Rapid-Prototyping integrierter Steuerungssysteme

Frank Golatowski, Jens Hildebrandt, Dirk Timmermann

Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31, 18119 Rostock- Warnemünde
gol@e-technik.uni-rostock.de

Abstrakt. In diesem Beitrag wird eine Methodik zur Entwurfsunterstützung, zur Analyse, zur Evaluierung und zum Test von integrierten Steuerungssystemen, bei der die Einhaltung harter Zeitschranken im Vordergrund steht, beschrieben. Sie ist insbesondere für jene integrierte Steuerungssysteme geeignet, die durch parallel auszuführende Aktivitäten (Tasks, Prozesse) und einer starken Wechselwirkung mit der Umgebung des Systems (reaktive Systeme) gekennzeichnet sind.

Einleitung

Integrierte Steuerungssysteme werden zur Steuerung technischer Prozessen eingesetzt. Mit den klassischen Entwurfstechniken ist es nicht möglich, die Einhaltung der zeitlichen Anforderungen (Echtzeitbedingungen) einer Echtzeitanwendung nachzuweisen. Ein solcher Nachweis ist jedoch notwendig, um Aussagen bezüglich der Deterministik einer Anwendung geben zu können. Er stellt die Basis für ein gezieltes Rapid-Prototyping dar.
Die Echtzeitschedulinganalyse ist ein geeigneter Weg, die Einhaltung von Zeitschranken in Multitasking-Umgebungen nachzuweisen [1, 2, 3, 7, 9]. Allerdings existiert eine große Lücke zwischen der zugrundeliegenden Theorie und deren Einsatz beim Entwurf realer Systeme. Das hier vorgestellte Werkzeug EVASCAN liefert einen Beitrag, die Lücke zu schließen. Es ist ein Hilfsmittel zur Applikationsentwicklung und kann während der Spezifikations-, Entwurfs- und Implementierungsphase eingesetzt werden. Im Beitrag wird das Werkzeug, ausgehend vom zugrundeliegenden System- und Prozeßmodell, beschrieben.

Systembeschreibung

Systemmodell

Ein Echtzeitsystem besteht prinzipiell aus dem steuernden System (control system), dem zu steuernden System (controlled system) und einem Bediensystem. Das Steuersystem führt die notwendigen Berechnungen und Steueranweisungen aus, um die Aktivitäten mittels Computer- und Bedienungsinterfaces des zu steuernden Systems zu organisieren, zu koordinieren und zu garantieren. Das zu steuernde System kann als Umgebung betrachtet werden, die durch das Echtzeitcomputersystem beeinflußt wird. Das zu steuernde System besteht aus mehreren Tasks, die durch das Steuerungssystem ausgeführt werden.
Anwendungstasks können bezüglich ihrer Zeitanforderungen in harte Echtzeittasks (hard real-time, HRT), weiche Echtzeittasks (soft real-time, SRT) und Nicht-Echtzeittasks (non-real-time, NRT) klassifiziert werden. Zeitkritische Tasks haben Zeitschranken (Deadlines, Endtermine). Eine HRT-Task ist eine Task, deren zeitlich und logisch korrekte Ausführung kritisch für das Gesamtsystem ist. Kann eine HRT-Task ihre Aufgabe nicht termingerecht erfüllen, kann das fatale Folgen für das Gesamtsystem haben. Die Antwort einer HRT-Task muß innerhalb einer Zeitgrenze erfolgt sein. Eine SRT-Task weiche Deadlines. Das Einhalten einer weichen Deadline durch eine SRT-Task ist erwünscht. NRT-Tasks sind Tasks, die keine explizite Deadline besitzen und als Hintergrund-Tasks arbeiten.

