Der PC als Echtzeit-Regelungsrechner in der Entwicklungsphase neuer Antriebsregelungen

Dipl.-Ing. J. Faßnacht; Prof. Dr.-Ing. P. Mutschler
Institut für Stromrichtertechnik und Antriebsregelung
Technische Universität Darmstadt, Landgraf-Georg-Str. 4, 64283 Darmstadt

1 Anforderungen während der Entwicklungs- und der Produktphase

Als erster Schritt zur Umsetzung einer Idee für ein neues Antriebs-Regelkonzept dient meistens die digitale Simulation auf einem PC. Das zu simulierende System wird nach Bild 1 gegliedert in a) die Regelstrecke und b) in die zu entwickelnde Regelung und Steuerung.

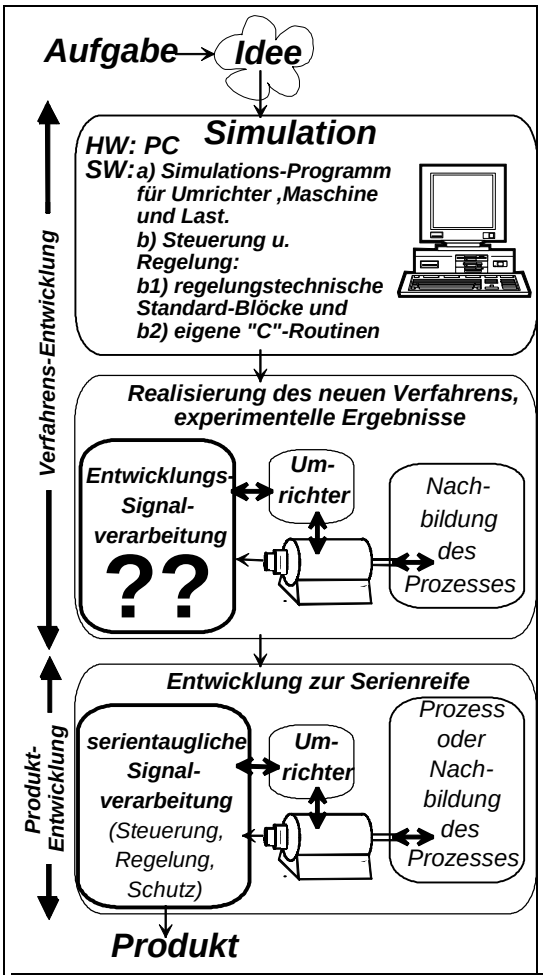

Bild 1: Schritte bei der Entwicklung

Die zu regelnde Strecke besteht typischerweise aus dem Umrichter, der Maschine und der angeschlossenen Last d.h. dem Prozeß, meist in vereinfachter Form. Wenn eine maschinennahe Regelung entwickelt wird, muß in der Simulation die schaltende Arbeitsweise des Umrichters, ggf. mit Berücksichtigung von parasitären Effekten wie Mindesteinschalt- oder Verzugszeiten nachgebildet werden. Bei der Maschinennachbildung greift man meistens auf ein Grundwellenmodell, ggf. mit Berücksichtigung der Sättigung, zurück.

Bei der Simulation der Steuer- und Regelalgorithmen kann man nur teilweise auf standardisierte regelungstechnische Blöcke zurückgreifen, da für neue Verfahren solche nicht existieren. Das Simulationsprogramm muß daher auf jeden Fall die Einbindung eigener Regelroutinen, sinnvollerweise in der Sprache „C“ erlauben. Am Markt werden zahlreiche Simulationsprogramme angeboten, aber nur wenige erfüllen alle Forderungen gleichzeitig. Aus eigener Erfahrung kann gesagt werden, daß eine detaillierte Nachbildung des Umrichters z.B. mit Matlab/Simulink zu unvertretbar langen Simulationszeiten führt. Aber auch bei Programmen, die sich gut zur Umrichternachbildung eignen, wie z.B. SIMPLORER, können Wünsche offen bleiben. Eigene Untersuchungen zeigen, daß (zum Zeitpunkt der Manuskripterstellung) die Einbindung eigener umfangreicher „C“-Routinen noch nicht befriedigend gelöst ist. In der vorliegenden Arbeit wurde das am Institut entwickelte Simulationsprogramm „PECSIM“ [Ansch] verwendet, das obige Anforderungen erfüllt.

Nach der Simulation findet als zweiter Schritt die Realisierung und experimentelle Untersuchung des Verfahrens an einem Antriebs-Prüfstand statt. Dabei stellt sich unter anderem die Frage, mit welcher Signalverarbeitung (HW u. SW) diese Untersuchung durchgeführt werden soll. Die Anforderungen an die Signalverarbeitung sind während der Verfahrens-Entwicklung andere als beim fertigen Serienprodukt. Einige der wichtigsten Unterschiede sind in der folgenden Tabelle gegenübergestellt.

