Projekt MUMPITZ

K. Krause, W. Moser, M.T. Schneider,
J. Walter, A. Zimmermann, H.G. Zipperer
Abteilung Computersysteme
Institut für Informatik
Universität Stuttgart
Breitwiesenstr. 20 - 22
7000 Stuttgart 80

Kurzfassung. Im Rahmen der Erneuerung des Grundlagenpraktikums im Informatikstudium der Universität Stuttgart wurde der Mikrocomputer MUMPITZ (Modulare Universelle Mikrocomputer-Platine mit Integrierter Tastatur und Zeilendisplay) entwickelt. Er kombiniert eine übersichtliche und flexible Hardware mit einer umfassenden Basissoftware für den Einsatz in der Ausbildung. Die Benutzeroberfläche ist in besonderer Weise an die Fähigkeiten und Einschränkungen der Hardware angepaßt.

Einleitung

Der Studienplan der Fakultät Informatik der Universität Stuttgart sieht für alle Studenten ein Hardware-Praktikum als Pflichtleistung zum Vordiplom vor. Dieses Grundlagenpraktikum wird üblicherweise im 3. Semester absolviert. Es umfaßt 12 Versuche, die von den Grundlagen der Elektronik (Transistorversuche) über Digitalschaltungen (mit TTL-ICs) bis zur Mikroprozessortechnik reichen.

Die Mikroprozessorversuche beinhalten den Anschluß von Peripheriebausteinen an den Rechner und die Erstellung kurzer Maschinenprogramme für einfache Steuerungsaufgaben, wie z. B. die immer populäre Ampelsteuerung. Hierfür steht bisher ein Einplatinencomputer auf der Basis des 8-bit-Prozessors Intel 8080 zur Verfügung. Er wird über eine Hexadezimal-Tastatur in Maschinensprache programmiert; als Ausgabe dienen 4 7-Segment-LED-Anzeigen. Dieser Rechner wird ebenfalls für das Praktikum 'Aufbau und Einsatz von Mikrocomputern' verwendet, das für Studenten des Schwerpunktes Hardware im Hauptdiplom Pflicht ist und im wesentlichen eine Fortsetzung des Grundlagenpraktikums darstellt.

Nach ca. 10-jährigem Einsatz unter härtesten Bedingungen, nämlich in den Händen von Studenten, die immer weniger Hardware-Kenntnisse ins Studium mitbringen, sind die Rechner weitgehend verschlissen. Als besondere Achillesferse haben sich die 3 verschiedenen Betriebsspannungen des 8080-Chips erwiesen. Fehler beim Anlegen dieser Spannungen führen meist sofort zur Zerstörung des Prozessors. Da dieser nicht mehr hergestellt wird, ist die Ersatzteilbeschaffung ein dauerndes Problem.

Aus diesem Grund beschloß die Fakultät Informatik im Jahre 1991 eine Neuausstattung des Praktikums, verbunden mit einer gewissen Anpassung des Lehrstoffs an den aktuellen Stand der Technik.

Didaktische Überlegungen

Simulation oder Realität?

In neuerer Zeit stellen sich viele Informatiker auf den Standpunkt, Hardware sei für die Informatik nicht relevant; Informatik befasse sich mit Systementwurf, Algorithmen und Datenstrukturen usw. Aus dieser Haltung heraus wird von jenen Informatikern vorgeschlagen, den konkreten Umgang mit Hardware im Praktikum zu reduzieren und durch Logik- und Schaltungssimulation auf PCs zu ersetzen. Als Vorteil führen sie an, daß dabei weniger Zeit für den mechanischen Aufbau von Schaltungen sowie die Suche nach abgerissenen Drähten u. ä. vertan werde, so daß in der selben Zeit wesentlich komplexere und damit interessantere Systeme entwickelt werden könnten. Ferner lehre der Gebrauch von CAD-Werkzeugen strukturiertes Vorgehen und entspreche der industriellen Praxis.