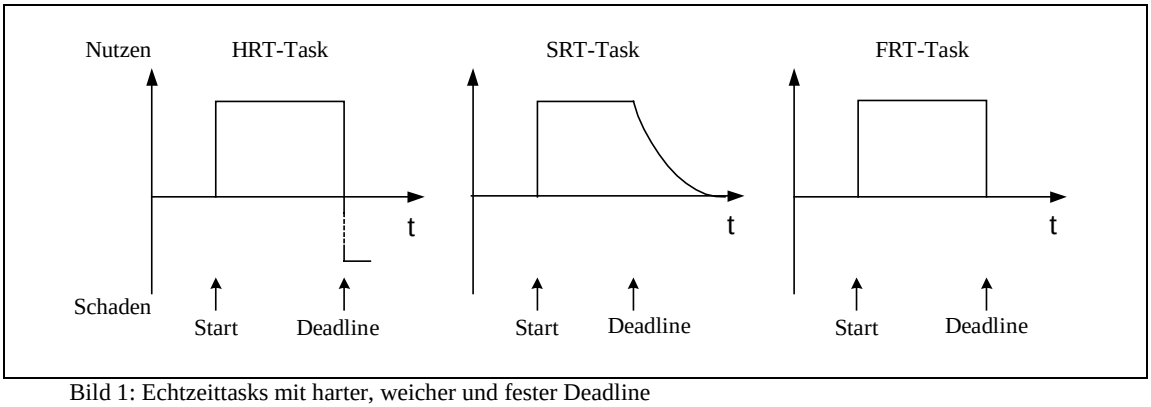

Bild 1: Echtzeittasks mit harter, weicher und fester Deadline

Ein Echtzeitsystem ist charakterisiert durch einen Mix aus HRT-, SRT- und NRT-Tasks. Die Tasks können durch eine Schaden-Nutzen-Funktion charakterisiert werden. Bild 1 zeigt eine HRT- und SRT-Task. Wird die Deadline verpaßt, nimmt der Schaden (HRT) sprunghaft zu bzw. nimmt der Nutzen (SRT) ab.
Zur Leistungsbewertung von Schedulingalgorithmen kommt das Modell der festen (firm) Deadline zur Anwendung. Nur solche festen Echtzeittasks (firm real-time, FRT), die bis zum Erreichen der Deadline ausgeführt werden, nützen dem System.

Modell periodischer, aperiodischer und sporadischer Tasks

Die Tasks eines Systems können in periodische, aperiodische und sporadische Tasks unterteilt werden. Die Unterscheidung ergibt sich aus den zeitlichen Eigenschaften der Tasks.

Periodische Tasks sind Rechentasks, die nach einem festen Zyklus wiederholt gestartet und ausgeführt werden. Sie sind durch die Parameter (C_i, D_i, T_i, I_i) charakterisiert.

- C_i ist die Worst-Case-Rechenzeit
- T_i ist die Periode der Task
- D_i ist die relative Deadline (Endtermin), die bezogen auf die Startzeit der Task innerhalb der Periode, einzuhalten ist.
- I_i ist der Startzeitpunkt der Task

Der Fall $I_i=0$ stellt für harte periodische Task den Worst-Case-Fall dar. Häufig wird von einem vereinfachten Tasksatz ausgegangen: mit $T_i=D_i$ und $I_i=0$ [7, 9].
Aperiodische Tasks können sowohl harte als auch weiche Endtermine besitzen. Es werden zwei Typen aperiodischer Tasks unterschieden. Aperiodische Tasks, die durch eine harte Deadline und einen minimalen Ereignisabstand charakterisiert sind, werden als ***sporadische*** Prozesse bezeichnet. Sie werden durch die Parameter (C_i, D_i, T_i) beschrieben. Die ersten beiden Parameter entsprechen denen der periodischen Prozesse. Durch T_i wird hier die minimale Ankunftszeit angegeben. Weiche aperiodische Tasks besitzen keine Deadline. Sie werden in der Regel als Hintergrund-Tasks ausgeführt. Diese Tasks werden charakterisiert durch die Rechenzeit C_i und eine durchschnittliche Ankunftszeit.
Ein Tasksatz τ ist eine Menge von Tasks $\{T_1, T_2,..,T_n\}$. Wenn für alle Tasks eines Tasksatzes die erste Aktivierung der Tasks zum gleichen Zeitpunkt erfolgt, dann heißt der Tasksatz synchron.
Für weitere Ausführungen zu dieser Thematik sei auf [7] verwiesen. Analytischen Verfahren stellen den Ausgangspunkt für weitergehende Analysemethoden dar. Diese sind wichtige Voraussetzungen für die im nächsten Abschnitt beschriebene Methodik EVASCAN.

EVASCAN: Entwurfsunterstützung für integrierte Steuerungssysteme

EVASCAN ist eine Methodik zur Entwurfsunterstützung, zur Analyse, zur Evaluierung und zum Test von eingebetteten Systemen mit harten Zeitschranken [5, 6].
Das Framework besteht aus mehreren Komponenten, die unter Windows-NT ablaufen. Wesentliche Teilkomponenten sind für die Echtzeitbetriebssysteme LynxOS, SORIX und RMOS realisiert.
Die Komponenten im Einzelnen sind: Schedulinganalyser, Spezifikationseditor und Ausführungseinheit.
Am Anfang des Systementwurfs steht die *Spezifikation* des Systems. Diese ergibt sich aus den physikalischen Anforderungen und kann sowohl graphisch (z.B. Task- und Ressourcengraphen) in Abhängigkeit von der gewählten Entwurfsmethodik als auch mittels einer Meta-Level-Sprache oder tabellenbasiert erfolgen. In EVASCAN erfolgt die Spezifikation mit Hilfe von Tabellen, in denen wesentliche Informationen über die Anwendung enthalten sind.