Anforderungen an die Signalverarbeitung von Antrieben		
	während der Verfahrens-Entwicklung	beim Serienprodukt
	„Spielwiese“ für neue Konzepte erfordert möglichst hohe Rechenleistung	an Aufgabe angepaßte Rechenleistung ist wirtschaftlicher
	Programmierung in Hochsprache (z.B. „C“)	zur Not (z.B. um HW-Kosten zu sparen) auch besonders zeitkritische Abschnitte in Assembler programmieren
	Sehr großer Transienten-Speicher (viele MB) zur Aufzeichnung aller interessierenden Systemgrößen erleichtert die Entwicklung / Fehlersuche erheblich	Serien-Produkt muß ausgereift sein. Einbau von extra HW-Ressourcen zur Fehlersuche ist unwirtschaftlich.
	Die Kosten der während der Entwicklungsphase verwendeten Signalverarbeitung dürfen höher als beim fertigen Serienprodukt sein.	Kosten**minimale** Signalverarbeitung notwendig.
	Die für die Erprobung neuer Regelungs-Konzepte erforderliche Signalverarbeitungs-HW soll selbst möglichst wenig Entwicklungsaufwand erfordern.	Der Entwicklungsaufwand für die Signalverarbeitung darf höher sein, wenn damit die Stückkosten beim Serienprodukt geringer werden.

Wenn es sich um ein weitgehend neues Verfahren handelt, dann ist die Abschätzung der erforderlichen Rechenleistung mit großen Unsicherheiten behaftet. In diesen Fällen ist eine Signalverarbeitungs- Plattform als „Spielwiese“ für die Verfahrens-Entwicklung mit möglichst hoher Rechenleistung erwünscht. In Kap. 3 wird die Verwendung des PC mit Zusatzhardware für diese Aufgabe diskutiert.
Wenn die Verfahrens-Entwicklung zu guten experimentellen Ergebnisse geführt hat, dann kann die Produktentwicklung zur Serienreife erfolgen. Dabei ist die Auswahl des „richtigen“ Prozessors zu treffen, was gleichzeitig verbunden ist mit der Festlegung einer optimalen Aufgabenverteilung zwischen (Signal-)Prozessor und zusätzlicher Hardware, die typischerweise als ASICs ausgeführt wird. Dieser letzte, schwierige Schritt ist aber nicht Gegenstand des vorliegenden Papiers.

2 Beispiel für neues Regelverfahren: Direct Mean Torque Control (DMTC)

Zur hochdynamischen Regelung der Asynchronmaschine sind seit [Has68] zahlreiche Ausgestaltungen der Feldorientierten Regelung (FOC) entwickelt worden. Die FOC basiert darauf, die Drehfeldmaschine gedanklich durch eine Transformation auf eine Gleichstrommaschine (GM) abzubilden und auf dieses Abbild die bewährte Regelung für Gleichstrommaschinen anzuwenden. Ein anderer Ansatz [For73], [Dep85], [Tak86] betrachtet **direkt** die Beziehung für das **Drehmoment** der Drehfeldmaschine (-ohne Abbildung auf eine GM-) und trifft die Auswahl aus den 8 möglichen Schaltzuständen des U-Wechselrichters so, daß das Drehmoment in einem gewünschten Hystereseband gehalten wird. Das Prinzip der „Direkten Momenten Regelung“ (DTC) nach [Tak86] ist in Bild 2 angegeben. Der Anwender des Antriebes ist primär nur am oben besprochenen Drehmoment interessiert. Maschinenintern muß aber zusätzlich dafür gesorgt werden, daß der zur Drehmomentbildung nötige Betrag des Flusses im wesentlichen konstant gehalten wird, wozu ein zweiter Schaltregler verwendet wird. Die hierfür erforderlichen Ständerflußkomponenten werden durch Integration eines Ständerspannungsmodells gewonnen, was allerdings bei niedriger Drehzahl bekanntlich so nicht funktioniert und ein erweitertes Modell oder Beobachter erfordert. Charakteristische Eigenschaften des Verfahrens nach Bild 2 sind:

