP-Maschinen in Kurzvorträgen vorgestellt wurden. Anschließend erfolgten on-line Simultanvorführungen der vorgestellten Simulationssoft- und Hardware, wobei auch HYBSYS demonstriert wurde.

Ein weiterer Programmpunkt war eine Podiumsdiskussion über Modellvalidierung. Es fanden Arbeitskreissitzungen des Fachausschusses statt.

Bei der ASIM-Mitgliederversammlung wurde ein neuer Vorstand gewählt (Prof.Dr. W. Ameling, Dr. I. Bausch-Gall, Dr. F. Breitenecker, Dr. H. Fuss, Dr. J. Halin, Dr. W. Kleinert, Dr. D. Möller, Prof.Dr. B. Schmidt). Der Fachausschuß 4.5 Simulation der Gesellschaft für Informatik, ASIM, ist eine Vereinigung, die sich der Förderung und der Weiterentwicklung der Simulation in allen Fachrichtungen widmet.

231 Teilnehmer aus 5 europäischen Ländern sowie den USA besuchten die Tagung.

Der Tagungsband ist in der Reihe "Informatik-Fachberichte", Springer Verlag, erschienen (Informatik-Fachberichte, 85, Springer, 1984).

EAI Computer Users' Group Meeting 1984

Vom 17. bis 20. Oktober 1984 fand wieder ein Treffen des International Chapters der EAI Computer Users' Group statt. Diesmal wurde es vom Institut für Dynamik der Flugsysteme der Deutschen Forschungs- und Versuchsanstalt für Luft- und Raumfahrt (DFVLR) in Oberpfaffenhofen, Bundesrepublik Deutschland, organisiert. Die 46 Teilnehmer kamen aus 9 Ländern, darunter auch aus den USA und der Volksrepublik China.

Am ersten Tag erfolgten einführende Vorträge und Vorführungen an der DFVLR. Das Meeting wurde in Herrsching am Ammersee fortgesetzt. Die Firma EAI stellte ihre Simulationsprodukte vor. Die Benützer berichteten über ihre Anwendungen und ihre Probleme.

W. Kleinert berichtete über den geplanten Ersatz des alten Hybridsystems an der TU Wien durch einen SIMSTAR Multiprozessor zusammen mit einem selbstentwickelten digitalen Parallelrechner LATOUR.

Die Working Groups fanden diesmal nicht soviel Anklang wie früher. Angeregte Diskussionen und Interessenaustausch zwischen den Teilnehmern erfolgte während der ganzen Tagung in kleinen Gruppen. Der Vorstand des International Chapters der EAI Computer Users' Group war dieses Jahr neu zu wählen. Es wurden Anne MacKinnon, Universität Glasgow, zum Präsidenten und Irmgard Husinsky, TU Wien, zum Vizepräsidenten und zugleich Editor des Newsletters gewählt.

Simulation eines Kraftwerksmodells am Hybridrechner

Seminar aus Regelungstechnik
Institut für elektrische Regelungstechnik

W. Kausel
H. Stallbaumer
H. Silberbauer

1. Problemstellung

Aufgabe der Seminargruppe war, ein in der Diplomarbeit von Herrn Ehrendorfer /1/ behandeltes digitales Kraftwerksmodell auf dem Hybridrechner zu erstellen und zu testen.

Im Lauf der Donau wurden in den letzten Jahrzehnten eine Reihe von Kraftwerken errichtet. Alle diese Kraftwerke haben eine eigene Regelung, üblicherweise einen Prozeßrechner. Da der Abstand der Donaukraftwerke nicht allzu groß ist, beeinflussen die einzelnen Kraftwerksregler einander. Das Ziel der Diplomarbeit von Herrn Ehrendorfer war nun, diese Zusammenhänge zu untersuchen. Dazu wurde auf einem Digitalrechner PDP 11 die Kraftwerkskette aller Donaukraftwerke simuliert. Die Ergebnisse sind in der Diplomarbeit nachzulesen.

In der Seminararbeit sollte nun von einem dieser Kraftwerke (Ottensheim) eine analoge Simulation erstellt werden. Zweck dieser Simulation war, Vergleiche zwischen analoger und digitaler Simulation anzustellen, etwa von Genauigkeit und Geschwindigkeit, aber auch Leichtigkeit der Modelluntersuchung (Parametervariation).

Dabei muß gleich darauf hingewiesen werden, daß Modelle dieser Größenordnung die Kapazität des Hybridrechners bezüglich der verfügbaren Analogrechenelemente bis in die Grenzbereiche nutzt, ja sogar einige komplexe Elemente durch Kombination mehrerer einfacher zusätzlich geschaffen werden mußten.

Die Funktionsgruppen bestehen aus Pegelabsenkung, Störgrößenaufschaltung, Wendepegel, Sollwertintegrator, Kraftwerkssolldurchfluß, Turbinen- und Wehrregelung, Oberwasserpegelfilter, Ober- und Unterwasserpegel (siehe Bild 1).

Zwei der Hauptschwerpunkte in der Problemstellung waren die Turbinenregelung einerseits und die doppelt geschlossene Regelschleife andererseits, die Turbinenregelung durch ihre Komplexität und die zweifache Regelschleife aufgrund von Stabilitätsproblemen, die ja auf einer analogen Rechenanlage durch Ungenauigkeit noch kritischer sind.

Aus ähnlichen Gründen wurde auch die Einführung eines Offsets der Meereshöhe notwendig, denn Veränderungen von ca. 0.5% des Oberwasserpegels (was einem Pegelsprung der Donau von 1 Meter entspricht) bewirken bei mehrfach geschlossenen Regelschleifen nicht mehr solch signifikante Änderungen, die eigentlich zu erwarten wären.

Weiters mußten zwei Funktionsgruppen - nämlich Oberwasser- und Wendepegel - umgebaut werden, denn eine Integration der Zu- und Abflüsse vor der Simulation sind natürlich an einem Analogrechner nicht sinnvoll, in der realen Welt entspräche das einem Stauen der Donau bei den Zuflüssen und erzeugt natürlich irgendwann einen Operating-Range-Overflow an einem analogen Element.

Prinzipiell kann dazu gesagt werden, daß ein auf einem Analogrechner abgesetztes Modell der realen Welt näher ist als die Simulation auf einem Digitalrechner, da die Nachbildung realer Vorkommnisse durch elektrische Schaltkreise der Wirklichkeit weit näher ist als die Simulation zu festgesetzten Abtastzeitpunkten durch Funktionen mit digitalen Variablen, welche unabhängig von der in der realen Welt vorkommenden Größen i.a. den gleichen Operationsbereich haben.

