Energiebeschränkte Echtzeitsysteme und ihre Worst-Case-Analysen[1]

Peter Wägemann[2]

Abstract: Der zuverlässige Betrieb von Anwendungen, welche sowohl Zeit- als auch Energiebeschränkungen aufweisen, ist eine zentrale Herausforderung im Bereich der eingebetteten Systeme. Um die Ausführung von Aufgaben innerhalb von Ressourcenbudgets zu garantieren, benötigen energiebeschränkten Echtzeitsysteme Schranken der Worst-Case-Ausführungszeit sowie des Worst-Case-Energieverbrauchs. Neben dem Problem der Bestimmung von oberen Schranken haben Worst-Case-Analysewerkzeuge das grundlegende Problem, dass die Genauigkeit der ausgegebenen Schranken unbekannt ist, in Hinblick auf den tatsächlichen schlimmsten anzunehmenden Fall.

Die Dissertation adressiert die genannten Probleme, indem zunächst ein Analyseansatz für Schranken des Energieverbrauchs aufgezeigt wird, welcher vollständige Echtzeitsysteme mit allen Leistungsverbrauchern verarbeitet. Hinsichtlich des Analysepessimismus der Schranken stellt die Dissertation einen Ansatz zur Bestimmung der Genauigkeit von Analysen basierend auf automatisch generierten Benchmark-Programmen vor. Um die notwendigen Bestandteile für den sicheren Betrieb zu komplettieren, präsentiert die Dissertation einen Betriebssystemkern, welcher auf Szenarien dynamisch reagiert, bei denen eine Ressource stärkere Beschränkungen aufweist als die andere.

1 Einleitung

Eine zentrale Herausforderung dieses Jahrzehnts im Bereich der Computersysteme ist der zuverlässige Betrieb von eingebetteten Systemen mit starken Energiebeschränkungen, zum Beispiel durch die unkontinuierliche Energiezufuhr durch Energie-Harvesting-Mechanismen [Co18]. Diese Klasse von Rechensystemen ist von praktischer Relevanz in Form von implantierbaren medizinischen Geräten, wie beispielsweise Herzschrittmacher oder Defibrillatoren [Ou19]. Diese Anwendungen haben einerseits harte Echtzeitanforderungen, da sowohl die Sensordatenerfassung der Herzaktivität als auch die Aktorik von möglichen Schockimpulsen innerhalb von bestimmten Zeitschranken ausgeführt werden müssen. Zusätzlich existieren – aufgrund des Batteriebetriebs – Anforderungen an den Energiebedarf aller Aufgaben des Systems. Um eine verlässliche Planung unter Berücksichtigung der verfügbaren Zeit-/Energieressourcen zu garantieren, sind die Werte der *schlimmsten anzunehmenden Ausführungszeit (engl. worst-case execution time, WCET)* sowie des *schlimmsten anzunehmenden Energieverbrauchs (engl. worst-case energy consumption, WCEC)* für jede Aufgabe im System notwendig.

Statische Programmcodeanalysen sind ein grundlegendes Mittel im Bereich der Echtzeitsysteme für die Bestimmung von verlässlichen Schranken des Ressourcenbedarfs. Zwar

[1] Englischer Titel der Dissertation: „Energy-Constrained Real-Time Systems and Their Worst-Case Analyses"

