B3 Anwendung rekonfigurierbarer Logik im Maschinenbau am Beispiel einer adaptiven Regelung

Thomas Reinemann, Roland Kasper

Otto-von-Guericke-Universität Magdeburg
Institut für Mechatronik und Antriebstechnik (IMAT)
PF 4120; D-39016 Magdeburg

Zusammenfassung

Mikro-Controller und Signalprozessoren bilden heute die Standard-Implementierungsplattform für eingebettete Controller. Aufgrund ihrer festen Hardware-Architektur lassen sie sich jedoch kaum an veränderte Hardware-Anforderungen anpassen, z.B. bei Redesigns, Upgrades oder dynamisch in bestimmten Betriebspunkten. Konfigurierbare Controller auf der Basis von FPGAs bieten in diesem Punkt eine wesentlich größere Flexibilität. Durch die Anpassung von Rechen- und Kommunikationsleistung, des benötigten Speicherplatzes sowie der externen Schnittstellen an die jeweilige Anwendung, erlauben konfigurierbare Controller eine wesentlich bessere Nutzung der verfügbaren Hardware-Ressourcen.

Dabei sind die statische und dynamische Rekonfiguration zu unterscheiden. Statische Rekonfiguration findet außerhalb des regulären Betriebs der Anlage statt und erlaubt Änderungen im späten Entwicklungsstadium bzw. dient der Aktualisierung der Steuerung, während bei der dynamischen Rekonfiguration im Betrieb Logik- und Rechenkomponenten ersetzt bzw. modifiziert werden.

Am Beispiel einer adaptiven Regelung werden die Möglichkeiten der rekonfigurierbaren Logik im Maschinenbau gezeigt.

Stand der Technik

Die rasante Entwicklung der Mikroelektronik im Allgemeinen und der programmierbaren Logikbausteine (FPGAs und CPLDs) im Speziellen, eröffnen völlig neue Möglichkeiten der digitalen Implementierung von Steuerungen, Regelungen und der Signalverarbeitung in eingebetteten Systemen. War der Siegeszug der Mikrocontroller und der digitalen Signalprozessoren durch eine leistungsfähige Hardware mit flexiblen und anpassbaren Softwarestrukturen geprägt, so weisen programmierbare Logikbausteine durch flexible und anpassbare Hardwarestrukturen den Weg in eine neue Ära rekonfigurierbarer Controller. Diese ermöglichen die Anpassung Hardware-basierter Controller an bestimmte Betriebsbedingungen durch Änderung der Konfiguration im Betrieb, wodurch die notwendige Chipfläche und der Energiebedarf reduziert wird. In [1, 2] wird ein guter Überblick der Potentiale und Implementierungsmöglichkeiten rekonfigurierbarer Rechnerarchitekturen gegeben. Besonders hervorzuheben sind Überlegungen zur Parallelisierung auf Blockebene und Serialisierung auf Signalebene in [2], welche zu sehr leistungsfähigen Lösungen führen.

Die Entwicklung *universell* verwendbarer dynamischer rekonfigurierbarer Rechnerarchitekturen wirft eine Reihe schwieriger Fragestellungen auf. Die dynamische Rekonfiguration des Systems wird durch den Austausch einiger dieser Komponenten zur Laufzeit ermöglicht. Dabei werden häufig Komponenten in Form eines Betriebssystems benötigt, die den Rekonfigurationsprozess steuern und überwachen. Die Basis für die Rekonfiguration bilden FPGA Konfigurationsdateien, welche den Ressourcenbedarf einer Anwendung abbilden und für eine dynamische Zuordnung benötigter Ressourcen genutzt werden. Allerdings verursacht die Rekonfigurierbarkeit auch Kosten. Zur Lösung eines Problems sind im Allgemeinen mehrere Tasks erforderlich, darum ist sicherzustellen, dass alle diese Tasks gleichzeitig geladen und verbunden werden können.

Häufig werden rekonfigurierbare Systeme jedoch gezielt für bestimmte Anwendungsfälle konzipiert, da dort genau definierte Einsatzbedingungen die Anpassung der Hardware-Freiheitsgrade erleichtern. So gibt es viel versprechende Hochgeschwindigkeitsanwendungen für rekonfigurierbare Systeme im Bereich der Signalverarbeitung und Bilderkennung.

Jedoch lassen sich sequentielle Algorithmen besser in Software als in Hardware abbilden. Darum bietet sich die Kombination von Prozessor und programmierbarer Logik an. Um jedoch die Flexibilität zu erhöhen, ist es besser den Prozessorkern in den PLD als Softcore zu integrieren.