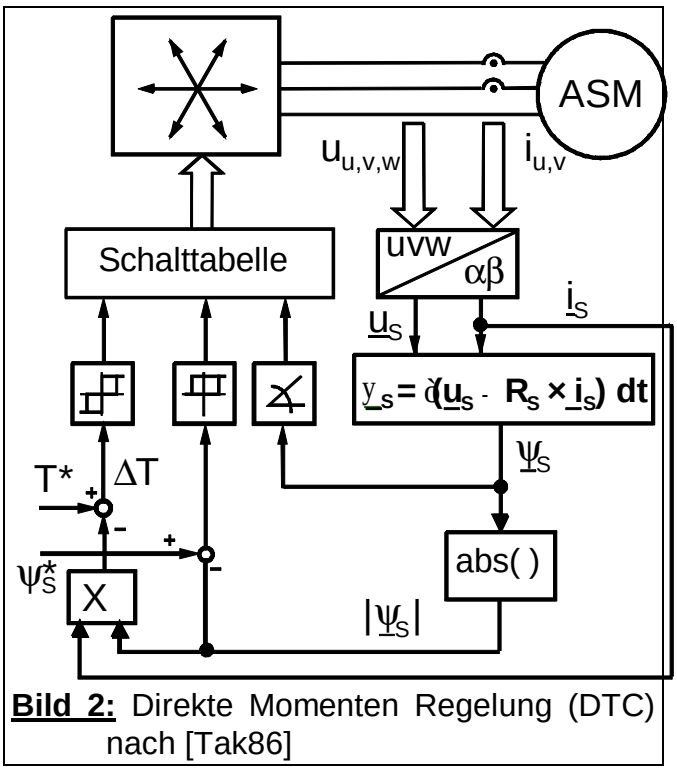

Bild 2: Direkte Momenten Regelung (DTC) nach [Tak86]

1) ***Variable Schaltfrequenz***. (Durch zusätzliche Schaltfrequenzregelung kann nur im **Mittel** konstante Schaltfrequenz erreicht werden).
2) zeitkontinuierliche (=***analoge***) Realisierung.

Beide Eigenschaften sind unerwünscht. Eine konstante Schaltfrequenz hat zahlreiche Vorteile und moderne Antriebsregelungen werden zeitdiskret (=digital) realisiert. Realisiert man das Verfahren aus Bild 2 zeitdiskret (=digital), d.h. mit einem Rechner, dann ergibt sich ein Verhalten nach Bild 3.

Beim Abtastzeitpunkt t_{n-2} erfaßt der Rechner die Meßwerte. In der folgenden Rechenzeit erkennt der Rechner, daß das zulässige Hystereseband bei t_{n-2} gerade unterschritten wurde. Er wählt daraufhin einen momentenaufbauenden Schaltzustand des WR aus, der aufgrund der erforderlichen Rechenzeit erst zum Schaltzeitpunkt t_{S1} eingeschaltet wird. Bei den Abtastzeitpunkten t_{n-1} und t_n ist die obere Grenze des Hysteresebandes noch nicht überschritten, so daß weiterhin der momentenaufbauende Schaltzustand bestehen bleibt. Erst die Meßwerte bei t_{n+1} ergeben eine Über-

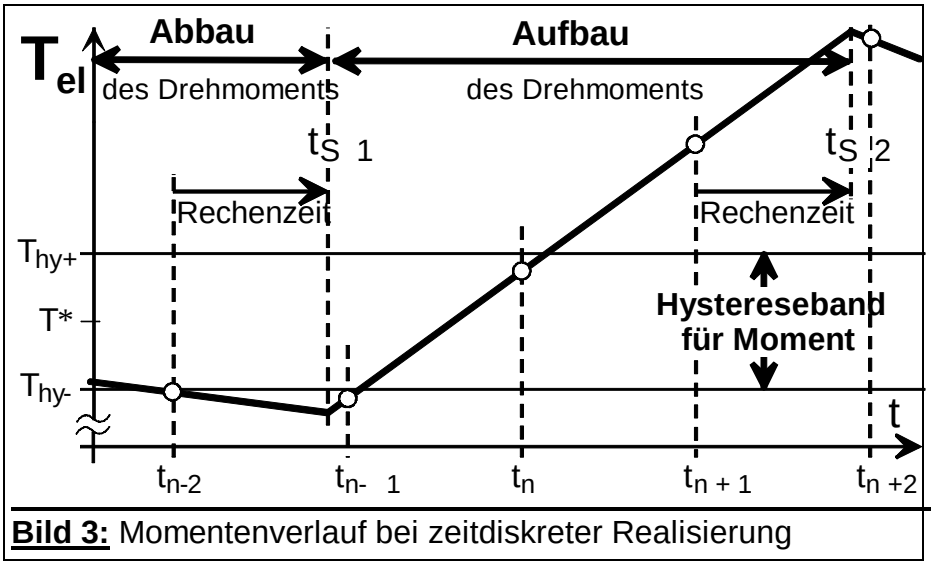

Bild 3: Momentenverlauf bei zeitdiskreter Realisierung

schreitung des Hysteresebandes. Nach der erforderlichen Rechenzeit wird bei t_{S2} ein momentenabbauender Schaltzustand hergestellt. Man sieht daraus, daß das Moment die Hysteresebreite um fast 200% überschreiten kann. Bei streuarmen, hochdynamischen Servomotoren kann die Zeit zum Durchlaufen des Hysteresebandes extrem kurz, z.B. 5 *ms* werden.