2. Realisierung des Modells am Hybridrechner

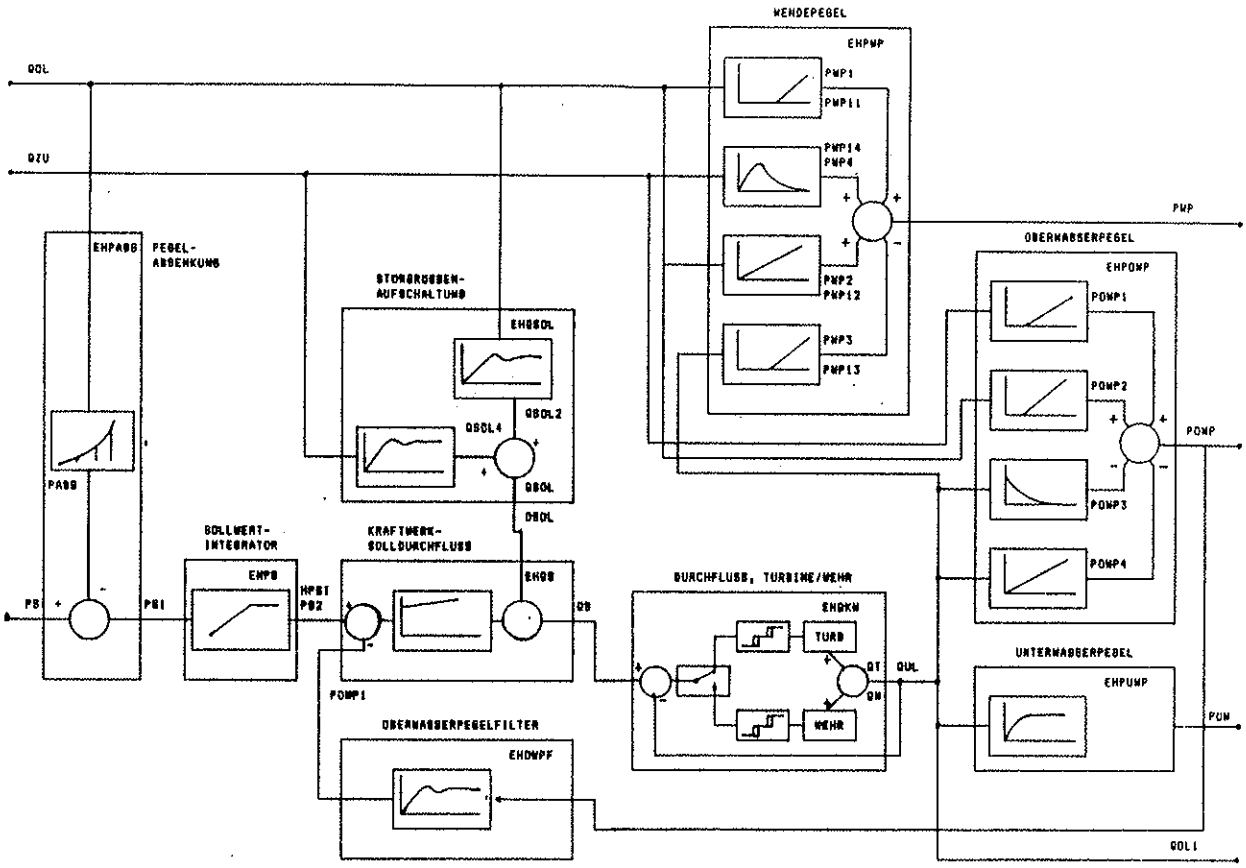


Bild 1
Blockdiagramm des Kraftwerksmodells

Das Modell wurde unter Verwendung der Simulationssprache HYBSYS auf dem Hybridrechner abgesetzt.

Besonderes Gewicht muß auch auf die absoluten Zeitverhältnisse am Analogrechner gelegt werden. Durch Totzeiten und Zähler sind die absoluten Zeitverhältnisse nur mehr sehr eingeschränkt variabel bestimmbar.

Totzeitglieder stellen in der Regelungstechnik oft benutzte Elemente dar, welche die in der realen Natur vorkommenden Verzögerungen bei Prozessen simulieren sollen. Da diese Elemente im allgemeinen durch spezielle Differential-gleichungssysteme (sogenannte Allpässe) angenähert werden, bei einem Problem dieser Größenordnung jedoch nicht die Elemente für alle notwendigen Totzeitelemente vorhanden sind, wurde die Möglichkeit des Schreibens eines Useroverlays unter HYBSYS benutzt, um die für den Prozeßablauf wichtigen Verzögerungen simulieren zu können.

Der Useroverlay besteht aus einem FORTRAN-Programm, welches bei der Durchführung eines hybriden Runs unter HYBSYS abläuft und über A/D-Wandler die Totzeiteingangswerte einliest, sie entsprechend der Totzeit verzögert und sie schließlich über spezielle D/A-Wandler (sogenannte DACFs) wieder in den analogen Prozeß einspeist.

Dieses FORTRAN-Programm verwendet wiederum bereits vorhandene HYBSYS-Systemroutinen zum Ein- und Auslesen während eines analogen Runs. Nach dem Übersetzen wird das Programm mittels spezieller Routinen als Overlay gebunden und ist damit ein fester Bestandteil des vom User benutzten HYBSYS-Systems. Mittels eines eigenen Aufrufes wird es dann während des Runs aktiviert.

Aufgrund beschränkter Elementanzahl konnten nur vier Totzeitglieder realisiert werden, zur Simulation bot sich dadurch das Kraftwerk Ottensheim an. Ergänzend muß aber erwähnt werden, daß Totzeiten in der Größenordnung von 40 Sekunden

(in der Turbinenregelung) bei einer Laufzeit von 85 Stunden als simulierte Periode keinen wesentlichen Einfluß mehr haben können. Überhaupt scheinen die Totzeiten in der realen Praxis - die längste hat 1600 Sekunden - relativ wenig Einfluß auf das Problem zu haben, zumal die Donau ja auch in der realen Welt keine Sprünge macht wie in der zur Veranschaulichung des Regelsystems eingespeisten Sprungantwort.

Die Skalierung eines am Hybridrechner abgesetzten Problems ist im allgemeinen durch einen einzigen Befehl erledigt, welcher den Skalierungsalgorithmus startet. Durch Variation der Parameter findet dieser dann automatisch die für die Analogrechenelementeingänge richtigen Parametereinstellungen mittels mehrerer analoger Runs heraus.

Bei Regelschleifen mit vielen einander beeinflussenden Größen, Regelschleifen mit sehr vielen Elementen und mehrfach geschlossenen Regelschleifen kann es jedoch zu nicht skalierbaren Problemdefinitionen kommen, bisweilen sind sogar sogenannte Deadlocks - zum Beispiel beim Pendeln von Einer- auf Zehnereingang bei einem DCA - möglich. Die Ursache dafür sind entweder schwach stabile Systeme oder Rechnerungenauigkeiten oder beides.

Bei diesem speziellen Kraftwerksmodell war auch eine automatische Skalierung nicht möglich, wodurch ein "händisches" Skalieren notwendig wurde. Dieses nahm den Hauptteil der Zeit in Anspruch, denn in einem nicht bekannten Regelsystem ist dies eine Arbeit mit sehr viel Feingefühl, um ein optimal funktionierendes Modell zu erhalten.

3. Ergebnisse

Die Parametervariation erfolgte großteils analog zu /1/, um vergleichbare Ergebnisse zu erzielen. Die hybride Parametervariation eines einmal abgesetzten und getesteten Modells ist sehr einfach, schnell und komfortabel; vielerlei Möglichkeiten zur Erstellung von graphischen Darstellungen stehen zur Verfügung.

Es wurden im Rahmen der Genauigkeit ähnliche Ergebnisse wie in /1/ erzielt.

Zum Vergleich seien die Vor- bzw. Nachteile der hybriden Simulation angeführt. Vorteilhaft wirkten sich die naturnähere Modellbildung am Hybridrechner aus, der sehr mächtige Sprachprozessor HYBSYS, welcher viel Kleinarbeit abnimmt, sowie die überaus schnelle Parametervariation (bei diesem Modell im Multiuserbetrieb etwa 3-5 Sekunden pro Run: zum Vergleich: die PDP 11/23 benötigte pro Kraftwerk ca. 8 Minuten). Nachteilig bei der Problemlösung am Hybridrechner sind kleine Ungenauigkeiten durch Rechnerelemente sowie bei nicht skalierbaren Problemen ein relativ hoher Skalierungsaufwand in der Setup-Phase.

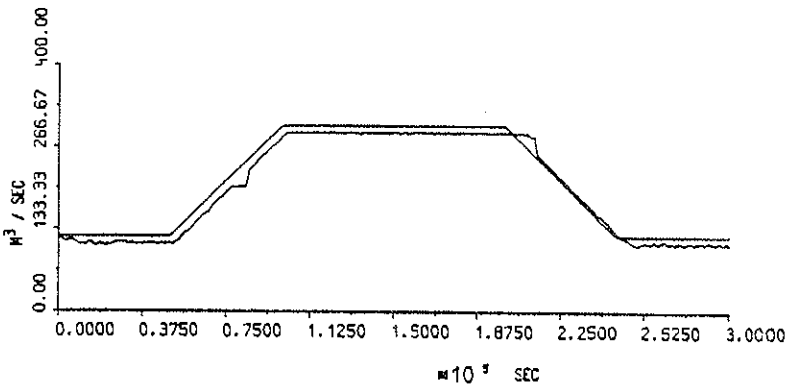


Bild 2
Wehrregelung bei Flutwelle

Die in Bild 2 gezeigte Simulation nimmt eine Flutwelle an, die auf das Kraftwerk zuläuft. Die Welle hat eine Änderung der Wassermenge von 1200 auf 2800 Kubikmeter pro Sekunde zur Folge. Die Form ist aus dem Bild ersichtlich. Man sieht, daß bis etwa 1800 m^3/s das Turbinenschluckvermögen ausreicht. Dann werden die Wehre abgefahren. Die Zacke im Bild entsteht dadurch, daß die Wehre ursprünglich einen Meter über dem Wasserpegel stehen und das Abfahren der Wehre einige Zeit braucht, währenddessen durch die Turbinen noch mehr Wasser hindurch kann.

