Offene Antriebsreglerplattform

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von
Dipl.-Ing. Christian Kramer
aus Frankfurt a.M.

Hauptberichter:
Prof. Dr.-Ing. Dr. h.c. mult. Dr.-Ing. E.h. Günter Pritschow (i. R.)
Mitberichter:
Prof. Dr.-Ing. habil. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. Reimund Neugebauer

Tag der mündlichen Prüfung: 5. April 2011

Institut für Steuerungstechnik der Werkzeugmaschinen
und Fertigungseinrichtungen
der Universität Stuttgart

2011

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen
der Universität Stuttgart und dem Fraunhofer-Institut für
Produktionstechnik und Automatisierung IPA

Herausgeber: Prof. Dr.-Ing. A. Verl

Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen

Fraunhofer IPA

Christian Kramer

Offene Antriebsreglerplattform

Nr. 178

JOST-JETTER VERLAG
Fachverlag · 71296 Heimsheim

D 93

ISBN 978-3-939890-72-0
Jost Jetter Verlag, Heimsheim

Druck: printsystem GmbH, Heimsheim

Geleitwort des Herausgebers

In der Reihe „ISW Forschung und Praxis“ wird fortlaufend über Forschungsergebnisse des Instituts für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW) berichtet, das sich in vielfältiger Form mit der mechatronischen, steuerungstechnischen und antriebstechnischen Weiterentwicklung von Werkzeugmaschinen und Fertigungseinrichtungen beschäftigt. Die Arbeiten dieses Instituts konzentrieren sich im Besonderen auf die Bereiche Numerische Steuerungstechnik, Planungs- und Leitsysteme, Softwaretechnik, Maschinen- und Industrierobotertechnik sowie Mess-, Regel- und Antriebssysteme, also auf die aktuellsten Bereiche der Fertigungstechnik. Dabei stehen Grundlagenforschung und anwenderorientierte Entwicklung in einem stetigen Austausch, wodurch ein ständiger Technologietransfer zur Praxis sichergestellt wird.

Diese Buchreihe erscheint in zwangloser Folge und stützt sich auf Berichte über abgeschlossene Forschungsarbeiten und Dissertationen. Sie soll dem Ingenieur bei der Weiterbildung dienen und ihm Hilfestellungen zur Lösung spezifischer Probleme geben. Für den Studierenden bietet sie eine Möglichkeit zur Wissensvertiefung. Sie bleibt damit unter erweitertem Namen und in der inzwischen dritten Generation unverändert in der bewährten Konzeption, die ihr der Gründer des ISW, der leider allzu früh verstorbene Prof. Dr.-Ing. G. Stute, im Jahre 1972 gegeben hat und die Prof. Dr.-Ing. h.c. mult. Dr.-Ing. E.h. G. Pritschow von 1984 bis 2005 fortgesetzt hat.

Seit 2007 ist das ISW durch die gemeinsame Institutsleitung mit dem Fraunhofer IPA verbunden. Folgerichtig erscheinen nun auch die Berichte des IPA im Bereich Automatisierungstechnik in dieser Reihe.

Der vorliegende Band beschäftigt sich mit dem Entwurf einer offenen und modularen Antriebsreglerplattform, die den aktuellen technischen Anforderungen nachkommt und eine leistungsfähige Basis für neue Technologien, z. B. im Bereich der Sensorik oder Regelungstechnik bietet. Im ersten Teil der Arbeit werden zunächst die derzeit bestehende Antriebstechnik und die zur Verfügung stehenden Technologien beleuchtet. Anschließend werden die Anforderungen der Antriebstechnik an den Antriebsregler betrachtet. Anhand der daraus gewonnenen Erkenntnisse wird eine Antriebsreglerplattform, sowie die notwendigen Entwicklungswerkzeuge entworfen. Abschließend wird die Realisierung des im Rahmen dieser Arbeit entstandenen Konzepts vorgestellt.

Der Herausgeber dankt der Druckerei für die drucktechnische Betreuung und dem Jost Jetter Verlag für die Aufnahme der Reihe in sein Lieferprogramm.

A. Verl

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Dr. h.c. mult. Dr.-Ing. E.h. Günter Pritschow für die Betreuung und die Unterstützung meiner Arbeit während seiner Zeit als Direktor des ISWs und die Zeit darüber hinaus, sowie für die Erstellung des Hauptberichts. Herrn Univ.-Prof. Dr.-Ing. habil. Prof. E. h. Dr.-Ing. E. h. Dr. h. c. Neugebauer danke ich recht herzlich für die Übernahme des Mitberichtes.

Des Weiteren danke ich meinen ehemaligen Kollegen am ISW, die mich fachlich mit Rat und Tat bei meiner Forschungstätigkeit unterstützt haben, insbesondere Herrn Antoni Simon für die Durchsicht meiner Arbeit und die Fortführung der offenen Antriebsreglerplattform am ISW.

Für die konstruktive Kritik, die wertvollen Anregungen sowie die Durchsicht meiner Arbeit möchte ich Herrn Dr. Peter Pruschek, der mit seinem Engagement insbesondere in der Abschlussphase meiner Arbeit viel zum Gelingen beigetragen hat, meinen besonderen Dank aussprechen.

Ich danke meinen Eltern, die mich immer unterstützt haben und mir damit Studium und Promotion ermöglichten.

Meiner Frau Katharina möchte ich einen besonders herzlichen Dank aussprechen für die viele Zeit, Geduld und Mühe, die sie aufbrachte, um mir den notwendigen Freiraum für diese Arbeit zu schaffen, sowie für die Korrekturen. Meiner kleinen Tochter Janne möchte ich dafür danken, mir ausreichend Schlaf und Ruhe in dieser Zeit gegönnt zu haben, sowie für jedes Lächeln, das die Zeit am Schreibtisch auflockerte.

Christian Kramer

Inhaltsverzeichnis

Formelzeichen

B	Biaswert in der Floatingpoint-Darstellung
e	Exponent in der Floatingpoint-Darstellung
E	modifizierter Exponent in der Floatingpoint-Darstellung
k	Anzahl der Bits der Mantisse m in der Floatingpoint-Darstellung
l	Anzahl der Bits des Exponenten E in der Floatingpoint-Darstellung
m	Mantisse in der Floatingpoint-Darstellung
r	Anzahl der Vorkommabits bei Zahlen in Q-Notation
s	Anzahl der Nachkommabits bei Zahlen in Q-Notation
t_n	Ausführungszeit des Funktionsblocks n
T_n	Maximale Ausführungszeit des Funktionsblocks n

Abkürzungen

ANSI	American National Standards Institute
AO	Application Object
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
CNC	Computerized Numerical Control
CPLD	Complex Programmable Logic Device
DFG	Deutsche Forschungsgemeinschaft
DFT	Diskrete Fourier Transformation
DMA	Direct Memory Access
DR	Drehzahlregler
DSP	Digitaler Signalprozessor
EDA	Electronic Design Automation
FB	Funktionsblock
FPGA	Field Programmable Gate Array
FPU	Floatingpoint Unit
HMI	Human Machine Interface
ID	Identifikationsnummer
IGBT	Insulated Gate Bipolar Transistor
JTAG	Joint Test Action Group
LR	Lageregler
LVDS	Low Voltage Differential Signaling
NC	Numerical Control
NCK-OA	Numerical Control Kernel Open Architecture
MC	Motion Control
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
OCEAN	Open Controller Enabled by an Advanced real-time Network
OCP	Open Controller Platform
OEM	Original Equipment Manufacturer
OSACA	Open System Architecture for Controls within Automation Systems
OSADL	Open Source Automation Development Lab
PC	Personal Computer
PCI	Peripheral Component Interconnect
PWM	Pulsweitenmodulation
SOFIA	modulares Softwaresystem für intelligente Antriebe
SPS	Speicherprogrammierbare Steuerung
SR	Stromregler
UMOAC	University of Michigan Open Architecture Controller
VHDL	Very High Speed Integrated Circuit Hardware Description Language
µC	Mikrocontroller

Abstract

The increasing demands of dynamics and accuracy in machine tools and the requirement of many additional features, like process or machine monitoring, effect not only the construction of tool machines but also the drive control. In the drive control of the present days, specialized digital signal processors and microcontrollers are used. Because they are made for standard drive control with high quantities and low prices, there is a lack of flexibility and openness towards new sensors or control algorithms.

To include new technologies in an existing drive control severe changes in both software and hardware are required. This makes it difficult and expensive to adjust the drive control to the special needs of manufacturing processes and machines differing from the standard.

In this thesis a new approach for drive control is shown which fulfills the requirements of present-day machines and future developments in that it is flexible and open so it can be easily adjusted to special processes and machines.

This flexibility is achieved by a hardware based on programable logic. A library provides modules, containing all the functions for the drive control, process control, sensors analysis, machine monitoring, etc. The drive controller can be configured out of this library so that it provides exactly the functions needed by the process or machine. Besides the programable logic, the drive control provides slots where hardware interfaces towards sensors, actors, or fieldbusses can be added.

This thesis introduces the architecture of an open drive control, covering the modularization of the drive functions, the infrastructure needed for execution of the modules, for communication in between the modules, and for monitoring them. The openness and flexibility of the system requires special development tools also shown here.

As verification of this concept, an implementation of the open drive control architecture is shown controlling a linear direct drive, which demonstrates a basic library of drive function and a drive control configured with modules from this library.

1 Einleitung

Die ständig wachsenden Anforderungen hinsichtlich Dynamik und Genauigkeit, die an automatisierte Maschinen gestellt werden, sowie der Trend zu immer umfangreicher werdenden Zusatzfunktionen, wie z. B. die Prozess- und Maschinenüberwachung, betreffen nicht nur die Konstruktion und die Steuerung der Maschine, sondern auch die Antriebsregelung, die den direkten Zugriff auf die Sensorik und Aktorik der Maschine herstellt. Diese Anforderungen betreffen gleichermaßen Werkzeugmaschinen, Druckmaschinen, Verpackungsmaschinen sowie Roboter und Handlingeinrichtungen.

In heutigen Antriebsreglern werden speziell für diese Aufgabe gestaltete digitale Signalprozessoren (DSPs) und Mikrocontroller eingesetzt, die, wie auch die Mikroprozessoren, in den letzten Jahren in immer weiter steigenden Takten arbeiten. Ihr Funktionsumfang beschränkt sich jedoch auf die Anforderungen der Standard-Antriebsregelungsanwendung, da ein niedriger Preis und damit hohe Stückzahlen im Vordergrund stehen. Die Flexibilität hinsichtlich Schnittstellen und Regelalgorithmen oder mehrachsiger Regelung ist dabei nicht gegeben.

Neue Entwicklungen oder stark voneinander abweichende, vom Einsatzgebiet abhängige Anforderungen erfordern meist einen harten Eingriff in die bestehende Antriebsreglerhard- und -software und sind damit sehr zeit- und kostenintensiv. Jedoch sind aufgrund der wachsenden Anforderungen fortlaufend Änderungen, Anpassungen und Erweiterungen des Antriebsreglers notwendig.

Aufgrund hoher Reglertaktraten, Abtastraten und bei den Prozessen auftretenden Frequenzen sind der Rechenkapazitätsbedarf sowie die Rechen- bzw. Verarbeitungsgeschwindigkeiten der antriebsnahen Funktionalitäten sehr hoch. Dadurch können diese Funktionalitäten oft nur als eigenständige Hardware zusätzlich zum DSP realisiert werden.

Die höchste Flexibilität, die aus Herstellersicht ein Antriebsregler derzeit aufweisen muss, besteht bezüglich der Feldbusanbindung. Da durch die Kunden von Werkzeugmaschinen häufig vorgegeben wird, welche Steuerungstechnik und damit welche Feldbusse eingesetzt werden müssen, vergrößert sich mit der Auswahl von Feldbusschnittstellen der potenzielle Markt für das Antriebsregelgerät.

Diese Flexibilität wird heute schon in vereinzelten Anwendungen dadurch erreicht, dass der Feldbustreiber auf programmierbaren Logikbausteinen, z. B. *Field Programmable Gate Arrays* (FPGAs) oder *Complex Programmable Logic Devices* (CPLDs), neben dem eigentlichen, mikrocontrollerbasierten Antriebsregler untergebracht werden, sodass nur noch die physikalische Ankopplung an das Feldbusmedium als austauschbare Hardware realisiert werden muss. Je nach Feldbusprotokoll muss nur der FPGA umprogrammiert werden. Aufgrund geringerer Kosten wird jedoch weitestgehend auf diese Flexibilität verzichtet, und spezielle *Application Specific Integrated Circuits* (ASICs) eingesetzt.

FPGAs werden für die Antriebstechnik durch die ständig wachsende Logikkapazität und die fallenden Preise dieser Technik zunehmend interessanter. Die derzeit verfügbaren und auch wirtschaftlich gewinnbringend einsetzbaren FPGAs erlauben es, neben der Feldbusanbindung auch die vollständige Antriebsregelung auf diesem Baustein zu implementieren und auf den Einsatz eines DSPs zu verzichten. Sie bieten außerdem die Basis für die Umsetzung neuer Funktionen im Antriebsregler, die sich auf DSP-basierten Systemen nicht oder nur schwer umsetzen lassen.

1.1 Problemstellung

Die in Werkzeugmaschinen eingesetzten Vorschubantriebe sind zumeist elektrische Servoantriebe, die ein Werkzeug oder Werkstück für die Bearbeitung bewegen. Dabei geht der Trend auch bei Spindeln, Hilfsaggregaten und Hilfsachsen zunehmend zur Servoregelung, da sich hiermit höhere Genauigkeiten, Leistungssteigerungen und Energiebedarfsreduktionen erzielen lassen.

Die Bewegung der Antriebe wird meistens von einer Numerischen Steuerung (NC) oder einer Bewegungssteuerung (Motion Control, MC) in Form von Sollpositionen oder -geschwindigkeiten vorgegeben. Die Antriebsregler haben die Aufgabe, die Antriebe so anzusteuern, dass sie den Sollwerten so dynamisch und exakt wie möglich folgen. Daneben besteht die Anforderung zusätzliche Funktionalitäten, wie z. B. Inbetriebnahme- und Überwachungsfunktionen oder Prozessregelungen auf den Antriebsreglern zu integrieren, bzw. die übergeordnete Steuerung mit entsprechenden Daten in Echtzeit zu versorgen.

Die Regelung des einzelnen Antriebs erfolgt üblicherweise in kaskadierten Reglern für Lage, Geschwindigkeit bzw. Drehzahl und Strom. Die Stellgröße ist ein Strom, der von einem Umrichter in die Motoren eingeprägt wird.

Die auf dem Markt befindlichen Antriebsregler bestehen meist aus einer geschlossenen, DSP- oder mikrocontrollerbasierten Hardware, die an die Anforderungen des vom Antriebshersteller vorgesehenen Anwendungsspektrums angepasst ist. Die Antriebsregler verfügen über spezielle Schnittstellen zu den Sensoren und Aktoren der Maschine, sowie zur übergeordneten Steuerung. Die Reglung läuft in Zykluszeiten von typischerweise 62,5 µs für den Stromregler und 125-250 µs für den Drehzahlregler, wodurch sie sich deutlich von der Zykluszeit der übergeordneten Steuerung unterscheiden, die im Bereich einiger Millisekunden liegt.

Eine Anpassung eines Antriebsreglers an eine spezielle Achse, das Hinzufügen von Sonderfunktionalitäten oder der Austausch von Funktionalitäten ist in der Regel mit einem hohen Aufwand verbunden, da sie meist einen Eingriff in die Hardware und in die komplexen zeitlichen Abläufe der Software bedeuten. Wenn für den Antriebsregler spezielle ASICs produziert wurden, ist eine spätere Änderung immer mit einer kostspieligen Neuentwicklung des ASICs verbunden. Auf Einzelanforderungen und Sonderlösungen kann daher in den meisten Fällen aus wirtschaftlichen Gründen keine Rücksicht genommen werden. Beispiele für Sonderfunktionalitäten sind hier die Berücksichtigung des Beschleunigungssignals in der Regelung, die verbesserte Geberauswertung durch Oversampling, oder die Integration von Prozessreglern in die Antriebsregelung.

Jedoch ist sowohl im Bereich der Kombinationsmaschinen oder rekonfigurierbaren Maschinen als auch bei hochdynamischen Anwendungen und Prozessen eine flexible und modulare Anpassung des Antriebsreglers an die jeweiligen Gegebenheiten und Anforderungen gewünscht. Viele der auf dem Markt verfügbaren Geräte bieten zwar unterschiedliche Abstufungen von Antriebsreglern an, die unterschiedlich viele Antriebe und Gebersysteme unterstützen, jedoch ist der Maschinenbauer auf die angebotenen Varianten festgelegt und er kann die Geräte nicht entsprechend der exakt benötigten Anzahl von Antrieben und Gebern frei wählen. Der Maschinenbauer bezahlt für seine besondere Anwendung mehr Funktionen, als er letztendlich benötigt. Die Modularität hilft an dieser Stelle, dem vorherrschenden Kostendruck in der Automatisierungstechnik entgegenzuwirken.

Es ist daher notwendig, ein neues Konzept für Antriebsregler zu entwickeln, dass zum einen die Möglichkeiten neuer Hardware ausschöpft und zum anderen die heute, verstärkt durch die hohen Kundenanforderungen bzgl. Dynamik und Effizienz, gestellten Anforderungen erfüllen kann. Ziel ist es, ein leistungsfähiges, offenes und modulares Antriebsreglersystem zu schaffen.

1.2 Lösungsansatz

Um den wachsenden Anforderungen an die Antriebsregelung jetzt und auch zukünftig gerecht zu werden, soll ein neues Antriebsreglerkonzept entwickelt werden. Es wird dabei ein offenes und modulares Konzept angestrebt. Durch die Offenheit können mit wenig Aufwand auch langfristig neue Anforderungen erfüllt und Sonderfunktionalitäten in den Antrieb integriert werden. Mit Hilfe von Offenheit und Modularität kann das Leistungspotenzial einer Maschine besser genutzt werden, da sich die Antriebsregelung speziell auf die Maschine und den Prozess abstimmen lässt und Neuentwicklungen schnell und mit geringem Aufwand integriert werden können.

Es gibt unterschiedliche Wege, wie ein Antriebsregler offener und flexibler gestaltet werden kann. Die auf dem Markt befindlichen Antriebsregler verfügen schon heute zum Teil über austauschbare Hardwaremodule, um über unterschiedliche Feldbussysteme mit einer Vielzahl von Steuerungen kommunizieren zu können. Da im industriellen Umfeld eine große Vielfalt unterschiedlicher Bussysteme, wie in Bild 1-1 dargestellt, existiert, ist diese Modularität zwingend notwendig, um auf dem internationalen Markt bestehen zu können. Auch sind softwareseitig schon heute einzelne Komponenten der Antriebsregelung auswählbar.

Das gestellte Ziel, ein offenes Konzept für einen Antriebsregler zu entwickeln, lässt sich mit der marktüblichen DSP-basierten Hardware in Teilen schon erreichen. Jedoch muss hier insbesondere bei Aufgaben mit sehr kurzen Zykluszeiten oder sehr rechenintensiven Algorithmen dem DSP zusätzliche externe Hardware zur Seite gestellt werden. Daher wird für ein neues Konzept nicht an einer DSP-Lösung festgehalten, sondern es werden die aktuellen Entwicklungen im Bereich der Flexibilität, Offenheit und Austauschbarkeit in der PC-Technik und bei programmierbaren Logikbausteinen (FPGAs) in Hinblick auf die sich dadurch ergebenden Perspektiven für die Antriebsregelung betrachtet.

Viele neu entwickelte Methoden und Verfahren, die zur Steigerung der Dynamik oder Erweiterung des Funktionsumfangs der Maschine beitragen, müssen aufgrund ihrer Antriebsnähe auf dem Antriebsregler umgesetzt werden. Viele dieser Verfahren können in den existierenden Antriebsreglern nicht realisiert werden, da sie entweder sehr hohe Anforderungen an die Rechenkapazitäten, die Rechengeschwindigkeiten oder die Zykluszeiten des Antriebsreglers stellen und wurden daher auf zusätzlicher, größtenteils FPGA-basierter Hardware umgesetzt.

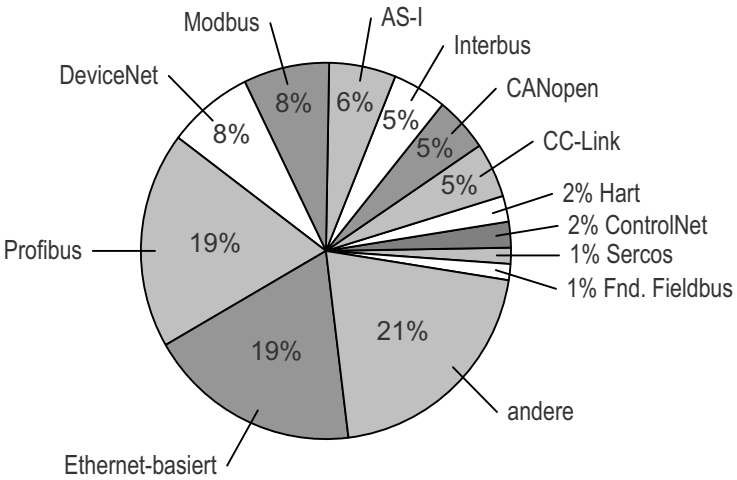

Bild 1-1: Marktanteile industrieller Netzwerke /1/.

Anhand der heute durch den Stand der Technik geltenden Anforderungen, sowie der neuen Anforderungen, die sich aus der Forschung für die Zukunft ergeben, wird ein umfassendes Konzept für einen modularen, offenen Antriebsregler abgeleitet. Dieses Konzept legt fest, welche Hardware verwendet wird, um die notwendigen Rechenkapazitäten und -geschwindigkeiten zu erreichen. Für die Offenheit werden Schnittstellen für Hard- und Softwarekomponenten festgelegt, die jederzeit eine Erweiterung des Systems zulassen.

Es muss eine Softwarearchitektur definiert werden, die die durch die Hardware und die Schnittstellen gegebene Offenheit auch softwareseitig realisiert. Es muss gewährleistet werden, dass einzelne Funktionalitäten dem System hinzugefügt, erweitert oder

ausgetauscht werden können. Dabei muss die volle Leistungsfähigkeit der Hardware für die Funktionen nutzbar sein.

Des Weiteren werden umfangreiche Entwicklungswerkzeuge benötigt, die eine effiziente und sichere Programmierung, Konfiguration und Anwendung des Antriebsreglers für die verschiedenen Nutzungsebenen ermöglicht.

2 Stand der Forschung und Technik

Die Steuerungs- und Antriebstechnik der Vorschubsachsen und Spindeln einer typischen Werkzeugmaschine oder Handlingachse besteht aus den Wegmessgebern, den Motoren, den Umrichtern, dem Antriebsregler und der NC-Steuerung oder Motioncontrol, welche die Lagesollwerte der einzelnen Achsen generiert. Diese werden zumeist über einen digitalen Feld- oder Antriebsbus an die Antriebsregler übertragen. Je nach Ausführung beinhaltet die Motioncontrol bzw. NC-Steuerung, im Folgenden zusammenfassend als Steuerung bezeichnet, eine Bahnvorbereitung, Sollwertinterpolation und die Lageregler sowie für Sonderkinematiken, wie z. B. Gantryachsen, eine achsübergreifende Regelung.

Der Antriebsregler beinhaltet eine Feininterpolation, die die im Interpolations- oder Lagereglertakt kommenden Sollwerte für höher getaktete Regelung aufbereitet. Diese Regelung besteht aus einem Drehzahl- oder Geschwindigkeitsregler, einem Stromregler und ggf. einem weiteren antriebsreglerseitigen Lageregler. Des Weiteren können sich verschiedene Soll- und Istwertfilter, Begrenzer, Referenzmodelle und weitere Verfahren zur Verbesserung der Regelgüte im Antriebsregler befinden.

Die Antriebsregler bestehen üblicherweise aus einem speziell für die Antriebsregelung angepassten digitalen Signalprozessor (DSP). Die Software der Antriebsregelung lastet mit der Regelung, den Filtern und einer Vielzahl von Überwachungsfunktionen meist vollständig aus, sodass kaum eine Möglichkeit besteht, nachträglich anwendungsspezifische Software hinzuzufügen. Spezielle Funktionalitäten, die besonders hohe Taktraten benötigen, wie z. B. die Stromregelung, die PWM-Erzeugung und die Schnittstellen zu Steuerung, Sensorik und Aktorik, bestehen häufig aus festverdrahteten Komponenten. Das Bild 2-1 zeigt den typischen Aufbau DSP-basierter Antriebsregler. Anwenderspezifische Sonderfunktionen oder Funktionserweiterungen ziehen meist eine aufwendige und teure Überarbeitung der Schaltung und des Platinenlayouts nach sich.

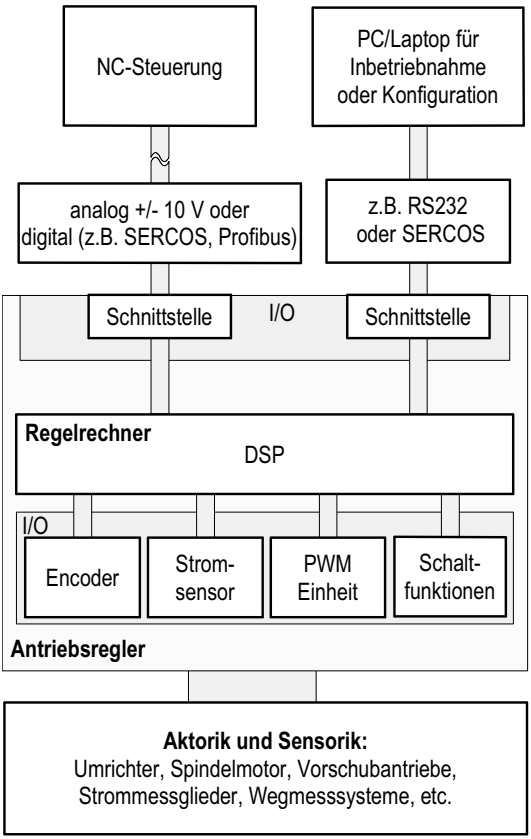

Bild 2-1: Schematischer Aufbau eines DSP-basierten Antriebsreglers /2/.

Für die Realisierung einer flexibel an Kundenanforderungen anpassbaren Antriebsreglereinheit muss eine Plattform bereitgestellt werden, die sowohl bzgl. der **Softwarearchitektur** als auch im Hinblick auf die eingesetzte **Hardwareplattform** eine einfache Skalierbarkeit und Funktionserweiterung zulässt. Weiterhin ist entscheidend, welche Methoden für eine schnelle **Projektierung** der Hard- und Softwaremodule bereitstehen. Nachfolgend wird anhand dieser drei Kriterien der Stand der Technik und Forschung beschrieben.

2.1 Offene Softwarearchitekturen für die Antriebsregelung

Die große Vielfalt an Prozessen und Arbeitsumgebungen, für die Maschinen gebaut werden, erfordern eine ebenso große Vielfalt an anwendungsspezifischen Steuerungs- und Regelungsfunktionen. Diese Vielfalt kann aus wirtschaftlichen Gründen nur zu einem kleinen Teil durch den Steuerungs- oder Antriebsreglerhersteller abgedeckt werden. Um eine umfassend angepasste Steuerungs- und Antriebstechnik für jeden Maschinentyp realisieren zu können, ist eine Öffnung von Steuerung und Antriebsregler für anwendungs- und maschinenspezifische Anpassungen und Erweiterungen durch den Maschinenhersteller notwendig. Auf der Antriebsreglerebene steht sie nur sehr eingeschränkt zur Verfügung.

Unter den verfügbaren **Antriebsreglern** gibt es derzeit unterschiedliche Ansätze, dem Anwender Offenheit zur Verfügung zu stellen.

- Die meisten Hersteller bieten insbesondere bei der Feldbusankopplung an die Steuerung Schnittstellenmodule für verschiedene Systeme an. Beispiele dafür sind unter anderem die Antriebsreglermodelle ACOPOS der Firma B&R /3/ sowie Compax3 der Firma Parker Automation /4/, bei denen sich unterschiedliche Feldbusankopplungsmodule einstecken lassen.
- Für einige Antriebsregler gibt es wählbare Einsteckmodule für verschieden Gebersysteme, wie beim Antriebsreglermodell ACOPOS der Firma B&R.
- Eine sehr eingeschränkte Offenheit bietet eine Reihe von Herstellern an, die in ihren Antriebsreglern eine kleine Auswahl von Sonderfunktionalitäten bereitstellen, die sich durch Parametrierung aktivieren lassen. So besteht z. B. bei dem Antriebsregler Compax3 die Möglichkeit, über Parameter eine Reglerstruktur zu aktivieren, die eine Regelung mit Beschleunigungssensoren ermöglicht.

Keines der verfügbaren Systeme bietet jedoch dem Anwender die uneingeschränkte Möglichkeit, eigene Funktionen und Algorithmen im Antriebsregler zu integrieren, noch auf antriebsreglerinterne Signale in einem beliebigen Takt zuzugreifen.

Offenheit wird durch die Hersteller derzeit hauptsächlich auf der **Steuerungsebene** angeboten. Zu den auf dem Weltmarkt führenden vertretenen Steuerungen gehören die offenen NC-Steuerungen von Fanuc und Siemens, die unterschiedlich weitreichende

Offenheit bieten. Ein bedeutender, darüber hinaus gehender Standard für offene Steuerungen wird durch OSACA definiert, welcher die Basis für die Produkte einiger anderer Anbieter bildet.

2.1.1 Begrenzt modulare Systeme: Fanuc Open CNC und Siemens 840D

NC-Steuerungen von Fanuc und Siemens bieten dem Anwender die Möglichkeit, eigenen Code auf unterschiedlichen Ebenen auszuführen. Neben der bei nahezu allen Steuerungen vorgesehenen Erweiterung um Schaltfunktionen auf der Ebene der speicherprogrammierbaren Steuerung (SPS) stehen bei beiden Herstellern auch Schnittstellen zur Verfügung, um auf der Ebene der Benutzerschnittstelle (HMI) sowohl eigene Bedienmasken als auch Hintergrunddienste in Nicht-Echtzeit zu starten. Darüber hinaus stellt Fanuc mit den „High Level Tasks“ des „C-Language Executor“ die Möglichkeit bereit, schnelle, echtzeitnahe Prozesse für eine adaptive Regelung oder Prozessbeobachtung zu realisieren. Für harte Echtzeitanforderungen wird eine eigene, in C programmierbare Hardwareerweiterung („Customer’s Board“) angeboten /5/.

Siemens geht bei der Offenheit seiner 840D Steuerung bzgl. der Integration von Anwenderroutinen in den Steuerungs-Kernel einen Schritt weiter. Neben der Möglichkeit, Anwendermasken und Hintergrunddienste in verschiedenen Programmiersprachen (standardmäßig in C++ mit Qt) zu realisieren, kann der Anwender durch die NC-Kern Open Architecture (kurz NCK-OA) /6/ eigene Funktionen programmieren, die in harter Echtzeit auf der Ebene des NC-Kerns eingebunden werden. Die Funktionen werden, wie in Bild 2-2 gezeigt, zyklisch oder zu definierten Zeitpunkten oder Ereignissen ausgeführt (Event Binding). Der Anwender hat zu einem ausgewählten Satz von Variablen der Steuerung Lesezugriff und kann in bestimmte Variablen zurückschreiben. Sowohl die Zeitpunkte, an denen eine Funktion ausgeführt wird, als auch die zu lesenden und beschreibenden Variablen sind von Siemens festgelegt.

Diese Offenheit gewährt dem Anwender einen eingeschränkten Eingriff in die Steuerungssoftware. Die durch OSACA definierten Eigenschaften werden dabei weitestgehend erfüllt. Die Anwendersoftware ist modular aufgebaut und wird über klar definierte Schnittstellen in die Steuerung integriert. Die Steuerung kann jederzeit durch Anwendersoftware ergänzt werden. Skalierbarkeit und Herstellerunabhängigkeit werden durch die Siemens-Steuerung jedoch nicht umgesetzt. Ein zusätzliches Problem ist die

Wahl der Programmiersprache und die Einbindung der Anwenderfunktionen. Da diese in ANSI C entwickelt und dem System als kompilierte Binärdateien hinzugefügt werden, kann es durch Fehler in der Anwenderfunktion zu einem Absturz des Gesamtsystems kommen. In vielen Fällen kann der Verursacher des Absturzes (Siemens oder OEM-Entwickler) nicht mehr nachvollzogen werden. Die Umgebung der NCK-OA-Entwicklung ist dementsprechend hochpreisig und steht somit nur wenigen Werkzeugmaschinenherstellern sowie ausgewählten Universitätsinstituten und sogenannten Solution Providern zur Verfügung.

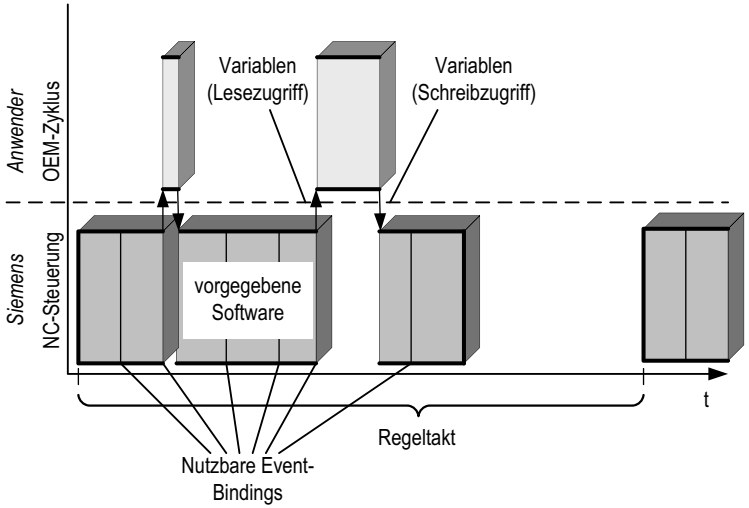

Bild 2-2: Vereinfachte Darstellung der Funktionsweise von OEM-Zyklen.

Den Systemen von Fanuc und Siemens sind mehrere Eigenschaften gemeinsam:

- Sie erlauben nur auf wenige Antriebsparameter einen Schreib- und/oder Lesezugriff.
- Der Zugriff auf Parameter wird in Echtzeitroutinen deutlich gegenüber Nicht-Echtzeitroutinen eingeschränkt.

- Maschinenhersteller bzw. OEMs können dem bestehenden Funktionsumfang lediglich Routinen hinzufügen, aber keine bereits implementierten Funktionen ersetzen.

2.1.2 Erweiterte Offenheit für kommerzielle Systeme: OSACA

OSACA ist ein europäisches Verbundprojekt, an dem sich mehrere Maschinen- und Steuerungshersteller, sowie Forschungseinrichtungen beteiligt haben. Ziel des Projekts war es, eine herstellerübergreifende Steuerungsarchitektur zu entwickeln. Die OSACA-Architektur sieht eine Trennung von Hardware, Systemsoftware einerseits und der Anwendersoftware andererseits vor. Hardware und Systemsoftware bilden eine Systemplattform, die proprietär ausgeführt sein kann und selbst nicht zum Definitionsumfang von OSACA gehört. Sie muss lediglich OSACA-konforme Schnittstellen aufweisen, um beliebige Anwendersoftwaremodule aufnehmen zu können, wie in Bild 2-3 gezeigt. Die Anwendersoftware, die die eigentlichen Steuerungs- und Service-Funktionen realisiert, wird über eine einheitliche Programmierschnittstelle sowie einen modularen Aufbau der Anwendersoftware gekennzeichnet.

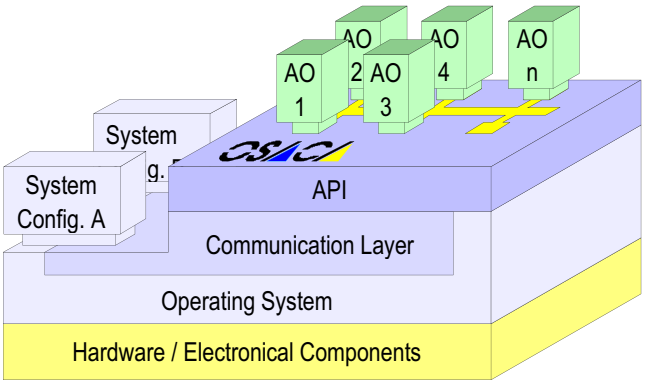

Bild 2-3: Schematischer Aufbau der OSACA-Plattform /7/.
API: Application Programming Interface, AO: Application Object

Die durch OSACA definierte Steuerungsarchitektur zeichnet sich insbesondere durch die folgenden Eigenschaften aus /8/:

- Portabilität: Ein Modul kann in verschiedenen Produkten bzw. Umgebungen eingesetzt werden.
- Erweiterbarkeit: Es muss möglich sein, nachträglich Module zu ergänzen.
- Austauschbarkeit: Module können durch vergleichbare Module, z. B. Von einem anderen Hersteller ersetzt werden.
- Skalierbarkeit: Die Leistungsfähigkeit des Systems lässt sich durch Hinzufügen oder Entfernen von Modulen variieren.
- Kombinierbarkeit: Module sind in der Lage, mit anderen Modulen zusammenzuarbeiten.

