Leistungsbewertung von Echtzeitbetriebssystemen und Echtzeitsystemen

Frank Golatowski, Steffen Dolling, Dirk Timmermann
{gol, dol, dtim}@baltic.e-technik.uni-rostock.de
Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard- Wagner- Str. 31
18119 Rostock- Warnemünde

1 Einleitung

Die Angabe von Leistungsparametern ist für Echtzeitbetriebssysteme von entscheidender Bedeutung, da diese Systeme häufig in sicherheitskritischen Umgebungen eingesetzt werden. Auf der Basis dieser Leistungsparameter ist der Nachweis möglich, daß ein Prozeßautomatisierungssystem in seiner Gesamtheit über ein deterministisches Gesamtverhalten verfügen muß. Diese Werte fließen ein in den gesamten Entwicklungsprozeß von Prozeßautomatisierungssystemen, der mehrere Entwicklungsstufen durchschreitet. Prozeßautomatisierungssysteme sind komplexe Echtzeitsysteme in denen die Richtigkeit eines Systems nicht nur von den logischen Ergebnissen abhängig ist, sondern auch vom Zeitpunkt an dem diese Ergebnisse produziert werden.
In diesem Beitrag werden Performance-Meßmethoden, Bewertungsmethoden und Kennzahlen vorgestellt. Dabei wird herausgestellt inwieweit diese Methoden eine Aussagefähigkeit über die Leistung von Echtzeitbetriebssystemen ermöglichen. Am Beispiel von drei Echtzeitbetriebssystemen (Lynx, RMOS, SORIX) werden Ergebnisse zu diesen Meßmethoden vorgestellt.
Des weiteren wird das Evaluierungs- und Schedulability- Tool EVASCAN vorgestellt. Mit der Entwicklung dieses Tools berücksichtigen wir einen Trend bei der Entwicklung komplexer Automatisierungssysteme, der bei der Entwicklung von Echtzeitsystemen eine größere Rolle spielt: die Schedulinganalyse. Diese kann auf der Basis der ermittelten Kennzahlen konkrete Ergebnisse über die Durchführbarkeit bestimmter Prozeßabläufe in einem Prozeßsystem liefern.

2 Meßmethoden und Kennzahlen von Echtzeitbetriebs-systemen

Echtzeitbetriebssysteme zeichnen sich durch besondere Eigenschaften aus, die sie von der Klasse universeller und Standardbetriebssysteme unterscheidet. Diese betreffen insbesondere das Interruptsystem, die Zeitsteuerung, das Scheduling, das Exceptionhandling und die Fehlertoleranz.
Eine mögliche Methode Aussagen über die Leistungsfähigkeit eines Echtzeitbetriebssystems zu gewinnen, ist die Ermittlung spezieller Zeiten.

Die Bestimmung solcher Parameter erfolgt in der Regel nach sehr unterschiedlichen Gesichtspunkten und entspringt häufig einer konkreten Situation. Es hat sich bis heute keine Standardmethode zur Leistungsevaluierung von Echtzeitbetriebssystemen durchgesetzt.

2.1 Klassifikation von Monitoring-Methoden

Zur Leistungsmessung werden unterschiedliche Monitoring-Methoden verwendet. Diese lassen sich ausgehend von der Art der Implementierung in folgende drei Kriterien unterscheiden

1. Ein *Hardwaremonitor* besteht aus speziellen Hardwareeinheiten, die an das zu untersuchende System angepaßt sind. Ein Hardwaremonitor beeinflußt das zu untersuchende nicht.
2. Ein *Softwaremonitor* besteht aus Softwaremodulen, die im zu untersuchenden System eingebettet sind. Dadurch wird das zu untersuchende System beeinflußt. Softwaremonitore haben den großen Nachteil, daß wenn die Ermittlung hoher zeitlicher Auflösungen erforderlich ist, dann lassen sich mit den Modulen des Softwaremonitors nur durchschnittliche Werte ermitteln.
3. Ein Hybridmonitor besteht sowohl aus Software als auch aus Hardwarekomponenten.