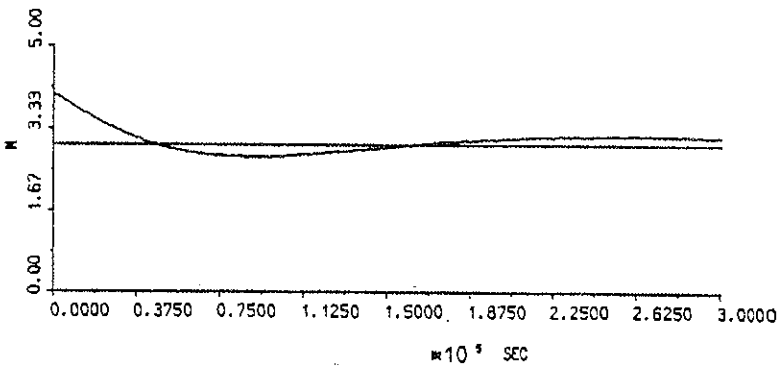


Bild 3
Wasserpegel bei Sollwertsprung

Bild 3 zeigt die hypothetische Änderung des Wasserpegels im Staubecken, wenn der Pegelsollwert der Regelung um einen Meter gesenkt wird. Wie man sieht, wird PT2 Verhalten erzielt.

Literatur

1. W. Ehrendorfer: Simulation des dynamischen Verhaltens und der Steuerung einer Laufkraftwerkskette. Diplomarbeit, Technische Universität Wien (1984)

Simulation lagegeregelter Stellantriebe mit ausgeprägten nichtlinearen Eigenschaften

S. Galeev
Hochschule für Flugzeugwesen, Ufa, UdSSR
W. Kleinert
Hybridrechenanlage, Technische Universität Wien

1. Problemstellung

Die Funktion lagegeregelter Stellantriebe besteht in der koordinierten Realisierung von Bewegungsvorgängen in automatisch arbeitenden Maschinen (Werkzeugmaschinen oder Industrieroboter mit numerischer Steuerung). Bei der Auslegung solcher Antriebe muß auf besondere Eigenschaften Rücksicht genommen werden /1,2/.

Eine Lageregelung im Stellantrieb mit Gleichstrommotoren wird üblicherweise als Kaskadenregelung ausgeführt. Bild 1 stellt einen lagegeregelten Vorschubantrieb mit unterlagerter Drehzahl- und Stromregelung und vereinfachtem schwingungsfähigem Übertragungssystem dar. In der Lageregeleinrichtung wird der vom Lagemeßsystem ermittelte Lage-Istwert ϕ_m (bei der indirekten Lagemessung) oder ϕ_l (bei der direkten Lagemessung) mit dem Lage-Sollwert ϕ_s verglichen. Beim proportionalen Verhalten des Lagereglers erzeugt dieser in Abhängigkeit von der Lagedifferenz den Sollwert ω_g für den Drehzahlregelkreis.

Die Vorgabe des Stroms erfolgt durch einen Drehzahlregler. Die Begrenzungen dieses Blocks bestimmen die Extremwerte $\pm I_{amax}$ des Stroms (und auch der Beschleunigung).

Beim Anfahren kommt der Drehzahlregler in jedem Fall zunächst an seine Begrenzung, sodaß stets mit der maximal zulässigen Beschleunigung angefahren wird. Jeder Bremsvorgang des Systems wird immer in der richtigen Entfernung vom Zielpunkt eingesetzt, sodaß Bremsen mit der vollen Verzögerung erfolgt und bei Nullgeschwindigkeit die gerade vorgegebene Zielposition erreicht wird.

Für lagegeregelte Stellantriebe numerisch gesteuerter Arbeitsmaschinen ist es notwendig, das mechanische Übertragungssystem in die Betrachtung einzubeziehen /3,4/. Bei Industrierobotern führen der konstruktive Aufbau und die hohen Arbeitgeschwindigkeiten zu weitaus größeren Schwierigkeiten bei der Bahnerzeugung als z.B. bei Werkzeugmaschinen. Hier können die mechanischen Übertragungsglieder nicht so steif ausgelegt werden wie an Werkzeugmaschinen (die Ursache liegt

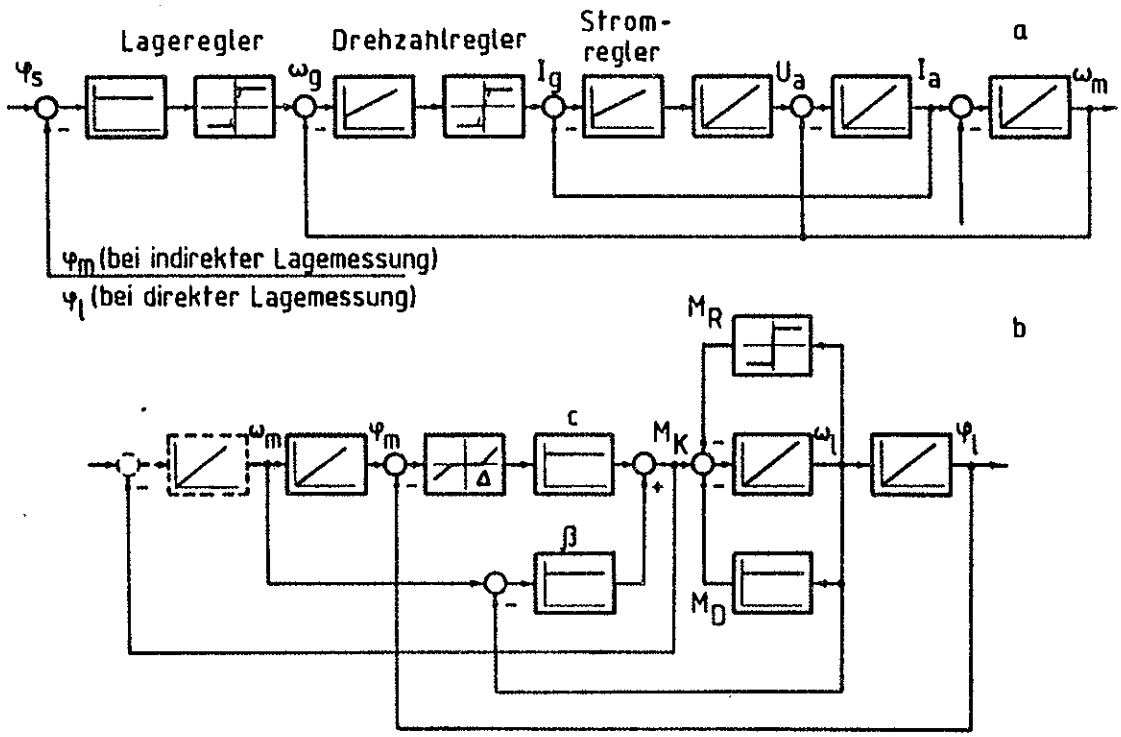


Bild 1
Strukturplan eines Lageregelkreises an einem schwingungsfähigen Zweimassensystem
a) elektrischer Antriebsteil
b) mechanischer Antriebsteil

in der Nachgiebigkeit des Roboterarms und in der Elastizität des Harmonie-Drive-Vorschubgetriebes).

In erster Linie sind drei wesentliche Eigenschaften mechanischer Systeme zu beachten: die Elastizität der Drehmomentübertragung, die Lose in Kupplungen und Getrieben und die Reibung. Es ist praktisch möglich, die Parameter typischer Antriebssysteme in der Annahme eines elastisch gekoppelten Zweimassensystems mit ausreichender Genauigkeit zu bestimmen.