Die Autoren erkennen die Vorteile einer Simulation durchaus an, kennen jedoch aus ihrer täglichen Praxis viele Phänomene realer Hardware, die die Funktion einer prinzipiell korrekten Digitalschaltung verhindern können, aber bei Logiksimulation fast nie erkannt werden. Dazu gehören z. B. Reflektionen und Signalverformungen auf langen Kabeln, Verschiebungen des Massepotentials durch schlechte Masseverbindungen, metastabile Zustände bei Flip-Flops infolge nicht eingehaltener setup- oder hold-Zeit, race conditions bei asynchronen Schaltungen usw.

Ferner zeigt sich oft, daß manche Sonderfälle eines Problems von den Studenten nicht erkannt und folglich in den Entwürfen nicht berücksichtigt werden. Wenn ein Entwickler ein solches grundlegendes Verständnisproblem hat, ist es sehr wahrscheinlich, daß er auch in den Stimuli der Simulation diesen Sonderfall nicht berücksichtigt. Eine solche Schaltung ist das Äquivalent zu jenen Programmen, die einwandfrei arbeiten, solange sie vom Entwickler benutzt werden, aber in den Händen anderer Benutzer bei unerwarteten Eingaben abstürzen. Die harte Realität bringt solche Probleme meist ans Tageslicht.

Aus diesen Gründen wird auch das reformierte Hardware-Praktikum mit realer Hardware arbeiten und nicht zu einem 'Simulatorium' werden.

Inhalte

Die Teilnahme am Hardware-Praktikum ist für alle Studenten der Informatik Pflicht. Das Niveau und der vermittelte Stoff sollten deshalb für ein breites Spektrum an Teilnehmern - vom Theoretiker über den Windows-Programmierer bis zum Elektronikbastler - angemessen sein. Die Autoren sind deshalb der Meinung, daß im Hardware-Praktikum neben allgemeinen Grundlagen der Elektronik

und Digitaltechnik auch praxisbezogene Hardwareprobleme behandelt werden müssen. Das Ziel ist nicht, jeden Studenten zu einem Hardware-Designer auszubilden, aber in jedem Verständnis für die Fähigkeiten und Einschränkungen der Hardware zu wecken, sowie ihm die Fähigkeit zu vermitteln, einfache Probleme der Praxis zu erkennen und zu lösen.

Der Praktikumsrechner soll deshalb nicht nur Lerngegenstand sein, sondern auch Hilfsmittel und Werkzeug, um andere Themen zu behandeln. Für den Einsatz des MUMPITZ im Praktikum sind folgende Schwerpunkte vorgesehen:

- Programmierung in Assemblersprache.
- Anschluß von Peripherie- und Speicherbausteinen.
- Datenkommunikation, Meßwerterfassung und -verarbeitung.
- Programmierung von programmierbaren Logikbausteinen.

Anforderungen

Aus den Erfahrungen des bisherigen Praktikumsbetriebs und den Zielen des reformierten Hardware-Praktikums wurden folgende Forderungen an den neuen Praktikumsrechner gestellt:

- Einfacher und übersichtlicher Aufbau. Die Komponenten des Rechners (CPU, RAM, ROM, E/A) sollen zur Erkennung der Rechnerstruktur als separate, einzelne Bausteine vorliegen.
- Einsatz eines relativ einfachen Prozessors, um die Studenten nicht mit unnötigen prozessorspezifischen Details zu verwirren und vom Lernstoff abzulenken.
- Stand-alone-Betrieb mit integrierter Tastatur und Anzeige. Der Benutzer soll mit dem Rechner arbeiten können, ohne ein Terminal oder einen PC als E/A-Einheit zu benötigen.
- Niedrige Kosten. Die Grundversion des Rechners ohne Tastatur und Display soll weniger als 100.- DM kosten.