Diese Eigenschaften sind charakteristische und grundlegende Eigenschaften, die teilweise oder vollständig von offenen Systemen erfüllt werden. Das Konzept von OSACA wird von einigen Herstellern weitestgehend umgesetzt, jedoch wird die definierte vollständige Offenheit hauptsächlich intern genutzt. Nur wenige der Schnittstellen werden für den Anwender zur Verfügung gestellt. Eingesetzt wird diese Architektur unter anderem von der ISG /9/ für ihren Steuerungskern, sowie in den Derivaten von z. B. Fagor /10/ und Beckhoff /11/.

Die Schnittstelle zu den Antrieben befindet sich bei OSACA oberhalb der Antriebsregler. Für die Soll- und Istwerte von Strom, Geschwindigkeit und Lage sind Variablen definiert, die von einem Antriebsregler, eventuell über einen Feldbus, gelesen und geschrieben werden. Der Antriebsregler selbst wird nicht spezifiziert.

Nachfolgend zum Projekt OSACA, das vor allem auf die Umsetzung der Steuerung auf einem PC mit Windows-Betriebssystem zielte, wurde das Projekt OCEAN /12/ umgesetzt. Im Projekt OCEAN setzt man ebenfalls auf eine modulare Steuerungsarchitektur, jedoch liegt hier der Schwerpunkt auf der Echtzeitfähigkeit des Steuerungssystems, insbesondere dem echtzeitgerechten Datenaustausch zwischen den Steuerungskomponenten. So lassen sich auch echtzeitkritische Erweiterungen, wie z. B. eine SPS, Sicherheitsfunktionen oder eine Prozessregelung integrieren. Die in diesem Projekt entstandene Kommunikationsplattform, sowie alle definierten Schnittstellen stehen quelloffen und lizenzfrei zur Verfügung und ermöglichen auch klein- und mittelständigen Unternehmen einen kostengünstigen Einstieg in die Steuerungstechnik.

Die Idee einer quelloffenen und lizenzkostenfreien Steuerung wurde anschließend im Projekt OSADL /13/ aufgegriffen. Hier wird die Entwicklung durch eine Interessengemeinschaft verschiedener Maschinen- und Steuerungshersteller finanziert und durchgeführt. Die entstehende Steuerung kann von jedem lizenzfrei genutzt und weiterentwickelt werden, solange die Weiterentwicklungen und Verbesserungen anschließend der Allgemeinheit wieder zur Verfügung gestellt werden.

Beide Steuerungsarchitekturen, OCEAN und OSADL berücksichtigen jedoch nicht die Antriebsregelung, sondern ziehen ihre Grenze zum Antrieb bei den Soll- und Istwerten für Strom, Geschwindigkeit und Lage.

2.1.3 **Vollständige Offenheit: Forschungsprojekte SOFIA und UMOAC**

Es wurden bereits mehrere Forschungsprojekte initiiert, die es zum Ziel hatten, ein flexibleres, offenes Konzept für Steuerung *und* Antriebsreglung zu entwickeln.

Das BmBF-Projekt SOFIA (Modulares **Sof**twaresystem für **i**ntelligente **A**ntriebe) des Instituts für Prozessrechentechnik, Automation und Robotik (IPR) in Karlsruhe /14/ stellt ein offenes Antriebsreglersystem vor, das angelehnt an die offene Steuerungsplattform OSACA aus modularen Softwarebausteinen besteht und das vom Anwender nach seinen Bedürfnissen konfiguriert werden kann, wie in Bild 2-4 gezeigt. Die Hardware dieses Systems besteht im Wesentlichen aus einem PC bzw. einer PC-basierten Hardware, auf der ein Echtzeitbetriebssystem installiert ist. Die einzelnen Funktionsblöcke umfassen Funktionen zur Überwachung und Diagnose, Schnittstellen zu einer Leitsteuerung, Regler- und Bahnoptimierung und Regler zur Strom-Geschwindigkeits- und Lageregelung und befinden sich ausschließlich auf Softwareebene.

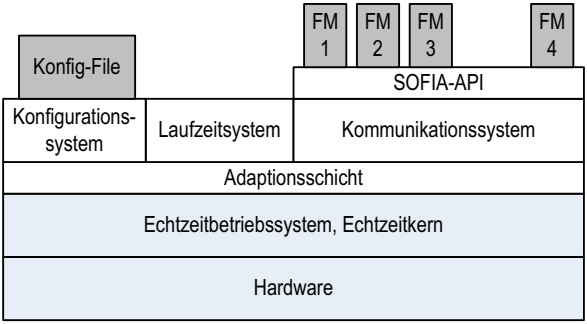

Bild 2-4: Schematischer Aufbau von SOFIA /14/.

Die Echtzeitanforderungen, denen SOFIA genügt, liegen bei Regeltakten von 1 ms und einer Synchronität im Mikrosekundenbereich, wie sie im einfachen Positionierbetrieb und bei Handlingachsen üblich sind. Die typischen Anwendungsfälle ergeben sich dadurch insbesondere für die Robotik. Der Aufbau von Regelkreisen aus einzelnen Modulen, die den Anforderungen von hochdynamischen Werkzeugmaschinen mit Drehzahlregeltakten von 125 µs und Stromregeltakten von weniger als 62,5 µs entsprechen, ist innerhalb dieses Projekts nicht vorgesehen. Anwendungen mit sehr hohen Taktraten bzw. sehr hardwarenahe Anwendungen werden nicht berücksichtigt.

Ein weiteres Konzept für eine offene Antriebsreglerarchitektur wurde an der Universität von Michigan untersucht. Das Projekt UMOAC (**U**niversity of **M**ichigan **O**pen **A**rchitecture **C**ontroller) /15,16/ befasst sich mit einer offenen Architektur, die ebenfalls Funktionen als Softwaremodule zusammenfasst. Ziel der Architektur ist es, möglichst viele Module in verschiedenen Anwendungsfällen wiederverwenden zu können. Dazu werden maschinenabhängige Module für hardwarenahe Funktionen und maschinenunabhängige Module für übergeordnete Funktionen unterschieden. Die hardwarenahen Module sind Treiber, die direkt oder über einen Feldbus auf angeschlossene Hardware zugreifen können. Die maschinenunabhängigen Module beinhalten z. B. Überwachungen oder Regler. Sie greifen über eine einheitliche Schnittstelle auf die Treibermodule zu, sodass sie unabhängig von der angeschlossenen Hardware arbeiten können. Der modulare Aufbau der Architektur wird in Bild 2-5 gezeigt.

Bisher gibt es eine Umsetzung des Open Architecture Controller auf einem PC-basierten System mit einem Echtzeitbetriebssystem, die den modularen Austausch von

Regelalgorithmen nicht unterstützt. Die Antriebsregelung, bestehend aus Strom- und Drehzahlregelung, ist als ein in sich geschlossenes Modul implementiert.

Die Forschungsarbeiten fokussierten sich im Laufe des Projekts auf die Softwareentwicklung für offene Systeme und auf die Echtzeitkommunikation zwischen Systemkomponenten. Wie auch bei dem Projekt SOFIA werden die Echtzeitanforderungen hochdynamischer Werkzeugmaschinen von UMOAC nicht erfüllt, da auch hier nur Reglertakte im Millisekundenbereich erreicht werden können.

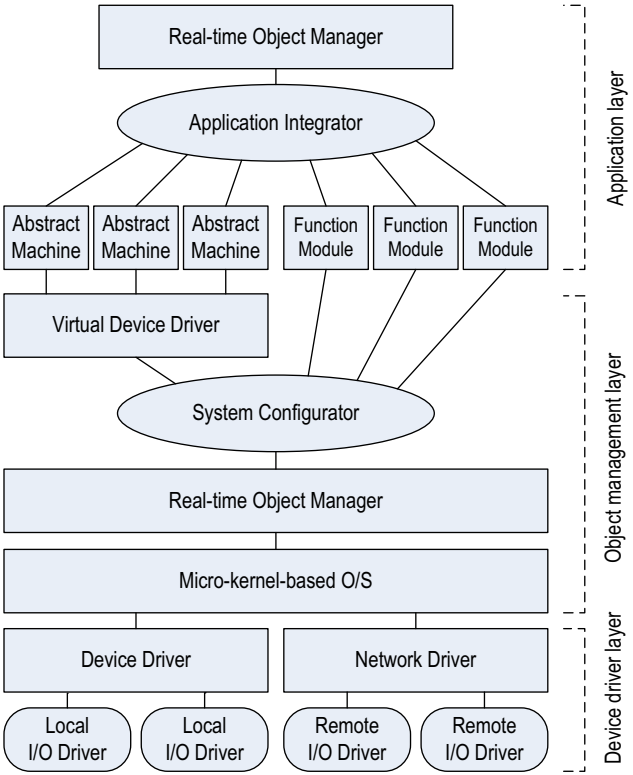

Bild 2-5: Aufbau der UMOAC-Architektur /15/

2.1.4 Zusammenfassende Betrachtung zur Offenheit von Softwarearchitekturen

Offene Softwarearchitekturen beschränken sich im kommerziellen Bereich derzeit auf die Steuerungsebene. Hier muss jedoch differenziert werden, ob es sich um eine ausschließlich intern genutzte Offenheit, eine eingeschränkt nutzbare Offenheit oder eine vollständige Offenheit handelt. Vollständig offene Steuerungen, wie sie z. B. im Projekt OSADL entstehen, haben bislang vorwiegend Forschungscharakter und müssen sich auf dem Markt noch durchsetzen.

Alle offenen Softwarearchitekturen beschränken sich auf die Umsetzung auf einer PC-basierten Hardware und schließen die Antriebsregelung nicht als offenes Element mit ein. Stattdessen definieren sie eine Soll- und Istwertschnittstelle, über die mit externen Antriebsreglern kommuniziert werden kann. Eine antriebsreglerübergreifende Softwarearchitektur fehlt bislang.

2.2 Offene Hardwareplattformen für die Antriebsregelung

Die Offenheit beschränkt sich im Allgemeinen zumeist auf die Software von Steuerungs- und Antriebsreglersystemen. Aufgrund der sehr hohen Rechengeschwindigkeiten in der Antriebsregelung ist die Einschränkung der Betrachtung auf reine Softwarelösungen nicht ausreichend. Einige Funktionen müssen aufgrund hoher Rechenanforderungen oder der direkten Anbindung an z. B. Sensoren und Aktoren von speziellen Hardwarekomponenten und eigens dafür gefertigten Chips erfüllt werden.

In der Chipfertigung gibt es ebenfalls Ansätze für offene, modulare Architekturen, die die gleichen Ziele verfolgen, wie die offenen Softwarearchitekturen.

Die elektrische Antriebsregelung wurde schon lange vor der Entwicklung der Digitaltechnik eingesetzt, und zunächst mit Hilfe von analogen Schaltungen realisiert. Die ersten Ansätze für eine digitale Antriebsregelung, sowie die zu erwartenden Vorteile werden in /17/ beschrieben. Erst seitdem die Digitaltechnik ausreichende Geschwindigkeit und Auflösung bieten kann, sodass die Zeit- und Größenquantisierung fein genug ist, um eine stabile und dynamische Regelung zu realisieren, setzen sich Mikrocontroller- und DSP gestützte Systeme durch. In /18/ wird eine digitale Regelung für Asynchronmotoren vorgestellt. Die immer weitergehende Optimierung der Antriebsregelung, sowie die wachsenden Anforderungen an Flexibilität und

Funktionsumfang führten mitunter dazu, dass zunehmend mehr Funktionen der Antriebsregelung in eigene digitale Schaltnetzwerke ausgegliedert werden, die entweder als festverdrahtete Hardware in speziellen DSPs oder ASICS /19,20/ realisiert sind, oder flexibel in programmierbarer Logik umgesetzt werden.

2.2.1 Analoge Antriebsregelung

Die ersten Antriebsregler wurden bereits in den 50er und 60er Jahren vollständig in analoger Hardware aufgebaut /21/. Analogtechnik hat gegenüber der Digitaltechnik den Vorteil nicht auf diskrete Zustände und Werte eingeschränkt zu sein. Die Auflösung der Signale, sowie die Zeitauflösung eines analogen Systems ist theoretisch unendlich hoch, da ein Diskretisierungsfehler und das dadurch hervorgerufene Diskretisierungsrauschen nicht existieren. Die Probleme diskreter Regelung, die sich insbesondere in der mangelnden Stabilität von zu niedrig auflösenden Systemen ergibt, bestehen nicht. Jedoch beeinflusst das analoge Rauschen der Signale die Stabilität des Reglers. Die verwendeten Bausteine sowie die Signalwege auf den Leiterplatten müssen daher mit besonderer Sorgfalt gewählt werden, um das analoge Rauschen innerhalb des Regelkreises minimal zu halten /22/. Ein weiteres Problem bereiten Signaloffsets sowie das temperaturabhängiges Driften der Eingänge von analogen Komponenten. Dies kann kaum oder nur mit großem Aufwand kompensiert werden.

Es gibt im analogen Regler zwar keine Geschwindigkeitseinschränkung durch einen Systemtakt, wie dies in abgetasteten, digitalen Systemen der Fall ist, jedoch begrenzt das Einschwingverhalten der eingesetzten analogen Bauteile und Schaltgruppen die Geschwindigkeit des Gesamtsystems, so dass hier nur eine beschränkte Regelbandbreite erreicht werden kann.

Der Regler selbst ist aus einer analogen Schaltung verschiedener elektronischer Bauteile aufgebaut. Mit Hilfe von Widerständen, Kapazitäten, Induktivitäten und Verstärkern können die mathematischen Gleichungen eines Regelkreises abgebildet und, meist über verstellbare Widerstände, parametriert werden. Die eingestellte Parametrierung ist nur schwer exakt zu erfassen und damit kaum auf weitere Antriebsregler übertragbar. Jeder Antrieb muss einzeln in Betrieb genommen werden. Der Antriebsregler ist somit ein sehr unflexibles System, das lediglich eine Parametrierung zulässt. Eine weitergreifende

Veränderung ist nur durch Änderung der Hardware zu erreichen. Selbst minimale Variationen des Regleralgorithmus erfordern einen Hardwareeingriff.

Die Flexibilisierung eines analogen Antriebsreglers ist nur durch aufwendige Hardwarebaukastensysteme realisierbar, die für jede Variation eigene Hardwarekomponenten bereitstellen.

2.2.2 Digitale, DSP- oder mikrocontrollerbasierte Antriebsregelung

Durch die Fortschritte in der Digitaltechnik der letzen Jahre kann in einem digitalen System sowohl die zeitliche Auflösung sowie die Signalauflösung erreicht werden, die für eine dynamische, stabile Regelung eines Antriebs notwendig ist. Seit Beginn der 80er Jahre setzen sich in der Automatisierungstechnik die digitalen Antriebsregler durch /23/. Heutige Vertreter sind z. B. die *SimoDrive*-Serie von Siemens /24/, *bmaxx 4000* von Baumüller/25/, oder die *IndraDrive*-Familie von Bosch-Rexroth /26/. Die wesentlichen Gründe dafür sind die Reproduzierbarkeit und Übertragbarkeit der Reglerparameter sowie die Übertragbarkeit der Reglerprogramme auf andere Systeme /27/. Es können Parametersätze für einen Antriebstyp bzw. Maschinentyp eingestellt werden und auf alle gleichen Antriebe und Maschinen kopiert werden. Dadurch wird der Inbetriebnahmeaufwand einer Maschine deutlich reduziert.

Ein weiterer Vorteil der digitalen Antriebstechnik ist die flexiblere Programmierung des Antriebsreglers. Änderungen und Erweiterungen von Funktionen und Algorithmen können weitestgehend ohne eine Änderung der Hardware durchgeführt werden. Die Entwicklung von Antriebsreglersystemen wird dadurch wesentlich einfacher und billiger.

Die Regleralgorithmen, sowie die antriebsreglereigenen Steuerungs- und Überwachungsfunktionen werden in der Regel von einem speziellen Mikrocontroller oder einem digitalen Signalprozessor (DSP) übernommen. In modernen Antriebsreglern wird der Mikrocontroller oder DSP je nach Anforderungen mit sehr hohen Rechentakten betrieben, wie beispielsweise der *GeoBrick* von DeltaTau mit bis zu 240 MHz /28/. Dies ist ausreichend, um auf unterster Reglerebene Stromregeltakte von 31,25 µs für bis zu acht Achsen zu realisieren. Für Funktionen, die mit höheren Takten ausgeführt werden müssen, wie zum Beispiel die PWM-Erzeugung oder eine Encoder-Auswertung, werden

eigene Schaltkreise in den Mikrocontroller integriert. Ein Beispiel für einen auf die Antriebsregelung spezialisierten Mikrocontroller ist der *C2000* von Texas Instruments /29/.

Die digitalen Antriebsregler können sehr flexibel programmiert werden. Durch die eingeschränkte Leistungsfähigkeit der Mikrocontroller bzw. DSPs, und durch die Prozessorarchitektur bedingte Beschränkung, einzelne Funktionen nur sequenziell bearbeiten zu können, erweist sich der Entwurf der Software jedoch als sehr komplex. Um ein stabiles und vorhersagbares Reglerverhalten mithilfe der Gesetze zeitdiskreter Regelungstechnik /30/ garantieren zu können, muss die Software die Echtzeitbedingungen erfüllen, die die Ausführung einzelner Funktionen zu festen Zeitpunkten in konstanten Takten gewährleistet. Ein bestehendes System kann nur begrenzt um weitere Funktionen erweitert werden, ohne dass das zeitliche Verhalten des Systems dadurch beeinflusst wird. Erweiterungen müssen daher meist mit großer Sorgfalt durchgeführt werden.

Für eine Erweiterung der heute verfügbaren DSP-basierten Antriebsregler um zusätzliche Funktionen, wie z. B. die *Schnelle Stromregelung* /31,32/, die eine Signalvorverarbeitung von ca. 5 MHz erfordert und mit Stromreglertakten von weniger als 20 µs arbeitet, ist die Leistungsfähigkeit nicht mehr ausreichend. Solche höheren Anforderungen können heute nur mithilfe zusätzlicher Hardware oder in Zukunft durch schnellere DSPs erfüllt werden.

2.2.3 FPGA-basierte Antriebsregelung

Auf einem programmierbaren Logikbaustein werden prinzipbedingt alle Funktionen als Schaltkreise parallel zueinander realisiert. Dadurch ist die parallele Ausführung immer möglich. Der Ausführungstakt ist dabei der Systemtakt (Clock). Durch die parallele Anordnung logischer Schaltungsnetzwerke ist es daher möglich, innerhalb eines Systemtakts mehrere Rechenoperationen gleichzeitig auszuführen. Die Leistungsfähigkeit eines FPGA-basierten Systems hängt neben dem Systemtakt von der Parallelisierung der auszuführenden Operationen ab. Die möglichen Systemtakte von FPGA-Bausteinen liegen je nach Baustein auf vergleichbarem Niveau mit modernen DSPs. Eine Leistungssteigerung gegenüber dem DSP wird maßgeblich durch die Parallelisierung von Operationen bestimmt. Hierbei sind jedoch zwei Grenzen gesetzt: Zum einen haben

die Logikbausteine nur eine begrenzte Anzahl von programmierbaren Gattern. Es muss jedoch bei der Umsetzung von Algorithmen zwischen Geschwindigkeit, Platzbedarf und den damit einhergehenden Kosten abgewogen werden. Gegebenenfalls ist es notwendig, Operationen in mehreren Schritten unter Mehrfachverwendung von Gattern durchzuführen. Zum anderen verhindert ein Signalfluss zwischen einzelnen Funktionen innerhalb des Systems eine parallele Ausführung dieser Funktionen, da es Signalabhängigkeiten zwischen ihnen gibt. Es muss daher insbesondere dafür Sorge getragen werden, dass alle Funktionen zeitlich optimal aufeinander abgestimmt werden, um die Signallaufzeiten möglichst kurz zu halten.

In der Antriebsregelung gibt es eine Reihe von Operationen, die im Idealfall zeitgleich und in sehr hohen Taktraten ausgeführt werden sollten. Dies betrifft insbesondere die Erfassung der Istgrößen Phasenströme, Rotorwinkel sowie Geschwindigkeit. Diese Messgrößen sollten möglichst ohne zeitlichen Versatz zueinander aufgenommen werden. Die Signalaufbereitung, wie z. B. das Oversampling /33/, ein spezielles Verfahren zur Verbesserung der Spursignale von SinCos-Gebern, erfordert eine Abtastung, die um ein Vielfaches höher ist, als der eigentliche Regeltakt. Diese Aufgaben können auf einem FPGA in eigenständigen Schaltkreisen ausgeführt werden, sodass die Ausführung des restlichen Antriebsreglers nicht beeinflusst wird.

Die FPGA- bzw. CPLD-Technik wird bereits in der kommerziell verfügbaren Antriebstechnik eingesetzt. Dazu wird in der Regel die Hardware vorhandener DSP-basierter Antriebsregler um einen Baustein mit programmierbarer Logik ergänzt. Die Funktionen der Antriebsregelung, die eine Verwendung programmierbarer Logik notwendig machen, werden ausgelagert und unabhängig vom DSP ausgeführt.

Als Beispiel kann der Antriebsregler M-Drive der Firma Baumüller genannt werden, der zusätzlich zu einem DSP, auf dem die Antriebsreglung untergebracht ist, über einen FPGA für die Überabtastung des Lagesignals und die Auswertung eines Beschleunigungssensors verfügt, die in Takten > 1 MHz ausgeführt werden.

Ein weiteres Beispiel ist der digitale Servoregler *DSMRW* der Firma IDAM /34/, in dem sowohl die schnelle Stromregelung, als auch das digitale Oversampling in einem FPGA realisiert sind /35/. Die eigentliche Achsregelung, sowie Motion-Funktionen und Kommunikation werden von einem DSP übernommen.

In nur wenigen auf dem Markt verfügbaren Antriebsreglern bildet ein FPGA die alleinige Basis für die Regelung. Beispiele hierfür sind die Geräte *VECTORDRIVE* von ARADEX /36/, das jedoch ein für den Anwender geschlossenes System bildet, und *eDARC* von Ferrocontrol /37/, das dem Anwender die Möglichkeit bietet, mit Hilfe von Matlab/Simulink die vorhandene Reglersoftware zu erweitern.

2.2.4 PC-basierte Antriebsregelung

Die Fortschritte der PC-Technik, insbesondere durch die kontinuierliche Erhöhung der Prozessortakte, sowie durch die Vervielfachung von parallel arbeitenden Prozessorkernen auf einem Chip, führen zu einem sehr leistungsfähigen Rechensystem, das zunächst für hochdynamische regelungstechnische Aufgaben einsetzbar erscheint.

Die zur Verfügung stehenden Rechenkapazitäten sind um ein Vielfaches höher, als die von DSP-Systemen. Durch den Massenmarkt der PC-Technik ist diese zudem zu deutlich geringeren Kosten erhältlich.

Die modernen PC-Betriebssysteme erfüllen zunächst nicht die gestellten Echtzeitanforderungen und können somit nicht für die Antriebsreglung eingesetzt werden. Über spezielle Betriebssystemerweiterungen kann jedoch in Grenzen eine Echtzeit nachträglich gewährleistet werden. Als Beispiele dafür können die Microsoft Windows-Erweiterungen VxWorks /38/, sowie TwinCAT /39/ und im freien Softwarebereich RTAI /40/ als Erweiterung von Linux genannt werden.

Die mit diesen Betriebssystemerweiterungen erreichte Echtzeitfähigkeit wird nur noch durch die PC-Hardware, insbesondere durch die Mikroprozessorarchitektur eingeschränkt. Die größte Einschränkung PC-basierter Systeme stellt der hardwarebedingte zeitliche Jitter bei der Ausführung von Tasks dar. Die zeitliche Abweichung des Ausführungszeitpunkts eines Tasks wird durch zwei Komponenten bestimmt: der konstanten und berechenbaren Latenz (Verspätung) und dem zufälligen, nicht berechenbaren Jitter. Im folgenden Bild 2-6 werden die zeitlichen Einflüsse von Latenz und Jitter anhand der Reaktion auf ein Ereignis verdeutlicht. Nach Auftreten eines Ereignisses verstreicht eine bestimmte Zeit, bevor das System reagieren kann. Dies wird beispielsweise dadurch verursacht, dass ein Ereignis die Ausführung eines bestimmten Programmteils erforderlich macht, das zunächst aus dem Speicher geladen werden muss.

Diese Latenz ist konstant, da die dazu notwendigen Arbeitsschritte bekannt und ihre Ausführungszeiten deterministisch sind. Sind jedoch Arbeitsschritte erforderlich, deren Ausführungszeiten nicht deterministisch sind, da sie auf Grund von äußeren und unbekannten Einflüssen variieren können, kommt zur planbaren Latenz noch eine zusätzliche, zufällige, als Jitter bezeichnete Verzögerung hinzu.

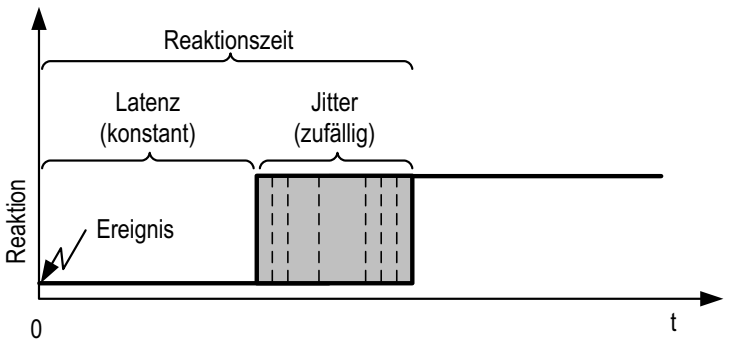

Bild 2-6: Anteile von Latenz und Jitter an der Reaktionszeit

Die digitale Regelungstechnik stützt sich auf die Theorie der zeitdiskreten Systeme /41/, mit der sich abgetastete Systeme beschreiben und auf ihre dynamischen Eigenschaften, insbesondere hinsichtlich ihrer Stabilität, untersuchen lassen. Voraussetzung für diese Betrachtung ist die jitterfreie zeitlich äquidistante Abtastung.

Welchen Einfluss ein Jitter im System auf das Systemverhalten hat, und in wie weit die Systemstabilität dadurch gefährdet wird, kann mathematisch berechnet werden /42,43/ und ist von mehreren Faktoren, insbesondere vom Regeltakt und den dynamischen Eigenschaften der Achse, abhängig. In der Praxis hat es sich bewährt, den Jitter deutlich geringer als 5% des Regeltakts zu halten. Bei den heute gängigen schnellen Regeltakten von 20µs im Stromregler entspricht das dem Jitter von 1 µs.

Die in PC-Systemen eingesetzten Prozessoren mit einer x86-Architektur bieten, um ihre Performance zu erhöhen, eine Reihe von Mechanismen, die einen deterministischen, jitterfreien Betrieb nahezu unmöglich machen.

- Ein Befehls- und Daten-Cache muss, je nachdem welche Tasks als nächstes ausgeführt werden sollen, in seinem aktuellen Zustand gehalten oder komplett neu geladen werden. Dadurch variiert die Zeit, die bei einem Taskwechsel benötigt wird.
- Die Befehle des x86-Prozessors werden in einer Instruction Pipeline bereitgehalten. Die Befehle dieser Prozessorfamilie haben keine einheitliche Länge. Das bedeutet, es können unterschiedlich viele Befehle in der Pipeline stehen, wodurch keine exakten Ausführungszeiten bestimmt werden können.
- Zur Beschleunigung des Prozessors wird die sogenannte Speculative Execution eingesetzt. Dies ist ein Verfahren, bei dem der Prozessor bei geringer Auslastung die verbleibende Rechenzeit dazu nutzt, eine Reihe verschiedener im aktuellen Kontext wahrscheinlicher Berechnungen auszuführen. Wird im weiteren Verlauf des ausgeführten Programms tatsächlich eine dieser Berechnungen benötigt, kann das vorab berechnete Ergebnis direkt zurückgeliefert werden. Auch dies erschwert die Vorhersagbarkeit von Ausführungs- und Taskwechselzeiten.
- Des Weiteren beeinflusst die Anbindung der Peripherie (Speicher, IOs, etc.), die über die Systembusse mit dem Prozessor verbunden sind, das Echtzeitverhalten. Die kann beispielsweise durch DMA-Zugriffe von PCI-Karten auf den Hauptspeicher, die den Zugriff des Prozessors auf den Hauptspeicher verhindern oder durch das Auslösen von Interrupts, die den Prozessor kurzeitig beanspruchen, geschehen.

Der in einem x86-basierten System zu erwartende Jitter ist sehr stark abhängig vom eingesetzten Prozessor, der umgebenden Hardware (Grafikkarten, Netzwerkkarten, Bustakte) und den Testbedingungen. Messungen mit aktuellen Prozessoren in unterschiedlichen Szenarien haben einen Jitter von 3,5 µs bis zu 200 µs ergeben. In Bild 2-7 ist die zeitliche Abweichung eines in einem 500 µs Takt ausgeführten Task über einen Zeitraum von 5 Sekunden gezeigt. Die Messung wurde auf einem Pentium-Prozessor durchgeführt /44/.

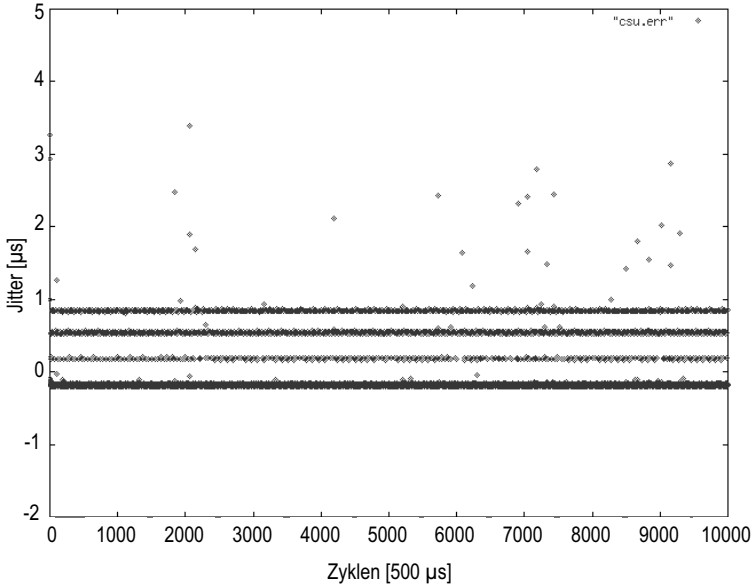

Bild 2-7: Jitter Messung auf einem Pentium Prozessor /44/.

Die zu diesem Zeitpunkt abzusehenden zukünftigen Entwicklungen in der PC-Technik führen dazu, dass die Tasks schneller und teilweise parallel ausgeführt werden können. Dies wird vor allem durch Prozessoren mit mehreren, parallel arbeitenden Kernen erreicht. Jedoch treten die zuvor beschriebenen Probleme der PC-Architektur auch bei mehreren Prozessorkernen auf, sodass die Nutzung von PC-Technik für hochdynamische Antriebsregelung auch langfristig nur eingeschränkt geeignet ist.

2.2.5 Zusammenfassende Betrachtung zur Offenheit von Hardwareplattformen

Betrachtet man die in diesem Kapitel vorgestellten, derzeit zur Verfügung stehenden Hardwarelösungen, lässt sich zusammenfassend festhalten:

- Die Analogtechnik bietet neben mangelnder Parametrierbarkeit und Störunsicherheit nicht die notwendige Flexibilität, die benötigt wird, um neue

Regelkreise aufzubauen oder Regelkreise an kundenspezifische Anwendungen anzupassen.

- Die PC-Architektur ist durch die freie Programmierbarkeit sehr flexibel und durch hohe Taktraten sowie Mehrkernprozessoren sehr leistungsfähig, und erfüllt für die meisten Antriebsregelungsanwendungen die gestellten Echtzeitanforderungen. Bei sehr hohen Anforderungen, insbesondere hohen Regeltakten von 50 kHz können die Echtzeitanforderungen, die an den Antriebsregler gestellt werden, mit der derzeit verfügbaren Hardware nicht erfüllt werden. Es müssen dazu geeignete Maßnahmen ergriffen werden, um den in der PC-Architektur auftretenden Jitter zu reduzieren.
- Spezielle auf die Antriebsregelung angepasste Mikrocontroller und DSPs können die gestellten Echtzeitanforderungen erfüllen und sind sehr leistungsstark. Die antriebsregelungsspezifischen Operationen, die in sehr hohen Takten gerechnet werden müssen, wie z. B. die PWM-Erzeugung oder die mehrkanalige AD-Wandlung, sind im Mikrocontroller als eigenständige Hardware integriert und laufen unabhängig vom eigentlichen Mikrocontroller. Die Anforderung einer flexiblen Plattform lässt sich hier nur unvollständig erfüllen, da diese Operationen durch die Hardware festgelegt sind, und sich nicht ändern lassen.
- Ein FPGA bietet sowohl die notwendige Leistungsfähigkeit, als auch die geforderte Flexibilität, die für eine offene Antriebsreglerplattform benötigt wird. In das Regelsystem können beliebige, frei programmierbare neue und ressourcenintensive Algorithmen und Verfahren eingefügt und parallel in hohen Takten ausgeführt werden, ohne die für das restliche System verbleibende Rechenzeit oder das zeitliche Gefüge zu beeinflussen. Es kann ein modularer, offener Antriebsregler realisiert werden, der in seiner Basisfunktion den herkömmlichen DSP-basierten Antriebsregler ersetzt und darüber hinaus eine offene Plattform für die Integration neuartiger, leistungsintensiver Funktionen bietet und damit auch langfristig die Basis für die zukünftigen Entwicklungen im Bereich der Antriebstechnik bietet. Für flexible Anwendungen erweist sich der FPGA damit als ideal.

Eigenschaft \ Hardware	Analogtechnik	DSP- oder Mikrocontroller-basiert	PC-basiert	FPGA-basiert
Flexibilität	○	◑	◕	●
Parametrierbarkeit	○	●	●	●
Erweiterbarkeit	○	○	●	●
Leistungsfähigkeit	◑	●	●	●
Echtzeitfähigkeit	●	●	◑	●

Tabelle 2-1: Eigenschaften der verschiedenen Hardwarelösungen:
● = Kriterium erfüllt, ◑ = teilweise erfüllt, ○ = nicht erfüllt.

2.3 Methoden für die Hard- und Softwareprojektierung

In der Forschung und Entwicklung gibt es eine Reihe von Methoden und Verfahren, die zu einer Verbesserung der Antriebsregelung beitragen können. Die meisten Forschungsarbeiten wurden bislang auf speziellen Testsystemen oder auf eigens dafür entwickelter Hardware aufgebaut. Für die prototypische Umsetzung von Forschungs- und Entwicklungsergebnissen stehen eine Reihe von Entwicklungsumgebungen und Projektierungswerkzeugen bereit, mit denen zum einen die prinzipielle Funktionsweise entwickelt und geprüft, und zum anderen die Ressourcenanforderungen im Vorfeld untersucht werden können. Diese kommerziell vertriebenen Prototyping-Systeme bieten auch eine wichtige Grundlage für die Gestaltung eines Projektierungswerkzeugs und einer Entwicklungsumgebung für eine industrietaugliche, flexible Antriebsregler-plattform.

2.3.1 DSP-basierte Prototypingsysteme

Die am weitesten verbreitete Form der Prototyping-Systeme ist DSP-basiert. Diese lassen sich leicht in Hochsprachen, oder mit speziellen Mathematik- und Simulations-

softwarepaketen programmieren. Die während der Prototypingphase entwickelten Programme können mit geringem Aufwand auf das Endprodukt übertragen werden.

Eines dieser Systeme ist z. B. das DSP-basierte Prototypingsystem dSPACE /45/, das sich mit Hilfe der Mathematik-Software Matlab /46/ programmieren lässt. Matlab liefert dabei nicht nur ein Programmiersystem, in dem Regler als Software oder auch grafisch programmiert werden können, sondern es kann gleichzeitig auch das programmierte System simulieren. Die Hardwareplattform der dSPACE-Systeme bietet eine Vielzahl von analogen und digitalen Schnittstellen bis hin zur Feldbusankopplung. Alle Schnittstellen sind soweit offen, dass das System um anwenderspezifische Hardware erweitert werden kann. Auf der Basis dieser Systeme lassen sich neue Technologien und Verfahren experimentell testen. Für die Serienfertigung sind sie jedoch zum einen zu teuer und zum anderen nicht auf den industriellen Einsatz in einer Serienmaschine ausgelegt.