2.2 Klassifikation von Echtzeit-Benchmarks

Im Bereich der Echtzeitsysteme und Echtzeitbetriebssysteme können folgende Echtzeit-Benchmarks unterschieden werden:

1. feinkörnige Benchmarks [1], [2]

2. applikationsorientierte Benchmarks [3], [4]

3. simulationsbasierende Benchmarks [5]

Darüber hinaus sind auch Kombinationen dieser Testmethoden möglich.

3 Leistungsbewertung von Echtzeitbetriebssystemen

In den folgenden Abschnitten werden Ergebnisse dargestellt, die wir bei der Evaluierung von Echtzeitbetriebssystemen gewonnen haben.
Dazu haben wir die Methode des Softwaremonitorings verwendet und diese eingesetzt bei der Durchführung feinkörniger systemnaher Benchmarks und bei der Ausführung eine applikationsbasierten Benchmarks.
Mit den ermittelten feinkörnigen Zeiten, die den Definitionen im Echtzeitbenchmark „Rhealstone“ entsprechen, führen wir einen Vergleich verschiedener Betriebssyteme durch.
Da feinkörnige Benchmarks nicht das Gesamtverhalten betrachten, haben wir die Evaluierung auf der Basis eines applikationsorientierten Benchmarks durchgeführt. Die Konzeption und die Präsentation der Ergebnisse erfolgt im Abschnitt 3.2.

3.1 Feinkörnige Benchmarks

Bei der Leistungsbewertung auf der Basis feinkörniger Test- und Analysemethoden werden Absolutzeiten ermittelt.
Der bekannteste feinkörnige Benchmark im Echtzeitbereich ist der Rhealstone. Im Rhealstone-Benchmark werden sechs Zeiten bestimmt, die die Echtzeiteigenschaften eines Betriebssystems charakterisieren (s. Abbildung 1). Der entscheidende Nutzen den dieser Benchmark bringt, ist die Definition dieser Zeiten. Aus den ermittelten Werten kann die eigentliche „Rhealstone Performance Number“ durch Addition der einzelnen Werten ermittelt werden. Der so gewonnene Wert hat aber nur theoretische Bedeutung, da die Aussagekraft der einzelnen Parameter größer ist.
Wir haben diesen Benchmark für die Betriebssysteme SORIX, Lynx-OS und RMOS implementiert. Ausgangspunkt unserer Untersuchungen war dabei die Verwendung einer identischen Hardwareplattform für unterschiedliche Echtzeitbetriebssysteme. Deshalb wurden diese Betriebssysteme auf der gleichen Hardwareplattform installiert. Es wurde großer Wert auf Portabilität gelegt, und es sollte keine zusätzliche Hardware Verwendung finden.
Das untersuchte Rechensystem ist charakterisiert durch folgende Parameter:

- Prozessor: INTEL 80486, Taktfrequenz: 33 MHz, 2nd Level Cache: 256 kBytes,
- Hauptspeichergröße: 16 MBytes

Rhealstone Metric

1. Task switching time t_{TS}
2. Preemption Time t_P
3. Interrupt latency time t_{IL}
4. Semaphore shuffling time t_{SS}: Zeit, vom Freigeben eines Semaphors durch einen Prozeß bis zum Neubelegendurch einen anderen.
5. Deadlock breaking time t_{DB}:Zeit, die benötigt wird, um einen Deadlock zu lösen
6. Intertask Message time t_{IM} :Zeitdauer für die Übertragung eines Message

Rhealstone-Wert

$R = f_1 + f_2 + f_3 + f_4 + f_5 + f_6$

Gewichteter Rhealstone-Wert

$R_W = c_1f_1 + c_2f_2 + c_3f_3 + c_4f_4 + c_5f_5 + c_6f_6$

Abbildung 1: Die Rhealstone-Charakteristik

	Interrupt Latency Time	*Task Switch Time*	*Preemption Time*	*Intertask Message Latency Time*	*Semaphor Shuffling Time*	*Deadlock Breaking Time*
RMOS3-PC1	10,1	4,0	4,1	5,8	4,4	6,2
RMOS3 für Windows	31,1	23,0	23,4	32,0	23,0	29,0
Lynx	17,6	27,0	105,0	337,0	113,5	n.e.
Unix SV Rel. 4 (SORIX)	n.e	137,0	480,0	755,6	450,0	n.e