Die Dauer der Produktzyklen im Maschinen- und Fahrzeugbau liegt deutlich über denen in der IT-Industrie, weshalb Prozessoren und andere ICs am Ende des Produktzyklus häufig nicht mehr im Handel sind. Dem kann durch statische Rekonfiguration und damit einer universellen Steuergerätehardware abgeholfen werden.

Gegenwärtig erfolgt die Spezifikation von Regelalgorithmen für Hardware auf der Basis von Hardwarebeschreibungssprachen (HDL). Dies ist jedoch nicht die Methode für den Entwicklungsingenieur in der Mechatronik, welcher Blockschaltpläne bevorzugt. Auch stellen HDLs arithmetische Funktionen nur im geringen Umfang zur Verfügung. Diese sind vom Entwickler oder PLD-Hersteller bereit zu stellen und es fehlt demzufolge eine einheitliche Bibliothek arithmetischer Funktionen, die die Portabilität unterstützt. Die so erstellten Rechenelemente benutzen im Allgemeinen bitparallele Arithmetik in Verbindung mit logikintensiven und komplizierten Operatoren. Darum wurde in [3] ein Verfahren zur Spezifikation und Synthese von Steuer- und Regelalgorithmen auf bitserieller Basis entwickelt. Hier wird zur Spezifikation ein Blockdiagrammeditor benutzt, der auf eine Bibliothek mit bitseriellen Operatoren zurückgreift.

Konfiguration und Rekonfiguration

Durch ihre direkte Einbettung in kostensensitive Produkte (Kfz-Mechatronik, Consumer-Elektronik, …) sind Controller stark optimiert bezüglich aller verwendeten Ressourcen. Das bedeutet, dass ihre Soft- und Hardware-Architektur sehr eng auf die jeweilige Anwendung abgestimmt ist und dass alle Ressourcen möglichst vollständig genutzt werden. Der Preis für diese Art der Kostenoptimierung besteht im Verlust fast jeglicher Flexibilität im Fall von Änderungen, welche zusätzliche oder andere als die vorhandenen Hardwareressourcen benötigen. Diese Änderungen können z.B. durch Fehler erzwungen werden, die erst spät in der Entwicklung entdeckt werden, durch Änderungen gesetzlicher Vorschriften, durch die kontinuierliche Weiterentwicklung des Produkts aber auch durch Änderungen in späten Entwicklungsphasen nachdem die Hardware bereits festgelegt wurde. Eine Ursache für die fehlende Flexibilität besteht darin, dass die heute verwendeten Hardware-Ressourcen in Mikro-Controllern und Signalprozessoren sehr komplex aufgebaut sind. Arithmetik-Einheiten zur Bereitstellung von Rechenleistung, Datenspeicher zur Aufnahme von Variablen, Konstanten und Kennfeldern, Kommunikations- oder Prozessschnittstellen zum Informations- und Datenaustausch sind kostspielige Komponenten, die große Teile der verfügbaren Chipfläche belegen. Ein Herunterbrechen dieser komplexen Einheiten in konfigurierbare Elemente feinerer Granularität schafft die Voraussetzung dafür, die vorhandenen Ressourcen wesentlich besser an veränderte Aufgabenstellungen anzupassen. Da konfigurierbare Elemente in FPGA-Architekturen sowohl Logik- als auch Speicherfunktionen übernehmen können, lassen sich durch eine Rekonfiguration auch heute oft auftretende Speicherplatzprobleme entschärfen, da das Speicherplatzangebot in beliebigen Inkrementen an den Bedarf angepasst werden kann. Diese Anpassung der verfügbaren Ressourcen an gegebene Anforderungen wird als Konfiguration, die Anpassung an veränderte Anforderungen wird als statische Rekonfiguration bezeichnet. Daneben treten im Betriebszyklus eines Controllers sehr unterschiedliche Anforderungen auf. So sind z.B. betriebspunktabhängig unterschiedliche Teile der Software aktiv. In diesen Fällen kann eine dynamische Anpassung der implementierten Strukturen eine bessere Ausnutzung der eingesetzten Ressourcen bringen. Diese Anpassung der verfügbaren Ressourcen an dynamisch veränderliche Anforderungen im laufenden Betrieb wird dynamische Konfiguration genannt.