2.3.2 FPGA-basierte Prototypingsysteme

Aufgrund der rapiden Weiterentwicklung von FPGAs hinsichtlich ihrer Größe und Geschwindigkeit sowie dem starken Preisverfall dieser Technologie werden diese Bausteine in immer mehr Bereichen eingesetzt. Auch die Entwicklungsumgebungen und Softwarepakete wurden soweit weiterentwickelt, dass aus Mathematik-Programmen, wie z. B. Matlab, eine Hardwarebeschreibung in einer Hardwarebeschreibungssprache für FPGAs automatisch generiert wird, die die programmierten Funktionen auf dem FPGA abbildet.

Ein Beispiel für ein System, das diese Möglichkeiten nutzt, ist das FPGA-basierte System CompactRIO von National Instruments /47/, das sich mit Hilfe von LabView /48/ programmieren lässt, und mit dem FPGA verbundene Steckplätze für unterschiedliche Ein- und Ausgangskarten bietet. Ziel dieser Plattform ist jedoch die prototypische Implementierung und Untersuchung einzelner Algorithmen und Verfahren und nicht der serienmäßige Einsatz in einer Maschine.

2.3.3 Weiterführende Ansätze: Open Core Protocol und OpenCores

Für eine reine FPGA-Entwicklung wurde das Projekt OCP (**O**pen **C**ore **P**rotocol) /49/ gegründet. In diesem Projekt werden einheitliche Schnittstellen, über die Schaltungsgruppen (Cores) miteinander verbunden werden können, definiert. Verschiedene Funktionalitäten werden zu Cores modularisiert, die einzeln getestet und einfach ausgetauscht werden können. Die Entwicklungszeit eines neuen Chips lässt sich dadurch drastisch senken, da man die einzelnen Cores wieder verwenden kann und nicht erneut testen muss.

Ein ähnlicher Ansatz wird vom Opensource-Projekt **OpenCores** /50/ verfolgt. Einzelne Funktionalitäten werden quelloffen mit Hilfe von Hardwarebeschreibungssprachen als Module implementiert. Die Module verfügen entweder über gut dokumentierte, proprietäre Schnittstellen oder über eine Wishbone-Interface Schnittstelle /51/. Wishbone-Interface ist eine sehr schlank und einfach gehaltene Schnittstelle, mit der Platz sparende und schnelle Verbindungen zwischen den einzelnen Modulen hergestellt werden können.

Mit Hilfe der quelloffenen und frei verfügbaren Module können in sehr kurzer Zeit vollständige Schaltungen mit einem umfangreichen Funktionsumfang aufgebaut und an anwendungsspezifische Anforderungen angepasst werden. Die Schwerpunkte von OpenCores liegt derzeit in der Nachrichtentechnik, sowie in der Bereitstellung verschiedener Mikrocontrollerarchitekturen.

2.3.4 Zusammenfassende Betrachtung bestehender Projektierungslösungen

Die verfügbaren Projektierungslösungen bieten einfach zu handhabende und sehr flexible Systeme, die dem Anwender den prototypischen Aufbau von Systemen im Laborumfeld erlaubt. Der Schwerpunkt dieser Systeme liegt auf Geschwindigkeit und Flexibilität sowie der Implementierung von Algorithmen. Sie sind jedoch nicht an die Bedürfnisse der Antriebstechnik angepasst und eignen sich aufgrund ihres Preises und ihres überdimensionierten Funktionsumfangs nicht für den Serieneinsatz in einer Maschine.

Eigenschaft \ Projektierungslösung	DSP-basierte Prototypingsysteme	FPGA-basierte Prototypingsysteme	Opencores / OCP
Flexibilität / Offenheit	◑	●	●
Leistungsfähigkeit	◑	●	●
Handhabbarkeit	◑	●	●
Varianten für die Antriebstechnik	●	◑	◑
Geeignet für den Einsatz in Serienmaschinen	○	◑	◑

Tabelle 2-2: Eigenschaften der verschiedenen Projektierungslösungen:
● = Kriterium erfüllt, ◑ = teilweise erfüllt, ○ = nicht erfüllt.

2.4 Abschließende Analyse des Standes der Technik und Forschung

Der Stand der Technik in der Antriebsregelung ist derzeit geprägt durch eigenständige Antriebsregelgeräte, die auf speziell zugeschnittenen DSPs oder Mikrocontrollern basieren. Sie sind in ihren Schnittstellen und in ihrem Funktionsumfang auf den vom Antriebsreglerhersteller vorgesehen Anwendungsfall beschränkt. Modularität und Offenheit sind nur bei wenigen Geräten und nur eingeschränkt zu finden.

Vollständige Offenheit findet sich vornehmlich im Forschungsumfeld, und hier hauptsächlich auf der Steuerungsebene. Offene Systeme, die für eine Antriebsregelung nutzbar sind, stehen überwiegend für den Laboreinsatz in Form von Projektierungssystemen zur Prototypenentwicklung zur Verfügung.

3 Anforderungen an eine modulare Antriebsreglerplattform

Durch den heutigen Stand der Technik werden einige grundlegende Anforderungen hinsichtlich der notwendigen Schnittstellen, Funktionen und Leistungsfähigkeit des Antriebsreglers gestellt. Der Stand der Forschung und der Ausblick auf zukünftige Entwicklungen stellen jedoch noch viel weitergehende Anforderungen an einen modernen Antriebsregler.

Die künftig gestellten Anforderungen beschränken sich dabei nicht auf die reine **Leistungsfähigkeit** und Integration neuer, offener **Schnittstellen**, sondern betreffen insbesondere auch eine maschinen- und prozessspezifische **Anpassbarkeit** und **Erweiterbarkeit** der Antriebsregelung. Des Weiteren müssen bei neuen Systemen vor dem Hintergrund einer größeren Offenheit verlässliche Methoden für die **Handhabbarkeit** und **Sicherheit** vorgesehen werden. Die Anforderungen werden gegliedert nach diesen wesentlichen Aspekten in den nachfolgenden Abschnitten dargelegt.

3.1 Leistungsfähigkeit

Die grundlegenden Funktionen eines Antriebsreglers umfassen die Berechnung eines kaskadierten Strom- und Drehzahlreglers sowie in einigen Fällen zusätzlich des Lagereglers für eine oder mehrere Achsen. Die einzelnen Regler werden dabei üblicherweise in voneinander abweichenden Takten ausgeführt. Diese liegen gewöhnlich zwischen 31,25 – 125 µs für den Stromregler, bei 62,5 µs – 500 µs für den Drehzahlregler und bei 125 µs – 2 ms für den Lageregler. Um die Taktunterschiede zwischen den Reglern auszugleichen, kann beispielsweise durch Inter- oder Extrapolation die zeitliche Auflösung des Stellsignals eines Reglers für den im Signalfluss nachfolgenden Regler angepasst werden. Neben den Reglern und Interpolationsstufen werden in der Regel an den unterschiedlichsten Stellen digitale Filter eingesetzt. Die Anzahl, Position und Ordnung dieser Filter sind dabei herstellerabhängig. Die üblichsten Filter sind Sollwertfilter an den Sollwerteingängen sowie Istwertfilter an den Istwerteingängen der Regler. Neben den Reglertakten und den notwendigen Rechenoperationen müssen auch die Bitbreiten der auftretenden Größen bei der Abschätzung der Leistungsfähigkeit mit berücksichtigt werden. Dabei spielen die Faktoren Genauigkeit, Auflösung der Soll- und Istwerte, Stabilität der eingesetzten

Algorithmen und das gewählte interne Darstellungsformat der Größen die wesentliche Rolle.

Neben diesen rein regelungstechnischen Aufgaben muss der Antriebsregler über eine ausreichende Rechenkapazität verfügen, um eine Vielzahl von sekundären Aufgaben übernehmen zu können. Zu diesen zählen zum Beispiel die Parameterverwaltung, Diagnose- und Inbetriebnahmefunktionen und verschiedene Überwachungsaufgaben. Darüber hinaus bestehen der Wunsch und die Notwendigkeit, zusätzlich zu diesen Grundfunktionalitäten weitere Funktionalitäten in den Antriebsregler zu integrieren. Dies sind zum einen fortgeschrittene Verfahren zu Signalaufbereitung und -verarbeitung in hohen Takten, Algorithmen zur Signalauswertung für die Prozess- und Maschinendiagnose sowie zusätzliche Regelkreise für beispielsweise eine achsübergreifende Regelung oder prozessspezifische Regelkreise.

Die Leistungsfähigkeit des Antriebsreglers muss daher ausreichend bemessen sein, um nicht nur die Grundfunktionalitäten der Antriebsregelung abdecken zu können, sondern sie muss auch Rechenkapazität und Rechengeschwindigkeit für neue, zusätzliche Funktionalitäten zur Verfügung stellen.

3.2 Offenheit und Schnittstellen

Die voranschreitende Weiterentwicklung der Antriebstechnik, sowie die Anforderung, maschinen- und prozessspezifische Regelungen und Funktionalitäten in den Antrieb zu integrieren, erfordern Offenheit des Antriebsreglers gegenüber Neuerungen, Erweiterungen oder Änderungen.

Die hier getroffene Definition der Offenheit für einen Antriebsregler orientiert sich an der in IEEE festgelegten Beschreibung /52/ von Offenheit: „Ein offenes System stellt Eigenschaften bereit, die es korrekt implementierten Anwendungen erlauben, auf einer Vielzahl von Plattformen unterschiedlicher Hersteller zu laufen, mit anderen Anwendungen zusammenzuarbeiten und ein einheitliches Erscheinungsbild bei der Zusammenarbeit mit Anwendern zu zeigen“. An diese Definition wird auch die Definition von Offenheit numerischer Steuerungen angelehnt /53/.

Genau wie für offene Steuerungen gelten für einen offenen Antriebsregler die Systemeigenschaften Portierbarkeit, Erweiterbarkeit, Austauschbarkeit, Skalierbarkeit und Interoperabilität.

Die Bedeutung dieser Eigenschaften unterscheidet sich jedoch im Detail aufgrund der abweichenden technischen Anforderungen und Randbedingungen von der offenen Steuerung. Die Beschränkung der Offenheit auf die Anwendungen, die in der Regel eine in sich gekapselte Software mit einer festgelegten Funktion sind, ist für die Antriebsregelung nicht ausreichend. Die Nähe zur Antriebs- und Prozessphysik erfordert die Ausweitung der Betrachtung auch auf die Hardware, für die die gleiche Offenheit gelten muss.

- **Portierbarkeit** bedeutet in diesem Zusammenhang, dass sich einzelne Funktionen auf jede Systemplattform übertragen lassen. Dies gilt sowohl für Funktionen, die als Software, als auch in Hardware bereitgestellt werden.
- **Erweiterbarkeit** gewährleistet, dass neue Funktionen in Soft- und Hardware der Systemplattform hinzugefügt werden können.
- **Austauschbarkeit** ermöglicht den Ersatz beliebiger einzelner Funktionen in Soft- und Hardware durch andere Funktionen.
- **Skalierbarkeit** heißt, dass innerhalb des System die gleichen Funktionen mehrfach eingesetzt werden können.
- Die **Interoperabilität** erlaubt die Zusammenarbeit aller eingesetzten Funktionen durch geeignete Schnittstellen in Soft- und Hardware.

Die Offenheit kann dabei in zwei Stufen realisiert werden. In der ersten Stufe kann der Antriebsreglerhersteller eine Offenheit für die eigene Nutzung bereitstellen. Dies ermöglicht ihm eigene Neuentwicklungen, Fehlerkorrekturen und kundenspezifische Varianten mit geringem Aufwand zu realisieren. Er behält dabei die vollständige Kontrolle über das System. In der zweiten Stufe stellt der Antriebsreglerhersteller die Offenheit ganz oder teilweise seinen Kunden zur Verfügung. Diese können dann selbstständig ihre spezifischen Erweiterungen dem System hinzufügen. Die Kunden sind damit unabhängiger vom Antriebsreglerhersteller und können ihre eigenen spezifischen Funktionen integrieren. Schützenswertes Know-how des Kunden muss nicht an den

Antriebsreglerhersteller herausgegeben werden. Jedoch kann eine Funktionsgewährleistung durch den Antriebsreglerhersteller nur schwer gegeben werden.

3.3 Anpassbarkeit und Erweiterbarkeit

Durch die immer weiter gehende Optimierung von Maschinen sowie die Neu- und Weiterentwicklung von Technologien werden die maschinen- und anwendungsspezifischen Anforderungen an den Antriebsregler immer umfangreicher. Das breite Spektrum der benötigten Schnittstellen, Algorithmen und Funktionen kann nur mit hohem Kostenaufwand von einem einzelnen Gerät abgedeckt werden. Sowohl die Software als auch die Hardware von Antriebsreglern ist in der Regel ein komplexes und stark ineinander verflochtenes System. Eine Erweiterung oder Änderung dieses Systems ist daher nur mit erheblichem Aufwand möglich.

In der Softwaretechnik ist das Prinzip der Modularisierung verwendet. Überträgt man dieses auf die Antriebsreglung, kann ein Regelgerät entworfen werden, das sich mit vertretbarem Aufwand an anwendungsspezifische Anforderungen anpassen lässt.

Module in der Softwaretechnik zeichnen sich nach /54/ durch 5 Eigenschaften aus:

1. Darstellung einer funktionalen Einheit oder einer semantischen zusammengehörenden Funktionsgruppe.
2. Weitgehende Kontextunabhängigkeit, d. h., ein Modul ist in sich abgeschlossen. Die Kontextunabhängigkeit beinhaltet, dass ein Modul von der Modulumgebung weitgehend unabhängig entwickelbar, prüfbar, wartbar und verständlich ist.
3. Definierte Schnittstellen für Externbezüge. Die externen Bezüge eines Moduls sollten klar erkennbar und möglichst in einer Schnittstellenbeschreibung zusammengefasst sein.
4. Im qualitativen und quantitativen Umfang handlich, überschaubar und verständlich.

Dies erfordert die Modularisierung aller Funktionalitäten des Antriebsreglers, für die eine Erweiterung, Änderung oder Austausch infrage kommen. Durch die Kapselung zusammenhängender Funktionseinheiten des Antriebsreglers in eigenständige Module

und die Definition von Schnittstellen können die Funktionseinheiten durch alternative Funktionseinheiten ersetzt werden, ohne dass die umgebenden Module dadurch in ihrer Funktionsweise beeinflusst werden. So können softwareseitig beispielsweise verschiedene Regleralgorithmen in eigenständige Module gekapselt werden und in der Anwendung das für die Regelstrecke optimale Modul eingesetzt werden.

Die für die Software getroffene Definition muss für den Antriebsregler auch auf die Hardware übertragen werden, da auch die physikalische Anbindung des Antriebsreglers an seine Umgebung, bestehend aus Steuerung, Sensoren und Aktoren, abgedeckt werden soll. So können beispielsweise verschiedene Feldbus- oder Geberschnittstellen als austauschbare Hardwaremodule realisiert werden.

Die wesentlichen Vorteile einer modulbasierten Software lassen sich auf das vollständige Antriebsreglersystem übertragen:

- **Flexibilität**. Die Modularisierung von Funktionseinheiten führt dazu, dass Funktionseinheiten mit geringem Aufwand gegen alternative Funktionseinheiten ausgetauscht werden können. Der Antriebsregler kann sehr genau an die gegebenen Bedingungen und an die Regelstrecke angepasst werden.
- **Wiederverwendbarkeit**. Die zu einem Modul zusammengefasste Funktionseinheit lässt sich in unterschiedlichen Systemen wieder verwenden. Die Integration in ein anderes System wird durch die klar definierten Schnittstellen vereinfacht.
- **Fehlersicherheit**. Ein Modul kann losgelöst vom Gesamtsystem auf seine Funktionsfähigkeit geprüft werden. Damit ist ein evtl. auftretender Fehler leichter einzugrenzen und zu beheben. Des Weiteren lassen sich Module mit geringerem Aufwand automatisiert in sämtliche Betriebszustände versetzen, um Fehlverhalten aufzudecken.
- **Übersichtlichkeit**. Die Kapselung von vielen Unterfunktionen zu Funktionseinheiten in Modulen führt zu einer übersichtlichen Strukturierung und Darstellung des Gesamtsystems.
- **Verkürzung der Entwicklungszeit**. Durch die Wiederverwendung von Modulen, die entfallenden Überprüfungen sowie Fehlerkorrektur dieser Module und die übersichtliche Anordnung des Gesamtsystems wird die Entwicklung vereinfacht und damit auch die Entwicklungszeit verkürzt.

- **Know-how-Schutz**. Für den Antriebsreglerhersteller sowie für den Anwender ist es möglich, schützenswerte Algorithmen, Verfahren oder Funktionen in Modulen zu kapseln. Für den Aufbau eines Gesamtsystems aus Modulen müssen nur die Schnittstellen der Module bekannt sein.

Die Modularisierung des Antriebsreglers sowohl in Software als auch in Hardware ist daher eine der wesentlichen Anforderungen an ein zukunftsorientiertes Konzept.

3.4 Handhabbarkeit

Eine offene Antriebsreglerplattform bietet zunächst durch die Offenheit und freie Programmierbarkeit sehr viele Freiheitsgrade bezüglich ihrer Programmierung. Darüber hinaus soll der Antriebsregler durch die Offenheit verschiedenen Gruppen von Entwicklern zur Verfügung stehen, die mit unterschiedlichen Zielen und Sichtweisen den Antriebsregler programmieren wollen. Um das System für alle beteiligten Entwickler und Anwender handhabbar zu machen, müssen Entwicklungsrichtlinien, Werkzeuge, Methoden sowie ein Rahmen aus Funktionen und Mechanismen auf der Antriebsreglerplattform zur Verfügung gestellt werden.

Während bei kleineren Antriebsreglersystemen mit geringem Funktionsumfang alle Aufgaben der Programmierung, des Reglerentwurfs, sowie im Ausnahmefall auch der Hardwareentwicklung von einem einzigen Entwickler übernommen werden können, ist bei komplexeren Systemen und Anwendungen die Mitarbeit unterschiedlicher Entwicklergruppen sinnvoll und notwendig. Dazu müssen zunächst die möglichen Entwickler- und Anwendergruppen und ihre Ziele und Sichtweisen definiert werden.

Die einzelnen Gruppen lassen sich, wie folgt charakterisieren:

- **Programmierer des Antriebsreglersystems** stellen die Basisfunktionalitäten des Antriebsreglers bereit. Dazu gehören Mechanismen, die für die korrekte Ausführung aller Module sorgen, sowie für eine Infrastruktur für den Transport von Signalen und Parametern. Diese Entwicklergruppe braucht den uneingeschränkten Zugriff auf das gesamte System.
- **Programmierer der Antriebsfunktionen** entwickeln die Funktionen, die für einen vollständigen Antriebsregler benötigt werden. Dies umfasst z. B.

verschiedene Reglertypen, die Signalvorverarbeitung, die Ansteuerung der Leistungselektronik der Antriebe oder die Feldbusanbindung. Diese Entwicklergruppe braucht eine funktionierende Infrastruktur sowie festgelegte Schnittstellen, über die sie auf andere Komponenten des Systems zugreifen kann.

- **Regelungs- und Prozesstechniker** entwickeln Regleralgorithmen, die mithilfe vorhandener Funktionsblöcke umgesetzt werden können, oder für die eigene Funktionsblöcke implementiert werden müssen. Sie kombinieren die Funktionsblöcke zu einem vollständigen, lauffähigen Antriebsreglersystem. Die Regelungs- und Prozesstechniker lassen sich in weitere Gruppen unterteilen. Zum einen werden seitens des Antriebsreglerherstellers Entwickler benötigt, die ein für den Kunden nutzbares Grundsystem, eventuell bereits mit kundenspezifisch angepassten Funktionsblöcken, erstellen. Zum anderen kann es auf der Seite des Kunden, also dem Maschinenbauer, Entwickler geben, die den Antriebsregler selbst an die eigenen Anforderungen anpassen. Diese Entwicklergruppe benötigt fertige Bibliotheken von Funktionsblöcken, aus denen sie Regelsysteme zusammensetzen können. Für besondere Anwendungen benötigen sie darüber hinaus den direkten Zugriff auf die Infrastruktur des Systems, um eigene Funktionsblöcke implementieren zu können.
- Die **Hardwareentwickler des Basissystems** implementieren die definierte Hardwarearchitektur der Antriebsreglerplattform. Sie wählen die Bauteile aus, die den Ressourcenbedarf des geplanten Gesamtsystems abdecken und stellen Hardwareschnittstellen für die Hardwareerweiterungen bereit. Für die Hardwareentwicklung müssen sie den geplanten Ressourcenbedarf des Antriebsreglersystems, sowie die Schnittstellenbeschreibung kennen.
- **Hardwareentwickler von Erweiterungen** liefern die Hardwaremodule mit den Schnittstellen zur Umgebung, wie z. B. Gebersysteme und Feldbusanbindung. Darüber hinaus programmieren sie ggfs. hardwarenahe Treiberfunktionen für die Schnittstelle zwischen Hard- und Software der Antriebsreglerplattform. Sie benötigen hardwareseitig die Beschreibung der Schnittstellen, die sie für die Hardwaremodule nutzen können, sowie für die Bereitstellung von Treiberfunktionen Zugriff auf die Infrastruktur der Antriebsreglerplattform.
- Der **Elektrokonstrukteur** des Maschinenherstellers legt die Anforderungen an den Antrieb und den Antriebsregler fest. Er plant den Antrieb, die Sensoren und die Feldbusanbindung an die Maschinensteuerung.

- **Inbetriebnehmer** parametrieren den in einer Maschine eingebauten Antriebsregler. Sie benötigen nur den eingeschränkten Zugriff auf die für die Inbetriebnahme relevanten Parameter.

Bild 3-1 zeigt eine Übersicht der Entwickler- und Anwendergruppen und ihre Aufgaben in Zusammenhang mit dem Antriebsregler.

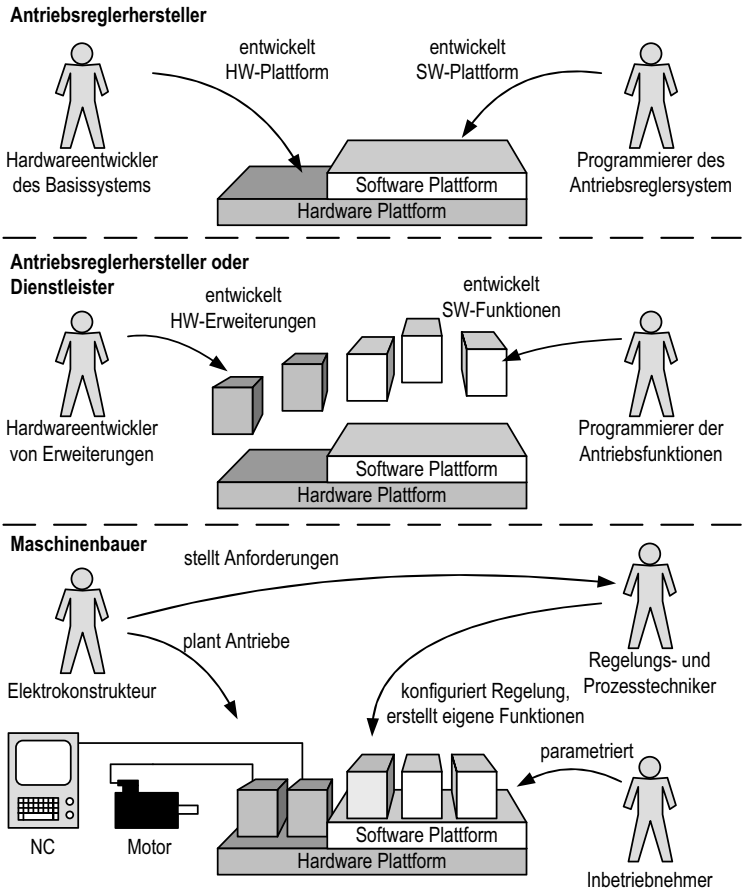

Bild 3-1: Aufgaben der Entwickler und Anwender einer Antriebsreglerplattform.

3.5 Sicherheit

Sicherheit in technischen Prozessen ist ein weites Feld, das von der Personensicherheit des Maschinenbedieners bis zur sicheren Übertragung eines einzelnen Signals innerhalb eines Steuergerätes reicht. Die Sicherheitsanforderungen, die an den Antriebsregler gestellt werden, lassen sich zu den folgenden Punkten zusammenfassen:

- **Schutz vor Fehlern bei der Programmierung und Parametrierung:** Die grundlegendsten Sicherheitsanforderungen an den Antriebsregler werden schon während der Entwicklung und der Inbetriebnahme gestellt. In einem offenen, frei programmierbaren System ist es für den Benutzer sehr aufwendig und oftmals auch unmöglich zu durchschauen, ob die von ihm durchgeführten Änderungen oder Ergänzungen die Funktionsweise des Gesamtsystems beeinträchtigt.
- **Betriebs- und Ausfallsicherheit:** Es kann während des Betriebs zu Ausfällen von Hardwarekomponenten und Fehlern in der Software kommen. Bei den Hardwarekomponenten liegt der Fehler meist in der falschen Dimensionierung oder der Ausfall wird durch Verschleiß oder durch Alterung verursacht. Fehler in der Software entstehen in der Entwicklung. Sie treten nur auf, wenn der fehlerhafte Teil des Programms durchlaufen wird. Aufgrund der Komplexität der Software gibt es meist eine unüberschaubare Anzahl von unterschiedlichen Systemzuständen, sodass es schwer ist, die Fehlerfreiheit zu gewährleisten. Eine Betriebs- und Ausfallsicherheit lässt sich, unabhängig davon, welche Komponenten betroffen sind, oder durch was ein Ausfall verursacht wurde, im Betrieb nur durch ständige Überwachung der Komponenten, z. B. durch Plausibilisierung ihrer Ausgangssignale erreichen.
- **Überwachung von Maschinenkomponenten und des Prozesses:** Die Überwachung einzelner Maschinenkomponenten, der gesamten Maschine sowie des Prozesses geschieht im Rahmen der Maschinen- oder Prozessdiagnose /55,56/. Diese basiert klassischerweise auf externer Sensorik, deren Signale herangezogen werden, um Aussagen über den Zustand der Maschine oder der Werkzeuge zu treffen. Um dem erhöhten Verdrahtungsaufwand und zusätzlichen Kosten externer Sensorik entgegenzuwirken, werden für die Diagnose zunehmend mehr

antriebsinterne Signale genutzt, und zum Teil sogar vollständig auf zusätzliche Sensorik verzichtet /57,58/. Die Algorithmen der Diagnose werden teilweise direkt im Antriebsregler gerechnet, und der übergeordneten Steuerung aufbereitete und ausgewertete Daten zur Verfügung gestellt /59/. Dazu muss im Antriebsregler die Möglichkeit bestehen, die maschinen- bzw. prozessspezifischen Algorithmen auszuführen.

- **Sichere Fehlerreaktion:** Neben der Vermeidung sowie der sicheren Detektion von Fehlern muss schließlich auch sicher auf die Fehler reagiert werden. Die Reaktionen sind dabei abhängig von der Art und der Schwere eines Fehlers. Durch die Norm EN-60204-1 /60/ werden sie gemäß Tabelle 3-1 für drei Kategorien definiert.

Wie weit die einzelnen Sicherheitsanforderungen erfüllt werden müssen, hängt stark von dem jeweiligen Einsatzgebiet des Antriebsreglers ab. Eine neue Antriebsreglerarchitektur muss grundsätzlich die Möglichkeit bieten, jede Sicherheitsanforderung umzusetzen.

Kat.	Kurzbeschreibung	Erklärung
0	Not Aus: Sofortige Trennung der Energiezufuhr des Antriebs.	Mit dieser Kategorie ist die sichere Abschaltung des Antriebsmoments gewährleistet. Der Antrieb trudelt frei aus. Vorhandene Bremsen fallen ein. Dieser Stopp eignet sich als Fehlerreaktion auf schwere Fehler, die verhindern, dass der Antrieb geregelt oder gesteuert werden kann, etwa bei dem Ausfall des Kommutierungsgebers, oder einem IGBT-Ausfall in der Endstufe.
1	Not Halt: Geregeltes oder gesteuertes Abbremsen	Die Energie zum Antrieb bleibt zunächst bestehen. Der Antrieb wird geregelt oder gesteuert abgebremst. Anschließend wird der Antrieb stromlos und damit momentenfrei geschaltet. Dieser Stopp eignet sich für Fehlerfälle, in denen der Antrieb weiterhin regelbar ist. Eine geregelte bzw. gesteuerte Bremsung ist einer Energieabschaltung in diesem Falle vorzuziehen, da die Bremsung unter Einhaltung aller Randbedingungen der Maschine, wie z. B. maximal zulässiger Ruck oder Beschleunigung, weniger Gefahr für Mensch und Maschine birgt als das unkontrollierte Austrudeln des Antriebs.
2	Anhalten in Regelung	Die Energie zum Antrieb bleibt bestehen. Der Antrieb wird geregelt oder gesteuert abgebremst. Der Antrieb bringt auch im Stillstand noch ein Moment auf. Dieser Stopp wird insbesondere dann benötigt, wenn es sich um Achsen handelt, die für den Stillstand ein Antriebsmoment benötigen, wie z. B. bei hängenden Achsen ohne Selbsthemmung. Über diese grundlegenden Sicherheitsfunktionen hinaus sind noch zahlreiche anwendungsspezifische Fehlerreaktionen möglich, die beispielsweise bei mehrachsigen Maschinen eine Bremsung auf der Bahn oder bei Bohrprozessen einen Rückzug des Werkzeugs vorsehen.

Tabelle 3-1: Fehlerreaktionen nach Kategorien

4 Entwurf einer Antriebsreglerplattform

Heutige Antriebsregler sind für die Ausführung auf sequenziell arbeitenden Mikrocontrollern bzw. DSP-basierten Systemen entwickelt. Dabei werden meist die von den Echtzeitbetriebssystemen zur Verfügung gestellten Mechanismen genutzt. Die Funktionen werden gruppiert nach Ausführungstakten und Prioritäten in eigenständige Tasks gekapselt, die koordiniert von einem Scheduler ausgeführt werden. Spezielle, für Echtzeitbetriebssysteme typische Kommunikationsmechanismen, wie z. B. Message-queues, Semaphore und Shared Memory, stellen den Signal- und Datenfluss zwischen den Tasks bereit und liefern Mechanismen, mit denen Systemverklemmungen verhindert werden können.

Aufgrund der unterschiedlichen Systemeigenschaften von Mikrocontrollern gegenüber FPGAs lassen sich vorhandene Antriebsregler nicht ohne Weiteres übertragen. Mikrocontroller arbeiten **sequenziell** einzelne Rechenschritte ab, wohingegen auf einem FPGA die Rechenschritte als Netzwerk frei programmierbarer Logikgatter implementiert sind und **parallel** ausgeführt werden können. Es ist daher ein umfassendes Konzept notwendig, welches eine Migration von bestehenden DSP-basierten Reglersystemen auf programmierbare Logik ermöglicht und gleichzeitig das volle Potenzial dieser Technik nutzbar macht um eine Leistungssteigerung und die geforderte Offenheit und Flexibilität zu erreichen.

Es erscheint zunächst naheliegend die Migration der, meist in C-Code implementierten, anwendungs- bzw. prozessspezifischen Algorithmen und Verfahren mit Hilfe von DSPs und Mikrocontrollern, die als logische Schaltung auf dem FPGA konfiguriert werden, vorzunehmen. Diese können in Hochsprachen programmiert werden und sind für die direkte Übertragung bestehender Software auf den FPGA geeignet. Jedoch sind dabei Einschränkungen zu beachten, die sich durch eine relativ geringe Taktrate und einen hohen Bedarf von logischen Zellen auf dem FPGA ergeben. Die Funktionen werden auf dem Mikrocontroller rein sequenziell ausgeführt. Der Vorteil des FPGAs, Funktionen parallel auszuführen zu können, kann von einer Mikrocontrollersoftware nicht genutzt werden. Soll das durch den Einsatz von FPGAs vorhandene Leistungspotenzial genutzt werden, ist es notwendig, möglichst viele Funktionen in logische Schaltungen zu übertragen.

Für die Entwicklung eines Konzepts zur Übertragung der Antriebsreglerfunktionen ist es zunächst erforderlich, ihre einzelnen Aufgaben und Anforderungen an das Antriebsreglersystem zu untersuchen. Dazu werden die Aufgaben und Anforderungen der heutigen Technik und der absehbaren zukünftigen Entwicklungen betrachtet. Darüber hinaus werden die derzeit zur Verfügung stehenden technischen Möglichkeiten berücksichtigt. Darauf aufbauend kann eine offenen Antriebsreglerplattform entworfen werden, die den Anforderungen entspricht und die verfügbare Technik optimal ausnutzt. Es werden Funktionen, Abläufe, Signale und Mechanismen entworfen, die benötigt werden, um alle erarbeiteten Anforderungen zu erfüllen.

Das Konzept umfasst dabei den prinzipiellen Aufbau eines Reglersystems und die dazu benötigte Infrastruktur, bestehend aus Schnittstellen, Ausführungs- und Überwachungsmechanismen sowie die für die sichere und effiziente Entwicklung benötigte Werkzeuge.

4.1 Leistungsanforderungen

Im Kapitel 3.1 wurden die Anforderungen bezüglich der Leistungsfähigkeit aufgeführt, die an ein flexibles und offenes Antriebsreglersystem gestellt werden. Die Leistungsfähigkeit umfasst die regelungstechnischen Anforderungen, die letztendlich die erreichbare Reglerdynamik ausmachen, sowie die funktionellen Anforderungen bezüglich Funktionsumfang, Sicherheit und Bedienbarkeit.

Für parallel ausführbare Rechenoperationen ist die Leistungsfähigkeit hinsichtlich der Rechengeschwindigkeit nur durch die Anzahl der zur Verfügung stehenden programmierbaren Logikelementen beschränkt. Theoretisch ist das Ergebnis bereits nach der Signallaufzeit eines einzelnen Gatters verfügbar.

Eine bedeutendere Einschränkung der Leistungsfähigkeit ist durch den Signal- und Datenfluss gegeben. Nicht alle Signale stehen jederzeit zur Verfügung und können parallel bearbeitet werden. Insbesondere bei kaskadierten Regelkreisen gibt es Signalabhängigkeiten, die eine sequenzielle Ausführung erfordern. Hinzu kommen die weiteren peripheren Funktionalitäten des Antriebsreglers, wie z. B. Überwachungen, Feldbusschnittstellen und Benutzerschnittstellen, die ebenfalls in einem zeitlich beschränkten Rahmen ausgeführt werden können.

Im Folgenden werden die Funktionen des klassischen Antriebsreglers betrachtet, die maßgeblich für die zeitlichen Abläufe und damit auch für die Leistungsfähigkeit sind. Sie müssen in dieser Form auch auf der offenen Antriebsreglerplattform zur Verfügung gestellt werden.

Die Funktionen lassen sich in eine Gruppe mit hohen zeitlichen Anforderungen, als Echtzeit bezeichnet, und in eine Gruppe mit niedrigen zeitlichen Anforderungen, als Nicht-Echtzeit bezeichnet, unterteilen.

Die Funktionen mit Echtzeitanforderung sind:

- Regelkreise (Strom-, Drehzahl- und Lageregler)
- Messdatenerfassung (Bedienung von Hardwareschnittstellen, Datenvorverarbeitung, wie z. B. Filterung, Berechnung der Geschwindigkeit)
- Stellwertausgabe
- Überwachung (Fehlerüberwachung, Fehlerreaktion, Diagnosefunktionen)
- Echtzeitkommunikation (z. B. Feldbusanbindung an eine Steuerung)

Nicht-Echtzeit Funktionen sind:

- Betriebsartensteuerung (Reaktion auf Freigaben, Umschaltung zwischen Betriebsarten)
- Parameterverwaltung
- Benutzerschnittstelle (für Parametrierung, Inbetriebnahme, Abfrage von Fehlerzuständen und Diagnoseergebnisse)

Es zeigt sich, dass in der Antriebsregelung eine Reihe von Funktionen benötigt werden, die in sehr unterschiedlichen Takten, mit sehr unterschiedlichen zeitlichen Anforderungen laufen können. Für eine Signalerfassung für die schnelle Stromreglung werden Ausführungstakte im Bereich mehrerer Megaherz benötigt, wohingegen die einzelnen Reglerkaskaden in Takten von wenigen Mikrosekunden bis zu Millisekunden ausgeführt werden können. Diese Funktionen werden regelmäßig mit hohen zeitlichen Anforderungen bezüglich Takt und Jitter ausgeführt. Es gibt jedoch auch Funktionen, die nur gelegentlich ausgeführt werden müssen und nur schwache zeitliche

Anforderungen haben. Dies trifft insbesondere auf die peripheren Funktionen, wie z. B. die Benutzerschnittstelle zur Parametrierung des Antriebsreglers zu. Diese wird nur zur Inbetriebnahme des Antriebsreglers benötigt, die zeitlichen Anforderungen werden hier durch die Bedienbarkeit gesetzt. Der Benutzer soll das Gefühl haben, dass sich die Benutzerschnittstelle flüssig und für sein Empfinden verzögerungsfrei bedienen lässt. Dazu sind Reaktionszeiten bis zu mehreren Millisekunden zulässig.