Tabelle 1: Ergebnisse des Rhealstone- Benchmarks für vier Echtzeitbetriebssysteme (Werte in µs)

Es ist ersichtlich, daß das proprietäre Echtzeitbetriebssystem RMOS3-PC1 die besten Zeiten erreicht. Die dort ermittelten Werte stehen in einem ausgewogenen Verhältnis zueinander. Wenn RMOS3 in einer MS-Windows-Umgebung läuft, verschlechtern sich die einzelnen Werte um das 3- bis 6-fache. Mit dem System Lynx steht ein leistungsfähiges Echtzeit-Unix-Betriebssystem zur Verfügung, das allerdings bezüglich der gemessenen Zeiten nur für die Interruptlatenz und die

Taskwechselzeit in den Bereich der RMOS3 für Windows-Zeiten kommt.
Die Angabe der ermittelten Werte ermöglicht nur eine eingeschränkte Aussagefähigkeit über die Leistungsfähigkeit in sicherheitskritischen Systemen, da mit diesem Benchmark nur die Bestimmung durchschnittlicher Werte möglich ist. Darüber hinaus können auf der Basis dieser Werte keine Aussagen zum Verhalten des Systems unter Lastbedingungen gemacht werden.
Zusammenfassend ergeben sich folgende Nachteile beim Einsatz feinkörniger Benchmarks:

1. Bei universellen Benchmarks werden in der Regel durchschnittliche Werte ermittelt. Das ist dem Fakt geschuldet, daß die implementierten Zeitmeßfunktionen eines Betriebssystems nicht hinreichend hochauflösend sind. Von besonderem Interesse sind aber Worst-Case-Werte. Ermittlung von Worst-Case-Werten mittels Software gestaltet sich jedoch aus diesem Grund als schwierig. Eine genaue Methode zur Bestimmung relevanter Zeiten ist mit Hardware-Unterstützung möglich: Es werden Analyser, Hardware- bzw. Hybrid-Monitore verwendet. Der Einsatz ist aber sehr aufwendig und wenig portabel.
2. Selbst bei Ermittlung von Worst-Case-Zeiten mit Hardwareunterstützung ist die Frage des Einflusses der Last auf diese Zeiten nicht geklärt. Die Angabe bestimmter Zeitwerte ermöglicht zwar eine grobe Einschätzung der Leistungsfähigkeit eines Betriebssystems, das Verhalten unter bestimmten Lastsituationen ist jedoch nicht geklärt.

3.2 Applikationsorientierte Bewertung

Ein interessanter Ansatz, um das Gesamtverhalten eines Echtzeitsystems bzw. Echtzeitbetriebssystems zu analysieren auch unter Berücksichtigung verschiedener Lastkomponenten, stellt der Hartstone-Benchmark [4] dar. Dieser Benchmark wurde an der Carnegie Mellon Universität entwickelt. Der Hartstone-Benchmark zielt auf die *Definition* von Anforderungen im Echtzeitumfeld. Das HUB- (Hartstone Uniprocessor Benchmark) Modell betrachtet ein Real-Time-System als einen Satz von periodischen, aperiodischen (sporadischen) und Synchronisations-(Server-) Tasks.
Der Benchmark besteht aus fünf unterschiedlichen Testserien zu denen mehrere Experimente gehören. Ausgangspunkt einer Testserie ist ein Basistasksatz (s. Tabelle 2), der charakterisiert ist durch die Anzahl der Tasks, die Priorität der Tasks, deren Ankunftsrate, deren Arbeitslast, deren Ausführungszeit und deren Deadlines.
Die verschiedenen Experimente, die aus dem Rate- Monotonic und Earliest Deadline First Algorithmus abgeleitet sind, verwenden ein Sammlung von Tasks, die aus

- **(1)** nur periodischen Tasks mit harmonischen Perioden,
- **(2)** nur periodischen Tasks mit nicht harmonischen Perioden,
- **(3)** sowohl periodischen als auch aperiodischen Tasks,
- **(4)** periodischen harmonische Tasks und einer Synchronisationstask, mit der die anderen Tasks sich zu bestimmten Intervallen synchronisieren, und
- **(5)** einer Kombination von periodischen, aperiodischen und Synchronisationstasks bestehen.