Statische Rekonfiguration

Die statische Rekonfiguration ermöglicht Änderungen, welche aus geänderten Anforderungen bzw. der Weiterentwicklung resultieren oder der Fehlerbeseitigung dienen, bei Wiederverwendung der Hardware. Dies ist besonders interessant vor dem Hintergrund rekonfigurierbarer Analog- und Digitalschnittstellen, die mit verschiedenen Taktraten, Wortbreiten und Protokollen arbeiten können. Dadurch besteht die Möglichkeit die Konfigurierbarkeit aus dem PLD heraus zu tragen und so eine universelle Hardware zu entwickeln, die mehrere Algorithmengenerationen alternativ aufnehmen kann. Dadurch wird die Variantenvielfalt der Hardware reduziert.

Wenn man sich den Zyklus eines Produktes des Maschinen- bzw. Fahrzeugbaus vorstellt, dauert dieser von der Entwicklung über die Produktion bis zum Ende der Bereitstellung von Ersatzteilen bis zu 30 Jahre. Die Produktzyklen der IT-Industrie sind jedoch deutlich kürzer. Vor diesem Hintergrund stellt die Sicherstellung der Versorgung mit programmierbaren Bauelementen ein großes Problem dar. Bei einem umfangreichen Typenbedarf ist das Einlagern dieser Bauelemente teuer, zumal allein durch die jahrelange Lagerung Ausfälle zu beklagen sind. Eine universelle Hardware reduziert dieses Problem deutlich.

Allgemein werden nur Algorithmen in Hardware realisiert die eine entsprechend hohe Performance benötigen. Alle anderen und insbesondere sequentielle Algorithmen werden durch Software realisiert. Demzufolge ist die Nutzung eines Prozessors vorteilhaft. Durch die Verlagerung des Prozessors in Form eines Softcores in den PLD lässt sich die Integrationsdichte bei Reduzierung des Bauelementeaufwandes reduzieren. Bei steigendem Bedarf an Prozessorperformance kann dieser durch Verlagerung von Signalverarbeitungsalgorithmen in Hardware oder durch Integration eines weiteren Prozessorkerns einfach befriedigt werden.

Vor dem Hintergrund einer bestimmten Hardware ergeben sich natürlich einige Randbedingungen hinsichtlich der vorhandenen Ressourcen, die einzuhalten sind. So muss der Logikaufwand in einen entsprechenden Rahmen

passen. Bei Existenz unterschiedliche Taktdomänen ist zu berücksichtigen, dass dies die Platzierung der Logik in bestimmten Bereichen des Chips erzwingt. Wenn spezielle Verbindungsleitungen in Form von Tri-State-fähigen Bussen erforderlich sind, muss eine entsprechende Anzahl von Tri-State-Treibern vorhanden sein. Schließlich ist auch noch auf die Anzahl, Art und Position der I/O-Pins zu achten.

Damit sind zur Nutzung der statischen Rekonfiguration u.a. folgende Probleme zu lösen:

- Modellierung des Ressourcenbedarfs

  Um Komponenten schnell und zielgerichtet verwenden und austauschen zu können, ist Wissen über ihren Ressourcenbedarf schon vor der Synthese erforderlich. Darum sind Modelle erforderlich, die den Logikaufwand, notwendige I/O-Pins und spezielle Signalleitungen wieder spiegeln.

- Spezifikation unter Berücksichtigung von Ressourcenmodellen

  Die erhaltenen Modelle sind in eine Spezifikationsumgebung so zu integrieren, dass sofort nach deren Verwendung der Umfang der verwendeten Ressourcen ersichtlich ist.

- Synthese unter Berücksichtigung von Ressourcenmodellen

  Vor dem Hintergrund der Rekonfiguration ist davon auszugehen, dass bei der Synthese z.B. die Lage der I/O-Pins vorgegeben ist. Diese Vorgaben sind automatisch in die Synthese neuer Implementierungen zu integrieren.

Schnittstellen

Alle gängigen digitalen Kommunikations- und Netzwerkschnittstellen, lassen sich sehr effizient direkt in Logik implementieren. Allerdings werden aus Platzgründen Feldbusschnittstellen hier nicht behandelt, sondern ausschließlich Prozessschnittstellen. Auf der Eingabeseite bestehen grundsätzlich zwei Probleme. Das ist einerseits die Analog-Digital-Wandlung von kontinuierlichen Signalen und andererseits das Bestimmen der Abstände von Flanken. Dies bedarf eines Schmitt-Trigger-fähigen Einganges und einer Timer-Baugruppe (Capture/Compare). Die Implementierung von Timern und Vergleichern stellt im PLD kein Problem dar. Wenn andererseits Eingangssignale mit einer ausreichenden Flankensteilheit vorliegen, kann auf den Schmitt-Trigger verzichtet werden. Auf der Ausgabeseite ist die Situation ähnlich, dort sind binäre Signale zum Schalten von Transistoren, Relais u.a. notwendig, sowie analoge Signale zu generieren.