Die Beschreibung des dynamischen Verhaltens dieses lagegeregelten Stellantriebs ist mit Hilfe nichtlinearer Differentialgleichungen möglich. Auf der Grundlage dieses mathematischen Modells werden die Einflüsse charakteristischer nichtlinearer Eigenschaften der einzelnen Übertragungsglieder auf das Regelverhalten des Antriebs durch Hybridsimulation unter Verwendung der Simulationssprache HYBSYS untersucht und davon ausgehend Möglichkeiten zur Verbesserung des Übergangsprozesses mit der klassischen Regelstruktur gesucht.

2. Simulationsmethode

Die Programmierung auf dem Hybridrechner erfolgt in bekannter Weise über Differentialgleichungssysteme und einen Strukturplan, wie in /5/ ausführlich beschrieben. Die Optimierung aller Regler erfolgt nach der Frequenzgangmethode. Die Switch-Operatoren bestimmen die Begrenzungen der Lage- und Drehzahlregler. Im Modell wird die Lose Δ als Totzone berücksichtigt.

Für $|\phi_m-\phi_1|>\Delta$ besteht Kraftschluß und für das von dieser Kupplung mit der Federkonstanten c und dem Dämpfungswert β übertragene Drehmoment gilt die Gleichung

$$M_K = c(|\phi_m-\phi_1|-\Delta) + \beta(\omega_m-\omega_1)$$

Im Fall $|\phi_m-\phi_1|<\Delta$ sind Antriebs- und Arbeitsmaschinen entkoppelt. Diese Funktion wird mit Schaltern und Komparatoren realisiert. Die Art der Darstellung der trockenen Reibung M_R als Signumfunktion und des geschwindigkeitsproportionalen Anteils M_D als Rückführkonstante ist bekannt /4/.

Die Simulation erfolgt in nachstehenden Schritten:
- schrittweise Änderung der mechanischen Eigenschaften bei unveränderter Reglereinstellung;
- Verbesserung des Übergangsverhaltens durch die Änderung der Reglereinstellung und Kompensationsmaßnahmen.

Die Übergangsvorgänge wurden für sprung- und rampenförmige Sollwertänderungen aufgezeichnet.

3. Einige Ergebnisse

Hier werden einige Simulationsergebnisse der Stellantriebe einer Arbeitsmaschine mit einem Proportional-Regler für den Lage- und Drehzahlregelkreis und einem Proportional-Integral-Regler für den Stromregelkreis wiedergegeben. Die wichtigsten technischen Daten der Baugruppen des Lageregelkreises sind in der Tabelle dargestellt.

Motordaten		
Nennmoment	14.7	Nm
Nennstrom (I_{aN})	20.7	A
Nennankerspannung (U_{aN})	104	B
Nennwinkelgeschwindigkeit (ω_{mN})	104.7	rad/s
max. Strom ($\pm I_{amax}$)	200	A
Trägheitsmoment	0.04	kgm^2
elektrische Zeitkonstante	0.004	s
mechanische Zeitkonstante	0.04	s

Mechaniksystemdaten		
Ersatz-Trägheitsmoment der mechanischen Übertragungsglieder	0.04	kgm^2
Federkonstante (c)	10000	Nm/rad
Dämpfungswert (β)	0.2	Nms/rad
Lose (Δ)	0.01	rad
trockene Reibung (M_R)	1	Nm
geschwindigkeitsproportionale Reibung (M_D)	1	Nm

Im Hybridrechenprogramm werden folgende Bezeichnungen eingeführt:

ϕ_s - FS (rad); ϕ_m - FM (rad); ϕ_1 - FL (rad)

ω_g - WG (rad/s); ω_m - WM (rad/s); ω_1 - WL (rad/s)

I_a - IA (A); I_g - IG (A)

$(\phi_m-\phi_1)$ - PHIML (rad); Δ - PHIO (rad)

Bild 2 zeigt den Verlauf der Parameter FL und WL über der Zeit für eine Bewegung des Schlittens auf 100 rad (FS = 100 rad). Die Positionierungszeit beträgt etwa 1 s. Das Reibmoment und die Lose bewirken eine Totzeit beim Anfahren und einen Schleppfehler. Die Bilder 3 und 4 zeigen den Einfluß der Ausgangsgrößenbegrenzung der Lage- und Drehzahlregler (WG und IG) auf die Regelparameter WM und IA.

Der Einfluß der Lose äußert sich bei unbelastetem (M_R=1 Nm) und dämpfungsarmem (β=0.2 Nms/rad; M_D=1 Nm) Antrieb in stabilen Grenzschwingungen um die Sollage und kann bei großen Δ-Werten zur Instabilität führen. Das ist vorrangig bei Rückführung der Motordrehzahl der Fall (Bild 5),