Die Ausführung der Funktionen in einem hohen und präzisen Takt ist jedoch nicht die einzige Anforderung, die gestellt wird. Die Funktionen haben in der Regel die Aufgabe, die an ihren Eingängen anliegenden Daten und Signale weiter zu verarbeiten. Dies setzt immer voraus, dass an ihren Eingängen aktuelle und gültige Daten vorhanden sind. Die Funktionen müssen daher so zueinander versetzt ausgeführt werden, dass der korrekte Signalfluss gewährleistet wird. Die Ausführungszeitpunkte der einzelnen Funktionen müssen so optimiert werden, dass die Signallaufzeit vom Eingang zum Ausgang des Gesamtsystems minimal ist. Bei einem Antriebsregler entsprechen die Eingänge den Istgrößen Phasenströme und Rotorlage sowie dem Sollwert von der Steuerung. Die Ausgänge liefern die Steuersignale für eine Leistungsendstufe.

Die Signallaufzeit des Gesamtsystems ist gleichbedeutend einer Totzeit in einem zeitkontinuierlichen Regler. Die Totzeit hat einen direkten Einfluss auf die erreichbare Reglerdynamik, da eine Totzeit eine negative Phasendrehung mit sich bringt, wodurch die Reglerbandbreite geschmälert wird /61/.

Zusammenfassend kann festgestellt werden, dass es nicht ausreicht, die Funktionen in hoher Geschwindigkeit parallel abzuarbeiten, um die maximale Leistungsfähigkeit des Systems nutzen zu können. Zusätzlich wird ein Mechanismus benötigt, der die zeitlichen Abläufe der einzelnen Funktionen anforderungsgerecht steuert und dabei den Daten- und Signalfluss berücksichtigt.

Der dazu verwendete Mechanismus besteht aus einer planerischen und einer ausführenden Komponente. Die planerische Komponente ist ein Entwicklungswerkzeug, das die Signal- und Datenabhängigkeiten zwischen den Funktionen, ihre Ausführungszeiten, sowie ihre Taktraten kennt. Mit diesem Werkzeug kann im Voraus für das Gesamtsystem ein Ausführungszeitplan erstellt werden, der die Funktionsaufrufe und deren zeitlichen Versatz beinhaltet. Die Erstellung erfolgt unter der Bedingung, die optimale Signallaufzeit zwischen auswählbaren Systemein- und -ausgängen zu

erreichen. Die ausführende Komponente wird durch einen Scheduler gebildet. Der Scheduler ist Teil der Antriebsreglerplattform. Er arbeitet den vorab erzeugten Ausführungszeitplan ab und aktiviert die einzelnen Funktionen des Antriebsreglers.

4.2 Modulare Architektur

Auch um die in Kapitel 3.3 geforderte Modularität zu erfüllen, kommt die Eigenschaft des FPGAs zum Tragen, Funktionen parallel und unabhängig als eigene Schaltkreise zu realisieren. Der FPGA bietet die geeignete Grundlage für ein Antriebsreglersystem, dessen Funktionalitäten in einzelnen Modulen aufgliedert sind, die völlig unabhängig voneinander ausgeführt werden können.

In DSP-basierten Systemen ist es gängig, die Funktionalitäten gemäß ihrer zeitlichen Anforderungen zu gliedern und zu gruppieren, sodass diese bei sequenzieller Ausführung die Anforderungen erfüllen können. So gibt es beispielsweise einen Task, der im Feldbustakt ausgeführt wird und sämtliche Funktionen enthält, die in diesem Takt ausgeführt werden müssen. In einem weiteren Task, der im Drehzahlregeltakt ausgeführt wird, sind sämtliche Funktionen enthalten, die in diesem höheren Takt ausgeführt werden müssen. Diese zeitorientierte Gliederung von Funktionen entfällt auf dem FPGA. Die Funktionen können hier rein funktionsorientiert gegliedert und zu Modulen zusammengefasst werden, wodurch der ursprünglichen Idee der Modularisierung vollständig entsprochen werden kann.

4.2.1 Modularisierung der Software

Als Software wird im Allgemeinen ein ausführbares Programm, das von einem digitalen Prozessor ausgeführt wird, verstanden. Im Zusammenhang mit der Offenen Antriebsreglerplattform soll der Begriff Software erweitert werden. Er umfasst nicht nur ausführbare Programme, sondern auch die auf dem FPGA programmierten Logikschaltungen, die ähnlich wie Programme eigene Funktionen erfüllen.

Um einen Antriebsregler auf einem FPGA handhabbar und übersichtlich zu gestalten, werden die einzelnen Funktionen der Antriebsregelung in Module getrennt, die die jeweils benötigte Logikschaltung beinhalten. Diese Modularisierung der

Antriebsreglersoftware dient vor allem der Übersichtlichkeit und damit der Handhabbarkeit sowie Erweiterbarkeit des Antriebsreglersystems.

Die Module verfügen über einheitliche Schnittstellen, sodass der Antriebsregler nach Bedarf und Anwendungsfall aus unterschiedlichen Modulen flexibel zusammengesetzt werden kann. Einmal entwickelte Module können für jeden Anwendungsfall wieder verwendet werden, wodurch sich die Entwicklungszeit für neue Antriebsregler stark verkürzt, da im einfachsten Fall ein bestehender Antriebsregler nur durch Austausch oder Hinzufügen neuer Module angepasst werden kann.

Die Modularisierung bietet sowohl dem Antriebsreglerhersteller als auch dem Anwender die Möglichkeit, sein Know-how zu schützen. Ähnlich wie es auch bei Software für Prozessoren möglich ist, Software und die darin enthaltenen Funktionen und Algorithmen nur fertig kompiliert und nicht mehr im Klartext lesbar als Maschinensprache herauszugeben, können die Module äquivalent zur Maschinensprache als fertig synthetisierte Netzliste weitergegeben werden.

4.2.2 Modularisierung der Hardware

Durch die enge Kopplung des FPGAs an die Hardware und die freie Programmierbarkeit der physikalischen Ein- und Ausgänge dieses Bausteins lässt sich die Modularität und Flexibilität sehr einfach auch auf die umgebende Hardware ausweiten. Hardwarekomponenten können, soweit sie über digitale Ein- und Ausgänge verfügen, direkt an den FPGA angeschlossen und von einer entsprechenden Logikschaltung auf dem FPGA genutzt werden.

Die Modularisierung erfolgt durch die Gruppierung zusammengehöriger Schaltungen auf einzelnen Platinen, die über einen standardisierten Stecker mit der Hardware-plattform verbunden werden. So können beispielsweise die Schnittstellen zu unterschiedlichen Feldbus- oder Gebersystemen auf eigenständigen Platinen implementiert werden, die dann anwendungsabhängig in der gewünschten Zusammenstellung der Hardwareplattform hinzugefügt werden.

4.3 Offenheit auf drei Ebenen

Die FPGA-basierte, offene Antriebsreglerplattform gliedert sich, wie in Bild 4-1 dargestellt, in drei Ebenen. Die unterste Ebene bildet die Hardware, die Schnittstellen zu Hardwareerweiterungen bietet. Ein Teil der Hardware ist der FPGA, der die Basis für die zweite Ebene bereitstellt. In der zweiten Ebene können Funktionen und Algorithmen in Form von logischen Schaltnetzwerken konfiguriert werden. Durch die Konfiguration von Mikrocontroller- oder DSP-Kernen in der zweiten Ebene entsteht eine dritte Ebene, in der wiederum Software ausgeführt werden kann.

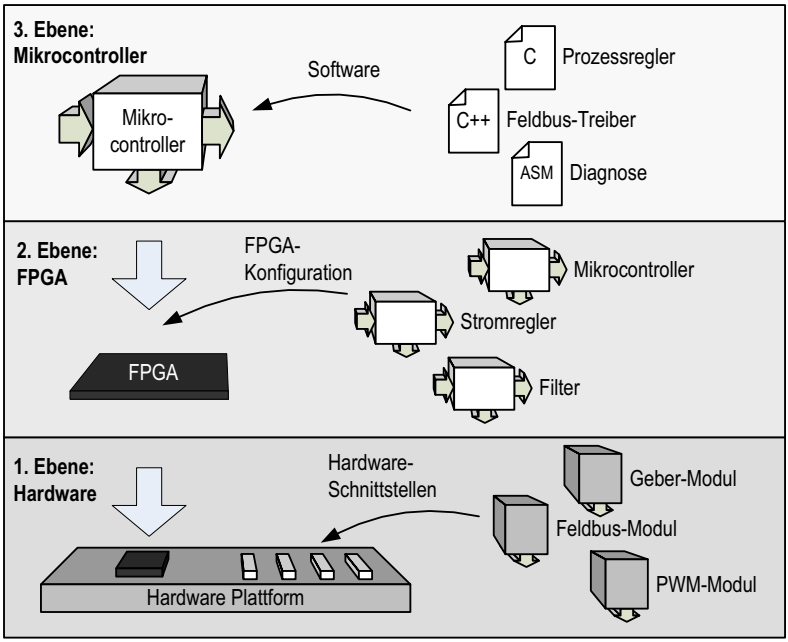

Bild 4-1: Offenheit auf drei Ebenen.

4.3.1 Offenheit auf der Hardware-Ebene

Erweiterungen und neue Entwicklungen finden häufig auch auf der Hardware-Ebene statt. Zum Beispiel die Physik eines neuen Feldbussystems oder eine neuartige Sensorik erfordern elektronische Komponenten, die sich auf dem FPGA nicht programmieren lassen, da analoge Schaltungsteile benötigt werden. Für Hardwareerweiterungen müssen daher physikalische Schnittstellen vorgesehen werden, die eine enge und schnelle Anbindung neuer Hardwarekomponenten an den FPGA ermöglichen.

4.3.2 Offenheit auf der FPGA-Ebene

Der FPGA bildet funktionstechnisch den Mittelpunkt der Antriebsreglerplattform. Alle Hardwarekomponenten sind mit diesem Baustein verbunden und das gesamte Reglersystem ist in ihm enthalten. Alle Erweiterungen und Änderungen am Antriebsregler sind immer mit einem Eingriff in programmierte Logik auf dem FPGA verbunden. Daraus entsteht die Anforderung, auf dem FPGA eine Infrastruktur aus Schnittstellen und Mechanismen zu schaffen, die es zulässt, einem bestehenden System neue Komponenten hinzuzufügen und ihnen Zugriff auf alle Signale innerhalb des Antriebsreglers zu gewähren. Dadurch wird ein System geschaffen, das die Integration zukünftiger Entwicklungen ermöglicht.

4.3.3 Offenheit auf Mikrocontroller-Ebene

Es lassen sich nicht alle Funktionen und Algorithmen effizient in logischen Schaltungen ablegen. Vor allem Funktionen, die komplexere Abläufe beschreiben oder umfangreiche, mehrstufige Berechnungen lassen sich ressourcensparender sequenziell auf einem Signalprozessor oder Mikrocontroller abarbeiten. Der FPGA bietet die Möglichkeit, vollständige Mikrocontroller in ihrer programmierbaren Logik abzubilden. Moderne Varianten von FPGAs haben bereits fest eingebaute DSP-Kerne auf ihrer Chipfläche integriert. Werden in einem Antriebsreglersystem diese Möglichkeiten genutzt, sollte auch auf dieser Ebene Offenheit gegeben sein. Der Entwickler muss die Möglichkeit haben, eigenen Code auf den Mikrocontrollern auszuführen, und er sollte Zugriff auf alle antriebsreglerinternen Signale haben.

4.4 Module der Hardware-Ebene

Die Module der Hardware-Ebene, im Weiteren als Hardwaresubmodule, stellen einerseits Schnittstellen zur Aktorik und Sensorik sowie zur übergeordneten Steuerung bereit, können aber andererseits auch als Ressourcen- oder Funktionserweiterungen der Antriebsreglerplattform dienen.

Ihre wesentliche Aufgabe besteht darin, den Antriebsregler offen gegenüber allen auf dem Markt befindlichen und zukünftigen Schnittstellen zu halten. Hier ist ebenfalls eine Modularisierung sinnvoll, die es dem Anwender ermöglicht, sich eine Antriebsreglerhardware aus den Submodulen zu konfigurieren, die exakt die von ihm benötigten Schnittstellen und Hardwareerweiterungen bereitstellt.

Bei den Wegmesssystemen gibt es abhängig von der geforderten Geschwindigkeit, Genauigkeit, Störanfälligkeit und Kosten eine Vielzahl von Schnittstellen und Übertragungsmedien, beginnend bei einfachen TTL-Signalen bis hin zu Ethernet-basierenden, digitalen Sensor-Bussen. Auch ist die Anzahl der benötigten Wegmesssysteme von der Anwendung abhängig. Oft reicht ein einfacher Motorgeber, jedoch bei hochgenauen Werkzeugmaschinen wird ein zweiter, direkter Geber an der Vorschubachse benötigt.

Bezüglich der Anbindung an eine übergeordnete Steuerung ist eine ähnliche Flexibilität notwendig, da auch hier eine Vielzahl von Antriebs- und Feldbussen eingesetzt wird.

Für die antriebsnahe Prozessregelung können spezielle, prozessspezifische Sensoren, die nicht über eine standardisierte Wegmesssystem-Schnittstelle verfügen, über Hardwaresubmodule eingelesen werden. Damit lassen sich beispielsweise Temperaturfühler, Abstandssensoren, Beschleunigungssensoren oder Kamerasysteme in die Antriebsregelung integrieren.

Eine weitere Option ist es, die Hardwaresubmodule für eine Ressourcenerweiterung der Plattform zu nutzen. Es können beispielsweise weitere FPGA-Bausteine, Speicher oder Mikrocontroller dem System hinzugefügt werden, um Teile der FPGA-Software bzw. Funktionen auf diese auszulagern.

4.4.1 Hardware-Schnittstelle

Um der Offenheit der Hardware-Schnittstelle, und der damit verbundenen Vielzahl unterschiedlicher Anforderungen der Schnittstelle zwischen Submodul und Antriebsreglerplattform gerecht zu werden, muss die Schnittstelle ebenfalls flexibel und offen gestaltet werden.

Das Spektrum der elektrischen Schnittstellenanforderung reicht von sehr schneller Übertragung geringer Datenmengen, bis zur langsamen Übertragung großer Datenmengen oder Kombinationen aus beiden, bei nicht festgelegter Übertragungsrichtung.

Es ist für dieses Anforderungsspektrum nicht ohne Weiteres möglich, die ideale Schnittstelle zu definieren. Die elektrische Schnittstelle wird daher auf einige Gemeinsamkeiten aller Anwendungen reduziert:

- **Spannungsversorgung:** Die elektrische Schnittstelle stellt den Submodulen die gebräuchlichsten Versorgungsspannungen (z. B. 3,3 V, 5 V, 15 V) zur Verfügung.
- **Datenleitungen:** Es werden ausreichend viele, parallel und bidirektional nutzbare Datenleitungen bereitgestellt.
- **Konfigurationsleitungen:** Es wird eine minimale Anzahl von Konfigurationsleitungen bereitgestellt, die von allen Submodulen gleichermaßen bedient werden müssen.

Ein zu lösendes Problem dieser Schnittstelle liegt in der Bidirektionalität der Datenleitungen. Es muss verhindert werden, dass eine plattformseitig als Ausgang beschaltete Datenleitung mit einer submodulseitig ebenfalls als Ausgang beschalteten Datenleitung verbunden wird, da dies ggf. zu einem unzulässigen Stromfluss führen kann. Jede Schnittstelle muss seitens der Plattform immer passend zu dem eingesteckten Submodul beschaltet werden. Mithilfe der Konfigurationsleitungen, die einheitlich von allen Submodulen bedient werden, kann dies erreicht werden. Über diese Leitungen kann das eingesteckte Submodul von der Antriebsreglerplattform identifiziert und anschließend die Datenleitungen der Schnittstelle in passender Richtung aktiviert werden.

Neben der elektrischen Schnittstelle muss auch die mechanische Schnittstelle festgelegt werden. Diese besteht zum einen aus der mechanischen Ausführung des Steckverbinders zwischen Antriebsreglerplattform und Submodul und zum anderen aus den mechanischen Abmessungen der Submodul-Platine sowie einer Gehäuseplatte (z. B. Frontplatte oder Slot-Blech), auf der sich Bedienelemente und Stecker für Feldbus, Sensorik und Aktorik befinden.

Um der Modularität gerecht zu werden, werden alle Submodule so konstruiert, dass jedes Submodul in jeden Steckplatz der Antriebsreglerplattform eingesteckt werden kann. Bild 4-2 zeigt eine mögliche Realisierung austauschbarer Hardware-Submodule einer Antriebsreglerplattform. Die Schnittstelle zur Plattform, sowie die Frontplatte aller Submodule sind gleich, so dass sie beliebig eingesteckt werden können.

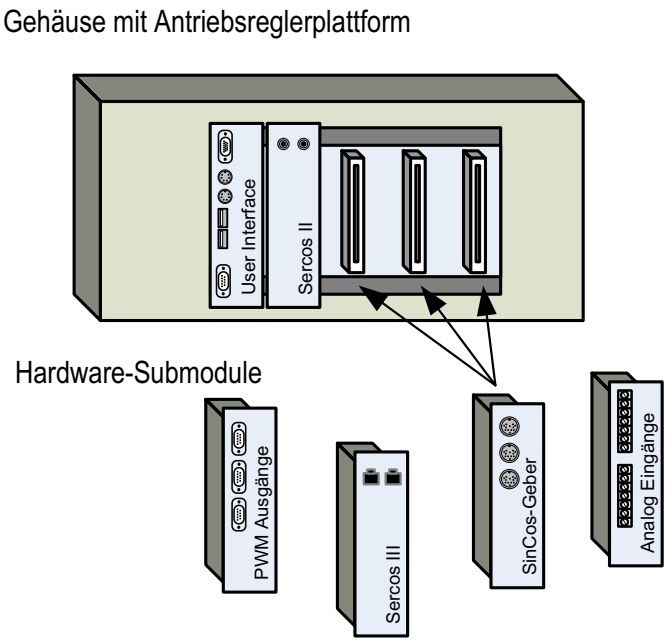

Bild 4-2: Die einheitliche Steckverbindung zur Plattform, sowie die einheitlichen Frontplatten ermöglichen beliebige Konfigurationen von Submodulen.

4.5 Module der FPGA-Ebene

Wie in Kapitel 4.1 bereits zur Ermittlung der Leistungsanforderungen beschrieben, wird der Antriebsregler in einzelne Funktionen aufgetrennt, welche nach Aufgaben gegliedert sind und weitestgehend unabhängig voneinander ausgeführt werden können. Diese Funktionen werden in Module zusammengefasst, die über eine einheitliche Schnittstelle verfügen und beliebig zu einem Antriebsreglersystem kombiniert werden können.

Eine Bibliothek dieser Module bietet dem Anwender beispielsweise unterschiedliche Regleralgorithmen, Prozessreglungen, verschiedene Filter und Begrenzer oder Diagnosefunktionen. Der Anwender kann aus der Bibliothek die benötigten Module auswählen und daraus seinen anwendungsspezifischen Antriebsregler konfigurieren.

Damit sich die Module dieser Bibliothek einfach und flexibel zu einem vollständigen Reglersystem konfigurieren lassen, muss ein einheitlicher Aufbau sowie klare Schnittstellen zur Antriebsreglerplattform, als auch zu anderen Modulen definiert werden. Die Antriebsreglerplattform muss die Ausführung der Module steuern und überwachen können, um den fehlerfreien Ablauf gewährleisten zu können.

4.5.1 Definition von Funktionsblöcken

Die Module der Antriebsreglerplattform werden als Funktionsblöcke bezeichnet. Jeder Funktionsblock besteht aus einer Funktionsblockbeschreibung und einer eigenständige Einheit logischer Verknüpfungen, die eine in sich abgeschlossene Funktion erfüllen.

Die Schnittstelle, über die der Funktionsblock in die Antriebsreglerplattform integriert wird, besteht aus:

- **Systemtaktversorgungseingang (Clock):** Die logischen Schaltungen innerhalb des Funktionsblocks benötigen einen hochfrequenten Systemtakt, um stabil und synchronisiert arbeiten zu können.
- **Ausführungssignaleingang (Trigger):** Eine positive Flanke auf diesem Eingang signalisiert dem Funktionsblock, dass er nun seine Eingänge übernehmen und seine Funktion ausführen darf.

- **Reseteingang:** Eine positive Flanke auf diesem Eingang setzt den Funktionsblock in einen definierten Anfangszustand zurück.
- **Überwachungsausgang:** Der Funktionsblock sendet ein Signal, dass er seine Ausführung beendet hat.

Neben diesen Steuerungsein- und -ausgängen kann der Funktionsblock über Ein- und Ausgänge für die Anbindung an einen plattformweiten Datenbus verfügen. Über diesen können nichtechtzeitkritische Daten, wie z. B. Parameter ausgetauscht werden.

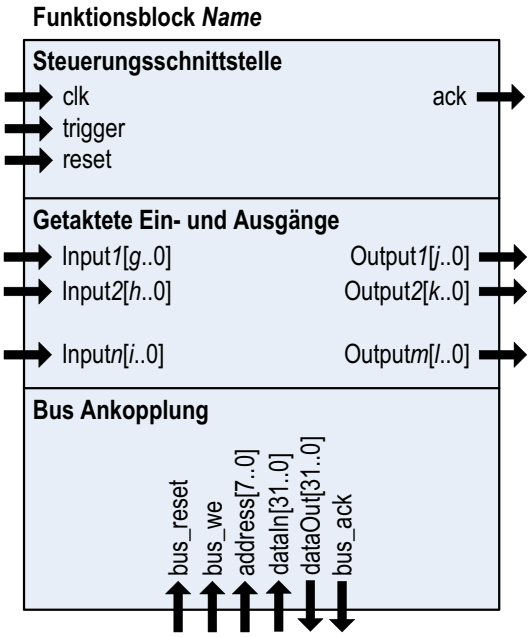

Bild 4-3: Schnittstellen der Funktionsblöcke.

In Bild 4-3 sind die Ein- und Ausgänge der Steuerungsschnittstelle im oberen Schnittstellenbereich eines Funktionsblockrumpfs, wie er in ähnlicher Form in einem grafischen Programmierwerkzeug für FPGAs /62/ verwendet wird, zu sehen.

4.5.2 Schnittstellen zwischen den Funktionsblöcken

Die Signale zwischen den Funktionsblöcken stellen, je nach ihrer Aufgabe, ganz unterschiedliche Anforderungen an ihren Übertragungsweg. Es lassen sich dabei im Wesentlichen *getaktete* und *episodische* Daten und Signale unterscheiden, die sich wiederum in *echtzeitkritisch* und *echtzeitunkritisch* unterteilen, wie in Bild 4-4 verdeutlicht.

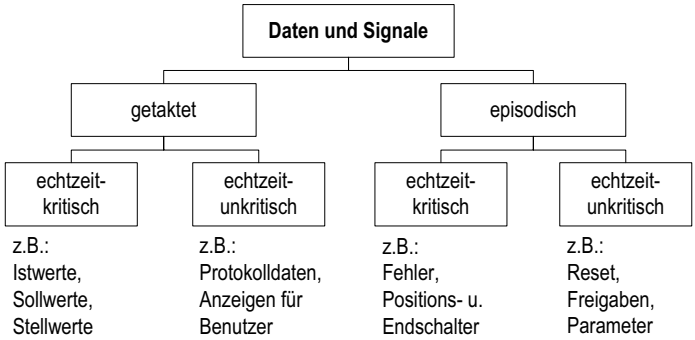

Bild 4-4: Eigenschaften von Signalen und Daten.

Zu den echtzeitkritischen, getakteten Daten und Signalen in der Antriebsregelung gehören die Istwertsignale, die Stellwerte und alle direkt an der Regelung beteiligten Signale.

Echtzeitkritische, getaktete Signale müssen zu äquidistanten Zeiten zwischen der Signalquelle und den Signalempfängern übertragen werden. Dies wird durch die direkte Verbindung der Funktionsblockein- und -ausgänge erreicht. Der Scheduler übernimmt dabei die Aufgabe der Steuerung und Überwachung der Übertragung.

Zu den echtzeitunkritischen, getakteten Signalen zählen z. B. Messsignale für eine verzögerte oder Offline-Weiterverarbeitung oder die Kommunikation mit einer Benutzerschnittstelle. Diese Signale müssen nicht in äquidistanten Zeiten übertragen werden. Da hierdurch der Zeitpunkt, an dem ein gültiges Signal zur Verfügung steht,

unbekannt ist, muss durch ein zusätzliches Signal von der Signalquelle angezeigt werden, ob gültige Signale für die Weiterverarbeitung vorhanden sind.

Nichtgetaktete, echtzeitkritische Signale sind beispielsweise Triggersignale, die ein Ereignis anzeigen und eine sofortige Reaktion erfordern, wie z. B. das Auslösen von Bremsen bei einem Spannungsabfall in der Leistungsversorgung des Antriebs. Diese Signale werden direkt mit den betreffenden Funktionsblöcken oder mit dem Scheduler verbunden, wenn diese zur Ausführung alternativer Abläufe führen sollen.

Nichtgetaktete, echtzeitunkritische Signale sind z. B. Parameter, die bei der Inbetriebnahme durch den Benutzer gesetzt werden. Insbesondere die Kommunikation mit einer Benutzerschnittstelle erfolgt über nicht getaktete, echtzeitunkritische Signale.

Die Funktionen innerhalb eines Funktionsblocks werden direkt mit den Ein- und Ausgängen des Funktionsblocks verbunden. Zum Ausführungsbeginn des Funktionsblocks werden an den Eingängen gültige Daten erwartet. Nach erfolgreicher Ausführung der Funktion liegen bis zum nächsten Funktionsaufruf gültige Daten an den Ausgängen des Funktionsblocks an.

Das Format der Daten kann vom Programmierer des Funktionsblocks frei gewählt werden. Da sich der Funktionsblock jedoch mit anderen Funktionsblöcken verbinden lassen soll, liegt es in der Verantwortung des Programmierers Konverter bereitzustellen, die das von ihm gewählte Datenformat in andere Datenformate umwandeln. Die gebräuchlichsten Datenformate sind:

- **Ganzzahlen**: Jedes Bit des Datenworts steht für eine 2er Potenz. Bei vorzeichenbehafteten Daten stellt das hochwertigste Bit das Vorzeichen dar. Die Darstellung von negativen Zahlen erfolgt über das Einerkomplement /63/.
- **Festkommazahlen** in Q-Notation /64/: Auch hier steht jedes Bit des Datenworts für eine 2er-Potenz. Bei vorzeichenbehafteten Zahlen steht das höchst wertige Bit für das Vorzeichen. Bei negativen Zahlen erfolgt die Darstellung als Einerkomplement. Zusätzlich zum Datenwort muss bekannt sein, wie viele der niederwertigen Bits die Nachkommastelle darstellen, wie in Bild 4-5 verdeutlicht. Diese Nachkommastellen fließen als 2er-Potenzen mit negativem Exponent in die

dargestellte Festkommazahl mit ein. Eine Festkommazahl in Q-Notation Q*r.s* berechnet sich nach (4.1):

$$x = \sum_{i=0}^{r+s-1} b_i 2^{i-s} \tag{4.1}$$

Der Vorteil dieser Darstellungsform ist, dass die gleichen Rechenwerke, wie für Ganzzahlen genutzt werden können. Der abdeckbare Wertebereich sowie die erreichbare Auflösung ist durch Vor- und Nachkommabits bestimmt.

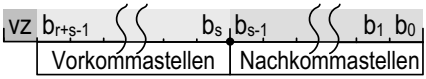

Bild 4-5: Interpretation der Bits einer Festkommazahl

- **Gleitkommazahlen:** Nach Norm /65/ steht das hochwertigste Bit für das Vorzeichen, eine festgelegte Anzahl von nachfolgenden Bits stellen den Exponenten *E* und die verbleibenden Bits die Mantisse *m* dar, wie in Bild 4-6 gezeigt. Der Exponent *E* ist immer positiv. Um auch negative Exponenten zu erhalten, wird von *E* ein Biaswert *B* abgezogen um den tatsächlichen Exponenten *e* zu berechnen. Die standardisierte Darstellungsform von Gleitkommazahlen sieht darüber hinaus eine Normierung der dargestellten Zahl vor, sodass die Mantisse *m* die Nachkommastellen einer ‚1' in der Form *1,m* abbildet. Da diese Normierung zu erheblichem Rechenaufwand führt, wird beim Einsatz auf FPGAs auf diese Normierung verzichtet /66/. Die Gleitkommazahl berechnet sich gemäß der Formel (4.2):

$$x = m \cdot 2^e \tag{4.2}$$

Der Vorteil der Gleitkommazahl gegenüber Festkommazahlen besteht darin, dass die Position des Kommas durch den Exponenten im Datenwort enthalten und kein zusätzliches Wissen für die Interpretation des Datenworts notwendig ist. Jedoch sind Rechenwerke für die Verarbeitung dieses Zahlenformats sehr aufwendig und werden in Mikrocontrollern oft in eigenständige *Floatingpoint-Units* (FPUs) ausgelagert. Die Anzahl der Bits *k*, die auf die Mantisse entfallen und die Anzahl der Bits *l*, die auf den Exponenten entfallen, müssen vorab festgelegt werden. Auf

dem FPGA sollte, um Ressourcen zu sparen, vorzugsweise mit Festkommaarithmetik gearbeitet werden.

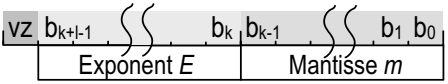

Bild 4-6: Interpretation der Bits einer Gleitkommazahl

Die Ein- und Ausgänge der Funktionsblöcke werden entweder direkt miteinander verbunden oder es können Signalblöcke dazwischen geschaltet werden, die Anpassungs- und Überwachungsfunktionen übernehmen. Mit Hilfe von Signalblöcken zur Signalanpassung, wie in Bild 4-7 gezeigt, können Datenverbindungen mit unterschiedlichen Ausgangs- und Eingangsbitbreiten hergestellt werden:

- Ist die Eingangsbitbreite kleiner als die Ausgangsbitbreite, muss ein Begrenzer eingesetzt werden, der das Signal auf einen Wert oder bei vorzeichenbehafteten Signalen auf einen Wertebereich begrenzt, der sich mit der Eingangsbitbreite darstellen lässt.
- Ist die Eingangsbitbreite größer als die Ausgangsbitbreite, muss dass Signal entsprechend erweitert werden. Bei vorzeichenlosen Signalen werden die erweiterten höherwertigen Bits auf ‚0' gesetzt. Bei vorzeichenbehafteten Signalen werden die erweiterten höherwertigen Bits auf den Wert des höchst wertigen Bits des Eingangsignals gesetzt, um das Vorzeichen zu erhalten.
- Haben die verbundenen Ein- und Ausgänge unterschiedliche Datenformate, muss ein Konverter zwischengeschaltet werden. Die zugelassenen Datenformate sind beliebige Darstellungen als Ganzzahlen oder Festkommazahlen in Q-Notation der Form Q*n.m*, sowie Gleitkommazahlen. Die Konvertierung erfolgt durch Verschieben und eventuell Erweitern des Eingangswerts.

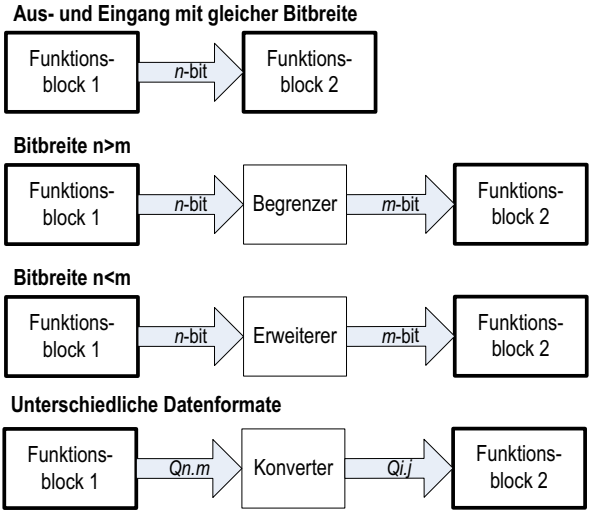

Bild 4-7: Signalanpassung zwischen Funktionsblöcken

Überwachungssignalblöcke überwachen den Wertebereich der Daten oder prüfen die Plausibilität, indem sie das Eingangssignal mit einem zweiten Eingangsignal vergleichen und bei einer zu starken Abweichung ein Warnsignal setzen.

4.5.3 Funktionsblockbeschreibung

Eine wichtige Komponente der Funktionsblöcke ist ihre Beschreibung. Die Beschreibung erfüllt zweierlei Aufgaben:

1. **Funktionsbeschreibung für den Anwender:** Der Anwender kann der Beschreibung entnehmen, welche Aufgaben, Ein- und Ausgänge und Parameter der Funktionsblock hat. Die Beschreibung unterstützt den Anwender bei der Auswahl von Funktionsblöcken aus der Bibliothek.
2. **Funktionsbeschreibung für eine Entwicklungsumgebung:** Hier werden alle Informationen hinterlegt, die für eine Entwicklungsumgebung notwendig sind. Dazu gehören:

- **Datenschnittstellen**: Welche Ein- und Ausgänge gibt es, wie sehen die Datenformate aus (Bitbreiten, Datenformat in Q-Notation)?
- **Busschnittstellen**: Welche nicht-echtzeitkritischen Daten können über den Bus übertragen werden? Wie sind die Adressen dieser Daten?
- **Zeitverhalten**: Welche Ausführungstakt-Anforderungen müssen erfüllt werden, wie lang ist die Ausführungszeitdauer?
- **Ressourcenbedarf:** Wie viele logische Einheiten, Speicherzellen und DSP-Blöcke werden auf dem FPGA belegt?

Auf Basis dieser Funktionsbeschreibung kann die Entwicklungsumgebung den Anwender bei der korrekten Konfiguration eines Antriebsreglersystems unterstützen und mithilfe der Informationen zum Zeitverhalten einen Ausführungszeitplan generieren.

4.5.4 Treiberfunktionsblöcke: Schnittstelle zur Hardware

Die Schnittstelle zu den Hardwaresubmodulen wird von einer eigenen Klasse von Funktionsblöcken, den Treiberfunktionsblöcken, hergestellt. Neben den Schnittstellen der Funktionsblöcke verfügen sie über eine weitere Schnittstelle zu einer Hardware-submodulverwaltung. Diese Schnittstelle besteht aus den Datenleitungen, die mit den Datenleitungen des zugehörigen Hardwaresubmoduls verbunden werden müssen. Des weiteren enthält sie Konfigurationsleitungen, über die der Treiberfunktionsblock der Hardwareverwaltung mitteilen kann, mit welchem Hardwaresubmodul er verbunden werden soll und über die er weitere Informationen mit dem Hardwaresubmodul, z. B. Parameter, austauschen kann.

4.5.5 Hardwareverwaltung

Die Verbindung zwischen den Hardwaresubmodulen und den Treiberfunktionsblöcken erfolgt, wie zuvor beschrieben, über eine Datenschnittstelle, die aus mehreren Datenleitungen besteht, deren Richtung nicht definiert ist.

Um zu gewährleisten, dass die Hardwaresubmodule mit den passenden Treiber-funktionsblöcken verbunden und somit auch die Datenleitungen richtig geschaltet

werden, wird zwischen den Steckplätzen der Submodule und den Treiberfunktionsblöcken eine Hardwareverwaltung eingefügt. Die Hardwareverwaltung liest über die Konfigurationsleitungen die eindeutigen Identifikationsnummern der Submodule sowie die der Treiberfunktionsblöcke und verschaltet die zusammenpassenden Komponenten. Die Verbindungen zwischen den Submodulen, der Hardwareverwaltung und den Treiberfunktionsblöcken sind in Bild 4-8 skizziert.