Damit liegt das Augenmerk auf rekonfigurierbaren AD- und DA-Wandler sowie einem rekonfigurierbaren analogen Front-end. Hier ist besonders das Sigma-Delta-Wandlungsverfahren geeignet, da es nur einen geringen analogen Aufwand benötigt. Ein Sigma-Delta-Wandler wandelt ein analoges Eingangssignal mit Hilfe eine Delta-Sigma-Modulators (Bild 1) in ein hochfrequentes binäres Ausgangssignal mit einem arithmetischen Mittelwert, der dem des Eingangssignals entspricht. Bei der AD-Wandlung wird dieses Signal in einen digitalen Wert umgesetzt, während bei der DA-Wandlung ein Tiefpass daraus ein kontinuierliches Signal erzeugt. Eine gute Beschreibung des Verfahrens ist unter [4] zu finden. In [5] wird die Implementierung von Sigma-Delta-Wandlern in programmierbarer Logik beschrieben.

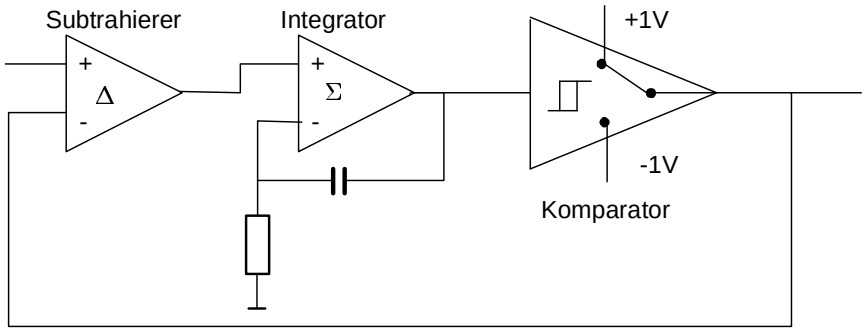

Bild 1 Prinzipieller Aufbau eines Sigma-Delta-Wandlers

Zur Nutzung des Sigma-Delta-Wandlerverfahrens sind die entsprechenden internen digitalen Komponenten bereitzustellen und ein rekonfigurierbares Analog-Frontend ist zu entwickeln.

Dynamische Rekonfiguration

Im Allgemeinen werden im Controller arbeitpunktabhängig- oder betriebszustandsabhängig verschiedene Algorithmen abgearbeitet. Als Beispiel kann hier ein ABS-System betrachtet werden, das während des Bremsvorgangs sehr komplexe Regelungs- und Steuerungsfunktionen auszuführen hat, wogegen die Zwischenzeiten hauptsächlich der Überwachung dienen. Häufig sind Funktionen an externe Ereignisse gekoppelt, wodurch der Rechenzeitbedarf stark vom jeweiligen Betriebsfall abhängt. Als Beispiel sei hier ein Einspritzsystem genannt, das abhängig von der Motordrehzahl zu definierten Kolbenpositionen bestimmte Funktionen auszuführen hat, was u. a. dazu führt, dass bei hohen Drehzahlen fast die gesamte Rechenleistung des Prozessors für diese Aufgabe verbraucht wird.

Gegenwärtig gibt es noch keine Hardwareplattform, die einfach und in Echtzeit die Rekonfiguration erlaubt. Prinzipiell steigt die Rekonfigurationszeit mit dem Umfang der Rekonfiguration, da entsprechend mehr Informationen zu übertragen sind. Einige FPGA-Typen unterstützen schon die partielle Rekonfiguration, jedoch kann dies nicht beliebig erfolgen, sondern nur zeilen- oder spaltenweise.

Ein weiteres Problem stellt die Signalkonsistenz dar, insbesondere wenn Werte aus mehreren Abtastzeitpunkten zu verarbeiten sind, oder die Berechnung länger als eine Tastperiode andauert. Gegenwärtig läuft ein Algorithmus weiter, auch wenn seine Ergebnisdaten nicht genutzt werden. Im Umschaltpunkt liegen die Ergebnisse der umzuschaltenden Algorithmen daher nah bei einander und Unstetigkeiten werden somit vermieden. Wenn die Algorithmen nun ausgetauscht werden, erfolgt in den Ruhepausen keine Berechnung, es stellt sich damit das Problem der Vermeidung von Unstetigkeiten im Umschaltpunkt.