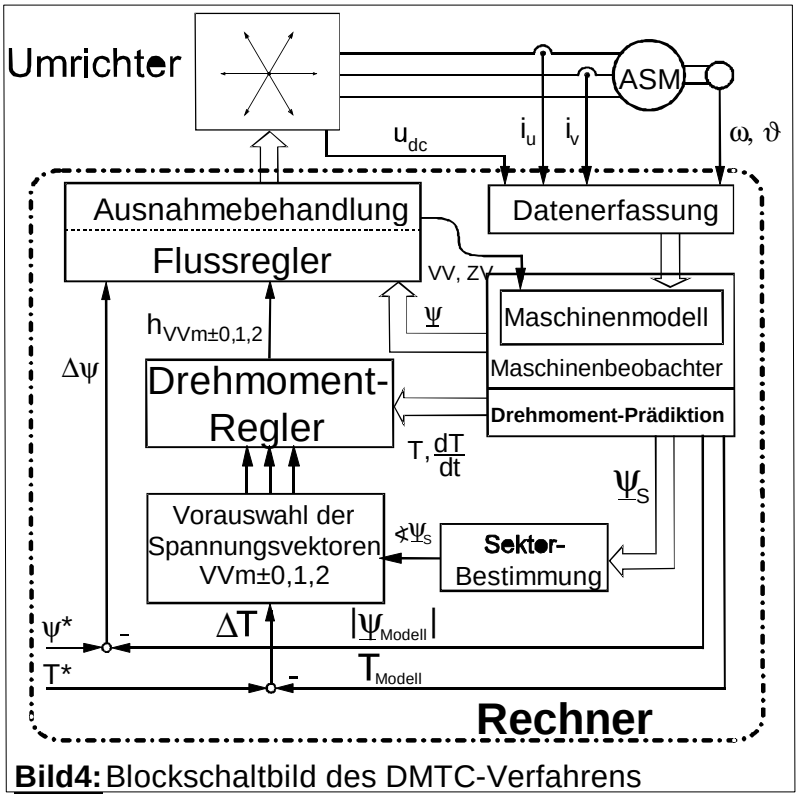

Bild4: Blockschaltbild des DMTC-Verfahrens

In solchen Fällen müßte die Abtastzeit des Rechners deutlich kürzer als diese 5 *ms* sein. Damit müßte eine enorme Rechnerleistung installiert werden, was völlig unpraktikabel ist. Um sowohl dieses Problem zu lösen, als auch eine konstante Schaltfrequenz zu erzeugen, wurde das Verfahren „Direct Mean Torque Control“ (DMTC) [Fla98], [Mu98/1], [Mu98/2] entwickelt.

DMTC ist ein **prädictives** Verfahren, das für ein *konstantes* Intervall (wg. konstanter Schaltfrequenz) die Einschaltzeitpunkte für **zwei** folgende Schaltzustände (ein momentenaufbauender und ein momentenabbauender) so vorausberechnet, daß nach Ablauf dieses Intervalls bereits Stationärbetrieb vorliegt. Somit ist DMTC ein Regelverfahren mit minimaler, endlicher Einstellzeit (dead beat). Die Einschaltdauer eines Schaltzustandes ist

durch das Verfahren nach unten nicht begrenzt, so können z.B. die oben erwähnten $5\,ms$ problemlos realisiert werden.

Bild 4 zeigt das Blockschaltbild des DMTC-Verfahrens. Es wird hier ein vollständiger Maschinenbeobachter verwendet, mit dem die Flusskomponenten, das Drehmoment und auch die, für die Prädiction erforderlichen, zeitlichen Ableitungen (Steigungen) dT/dt des Momentenverlaufes berechnet werden. Anhand der Drehmomentabweichung DT erfolgt die Vorauswahl von drei prinzipiell in Frage kommenden Spannungsvektoren. Der Drehmomentregler berechnet für die Spannungsvektoren der Vorauswahl eine potentielle Einschaltdauer (h_{vv}) so, daß sein Regelziel möglichst gut erfüllt wird, d.h. daß am Ende des Intervalls -falls möglich- Stationärbetrieb vorliegt (dead beat). Welcher der vorausgewählten Spannungsvektoren tatsächlich eingeschaltet wird, hängt davon ab, welcher dieser Vektoren den Fluß am besten bei seinem Sollwert hält. Mit Hilfe einer ausgefeilten Entscheidungsstruktur wird bei widersprechenden Wünschen des Drehmoment- und des Flußreglers ein optimaler Kompromiß gefunden. Der Block „Ausnahmebehandlung“ (s. Bild 4) ermöglicht u.a. die situationsangepasste Umkehr der Schaltreihenfolge. Wenn sich z.B. der Momentensollwert am Intervallbeginn deutlich verkleinert hat, dann ist es nicht sinnvoll, die Regelabweichung zuerst durch einen momentenaufbauenden Schaltzustand noch zusätzlich zu vergrößern, um sie erst danach mit dem momentenabbauenden Schaltzustand wieder kleiner zu machen. Vielmehr ist es in diesem Fall sinnvoll, die Schaltreihenfolge umzukehren, d.h. sofort den momentenabbauenden Schaltzustand zu verwenden.