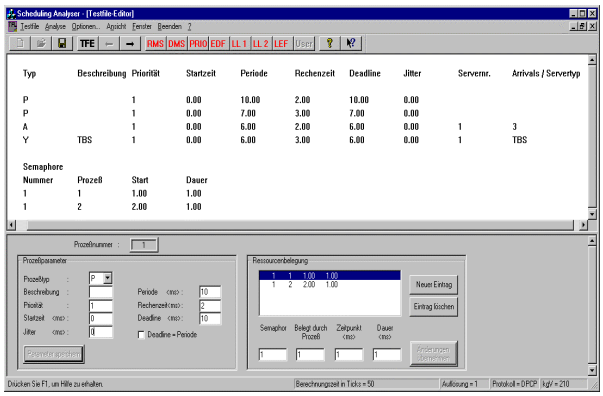

Bild 2: Prozeßsatz-Spezifikation

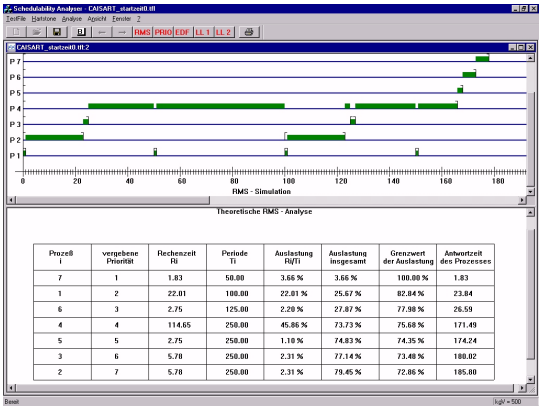

Bild 3: Visualisierung der Prozeßverläufe

Zu diesen Informationen gehören z.B. der Typ einer Task (periodisch, aperiodisch, sporadisch) und die erforderlichen Zeitangaben (Startzeit, Freigabezeit, Worst-Case-Rechenzeit). Bei der Beschreibung wird der Anwender durch einen Spezifikationseditor und Testfileeditor unterstützt (s. Bild 2). Problematisch in Echtzeitsystemen ist die Behandlung zufällig auftretender Ereignisse mit harten Zeitschranken (sporadische Ereignisse). Um eine deterministische Behandlung zu ermöglichen, können Server, die eine vorhersagbare Antwortzeit liefern, eingesetzt werden. Diese Server sind vom eingesetzten Betriebssystem und dessen zugrundeliegenden Schedulingverfahren abhängig.
Durch den Einsatz solcher deterministischer Server werden die aperiodischen oder dynamischen Forderungen durch ein deterministisches statisches oder dynamisches Prioritätenschema berücksichtigt und die Anwendung der Echtzeitscheduling-Analyseverfahren ermöglicht.
Der Schedulinganalyser führt eine *Schedulinganalyse* durch, gibt die Ergebnisse dieser aus und zeigt die Prozeßverläufe graphisch an (siehe Bild 3). Die Schedulinganalyse hilft, die Durchführbarkeit (Feasibility) einer Applikation (eines Prozeßsatzes) zu bestimmen. Durch Interaktion ist es dem Nutzer möglich, Einfluß auf die Gestaltung der Prozeßsätze zu nehmen. Wenn beispielsweise die Zeitschranken eines bestimmten Prozesses nicht garantiert werden können, gibt es verschiedene Möglichkeiten dahingehend Einfluß zu nehmen, daß die Deadlines eingehalten werden:

- Der Prozeßsatz wird neu organisiert
- Der entsprechende Prozeß wird ausgeschlossen
- Die Verwendung eines leistungsstärkeren Prozessors muß erwogen werden
- Es wird ein anderes Schedulingverfahren ausgewählt.