Die intensiven Forschungs- und Entwicklungsanstrengungen weltweit belegen das große Potential rekonfigurierbarer Rechensysteme.

- Entwicklung von Rekonfigurationsmechanismen auf Komponentenebene bzw. auf Systemebene. An dieser Stelle liegt das Augenmerk auf der unterschiedlichen Größe der zu rekonfigurierenden Bereiche.
- Erweitern der Spezifikationsumgebung

  Die dynamische Rekonfiguration muss spezifiziert werden können, d.h. der Austausch der Komponenten muss bekannt sein, dahin ist das Spezifikationswerkzeug zu erweitern.
- Erweitern der Syntheseumgebung

  Die Rekonfiguration setzt voraus, dass die Synthese komponentenweise erfolgt. Zur Steuerung der Synthese sind dafür entsprechende Anweisungen zu generieren.

Anwendungsbeispiel

Die Anwendungsmöglichkeiten der Rekonfiguration sollen am Beispiel eines spurgeführten Fahrzeugmodells gezeigt werden. Das Fahrzeug folgt einer Markierung am Boden. Dies kann sowohl eine farbliche Markierung als auch eine magnetische Spur sein. Je nach Art und Weise der Markierung ist ein entsprechender Sensor mit der dazugehörigen Auswertung anzuschließen. Abweichungen des Fahrzeuges von der Spur sollen ausgeregelt werden, um so die geradeaus Fahrt sicher zu stellen. Zur optischen Erkennung der Regelabweichung e (siehe Bild 2) käme eine CCD-Zeile [8] infrage, welche aufgrund der vielen lichtempfindlichen Photodioden (z.B. 256) mit einer entsprechend hohen Abtastrate (z.B. 256 x 1 kHz) aber geringer Auflösung auszulesen wäre, u.U reicht eine einfachen hell-dunkel Unterscheidung (1 Bit). Dagegen kommt man bei einer magnetischen Spur mit zwei Magnetfeldsensoren aus. Diese erfordern eine geringere Abtastrate (z.B. 2 x 1 kHz) dafür aber eine höhere Genauigkeit (z.B. 12 Bit), da die Signalamplitude dem Abstand entspricht. Bei Existenz einer rekonfigurierbaren Analogschnittstelle kann der Sensor einfach ausgetauscht werden.

Für die Implementierung wurde von dem folgenden einfachen Modell der Lenkung ausgegangen. Wenn Δx der innerhalb einer Zeiteinheit ΔT zurückgelegte Weg ist (proportional zur Fahrzeuggeschwindigkeit v), ergibt sich bei einem Lenkwinkel α eine Verschiebung der Radmitte um Δy.

$$\Delta y = f(\Delta x, \alpha)$$
$$\Delta y(k) = \Delta x(k) \cdot \tan\alpha(k)$$

Diese Näherung gilt nur für genügend kleine ΔT, da dann im Zeitintervall von einer konstanten Geschwindigkeit ausgegangen werden kann. Für üblich Anwendungen ist diese Voraussetzung erfüllt. Damit hängt Δy sowohl vom Lenkwinkel als auch von der zurückgelegten Strecke ab, womit sich ein nichtlineares Verhalten der Regelstrecke ergibt. Sowohl für Δx als auch für Δy wird davon ausgegangen, dass ein Digit einem Millimeter

entspricht. Solche Festlegungen zur Skalierung sind insbesondere vor dem Hintergrund der Rekonfiguration wichtig. Denn sie beeinflussen die Auslegung des Reglers und erlauben nur dann einen einfachen Austausch der Sensoren und Aktoren, wenn die auszutauschenden Komponenten denselben Verstärkungsfaktor haben..