[2] Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), waegemann@cs.fau.de

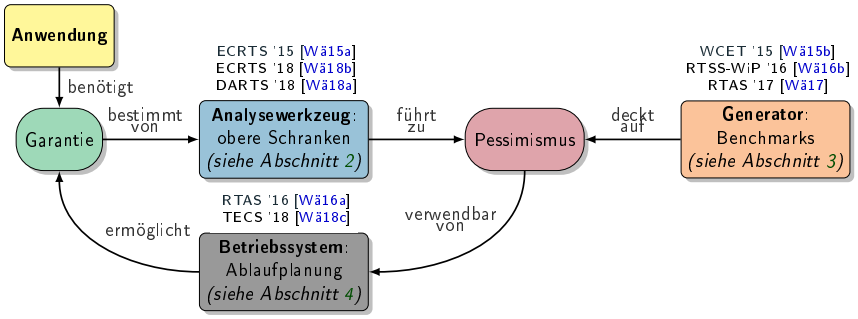

Abb. 1: Konzeptionelle Struktur der Dissertation

existieren praxistaugliche Analysen für die WCET [Wi08], jedoch sind diese nicht direkt verwendbar für das fundamental andere Verhalten des Energieverbrauchs und somit für die Bestimmung von WCEC-Werten. Im Gegensatz zur Laufzeitanalyse ist beispielsweise eine Betrachtung von Echtzeitprioritäten für den Energieverbrauch unzureichend: So können auch niederpriore Aufgaben verschiedene Geräte (z. B. Transceiver) temporär aktivieren, welche dadurch den Leistungsbedarf des Gesamtsystems ändern. Dieser erhöhte Leistungsbedarf beeinflusst wiederum den Energieverbrauch aller Aufgaben, unabhängig von deren Echtzeitpriorität. Diese *wechselseitige Beeinflussung* des Ressourcenbedarfs zwischen allen Aufgaben im System muss bei der WCEC-Analyse berücksichtigt werden.

Ein fundamentales Problem bei der Verwendung von statischen Worst-Case-Analysen ist die Evaluation und die Validierung der oberen Schranke des Ressourcenbedarfs, die das Analysewerkzeug ermittelt. Durch sichere Abstraktionen bestimmen Analysewerkzeuge wiederum sichere Schranken. Allerdings ist sowohl der tatsächliche WCET- als auch der tatsächliche WCEC-Wert von beliebigen Programmen unbekannt, da allgemein solchen nicht-trivialen Eigenschaften aus Programmen nicht automatisiert abgeleitet werden können [KKZ13]. Dadurch ist die Genauigkeit der Abstraktionen der Analysewerkzeuge nicht bewertbar, wodurch der Analysepessimismus unbekannt bleibt. Das fehlende Wissen über diese Vergleichslinien (d. h. tatsächliche WCET-/WCEC-Werte) verhindert auch die Beantwortung der Frage, ob die Implementierung der Analyse Fehler aufweist und möglicherweise Unterschätzungen des Ressourcenbedarfs ausgibt.

Verlässliche Schranken des Ressourcenbedarfs sind notwendige Voraussetzungen für den Entwurf von energiebeschränkten Echtzeitsystemen. Zur Laufzeit benötigen diese Systeme einen Betriebssystemkern zusammen mit einer ressourcengewahren Ablaufplanung, um einen *sicheren Betrieb* zu gewährleisten. Für einen *effizienten Betrieb* ist Wissen über den erwarteten Analysepessimismus notwendig, da die Ablaufplanung diesen nutzen kann.

Die Dissertation [Wä20] adressiert die genannten Probleme der Ermittlung von WCEC-Schranken, der Bestimmung von Analysepessimismus und des Betriebs der energiebeschränkten Echtzeitsysteme. Die konzeptionelle Struktur ist in Abbildung 1 illustriert. Folgende drei Lösungsansätze stellen die Hauptbeiträge der Dissertation dar: Der WCEC-Analysator *SysWCEC* bestimmt sichere Schranke des Energiebedarfs für eingebettete Echtzeitsysteme mit festen Prioritäten (siehe Abschnitt 2). Der Benchmark-Generator *GenE*