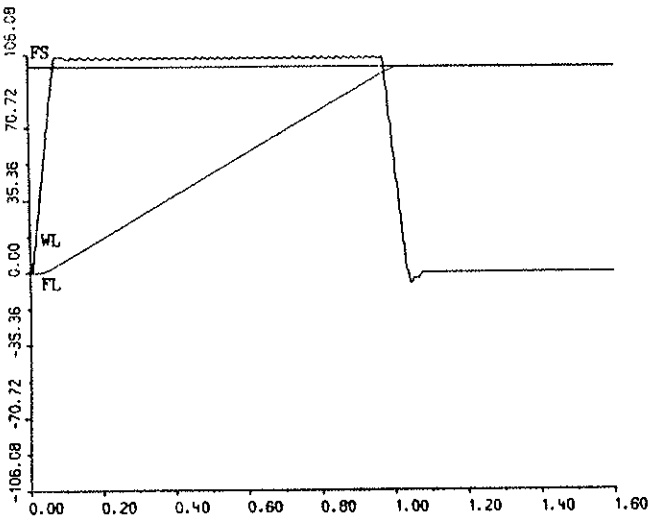


Bild 2

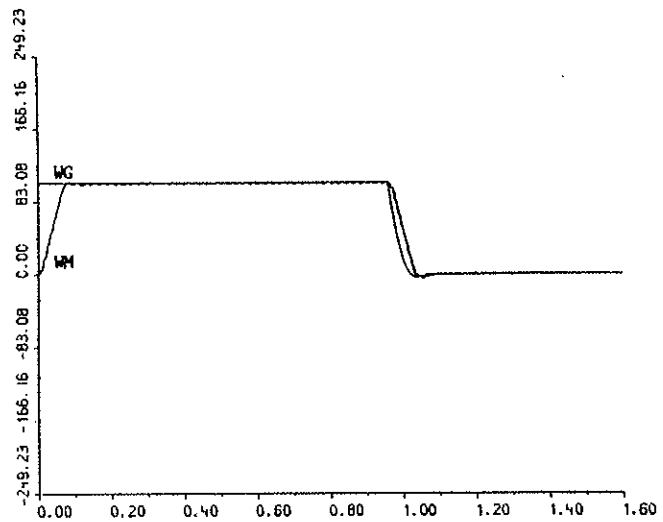


Bild 3

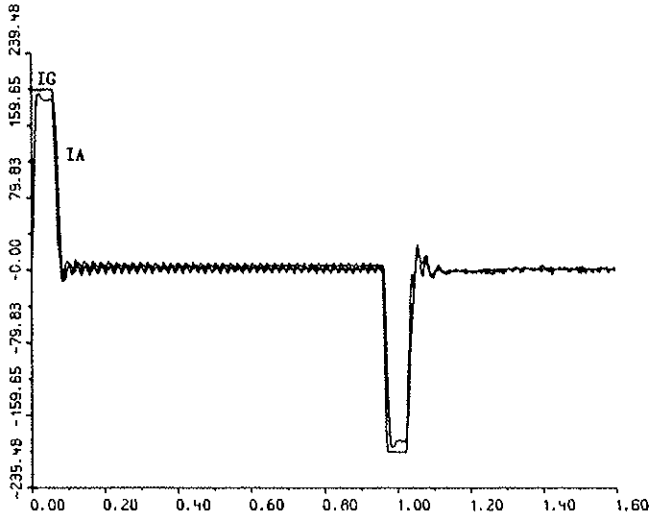


Bild 4

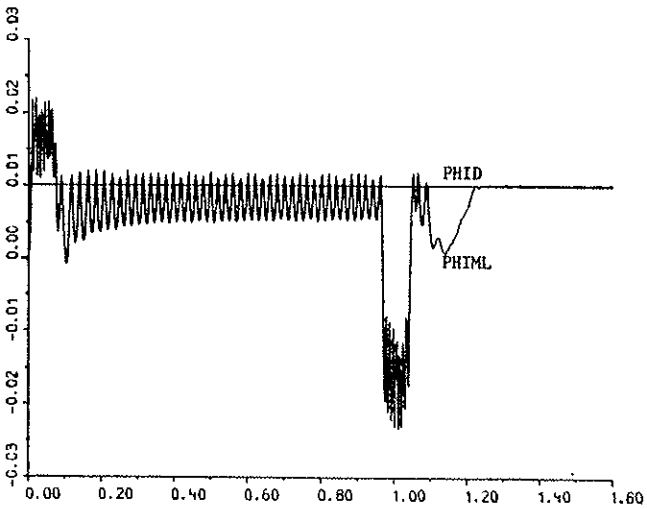


Bild 5

während bei Rückführung der Schlittendrehzahl sich nur stabile Grenzschwingungen aufbilden. Die Lose führt bei dynamischen Vorgängen im reibungsbehafteten elastischen System zu erhöhten Schwingungsamplituden der Schlittendrehzahl WL (siehe Bild 2). Die Auswirkungen der Getriebelose sind durch Veränderung der Reglerparameter nicht zu beseitigen. Die Schwingungen der Schlittendrehzahl und die damit verbundene hohe Beanspruchung des Verbandes Motorwelle - Getriebe - Schlitten kann durch erhöhte Dämpfungswerte β (bis 1 Nms/rad) unterdrückt werden.

Abschließend läßt sich bemerken, daß mit der Simulation an der Hybridrechenanlage der in Bild 1 gezeigten "klassischen" Regelkreisstruktur mit ausgeprägten nichtlinearen Eigenschaften Ergebnisse erzielt werden können, die der Realität sehr nahe sind.

Literatur

1. K. Boelke: Analyse und Beurteilung von Lagesteuerungen für numerisch gesteuerte Werkzeugmaschinen. Springer (1977)
2. G. Pfaff: Regelung elektrischer Antriebe. Methoden der Regelungstechnik. Oldenburg (1982)
3. H. Neundorf: Ermittlung des Übertragungsverhaltens der mechanischen Elemente von Vorschubantrieben. Elektrie 32 H.6, S.292-293 (1978)
4. I.Troch, P.Kopacek, K.Desoyer, F. Breitenecker, F. Rattay: Modulare Simulation von Industrieroboternn. Interface 21 (Juni 1984)
5. D. Solar: HYBSYS User Manual, Version STS. Abt. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien (1984)

Simulation der Glukosekonzentration im Blut*)

F. Breitenecker, I. Troch
Institut für Technische Mathematik
Abt. Regelungsmathematik und Simulations- und
Hybridrechentechnik

Einleitung

In Zusammenarbeit mit einer Forschungsgruppe der Abteilung für Klinische Endokrinologie und Diabetes Mellitus an der I. Medizinischen Universitätsklinik (Prof.Dr. W. Waldhäusl, Doz.Dr. P. Bratusch-Marrain, Dr. M. Komjati) wurden und werden regelungsmathematische Modelle zur Simulation des metabolischen Kreislaufes der Glukose entwickelt. Die Modelle sind "aktive" Modelle, d.h. sie versuchen das Systemverhalten makroskopisch zu beschreiben, indem die Reaktion des Systems auf bestimmte Erregungen studiert wird (regelungsmathematischer Ansatz). Um allerdings eine Verbindung zum physiologischen Hintergrund zu erhalten, wird dabei versucht, die auftretenden Modellvariablen physiologisch interpretierbar zu gestalten. Die beste Vorgangsweise ist hiezu ein Kompromiß zwischen Kompartment-Modellbildung und regelungstechnischer Eingangs-/Ausgangsbeschreibung: das System wird in möglichst wenige Kompartments zerlegt, innerhalb dieser Kompartments (die nun auch durchaus fiktive Modellvariable enthalten können, die in einfacher Weise ein komplexes Verhalten anderer nichtmodellierter Kompartments approximieren) wird das Eingangs-/Ausgangsverhalten mit regelungsmathematischen Methoden beschrieben, die Verbindung zwischen diesen "verallgemeinerten" Kompartments modellieren Bilanzgleichungen. Diese Art der Modellbildung bewährt sich auch zur Beschreibung anderer physiologischer Prozesse (/1/, /2/, /3/, /4/, /6/).

Modellbeschreibung

Der erste Ansatz zur Modellbildung des Glukose-Metabolismus war die Simulation der Glukoseproduktion einer isolierten Rattenleber, die durch Glukagoninfusion angeregt wurde (/3/, /4/, /6/). Dabei standen neben dem zeitlichen Verlauf der Glukagoninfusion als Meßreihen die Glukoseproduktion und ein sogenannter "Messenger", eine Art Enzym, das durch das Hormon Glukagon aufgebaut wird und die Glukoseproduktion (das Freisetzen der in der Leber gespeicherten Glukose (=Glykogen)) in Gang setzt, zur Verfügung . Zur Beschreibung reichten zwei Kompartments aus, ein "Messenger-Kompartment" mit der Glukagondosis als Eingang und dem "Messenger" als Ausgang, und ein "Glukose-Kompartment" mit dem "Messenger" als Eingang und der Glukoseproduktion als Ausgang. Das Verhalten in den Kompartments konnte prinzipiell durch PT1-Glieder (mit Differentialanteil) charakterisiert werden, allerdings stellte sich die Notwendigkeit der Verwendung variabler Verstärkungsfaktoren heraus: höhere Sensibilisierung des "Messenger"-Aufbaus am Beginn der Glukagoninfusion, Desensibilisierung der Glukoseproduktion wegen Abnahme der in der Leber gespeicherten Glukose (Glykogen).

Für die Beschreibung der Glukosekonzentration im menschlichen Blutkreislauf wurde nun als Basis auf dieses isolierte Modell zurückgegriffen. Allerdings stand als Meßreihe aus physiologischen Gründen nur die Glukosekonzentration im Blut zur Verfügung. Als Erregung des Systems wurde den Probanden nun nicht allein das Hormon Glukagon infundiert, sondern eine ausgewogene Mischung von Hormonen, die im menschlichen Körper bei Stresseinwirkung ausgeschüttet werden. Das gesuchte Modell sollte damit in "aktiver" Form die Reaktion der Glukosekonzentration im Blut auf Stress beschreiben - sowohl bei gesunden Menschen, wo eine sofort einsetzende Insulinproduktion der Erhöhung des Zuckerspiegels entgegenwirkt, als auch bei Diabetikern, wo der Zuckerspiegel gefährlich hohe Werte erreichen kann.

Zu diesem Zweck wurde nun das vorhandene Basismodell (/3/) erweitert. Zu beachten war, daß die nichtlinearen Einflüsse nun zusätzlich von weiteren Mechanismen bestimmt wurden. Die Glukosekonzentration spiegelt nun nicht mehr direkt die Glukoseproduktion der Leber (Ausschüttung des gespeicherten Glykogens) wieder, sie ist vermindert um die Glukoseutilisation (Energieverbrauch der Muskeln) und wird erhöht durch Glukoneogenese (Neubildung und Abspeicherung

---

*) Dieser Artikel stellt ein Forschungsvorhaben vor, das von der Forschungsgruppe Prof.Dr. W. Waldhäusl, Dr. P. Bratusch-Marrain, Dr. M. Komjati (Abt. für Endokrinologie und Diabetes Mellitus der I. Medizinischen Universitätsklinik, Universität Wien) und Prof.Dr. I. Troch, Doz.Dr. F. Breitenecker (Abt. für Regelungsmathematik und Simulations- und Hybridrechentechnik am Institut für Analysis, Technische Mathematik und Versicherungsmathematik, Technische Universität Wien) durchgeführt wird.

von Glukose in der Leber, bedingt durch die infundierten Hormone). Abbildung 1 zeigt diesen physiologischen Hintergrund.