Die Experimente in diesem Benchmark werden so ausgeführt, daß bestimmte Prozeßkenngrößen schrittweise erhöht werden, während die anderen konstant

gehalten werden. Der Abbruch einer Testreihe erfolgt, wenn eine bestimmte Anzahl an Endterminen (Deadlines) nicht mehr eingehalten werden kann bzw. wenn die Dauer einer Testzeit überschritten wird. Die Benchmarks sind nützlich zur Evaluierung des Gesamtsystems.
Ausgehend von der Idee des Hartstone-Benchmarks haben wir das Evaluierungssystem EVASCAN entwickelt. Mit EVASCAN ist die Ausführung aller im Harstone-Benchmark definierten Testserien möglich. Mit der Durchführung dieser Testserien läßt sich das konkrete Verhalten eines Betriebssystems unter steigenden Lastbedingungen bei Berücksichtigung der zeitlichen Bedingungen abschätzen. Da die Testserien aus der Echtzeit-Schedulingtheorie hergeleitet wurden, kann somit die Abweichung des realen Verhaltens vom theoretischen ermittelt werden Damit besteht das Ziel der Ausführung dieser Testserien in der Untersuchung kommerzieller Echtzeit-Betriebssysteme.
Im weiteren sollen Ergebnisse vorgestellt werden, die mit dem Echtzeit-UNIX-Betriebssystem SORIX gewonnen wurden. Entsprechend dem Grundkonzept der einzelnen Experimente wird eine analysierbare synthetische Arbeitslast ausgeführt.

3.2.1 Eigenschaften der zugrundeliegenden Arbeitslast

Jede periodische Task muß als Reaktion auf ein Ereignis eine bestimmte Arbeit verrichten. Dafür wurde eine synthetische Arbeitslast gewählt. Es handelt sich dabei um eine Variante des bekannten Whetstone-Benchmarks, bei der auf einige Berechnungsmodule verzichtet wird (Small-Whetstone). Durch die damit verbundene geringere Anzahl von Operationen kann die Arbeitslast einer Task feiner eingestellt werden. Um eine möglichst allgemeingültige Last zu erhalten, sind die Operationen aus den folgenden repräsentativen Teilgebieten zusammengestellt worden:

- Ganzzahlige Berechnungen (Addition, Subtraktion, Multiplikation und Division),
- Gleitkommaberechnungen (Kosinus-, Logarithmus-, Exponential- und Wurzelfunktionen),
- Funktionsaufrufe,
- Feldzugriffe,
- Zeigeroperationen.

Der Small- Whetstone ist dadurch charakterisiert, daß in jedem Durchlauf etwa 1000 Operationen ausgeführt werden, wofür im weiteren die Abkürzung 1 KWI verwendet wird (Kilo-Whetstone-Instructions). Die von einer periodischen Task je Periode zu verrichtende Arbeit UKWIPP wird dementsprechend in KWIPP ausgedrückt (Kilo-Whetstone-Instructions per Period), die je Sekunde zu verrichtende Arbeit UKWIPS in KWIPS (Kilo-Whetstone-Instructions per Second):

$$U_{KWIPS} = f * U_{KWIPP} = \frac{1}{T} * U_{KWIPP}$$

Dabei ist T die Ausführungsperiode der Task und f die Frequenz, mit der die Ereignisse eintreffen.