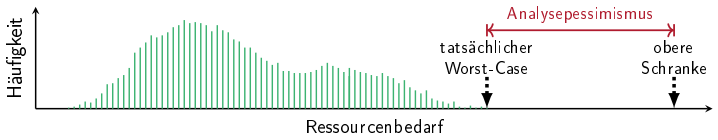

Abb. 2: Exemplarisches Histogramm des Ressourcenbedarfs einer Aufgabe

ermöglicht erstmals eine Evaluation und Validierung von statischen Worst-Case-Analysatoren, basierend auf dem Wissen des tatsächlichen Worst-Case-Ressourcenbedarfs (siehe Abschnitt 3). Der Betriebssystemkern *EnOS* unterstützt den sicheren Betrieb von energiebeschränkten Echtzeitsystemen durch a priori Wissen über WCET- und WCEC-Werte (siehe Abschnitt 4). Die Struktur dieses Artikels ergibt sich aus diesen drei Hauptbeiträgen, wie in Abbildung 1 dargestellt. Abschnitt 5 fasst die Ergebnisse zusammen.

2 Worst-Case-Analysen

Die Probleme bei der Bestimmung von WCEC-Werten ist Inhalt von Abschnitt 2.1. Abschnitt 2.2 beschreibt die Lösungen durch *SysWCEC*. Zunächst werden jedoch Hintergrundinformationen zu Worst-Case-Analysen und das Systemmodell präsentiert.

Analysepessimismus Abbildung 2 zeigt ein exemplarisches Histogramm von Ressourcenverbräuchen (entweder Zeit oder Energie) einer Aufgabe. Dabei variiert der Ressourcenbedarf zum Beispiel durch verschiedene Eingabedaten oder durch unterschiedliche initiale Hardwarezustände zu Beginn der Ausführung der Aufgabe. Der tatsächlich Worst-Case-Wert ist im Allgemeinen nicht genau bestimmbar, weshalb Analysatoren pessimistische Annahmen treffen, die zu Überschätzungen dieses Wertes führen. Der Abstand zwischen Worst-Case-Wert und oberer Schranke kennzeichnet den Pessimismus des Analysators für die sichere Ausführung der Aufgabe. Die Bestimmung vom erwarteten Grad des Pessimismus (siehe Abschnitt 3) ist eine notwendige Voraussetzung für den effizienten Betrieb von energiebeschränkten Echtzeitsystemen (siehe Abschnitt 4).

Systemmodell Die adressierte Klasse von energiebeschränkten Echtzeitsystemen hat einen Rechenkern. Die Aufgaben der Anwendung werden mit festen Prioritäten abgearbeitet. Weiterhin können die Aufgaben Betriebsmittel zu Synchronisationszwecken anfordern. Das Auftreten von asynchronen Interrupts ist in diesen Systemen oftmals gegeben, wenn beispielsweise Zeitgeber-Interrupts für die Laufzeitüberwachung eingesetzt werden. Für die statische Worst-Case-Analyse stellen diese Interrupts eine große Herausforderung dar, da das mögliche dynamische Verhalten statisch abgebildet werden muss. Das Auftreten dieser Unterbrechungen ist über ihre minimale Zwischenankunftszeit begrenzt. Für den Leistungsbedarf des Systems (und somit auch den Energiebedarf über die Zeit gesehen) spielen die softwaregesteuerten Geräte eine entscheidende Rolle. Dabei sind Geräte nicht notwendigerweise als extern anzusehen, wie beispielsweise Transceiver zur Kommunikation. Interne Geräte (wie Zeitgeber-Subsysteme oder Analog-Digital-Konverter) haben das gleiche Verhalten aus Sicht der statischen Analyse: Das Aktivieren des Geräts führt zu einer erhöhten Leistung des Gesamtsystems, und das Deaktivieren reduziert wiederum den Verbrauch, wie im Folgenden näher erläutert.