Die Bildung des "Messengers" ist nun beeinflußt vom Vorhandensein oder Fehlen von Insulin, also von der Stärke der Gegenregulation; das verstärkt bzw. schwächt auch die Sensibilisierung des "Messengers", der nun nur mehr eine fiktive physiologische Größe repräsentiert.

Abbildung 2 zeigt nun ein Blockschaltbild für dieses neue Modell, bei dem die erwähnten Einflüsse durch (nichtlineare) Beobachter approximiert wurden (diese Vorgangsweise bewährt sich prinzipiell bei der Modellierung physiologischer Prozesse (/2/). Das "Messenger"-Kompartment beschreibt mit Hilfe eines Verzögerungsgliedes mit Differentialanteil die Bildung des "Messenger"-Signals; der generelle Verstärkungsfaktor ist dynamisch (nichtlinear) abhängig von der Sensibilisierung des Systems; die beiden konstanten Verstärkungsfaktoren K_{M1} und K_{M2} charakterisieren den Grad der "Gesundheit" des Patienten: ein stärkeres Durchschlagen des Differentialanteiles (höheres K_{M1}) deutet auf "stärkere" Krankheit ("Diabetes"). Das Glukose-Kompartment bildet die Änderung der Glukosekonzentration abhängig vom "Messenger" mit einem weiteren Verzögerungsglied nach; ein allgemeiner dynamischer Verstärkungsfaktor berücksichtigt dabei die Glukoseutilisation und die Glukoneogenese, beide modelliert mit Beobachtern (Verzögerungsglieder 1. Ordnung mit Zeitverzögerung).

Simulation

Das vorgestellte Modell wurde mit Hilfe der hybriden Simulationssprache HYBSYS (/5/) nicht nur simuliert und identifiziert sondern auch interaktiv entwickelt. Dabei leisteten die Möglichkeiten zur interaktiven Modelländerung sehr gute Dienste. Die nötigen Identifikationen (Berechnung optimaler Modellparameter, die ein Anpassen der Meß- an die Modellkurven gewährleisten) wurden mit Hilfe des HYBSYS-Standard-Makros ZERO durchgeführt (/S/).

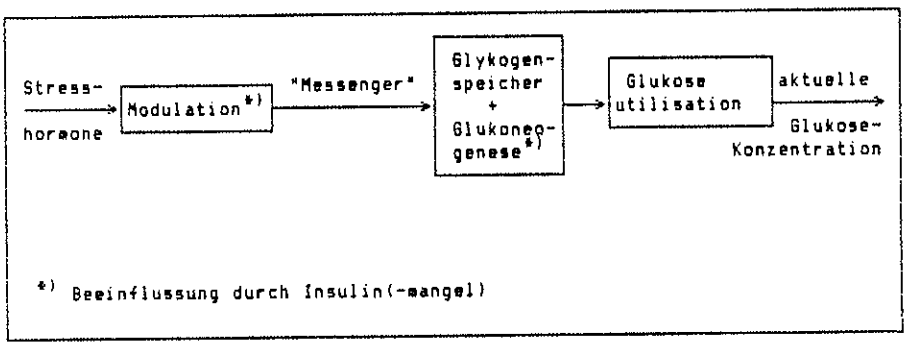


Abbildung 1
Physiologischer Hintergrund

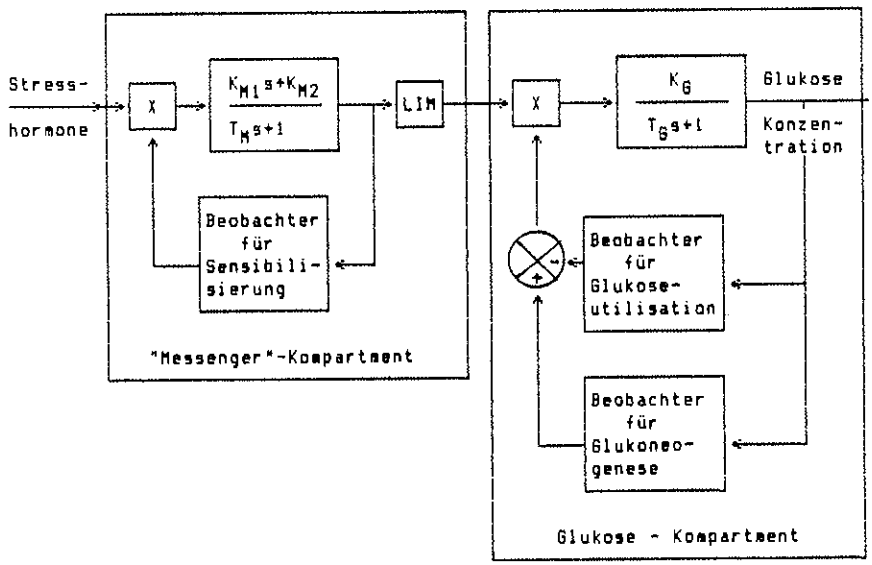
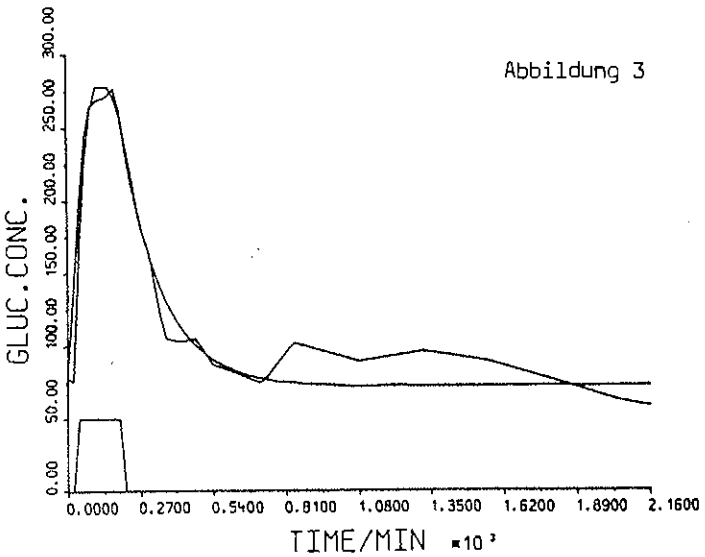


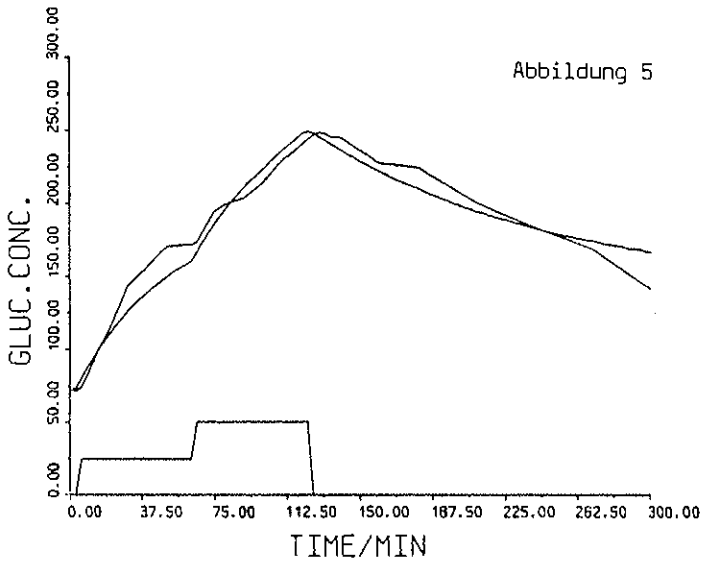
Abbildung 2
Blockschaltbild für mathematisches Modell