Als Schedulingalgorithmen sind u.a. folgende optimale statische und dynamische Verfahren verfügbar: Rate-Monotonic (RMS), Deadline-Monotonic (DMS), Earliest Deadline First (EDF),

Least Laxity (LL). Durch die Modularität von EVASCAN ist auch die Verwendung selbst entwickelter Schedulingverfahren möglich. Der Schedulinganalyser berücksichtigt Einflußfaktoren (Betriebssystem-Takt, Scheduler-Overhead, Jitter, unterschiedliche Prioritätenniveaus, Interprozeßkommunikation), die in realen Systemen eine Rolle spielen.
Die Leistungsabschätzung eines Echtzeitsystems ist im entscheidenden Maße abhängig vom eingesetzten Modell. Da ein Modell die Realität nicht in seiner Gesamtheit umfassen kann, ist die Einbeziehung realer Systeme notwendig. Deshalb wird der ermittelte Prozeßsatz auf einem Echtzeitbetriebssystem als synthetische Last ausgeführt und beobachtet inwieweit unter realen Bedingungen die Deadlines eingehalten werden (Ausführungseinheit).
Neben diesen reinen Evaluierungsfunktionen steht insbesondere die Einsetzbarkeit der Analyseverfahren und Schedulingalgorithmen für reale integrierte Steuerungssysteme und deren ingenieurtechnische Umsetzung im Mittelpunkt. Deshalb verfügt das System über ein integriertes Hilfesystem [6].
Die Methodik wird eingesetzt beim Entwurf dynamischer integrierter Steuerungssysteme mit Scheduling-Coprozessoren [4, 5, 11]. Sie kann für den Nachweis der Reaktionszeiten in verteilten Mikrocontroller-Steuerungen, die mittels CAN-Bus miteinander vernetzt sind, verwendet werden [8, 10].

Schlußfolgerung

Die Einbeziehung der Schedulinganalyse ermöglicht den Nachweis, daß eine Echtzeitanwendung seine Echtzeitbedingungen erfüllen kann und seine Deadlines einhält. Um eine effektive Anwendung zu ermöglichen, muß das Systemverhalten des verwendeten Betriebssystems bekannt sein. Die Worst-Case-Kontextwechselzeiten, die Interruptsperrzeiten, der Scheduleroverhead, der Systemtimeroverhead müssen bekannt sein, und das Blockieren der Tasks durch das Betriebssystem muß berücksichtigt werden. Eine Herausforderung stellt der Einsatz der beschriebenen Methodik in Automatisierungssystemen dar, die sich der Internettechnologie (z.B. Java, Embedded-Webserver) bedienen, da diese Systeme schwer vorhersagbar und nicht deterministisch sind [12].

Literaturverzeichnis

[1] N. Audsley, A. Burns, R. Davis, K. Tindell, A. Wellings, „Fixed Priority Pre-emptive Scheduling: An Historical Perspective“, Real-Time Systems Journal, Vol. 8, S. 173-198, Kluwer Academic Publishers, 1995

[2] A. Burns, A. Wellings „Real-Time Systems and Programming Languages“, 2. Auflage, Addison Wesley, 1997

[3] G. C. Buttazzo, „Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications“, Kluwer Academic Publishers, 1997

[4] F. Golatowski, J. Hildebrandt, D. Timmermann, „Rapid Prototyping with Reconfigurable Hardware for Embedded Hard Real-Time Systems“, 19th IEEE Real-Time Systems Symposium 98, WIP-Session, Madrid, 2-4.12.1998

[5] F. Golatowski, „Ein Beitrag zur Entwurfsunterstützung, Leistungsanalyse und Leistungsbewertung von Echtzeitsystemen und Echtzeitbetriebssystemen", Dissertation, Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock, Sept. 1998

[6] F. Golatowski, D. Timmermann, „Using Hartstone Uniprocessor Benchmark in a Real-Time Systems Course", Third IEEE Real-Time Systems Education Workshop, Poznan, Polen, Nov. 98

[7] M. Joseph (Hrsg.), „Real-Time Systems: Specification, Verification and Analysis“, Prentice Hall, 1996

[8] H. Kopetz, „Real-Time Systems: Design Principles for Distributed Applications“, Kluwer Academic Publishers, 1997

[9] C. L. Liu, J. W. Layland, „Scheduling Algorithms for Hard Real-Time Environments“, Journal of the ACM, Vol. 20, Nr. 1, S. 46-61, 1973

[10] K. Tindell „Embedded Systems in the Automotive Industry“, In: H. Rogge (Hrsg.) Embedded Intelligence 99, Konferenzband, WEKA-Fachzeitschriften.-Verlag, 1999, S.90-101

[11] Rapid Prototyping mit rekonfigurierbarer Hardware für eingebettete Echtzeit-Systeme mit harten Echtzeiteigenschaften. Forschungsvorhaben im DFG-Schwerpunktprogramm Rapid Prototyping für integrierte Steuerungssysteme mit harten Zeiteigenschaften, 1998

[12] http://www-md.e-technik.uni-rostock.de/forschung/projekte/embedded-web