Vergleicht man die Blockdiagramme Bild 2 und Bild 4, so kann man erahnen, daß die DMTC deutlich mehr Rechenzeit benötigt. Dies liegt zum einen an dem vollständigen Beobachter, aber hauptsächlich daran, daß die DMTC nicht auf einer starren, off-line vorgefertigten Schalttabelle beruht, sondern daß eine optimale, situationsangepasste Auswahl jeder einzelnen Schalthandlung erfolgt. Darin liegt aber auch ein Vorteil gegenüber der Feldorientierten Regelung (FOC). Eine FOC mit konstanter Schaltfrequenz verwendet einen Modulator, der lediglich im Mittel die gewünschte Maschinen-Spannung approximiert, der Momentanwertverlauf des (den Anwender haupsächlich interessierenden) Drehmomentes wird dabei nicht optimiert.

3 Hard- u. Software der „Entwicklungs- Signalverarbeitung“

Das oben beschriebene DMTC-Verfahren wurde zuerst auf einem (nunmehr über 10 Jahre alten) TMS320C30-Signalprozessor mit 40 MHz Taktfrequenz mit einer Regelungs-Zykluszeit von $150\,ms$ implementiert. Da in einem Zyklus 2 Vektoren geschaltet werden, ergibt sich eine Schaltfrequenz von 6,6kHz. Einschließlich Pulsgeber-Interpolation, Drehzahlregelung, Transientenrecorder usw. ist der in „C“ programmierte TMS320C30 damit zu $\approx$80% ausgelastet. Zur Erhöhung der Schalfrequenz, zur Abarbeitung komplexer überlagerter Regelungsalgorithmen, wie zum Beispiel zur Dämpfung von mechanischen Resonanzen im Antriebsstrang, zur on-line Parameteridentifikation usw. wird mehr Rechenleistung benötigt. Daher wurde untersucht, in wie weit sich ein PC mit Zusatzhardware zur „Entwicklungs- Signalverarbeitung“ eignet.

3.1 Betrachtete Systeme

Es wurden zwei PCs untersucht, einer mit dem Prozessor 486DX33, der andere mit einem PentiumII bei 233MHz. Beide Prozessoren können entweder im Real- oder im Protected-Mode betrieben werden. Wichtige Unterschiede der Modi zeigt nebenstehende Tabelle. Der PentiumII-Prozessor ist für den Protected Mode optimiert. Im Protected Mode sollte aber gewährleistet sein, daß die Anwendung eine ausreichend hohe Privilegierungsstufe hat, um auch ohne meist relativ langsame Treiberprogramme direkt auf Ein/Ausgabe-Ports zugreifen zu können.

	Real Mode	Protected Mode
Operationen	16 bit	auch 32 bit
ansprechbarer Speicher	640 kB /1MB	4GB
Multitaskingfähigkeit	nein	ja

Als Betriebssysteme wurden untersucht:
1) DOS Vers.5.0
2) DOS Vers.5.0 mit DOS-Extender DOS/4GW
3) RTTarget-32 [RTT]
4) QNX-Neutrino [QNX]

Die Systeme 3) bis 4) verwenden den Protected Mode, während 1) im Real Mode und 2) im Real oder Protected Mode arbeiten.

3.2 Interruptlatenzzeit

Die Interruptlatenzzeiten sind wesentlich für das Echtzeitverhalten. Sie wurden gemessen, indem eine Interruptanforderung über den ISA-Bus an den Interruptkontroller gegeben wurde. Die hierbei aufgerufene Interruptserviceroutine gab als ersten Befehl einen Schreibbefehl auf den ISA-Bus aus, welcher zusammen mit dem Triggerimpuls oszillografiert wurde. Hierbei waren alle anderen Interruptleitungen durch Programmierung des Interruptkontrollers gesperrt. Von der gemessenen Interruptlatenzzeit müßte somit noch ≈0,5 bis 1 ms für das Schreiben auf den ISA-Bus abgezogen werden.

verwendetes Betriebssystem	gemessene maximale Interruptlatenzzeit
DOS Version 5.0	≈ 11 ms
DOS 5.0 mit DOS/4GW (DOS-Extender)	≈ 32 ms *(mit teilweisem Übergehen einer Interruptanforderung !!!)*
RTTarget-32	≈ 6 ms

Interruptlatenzzeiten gemessen an einem Intel **486DX33** auf UMC491 VL-Bus Board