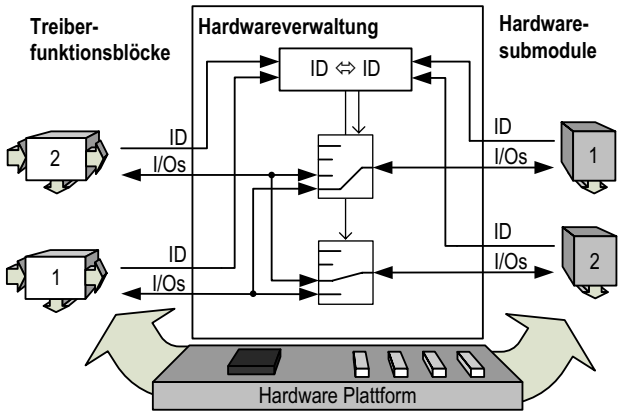

Bild 4-8: Verbindungen zwischen Treiberfunktionsblöcken und Hardwaresubmodulen über die Hardwareverwaltung.

4.5.6 Ausführung von Funktionsblöcken

Die Ausführungsweise der Funktionsblöcke kann in drei Kategorien unterteilt werden:

1. Funktionsblöcke, deren Ausführung durch das Auftreten bestimmter **Ereignisse** bestimmt wird.
2. Funktionsblöcke, die getaktet in einem festen **Zeitraster** ausgeführt werden.
3. Funktionsblöcke, die in Abhängigkeit von **Ereignissen** oder dem Systemzustand **getaktet** ausgeführt werden.

Für alle drei Kategorien muss die Antriebsreglerplattform geeignete Mechanismen anbieten, um die erforderlichen Funktionsabläufe realisieren zu können.

Im einfachsten Fall wird die Ausführung eines Funktionsblocks durch das Auftreten eines **Ereignisses**, wie beispielsweise das Überschreiten eines Schwellwertes oder das Setzen eines externen Digitaleingangs, ausgelöst. Das Auftreten eines Ereignisses wird an eine übergeordnete Überwachungsfunktion gemeldet. Diese sendet dann, wie in Bild 4-9 gezeigt, dem zugehörigen ereignisgesteuerten Funktionsblock das Ausführungssignal und wartet auf die Rückmeldung über den Rückmeldeausgang des Funktionsblocks, um die korrekte zeitliche Ausführung zu überwachen.

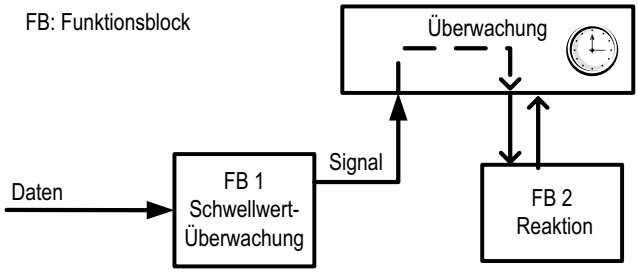

Bild 4-9: Ereignisgesteuerte Ausführung eines Funktionsblocks.

Sollen Funktionsblöcke getaktet in einem festen **Zeitraster** ausgeführt werden, muss zunächst durch ein, nachfolgend im Kapitel 5.4, beschriebenes Entwicklungswerkzeug ein Zeitplan erstellt werden, der die Ausführungszeitpunkte aller getakteter Funktionsblöcke nach vorgegebenen Optimierungskriterien festlegt.

Mithilfe dieses Zeitplans werden die Funktionsblöcke von einer übergeordneten Funktion, dem *Scheduler*, über den Ausführungssignaleingang aktiviert. Sie lesen die an den Eingängen anliegenden Signale ein und berechnen daraus innerhalb einer festen Ausführungszeit die Signale, die sie an ihre Ausgängen bereitstellen.

Die einfache zeitgesteuerte Ausführung der Funktionsblöcke setzt voraus, dass jeder einzelne Funktionsblock die angenommene maximale Ausführungszeit nicht überschreitet. Wenn eine Störung oder ein Fehler im System dazu führt, dass ein Funktionsblock zu spät oder nie ein gültiges Ausgangssignal erzeugt, führt das dazu,

dass die im Signalfluss nachfolgenden Funktionsblöcke mit fehlerhaften oder ungültigen Signalen weiterarbeiten und der Fehler weiter durch das System transportiert wird.

In Systemen, in denen die Ausführungszeiten einzelner Funktionsblöcke nicht garantiert werden können, ist es notwendig, die Ausführungszeiten zu **überwachen** und im Fehlerfall entsprechende Maßnahmen zu ergreifen.

Der Scheduler ist in diesem Falle nicht nur dafür zuständig, den Funktionsblöcken die Ausführungssignale zu senden, sondern er überwacht auch, wie Bild 4-10 gezeigt, ob von den Funktionsblöcken in einem vorgegebenen Zeitraum ein entsprechendes Rückmeldesignal zurückkommt. Empfängt der Scheduler dieses nicht innerhalb des festgesetzten Zeitraums, wird eine Fehlerbehandlung eingeleitet. Diese kann abhängig davon erfolgen, welcher Funktionsblock den Fehler erzeugt hat, sodass eine dem Fehler angemessene Reaktion möglich ist. Beispielsweise kann ein unkritischer Fehler als Warnung an den Nutzer weitergegeben werden, ohne den Ablauf der Regelung zu unterbrechen, wohingegen im Falle eines kritischen Fehlers auf alternative Zeitpläne für ein kontrolliertes Stillsetzen der Antriebe umgeschaltet wird.

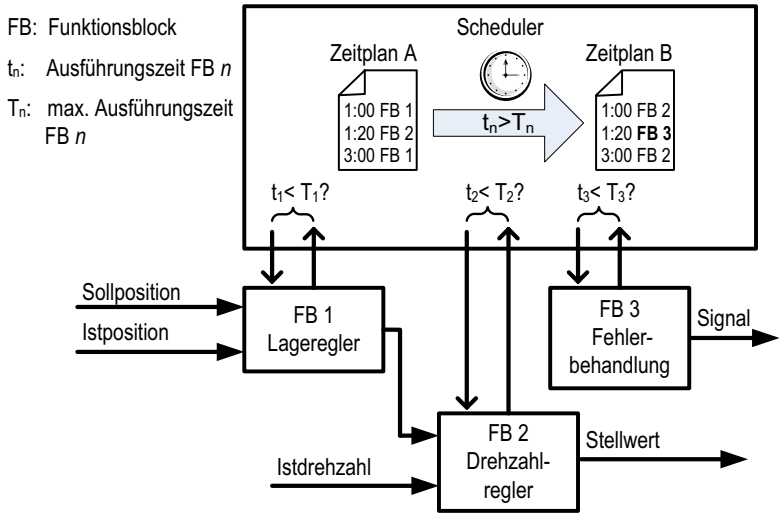

Bild 4-10: Ausführung von Funktionsblöcken mit Überwachung der Ausführungszeit.

Die Funktionsblöcke, die in diesen alternativen Zeitplänen eingeplant sind, bilden die dritte Kategorie von Funktionsblöcken, die ebenfalls in einem festen **Zeitraster** ausgeführt werden, jedoch nur beim Auftreten bestimmter **Ereignisse**. Dies sind beispielsweise Funktionsblöcke, die im Fehlerfall die Maschine geregelt in einen sicheren Zustand bringen, indem sie Sollwerte für eine Bremsrampe generieren und den Antrieb geregelt auf dieser Rampe herunterfahren.

4.5.7 Parallelisierung von Funktionen und Erstellung von Zeitplänen

Einer der wesentlichsten Unterschiede zwischen einem FPGA-basierten Antriebsreglersystem zu einem DSP-basierten ist die parallele Ausführung der Funktionen. Dadurch, dass jede Funktion als eigene digitale Schaltung auf dem Logikbaustein realisiert ist, kann sie, soweit der Signalfluss es zulässt, unabhängig von allen anderen Funktionen ausgeführt werden. Es können Signale in sehr hohen Takten aufgenommen und weiterverarbeitet werden, ohne dass dadurch die für die anderen Funktionen verbleibende Rechenzeit geschmälert wird, wie es bei einem sequenziell arbeitenden DSP-System der Fall ist.

In sequenziell arbeitenden Echtzeitsystemen werden Scheduler eingesetzt, die die Ausführung einzelner Tasks in einem Zeitplan festlegen. Über verschiedene Schedulingalgorithmen kann dabei beeinflusst werden, welche Tasks vorrangig abgearbeitet werden, und wie Tasks behandelt werden, die durch eine Kollision mit höherprioren Tasks nicht ausgeführt werden können.

Diese Algorithmen eignen sich nur bedingt für die Umsetzung auf einem parallel arbeitenden System, da sich hier die Tasks nicht gegenseitig beeinflussen, sondern unabhängig parallel ausgeführt werden können. In einem sequenziellen System ist prinzipbedingt dafür gesorgt, dass die vom Ergebnis eines vorangegangenen Tasks abhängigen Tasks erst nach deren Beendigung ausgeführt werden, vorausgesetzt, sie werden auf gleicher Prioritätsstufe ausgeführt. In einem parallel arbeitenden System kann ein Task auch vor Beendigung der im Signalfluss vorangehenden Tasks ausgeführt werden und somit auf der Grundlage eines veralteten oder ungültigen Signals operieren. Es muss daher gewährleistet werden, dass jeder Task erst dann ausgeführt wird, wenn an seinen Eingängen gültige Signale anliegen.

Auf der Antriebsreglerplattform entsprechen die Funktionsblöcke den Tasks eines Mikrocontrollers. Es zeigt sich, dass der Signalfluss die Parallelisierbarkeit der Funktionsblöcke begrenzt. Ein Regler kann beispielsweise erst dann auf einen gefilterten Sollwert regeln, wenn die Filterung abgeschlossen ist. Dies muss bei der Zeitplanerstellung berücksichtigt werden. Zwar kann schon ein neuer Sollwert gefiltert werden, während der Regler noch auf den alten Wert regelt, jedoch müssen beide Funktionsblöcke so zueinander versetzt ausgeführt werden, dass der Signalfluss nicht durch Wartezeiten unterbrochen wird.

Die Ausführungstakte jedes einzelnen Funktionsblocks sind abhängig von den Ein- und Ausgangstakten der mit ihnen verbunden Funktionsblöcke und von der benötigten Rechenzeit. Ein Funktionsblock muss in der ihm vorgegeben Taktzeit vollständig ausgeführt werden können.

In Bild 4-11 wird anhand eines Kaskadenreglers, bestehend aus Lageregler, Drehzahlregler und einem Stromregler, der Einfluss der Funktionsblockanordnung auf die Signallaufzeit verdeutlicht. Es zeigt sich in diesem Beispiel deutlich, dass die Signallaufzeit einer einfachen parallelen Anordnung der Funktionsblöcke erheblich verkürzt werden kann, wenn bei der Zeitplanung der Datenfluss berücksichtigt wird und die Ausführungszeiten der Funktionsblöcke um entsprechende Offsets verschoben werden.

Unter der Beachtung des Signalflusses und der Parallelisierbarkeit ergeben sich in einem System, das aus einer Vielzahl von Funktionsblöcken besteht, eine Vielzahl von Freiheitsgraden, wie die Blöcke im Ausführungszeitplan eingeplant werden können. Diese hohe Anzahl von Freiheitsgraden entsteht dadurch, dass die Funktionsblöcke zum einen in unterschiedlichen Takten ausgeführt werden und zum anderen ihre Eingangssignale aus mehreren unterschiedlichen Funktionsblöcken erhalten können, die wiederum zu unterschiedlichen Zeiten ihr Ausgangssignal zur Verfügung stellen.

Bei komplexen Regelsystemen, die mehrere Eingänge und Ausgänge haben, kann durch den Ausführungszeitplan das Zeitverhalten des Gesamtsystems beeinflusst werden. Aufgrund unterschiedlicher Takte und Ausführungszeiten wird es in der Regel nicht gelingen, die Funktionsblöcke so anzuordnen, dass es keine Verzögerungen im Signalfluss gibt. Diese Verzögerungen entstehen, wenn ein Funktionsblock auf das Ergebnis eines vorhergehenden Blocks warten muss. Es entstehen Totzeiten im System, die jedoch durch eine intelligente Zeitplanung in die Teile des Regelsystems verschoben

werden können, in denen sie den geringsten Einfluss auf die Dynamik des Gesamtsystems haben. Durch die gezielte Berücksichtigung eines ausgewählten Signalflusses zwischen einem bestimmten Eingang und einem bestimmten Ausgang ist es möglich, das Zeitverhalten und damit die Dynamik des jeweiligen Übertragungsverhaltens zu optimieren.

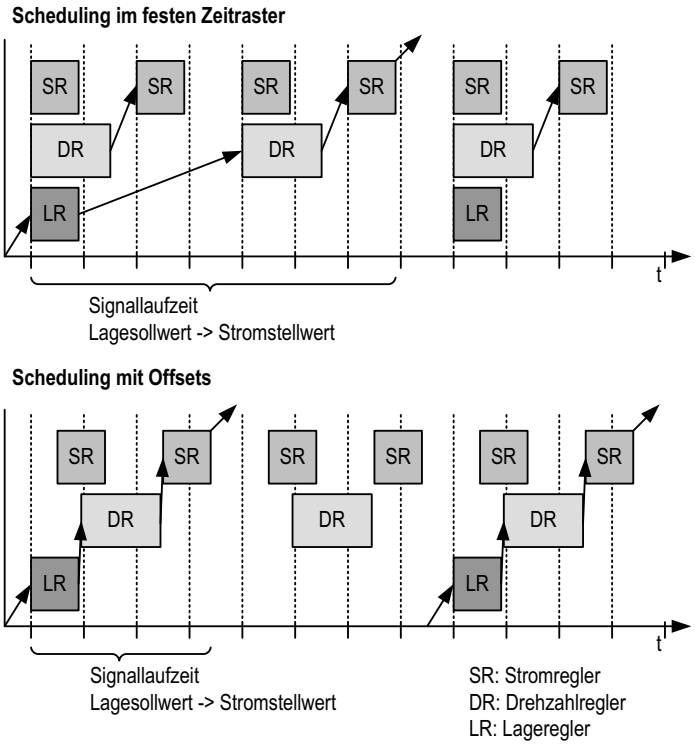

Bild 4-11: Gegenüberstellung von Zeitplänen mit Ausführungszeitpunkten in einem festen Zeitraster und verschoben mit Offsets.

Beispiele aus der Antriebstechnik verdeutlichen, wie sich unterschiedliche Optimierungskriterien nutzen lassen, um das Reglerverhalten zu beeinflussen. Für Werkzeug-

maschinen, bei denen es auf eine hohe Präzision und damit auf ein sehr gutes Führungsverhalten ankommt, werden die Ausführungszeiten der Funktionsblöcke so optimiert, dass die entstehenden Totzeiten im Kaskadenregler möglichst gering, und damit die Dynamik und das Führungsverhalten möglichst gut ist. Bei Maschinen mit einer Prozessregelung hingegen, wie beim Laserschweißen /67/, bei dem der Abstand zwischen Laserstrahl und Werkstück konstant gehalten werden muss, kann man die Funktionsblöcke so anordnen, dass der auf der Antriebsreglerplattform integrierte Prozessregler die geringsten Totzeiten aufweist, um hier die optimale Regelgüte zu erreichen.

Die Zeitplanerstellung wird außerhalb der Antriebsreglerplattform vom Programmierer mithilfe einer Planungssoftware erstellt, die die Anforderungen der eingesetzten Funktionsblöcken bezüglich Ausführungstakt und Ausführungszeit, sowie die Signalabhängigkeiten zwischen den Funktionsblöcken kennt.

4.6 Module in der Mikrocontroller-Ebene

Ein Funktionsblock kann neben einzelnen Antriebsfunktionen auch vollständige Mikrocontroller enthalten, deren Funktion wiederum flexibel durch Software bestimmt wird. Durch die freie Programmierbarkeit bilden die Mikrocontrollerfunktionsblöcke eine weitere, eigenständige Ebene der Offenheit auf der Antriebsreglerplattform.

Um Funktionsblöcke erstellen und sie in den Antriebsregler integrieren zu können, sind Kenntnisse in der FPGA-Entwicklung sowie eine FPGA-Entwicklungsumgebung erforderlich. Die Antriebsreglersoftware muss vor der Inbetriebnahme des Antriebsreglers auf das Regelgerät übertragen werden.

Mit den Mikrocontrollerfunktionsblöcken werden diese Nachteile umgangen. Der Funktionsblock muss zwar auch vor der Inbetriebnahme in das Antriebsreglersystem integriert werden, jedoch kann seine Funktionalität auch später, während der Laufzeit des Regelgeräts verändert werden, ohne dass der FPGA neu programmiert werden muss.

Die Entwicklung von Funktionen erfolgt in den weitverbreiteten Hochsprachen für Mikrocontroller (z. B. C oder C++) mithilfe entsprechender Entwicklungsumgebungen. Es sind keine Kenntnisse über FPGA-Entwicklung erforderlich.

4.6.1 Schnittstelle zum FPGA

Damit sich die Mikrocontrollerfunktionsblöcke korrekt in das Gesamtsystem einfügen können, muss eine klare Schnittstelle definiert werden. Zu dieser Schnittstelle gehören einerseits die Signale und Daten, die vom Mikrocontrollerfunktionsblock gelesen oder geschrieben werden und andererseits sein Zeitverhalten, damit er bei der Zeitplanerstellung berücksichtigt werden kann.

Die Signale und Daten, auf die zugegriffen werden soll, liegen als Ein- bzw. Ausgänge des Funktionsblocks an. Im Mikrocontroller stehen diese softwareseitig als Variablen oder Register zur Verfügung und können vom Anwender gelesen oder beschrieben werden. Welche Daten- und Signaleingänge dies sind und in welchem Format die Daten übertragen werden, wird bei der Erstellung des FPGA-Programms festgelegt und kann später, während der Laufzeit nicht mehr verändert werden.

Für Daten, die nicht in einem festen Takt gelesen oder geschrieben werden müssen, verfügt der Mikrocontrollerfunktionsblock über eine Anbindung an den Datenbus für nicht-echtzeitkritische Daten der Plattform.

Peripherie, die nur vom Mikrocontroller genutzt wird, kann entweder über die getakteten Daten- und Signalleitungen des Funktionsblocks oder direkt vom Funktionsblock angesprochen werden, ohne auf die Infrastruktur der Antriebsreglerplattform zurückzugreifen. Dazu gehört beispielsweise Speicher oder Schnittstellen zur visuellen Anzeigen bzw. Bedienelementen.

Des Weiteren verfügt der Mikrocontrollerfunktionsblock über die Steuerleitungen der Funktionsblöcke. Der Takteingang liefert den Systemtakt, während der Reset-Eingang das Reset-Signal für den Mikrocontroller liefert. Der Ausführungssignaleingang sowie der Rückmeldeausgang stehen der Software als Variablen oder Register zur Verfügung und müssen von der Software des Mikrocontrollers bedient werden.

Bild 4-12 zeigt einen Mikrocontrollerfunktionsblock mit einer Schnittstelle zu externem statischem, jedoch flüchtigem Speicher (SRAM, Static Random Access Memory), der ausschließlich vom Mikrocontroller genutzt wird.

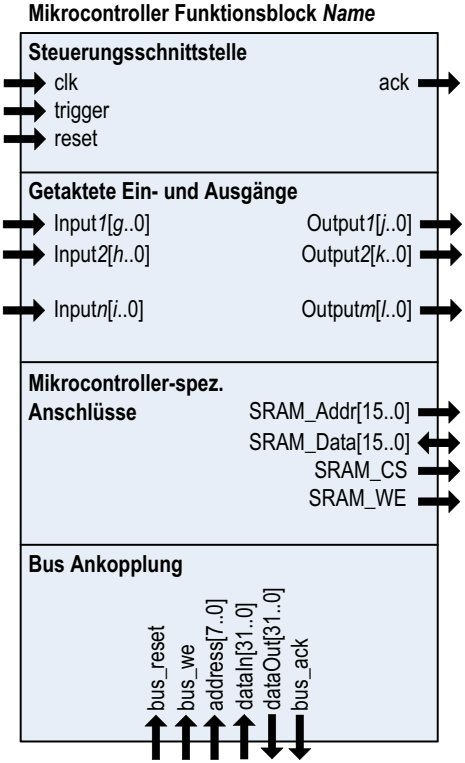

Bild 4-12: Mikrocontrollerfunktionsblock mit SRAM-Schnittstelle.

Abhängig davon, ob der Ausführungszeitplan vor oder nach der Mikrocontroller-Software erstellt wird, muss das Zeitverhalten des Funktionsblocks berücksichtigt werden:

- Wird der Ausführungszeitplan erstellt, bevor der Mikrocontroller programmiert wird, muss vorab festgelegt werden, in welchem Takt der Funktionsblock das Ausführungssignal erhält und in welchem Zeitraum er gültige Ausgangsdaten bereitstellen muss. Der Programmierer des Mikrocontrollers muss dann diese Randbedingungen einhalten. Ist sein Programm schneller, als vorgesehen, kann

das Gesamtsystem jedoch nicht von der Zeitersparnis profitieren, da der Zeitplan nicht mehr angepasst wird.

- Erfolgt die Zeitplanerstellung erst nach der Programmierung des Mikrocontrollers, können sowohl Ausführungstakt als auch Ausführungszeit flexibel den Anforderungen der Software angepasst und die Signallaufzeiten minimal gehalten werden, wie in Bild 4-13 zu sehen ist. Nachteilig ist hier jedoch, dass sich insbesondere bei einer Änderung des Ausführungstaktes auch die Ausführungstakte aller im Signalfluss nachfolgenden Funktionsblöcke ändern. Dies muss bei der Konfiguration des Systems berücksichtigt werden. Alternativ kann auch der Ausführungstakt des Mikrocontrollers vorab festgelegt werden.

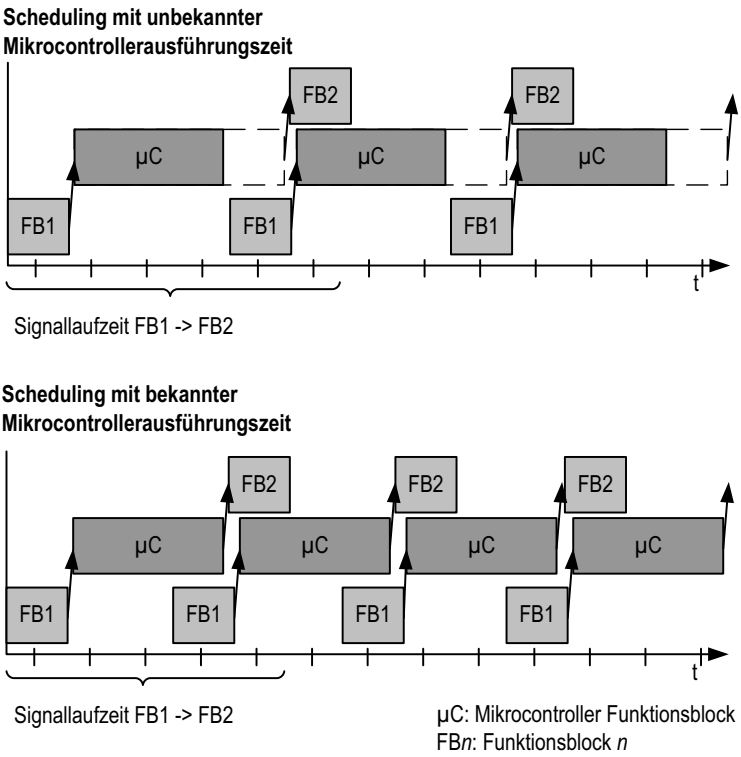

Bild 4-13: Scheduling bei unbekannter, aber begrenzter und bekannter Ausführungszeit der Mikrocontroller Software.

4.7 Bibliothek von Antriebsreglerkomponenten

Das Konzept der Offenheit und Modularität sieht vor, dass ein Antriebsreglersystem aus einzelnen, fertigen Komponenten konfiguriert wird. Diese Komponenten sind hardwareseitig die Submodule, die die Schnittstellen zum Umfeld des Antriebsreglers herstellen, sowie softwareseitig die Funktionsblöcke, die die für die Regelung benötigten Funktionen beinhalten.

Um ein Antriebsreglersystem konfigurieren zu können, muss eine umfassende Bibliothek zur Verfügung stehen, die die grundlegenden Schnittstellen und Funktionen bereitstellt, und zukünftig durch neue Schnittstellen und Funktionen, die von Forschung und Entwicklung hervorgebracht werden, erweitert wird.

Im Folgenden werden die Hardwaresubmodule und Funktionsblöcke des FPGAs vorgestellt, die für eine praxistaugliche Antriebsreglerkonfiguration notwendig sind.

4.7.1 Grundlegende Hardwaresubmodule

Für den Aufbau von vollständigen Achsregelkreisen einer Maschine, bestehend aus Steuerung, Antriebsreglern, Leistungsverstärker und Motoren, sind die in Tabelle 4-1 aufgeführten Hardwaresubmodule notwendig.

4.7.2 Grundlegende Funktionsblockbibliothek

Die grundlegenden Funktionsblöcke lassen sich in drei Gruppen unterteilen.

- **Elementare** Funktionsblöcke
- **Kombinierte** Funktionsblöcke
- **Antriebsreglerspezifische** Funktionsblöcke

Die **elementaren** Funktionsblöcke beinhalten in der Regel einfache mathematische Operationen und die Grundelemente der Regelungstechnik. Sie dienen vorrangig der Implementierung einfacher Funktionalitäten und der prototypenhaften Entwicklung neuer Algorithmen. Der Aufbau komplexer Schaltungen ist mit den elementaren

Submodul Bezeichnung	**Beschreibung**
Kommunikation	Feldbus- bzw. Antriebsbusschnittstelle zur Kommunikation mit einer übergeordneten Steuerung. Der Feldbus kann z. B. Sercos interface II/III, EtherCAT, oder ProfiDrive sein.
Geber	Geberschnittstellen für die direkten und indirekten Messsysteme des Motors. Die Schnittstellen sind z. B. Sinus-Cosinus, TTL, EnDAT
Digitale I/Os	Digitale Schnittstellen zum Einlesen von Referenz- oder Endschaltern, sowie zum Schalten einfacher Aktoren, wie Verriegelungen oder Bremsen, oder zur Anbindung an externe Schaltgeräte.
Motor	Schnittstelle zum Motor: Ausgabe von PWM-Signalen zur Ansteuerung des Leistungsverstärkers, Einlesen der gemessenen Istströme der Motorphasen

Tabelle 4-1: Grundlegende Hardwaresubmodule.

Blöcken grundsätzlich möglich, jedoch bringt jeder Funktionsblock einen Overhead, bestehend aus Schaltwerken für die Bedienung der Daten- und Steuerungsschnittstelle mit sich. Wird eine komplexe Verschaltung mehrerer elementarer Funktionsblöcke dauerhaft eingesetzt, kann sie zu einem kombinierten Funktionsblock zusammengefasst und in die Bibliothek zurückgeführt werden. In Tabelle 4-2 sind die wichtigsten **elementaren** Funktionsblöcke aufgeführt:

Funktionsblock Bezeichnung	**Beschreibung**
Grund-rechenarten	Jeweils ein Funktionsblock für jede Grundrechenart
geometrische Funktionen	Berechnet den Sinus, Cosinus, Tangens, Arcustangens eines Eingangssignals
P-Glied	Verstärker, Multiplikation mit einem konstanten Parameter
I-Glied	Integrierer, Element mit integralem Verhalten
D-Glied	Differenzierer, Element mit differenzierendem Verhalten
Begrenzer	Begrenzt den Wertebereich eines Signals
Totzeit	Verzögert das Eingangsignal um eine konstante Zeit

Tabelle 4-2: Elementare Funktionsblöcke der Funktionsblockbibliothek.

Darüber hinaus gibt es eine Vielzahl **kombinierter** Funktionsblöcke, die typische Zusammenstellungen von elementaren Funktionen enthalten. Insbesondere in der Regelungstechnik gibt es eine Reihe von Grundelementen, die sich aus den elementaren Funktionen kombinieren lassen. Tabelle 4-3 führt eine Auswahl der wichtigsten kombinierten Funktionsblöcke auf:

Funktionsblock Bezeichnung	**Beschreibung**
PI-Glied	Übertrager mit PI-Verhalten
PTx-Glied	Übertrager mit PTx-Verhalten mit parametrierbarer Ordnung
PID-Glied	Übertrager mit PID-Verhalten
FIR-Filter	Filter mit parametrierbarer Ordnung
Zähler	zählt Signalflanken an seinem Eingang
Vergleicher	vergleicht zwei Eingänge
Sliding Mode	Reglerbaustein für die Sliding Mode Regelung

Tabelle 4-3: Kombinierte Funktionsblöcke der Funktionsblockbibliothek.

Mit den elementaren und kombinierten Funktionsblöcken lassen sich beliebige Regelkreise und Antriebsfunktionen realisieren. Insbesondere spezielle nichtlineare Regler, wie sie in der heutigen Antriebstechnik kaum zu finden sind, wie beispielsweise Sliding Mode Regler /68/ oder H-unendlich-Regler, lassen sich mit Hilfe dieser Funktionsblöcke realisieren.

Es gibt jedoch einige Antriebsfunktionen, die in nahezu jedem Antriebsregler in gleicher Form wieder auftreten. Diese Antriebsfunktionen werden in eigenständigen, **antriebsreglerspezifischen** Funktionsblöcken zur Verfügung gestellt. Die wichtigsten Funktionsblöcke mit den Standardfunktionen der Antriebstechnik sind in Tabelle 4-4 dargestellt. Darüber hinaus können auch neuartige oder für spezielle Anwendungsfälle entwickelte Technologien in eigenständige Funktionsblöcke integriert werden. Zwei Beispiele für antriebsreglerspezifische Technologien, die aufgrund ihrer hohen Leistungsanforderungen eine Umsetzung auf dem FPGA notwendig machen, werden im Folgenden vorgestellt.

Funktionsblock Bezeichnung	**Beschreibung**
Geber-auswertung	Auswertung der Motorgeber
Feldbustreiber	Übertrager mit PTx-Verhalten mit parametrierbarer Ordnung
P u. PI-Regler	Vollständiger P bzw. PI-Regelkreis mit Eingangs- und Ausgangs-filtern, sowie Begrenzern
Kaskadenregler	Standard P-PI-PI Kaskadenregler, für die Strom- Drehzahl- und Lageregelung, mit Filtern und Begrenzern
Park-Clarke Transformation	Vorwärts- und Rückwärtstransformationen von Phasenströmen in feld- und momentenbildenden Strom für dreiphasige Motoren
Entkopplung	Kompensation der drehzahlabhängigen Gegenspannung des Antriebs im Stromsollwert
PWM-Generator	Erzeugung der 6 PWM-Signale für die Ansteuerung der IGBTs in den Halbbrücken des Leistungsverstärkers
Schnelle Stromregelung	hochdynamische Stromregelung mit Stromoversampling für niederinduktive Direktantriebe
Lage-oversampling	Verbesserte Geberauswertung durch hohe Abtastung der Spursignale eines Sinus-Cosinus-Gebers

Tabelle 4-4: Antriebsreglerspezifische Funktionsblöcke.

Der Funktionsblock **Schnelle Stromregelung** beinhaltet ein am ISW entwickeltes, FPGA-basiertes Stromregelverfahren /31,32/, das die Leistungsfähigkeit und Regelbandbreite der herkömmlichen Stromregelung steigert, um beispielsweise niederinduktive Direktantriebe mit einer höheren Dynamik betreiben zu können.

Bei der schnellen Stromregelung werden die Stromistwerte um ein Vielfaches des Stromregeltakts höher abgetastet (Stromoversampling), und kontinuierlich über einen Zeitraum, der deutlich kleiner ist als der Regeltakt, der Mittelwert gebildet, wie in Bild 4-14 gezeigt.

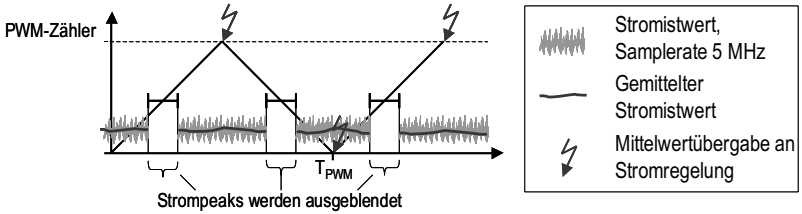

Bild 4-14: Messwertaufnahme bei schneller Stromregelung.

Der Mittelwert wird jeweils zu den Richtungswechseln des PWM-Zählers (Pulsweiten Modulation) für die Stromregelung verwendet, die noch während der Verriegelungszeit der Schaltbrücken des Leistungsverstärkers (IGBTs) den neuen Zähler-Schwellwert für die PWM-Umschaltung berechnen kann. Die im Regler durch die Sollwertberechnung entstehende Totzeit fällt dadurch nicht weiter ins Gewicht. Zusätzlich zur hohen Abtastung und Mittelung des Stroms werden bei der Stromistwerterfassung die Umschaltzeitpunkte der IGBTs ausgeblendet, da zu diesen große Störungen im Stromsignal auftreten. Die schnelle Stromregelung benötigt zum einen sehr schnelle Algorithmen zur Stromistwerterfassung und Mittelung und muss zum anderen mit der PWM-Erzeugung synchronisiert werden. Die hohen Anforderungen können durch die Umsetzung der PWM und der Stromregelung auf einem FPGA, wie in Bild 4-15 gezeigt, erfüllt werden. Der mit dem Stromregler maximal erreichbare Regeltakt ist durch das Schaltverhalten der IGBTs begrenzt, da diese kein ideales Ein- bzw. Ausschaltverhalten aufweisen. Wird zwischen Abschalten und Einschalten zweier IGBTs einer Motorphase eine IGBT-spezifische Wartezeit nicht eingehalten, kann ein Strom durch den noch nicht vollständig abgeschalteten IGBT direkt über den eingeschalteten fließen, wodurch die Verlustleistung des Antriebs ansteigt, und sich die IGBTs stark erwärmen.

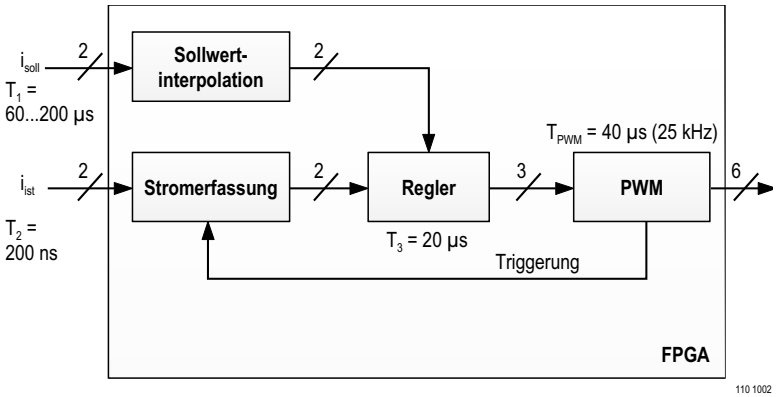

Bild 4-15: Aufbau der Stromregelung auf dem FPGA.

Der Funktionsblock **Lageoversampling** stellt das am ISW entwickelte Verfahren für eine verbesserte Geberauswertung durch eine Überabtastung der Spursignale zur Verfügung. Mit einem Lageoversampling kann Messrauschen reduziert und Ungenauigkeiten des Messsystems, das Lagesignal mit Oberwellen überlagern, die sowohl hör- und messbar sind, kompensiert werden. Auf die Funktionsweise soll hier nur oberflächlich eingegangen werden. Im Gegensatz zur gängigen Lageistwerterfassung werden die Geber nicht im jeweiligen Regeltakt mit bis zu 16 kHz, sondern mit einem deutlich höheren Takt von 1 MHz abgetastet. Aus den abgetasteten Werten wird der Winkel berechnet und die daraus differenzierte Geschwindigkeit in einen Ringspeicher geschrieben. Die Werte im Ringspeicher werden gemittelt und als hoch aufgelöste Geschwindigkeit und aufintegriert als hoch aufgelöste Lage ausgegeben, wie in Bild 4-16 gezeigt.

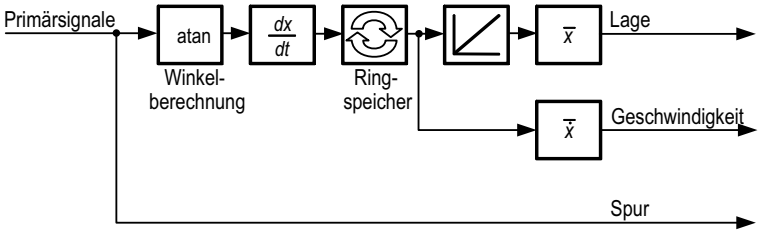

Bild 4-16: Oversampling des Lagesignals.

Eine weiterentwickelte Variante des Oversampling passt die Länge des Ringspeichers, in Abhängigkeit von der Geschwindigkeit, auf das Vielfache einer Periodenlänge der auftretenden Oberwellen an. Dadurch werden für die Mittelungen vollständige Perioden der Oberwellen herangezogen, wodurch die Genauigkeit erheblich steigt.