Für Studenten, die den Studienschwerpunkt Hardware wählen, wird ein weiteres Praktikum mit dem Titel 'Aufbau und Einsatz von Mikrocomputern' angeboten. Eine Wunschvorstellung der Autoren ist es, daß die Studenten im Rahmen dieses Praktikums den Rechner selbst aufbauen sollen. Da die Kosten für die Bauteile von den Studenten getragen werden müssen, wurde ein Zielpreis von 100.- DM angestrebt. Dies führte zu der Aufteilung des Rechners in die eigentliche Rechnerplatine, die für sich allein einsetzbar ist (z. B. als Steuerung), und die Tastatur- und Anzeigeplatine, deren Bestückung optional ist. Die Rechnerplatine bietet genügend Leistung und Flexibilität, um die o. a. Summe für die Studenten akzeptabel zu machen.

Software

Bedingt durch die heute zu Verfügung stehenden sehr großen Halbleiterspeicher war es möglich, Mumpitz mit einem vollen Speicherausbau zu versehen. Es stehen jeweils 32K RAM sowie 32K ROM zur Verfügung, womit der Adressierraum von den in Frage kommenden 8-Bit Mikroprozessoren vollständig mit Speicher ausgebaut ist.

Die Softwaregrundausstattung im ROM des Mumpitz teilt sich in zwei voneinander getrennte Teile auf:

1. Ein CP/M-BIOS

Es wurden die BIOS-Funktionen des Z80-Betriebssystems CP/M implementiert, die auf einem Rechner mit der Peripherie des Mumpitz sinnvoll sind: Das sind einmal alle Ein- /Ausgabefunktionen, die sich auf die Konsole, d.h auf Tastatur und LCD-Display beziehen (CON:):

- Read Console
- Write Console
- Console Status

Zum anderen die Ein- /Ausgaben über die serielle Schnittstelle (AUX:):

- Read Aux
- Write Aux

Da sowohl die Tastatur als auch die serielle Schnittstelle nur durch Software bedient wird, würde ein direktes Ansprechen durch Anwenderprogramme einen unverhältnismässig hohen Aufwand bedeuten. Das CP/M-BIOS initialisiert das LCD-Display nach dem Reset, es emuliert einige gebräuchliche ASCII-Steuerzeichen, (Cursorbewegung, Teile des Displays löschen u.a.). Ausserdem scannt es die 5 x 6 - Tastaturmatrix, führt die Tastenentprellung durch und setzt die physikalischen Tastenkoordinaten in ASCII-Zeichen um.

Grenzen hat die Emulation an den Stellen, wo das CP/M-System einen 'richtigen' Bildschirm mit z.B. 80 Spalten und 25 Zeilen erwartet. Ebenso besitzt Mumpitz keine vollständige ASCII-Tastatur, was die Texteingabe einschränkt. Es sind keine Floppy-Disk oder Fileoperationen implementiert, obwohl es prinzipiell möglich wäre, an den Prozessorbus ein Floppylaufwerk anzuschliessen, oder über die Parallelports und einen angeschlossenen Hostrechner oder EPROM's ein logisches Floppy-Laufwerk zu emulieren.

Das Ziel und der Maßstab für die Funktion der CP/M-Emulation war es, mit der CP/M-Version des Turbo-Pascal Compilers compilierte Programme in den Mumpitz laden zu können und ausführen zu lassen. Dieses wurde erreicht, und es ist nicht sehr schwierig, in Pascal z.B. einen Taschenrechner mit trigonometrischen Funktionen zu programmieren, der auf Mumpitz ausführbar ist.

Das EPROM bietet sogar soviel Platz, daß es möglich ist, die Turbo Pascal-Run-Time-Library darin zu speichern, und sie bei Bedarf vor das compilierte Programm zu kopieren, so daß nur das reine Anwenderprogramm auf den Einplatinenrechner übertragen werden muß, und nicht das 13K-Byte große Laufzeitsystem.