3.2.2 Beschreibung des Experimentes PH-1

Am Beispiel des Experimentes PH-1 der PH- Serie des Benchmarks soll auf die

Vorgehensweise bei der Leistungsevaluierung von Echtzeit-Unix-Systemen eingegangen werden. Die PH- Testserie besteht aus einem Basistasksatz von 5 periodischen Tasks, deren Taskfrequenzen ein ganzzahliges Vielfaches der Frequenz jeder niederfrequenten Task sind- die Frequenzen stehen in einem harmonischen Verhältnis zueinander. Die Task sind unabhängig, d.h. es erfolgt keine Synchronisation und keine Kommunikation zwischen ihnen. Tasks sind konkurrierend und bewerben sich um Prozessorzeit. Jede Task ist gekennzeichnet durch deren Frequenz (Periode), Bearbeitungszeit, Deadline und Arbeitslast. Jede Task führt bei Ihrer Aktivierung jeweils einmal die Basislast von einem KWIPP aus. Ausgehend vom Basistasksatz (s. Tabelle 2) wird bei jedem Schritt die Frequenz der 5. Task um den Betrag der Frequenz der 4. Task erhöht. Die anderen Kenngrößen des Basissatzes bleiben konstant. Mit der Erhöhung seiner Frequenz führt diese Task im Laufe des Experimentes eine immer höhere Arbeitslast aus. Der Test wird beendet, wenn eine bestimmte Zahl von Endterminen verfehlt bzw. überschritten wurde. Dabei gilt ein Endtermin (Deadline) als verfehlt, wenn die abzuarbeitende Routine nicht bis zur erneuten Aktivierung der Task abgearbeitet wird, d.h. hier gilt die Annahme, daß der Endtermin gleich der Periode ist. Da die Frequenz der 5. Task dabei relativ groß wird, kann man mit dieser Testreihe Informationen darüber gewinnen, wie gut das System in der Lage ist, schnell zwischen verschiedenen Tasks umzuschalten. Aus den Ergebnissen dieser Reihe lassen sich deshalb unter anderem auch Aussagen über die Auflösung der zugrundeliegenden Zeitbasis gewinnen.

Task-Nummer	Startzeit <s>	Dauer <s>	Priorität	Frequenz <Hz>	Arbeitslast je Periode <KWIPP>	Arbeitslast je Sekunde <KWIPS>
1	5	10	0	1.00	512	512
2	5	10	1	2.00	256	512
3	5	10	2	4.00	128	512
4	5	10	3	8.00	64	512
5	5	10	4	16.00	32	512
Gesamt:	5	10		31.00		2560

Tabelle 2: Definition des Basis-Tasksatzes für die PH-Testreihen

3.2.3 Meßergebnisse

Die Ergebnisse des zuvor beschriebenen Experimentes PH-1 wurden in zwei Diagrammen zusammengefaßt.

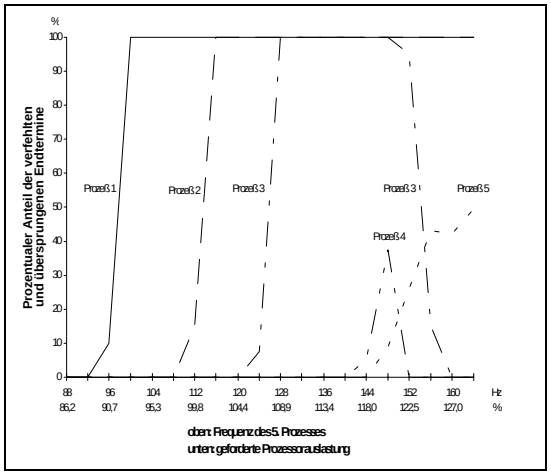

Abbildung 2: PH-1: Abhängigkeit der verfehlten und übersprungenen Endtermine von der geforderten Prozessorauslastung

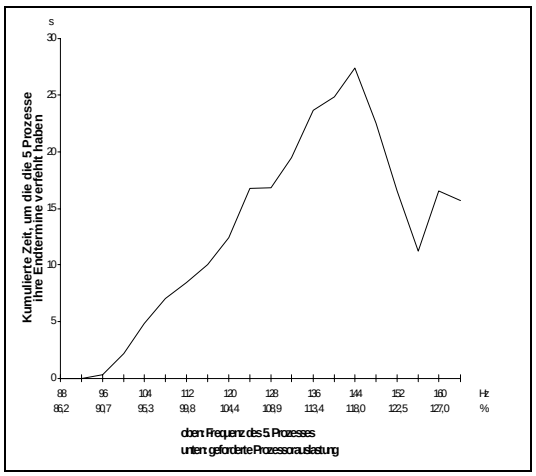

Abbildung 3: PH-1: Abhängigkeit der kumulierten Zeitüberschreitung der 5 Tasks von der geforderten Prozessorauslastung

