Rapid-Prototyping eingebetteter Systeme mit harten Zeitbedingungen

Frank Golatowski, Marc Schlegel, Dirk Timmermann

1 Einleitung

In diesem Beitrag wird eine integrierte Methodik und ein Entwicklungssystem zur Entwurfsunterstützung, zur Analyse, zur Evaluierung und zum Test von eingebetteten Systemen mit *harten* Zeitschranken vorgestellt. Sie eignet sich für Systeme, die aus parallelen Prozessen (Tasks) besteht und durch eine starke Wechselwirkung mit der Umgebung des Systems (reaktive Systeme) charakterisiert ist.
Eingebettete Computersysteme sind Real-Time-Computersysteme, die zur Steuerung, Regelung und Beobachtung von technischen Prozessen eingesetzt werden. Die Dynamik der technischen Prozesse erfordert die Einhaltung von Echtzeitbedingungen in Form von Zeitschranken (Deadline, Endtermin). Damit kann die Korrektheit und Sicherheit eines Echtzeitsystems gesichert werden. Mit der Echtzeitschedulinganalyse ist der Nachweis möglich, ob eine Anwendung durchführbar (feasible, schedulable) ist. Eine Anwendung gilt als durchführbar, wenn alle Tasks vor dem Erreichen ihrer Deadlines ausgeführt werden können.

2 Schedulinganalyse von Echtzeitsystemen

In diesem Abschnitt wird das im Institut für Angewandte Mikroelektronik und Datentechnik entwickelte Framework EVASCAN beschrieben. EVASCAN stellt eine Methodik zum Entwurf, zur Analyse, zur Evaluierung und zum Test von eingebetteten Systemen mit "harten" Zeitschranken dar. Hier werden Ziele, Aufbau, Vorgehensweise des Methodik und Ergebnisse des Frameworks vorgestellt. Ein wesentliches Element stellt dort die Schedulinganalyse dar [3].

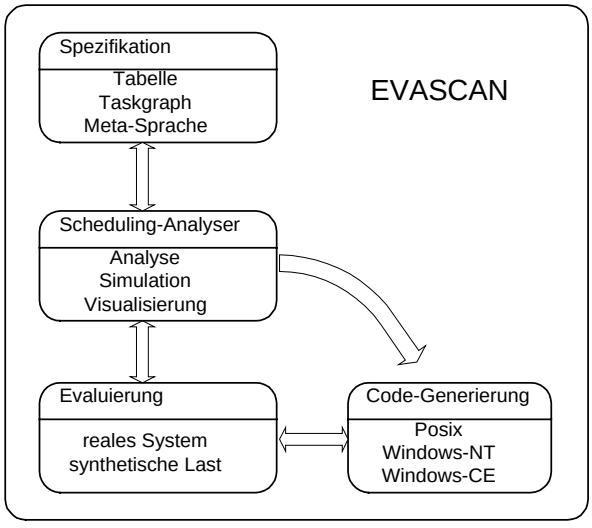

Abbildung 1: Architektur von EVASCAN

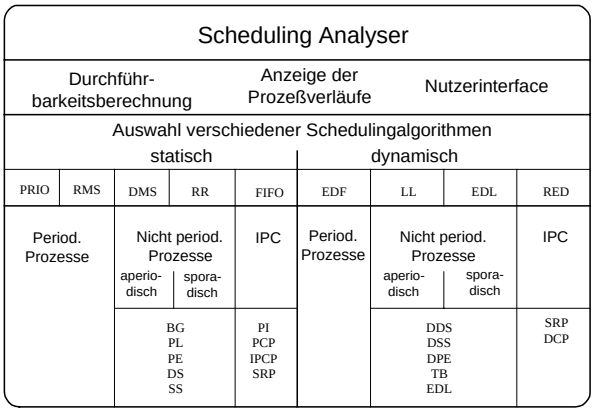

Abbildung 2: Aufbau des Schedulinganalysers

2.1 Ziele

Abbildung 1 zeigt den Aufbau des Frameworks EVASCAN. Es werden folgende Ziele verfolgt:

- Leistungsbewertung von Echtzeitsystemen, Echtzeitbetriebssystemen und Echtzeitcomputersystemen
- Einbindung von Analysemethoden der Real-Time-Schedulingtheorie in den Entwurf von Echtzeitsystemen mit harten Zeitbedingungen mit Betriebssystemunterstützung
- Unterstützung des statischen und dynamischen Prioritätenschedulings [1, 5]
- Berücksichtigung der für Echtzeitsysteme relevanten Synchronisationsprotokolle [4]
- Integration neuer Betriebssystem- und Internet-Technologien
- Bereitstellung eines leistungsfähigen Hilfesystems zur Vermittlung der integrierten Prinzipien, Algorithmen und Verfahren
- Erweiterbarkeit der Methodik durch Einsatz objektorientierter Technologie
- Generierung von Quellcode-Templates für Standard-Echtzeitbetriebssysteme (POSIX 1003.1c)