Die erste Meßreihe zeigt, daß beim 486DX33 eine Beschränkung auf den Real Mode unter DOS wie in [Abr97] eine deutliche Veringerung der Interruptlatenzzeit und der Zeiten für I/O- und Speicheroperationen bewirkt. Weiterhin mußte festgestellt werden, daß bei Verwendung des

verwendetes Betriebssystem	gemessene maximale Interruptlatenzzeit
DOS Version 5.0	≈ 22,2 ms
DOS 5.0 mit DOS/4GW	≈ 6,6 ms
RTTarget-32	≈ 3,5 ms

Interruptlatenzzeiten gemessen an **PentiumII** mit **233 MHz** auf Gigabyte GA-686LX3-Board

DOS-Extenders beim 486DX33 Interruptanforderungen teilweise nicht ausgeführt wurden!
Die zweite Meßreihe belegt, daß der PentiumII eher für Anwendungen im Protected Mode als im Real Mode geeignet ist, trotz der aufwendigeren Adressierung und Verwaltung der Speicher- und I/O-Ressourcen in diesem Modus [Mess].
In beiden Meßreihen wird mit RTTarget-32 die geringste Interrupt-Latenzzeit erreicht. Zusätzlich wurde auch das Betriebssystem QNX-Neutrino [QNX] untersucht. Es erreicht ähnlich kurze Interruptlatenzzeiten wie RTTarget-32.

3.3 Auswahl eines Betriebssystems

Das oben vorgestellte DMTC-Verfahren mit einer Regelungs-Zykluszeit von 150 ms stellt harte Echtzeitanforderungen. Um diese zu erfüllen muß jeglicher Overhead bei der Programmausführung vermieden werden. Es wird keine aufwendige Benutzerführung benötigt, da während der Laufzeit der Regelung nur einfache Eingaben, wie die der Solldrehzahl, zu tätigen sind. Auf Multitaskingfähigkeit kann verzichtet werden, da nur der Regelungsinterrupt ausgeführt werden muß. Multitasking wirkt sich aufgrund der nötigen hochprioren Timer- und sonstigen Scheduling-Interrupts negativ auf die Latenzzeit der Regelungsinterrupts aus.
Bei der Auswahl des Betriebssystemes wurde auch QNX-Neutrino [QNX] untersucht. Dabei traten während der Tests im Herbst 98 so viele Probleme auf, daß diese Linie jedoch nicht weiterverfolgt wurde.
DOS (ohne Extender) arbeitet nur im Real Mode und ist damit zu langsam.
Sowohl DOS mit Extender als auch RTTarget-32 sind nicht multitaskingfähig und benutzen das Flat-Memory-Modell, wobei alle Segmentregister auf dieselbe Adresse zeigen. Beim Programmieren von Echtzeitanwendungen für den DOS-Extender muß mit großer Sorgfalt vorgegangen werden, es dürfen z.B. keine DOS-Interrupts aufgerufen werden, die den DOS-Extender dazu veranlassen kurzzeitig in den Real Mode zu schalten [WatC]. Dies würde Verzögerungen im Millisekundenbereich bewirken.
Aufgrund der größeren Leistungsfähigkeit bei I/O-Operationen und der geringeren Interruptlatenzzeit wurde RTTarget-32 der Vorzug gegenüber DOS mit Extender gegeben. Allerdings muß bei RTTarget nach Beendigung der Echtzeitanwendung (d.h. Antrieb wird ausgeschaltet) neu gebootet oder das Programm erneut über die serielle Schnittstelle heruntergeladen werden. Die Programmentwicklung wird komfortabler, wenn ein zweiter, über die serielle Schnittstelle mit dem Regelungsrechner verbundener, PC als Programmiercomputer benutzt wird. Weiterhin bietet RTTarget keine Oberfläche zur Ausführung anderer Programme, zum Beispiel zur Visualisierung der Systemgrößen. Die interessierenden Systemgrößen werden während dem Echtzeitbetrieb in einen großen Ringspeicher geschrieben. Mit Beendigung des Echtzeitbetriebes können diese Daten auf Festplatte gerettet werden. Anschließend können sie mit den üblichen Visualisierungsprogrammen, z.B. Origin off-line unter Microsoft Windows dargestellt werden.