2. Der Monitor

Beim Entwurf des Monitors mußten neue Ideen verwirklicht werden: Einmal ist das Display leistungsfähiger als bei den sonst üblichen Einplatinenrechnern, die alle nur sechs- oder achtstellige Siebensegmentanzeigen haben, andererseits ist es mit seinen zwei Zeilen und 16 Spalten nicht so leistungsfähig, daß es möglich gewesen wäre, einen der existierenden bildschirmorientierten Monitore oder Debugger darauf zu übertragen. Zudem wird eine komplette ASCII-Tastatur für die Eingabe von z. B. Assembler-Mnemonics benötigt, die beim Mumpitz aber nicht vorhanden ist.

Am unteren Ende der Skala liegen die klassischen Monitore der Einplatinenrechner mit 7-Segmentdisplay, die üblicherweise genau eine Adresse und das darin befindliche Datenbyte anzeigen. Solche Monitore sind sehr einfach und praktisch für jeden Prozessortyp verfügbar, entsprechen aber nicht dem Stand der Technik.

Im Rahmen zweier (konkurrierender) Softwarepraktika wurden deshalb zwei Monitore realisiert, wobei der Hauptschwerpunkt auf der Bedieneroberfläche liegt, die die gegebene Tastatur und das Display möglichst effizient nutzen soll.

Zur Aufgabenstellung gehörten insbesondere drei Punkte:

A. Wegen des kleinen Displays ist es notwendig, sehr häufig in dem interessierenden Programm vorwärts und rückwärts 'scrollen' zu können. Dabei ist ein spezielles Problem zu lösen: Bei allen Prozessoren mit variabler Befehlslänge ist, wenn lediglich der Befehlszählerstand gegeben ist, prinzipiell nicht entscheidbar, an welcher Adresse der vorhergehende Befehl beginnt. D. h., beim 'rückwärtsscrollen' weiß der Disassembler nie, ob das vor dem gegenwärtigen Opcode liegende Byte der Adressteil eines vorangehenden Mehrbytebefehls ist, oder der Opcode eines Einbytebefehles.

B. Da genügend Platz im ROM zur Verfügung steht, um einen Inlineassembler unterzubringen, und man heute niemandem mehr zumuten kann, Opcodes als Hexadezimalzahlen einzugeben, waren Konzepte zu entwickeln, die es gestatten, auch ohne ASCII-Tastatur effiziente Texteingabe durchzuführen.

C. Die Benutzeroberfläche soll soweit selbsterklärend sein, daß es möglich ist, Mumpitz auch ohne intensives Studium eines Manuals einfach und effizient zu bedienen.

Als Ergebnis der beiden Sopras entstanden zwei Monitore, die jeweils etwa 16KBytes groß sind. Sie bieten beide in etwa die gleichen Features, und haben eine durch Cursortasten gesteuerte Menü-Oberfläche. Es ist jedoch auch möglich, die einzelnen Menüpunkte direkt über Hot-Keys anzuspringen.

Beispielsweise erfolgt die Eingabe eines Opcodes nach Druck einer Taste mit einem Menü, da ja keine ASCII-Tastatur vorhanden ist:

LCJAOSRIDPXBENH

Dies sind die Anfangsbuchstaben der existierenden Z80-Mnemonics nach ihrer Häufigkeit geordnet. (L für LD, C für CALL, J für JP sind die wichtigsten). Der Cursor wird auf den gewünschten Anfangsbuchstaben bewegt, die Taste CR bestätigt diesen Buchstaben, und es erscheint eine neue Buchstabenleiste, die jetzt alle Buchstaben enthält, die als zweite zu dem schon gewählten ersten Buchstaben in Frage kommen. Das Verfahren wird solange fortgesetzt, bis der Opcode eindeutig festliegt. Danach werden auf die gleiche Art die Operanden des Befehls eingegeben.

Korrekturen bei der Eingabe sind jederzeit mit der Taste 'U' (Undo) möglich, die jeweils wieder in die vorhergehende Buchstabenzeile zurückführt. Ist der Befehl komplett eingegeben, erscheint in der rechten oberen Ecke des Display das Zeichen '#' und mit ↓ kann zum nächsten Opcode weitergegangen werden.