2.2 Beschreibung des Systems

Das Framework besteht aus mehreren Komponenten, die unter Windows-NT ausführbar sind. Teilkomponenten sind für die Echtzeitbetriebssysteme LynxOS, SORIX und RMOS realisiert.
Am Anfang des Systementwurfs steht die *Spezifikation* des Systems. Diese ergibt sich aus den physikalischen Anforderungen und kann in Abhängigkeit von der gewählten Entwurfsmethodik sowohl graphisch (z.B. Task- und Ressourcengraphen) als auch mittels einer Meta-Level-Sprache bzw. tabellenbasiert erfolgen. In EVASCAN erfolgt die Spezifikation mit Hilfe von Tabellen, in denen wesentliche Informationen über die Anwendung enthalten sind. Dazu gehören z.B. der Tasktyp (periodisch, aperiodisch, sporadisch, Synchronisationsserver) und die erforderlichen Zeitangaben (Startzeit, Freigabezeit, Worst-Case-Rechenzeit, Jitter).
Der Schedulinganalyser (Abbildung 2) führt auf der Basis dieser Beschreibungsform eine *Schedulinganalyse* durch, gibt deren Ergebnisse aus und zeigt die Taskverläufe graphisch an. Mit der Schedulinganalyse wird bestimmt, ob die untersuchte Applikation (Tasksatz) im Sinne der Echtzeitfähigkeit durchführbar (feasible) ist (s.o.). Durch Interaktion ist es dem Nutzer möglich, Einfluß auf die Gestaltung der Tasksätze zu nehmen. Wenn beispielsweise die Zeitschranken eines bestimmten Prozesses nicht garantiert werden können, gibt es verschiedene Möglichkeiten den Entwurf so zu verändern, daß die Deadlines eingehalten werden:

- Der Tasksatz wird neu organisiert
- Die entsprechende Task wird ausgeschlossen
- Die Verwendung eines leistungsstärkeren Prozessors muß berücksichtigt werden
- Es wird ein anderes Schedulingverfahren ausgewählt.
- Die Ermittlung des BU- Wertes (breakdown utilisation) ist möglich. Das ist die maximale Prozessorauslastung, bei der alle Deadlines eines Tasksatzes noch eingehalten werden. Sie kann bestimmt werden, indem die Last eines zuteilbaren Tasksatzes erhöht wird bzw. indem die Last eines nichtzuteilbaren Tasksatzes erniedrigt wird. Aus dem ersten Fall ergibt sich die Systemreserve und aus dem zweiten die erforderliche Leistungserhöhung der Hardware.

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 des Systems ist auch die Verwendung selbst entwickelter Schedulingverfahren möglich ist. Dazu ist lediglich das Einbinden einer entsprechenden Windows-DLL-Funktion notwendig.

Abbildung 3: Visualisierung der Taskverläufe und Spezifikation

Mit dem Schedulinganalyser kann weiterhin ermittelt werden, wie Prozesse mit nicht regulären Ankunftszeiten (aperiodisch, sporadisch) durch Server bedient werden können und somit eine Vorhersagbarkeit der Taskverarbeitung möglich ist. Aperiodische und dynamische Forderungen können sowohl durch statische als auch durch dynamische Prioritätenschemata behandelt werden. Implementiert sind die Verfahren Deferrable (DS) und Sporadic Server (SS) sowie die dynamischen Realisierungsformen Dynamic Sporadic Server (DSS), Deadline Deferrable Server (DDS) und Total Bandwidth Server (TBS) (s. Abbildung 2) [4, 5].
Der Schedulinganalyser berücksichtigt Einflußfaktoren (Betriebssystemtakt, Scheduler-Overhead, Jitter, unterschiedliche Prioritätenniveaus, Interprozeßkommunikation), die in realen Systemen eine Rolle spielen. In einer Datenbank sind Informationen über die realen Werte abgelegt, wie z.B. Kontextwechselzeiten, Laufzeiten von Systemrufen, aber auch zugehörige Informationen über die Echtzeitrechnerkonfiguration, wie z.B. Prozessortyp, verwendetes Bussystem etc. Die Datenbank verfügt über eine Standard-Schnittstelle (ODBC, SQL) und kann in Intranets eingebunden werden, so daß registrierte Anwender Informationen in diese eingeben können. 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 Tasksatz auf einem Echtzeitbetriebssystem als synthetische Last ausgeführt und beobachtet, inwieweit die Deadlines unter realen Bedingungen eingehalten werden. Ergebnisse zu diesem Teil der Methodik liegen für die Betriebsysteme LynxOS, SORIX und Windows-NT vor [2].
Neben diesen reinen Evaluierungsfunktionen steht insbesondere die Einsetzbarkeit der Analyseverfahren und Schedulingalgorithmen in realen eingebetteten Systemen und die ingenieurtechnische Umsetzung im Mittelpunkt. Für die verschiedenen Verfahren können Quellcode-Templates berücksichtigt werden. An diesem Teil des Systems sind weitere Arbeiten erforderlich.