3.4 Reine Rechenleistung und Jitter

Zum Vergleich der Leistungsfähigkeit zwischen einem TMS320C30 mit 40 MHz und dem PC wurde der Kern der DTC-Regelungsroutine ohne Ein/Ausgabe-Routinen, ohne überlagerte Drehzahlregelung usw. zur reinen Messung der Rechenleistung implementiert. Das C-Programm war für TMS und PC identisch. Die Tabelle zeigt, daß der PC nur mit PentiumII-Prozessor schneller als der TMS320C30 wird. Weiter ist zu sehen, daß der PentiumII einen deutlich größeren ***relativen*** Jitter in der Ausführungsdauer als der Signalprozessor hat. Dies liegt an der Superskalararchitektur des PentiumII, der mehrere tiefe Befehlspipelines hat [Mess]. Um eine hohe Verarbeitungsleistung zu erreichen und die zwei parallelen Fließkomma- und Integerarithmetikeinheiten optimal auszulasten, versucht der Prozessor Sprünge anhand statistischer Auswertungen vorherzusagen und Befehle spekulativ auszuführen. Wenn dies fehlschlägt, so muß die Befehlspipeline geleert und mit den nun gültigen Anweisungen neu gefüllt werden, was sehr zeitaufwendig ist. Die Speicherorganisation mit verschiedenen Cache-Stufen ist ebenfalls sehr schnell, wenn der gesuchte Wert sofort im nächsten Cachespeicher gefunden wird, und eher langsam, falls seltene Werte erst nach einigen Fehlgriffen im Speicher gefunden werden. Die ganze Struktur ist eher auf eine ***hohe mittlere*** Verarbeitungsgeschwindigkeit optimiert als auf die Einhaltung möglichst kurzer maximaler Ausführungsdauern einzelner Programmteile, was allerdings für die Prozeßautomatisierung wichtiger wäre. Auch der Aufbau der Motherboards und des PCI-Busses ist auf eine hohe mittlerer Durchsatzrate ausgelegt. Bei sehr zeitkritischen Regelungsanwendungen muß dies bei der Programmierung beachtet oder eine ausreichende Zeitreserve eingeplant werden.

Rechnertyp	Ausführungs-dauer	Relativer (Absoluter) Jitter
TMS320C30; 40 MHz	max. 49 *ms*	5% (2,5 *ms*)
486DX33 mit DOS-Extender	max. 157 *ms*	5% (7,8 *ms*)
PentiumII; 233MHz mit DOS-Extender	max. 9 *ms*	20% (1,8 *ms*)
PentiumII, 233MHz mit RTTarget-32	max. 8 *ms*	20% (1,6 *ms*)

3.5 Zusatzhardware

Beim PentiumII tritt ein beachtlicher Jitter in der relativen Programm-Ausführungszeit und ebenso ein Jitter in der Interruptlatenzzeit auf. Bei praktisch allen Antriebsregelungen müssen aber die Schaltbefehle an das Leistungsteil mit sehr enger zeitlicher Toleranz ausgegeben werden. In gleicher Weise müssen Meßwerte, z.B. die Spursignale eines (Sin/Cos) Lagegebers genau zu vorgegebenen Zeitpunkten abgetastet werden. Aufgrund der oben erwähnten Jitter werden die zeitgenauen Ein/Ausgaben mittels einer Timerkarte (=Zusatzhardware) gesteuert.

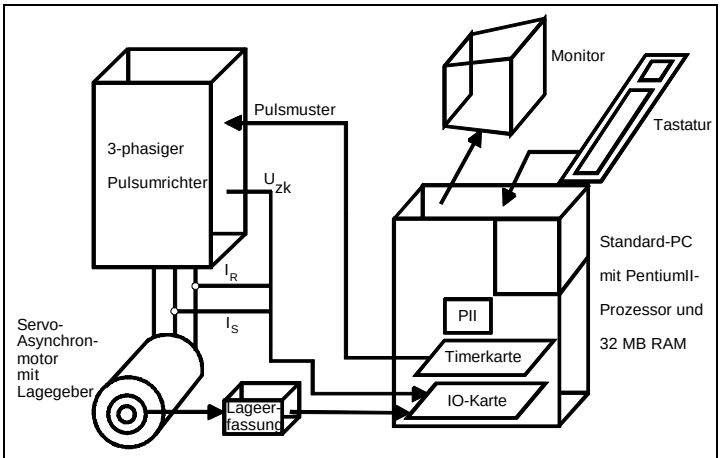

Bild 5: Übersicht über die Hardware des Versuchstandes

Diese Karte gibt auch die Startinterrupts für die Regelungsroutine aus. Auf einem CPLD der Timerkarte sind mehrere Zähler und Ausgaberegister implementiert. Ein PC verfügt typischerweise über 16 Hardwareinterrupts, von denen meist mehrere frei verfügbar sind. Um der damit getriggerten DMTC-Regelungsroutine die höchste Priorität zu geben, werden während der Laufzeit der Regelung alle anderen Interruptquellen durch Programmierung des Interruptkontrollers gesperrt. Dies ist nur möglich, da RTTarget-32 keine eigenen Timer- oder sonstigen Interrupts ausführen muß. Die Timerkarte erledigt auch die nötigen Schutz- und Überwachungsfunktionen im Zusammenspiel mit der Regelungssoftware. Das Einlesen der Statorströme und der Zwischenkreisspannung erfolgt über eine weitere ISA-Bus-Einsteckkarte, die auch noch die Werte der Lageerfassung zwischenspeichert.