Diese Art der Eingabe hat zwei Vorteile: Es ist nicht möglich, syntaktische Fehler zu machen, da der Assembler von vorneherein nur legale Buchstaben zur Eingabe anbietet. Außerdem sind nur relativ wenige Tastendrücke notwendig, weil jeder Befehl sofort komplett ergänzt wird, sobald er durch seine Anfangsbuchstaben eindeutig festgelegt ist: Aus CA wird z.B. automatisch CALL gebildet.

Es ist auch möglich, ein Programm Maschinenbefehl für Maschinenbefehl ausführen zu lassen (Trace-Modus), und nach jedem Befehl den nächsten auszuführenden Befehl in disassemblierter mnemonischer Form auf dem Diplay darzustellen, sowie die Registerinhalte anzuzeigen und zu modifizieren. Bei der großen Anzahl der Z80-Register (mehr als 20) ist es nicht möglich, alle Register gleichzeitig darzustellen, so daß man mit den Cursortasten zwischen einzelnen Registergruppen hin- und her scrollen kann. Innerhalb dieser Gruppen können einzelne Register ausgewählt und modifiziert werden. Weiter ist es möglich, Breakpoints zu setzen, und ein Programm bis zu einer gewünschten Stelle frei ablaufen zu lassen und ab dort in den Einzelschrittmodus überzugehen, sowie nur ein Unterprogramm am Stück ausführen zu lassen und nach dem Ende des Unterprogramms wieder mit dem Einzelschrittmodus fortzufahren.

Dieses sind nur zwei besonders wichtige Beispiele, an denen gezeigt werden kann, wie das gegebene Display und die Tastatur ergonomisch optimal ausgenutzt werden kann. Darüber stellt der Monitor noch eine Vielzahl weiterer Utilities zur Verfügung, zu deren genauer Verwendung auf die Bedienungsanleitung des Monitors verwiesen sei.

Die Speicherbelegung des Z80-Mumpitz (alle Adressangaben sind Hexadezimal) ist in der folgenden Tabelle aufgelistet:

Adresse	Funktion	
0000 - 00FF	RAM	reseviert für CP/M und Monitor
0100 - 6BFF	RAM	frei
6800 - 7FFF	RAM	reserviert für Monitor
8000 - 85FF	ROM	CP/M-BIOS + V24-download-Programm
8600 - 8D1F	ROM	Disassembler
8D20 - ABFF	ROM	Turbo-Pascal Runtime-Library
AC00 - BFFF	ROM	frei
C000 - FDFF	ROM	Monitor
FE00 - FEFF	I/O	Memory mapped I/O
FF00 - FFFF	ROM	frei

Schaltungsbeschreibung

Die Gesamtschaltung von MUMPITZ ist in Arbeitsspeicher (RAM), Programmspeicher (EPROM), Prozessor (CPU) und Ein-/Ausgabe (EA) gegliedert. Der Adreßbus (A0-A15), der Datenbus (EA0-EA7) sowie die Kontrollsignale Instruction Fetch (IFETCH), E/A Select (EA-SEL) und Read/Write (R/W) sind über Treiberstufen auf rote, grüne bzw. gelbe Leuchtdiodenreihen herausgeführt. Dies ist vor allem im Single-Step Betrieb nützlich, um die Arbeitsweise einzelner Befehle genau verfolgen zu können. Die Reset-Schaltung wurde mit vier NAND-Schmitt-Triggern realisiert. Sie gewährleistet einen sicheren Anlauf des Rechners durch den Power-on-Reset beim Einschalten, sowie die Möglichkeit während des Betriebs über einen Taster ein Reset-Signal auszulösen. Der adressierbare Speicherbereich der CPUs ist unterteilt in RAM, EPROM und Ein-/Ausgabeblöcke. Dabei muß eine Besonderheit der Z80-CPU berücksichtigt werden, die beim Boot-Vorgang mit der Abarbeitung von Programmen an der untersten Adresse (0000h) beginnt. Die anderen CPUs erwarten das Boot-Programm am Ende des Speicherbereichs. Aus diesem Grund wurde in der Decodierlogik zusätzlich die Möglichkeit vorgesehen, durch setzen eines Jumpers die Speicherbänke von RAM und EPROM zu vertauschen. Die einheitliche CPU-Schnittstelle der alternativ verwendbaren CPU-Typen zur übrigen Hardware besteht aus dem 16-Bit Adreßbus, dem 8-Bit Datenbus sowie den Signalen zur Speicheransteuerung und CPU-Kontrolle.