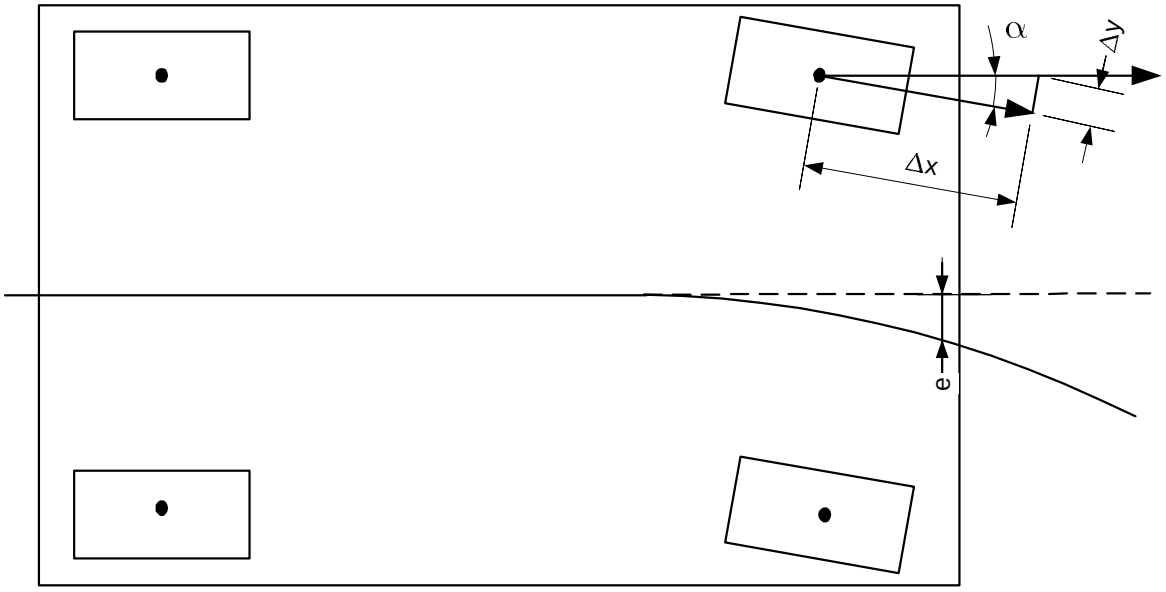

Bild 2 Definition der Größen am Fahrzeug

Um den Einfluss von Δx zu kompensieren, sind die Reglerparameter entsprechend anzupassen. Dazu werden für die Adaption die Regelkreispole vorgegeben und online an Änderungen von Δx angepasst. Somit ergibt sich die im Bild 3 dargestellte Struktur.

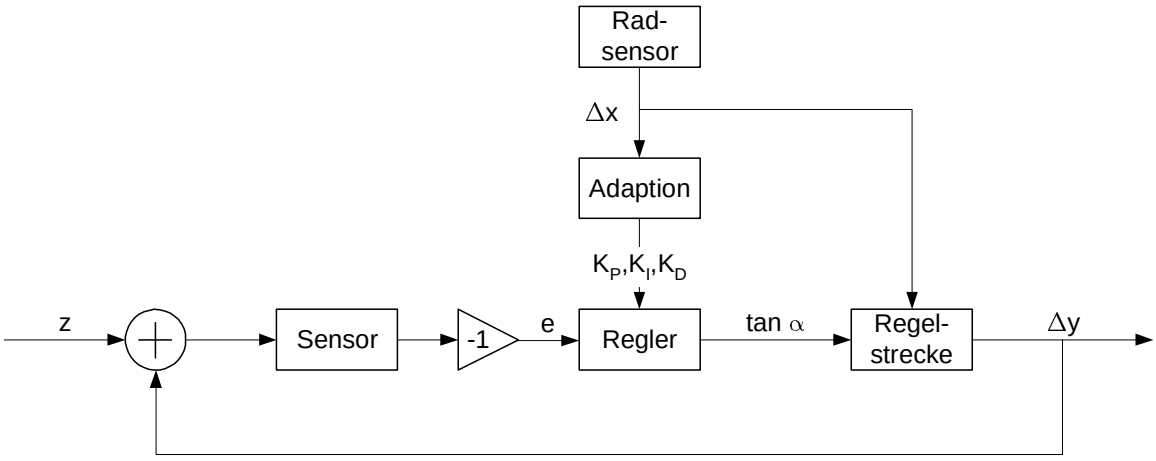

Bild 3 Regelkreis zu Spurführung

Das System wurde unter Verwendung von Simulink/Matlab simuliert. Als Störung wurde der im Bild 4 dargestellte Signalverlauf z verwendet. Eine vom Betrag her zunehmende Rampe bedeutet, dass der Radius der Spur immer kleiner wird. In diesen Phasen kommt es zu einem Geschwindigkeitsfehler im Regelkreis, der aufgrund der Reglerstruktur nicht ausgeregelt werden kann. Während der Zeitabschnitte in denen z konstant aber nicht Null ist, wird der Fehler durch den I-Anteil im Regler kompensiert.