Eine weitere Verbesserung des Lagesignals kann durch eine Offset-Amplitudenregelung erreicht werden, die das mit Offset- und Amplitudenfehlern behaftete Signal von Sinus-Cosinus-Gebern korrigiert, und damit genauere Berechnung der Lage ermöglicht.

Aufgrund der hohen Taktrate des Oversamplings und der zeitgleich laufenden Offset- und Amplitudenregelung ist für die Realisierung dieses Verfahren ein FPGA besonders geeignet.

4.8 Schutz von Know-how

Der Schutz von Know-how ist insbesondere bei offenen Systemen von großer Bedeutung, da ein besonderer Nutzen der Offenheit darin liegt, Komponenten verschiedener Hersteller zu kombinieren. Damit die verschiedenen Hersteller sich nicht gegenseitig ihr Know-how in Form von einsehbaren Quellcodes oder Logikschaltungen für die Fertigstellung des Gesamtsystems preisgeben müssen, ist es notwendig, geeignete Mechanismen vorzusehen, die das unnötig machen.

Ein grundlegender Bestandteil des Know-how Schutzes ist die zuvor beschriebene Modularisierung in Hardwaresubmodule und Funktionsblöcke sowie die Einhaltung der definierten Schnittstellen. Eine als Hardwaresubmodul oder Funktionsblock gekapselte Funktion muss für den Anwender nicht weiter einsehbar sein. Lediglich die Schnittstellen und Parameter, sowie deren Benutzung muss ausreichend dokumentiert werden.

Dem Hersteller von Hardwaresubmodulen stehen für den Know-how Schutz die gleichen Mechanismen zur Verfügung, wie jedem anderen Hardwarehersteller. Teile der Schaltung können beispielsweise durch programmierbare Logik oder Mikrocontroller realisiert werden, deren Software verschlüsselt abgespeichert wird.

Logische Schaltungen auf dem FPGA, aus denen die Funktionsblöcke bestehen, lassen sich in Form von Netzlisten (Netlists) bereitstellen. Netzlisten beschreiben die Verknüpfungen der logischen Elemente auf dem FPGA und sind vergleichbar mit fertig kompilierten Programmen für den Mikrocontroller. Diese Netzlisten lassen sich mit weiteren Netzlisten und eigenen Schaltungen zu einem Gesamtsystem kombinieren. Die ursprüngliche Schaltung und der dahinter stehende Algorithmus lässt sich nur begrenzt und mit erheblichem Aufwand aus der Netzliste rekonstruieren. Der Schutz vor einer Rekonstruktion der logischen Schaltung kann durch den Einsatz von verschlüsselten Netzlisten noch weiter verbessert werden. Die Hersteller von FPGAs sowie verschiedene Dienstleistungsunternehmen nutzen die verschlüsselten Netzlisten, um dem Kunden vorgefertigte Funktionen für den FPGA, wie beispielsweise USB-Schnittstellen, Mikrocontroller oder Filter, zur Verfügung zu stellen /69,70/. Der Decoder, der die verschlüsselte Netzliste mit einem mitgelieferten Schlüssel verarbeiten kann, ist fester Bestandteil der FPGA-Entwicklungsumgebung. Die decodierte Netzliste wird von der Entwicklungsumgebung direkt für die Erzeugung der Verschaltungen auf dem FPGA verwendet und ist für den Anwender unsichtbar. Bisher sind noch keine einheitlichen Verschlüsselungsverfahren festgelegt worden. Jedoch sind in den IEEE-Normen /71/ für Verilog bereits Mechanismen zur Einbettung von Verschlüsselungsmechanismen in Verilog vorgesehen. Geeignete Verschlüsselungsverfahren sind beispielsweise Data Encryption Standard (DES), Triple DES und Advanced Encryption Standard (AES) /72/.

Schützenswertes Know-how, das in der Software für Mikrocontrollerfunktionsblöcke enthalten ist, wird dadurch geschützt, dass es nur als fertig kompiliertes Programm in Maschinensprache weitergegeben wird und somit nicht direkt einsehbar ist.

Insbesondere die Mikrocontrollerfunktionsblöcke bieten eine weitere Möglichkeit, Know-how zu schützen. Besondere Prozess- oder Verfahrenskenntnisse, die beim Maschinenhersteller liegen, brauchen nicht an den Antriebsreglerhersteller weitergegeben zu werden, damit sie z. B. als Prozessregler in den Antriebsregler integriert werden können. Es reicht, wenn der Antriebsreglerhersteller dem Maschinenhersteller einen Mikrocontrollerfunktionsblock bereitstellt, der Zugriff auf die notwendigen, reglerinternen Signale bietet. Der Maschinenhersteller kann dann seine Prozessregelung in einer ihm gängigen Hochsprache programmieren und nachträglich den für ihn bereitgestellten Mikrocontrollerfunktionsblock laden.

Über die hier beschriebenen Möglichkeiten hinaus kann das Programm des FPGAs verschlüsselt auf einem nichtflüchtigen Speicher abgelegt werden. Ein Decoder, der sich auf dem FPGA befindet, liest die verschlüsselten Daten, decodiert sie und programmiert den FPGA. Der Inhalt des programmierten FPGA wird gegen ein Auslesen geschützt und zu keinem Zeitpunkt liegt die Schaltung des FPGAs in lesbarer, unverschlüsselter Form vor. Die Verschlüsselung erfolgt mithilfe einer im FPGA hinterlegten, eindeutigen Seriennummer. Dadurch kann die Software auch vor einer unerlaubten Vervielfältigung geschützt werden /73/.

5 Entwurf einer Entwicklungsumgebung

Bedingt durch die Offenheit der Antriebsreglerplattform und der vielfältigen und flexiblen Programmier- und Nutzungsmöglichkeiten ist eine umfangreiche Entwicklungsumgebung notwendig, die die Bedürfnisse und Anforderungen aller Nutzergruppen berücksichtigt. Die Entwicklungsumgebung stellt sämtliche Entwicklungswerkzeuge zur Verfügung, die von den einzelnen Anwendergruppen benötigt werden und definiert Schnittstellen zwischen den Werkzeugen, sodass jede Gruppe auf die Arbeiten der anderen zugreifen kann, ohne selbst in den anderen Ebenen tätig zu werden. Die Zuordnung der Anwendergruppen zu den drei Ebenen der Offenheit ist in Bild 5-1 dargestellt.

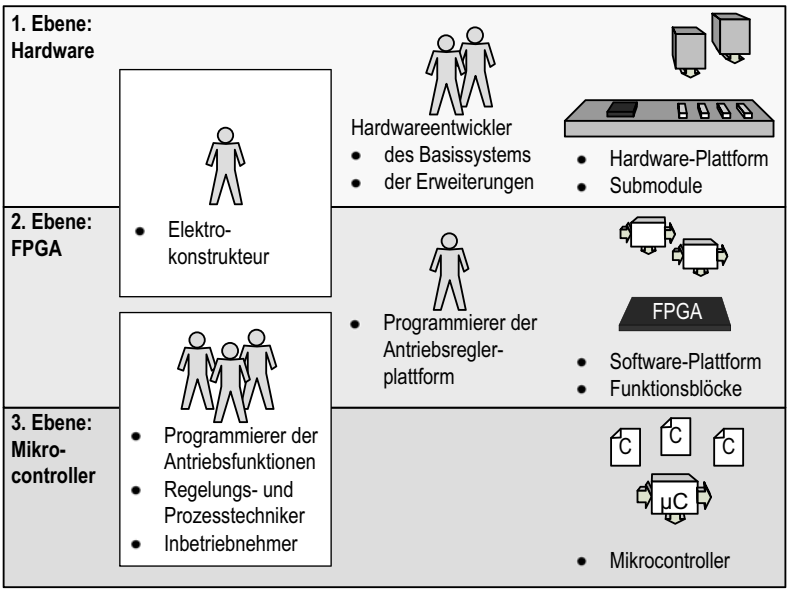

Bild 5-1: Zuordnung der Anwendergruppen zu den drei Ebenen der Offenheit.

Im Idealfall wird allen Anwendergruppen eine durchgängige Entwicklungsumgebung zur Verfügung gestellt, die den gesamten Prozess vom Schaltungsentwurf bis zur Inbetriebnahme in einer einheitlichen Umgebung zulässt, die den verschiedenen An-

wendergruppen entsprechende Funktionen und Sichten der Antriebsreglerplattform bietet. Aufgrund des sehr hohen Funktionsumfangs einer solchen Entwicklungsumgebung und den Herstellerabhängigkeiten der einzelnen Entwicklungswerkzeuge erweist es sich jedoch als flexibler und kostengünstiger, die Entwicklungsumgebung aus bereits bestehenden Entwicklungswerkzeugen zusammenzustellen und durch plattformspezifische Hilfswerkzeuge zu ergänzen. Die Anwendergruppen können dann mit ihren gewohnten Standardwerkzeugen arbeiten. Die Integration der Standardwerkzeuge sowie deren Zusammenspiel wird über fest definierte Schnittstellen und übergeordnete Software erreicht.

Im Folgenden werden die Anwendergruppen, sowie die von ihnen benötigten Werkzeuge und Schnittstellen betrachtet.

5.1 Hardwareentwickler des Basissystems und der Erweiterungen

Der Hardwareentwickler des Basissystems entwirft eine Hardwareplattform gemäß den Anforderungen, die durch das geplante Antriebsreglersystem gestellt werden. Dies betrifft die Auswahl des FPGAs, die auf der Hardwareplattform zur Verfügung stehenden Peripherie sowie die Anzahl der Schnittstellen zu Hardwareerweiterungen und deren Übertragungskapazität (Datenmenge, Übertragungsgeschwindigkeit) für Hardwaresubmodule.

Der Hardwareentwickler der Erweiterungen entwirft die notwendige zusätzliche Hardware für den Antriebsregler, sogenannte Submodule. Diese bestehen in der Regel aus Schnittstellenkarten für die Integration neuer Sensorik, Aktorik oder Steuerungen. In besonderen Fällen können auch Sonderfunktionalitäten, wie z. B. spezielle Prozessrechner oder Datenerfassungsgeräte, als Hardware auf der Antriebsreglerplattform integriert werden.

Schaltpläne und das Layout der Schaltung werden von den Hardwareentwicklern mit Standard EDA-Werkzeugen (Electronic Design Automation) erstellt und getestet. Die Schnittstelle zwischen den Hardwareentwicklern der Plattform und der Submodule beschränkt sich auf die Hardwareschnittstelle zwischen Plattform und Submodulen. Für die Entwicklung der Submodule müssen dabei zwei Komponenten der Schnittstelle berücksichtigt werden. Das Submodul muss elektrisch zur Schnittstelle der

Antriebsreglerplattform passen. Es muss festgelegt werden, wie das Submodul mit Energie versorgt wird. Spannungen sowie die maximale Stromaufnahme werden durch die Antriebsreglerplattform vorgegeben. Ebenfalls zur elektrischen Anbindung zählen die zur Verfügung stehenden Datenleitungen. Hier wird die Anzahl der Datenleitungen, die maximalen Taktraten auf den Leitungen, eventuell die unterstützten Übertragungsformen, wie z. B. das differenzielle LVDS-Verfahren (Low Voltage Differential Signaling), sowie die Übertragungsrichtungen auf den Leitungen definiert. Zu der beschriebenen elektrischen Schnittstelle gehört schließlich auch die mechanische Schnittstelle, die zum einen aus der Steckverbindung zwischen Submodul und Hauptplatine und zum anderen aus der geometrischen Abmessung für die Integration in ein Gehäuse besteht.

Zu jedem Submodul gehört auf dem FPGA ein Treiberfunktionsblock, der die Funktionen der Hardware anderen Funktionsblöcken auf der Hardwareplattform zur Verfügung stellen kann. Der Hardwareentwickler der Erweiterungen muss daher den Programmierern der Antriebsreglerplattform und Antriebsreglerfunktionen Informationen liefern, wie die Datenleitungen zwischen Plattform und Submodul zu nutzen sind, und wie die zeitlichen Anforderungen (Ausführungstakt, Ausführungsdauer) des Submoduls sind, damit die Programmierer einen passenden Treiber entwickeln können.

Der Entwickler der Hardwareplattform muss den Programmierern der Antriebsreglerplattform und Antriebsreglerfunktionen die Schnittstelle zu seiner Hardware bereitstellen. Diese besteht im Wesentlichen aus einer Beschreibung, über welche Anschlusspins des FPGAs die Peripherie und die Submodulschnittstellen mit dem FPGA verbunden sind.

5.2 Programmierer der Antriebsreglerplattform

Die Aufgabe des Programmierers der Antriebsreglerplattform ist es, die Infrastruktur der Plattform auf dem FPGA zu implementieren. Hierzu gehören der Scheduler, Kommunikationswege für die getaktete und episodische Kommunikation sowie sämtliche Sicherheitseinrichtungen, die die Abläufe und die Kommunikation auf der Plattform überwachen.

Die Infrastruktur befindet sich hauptsächlich auf dem FPGA. Es können jedoch Teile davon von DSPs und Mikrocontrollern übernommen werden, die sich entweder auf dem FPGA befinden oder auch als externe Hardware an diesen angeschlossen sind.

Speziell für die FPGA-Entwicklung werden herstellerspezifische Entwicklungssoftwarepakete, die eigens auf den jeweiligen FPGA zugeschnitten sind, eingesetzt. Sie bieten umfangreiche Werkzeuge zur Entwicklung, zur Simulation sowie zur Programmierung. Ähnlich sieht es auch bei der Entwicklungssoftware für die DSPs und Mikrocontroller aus. Auch hier sind die Entwicklungsumgebungen meist hersteller- oder prozessorarchitekturspezifisch.

Der Plattformentwickler benötigt darüber hinaus keine weiteren Hilfswerkzeuge. Die Durchgängigkeit der Entwicklung stellt er durch die Bereitstellung der Infrastruktur mit allen definierten Schnittstellen her, auf die alle anderen Anwender zugreifen.

5.3 Programmierer der Antriebsreglerfunktionen

Der Programmierer der Antriebsreglerfunktionen erstellt die Funktionsblöcke für die Antriebsreglerplattform. Seine Aufgabe ist es, Treiber für Hardwaresubmodule, spezielle Funktionen, wie z. B. Filter oder Funktionsgeneratoren oder Reglerstrukturen als Funktionsbausteine für den FPGA zu entwickeln. Er nutzt dabei die vorhandene FPGA-Entwicklungsumgebung und verwendet unterstützend für die Entwicklung und Simulation Mathematik- und Simulationswerkzeuge wie z. B. Matlab/Simulink. Realisiert er Funktionen auf Mikrocontroller Funktionsblöcken, benötigt er darüber hinaus noch die dem Mikrocontroller zugehörige *Toolchain* bestehend aus Compiler, Linker, Assembler, sowie evtl. einem Debugger.

Die Funktionsblöcke sowie deren Zeitverhalten (Ausführungstakt, Ausführungsdauer) und eine ausführliche Dokumentation werden vom Programmierer in einer Funktionsbibliothek hinterlegt, aus der sich der Regelungs- und Prozesstechniker für die Konfiguration eines Gesamtsystems bedient.

5.4 Regelungs- und Prozesstechniker

Der Regelungs- und Prozesstechniker stellt aus einer Bibliothek vorhandener Funktionsbausteine die Reglerstrukturen, Filter und sonstiger Funktionalitäten ein Gesamtsystem zusammen, das auf einen speziellen Anwendungsfall, wie z. B. die Antriebsregelung einer Maschine, mit einer übergeordneten Prozessregelung zugeschnitten ist.

Er benötigt dazu ein spezielles Entwicklungswerkzeug, das ihn bei der Konfiguration des Gesamtsystems unterstützt und die folgenden Aufgaben übernimmt:

- Bereitstellung der vorhandenen Funktionsblöcke in einer Bibliothek.
- Unterstützung des Regelungstechnikers bei der Verbindung und Konfiguration der Funktionsblöcke zu einem Gesamtsystem.
- Unterstützung der Softwareentwicklung der DSP- und Mikrocontroller Funktionsblöcke: Erzeugung von Templates mit den konfigurierten Ein- und Ausgängen, Aufruf von jeweiligen Entwicklungsumgebungen und Compilern.
- Überprüfung, ob die Verbindungen zwischen den Ein- und Ausgängen der Funktionsblöcke gültig sind (Taktraten, Bitbreiten, Datenformate).
- Erstellung von Ausführungszeitplänen für den Scheduler nach Vorgabe von zu optimierenden Signalpfaden.
- Automatische Erzeugung einer übergeordnete VHDL-Datei, die alle Komponenten der Infrastruktur der Antriebsreglerplattform sowie alle ausgewählten Funktionsblöcke zu einem Gesamtsystem verbindet.

Abschließend werden die erzeugten Dateien mit der FPGA-Entwicklungsumgebung synthetisiert und auf den FPGA geladen.

Der Regelungs- und Prozesstechniker stellt dem Elektrokonstrukteur und dem Inbetriebnehmer eine Beschreibung des konfigurierten Systems, sowie der zur Verfügung stehenden Parametern zur Verfügung.

5.5 Elektrokonstrukteur

Der Elektrokonstrukteur passt einen nach seinen Vorgaben fertig konfigurierten Antriebsregler an eine spezielle Maschine an. Im Gegensatz zum Regelungstechniker greift er nicht in die vorhandenen Reglerstrukturen ein, sondern unternimmt nur geringe Modifikationen, wie z. B. den Austausch eines Gebereingangs oder das Update vorhandener Funktionsblöcke. Er kann sowohl die Hardwaresubmodule als auch Funktionsblöcke auf dem FPGA austauschen oder konfigurieren.

Wie auch der Regelungs- und Prozesstechniker nutzt er dazu, das Konfigurationswerkzeug um Anpassungen im Antriebsregler vornehmen zu können. Anschließend benötigt er die FPGA-Entwicklungsumgebung, um aus dem veränderten VHDL-Code die FPGA-Software zu erzeugen und auf den FPGA zu laden.

5.6 Inbetriebnehmer

Der Inbetriebnehmer passt den Antriebsregler an die Anforderungen des laufenden Betriebs an. Er parametriert die Regler so, dass sie für den jeweiligen Anwendungsfall die optimale Einstellung haben. Des Weiteren führt er Diagnosen durch, um Probleme oder Fehler, die während des Betriebs auftreten, lokalisieren zu können. Die Parameter und Funktionen, auf die der er zugreift, können sowohl von Funktionsblöcken als auch von der Software der Mikrocontrollerfunktionsblöcke bereitgestellt werden.

Er benötigt ein Werkzeug, mit dem er einerseits auf die für ihn relevanten Parameter zugreifen, und andererseits Diagnosefunktionalitäten bedienen kann, wie z. B. Funktionsgeneratoren, Messwertaufzeichnung oder Fehlerspeicher.

Je nach Bedienkonzept der gesamten Maschine kann die Inbetriebnahme und Diagnose antriebsreglerintern oder extern von einer übergeordneten Steuerung aus erfolgen.

5.7 Maschinenbediener

Die Offene Antriebsreglerplattform ist einer Steuerung untergeordnet, die die Benutzerschnittstelle zwischen Maschinenbediener und Maschine bereitstellt. Der Maschinenbediener hat keinen direkten Zugriff auf die Offene Antriebsreglerplattform, sondern er erhält die für ihn relevanten Informationen, wie beispielsweise Status- oder Fehlermeldungen, ausschließlich von der Steuerung.

5.8 Komponenten der Entwicklungsumgebung

Dadurch, dass die Antriebsreglerplattform eine durchgängige Offenheit in allen Ebenen von der Hardwareentwicklung bis zur Inbetriebnahme bietet, ist die Entwicklungsumgebung sehr umfangreich und muss sehr viele verschiedene Funktionen bieten. Es ist daher sinnvoll, zunächst auf vorhandene Entwicklungswerkzeuge zurückzugreifen und die durchgängige Entwicklung durch eine klare Schnittstellenfestlegung und zusätzliche Hilfswerkzeuge, wie z. B. die Ausführungszeitplanung zu unterstützen. Die vorhandenen Entwicklungswerkzeuge decken drei Bereiche der Antriebsreglerplattform ab:

- Der Bereich der **Hardware** umfasst sämtliche Entwicklungs- und Simulationswerkzeuge, die für die Entwicklung von Hardware benötigt werden.

- Der Bereich der **programmierbaren Logik** umfasst Entwicklungs- und Simulationswerkzeuge für FPGAs sowie architekturspezifische Entwicklungswerkzeuge, z. B. einen Ausführungszeitplaner.

- Der dritte Bereich umfasst die **Mikrocontrollerprogrammierung** und stellt die entsprechenden Compiler und Debugger zur Verfügung, die für die Programmierung in Hochsprachen benötigt werden.

Der Vorteil in der Nutzung vorhandener Entwicklungswerkzeuge liegt darin, dass dadurch für die Konfiguration der FPGAs auf herstellereigenen und entsprechend optimierten Entwicklungsumgebungen zurückgegriffen werden kann. Auch für die Programmierung der Mikrocontroller können architekturabhängigen Toolchains genutzt werden. Die Entwickler arbeiten in den gewohnten Entwicklungsumgebungen, sodass eine Einarbeitungszeit entfällt.

Neben diesen Entwicklungswerkzeugen werden übergeordnete Entwicklungswerkzeuge benötigt, die das Zusammenspiel dieser drei Bereiche koordinieren, zweitens Schnittstellen zwischen den Anwendergruppen bereitstellen und drittens die speziellen Funktionen der Antriebsregleplattform zur Verfügung stellen.

Die Durchgängigkeit der Entwicklung wird durch die klare Definition der Schnittstellen zwischen den Anwendergruppen gewährleistet. Zwischen der Hardware- und FPGA-Ebene, sowie zwischen der Antriebsreglerplattform und den Funktionsblöcken kann dies durch ein übergeordnetes Entwicklungswerkzeug erreicht werden. Dieses Entwicklungswerkzeug unterstützt die Entwickler bei der Dokumentation der von ihnen bereitgestellten Schnittstellen und Funktionen, damit diese von den im Entwicklungsablauf nachfolgenden Entwicklern genutzt werden können.

Die Konfiguration des Antriebsreglers aus Funktionsblöcken lässt sich nur teilweise mit den Entwicklungswerkzeugen von FPGA und Mikrocontroller erstellen. Um es für die Anwendergruppen handhabbar zu machen, wird ein übergeordnetes Konfigurationswerkzeug benötigt, das die folgenden Aufgaben übernimmt:

- **Funktionsbibliothek:** Die Funktionsbibliothek enthält die Funktionsblöcke als FPGA-Programmcode, eine Beschreibung der Funktionsblöcke, ihrer Parameter und ihrer Ein- und Ausgänge sowie eine Beschreibung des Zeitverhaltens.

- **Konfiguration des Antriebsreglers:** Die Funktionen werden aus der Bibliothek von Regelungstechniker ausgewählt und zu einem Gesamtsystem verbunden. Das Konfigurationswerkzeug prüft dabei die Verbindungen zwischen den Funktionsblöcken. Aus der Konfiguration wird automatisch der Programmcode für den FPGA erzeugt.

- **Konfiguration und Programmierung der Mikrocontrollerfunktionsblöcke:** Die Ein- und Ausgänge der Mikrocontrollerfunktionsblöcke werden konfiguriert. Für den Programmierer werden Quellcode-Vorlagen erzeugt, in denen die Ein- und Ausgänge als Variablen enthalten sind. Der fertige Programmcode wird kompiliert und in den Funktionsblock bzw. in den ihm zugeordneten Speicher geladen.

- **Zeitplanerstellung:** Aus der Konfiguration des Antriebsreglers, den hinterlegten Informationen über das Zeitverhalten der Funktionsblöcke, sowie den vom Regelungs- und Prozesstechniker vorgegebenen Optimierungskriterien wird ein Zeitplan erstellt.

Das Zusammenspiel des übergeordneten Konfigurationswerkzeugs mit den Entwicklungswerkzeugen für Mikrocontroller und FPGAs ist in Bild 5-2 dargestellt.

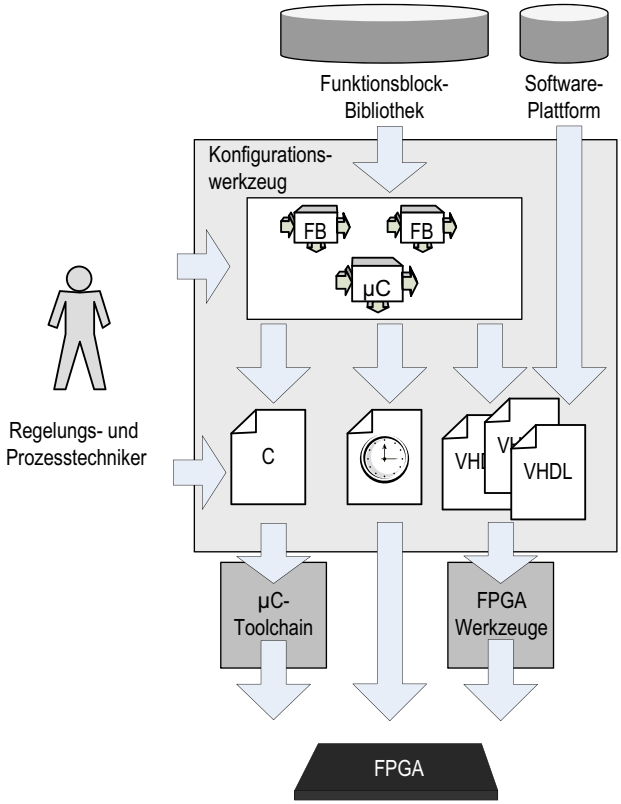

Bild 5-2: Konfiguration eines Antriebsreglers mit dem Konfigurationswerkzeug.

6 Realisierung einer Antriebsregelung

In diesem Kapitel wird die Realisierung des vorgestellten Konzepts beschrieben, mit der die Funktionsfähigkeit und Praxistauglichkeit nachgewiesen wird. Die hier beschriebene Realisierung der Offenen Antriebsreglerplattform wurde im Rahmen des DFG-Projekts „Entwurf und Implementierung einer Offenen Antriebsreglerplattform als Ausgangsbasis für die Umsetzung neuer Technologien“ entwickelt und umgesetzt. Die Offenheit dieser Plattform wurde in einem Projekt der Stiftung Industrieforschung genutzt, um eine neuartige bidirektionale Kommunikation zwischen Antrieb und Steuerung herzustellen. Die bidirektionale Kommunikation dient dazu, Daten aus dem Prozess wie z. B. die Schnittkraft am Fräswerkzeug oder die Schwingungen im Antrieb an die übergeordnete NC-Steuerung zurückzuliefern. Diese Daten werden mit dem NC-Programm verknüpft und können für eine weitere Auswertung, beispielsweise zur Anpassung des programmierten Vorschubs, verwendet werden /74/. In einem an das DFG-geförderte Projekt anknüpfende, durch das AiF geförderte Projekt „Offene Architektur zur achsübergreifenden Antriebsregelung“ /75/ und daran anknüpfende Projekte wird die Hardware weiterentwickelt, um die bisher forschungsorientierte Antriebsreglerplattform zu überarbeiten und für den industriellen Einsatz tauglich zu machen.

Um die Neuerung gegenüber herkömmlicher Antriebstechnik, sowie die Leistungsfähigkeit der FPGA-basierten Plattform zu verdeutlichen, wurden die Verfahren *Schnelle Stromregelung*, *Oversampling von Gebersignalen* und eine diskrete Fourier Transformation (DFT) zur Frequenzanalyse von antriebsinternen Signalen als Hardwaresubmodule bzw. Funktionsblöcke umgesetzt.

6.1 Aufbau der Hardwareplattform

Die Hardwareplattform, die im Rahmen des DFG-Projekts eingesetzt wurde, besteht aus einem zentralen FPGA und einer Reihe von Hardwarekomponenten, die für den Aufbau eines Demonstrators, mit dem ein vollständiger Antriebsstrang einer Maschine nachgestellt werden kann. Dazu gehören:

- Nichtflüchtige Speicher, auf denen die FPGA-Software abgespeichert wird, sowie Konfigurationen und Parameter, die nach einem Wiedereinschalten der Hardware erhalten bleiben sollen.
- Schneller, flüchtiger Speicher, der von Mikrocontrollerfunktionsblöcken, sowie von beispielsweise Diagnose-Funktionsblöcke zur Datenerfassung genutzt werden kann.
- Eine (JTAG-)Programmierschnittstelle zur Programmierung des FPGAs und zum Beschreiben des nichtflüchtigen Speichers.
- Eine serielle Schnittstelle als einfachste Form einer Bediener-Schnittstelle.

Einen wesentlichen Teil der Hardwareplattform bilden vier Steckplätze für Hardwaresubmodule, die die Offenheit gegenüber Hardwareerweiterungen und Schnittstellen zu verschiedenen Gebersystemen, Feldbussen, Sensoren und Aktoren herstellen. Die vier Steckplätze liefern die gängigen Versorgungsspannungen für die Submodule. Des Weiteren bieten sie 40 frei konfigurierbare Datenleitungen, die direkt mit dem FPGA verbunden sind und von diesem beliebig angesteuert werden können.

Die Hardwareplattform wurde als ISA-Steckkarte ausgeführt, sodass der Betrieb direkt innerhalb einer PC-basierten Steuerung möglich ist. Die Kommunikation zwischen Steuerung und Antrieb kann hier für die direkte Kommunikation ohne einen zwischengeschalteten Feldbus erfolgen.

Zu Forschungszwecken weist die Hardwareplattform zudem eine Schnittstelle zum marktgängigen Rapid-Prototyping System dSpace auf. Diese Schnittstelle schafft eine einfache Möglichkeit, Versuchsstände, die mit der Offenen Antriebsreglerplattform betrieben werden, mit einem dSpace-System ansteuern zu können. Auf diesem Wege können beispielsweise neuartige achsübergreifende Regler mit geringem Aufwand umgesetzt und ausgetestet werden.

6.2 Aufbau von Hardwaresubmodulen

Für die Realisierung einer funktionierenden Antriebsreglerplattform werden eine Reihe von Hardwaresubmodulen benötigt, die die wesentlichen Schnittstellen eines Antriebsreglers bereitstellen. Das umfasst zum einen die Sensorik, bestehend aus

Strommessung, Gebererfassung und zum anderen die Aktorik, hier die Ansteuerung der Leistungshalbleiter einer Leistungsendstufe für Servomotoren. Dazu kommt die Anbindung an eine übergeordnete Steuerung über einen Feldbus.

Die Submodule nutzen 40 Datenleitungen für die zyklische Kommunikation mit der Antriebsreglerplattform, wobei jedes Submodul auf diesen Datenleitungen unterschiedliche Kommunikationsprotokolle nutzt.

Um plattformseitig die Module in ihren Steckplätzen identifizieren zu können, übertragen sie auf vier Konfigurationsleitungen über ein einheitliches serielles Übertragungsprotokoll eine eindeutige Modulkennung. Über diese Konfigurationsleitungen können auch weitergehende Informationen der Submodule, z. B. Varianten oder Versionsnummern, abgerufen werden.

6.2.1 Submodul zur Ansteuerung eines Umrichters

Die stromreglernahe Hardware wurde auf einem Submodul zusammengefasst. Auf diesem Submodul befinden sich drei zweikanalige AD-Wandler, die synchron die Analogwerte von jeweils zwei Strommessgliedern für insgesamt drei Motoren erfassen können. Für jeden Motor werden zwei der drei Phasenströme gemessen. Der dritte Phasenstrom kann bei Motoren ohne angeschlossenen Nullleiter aus zwei Phasenströmen berechnet werden. Des Weiteren befinden sich digitale Ein- und Ausgänge auf der Hardware, über die die Schaltsignale für die Leistungstransistoren ausgegeben und von Sigma-Delta-Wandlern gemessene Ströme eingelesen werden können. Sigma-Delta-Wandler sind eine besondere Art von Analog-Digital-Wandlern, die den Analogwert nicht als digitalen Wert mit festgelegter Bitbreite bereitstellen, sondern den Analogwert als seriellen Datenstrom, ähnlich einem PWM-moduliertem Signal ausgeben /76/. Auch die digitalen Ein- und Ausgänge sind auf die Ansteuerung von drei Motoren ausgelegt. Zusätzlich befindet sich auf dem Submodul ein FPGA, der einerseits zur Kommunikation zwischen Submodul und Hardwareplattform dient, aber, falls aus Platzgründen erforderlich, auch Teile der Stromregelung übernehmen kann. Hier wurde der Funktionsblock *Schnelle Stromregelung* auf den FPGA des Submoduls ausgelagert.

6.2.2 Submodul zur Auswertung von Messsystemen

Das Submodul für die Erfassung von Lagegebern wurde auf zwei Sinus-Cosinus-Geber /77/, sowie zwei Ferraris-Sensoren /78,79/ ausgelegt. Die Beschränkung auf zwei Wegmesssysteme resultiert aus der Größe des zur Verfügung stehenden FPGAs, auf dem sich die Geberauswertung nicht für drei Wegmesssysteme realisieren ließ. Auch auf diesem Modul befindet sich ein FPGA für die Kommunikation zwischen Submodul und Hardwareplattform. Der FPGA bietet ausreichend Ressourcen, um neben der Kommunikation auch Teile der Geberauswertung zu übernehmen. Daher wurde der Funktionsblock *Lageoversampling* auf dem FPGA des Submoduls ausgelagert.

6.2.3 Submodul für die Feldbusanbindung

Auf einem weiteren Submodul wurde ein Feldbus für die Anbindung einer übergeordneten Steuerung realisiert. Es wird der Feldbus Sercos II /80,81/ eingesetzt, der über Lichtwellenleitern in einer Ring-Topologie mit der Steuerung und anderen Busteilnehmern verbunden ist. Die eigentliche Buskommunikation wird von einem speziellen Sercos-ASIC bereitgestellt. Ein überlagerter Sercos-Stack wird auf einem Mikrocontroller, der sich auf dem Submodul befindet, ausgeführt. Die Sende- und Empfangsdaten werden über einen Dualported RAM, einem Speicherbaustein mit zwei voneinander unabhängigen Daten- und Adressbussen, mit dem FPGA auf der Hardware-plattform ausgetauscht.

6.3 Aufbau einer Infrastruktur im Logikbaustein

Auf dem FPGA der Hardwareplattform wird eine Infrastruktur benötigt, die die Ausführung der Funktionsblöcke koordiniert, sowie die Kommunikation zwischen den Funktionsblöcken herstellt. Des Weiteren müssen die Hardwaresubmodule mit den je-weiligen Treibern verbunden werden.

6.3.1 Hardwareverwaltung für die Hardwaresubmodule

Jedes Hardwaresubmodul kann in einen beliebigen der vier Steckplätze eingesteckt werden. Jeder Steckplatz bietet 40 Datenleitungen, deren Übertragungsrichtung nicht festgelegt ist. Um Kurzschlüsse durch Zusammenschalten zweier Ausgänge zu vermeiden, muss jedes Submodul identifiziert werden, bevor die Datenleitungen auf dem FPGA aktiviert werden. Ist ein Submodul identifiziert worden, wird es automatisch mit dem passenden Treiberfunktionsblock auf dem FPGA verbunden.

Für die Identifikation gibt es zusätzlich zu den Datenleitungen vier Identifikationsleitungen. Diese sind für alle Submodule einheitlich definiert und werden von der Hardwareverwaltung des FPGAs genutzt, um die Submodule zu erkennen. Zur Identifizierung werden zwischen passiven und aktiven Submodulen unterschieden.

Bei passiven Submodulen wird die Identifikationsnummer durch eine feste Verdrahtung der Identifikationsleitungen konfiguriert. Die erste Identifikationsleitung wird hier auf ‚low' gelegt, während die drei weiteren die ID bilden. Diese passive Identifikation ermöglicht es, sehr einfache, kostengünstige Submodule aufzubauen, beispielsweise um schnell und kostengünstig neue Hardware testen zu können.

Aktive Submodule stellen auf den Identifikationsleitungen eine serielle Verbindung zu einem submoduleigenen Speicher zur Verfügung. In diesem Speicher ist die Identifikationsnummer des Submoduls hinterlegt. Zusätzlich können noch weitere Daten hinterlegt werden. Bei einer Geberkarte können dies Geberdaten sein, bei einem Submodul zur Ansteuerung von Umrichtern die Anzahl der angeschlossenen Motoren oder verschiedene Motordaten. Aktive Submodule können daran erkannt werden, dass die erste Identifikationsleitung dauerhaft auf ‚high' verdrahtet ist.