Auf der Rechnerplatine sind drei CPU-Sockel vorgesehen. Da die Pin-Belegung der Prozessoren MC6802 und MC6502 nahezu identisch ist, lassen sie sich im selben Sockel betreiben. Lediglich

zwei Jumper für die Clock- und Wait-Leitung müssen prozessorspezifisch gesetzt werden. Zur Interaktion mit externen Schaltungen sind auf der Platine insgesamt 16 User-Input und 16 User-Output Leitungen vorgesehen, die auf Buchsen sowie auf eine Pfostenleiste herausgeführt sind. Die Ein-/Ausgabeeinheit besteht zum einen aus dem zweizeiligen LCD-Display, das direkt am E/A-Datenbus anliegt und über eine der E/A-Selectleitungen angesprochen wird. Als Eingabefeld dient ein Tastenfeld aus 5x6 Tasten, die über eine einfache Scanner-Schaltung abgefragt werden. Die dabei nicht benötigten Signalleitungen werden zur Ansteuerung einer RS232-Schnittstelle mit einem MAX232-IC benutzt. Im Bedienteil ist noch die Schaltung für den Wechsel zwischen Normal- und Single-Step-Betrieb sowie der Single-Step Taster enthalten.

Mechanischer Aufbau

Die Schaltung des Praktikumsrechners ist auf einer zweilagigen Zweifach-Europakarte aufgebaut (Format 200 x 160 mm^2), auf der Bedien- und Rechnerteil jeweils die Hälfte belegen. Die Anbindung des Rechnerteils an den Bedienteils geschieht über eine 26-polige Verbindung. Da es keine weiteren Verbindungen zwischen den beiden Boards gibt, ist eine Trennung der Platine an dieser Stelle ohne weiteres möglich. An den Trennstellen sind Sockel für Pfostenstecker vorgesehen, so daß die beiden Teile über ein Flachbandkabel verbunden werden können. Für einen Minimalbetrieb brauchen Tastatur und Display nicht bestückt zu werden, als Verbindung zur Außenwelt steht neben den Ports eine serielle Verbindung auf dem Bedienteil zur Verfügung. Auf der Rechnerplatine ist der vollständige Computer untergebracht. Es war aus Platzgründen nicht möglich, drei Sockel für die unterschiedlichen Prozessoren nebeneinander zu plazieren. Aus diesem Grund wurden die drei Prozessorsockel verschränkt angeordnet. Damit wird gleichzeitig eine versehentliche Bestükkung mit mehreren Prozessoren verhindert.

Kosten

In der Minimalausstattung besteht der Rechner aus Prozessor, RAM, ROM einer E/A Schnittstelle und einer seriellen Schnittstelle. Da dieser Rechner für fortgeschrittene Anwendungen konzipiert ist, wird er über die serielle Schnittstelle bedient. Die Kosten für den Minimalaufbau belaufen sich auf ca. 120 Mark.