3 Beispielanwendung

An einem Beispiel soll die Umsetzung der Methodik erläutert werden. Es wird eine Beispielanwendung angenommen, die auf einem Industrie-PC mit 80486-Prozessor mit 33 MHz läuft. Als Echtzeitbetriebsystem wird LynxOS verwendet. Task T_A (P=1ms, C=0.4ms), Task T_B

(6ms, 1.4ms) und Task T_C (35ms, 4ms) sind drei periodische Tasks, die durch Periode P und Worst-Case-Rechenzeit (computation time) C beschrieben werden. Die Kontextwechselzeiten sind in den Rechenzeiten der Tasks enthalten. Task T_A und T_C teilen sich eine Ressource, die durch das Semaphor S1 geschützt wird. Die Rechenzeit im kritischen Abschnitts beträgt 0.2 ms. Task T_B nutzt Semaphor S2, der zugehörige kritische Abschnitt hat eine Dauer von 0.3 ms. Die Taktrate der Systemuhr beträgt 1 ms und die Laufzeit 0.06 ms. Hardware-Interrupts haben eine höhere Priorität als Tasks. Entsprechend der PC-Architektur hat der Systemtimer die höchste Priorität und bekommt in der Systemanalyse die Interruptpriorität 0 zugewiesen. Zusätzlich können Festplatte, Netzwerkkarte und Tastatur das Tasksystem unterbrechen. Die zugehörigen Unterbrechungszeiten sind durch das Betriebssystem begrenzt: In einer kurzen Interruptserviceroutine, deren Laufzeit maximal 20 µs beträgt, werden die Ereignisse registriert. Die Behandlung der Ereignisse erfolgt aber in einer Interrupttask (Kernel-Level-Thread). Bei einer Interrupttask unterliegt die Behandlung des Ereignisses der Prioritätensteuerung des Betriebssystems. Da die Interrupttasks eine niedrigere Priorität haben als die Real-Time-Tasks (T_A, T_B, T_C), werden diese maximal durch die begrenzte Dauer der ISR (3*20µs) blockiert. Mit diesen Informationen läßt sich die Schedulinganalyse ausführen. Es kann gezeigt werden, daß die drei Real-Time-Tasks ihre Deadlines treffen. Im Worst-Case beträgt die Antwortzeit der Task T_C 15.82 ms. Erst danach erfolgt die Behandlung der im Burst eingetroffenen Ereignisse in Kernel-Threads.

4 Schlußfolgerung

Die Einbeziehung der Schedulinganalyse ermöglicht den frühzeitigen 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 etc. müssen bekannt sein und das Blockieren der Tasks durch das Betriebssystem muß berücksichtigt werden. Diese Informationen werden in der hier vorgestellten Methodik über eine Datenbank bereitgestellt. Die Methodik wird eingesetzt zur Entwurfsunterstützung harter Echtzeitsysteme.

5 Literaturverzeichnis

[1] Audsley, N., Burns, A., Wellings, A.-J.: Deadline Monotonic Scheduling Theory and Application. Control Eng. Practice, 1/1993, S. 237-250

[2] Golatowski, F., Timmermann, D.: An Evaluation and Simulation Technique for Real-Time Operating System“, Embedded Computing Conference, Paris, 1996

[3] Joseph, M.: Real-time Systems- Specification, Verification and Analysis. Prentice Hall, 1996

[4] Rajkumar, R., Sha, L., Lehoczky, J. P.: Priority Inheritance Protocols: An Approach to Real-Time Synchronisation. IEEE Transactions on Computers, 9/1990, S. 1175-1185

[5] Spuri, M., Buttazzo, G.: Scheduling Aperiodic Tasks in Dynamic Priority Systems. The Journal of Real-Time Systems, 10/1996, S. 179-210

Verfasser: Dipl.-Ing. Frank Golatowski, Dipl.-Ing. Marc Schlegel, Prof. Dr. Dirk Timmermann,
Universität Rostock, Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
18051 Rostock
gol@e-technik.uni-rostock.de