Die Daten- und Identifikationsleitungen aller Steckplätze für Submodule und alle Treiberfunktionsblöcke sind mit der Hardwareverwaltung verbunden. Beim Einschalten der Antriebsreglerplattform erfragt die Hardwareverwaltung die Identifikationsnummern aller Hardwaresubmodule sowie aller Treiberfunktionsblöcke. Danach verbindet sie die Datenleitungen der Steckplätze mit den Datenleitungen der Treiberfunktionsblöcke und gibt sie frei. So wird gewährleistet, dass die Datenleitungen auf den Steckplätzen immer passend geschaltet werden.

6.3.2 Scheduler mit Ausführungszeitüberwachung

Wie zuvor gezeigt, müssen die Funktionsblöcke auf dem FPGA nach einem vorgegebenen, optimierten Zeitplan ausgeführt werden, um möglichst kurze Signallaufzeiten und damit ein gutes Reglerverhalten zu erreichen. Der auf der Antriebsreglerplattform realisierte Scheduler führt die Funktionsblöcke nach einem festen Zeitplan aus und überwacht die Ausführungszeiten.

Jeder Funktionsblock trägt eine eindeutige Identifikationsnummer. Außerhalb der Antriebsreglerplattform wird vorab eine Tabelle erzeugt, in der für jeden Funktionsblock der Ausführungstakt, die Ausführungsdauer, sowie ein Offset zum Zeitpunkt Null hinterlegt sind. Diese Tabelle wird in einem Speicher des Schedulers bei der Konfiguration der Antriebsreglerplattform abgelegt.

Der Scheduler hat für jeden konfigurierten Funktionsblock einen Triggerausgang und einen Überwachungseingang. Wird die Antriebsreglerplattform eingeschaltet, beginnt der Scheduler mit der Abarbeitung des hinterlegten Ausführungszeitplans. Die Funktionsblöcke bekommen über ihren jeweiligen Triggereingang ein Triggersignal, wenn sie ausgeführt werden sollen. Nach abgeschlossener Ausführung signalisieren sie dem Scheduler über den Überwachungseingang, dass die Ausführung abgeschlossen ist. Der Scheduler kann somit überwachen, ob die Funktionsblöcke, wie vorgesehen ausgeführt werden und auf Fehler mit einer Meldung an eine übergeordnete Steuerung oder der Ausführung alternativer Ausführungszeitpläne reagieren.

6.3.3 Getaktete Kommunikation

Die getaktete Kommunikation wurde zunächst als direkte Verbindung der entsprechenden Funktionsblöcke realisiert. Es wird davon ausgegangen, dass ein Funktionsblock, der nach der Triggerung im vorgesehenen Zeitfenster das Überwachungssignal an den Scheduler zurückmeldet, auch seine Ausgangsignale an den nachfolgenden Funktionsblock korrekt weiterschaltet.

6.3.4 Episodische Kommunikation

Für die episodische Kommunikation, insbesondere für die Parametrierung der einzelnen Funktionsblöcke, wurde der einfache und damit auch ressourcensparend zu realisierende Wishbone-Bus eingesetzt. Der Wishbone-Bus wurde im Rahmen des Opensource-Projekts für FPGA-Software *opencores.net* entwickelt und dient dem Datenaustausch zwischen FPGA-Software-Elementen.

Auf dem FPGA der Antriebsreglerplattform übernimmt ein Mikrocontrollerfunktionsblock, der auch für die Bedienerschnittstelle zuständig ist, die Aufgabe des Wishbone-Masters. Ein Arbitrierer-Funktionsblock verbindet die Datenleitungen des vom Master adressierten Slaves mit dem Mikrocontroller und stellt damit eine Verbindung her. Jeder Funktionsblock, der über episodische Ein- oder Ausgänge verfügt, kann als Wishbone-Slave angesprochen werden. Die Funktionsblöcke stellen ihre Parameter an diesen Ein- und Ausgängen zur Verfügung. Dadurch hat die Bedienerschnittstelle auf dem Mikrocontroller Zugriff auf die Parameter aller Funktionsblöcke.

Bild 6-1 verdeutlicht den Aufbau des Wishbone-Busses auf der Antriebsreglerplattform.

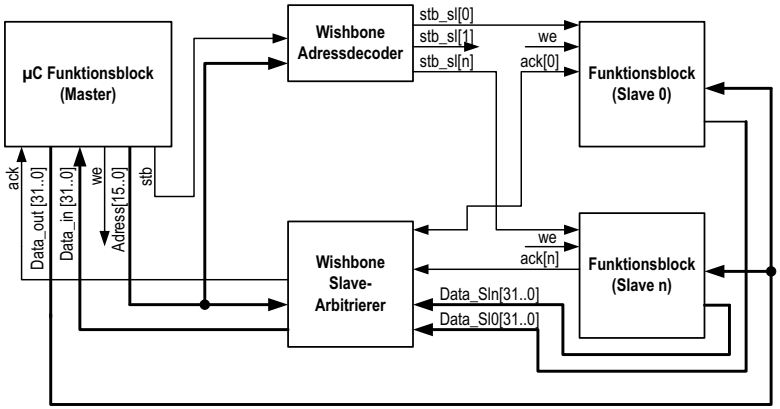

Bild 6-1: Schematische Darstellung einer Wishbone-Busverbindung zwischen einem Mikrocontrollerfunktionsblock als Master und Funktionsblöcken als Slaves.

6.4 Implementierung von Funktionsblöcken

Der eigentliche Antriebsregler wird aus einer Reihe von Funktionsblöcken konfiguriert, die in die zuvor vorgestellte Infrastruktur eingebettet werden. Für die Demonstration der Funktionsfähigkeit der Antriebsreglerplattform wurden die im Folgenden beschriebenen Funktionsblöcke implementiert. Dazu gehören Funktionsblöcke für den in der elektrischen Antriebstechnik üblichen Kaskadenregler, weitere Funktionsblöcke für die Bedienung und Messwerterfassung sowie die Anbindung an eine übergeordnete Steuerung. Aus dieser Bibliothek von Funktionsblöcken wurde ein zweiachsiger Antriebsregler konfiguriert.

6.4.1 Treiberfunktionsblock für die Messsystemauswertung

Die hardwareseitige Sensorschnittstelle wird durch ein Hardwaresubmodul bereitgestellt. Auf der FPGA-Seite wird ein Treiberfunktionsblock, sowie ein Funktionsblock für die Auswertung der Sensorsignale benötigt. Da das Submodul für die Messsystem-auswertung selbst über einen FPGA verfügt, wurde der Funktionsblock für die Auswertung auf dieses Submodul ausgelagert. Der Treiberfunktionsblock stellt die Kommunikation über die 40 Datenleitungen der Hardwareschnittstelle mit dem Submodul her. Auf der FPGA-Plattform liefert er die bereits ausgewertete und gefilterte Lage- und Geschwindigkeitsinformation von zwei Gebern und stellt das gefilterte Signal von zwei Ferraris-Beschleunigungssensoren zur Verfügung.

Der Vorteil der Auslagerung von Funktionsblöcken liegt in der effizienteren Nutzung der vorhandenen Hardware. Das Konzept der Offenen Antriebsplattform erlaubt das beliebige Verschieben von Funktionalität zwischen dem FPGA der Plattform und den Submodulen. Durch die Verlagerung von Funktionsblöcken auf die Submodule wird Platz auf dem FPGA der Plattform eingespart, der für die Neuentwicklungen zukünftiger Projekte zur Verfügung steht.

6.4.2 Treiberfunktionsblock für die Ansteuerung eines Umrichters

Auf dem FPGA wird, gleichsam wie für die Messsystemauswertung, ein Treiber bereitgestellt, der die Kommunikation mit dem Hardwaresubmodul für die Ansteuerung

von Umrichtern herstellt. Da das Submodul selbst über einen ausreichend großen FPGA verfügt, wurden sowohl die Funktionsblöcke für die Strommessung, in denen das Stromoversampling verwendet wird, sowie die Funktionsblöcke für die Stromreglung auf diesen FPGA ausgelagert.

Der Treiberfunktionsblock stellt die gemessenen Istströme von bis zu drei Umrichtern zur Verfügung und überträgt die momentenbildenden Ströme als Sollwerte an das Submodul.

6.4.3 Treiberfunktionsblock für die Feldbusschnittstelle

Das Hardwaresubmodul der Feldbusschnittstelle wird seitens des FPGAs von einem Treiber angesteuert, der im Feldbustakt die Sollwerte und Parameter liest, sowie Istwerte und Statussignale für den Transport zur Steuerung zurückschreibt.

Aufgrund der niedrigen Feldbustakte von 1 ms konnte der Treiber für die Feldbuskarte als Funktionsblock der dritten Offenheitsebene auf einem Mikrocontrollerfunktionsblock implementiert werden. Dieser Funktionsblock greift über Adress- und Datenbusleitungen direkt auf den Shared Memory des Submoduls zu. Der Treiber liest im Pollingverfahren ein Flag im Shared Memory aus, das signalisiert, ob neue Daten zur Verfügung stehen und zu sendende Daten abgelegt werden können.

Die für die Antriebsreglung relevanten, von extern kommenden zyklischen Daten werden an eigenen Ausgängen des Mikrocontrollers der Antriebsreglerplattform zur Verfügung gestellt. Zyklisch zu sendende Daten werden an eigenen Eingängen vom Treiber eingelesen und über den Shared Memory des Submoduls weitergereicht. Azyklische Daten, wie z. B. Reglerparameter, werden über den Wishbone-Bus zu den betreffenden Funktionsblöcken transportiert oder ausgelesen.

6.4.4 Funktionsblock Geschwindigkeitsregelung

Der Geschwindigkeitsregler ist als PI-Regler in einem eigenständigen Funktionsblock implementiert. Die Eingänge dieses Blocks ist die Soll- und die Istgeschwindigkeit sowie der Ausgang die Sollbeschleunigung. Der Ausgang des PI-Reglers wird durch

einen parametrierbaren Begrenzer eingeschränkt. Zusätzlich wird der I-Anteil des Reglers nach unten und oben beschränkt, um ein Wind-Up /82/ zu verhindern.

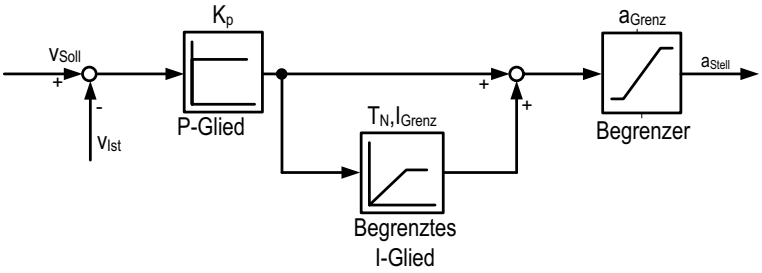

Bild 6-2: PI-Regler mit begrenztem I-Anteil und begrenztem Ausgang.

Alle Parameter des Drehzahlreglers und seiner Begrenzungen können über den Wishbone-Bus eingestellt werden.

6.4.5 Funktionsblock Lageregelung

Die Lageregelung wird auf der Antriebsreglerplattform in einem eigenständigen Funktionsblock durchgeführt. Realisiert wurde ein einfacher P-Regler, der die Soll- und Istlage voneinander subtrahiert und das Ergebnis mit dem Verstärkungsfaktor Kv multipliziert. Der Ausgang des Lagereglers wird mit einem Begrenzer auf einen parametrierbaren Wertebereich eingeschränkt. Der Verstärkungsfaktor kann dabei über den Wishbone-Bus parametriert werden.

6.4.6 Funktionsblock Benutzerschnittstelle

Die Benutzerschnittstelle bietet dem Benutzer die Möglichkeit, Funktionsblöcke der Antriebsregler zu parametrieren sowie manuell Funktionen zu triggern, Messwerte abzufragen und Freigabesignale zu setzen. Die Echtzeitanforderungen sind hier nur gering.

Aufgrund der für eine Benutzerschnittstelle typischen nichtechtzeitkritischen, nicht getakteten, aber stark verzweigten Abläufe, wie z. B. die in verschiedene Untermenüs verzweigende Darstellung verschiedener Parametergruppen, wurde ein Mikrocontroller-funktionsblock implementiert, auf dem die Benutzerschnittstelle als Programm in der dritten Ebene der Offenheit realisiert ist.

Der Mikrocontroller verfügt über eine serielle Schnittstelle, über die mithilfe eines Terminalprogramms auf die Benutzeroberfläche zugegriffen werden kann.

6.4.7 Funktionsblock Messung

Die Offenheit der Antriebsreglerarchitektur ermöglicht den Zugriff auf beliebige Daten in beliebigen Takten. Es wurde ein Funktionsblock implementiert, der diese Offenheit nutzt und beliebige Signale auf dem Antriebsregler mitprotokollieren kann.

Der Funktionsblock ist an die Hardwareschnittstelle eines externen SRAM Speichers angeschlossen. Er nimmt durch den Scheduler getriggert mehrere Daten zeitgleich auf und schreibt sie sequenziell in den SRAM. Wird keine Messung aufgenommen, werden die Steuer- und Datenleitungen des SRAM durch den Funktionsblock geschleift, sodass der Speicher von einem anderen Funktionsblock für die Weiterverarbeitung ausgelesen werden kann, um die Messdaten beispielsweise über den Feldbus an eine Steuerung zu übertragen.

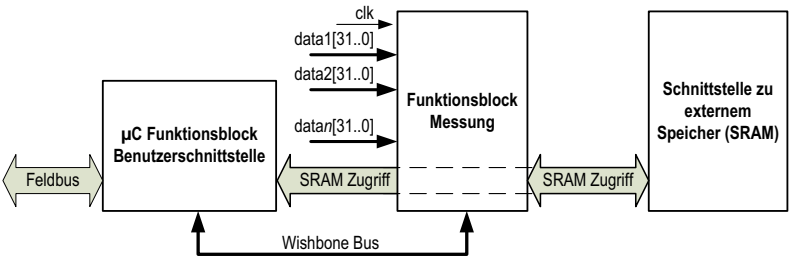

Bild 6-3: Schematische Darstellung der Verbindung zwischen Benutzerschnittstelle, Funktionsblock „Messung“ und einem externen Speicher.

Die Zykluszeiten für die Datenaufnahme sind abhängig von der Anzahl der Daten, da diese nacheinander in den SRAM geschrieben werden müssen, bevor die nächste Messung starten kann.

6.4.8 Funktionsblock Diskrete Fourier Transformation

Die Diskrete Fourier Transformation (DFT) dient der Berechnung des Frequenzspektrums eines Signals innerhalb eines Zeitabschnitts /83,84/. Der Algorithmus arbeitet in mehreren Stufen auf einem Datensatz von *M* Messwerten. In jeder Stufe werden jeweils 2 Messwerte nach den Formeln (6.1) verrechnet und durch die zwei resultierenden Werte ersetzt. Der Faktor W_M^n wird als Twiddle-Faktor bezeichnet. Er ist abhängig von der Anzahl der Messwerte sowie vom jeweiligen Berechnungsschritt. Diese sogenannte Butterfly-Operation wird in mehreren Stufen unter Verwendung unterschiedlicher Kombinationen auf den Datensatz angewendet.

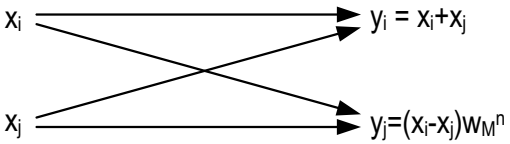

Bild 6-4: Butterfly-Operation.

$$
\begin{aligned}
y_i &= x_i + x_j \\
y_j &= (x_i - x_j) \cdot w_M{}^n \\
w_M{}^n &= e^{n(-2\pi i)/M}
\end{aligned}
\qquad (6.1)
$$

Aufgrund der Struktur des Algorithmus können die Butterfly-Operationen parallel berechnet werden. Dazu wird das Netzwerk von Butterfly-Operationen, wie in Bild 6-5 für den Fall *M=4* gezeigt, als eigenständige Logikschaltung auf dem FPGA implementiert.

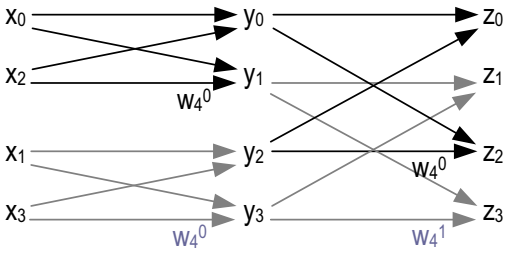

Bild 6-5: Netzwerk von Butterfly-Operationen für eine DFT mit 4 Messwerten.

Für die parallele Bearbeitung der Daten müssen diese dem Schaltnetzwerk zeitgleich zur Verfügung stehen, und können daher nicht in einem externen Speicher, der nur wortweise ausgelesen werden kann, bereitgestellt werden, sondern müssen in eigenen Speicherzellen auf dem FPGA zur Verfügung stehen. Daher ist die parallele Berechnung der DFT einerseits sehr schnell aber andererseits belegt sie große Mengen an logischen Elementen.

Alternativ lassen sich die Butterfly-Operationen auch mit einer einzelnen Schaltung ausführen, die schrittweise die zu verrechnenden Daten geliefert bekommt. Der Butterfly-Operator liest die Daten sowie den aktuellen Twiddle-Faktor von Speicheradressen aus, die für jeden Schritt neu erzeugt werden müssen, und schreibt die Ergebnisse an ebenfalls für jeden Schritt erzeugte Adressen. Sowohl die Daten als auch die Twiddle-Faktoren können in externen Speichern außerhalb des FPGAs ausgelagert werden, um Ressourcen aus dem FPGA zu sparen, oder hochauflösende DFTs für lange Zeiträume zu berechnen. Auf dem FPGA wird nur ein Butterfly-Operator und Adressgenerator implementiert, wodurch diese Variante der DFT erheblich weniger Ressourcen des FPGAs belegt, als die parallele Berechnung.

Je nach zur Verfügung stehenden Ressourcen auf dem FPGA und Rechengeschwindigkeitsanforderungen können auch Kombinationen von parallel und sequenziell ausgeführten Rechenschritten implementiert werden.

Für die Umsetzung auf der Offenen Antriebsreglerplattform wurde in Hinblick auf die Verwendung für die Maschinendiagnose eine rein sequenzielle Berechnung der DFT für eine parametrierbare Anzahl von Messwerten gewählt. Da die Eingangsdaten bei den

meisten Anwendungsfällen Strom- und Geschwindigkeits- sowie Lageistwerte sind, die in den jeweiligen Reglertakten zyklisch aufgenommen werden, ist die sequenzielle Verarbeitung im jeweiligen Takt ausreichend. Diese Daten werden in einen externen Speicher geschrieben. Sobald ein Messwertpaar im Speicher abgelegt ist, wird die jeweilige Butterfly-Operation ausgeführt. Dabei wird die gleiche Logik für jede Butterfly-Operation wieder verwendet. Die berechneten Zwischenergebnisse werden ebenfalls im externen Speicher abgelegt.

6.5 Entwicklungswerkzeuge

Im Rahmen eines studentischen Software Projekts wurde ein grafisches Konfigurationswerkzeug entwickelt, das den Anwender bei der Konfiguration eines Antriebsreglers aus Funktionsblöcken unterstützt.

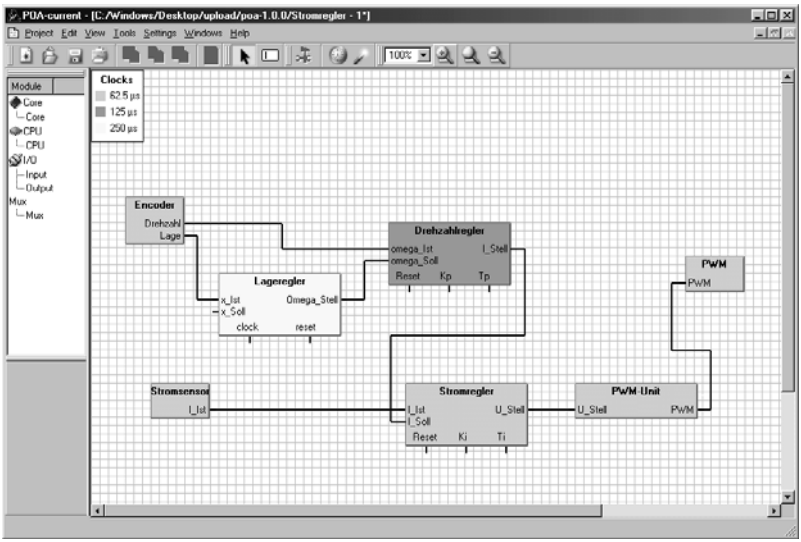

Bild 6-6: Screenshot des grafischen Konfigurationswerkzeugs.

Für die **Erstellung eines Antriebsreglers** bietet es dem Anwender:

- die grafische Konfiguration des Antriebsreglers aus den Funktionsblöcken einer Bibliothek.
- die Verknüpfung der Ein- und Ausgänge der Funktionsblöcke untereinander.
- die Prüfung der Bitbreiten der Verbindungen zwischen den Funktionsblöcken.

Für die **Erstellung von Zeitplänen** kann der Anwender die Ausführungstakte der Funktionsblöcke parametrieren. Der Ausführungszeitplan wird vom Entwicklungswerkzeug automatische unter Berücksichtigung der parametrierten Ausführungstakte, der Datenverbindungen, sowie den Optimierungsvorgaben des Anwenders berechnet.

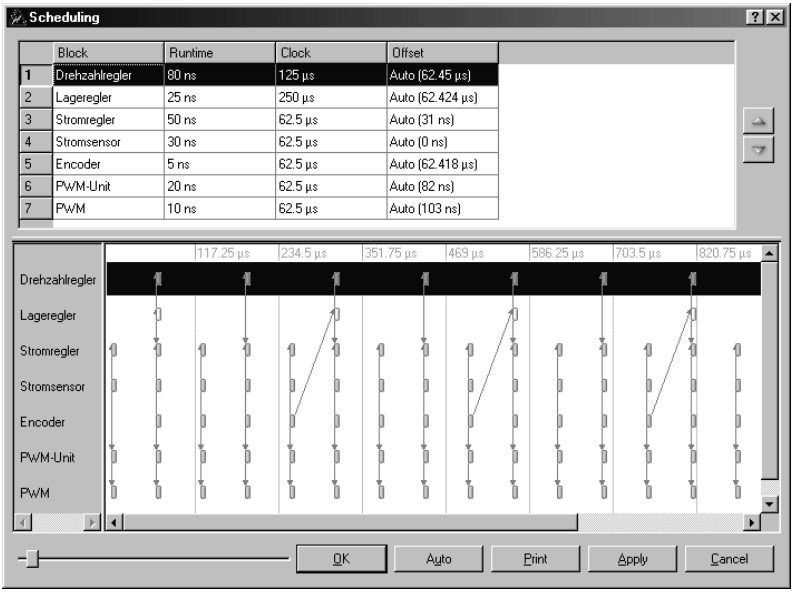

Bild 6-7: Screenshot der automatischen Zeitplanerstellung.

Des Weiteren unterstützt das Entwicklungswerkzeug die **Programmierung der Mikrocontrollerfunktionsblöcke** durch:

- die Verknüpfung von Mikrocontrollerfunktionsblöcken mit einer mikrocontroller-spezifischen Entwicklungsumgebung (Editor, Kompiler).
- die Erstellung von Quellcode-Vorlagen für die Mikrocontrollerentwicklung.
- das Herunterladen von Mikrocontroller-Code in den Speicher des Mikro-controllers.

Die Bedienung des grafischen Konfigurationswerkzeugs orientiert sich an markt-gängigen Mathematikwerkzeugen, wie z. B. Matlab/Simulink. Die Funktionsblöcke können vom Anwender mit der Maus auf einem Arbeitsblatt platziert werden, das das Antriebsreglersystem darstellt. Die Ein- und Ausgänge der Funktionsblöcke werden ebenfalls mithilfe der Maus verbunden. Die Parametrierung der Funktionsblöcke erfolgt über Kontextmenüs, in denen bestimmte Parameter, sowie der Ausführungstakt eingestellt werden können.

6.6 Ergebnisse

Die umgesetzten Submodule und Funktionsblöcke wurden erfolgreich an einem Versuchsstand in Betrieb genommen. Der Versuchsstand besteht aus einem Kreuztisch, wie in Bild 6-8 gezeigt, der von zwei hochdynamischen Lineardirektantrieben der Firma IDAM angetrieben wird. Die Leistungsversorgung erfolgt über zwei modifizierte Siemens 611U-Umrichter, die eine PWM-Schnittstelle zu ihren IGBTs, sowie eine digitale Schnittstelle zu den umrichterinternen Strommessgliedern zur Verfügung stellen. An den Linearachsen sind zwei hochauflösende Sinus-Cosinus-Geber, sowie Ferraris-Sensoren montiert.

Auf der Antriebsreglerplattform wurde aus Funktionsblöcken eine Lage- sowie Drehzahlregelung, die für die Stromregelung notwendige Park-Clarke-Transformation, eine Messdatenerfassung, eine digitale Fourier Transformation, sowie eine Benutzerschnittstelle auf einem Mikrocontroller-Funktionsblock konfiguriert. Die Funktionsblöcke zur Geberauswertung, sowie zur Stromregelung wurden aufgrund ihres hohen Bedarfs an Logischen Elementen auf die jeweiligen, mit FPGAs ausgestatteten, Hardware-Submodule ausgelagert. Die Geberauswertung für zwei Wegmesssysteme und zwei

Ferraris Sensoren benötigt ca. 3500 logische Elemente. Die Stromregelung benötigt für zwei Achsen ohne Park-Clarke Transformation ca. 1600 logische Elemente. Die FPGAs der Submodule und der Antriebsreglerplattform werden jeweils mit 40 MHz getaktet. Die Antriebsreglerplattform benötigt auf dem FPGA in einer Grundversion, die die notwendige Infrastruktur für die Funktionsblöcke, sowie grundlegenden Funktionsblöcke für einen zweiachsigen Geschwindigkeits- und Stromregler beinhaltet, ca. 10000 logische Elemente.

Das für die Geberauswertung implementierte Oversamplingverfahren erreicht auf der Hardware eine Abtastrate von 1 MHz und eine konstante Filterlänge von 128 Messwerten, wodurch die in /33/ beschriebene Verbesserung des Lagesignals und insbesondere des Geschwindigkeitssignals erreicht werden konnte. Die Abweichung des Geschwindigkeitssignals von der tatsächlichen Geschwindigkeit ist mit Oversampling ca. sechsmal niedriger als ohne Oversampling. Dies führt zu einer höheren Laufruhe des Antriebs und zu höher einstellbaren Parametern und damit zu einer höheren Bandbreite des Drehzahlreglers.

Für die Stromregelung wird das Verfahren der schnellen Stromreglung eingesetzt. Die Stromregelung wird mit einem sehr hohen Regeltakt von 20 µs und einer PWM-Frequenz von 40 kHz betrieben.

Die Sollwerte für die Antriebsreglerplattform werden von einer Beckhoff CNC Steuerung erzeugt und über den Sercos Feldbus in einem Feldbustakt von 1 ms übertragen.

An dem Versuchsaufbau wird nachgewiesen, dass die in dieser Arbeit vorgestellte Antriebsreglerarchitektur mit allen ihren Mechanismen, wie z. B. der Modularisierung einzelner Funktionen, dem Scheduling, der periodischen und episodischen Kommunikation, ordnungsgemäß funktioniert. Des Weiteren wurden, um die Leistungsfähigkeit zu demonstrieren, mit der schnellen Stromregelung und dem Oversampling der Lagegebersignale, zwei Verfahren umgesetzt, die sich nicht oder nur mit zusätzlichem Hardwareaufwand auf DSP-basierten Antriebsreglersystemen umsetzen lassen.

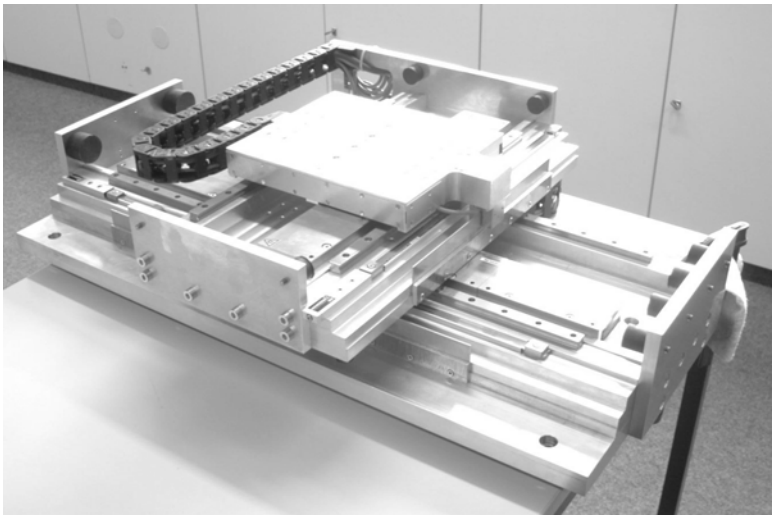

Bild 6-8: Kreuztisch mit zwei hochdynamischen Lineardirektantrieben.

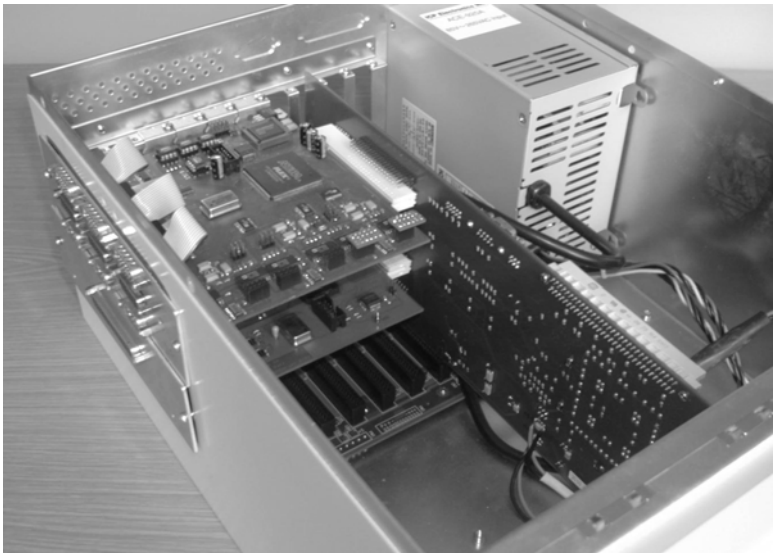

Bild 6-9: Offene Antriebsreglerplattform mit zwei Submodulen.

7 Zusammenfassung und Ausblick

In dieser Arbeit wird ein Konzept für eine Offene Antriebsreglerplattform vorgestellt, mit dem den steigenden Anforderungen an die Antriebstechnik begegnet werden kann. Dies wird durch eine hohe Leistungsfähigkeit sowie eine hohe Flexibilität und Offenheit gegenüber neuen Technologien erreicht. Für den Entwurf der Antriebsreglerplattform wurden die bestehenden Anforderungen und verfügbaren Technologien berücksichtigt, damit ein System geschaffen werden kann, das über den heutigen Stand der Antriebstechnik hinaus Möglichkeiten für die Realisierung neuer Technologien und Regelverfahren bietet. Dies wird durch den Einsatz einer geeigneten, auf programmierbarer Logik basierenden Hardwareplattform, sowie durch eine offene, modulare Softwarestruktur ermöglicht. Sie bietet den verschiedenen Anwendergruppen den notwendigen Zugang zur Antriebsreglerplattform, der benötigt wird, um das System an die anwenderspezifischen Anforderungen anzupassen.

Das Konzept umfasst den Aufbau der Antriebsreglerplattform, der sich in drei Ebenen gliedert. Diese Ebenen werden von der Hardware, einer programmierbare Logik sowie den in der Logik enthaltenen Mikrocontrollern gebildet. Darüber hinaus beinhaltet das Konzept eine Entwicklungsumgebung, die die unterschiedlichen Anwendergruppen bei Programmierung, Konfiguration und Einsatz des Antriebsreglers unterstützt.

Es wird gezeigt, wie die Funktionen der Antriebsregelung als Hardware- und Softwarekomponenten modularisiert werden können und wie die Infrastruktur aufgebaut werden kann, die die Kommunikation zwischen den Komponenten herstellt und die Ausführung und Überwachung der Komponenten koordiniert. Die Modularisierung der Antriebsfunktionen und die Definition einheitlicher Schnittstellen zur Antriebsreglerplattform ermöglicht den Aufbau einer Funktionsbibliothek, aus der der Anwender den auf seinen Anwendungsfall zugeschnittenen Regler konfigurieren, sowie neue Funktionalitäten in Form neuer Module hinzufügen kann.

Die durch die Antriebsreglerplattform allen Anwendern gebotene Offenheit und Flexibilität bedarf einer speziellen Entwicklungsumgebung, die die Funktionen in einer Bibliothek bereitstellt und die Anwender bei Konfiguration und Anpassung ihres Antriebsreglersystems unterstützt. Dabei werden die verschiedenen Anwendergruppen, die die Offenheit der Antriebsreglerplattform nutzen, berücksichtigt und die von ihnen benötigten grundlegenden Funktionen der Entwicklungsumgebung betrachtet.

Abschließend werden anhand einer Realisierung der Antriebsreglerplattform und einer Entwicklungsumgebung im Rahmen mehrerer Forschungsprojekte der Nutzen und die Leistungsfähigkeit eines Systems, das nach dem hier vorgestellten Konzept aufgebaut wurde, nachgewiesen.

Es zeigt sich, dass für die produktive Nutzung der Antriebsreglerplattform nicht nur die Leistungsfähigkeit der Hardware und der Funktionsumfang der Architektur ausschlaggebend sind, sondern dass für den optimalen Einsatz eine durchgängige und umfassende Entwicklungsumgebung notwendig ist.

Ausgehend von den für die Entwicklung und Konfiguration der Antriebsreglerplattform benötigten Werkzeugen ist eine Integration der Entwicklungsumgebung in eine maschinenübergreifende Projektierungslösung naheliegend. Die Funktionsblöcke des Antriebsreglers können zusammen mit den Funktionsbausteinen, wie sie beispielsweise aus OSACA bekannt sind, in einer Bibliothek zusammengefasst werden. Basierend auf dieser Bibliothek kann dann mit einem systemübergreifenden Entwicklungswerkzeug die vollständige Steuerungs- und Regelungssoftware entworfen, simuliert und entwickelt werden.

Die Antriebsreglerplattform bietet zum heutigen Zeitpunkt die Basis für eine Vielzahl von antriebstechnischen Neuentwicklungen, die sich die Eigenschaften der Plattform zunutze machen. Auch hier besteht großer Forschungsbedarf, um beispielsweise modellbasierte Regleralgorithmen schnell und effizient in schaltbarer Logik abzubilden, oder neuartige Filterfunktionen für die Maschinendiagnose zu integrieren.

Literatur

/1/ N.N.: Studie der Marktanteile industrieller Netzwerke. IMS Research, Wellingborough, England, 2008.

/2/ Kirchberger, R.; Willuweit, G.: Neue Konzepte für den Offenen Antrieb. Fortschritte in der Regelungs- und Antriebstechnik. 15. Lageregelseminar. ISW Stuttgart, 7.-8. März 2002.

/3/ N.N.: Anwenderhandbuch ACOPOS. B&R GmbH, Eggelsberg, Österreich, 2009.

/4/ N.N.: Produktbeschreibungen zu Compax3. Parker Hannifin GmbH, Offenburg, 2003.

.

/5/ N.N.: Produktbeschreibung Al Nano CNC for High-Speed. High Accuracy Machining. Fanuc, Japan, 2003.

/6/ N.N.: Produktbeschreibung sinumerik open architecture. Erlangen, 2003.

/7/ Pritschow, G., Sperling,W., Müller, J., Daniel, Ch.: OSACA – Auf dem Weg zur herstellerübergreifenden offenen Steuerung. In: Pritschow, G. [u.a.][Hrsg.]: Komponenten und Schnittstellen für offene Steuerungssysteme, S.7-27, München, Wien, Hanser Verlag, 1996.

/8/ Sperling, W., Lutz, P.: Enabling Open Control Systems. Robotics and Manufacturing. Recent Trends in Research and Applications, Vol. 6. Proceedings of the 6th International Symposium on Robotics and Manufacturing (ISRAM '96) , S. 613-620. New York: ASME Press, 1996.