Das erste Diagramm (Abbildung 2) zeigt die prozentualen Anteile der verfehlten und übersprungenen Endtermine an der Gesamtzahl der gesetzten Endtermine jeder beteiligten Task. Auf der Abszisse sind dabei sowohl der jeweilige Testparameter (oben) als auch die daraus resultierende Prozessorauslastung (unten) abgetragen. Task 1 hat stets die niedrigste und Task 5 die höchste Priorität.
Im zweiten Diagramm (Abbildung 3) folgt dann die verbrauchte Zeit, um die die Endtermine überschritten wurden. Dazu wurden die Zeiten aller Tasks addiert und wieder in Abhängigkeit vom Testparameter bzw. der entsprechenden Prozessorauslastung dargestellt.
In beiden Diagrammen entsprechen die ersten beiden Teilstriche der Abszisse jeweils den beiden letzten erfolgreichen Tests, bei denen noch keine Endtermine verfehlt wurden. Der zweite Teilstrich gibt demzufolge stets die maximal erreichbare Prozessorauslastung an.
Weitere Ergebnisse können [6] entnommen werden.

4 Eine Methode zur Evaluierung von Echtzeitsystemen

Softwareanalysetechniken von Echtzeitsystemen gewinnen zunehmend an Bedeutung. Es sind analytische Methoden verfügbar, mit denen die Durchführbarkeit des Schedulings von Tasksätzen berechenbar ist [7] [8] [9] [10].
Dabei gilt das Scheduling eines Tasksatzes als durchführbar, wenn gesichert ist, daß die Einhaltung von Echtzeitbedingungen und damit aller zeitlichen Zwänge garantiert werden kann. Diese zeitlichen Zwänge werden durch Zeitschranken (Deadlines) charakterisiert.
Im vorherigen Abschnitt wurden Ergebnisse präsentiert, die auf einem realen System ermittelt wurden. Als Leistungsindex wurde dabei die Auslastung des Systems und die Einhaltung der Deadlines betrachtet.
In diesem Abschnitt soll das Gesamtkonzept und die Ziele beschrieben werden, die

wir mit dem Tool EVASCAN verfolgen.
Da, wie bereits in der Einleitung erwähnt, in einem Prozeßautomatisierungssystem nicht allein die Bereitstellung eines Ergebnisses, sondern auch der Zeitpunkt an dem dieses Ergebnis vorliegt entscheidend ist, sollte die Berücksichtigung der Zeitschranken bereits in der frühen Phase des Entwurfs erfolgen.
Um diesem Anliegen gerecht zu werden, haben wir das Evaluierungs- und Schedulability Tool EVASCAN entwickelt. EVASCAN besteht aus den folgenden Modulen.

1. TESTFILE-EDITOR
2. SCHEDULABILITY-ANALYSER
3. SCHEDULABILITY-SIMULATOR
4. EXECUTION-MODUL

Beim Entwurf eines Echtzeitsystems ist es erforderlich die physikalischen Anforderungen in Systemanforderungen umzusetzen. Die Systemanforderungen werden in der derzeitigen Version in Form einer Tabelle in einem Testfile abgelegt. Die zugehörigen Informationen enthalten Aussagen über die Art der Task (periodisch, aperiodisch, sporadisch), deren Laufzeiten, Prioritäten, Verwendung von Synchronisationsmodulen).
Zum Erstellen dieser Test- und Spezifikationsfiles dient der *Testfile-Editor*.
Aus den in den Spezifikationsfiles enthaltenen Informationen läßt sich bestimmen, inwieweit ein Tasksatz durchführbar ist. Die Durchführbarkeit (Schedulability) bezieht sich dabei auf das Einhalten der Zeitschranken. Das Ergebnis dieses Durchführbarkeitstests liefert der *Schedulability-Analyser*. Der *Schedulability-Analyser* verwendet zwei unterschiedliche mathematische Modelle:
1. auf Basis der Auslastungsgrenzen und
2. auf Basis der Analyse der Reaktionszeiten