Bild 5 zeigt die Auswirkung der Adaption auf den Reglerparameter K_P. Übereinstimmend mit der persönlichen Erfahrung nimmt der Verstärkungsfaktor mit zunehmender Geschwindigkeit ab, da der Verstärkungsfaktor der Regelstrecke zunimmt..

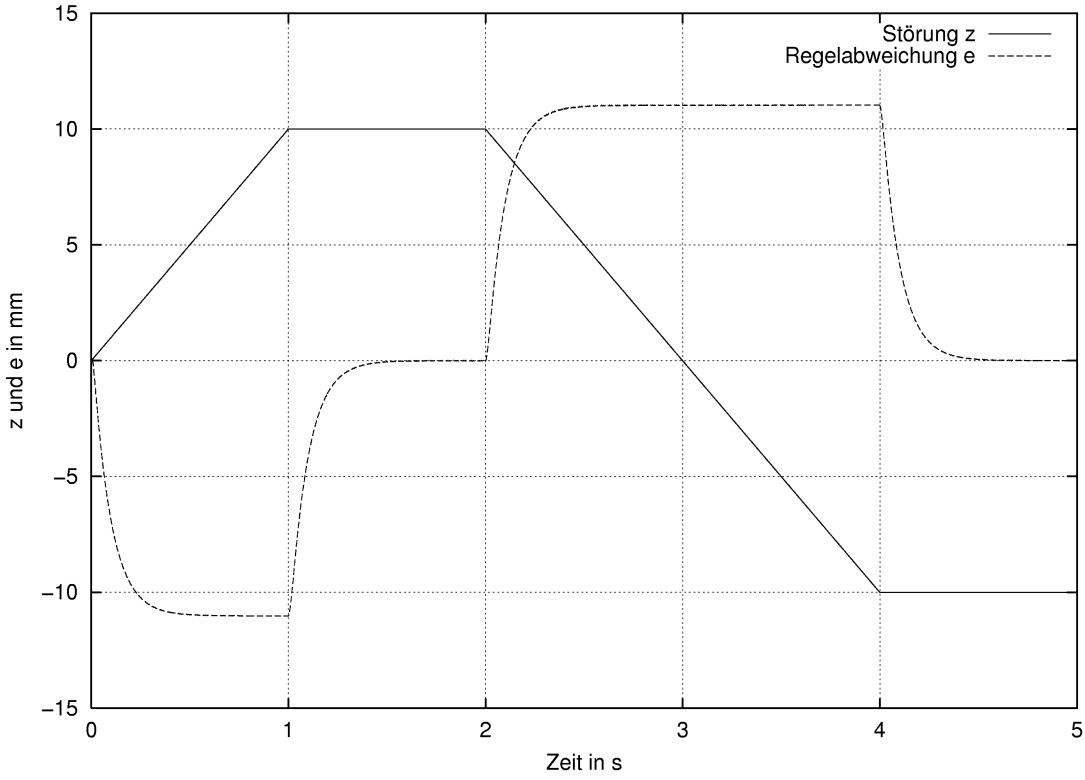

Bild 4 Regelabweichung und Störung

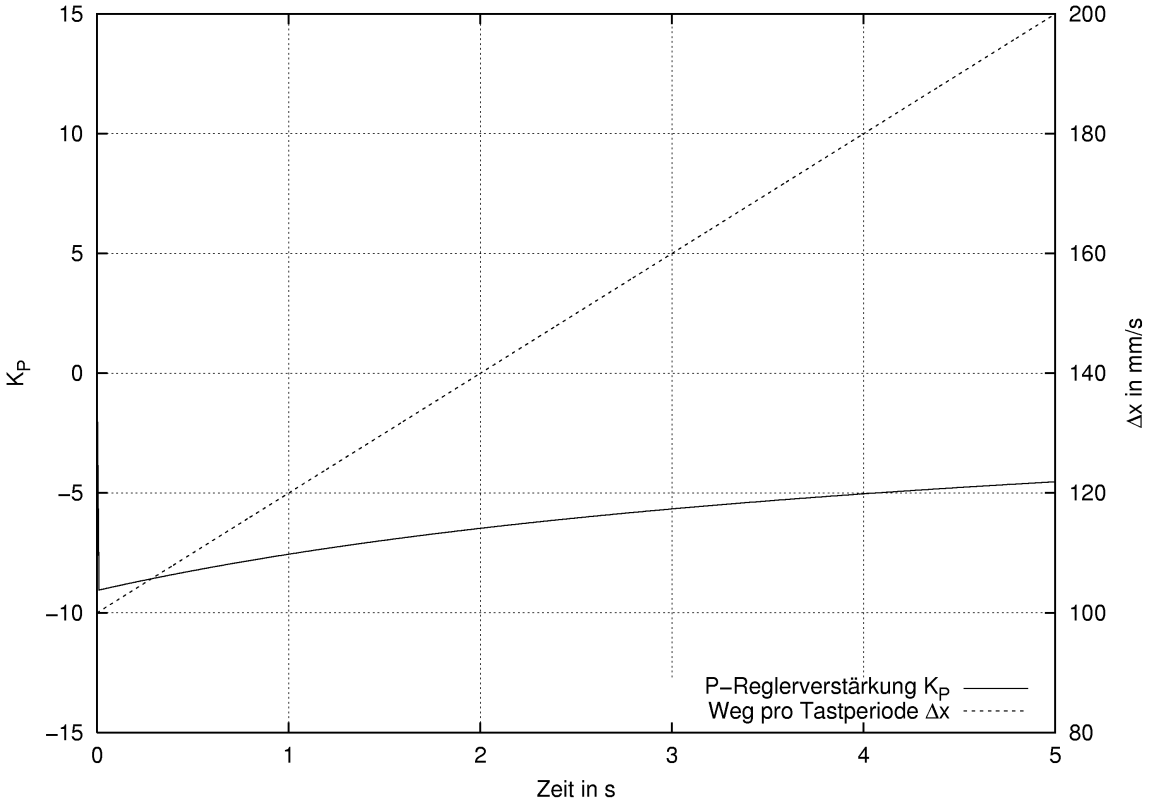

Bild 5 Zusammenhang von Geschwindigkeit und Reglerverstärkung

Durch die geringere Dynamik der Geschwindigkeit im Verhältnis zum Lenkwinkel kann die Adaption (ΔT=10 ms) mit einer geringeren Abtastfrequenz (ΔT) laufen als der Regelalgorithmus (ΔT=1 ms). Da die Adaption auch stark nichtlineare Funktionen benötigt, für die es Software-Bibliotheken gibt, bietet sich eine Realisierung in Software an. Der dazu benötigte Prozessor kann in einem FPGA untergebracht werden. Hier kommt ein 8051-Core [9] zum Einsatz. Dieser benötigt ca. 1400 Flip-Flops auf der Xilinx Virtex Architektur und läuft mit 10 MHz.

Ein auf bitserieller Basis [3] implementierter PID-Regler benötigt ca. 200 Flip-Flops und passt zusammen mit dem Prozessorkern problemlos in einen kleineren FPGA.

Unmittelbar nach dem Einschalten eines Gerätes erfolgt meist ein allgemeiner Funktionstest und eine Kalibrierung der Sensoren. Diese Algorithmen und Strukturen werden im regulären Betrieb nicht mehr gebraucht. Daher bietet sich ein Austausch dieser Algorithmen mit dem Regelalgorithmus in Form einer dynamischen Rekonfiguration an.