Für das Praktikum "Aufbau und Einsatz von Mikrocomputern" wird dagegen der Vollausbau mit Tastatur und Display verlangt. Bei verschleißanfälligen Teilen wie Tasten und Buchsen wurden keine Kompromisse eingegangen, sondern qualitativ hochwertige Bauteile wie beispielsweise vergoldete Buchsen und Druckpunkttasten eingesetzt. Beim Display wurde ein Supertwisted LCD Display gewählt, das auch bei ungünstigen Blickwinkeln noch eine gute Ablesbarkeit bietet. Ein MUMPITZ in Praktikumsausführung kostet knapp 350 Mark. Bei der Praktikumsausführung bestimmen die Kosten für Tastatur, Display, Netzteil und Gehäuse den Preis. Sie machen allein etwa 2/3 des Preises aus. Die Kosten für einen voll ausgebauten MUMPITZ liegen mindestens um den Faktor 5 unter ähnlichen, industriell angebotenen Rechnern, wobei der MUMPITZ hinsichtlich Bedienbarkeit und Flexibilität deutliche Vorteile aufweist.

Anwendung des MUMPITZ im Hardware-Praktikum

Verwendung von GALs zur Realisierung von State Machines

Moderne programmierbare Logik-Bausteine enthalten neben der kombinatorischen Logik Makrozellen mit Flipflops, mit deren Hilfe auch aufwendige sequentielle Schaltungen in einem Chip realisiert werden können. State Machines mit diesen Bausteinen verhielten sich in der Praxis fehlerhaft, obwohl die Simulation fehlerfrei durchlaufen wurde. Grund für die beobachteten falschen Zustandsübergänge waren asynchrone Eingangssignale, die bei Verletzung der Setup-Zeiten der GAL-/LCA-Flipflops falsche Folgezustände erzeugten.

Der MUMPITZ-Rechner und das dazugehörige GAL-Programmiergerät können nun dazu verwendet werden, eine Schaltung aufzubauen, mit deren Hilfe das fehlerhafte Verhalten von State Machines bei asynchronen Eingangssignalen demonstriert werden kann. Dabei wird ein GAL so programmiert, daß es zwei 4-Bit-Binärzähler enthält, deren Flipflops von einem gemeinsamen (hochfrequenten) Takt gespeist werden und deren Zustandsübergänge von einem gemeinsamen (niederfrequenten) Eingangssignal abhängen. Durch geeignete Vorschaltung kann das Eingangssignal wahlweise synchron oder asynchron zum Takt in das GAL gegeben werden. Ein Vergleicher im GAL prüft, ob die Zähler den selben Wert beinhalten. Wenn nicht, so schaltet der Vergleicher das niederfrequente Eingangssignal ab und die Zähler zeigen den letzten, also falschen Zählerstand an. Die Zählerwerte werden durch je vier Leuchtdioden sichtbar gemacht. Man beobachtet nun, daß die Binärzähler bei synchronem Eingangssignal problemlos arbeiten, während die Zähler bei asynchronem Eingangssignal sehr bald falsche Zustandsübergänge erzeugen und durch die Vergleichslogik angehalten werden.

Verwendung von Flachbandkabeln zur Datenübertragung

In vielen studentischen Hardware-Arbeiten ist die Verwendung von Flachbandkabeln zur Datenübertragung zwischen zwei Teilschaltungen erforderlich. Obwohl meist bekannt ist, daß große Leitungslängen bei entsprechend hoher Übertragungsrate, fehlerhafte oder fehlende Leitungs-Abschlüsse sowie unzureichende Masse-Verbindungen Probleme erzeugen können, wird diese mögliche Fehlerquelle häufig übersehen. Im Labor hatten wir innerhalb kurzer Zeit drei aus diesen Gründen fehlerhafte Schaltungen.

Der MUMPITZ ist zur Demonstration dieses Phänomens ebenso geeignet. Er besitzt zwei Ports, wobei der eine den Anschluß eines Flachbandkabels erlaubt. Schließt man das eine Ende eines ca. 1m langen Flachbandkabels so an den MUMPITZ-Rechner an, daß genau 8 Datenbits, ein Strobe-Signal, ein Lese-Signal und eine Masse-Leitung zu einem Schreib/Lese-Latch am anderen Leitungsende führen, kann eine fehlerhafte Datenübertragung provoziert werden.