/9/	Scheifele, D.:	Steuerungssoftware aus dem Baukasten für offene Steuerungen. Tagungsunterlagen zur ISW Arbeitstagung Steuerungstechnik '96, Stuttgart 1996.
/10/	N.N.:	Produktbeschreibung CNC FAGOR 8070. Fagor Automation S. Coop., Spanien, 2009.
/11/	N.N.:	Produktbeschreibung TwinCAT CNC. Beckhoff Automation GmbH, Verl, 2009.
/12/	Pritschow, G.; Weck, M.; Bauer, G.; Kahmen, A.; Kremer, M.; Wild, M.:	OCEAN: Open Controller Enabled by an Advanced Real-Time-Network. wt Werkstattstechnik online 93 - 5, S. 374-378, 2003.
/13/	N.N.:	Internetadresse: www.osadl.org.
/14/	Wörn, H.; Köller, T.; Leonhard, J.; Zimmermann, U.:	SOFIA – modulares Softwaresystem für intelligente Antriebe. Tagungsband SPS/IPC/Drives 2001, S.188-196, Hüthig Verlag, Heidelberg, 2001.
/15/	Park, J.; Pasek, Z., Shan, Y., [u.a]:	An Open Architecture Real-Time Controller for Machining Process. 27th CIRP Intl. Seminar of Manufacturing Systems, S. 27-44, Ann Arbor, 1995.
/16/	Koren, Y.:	Open-Architecture Controllers for Manufacturing Systems. Open Architecture Control Systems, ITIA Series, Vol.2, S. 85-101, ITIA (Selbstverlag), Mailand, 1998.

/17/ Keßler, K.: Geräte und Verfahren der digitalen Steuerung und Regelung elektrischer Antriebe. In: Die Technik der elektrischen Antriebe - Grundlagen, VEB Verlag Technik, Berlin, 1963.

/18/ Vogt, G.: Digitale Regelung von Asynchronmotoren für numerisch gesteuerte Fertigungseinrichtungen. Springer-Verlag, Berlin, 1985.

/19/ N.N.: Produktbeschreibung Blackfin Prozessoren. National Instruments Inc., USA, 2009.

/20/ N.N.: Fast "in time". Interview mit Hans-Jürgen Hilscher über die Lieferbarkeit eines Netzwerk-Controllers namens netX. Computer & Automation, Heft 8/2005, S. 90, WEKA Fachmedien GmbH, Poing, 2005.

/21/ Leonhard, A.: Elektrische Antriebe. F. Enke Verlag, Stuttgart, 1959.

/22/ Kallenbach, E.[Hrsg.]: Gerätetechnische Antriebe. Verlag Technik, Berlin, 1991.

/23/ Groß, H. [Hrsg.]: Elektrische Vorschubantriebe für Werkzeugmaschinen. Siemens AG, Berlin und München, 1981.

/24/ N.N.: Produktbeschreibung Simodrive 611D. Siemens AG, Erlangen, 2009.

/25/ N.N.: Produktbeschreibung bmaxx 4000. Baumüller GmbH, Nürnberg, 2009.

/26/	N.N.:	Produktbeschreibung IndraDrive. Bosch Rexroth AG, Lohr, 2009.
/27/	Stute, G. [Hrsg.]:	Regelung an Werkzeugmaschinen. Carl Hanser Verlag, München, 1981.
/28/	N.N.:	Produktbeschreibung GeoBrick. DeltaTau Data Systems, Inc., USA, 2009.
/29/	Duma R., Dobra P., Abrudean, M., Dobra M.:	Rapid Prototyping of Control Systems using Embedded Target for TI C2000 DSP. Proceedings of the 15th Mediterrane Conference on Control & Automation, 27.-29. Juli 2007, Athen.
/30/	Isermann, R.:	Digitale Regelsysteme. Springer Verlag, Berlin, 1987.
/31/	Kirchberger, R.; Müller, S.:	Hochdynamische, digitale Stromregelung und Stromerfassung mittels Oversampling für Servoantriebe. Tagungsband zum Kongress SPS/IPC/DRIVES 2000, S. 850-858. Nürnberg, 27.-30. November 2000.
/32/	Kirchberger, R.; Müller, S.:	Schnelle Stromregelung mittels programmierbarer Logik (FPGA). Fortschritte in der Regelungs- und Antriebstechnik. 15. Lageregelseminar. ISW Stuttgart, 7.-8. März 2002.
/33/	Hiller, B., Kirchberger, R.:	Oversamplingverfahren zur Verbesserung der Erfassung von Lage und Drehzahl an elektrischen Antrieben mit inkrementellen Gebersystemen. Tagungsband zum Kongress SPS/IPC/DRIVES 99. Nürnberg, 23.-25. November 1999.

/34/	N.N.:	Produktbeschreibung DSM-01. präTEC GmbH, Rohr, 2001.
/35/	Greiner, J.:	Verbesserte Zustandsgrößenerfassung bei Servo-Antrieben. Fortschritte in der Regelungs- und Antriebstechnik. 15. Lageregelseminar, ISW Stuttgart, 26.-27. Oktober 2001.
/36/	N.N.:	Produktbeschreibung VECTODRIVE VD 600. Aradex AG, Lorch, 2009.
/37/	N.N.:	Produktbeschreibung eDARC-C. Dokument: E-Darc_Cxx_DB_DE, Ferrocontrol Steuerungssysteme GmbH & Co. KG, Herford, September 2009.
/38/	N.N.:	Produktbeschreibung VxWorks. WindRiver, USA, 2009.
/39/	N.N.:	Produktbeschreibung TwinCAT. Beckhoff Automation GmbH, Verl, 2009.
/40/	Mantegazza, P.,Dozio, E. L., Papacharalambous, S.:	RTAI: Real Time Application Interface. Linux Journal archive, Volume 2000, Issue 72 (April 2000), Belltown Media Inc., USA, 2000.
/41/	Ackermann, J.:	Abtastregelung, Springer Verlag, Berlin, 1972.
/42/	Lincoln, B.:	A simple stability criterion for digital control systems with varying delays. IFAC World Congress, 2002.

/43/	Cervin, A., Henriksson, D., Lincoln, B., Eker, J., Årzén, K.:	How Does Control Timing Affect Performance? IEEE Control Systems Magazine, Juni 2003.
/44/	Proctor, F., Shackleford, W.:	Real-time Operating System Timing Jitter and its Impact on Motor Control. Proceedings of the 2001 SPIE Conference on Sensors and Controls for Intelligent Manufacturing II, Volume 4563, S.10…16, Boston, 2001.
/45/	Pöhlmann, T.:	Antriebsregler leicht gemacht. embedded engineering, Februar 2002, Carl Hanser Verlag, München, 2002.
/46/	N.N.:	Produktbeschreibung Matlab. Mathworks Inc., USA, 2009.
/47/	N.N.:	Produktbeschreibung CompactRIO. National Instruments Corp., USA, 2009.
/48/	N.N.:	Produktbeschreibung LabVIEW. National Instruments Corp., USA, 2009.
/49/	Weber, W.-D.:	Enabling Reuse via an IP Core-centric communications Protocol: Open Core Protocol. Proceedings of IP 2000 System-on-Chip Conference, S. 217–224, März 2000.
/50/	N.N.:	Internet-Adresse: http://www.opencores.net.
/51/	N.N.:	Specification for the WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores. Revision: B.3, 7. September 2002.

/52/ N.N.: IEEE P1003.1, Std. for IT-Portable Operating System Interface (POSIX) Part 1: Systems Application Program Interface (API). Los Alamitos, IEEE Computer Society Publications, 1995.

/53/ Sperling, W.: Modulare Systemplattformen für offenen Steuerungssysteme. Springer-Verlag, Berlin, 1999.

/54/ Balzert, H.: Lehrbuch der Software-Technik, Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum Akademischer Verlag, Heidelberg, 1996.

/55/ Fritz, S.: Antriebsinterne Prozeßkraftnachbildung. Tagungsband zum ABS-Treffen `99, Aachen, 21.-22. Juli 1999.

/56/ Walther, M.; Verl, A.: Antriebsnahe Maschinendiagnose. Zuverlässigkeit und Diagnose in der Produktion, Fortschritt-Berichte VDI, Reihe 2, Nr. 663, S. 39-55, VDI-Verlag, Düsseldorf, 2007.

/57/ Maier, D.; Nebel, S.; Rüdele, H.; Walther, M.; Oglodin, V.: Sensorlose vorausschauende Wartung von Vorschub-antrieben an Werkzeugmaschinen - Gezielte vor-beugende Wartung durch automatisierte Zustands-beobachtung Schwingungsüberwachung und Diagnose von Maschinen, VDI-Schwingungstagung 2007, (VDI-Berichte Nr. 1982), S. 235-249, VDI, Düsseldorf, 2007.

/58/	Walther, M.; Birenbaum Ch.; Maier, D.:	Sensorlose automatisierte Verschleißdiagnose an Vorschubsystemen mit Kugelgewindetrieben in Werkzeugmaschinen Schwingungsanalyse & Identifikation, Fachtagung, Stadthalle Leonberg, 23. und 24. März 2010 / VDI-Schwingungstechnik, (VDI-Berichte Nr. 2093), S. 231-243, VDI, Düsseldorf, 2010.
/59/	Wosnik, M.; Kramer, C.; Selig, A.; Klemm, P.:	Enabling feedback of process data by use of STEP-NC. International Journal of Computer Integrated Manufacturing 19 (2006) 6, S. 559-569, 2006.
/60/	N.N.:	EN-60204-1, Sicherheit von Maschinen - Elektrische Ausrüstungen von Maschinen - Teil 1: Allgemeine Anforderungen. Beuth Verlag, Berlin, Mai 2005.
/61/	Föllinger, O.:	Regelungstechnik, Einführung in die Methoden und ihre Anwendung. 10. Auflage, Hüthig Verlag, Heidelberg, 2008.
/62/	N.N.:	Produktbeschreibung Quartus II. Altera Corp., USA, 2009.
/63/	Oberschelp, W., Vossen, G.:	Rechneraufbau und Rechnerstrukturen. 10. Auflage, Oldenbourg Verlag, München, 2006.
/64/	Anderson, D., Padgett, W.:	Fixed Point Signal Processing. Morgan & Claypool Publishers, San Rafael, USA, 2009.
/65/	N.N.:	IEEE 754, Standard for Binary Floating-Point Arithmetic for microprocessor systems. Los Alamitos, IEEE Computer Society Publications, 1989.

/66/ Dido, J., Geraudie, N., Loiseau, L., Payeur, O., Savaria,Y.: A Flexible Floating-Point Format for Optimizing Data-Paths and Operators in FPGA Based DSPs. FPGA 2002, Tenth ACM International Symposium on Field-Programmable Gate Arrays, 24.-26. Februar 2002.

/67/ Otto, A., Hohenstein, R., Dietrich, S.: Diagnostik und Regelung beim Laserschweißen. 5. Laser-Anwenderforum, S.167-176, Bremen, 2006.

/68/ Kostikov, A.: Innovative Regelungskonzepte für Vorschubachsen: Sliding-Mode-Regelung, Fortschritte in der Regelungs- und Antriebstechnik. 16. Lageregel-seminar, ISW Stuttgart, 28.-29. Juni 2005.

/69/ N.N.: Internetadresse: http://www.altera.com/products/ip/ip-index.jsp.

/70/ N.N.: Internetadresse: http://www.xilinx.com/ipcenter/index.htm.

/71/ N.N.: IEEE 1076-2008. IEEE Standard VHDL Language Reference Manual, Los Alamitos, IEEE Computer Society Publications, 2009.

/72/ Dauman, A.: How to implement an open IP encryption flow. EETimes.com, 2006.

/73/ Kean, T.: Cryptographic Rights Management of FPGA Intellectual Property Cores, Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays , S. 113-118, Monterey, USA, 2002.

/74/ Klemm, P.; Selig, A.; Wosnik, M.; Kramer, C.: Rückführung von Prozessdaten mittels STEP-NC (Feedback of process data using STEP-NC).wt Werkstattstechnik online 95 (2005) 5, S. 368-372, 2005.

/75/ N.N.: Internet-Adresse: http://www.forschungskoop.de.

/76/ Ocampo Hidalgo, J.: System and Circuit Approaches for the Design of Multi-mode: Sigma-Delta Modulators with Application for Multi-standard Wireless Receivers, Dissertation, TU Darmstadt, 2004.

/77/ N.N.: Drehgeber. Dok.Nr. 349 529-11, 4/2000. Dr. Johannes HEIDENHAIN GmbH, Traunreuth, 2000.

/78/ Pritschow, G.; Hiller, B.: Hohe Regelgüte durch verbesserte Messsystemauswertung und Beschleunigungssensoren. wt Werkstattstechnik 88 (1998) 11/12, S. 473-478, 1998.

/79/ Hiller, B.: Ferraris-Sensor - Was steckt dahinter? Fortschritte in der Regelungs- und Antriebstechnik. 15. Lageregelseminar, ISW Stuttgart, 26.-27. Oktober 2001.

/80/ N.N.: Specification SERCOS interface Version 2.2. IGS, Stuttgart, 2001.

/81/ Staudt, S.: SERCOS Interface - die intelligente digitale Antriebsschnittstelle. Fortschritte in der Regelungs- und Antriebstechnik. 15. Lageregelseminar. ISW Stuttgart, 7.-8. März 2002.

/82/ Hippe, P., Wurmthaler, C.: Stellbegrenzungen in Regelkreisen. In: Elektrische Antriebe - Regelung von Antriebssystemen, 2. Auflage, Hrsg. Schröder, D., Springer Verlag, Berlin, 2001.

/83/ Brigham, E.: FFT Schnelle Fourier-Transformation. 3. Auflage, Oldenbourg Verlag, München, 1987.

/84/ Schüßler, H.W.: Digitale Signalverarbeitung 1. 5. Auflage, Springer Verlag, Berlin, 2008.

Lebenslauf

Persönliches:	Name	Christian Kramer
	Geburtsort	Frankfurt a. M.
	Geburtstag	29.9.1975
	Eltern	Manfred und Brigitta Kramer
	Familienstand	verheiratet
Schulbildung:	1986 - 1995	Fürst-Johann-Moritz Gymnasium, Siegen
Studium:	1996 - 2001	Studium der Technischen Kybernetik an der Universität Stuttgart Diplomzeugnis vom 16. November 2001
Berufstätigkeit:	Dez 2001 - Nov 2007	Wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart
	Dez 2007 - Feb 2008	Mitarbeiter der Forschungs- und Ingenieurs-gesellschaft für Steuerungstechnik (FISW) GmbH, Stuttgart
	Jul 2004 - Feb 2008	Leiter der Gruppe „Antriebs- und Regelungstechnik“ am ISW
	Seit März 2008	Mitarbeiter der Pilz GmbH & Co. KG in Ostfildern

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart und dem Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA

Herausgegeben bis Band 57 von Prof. Dr.-Ing. G. Stute †
ab Band 58 von Prof. Dr.-Ing. Dr. h. c.mult. Dr.-Ing. E.h G. Pritschow
ab Band 161 von Prof. Dr.-Ing. A. Verl

Erschienen bei Springer-Verlag:

1	Schmid, D.:	Numerische Bahnsteuerung, 1973.
2	Schwegler, H.:	Fräsbearbeitung gekrümmter Flächen, 1972.
3	Eisinger, J.:	Numerisch gesteuerte Mehrachsenfräsmaschinen, 1972.
4	Nann, R.:	Rechnersteuerung von Fertigungseinrichtungen, 1972.
5	Augsten, G.:	Zweiachsige Nachformeinrichtungen, 1972.
6	Karl, B.:	Die Automatisierung der Fertigungsvorbereitung durch NC-Programmierung, 1972.
7	Eitel, H.:	NC-Programmiersystem, 1973.
8	Knorr, E.:	Numerische Bahnsteuerung zur Erzeugung von Raumkurven auf rotationssymetrischen Körpern, 1973.
9	Bumiller, S.:	Viskohydraulischer Vorschubantrieb, 1974.
10	Maier, K.:	Grenzregelung an Werkzeugmaschinen, 1974.
11	Waelkens, J.:	NC-Programmierung, 1974.
12	Bauer, E.:	Rechnerdirektsteuerung von Fertigungseinrichtungen, 1975.
13	König, H.:	Entwurf und Strukturtheorie von Steuerungen für Fertigungseinrichtungen, 1976.
14	Damsohn, H.:	Fünfachsiges NC-Fräsen, 1976.
15	Jetter, H.:	Programmierbare Steuerungen, 1976.
16	Henning, H.:	Fünfachsiges NC-Fräsen gekrümmter Flächen, 1976.
17	Boelke, K.:	Analyse und Beurteilung von Lagesteuerungen für numerisch gesteuerte Werkzeugmaschinen, 1977.

18 Götz, F.-R.: Regelsystem mit Modellrückkopplung für variable Streckenverstärkung, 1977.

19 Tränkle, H.: Auswirkungen der Fehler in den Positionen der Maschinenachsen beim fünfachsigen Fräsen, 1977.

20 Stof, P.: Untersuchungen über die Reduzierung dynamischer Bahnabweichungen bei numerisch gesteuerten Werkzeugmaschinen, 1978.

21 Wilhelm, R.: Planung und Auslegung des Materialflusses flexibler Fertigungssysteme, 1978.

22 Kappen, N.: Entwicklung und Einsatz einer direkten digitalen Grenzregelung für eine Fräsmaschine mit CNC, 1979.

23 Klug, H. G.: Integration automatisierter technischer Betriebsbereiche, 1978.

24 Binder, D.: Interpolation in numerischen Bahnsteuerungen, 1979.

25 Klingler, O.: Steuerung spanender Werkzeugmaschinen mit Hilfe von Grenzregeleinrichtungen, 1979.

26 Schenke, L.: Auslegung einer technologisch-geometrischen Grenzregelung für die Fräsbearbeitung, 1979.

27 Wörn, H.: Numerische Steuersysteme – Aufbau und Schnittstellen eines Mehrprozessorsteuersystems, 1979.

28 Osofisan, P. B.: Verbesserung des Datenflusses beim fünfachsigen NC-Fräsen, 1979.

29 Berner, J.: Verknüpfung fertigungstechnischer NC-Programmiersysteme, 1979.

30 Böbel, K.-H.: Rechnerunterstützte Auslegung von Vorschubantrieben, 1979.

31 Dreher, W.: NC-gerechte Beschreibung von Werkstücken in fertigungstechnisch orientierten Programmiersystemen, 1980.

32 Schurr, R.: Rechnerunterstützte Projektsteuerung hydrostatischer Anlagen, 1981.

33 Sielaff, W.: Fünfachsiges NC-Umfangfräsen verwundener Regelflächen. Beitrag zur Technologie und Teileprogrammierung, 1981.

34 Hesselbach, J.: Digitale Lageregelung an numerisch gesteuerten Fertigungseinrichtungen, 1981.

35 Fischer, P.: Rechnerunterstützte Erstellung von Schaltplänen am Beispiel der automatisierten Hydraulikplanzeichnung, 1981.

36 Ackermann, U.: Rechnerunterstützte Auswahl elektrischer Antriebe für spanende Werkzeugmaschinen, 1981.

37 Döttling, W.: Flexible Fertigungssysteme – Steuerung und Überwachung des Fertigungsablaufs, 1981.

38 Firnau, J.: Flexible Fertigungssysteme – Entwicklung und Erprobung eines zentralen Steuersystems, 1982.

39 Herrscher, A.: Flexible Fertigungssysteme – Entwurf und Realisierung prozeßnaher Steuerungsfunktionen, 1982.

40 Spieth, U.: Numerische Steuersysteme – Hardwareaufbau und Ablaufsteuerung eines Mehrprozessorsteuersystems, 1982.

41 Schimmele, A.: Rechnerunterstützter Entwurf von Funktionssteuerungen für Fertigungseinrichtungen, 1982.

42 Sanzenbacher, M.: NC-gerechte Beschreibung von Werkstücken mit gekrümmten Flächen, 1982.

43 Walter, W.: Interaktive NC-Programmierung von Werkstücken mit gekrümmten Flächen, 1982.

44 Huan, J.: Bahnregelung zur Bahnerzeugung an numerisch gesteuerten Werkzeugmaschinen, 1982.

45 Erne, H.: Taktile Sensorführung für Handhabungseinrichtungen – Systematik und Auslegung der Steuerungen, 1982.

46 Plasch, D.: Numerische Steuersysteme – Standardisierte Softwareschnittstellen in Mehrprozessor-Steuersystemen, 1983.

47 Wang, Z. L.: NC-Programmierung – Maschinennaher Einsatz von fertigungstechnisch orientierten Programmiersystemen, 1983.

48 Schwager, J.: Diagnose steuerungsexterner Fehler an Fertigungseinrichtungen, 1983.

49 Klemm, P.: Strukturierung von flexiblen Bediensystemen für numerische Steuerungen, 1984.

50 Runge, W.: Simulation des dynamischen Verhaltens elektrohydraulischer Schaltungen – Einsatz von geräteorientierten, universellen Simulationsbausteinen, 1984.

51 Steinhilber, H.: Planung und Realisierung von Werkzeugversorgungssystemen für die NC-Bearbeitung, 1984.

52 Ohnheiser, R.: Integrierte Erstellung numerischer Steuerdaten für flexible Fertigungssysteme, 1984.

53 Keppeler, M.: Führungsgrößenerzeugung für numerisch bahngesteuerte Industrieroboter, 1984.

54 Kohler, P.: Automatisiertes Messen mit NC-Werkzeugmaschinen, 1985.

55 Rieger, K.-H.: Rechnerunterstützte Projektierung der Hardware und Software von Speicherprogrammierten Steuerungen, 1985.

56 Vogt, G.: Digitale Regelung von Asynchronmotoren für numerisch gesteuerte Fertigungseinrichtungen, 1985.

57 Chmielnicki, S.: Flexible Fertigungssysteme – Simulation der Prozesse als Hilfsmittel zur Planung und zum Test von Steuerprogrammen, 1985.

58 Renn, W.: Struktur und Aufbau prozeßnaher Steuergeräte zur Verkettung in flexiblen Fertigungssystemen, 1986.

59 Harig, K.: Quantisierung im Lageregelkreis numerisch gesteuerter Fertigungseinrichtungen, 1986.

60 Frank, H.: Programmier- und Überwachungsfunktionen für teileartbezogene NC-Werkzeugmaschinen, 1986.

61 Möller, H.: Integrierte Überwachungs- und Diagnose-Systeme für numerische Steuerungen, 1986.

62 Fink, H.: Einsatz speicherprogrammierbarer Steuerungen in der Fertigungstechnik, 1986.

63 Fleckenstein, J.: Zustandsgraphen für SPS – Grafikunterstützte Programmierung und steuerungsunabhängige Darstellung, 1987.

64 Wagner, E.: Steuerungen von Koordinatenmeßgeräten mit schaltenden und messenden Tastsystemen, 1987.

65 Grimm, W.: Diagnosesystem für steuerungsperiphere Fehler an Fertigungseinrichtungen, 1987.

66 Swoboda, W.: Digitale Lageregelung für Maschinen mit schwach gedämpften schwingungsfähigen Bewegungsachsen, 1987.

67 Gruhler, G.: Sensorgeführte Programmierung bahngesteuerter Industrieroboter, 1987.

68 Walker, B.: Konfigurierbarer Funktionsblock Geometriedatenverarbeitung für numerische Steuerungen, 1987.

69 Mayer, J.: Werkzeugorganisation für flexible Fertigungszellen und -systeme, 1988.

70 Lederer, R.: Programmierung von NC-Drehmaschinen mit mehreren Werkzeugschlitten, 1988.

71 Häberle, G.: NC-Musterprogrammierung für rechnerintegrierte Textilfertigung, 1988.

72 Pfeiffer, D.: Kompensation thermisch bedingter Bearbeitungsfehler durch prozeßnahe Qualitätsregelung, 1988.

73 Schmidt, W.: Grafikunterstütztes Simulationssystem für komplexe Bearbeitungsvorgänge in numerischen Steuerungen, 1988.

74 Egner, M.: Hochdynamische Lageregelung mit elektrohydraulischen Antrieben, 1988.

75 Schittenhelm, W.: Konfigurierbares Bedienungssystem für Steuerungen an Fertigungseinrichtungen, 1988.

76 Scheifele, D.: Grafisch dynamische Simulation des Bearbeitungsvorgangs für Doppelschlittendrehmaschinen, 1988.

77 Keuper, G.: Automatisierte Identifikation der Streckenparameter servohydraulischer Vorschubantriebe, 1989.

78 Kayser, K.-H.: Kollisionserkennung in numerischen Steuerungen mit der Distanzfeldmethode, 1989.

79 Viefhaus, R.: Fräsergeometriekorrektur in Numerischen Steuerungen für das fünfachsige Fräsen, 1989.

80 Zirbs, J.: Fertigungsgerechte Aufbereitung von Flächenverbänden bei der NC-Programmierung im Formenbau, 1989.

81 Ruoff, W.: Optische Sensorsysteme zur On-line-Führung von Industrierobotern, 1989.

82 Jantzer, M.: Bahnverhalten und Regelung fahrerloser Transportsysteme ohne Spurbindung, 1990.

83 Schumacher, H.: Einheitliche Programmierung von Automatisierungskomponenten roboterbestückter Bearbeitungs- und Montagezellen, 1991.

84 Schimonyi, J.: NC-Programmierung für das Werkzeugschleifen, 1991.

85 Wurst, K.-H.: Flexible Robotersysteme – Konzeption und Realisierung modularer Roboterkomponenten, 1991.

86 Hagl, R.: Erhöhung der Verfügbarkeit von Vorschubantrieben mit selbstanpassender Lageregelung, 1991.

87 Krebser, G.: Betriebssystem für NC mit einheitlichen Schnittstellen, 1992.

88 Lei, W.-T.: Flächenorientierte Steuerdatenaufbereitung für das fünfachsige Fräsen, 1992.

89 Diehl, G.: Steuerungsperipheres Diagnosesystem für Fertigungseinrichtungen auf Basis überwachungsgerechter Komponenten, 1992.

90 Nepustil, U.: Offene NC-Schnittstellen zur Korrektur von Fertigungsfehlern, 1992.

91 Bauder, M.: Konfigurierbare Robotersteuerung mit allgemeiner Transformation, 1992.

92 Philipp, W.: Regelung mechanisch steifer Direktantriebe für Werkzeugmaschinen, 1992.

93 Härdtner, G. M.: Wissensstrukturierung in Diagnoseexpertensystemen für Fertigungseinrichtungen, 1992.

94 Wiedmann, H.: Objektorientierte Wissensrepräsentation für die modellbasierte Diagnose an Fertigungseinrichtungen, 1993.

95 Rudloff, H.: Hochgenaue Konturerzeugung bei Bewegungsachsen mit einer dominanten mechanischen Resonanzstelle, 1993.

96 Brantner, K.: Adaptierbares Leitsteuerungssystem für flexible Produktionssysteme, 1993.

97 Kugler, W.: Kommunikationsmechanismen für offene Numerische Steuerungssysteme, 1994.

98 Schnurr, B.: Elektrodynamisches Antriebssystem zur Unrundbearbeitung, 1994.

99 Schneider, J.: Fehlerreaktion mit Speicherprogrammierbaren Steuerungen – ein Beitrag zur Fehlertoleranz, 1994.

100 Siewert, U.: Systematische Erstellung adaptierbarer Leitsteuerungssoftware am Beispiel der Durchsetzungsplanung, 1994.

101 Heger, G. F. J.: Maschinenferner Qualitätsregelkreis in flexiblen Fertigungssystemen, 1994.

102 Hofmeister, W.: Objektorientiert strukturiertes Programmiersystem für NC-Mehrschlittenmaschinen, 1994.

103 Horn, A.: Optische Sensorik zur Bahnführung von Industrierobotern mit hohen Bahngeschwindigkeiten, 1994.

104 Rentschler, U.: Fehlertolerantes Präzisionsfügen, 1995.

105 Junghans, G.: Modulares grafikunterstütztes Simulationssystem für Bearbeitungs- und Handhabungsvorgänge, 1995.

106 Heller, J.: Sensorgestützte Bewegungserzeugung leitlinienloser Transprtfahrzeuge, 1995.

107 Wieland, E.: Anwendungsorientierte Programmierung für die robotergestützte Montage, 1995.

108 Ketterer, G.: Automatisierte Inbetriebnahme elektromechanischer, elastisch gekoppelter Bewegungsachsen, 1995.

109 Reibetanz, Th.: Situationsorientierte Bearbeitungsmodellierung zur NC-Programmierung, 1995.

110	Frager, O.:	Durchgängige Programmierung von Fertigungszellen, 1996.
111	Ordenewitz, R.:	Betriebsweite Bereitstellung von Werkzeuginformationen, 1996.
112	Daniel, C.:	Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme, 1996.
113	Angerbauer, R.:	Anwenderorientierte Programmierung fahrerloser Transportsysteme, 1996.
114	Krauß, F.:	Splineverarbeitung in numerischen Steuerungen für das fünfachsige Fräsen, 1996.
115	Schittenhelm, K.-M.:	Einsatz vorgefilterter Führungsgrößen für Bewegungsachsen zur Bahnerzeugung, 1997.
116	Häberle, U.:	Einheitliche Anwenderschnittstelle für Feldbussysteme, 1997.
117	Strassacker, D.:	Testumgebung für die Implementierung und Inbetriebnahme eines adaptierbaren Leitsteuerungssystems, 1997.
118	Renz, B.:	Hochdynamische Strahllagekorrektursysteme zur Erhöhung der Bahngenauigkeit von CO_2-Laserbearbeitungsmaschinen, 1997.
119	Itterheim, C.:	Objektorientiertes Bearbeitungsmodell für Freiformflächen – Erstellung und maschinengebundene Modifikation –, 1997.
120	Müller, J.:	Objektorientierte Softwareentwicklung für offene numerische Steuerungen, 1997.
121	Glöckler, M.:	Verbesserung des Störverhaltens elektrohydraulischer lagegeregelter Zylinderantriebe, 1998.
122	Uhl, J.:	Entwurfssystematik für ein dezentral strukturiertes, objektorientiertes Fertigungsleitsystem, 1998.
123	Hammann, G.:	Modellierung des Abtragsverhaltens elastischer, robotergeführter Schleifwerkzeuge, 1998.
124	Scholich-Tessmann, W.:	Direktantriebe für Industrieroboter, 1998.
125	Wagner, R.:	Robotersysteme zum kraftgeführten Entgraten grobtolerierter Leichtmetallwerkstücke mit Fräswerkzeugen, 1998.
126	Anders, C.:	Adaptierbares Diagnosesystem bei Transferstraßen, 1998.
127	Fahrbach, C.:	Regelung hochdynamischer elektrischer Servo-Direktantriebe in Fertigungseinrichtungen, 1999.
128	Sperling, W.:	Modulare Systemplattformen für offene Steuerungssysteme, 1999.

Erschienen bei Jost-Jetter Verlag:

129 Kehl, G.: Gestaltung von Formgedächtnis-Aktorsystemen für sensorgeführte Inspektionsgeräte, 1999.

130 Brandl, Th.: Anlageninformationssystem - Informationsmodell und Erstellungssystematik, 1999.

131 Gronbach, H.: Simulationswerkzeug für die Gestaltung modularer CO_2-Laserbearbeitungsmaschinen, 1999.

132 Lutz, R.: Softwaretechnik für Maschinennahe Steuerungsfunktionen bei Fertigungseinrichtungen, 1999.

133 Tran, T. L.: Allgemeine Transformation für Maschinen mit Parallelkinematiken, 2000.

134 Bretschneider, J.: Reglerselbsteinstellung für digital geregelte, elektromechanische Antriebssysteme an Werkzeugmaschinen, 2000.

135 Schoenenberg, M.: Zuverlässiger Fertigungsprozess bei Transferstraßen durch präventive Maßnahmen, 2000.

136 Uhl, A.: Flexibles Telerobotersteuerungssystem auf der Basis offener numerischer Steuerungen, 2000.

137 Rui Li: Agentenbasierte NC-Planung für die Komplettbearbeitung auf Dreh-/Fräszentren, 2001.

138 Wildermuth, D.: Bahnvorbereitung in numerischen Steuerungen für Parallelkinematiken, 2001.

139 Handel, D.: Werkergerechte NC-Programmierung zur Komplett-Schleifbearbeitung von Bohrwerkzeugen, 2001.

140 Kosiedowski, U.: Adaptive Vorsteuerverfahren für elektromechanische Bewegungsachsen an Werkzeugmaschinen, 2001.

141 Haug, K.: Laser-Lichtschnittsensorik für die Automatisierung von Metall-Schutzgasschweißprozessen, 2002.

142 Hohenadel, J.: Einheitliches Steuerungssystem für NC und RC, 2002.

143 Wälde, K.: Sicherstellung der Softwarequalität von Anwendungsmodulen und Systemplattformen in offenen Steuerungssystemen, 2002.

144 Litto, M.: Störungsinformationssystem – Informationsmodell und Erstellungssystematik, 2002.

145 Schweiker, A.: Offene numerische Steuerungen für prozeßabhängige Bearbeitungen – vereinheitlichte Struktur, Funktionen und Schnittstellen – , 2003.

146 Rempp, B.: Regelungstechnische Untersuchung durchsatzgeregelter Produktionssysteme, 2003.

147 Eppler, C.: Kompensation fremderregter Schwingungen in Antriebssystemen mit Umlaufgetrieben, 2003.

148 Weiner, M.: Wiederverwendungsgerechte Entwicklungssystematik von Feinplanungssoftware für die flexible Fertigung, 2004.

149 McCormac, St.: Lageregelung hydraulischer Manipulatoren unter Einsatz eines Ferraris-Sensors, 2004.

150 Lehner, W.-D.: Regelung von Vorschubachsen unter Verwendung der Relativbeschleunigung, 2005.

151 Lewek, J.: Adaptierbares Informationssystem zur Erstellung baukastenbasierter Fertigungseinrichtungen, 2005.

152 Laible, U.: Aufbau numerischer Steuerungssysteme für sicherheitskritische Anwendungen, 2005.

153 Bürger, Th.: Durchgängige analytische Qualitätssicherung für numerische Steuerungssoftware, 2005.

154 Brinzer, B.: Produktionsregelung für die variantenreiche Serienfertigung, 2005.

155 Heusinger, S.: STEP-NC-basierter Korrekturkreis für die Schlichtbearbeitung von Freiformflächen, 2005.

156 Kirchberger, R.: Verbesserte Auswertung inkrementeller Messsysteme durch schnelle Signal-Vorverarbeitung, 2005.

157 Conrath, M.: Systematische Gestaltung von frequenzadaptierbaren Ultraschall-Werkzeugsystemen zum Einsatz in fertigungstechnischen Prozessen, 2005.

158 Wadehn, W.: Gestaltung von Antriebssystemen für formadaptive Strukturen, 2005.

159 Horber, H.: Fugendetektion bei Lichtbogenschweißprozessen mit robuster Signalverarbeitung für optische Sensoren, 2006.

160 Dreyer, J.: Situative Informationsbereitstellung an Fertigungseinrichtungen Informationsmodell und Erstellungssystematik, 2006.

161 Fritz, S.: Rekonstruktion von Prozesskräften aus Antriebssignalen von Werkzeugmaschinen, 2006.

162 Staudt, S.: Spezifikation und Konformitätstest zur Interoperabilität von automatisierten Produktionsmaschinen, 2006.

163 Reichle, R.: Einsatz von Internet-Werkzeugen und -Diensten in numerischen Steuerungssystemen für Werkzeugmaschinen, 2006.

164 Kaiser, L.: Systematik für das Qualitäts- und Projektmanagement bei der Abwicklung von Unikatprojekten, 2006.

165 Garber, Th.: Nutzung des redundanten Freiheitsgrades von sechsachsigen Parallelkinematik-Maschinen, 2007.

166 Altenburger, R.: Dynamische Eigenschaften und Regelung mechanisch verkoppelter Antriebssysteme, 2007.

167 Korajda, B.: Steuerungstechnische Verfahren zur echtzeitfähigen Kompensation der Fräserabdrängung, 2007.

168 Röck, S.: Echtzeitsimulation von Produktionsanlagen mit realen Steuerungssystemen, 2007.

169 Priese, J.: Verfahren zur durchgehenden dezentralen Planung in Werkstattstrukturen, 2007.

170 Boye, T.: Vorhersage der kinematischen Kalibriergüte von Parallelkinematiken, 2008.

171 Joannides, M.: Ein Beitrag zur volumenorientierten 3D-Objektrekonstruktion aus digitalen Daten, 2009.

172 Pruschek, P.: Verfahren zur anwendungsgerechten Parametrierung der Steuerung und Regelung von Vorschubachsen, 2009.

173 Fritsch, D.: Steuerung selbstorganisierender Multi-Roboter-Systeme für dynamische Sammelaufgaben am Beispiel der Bekämpfung maritimer Ölverschmutzungen, 2009.

174 Schmitz, S.: Industrielle Powerline-Kommunikation für Antriebseinheiten in Werkzeugmaschinen, 2010.

175 Bengel, M.: Workpiece-centered Approach to Reconfiguration in Manufacturing Engineering, 2010

176 Weimer, T.: Informationsmodell für die durchgängige Datennutzung in Fabrikplanung und -betrieb, 2010

177 Oglodin, V.: Maschinenübergreifender agentenbasierter Informationsaustausch für die Störungsbeseitigung, 2010

178 Kramer, C.: Offene Antriebsreglerplattform, 2011