Literaturverweise

[1] Paul Master: The Age of Adaptive Computing Is Here, LNCS 2438, p. 1 ff.

[2] Andre DeHon, John Wawrzynek: Reconfigurable computing: what, why, and implications for design automation, Proceedings of the 36th ACM/IEEE conference on Design automation conference, 1999, ISBN 1-58133-109-7, p. 610 – 615, New Orleans, Louisiana, United States, ACM Press.

[3] Reinemann; Kasper: Spezifikation und Synthese von Steuer- und Regelalgorithmen direkt in Hardware; 5. Magdeburger Maschinenbautage; Logos Verlag; Berlin; 2001

[4] Jim Thompson: Care and Feeding of the One Bit Digital to Analog Converter, http://www.ee.washington.edu/conselec/CE/kuhn/onebit/ primer.htm, 1995

[5] A. Sanz, J.L. Garcia-Nicolas and L. Urriza: Implementing Converters in FPLD, pp. 966 ff, LNCS 2438

[6] Reinemann; Kasper: Delay Optimization for Bit Serial Algorithms; International Signal Processing Conference, Dallas, 2003

[7] Auer, A; Rudolf, D. J.: FPGA Feldprogrammierbare Gate Arrays; Huethig; Heidelberg; 1995

[8] Foto Digital Service: http://www.foto-digital.de/glossar_c.html

[9] Dalton Project, University of California, Dept. of Computer Science Riverside, CA 92521, http://www.cs.ucr.edu/~dalton/8051/, Email: dalton@cs.ucr.edu