Benutzung der V.24 (RS-232)-Schnittstelle

Selbst ein erfahrener Hardware-Praktiker muß immer wieder feststellen, daß der Aufbau einer Datenübertragungsstrecke mit RS-232-Schnittstellen oft langwieriger verläuft als erwartet. Die Schwierigkeiten enstehen hauptsächlich durch die DCE-DTE-Verwechslung, durch die nicht funktionierende Schnittstellenbedienung der PC-Betriebssysteme und durch die nicht einheitliche und verwirrende Benutzung der Steuersignale der Schnittstelle. Mit der seriellen Schnittstelle des MUMPITZ können im Hardware-Praktikum anschauliche Versuche zu dieser Thematik durchgeführt werden.

Neben dem ursprünglichen Zweck des MUMPITZ-Rechners, nämlich der Lehre der hardware-nahen Maschinen- und Assembler-Programmierung, kann das Gerät also auch zur Demonstration von typischen Problemen aus der Hardware-Praxis eingesetzt werden.

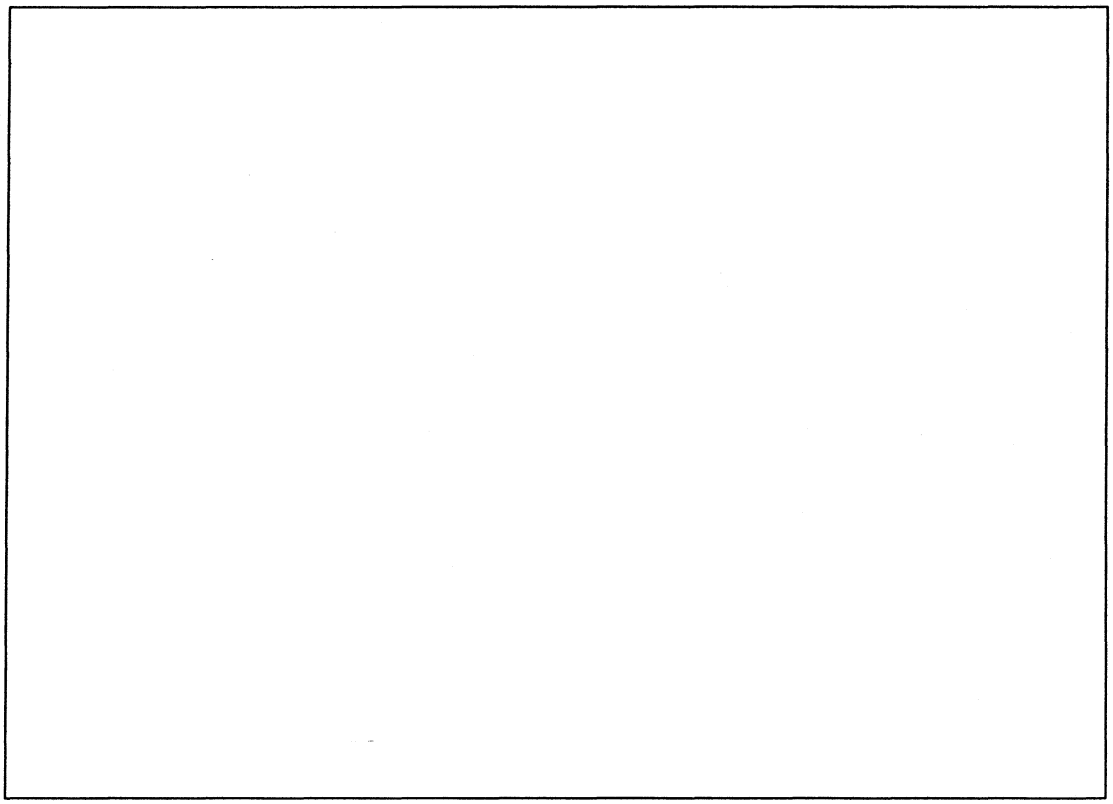

Bild 1 MUMPITZ-Prototyp