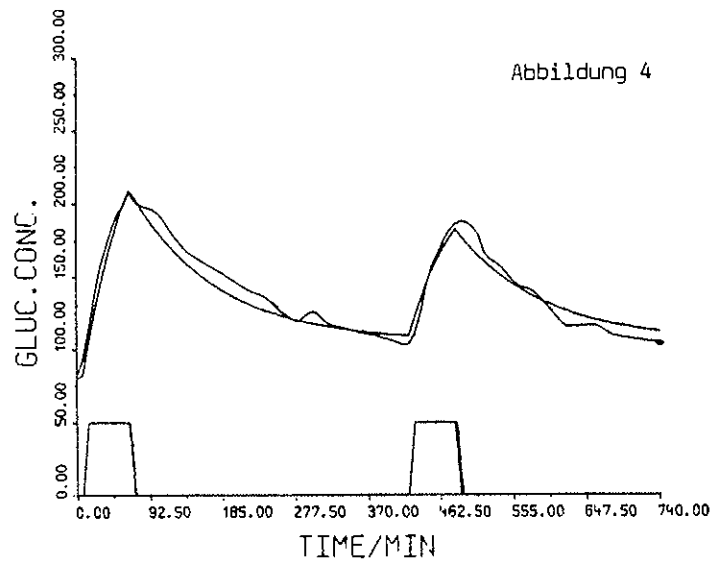
Ergebnisse

Wesentlich war auch eine "wechselseitige" Beeinflussung von Modellentwicklung und Experimentplanung (Gewinnung neuer Meßreihen). Im einzelnen wurden dabei folgende Schritte durchlaufen:

1. - Modellentwicklung für Meßreihen mit "einfachen Expositionen" (einfacher Stress-Stoß) bei künstlichen (=Labor)-Diabetikern (Ergebnisse in Abb. 3).
   - Vorhersage der Glukosekonzentration für wiederholte einfache Expositionen zur weiteren Experimentplanung.
2. - Modellerweiterung für Meßreihen mit wiederholten einfachen Expositionen (der Stress-Stoß wurde wiederholt, sobald die Erhöhung der Glukosekonzentration auf 1/3 (2/3) des absoluten Erhöhungswertes abgesunken ist) und "treppenförmigen" Expositionen führt zu Beobachtern für Glukoseutilisation und Glukoneogenese (Ergebnisse in Abb. 4 und 5).
   - Vorhersage der Glukosekonzentration für "multiple Expositionen" in unregelmäßigen Abständen und unregelmäßiger Höhe zur weiteren Experimentplanung.
3. - Modellerweiterung für Meßreihen mit multiplen Expositionen bei Gesunden, Diabetikern und künstlichen Diabetikern führt zum Beobachter für die Sensibilisierung des "Messenger"-Kompartments und zur Einführung zweier konstanter Verstärkungsfaktoren, die den Grad der Beeinflussung durch den Diabetes charakterisieren (Ergebnisse in Abb. 6).
   - Vorhersage der Glukosekonzentration für multiple Expositionen zur weiteren Experimentplanung.
   - Vorhersage mit "Bandbreite" bei unterschiedlichen Ruheglukosekonzentrationen (Ergebnisse in Abb. 7).
4. - Identifizierung der neuen und der alten Meßreihen; Trennung der Modellparameter in "Systemparameter", die im wesentlichen gleichbleiben, und "individuelle Parameter", die das individuelle Verhalten einer Person charakterisieren.







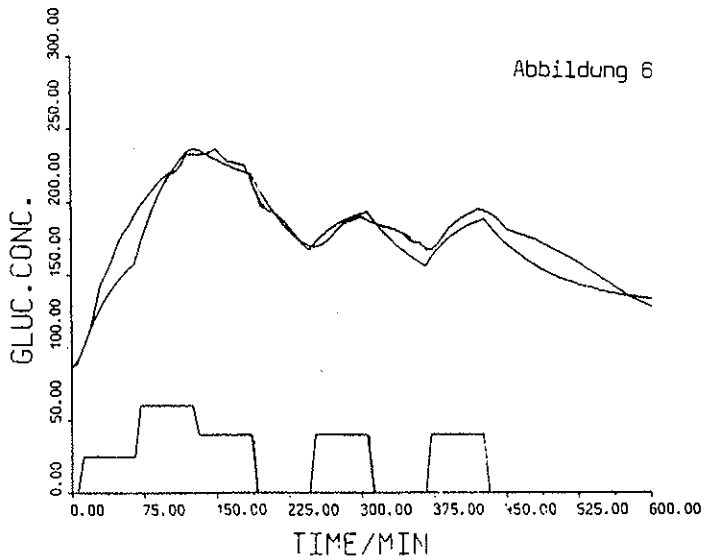


Abbildung 3 - 6
Stressverlauf, gemessene Glukosekonzentration
und identifizierte Modellkurve
für Glukosekonzentration

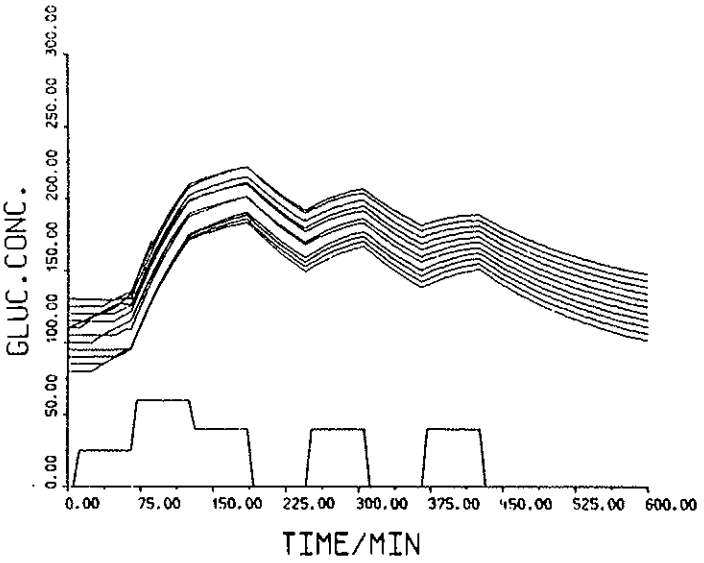


Abbildung 7
Bandbreitenvorhersage für Glukosekonzentration bei einem Diabetiker unter Stresseinwirkung

Derzeit ist die Modellentwicklung weitgehend abgeschlossen. Nach Identifizierung der meisten Meßreihen deutet sich an, daß von ca. 12 Modellparametern sieben "Systemparameter" sind und somit nur fünf Parameter individuell identifiziert werden müssen. Diese beschreiben dann quantitativ die charakteristischen Schwankungen der Glukosekonzentration eines Patienten.

In weiterer Folge ist die Durchführung von ergänzenden Experimenten (ergänzende Meßreihen) und deren Identifizierung geplant, um dann die individuellen Parameter statistisch zu untersuchen (Korrelationen bei Gesunden, Diabetikern, etc.).

Abschließend sei die Bedeutung des mathematischen Modells erwähnt: es beschreibt in vereinfachter Weise die Reaktion der Glukosekonzentration bei Diabetikern nach Stressreizen, ist also (bedingt) in der Lage, den "Tagesablauf" der Glukosekonzentration stressbedingt vorherzusagen. Es sind weitere Modellentwicklungen in Zusammenarbeit mit der Abt. für Endokrinologie und Diabetes Mellitus geplant, um für die Vorhersage dieses "Tagesablaufes" auch statische Kenngrößen zu gewinnen.