3.6 DMTC mit PC am Prüfstand

Das vollständige DMTC-Verfahren mit Drehzahlregelung wurde sowohl im TMS320C30 (40MHz) als auch im PC mit PentiumII (233MHz) unter RTTarget implementiert und beide am Antriebsprüfstand verglichen. Dabei wurde eine Ausführungsdauer von $\approx 120\,ms$ auf dem TMS320C30 und $\approx 30\,ms$ auf dem PC mit PentiumII ermittelt. Damit steht die angestrebte „Spielwiese“ für weitere Verfahrens-Entwicklungen zur Verfügung, zumal zu erwarten ist, daß die jetzt üblichen PentiumIII-Prozessoren mit 450MHz eine weitere Steigerung der Rechenleistung bringen.

3.7 Ergebnis

Der PC mit PentiumII (233MHz) und Zusatzhardware verfügt über eine vier- bis fünffach höhere Rechenleistung als der vorher vorhandene Signalprozessoraufbau. Ein anderer Vorteil ist der große nutzbare Speicher für Transientenrekorderfunktionen. Beim DMTC-Verfahren sind alle wichtigen Größen digital im PC verfügbar. Ein Transientenrekorder mit sehr vielen Känalen (auch mehr als 30) und je nach vorhandenem Hauptspeicher mit einer enormen Speichertiefe kann sehr billig und leicht realisiert werden. Der große verfügbare Speicher eignet sich aber auch gut zur Aufnahme von Korrekturtabellen (z.B. für Geberfehler) und nichtlinearen Kennlinien für die Regelung.
Die in „C“ geschriebenen Regelungsroutinen können (bis auf I/O-Operationen) zunächst in einem Simulationsprogramm, z.B. PECSIM [Ansch] getestet werden und dann -durch I/O-Routinen ergänzt- in Echtzeit am Antriebsprüfstand ausgeführt werden.

Als Betriebssystem kommt sowohl DOS mit DOS-Extender als auch RTTarget in Betracht. RTTarget ist bei I/O-Operationen schneller, hat die kleinere Interrupt-Latenzzeit, kostet aber wesentlich mehr.
Nachteilig beim PC gegenüber dem Signalprozessor ist die größere Interruptlatenzzeit und der größere relative Jitter in der Programmausführungszeit. Diese Nachteile können aber durch eine Timerkarte als Zusatzhardware abgefangen werden.

Schrifttum:

[Abr97] Abraham, L.; Frank, W.: Uncomplicated inverter drive for research development and teaching using standard personal computer. 7th European Conference on Power Electronics and Applications (EPE '97) Trondheim, 1997, Vol. 4, p. 4.952 - 4.957

[Ansch] Anschütz, W. : PECSIM-Benutzerhandbuch Version F02 und Zusatz zur C-Schnittstelle

[Dep85] Depenbrock, M.:Direkte Selbstregelung für hochdynamische Drehfeldantriebe mit Stromrichterspeisung. etzArchiv, Bd. 7,1985, Heft 7. S. 211-218

[Fla98] Flach, E: Improved Algorithm for Direct Mean Torque Control of an Induction Motor. PCIM'98, Intelligent Motion Nürnberg, 1998 Seite 261 - 267

[For73] Forsell,H.:Stromrichterantrieb. Offenlegungsschrift DE-A-2 318 602 Anmeldung vom 13.4.1973

[Has68] Hasse, K.: Zum dynamischen Verhalten der Asynchronmaschine beim Betrieb mit variabler Ständerfrequenz u. Ständerspannung. ETZ-A, Bd.89, 1968, S. 77-81

[Mess] Messmer, H: PC-Hardware, 5. Auflage, Addison-Wesley, 1997

[Mu98/1] Mutschler, P.; Flach, E. : Digital Implementation of Predictive Direct Torque Control Algorithms for Induction Motors, IEEE-IAS Industry Application Society Annual Meeting 1998, St. Louis, Vol.1, Seite 444 - 451

[Mu98/2] Mutschler, P.; Flach, E.: Predictive Direct Torque Control Algorithm for Induction Motors and its Digital Implementation
Internat. Conf. on Power Electronics ICPE Seoul, Korea,1998, Seite 1- 6

[QNX] QNX, Neutrino Microkernel, System Architecture; Oktober 1996

[RTT] RTTarget-32, 'Version 2.1; User's manual

[Tak86] Takahashi, I.; Noguchi, T.: A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor. IEEE Transactions on Industry Applications 1986, Vol. IA-22, Seite 820 - 827

[WatC] Watcom C/C++, Version 11.0: Programmer's Guide, 3rd Edition