Während die erste Methode eine grobe Einschätzung der Durchführbarkeit gibt, ermöglicht die zweite eine exakte Analyse des Ablaufs.
Für eine detaillierte Beschreibung dieser Modelle sei auf [9] verwiesen.
Neben dieser reinen Ergebnispräsentation ermöglicht der *Schedulability-Simulator* die graphische Anzeige der Abläufe.
Damit der Entwickler das Verhalten der im Testfile beschriebenen Spezifikation auf einem realen System abschätzen kann, dient das *Execution-Modul*. Dieses Modul führt eine Arbeitslast unter realen Bedingungen auf einem System zur Durchführung.

5 Ausblick

Im Bereich der eingebetteten Echtzeitsysteme und großer komplexer Prozeßautomatisierungssysteme gewinnen zunehmend Werkzeuge an Bedeutung, die den Entwurfsprozeß solcher Systeme unterstützen. Sie werden eingesetzt in Anwendung der Telekommunikation, der Automobilindustrie, Luft- und Raumfahrt. Solche Werkzeuge sollten integraler Bestandteil von Echtzeitbetriebssystemen sein, da diese mittlerweile auf einem soliden theoretischen Gerüst basieren und dem Entwickler die Möglichkeit geben, im

frühen Entwurfsprozeß Hinweise über das zeitliche Verhalten der Applikation zu erhalten.
Die hier vorgestellten Ergebnisse stellen die Grundlage für weitere Arbeiten auf diesem Gebiet dar. Dabei verfolgen wir folgende Ziele:

- Erweiterung von Evascan, um den Einfluß spezieller Implementierungsformen realer Betriebssystem im Entwurf zu berücksichtigen
- Verwendung alternative Eingabespezifikationen in Form von z.B. Task- und Ressourcengraphen oder einer Simulationssprache
- Integration verschiedener Serveralgorithmen zur Bedienung unterschiedlicher Forderungenströme
- Integration von EVASCAN in laufende Arbeiten zum Hardware-Software-Codesign

6 Literatur:

[1] Kar, R.P., Porter Kent, „Rhealstone -A Real-Time Benchmarking proposal," In: Dr. Dobb´s Journal, S.14-24, Febr. 1989

[2] Kar, R.P., „Implementing the Rhealstone Real-Time Benchmark," In: Dr. Dobb´s Journal, S.46, April 1990

[3] Donohoe,P., Shapiro, R., Weidermann, N., „Hartstone Benchmark User´s Guide," Software Engineering Institute, Carnegie Mellon University, Technical Report, Mai 1990

[4] Weidermann, N.H., Kamenoff, N.I., „Hartstone Uniprocessor Benchmark: Definitions and Experiments for Real-Time Systems," In: Real-Time Systems Journal, 4 (4), S 353-383, Kluwer Academic Publishers, Dez. 1992

[5] Panzieri, F., Donatiello, L., Poretti, L., „Scheduling Real-Time Tasks: A Performance Study," Laboratory for Computer Science, University of Bologna, Technical Report UBLCS-93-10, 1993

[6] Golatowski, F., Timmermann, D., „Application Oriented Performance Evaluation of Real-Time Systems Based on Modern Microprocessor Architectures," EMSYS'96, OMI 6th Annual Conf., Berlin, Sept. 1996, erschienen in: Müller-Schloer et al. (Eds.) Embedded Microprocessor Systems, IOS Press, S.354-360, 1996

[7] Harter, P.K.: „Response Times in level-structured Systems," Techn. Report, Univ. of Colorado, Boulder, 1984

[8] Joseph, M, Pandya,P., „Finding Response Times in a Real-Time System," Computer Journal J., vol. 29, Nr. 5, Mai 1986, S. 390-395

[9] Joseph, M. (ed.), „Real-time Systems: Specification, Verification and Analysis," Prentice Hall, 1996

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