Literatur

1. Breitenecker F., Kaliman J.: Simulation von pathologischem Blutdruckverhalten bei Verengung der Aorta. Interface 20 (1983)
2. Breitenecker F.: Modeling of physiological processes by the 'controlled compartment'-approach. Proc. 2nd Int. Symposium 'System Analysis and Simulation', August 26-31, 1985; Berlin, GDR (to appear)
3. Gampe J.: Hepatische Glucoseproduktion in vitro - mathematische Modellierung mittels regelungstechnischer Übertragungsglieder durch Simulation auf einem Hybridrechner. Interface 21 (1984)
4. Komjati M., Breitenecker F., Bratusch-Marrain P., Gampe J., Vierhapper H., Troch I., Waldhäusl W.: Contribution by the glycogen pool and Adenosine 3', 5' - monophosphate release to the evanescent effect of glycogen on hepatic glucose production in vitro. Endocrinology 116, No.3, pp 978 (1985)
5. Solar D.: HYBSYS User Manual, Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien (1984)
6. Troch I., Breitenecker F., Gampe J., Waldhäusl W., Bratusch-Marrain P., Komjati M., Vierhapper H.: Modelling by simulation of hepatic glucose production in vitro. Proc. Int. IMACS Conf. 'European Simulation Meeting on Simulation in Research and Development', August 27-30, 1984, Eger, Hungary (to appear)

FILTERSIMULATION

F. Rattay
Abteilung für Regelungsmathematik,
Hybridrechen- und Simulationstechnik
Institut für Analysis, Technische Mathematik
und Versicherungsmathematik
Technische Universität Wien

Praktikum "Simulation dynamischer Systeme"
Gruppe Gröller, Mondl

Ein einfacher Tiefpaßfilter dämpft hochfrequente Spannungsanteile und wird durch die folgende Schaltung realisiert:

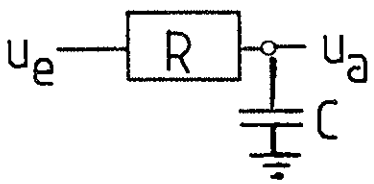


Abbildung 1

Die Beziehung zwischen Eingangsspannung u_e und Ausgangsspannung u_a in Abhängigkeit des Widerstandes R und des Kondensators C gibt die 1. Kirchhoff-Regel bezüglich des Stroms im mit o bezeichneten Punkt:

$$\frac{u_a - u_e}{R} + \dot{u}_a C = 0$$

Mittels einer Zeittransformation läßt sich diese Gleichung vereinfachen zu:

$$\dot{u}_a + u_a = u_e$$

Filter höherer Ordnung zeigen ein besseres Dämpfungsverhalten bei nur schwachem Einfluß auf die Schwingungsanteile mit niedrigen Frequenzen. Sie werden durch größere R,C-Netzwerke realisiert und lassen sich durch ein System von Differentialgleichungen beschreiben:

$$b_1 \ddot{x}_1 + a_1 \dot{x}_1 + x_1 = A_0 u_e$$

$$\vdots$$

$$b_i \ddot{x}_i + a_i \dot{x}_i + x_i = x_{i-1}$$

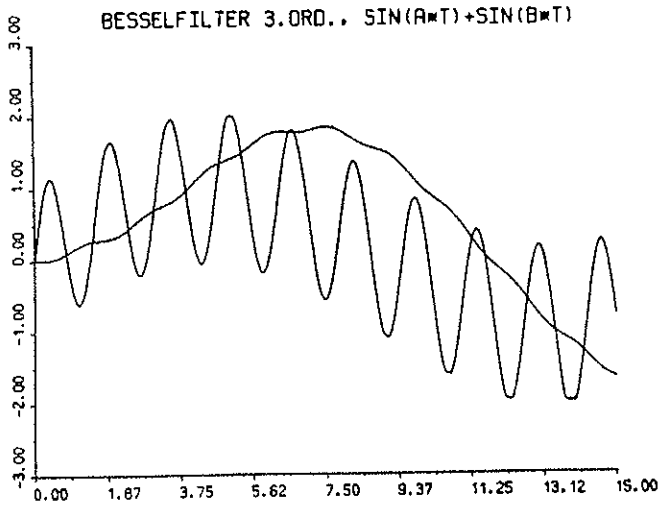
$$\vdots$$

$$b_n \ddot{x}_n + a_n \dot{x}_n + x_n = x_{n-1}$$

$$u_a = x_n$$

Bei gegebener Ordnung lassen sich die Parameter dieser Gleichung so wählen, daß gewisse Eigenschaften des Filters optimiert werden.

Da der Hybridrechner durch Verwenden von HYBSYS und Autopatch sehr effizient zur Modellerstellung sowie zur Parametervariation einsetzbar ist, konnten hier Untersuchungen verschiedener Filtertypen und Ordnungen mit einem wesentlich geringeren Zeitaufwand als auf den analogen Kleinrechnern durchgeführt werden. Zu Vergleichszwecken wurde noch eine digitale Simulation mit ACSL gerechnet. Aber auch hier ist insbesonders die Parametervariation mit einem erheblichen zeitlichen Mehraufwand verbunden.



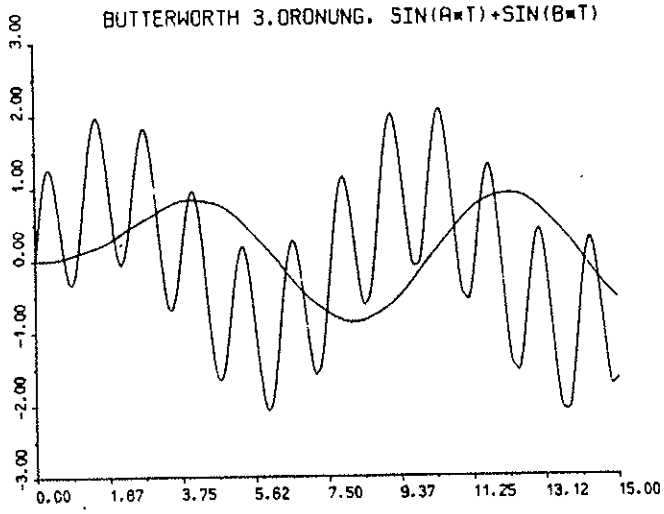


Abbildung 2

Während beim Besselfilter höhere Frequenzanteile noch sichtbar sind, ist die hochfrequente Überlagerung nach dem Butterworthfilter nicht mehr erkennbar.

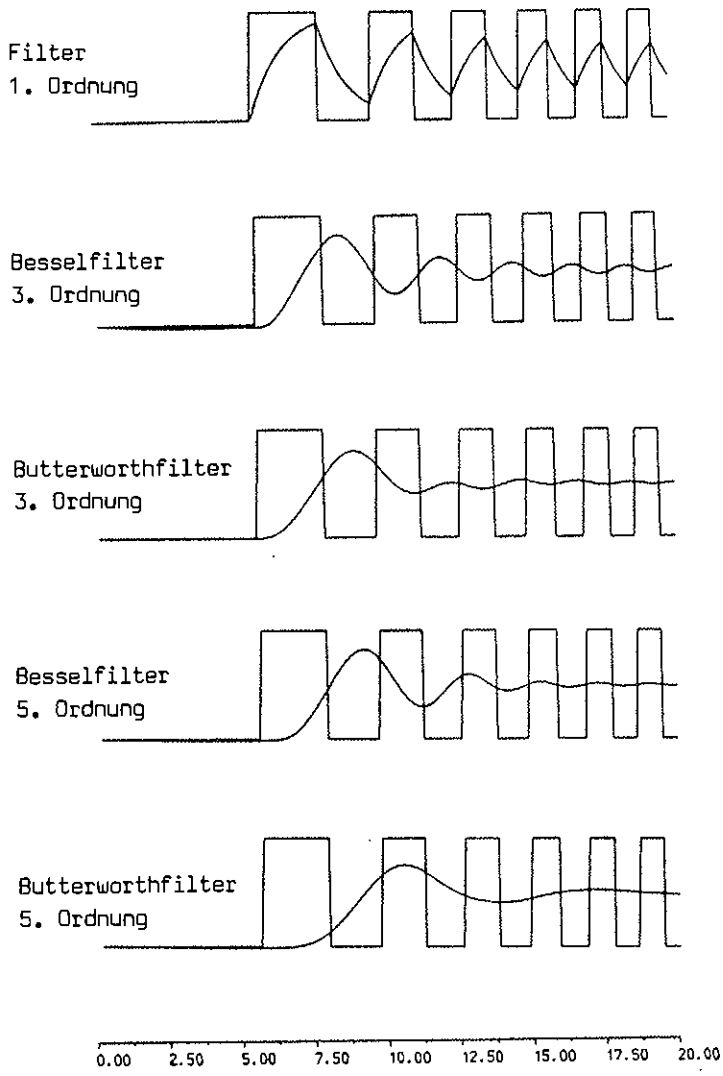


Abbildung 3

Die Zunahme der Frequenz bewirkt nach dem Filtern eine Abnahme der Amplitude. Ein Unterdrücken der Hochfrequenz ist mit größerer Trennschärfe erst durch Filter höherer Ordnung möglich.

INTERFACE Juni 1985