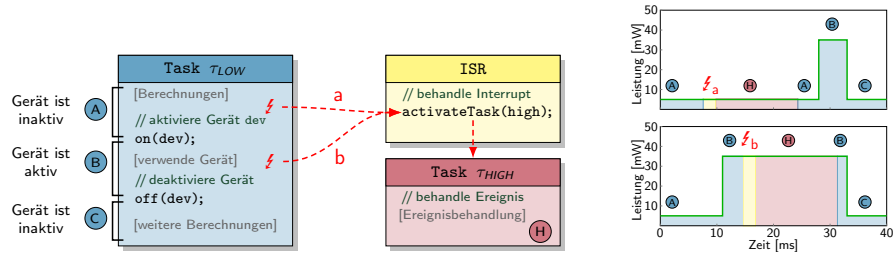

Abb. 3: Der Leistungsbedarf des Gesamtsystems beeinflusst die Energie jeder Aufgabe.

2.1 Problemstellungen der WCEC-Analyse

Zur Energieeinsparung werden Geräte nur dann aktiviert, wenn die Anwendung den Dienst benötigt. Abbildung 3 zeigt eine niederpriore Aufgabe (engl. task) τ_{LOW}, eine Interrupt-Service-Routine (ISR) und eine höherpriore Aufgabe τ_{HIGH}, die von der ISR aktiviert wird. Die niederpriore Aufgabe führt eine temporäre Aktivierung eines Geräts aus, welche den Leistungsbedarf des Gesamtsystems von 5 mW auf 35 mW anhebt. Wie auf der rechten Seite in Abbildung 3 illustriert ist, ergeben sich zwei Szenarien beim Energiebedarf zur Fertigstellung der niederprioren Aufgabe τ_{LOW}: In Szenario a (siehe $\lightning_a$ in Abbildung 3) tritt der Interrupt im Zustand der geringen Leistungsaufnahme auf. Im Gegensatz dazu wird der Interrupt in Szenario b ($\lightning_b$) im hohen Leistungszustand abgearbeitet, wodurch sich hier der Worst-Case hinsichtlich des Energiebedarfs ergibt (schlimmste anzunehmende Fläche unter der Kurve). Dieser Worst-Case-Energiebedarf gilt sowohl für τ_{LOW} (gesamte Fläche) als auch für τ_{HIGH} (rote Fläche unter (H)), weil τ_{HIGH} durch die Aktivierung von τ_{LOW} mit der hohen Leistung von 35 mW startet. Für die Analyse der Laufzeit sind die kontextsensitiven Leistungszustände irrelevant, da lediglich die Beeinflussung der niederprioren Aufgaben durch höherpriore Aufgaben betrachtet werden muss. Jedoch muss die WCEC-Analyse die systemweite Aktivität der Geräte sowie deren Leistungszustände berücksichtigen. Bei der Modellierung des Energieverhaltens ist eine Betrachtung der *wechselseitigen Beeinflussung* von Aufgaben erforderlich.

2.2 *SysWCEC* – systemweite WCEC-Analyse

Zur Lösung des Problems der WCEC-Analyse stellt die Dissertation den *SysWCEC*-Ansatz vor [Wä18b]. Die Grundidee von *SysWCEC* besteht aus vier Teilen: *(1.) Dekomposition*: Der Code des Zielsystems wird untergliedert in Abschnitte, welche eine gemeinsame Menge an aktiven Geräten besitzen. *(2.) Pfadanalyse*: Basierend auf dem Ergebnis der Dekomposition des Systems wird eine explizite Aufzählung aller möglichen Programmpfade ausgeführt. *(3.) Problemformulierung*: *SysWCEC* verwendet das gewonnene Wissen aus der Pfadanalyse für die Formulierung eines ganzzahlig linearen Programms (engl. integer linear program, ILP). *(4.) WCEC-Bestimmung*: Das Lösen des ILPs ergibt schlussendlich eine WCEC-Schranke der zu analysierenden Aufgabe.

Der folgende Abschnitt erläutert das Vorgehen von *SysWCEC* näher, ausgehend vom Beispiel in Abbildung 3. Die linke Seite von Abbildung 4 zeigt nun das untergliederte Sys-

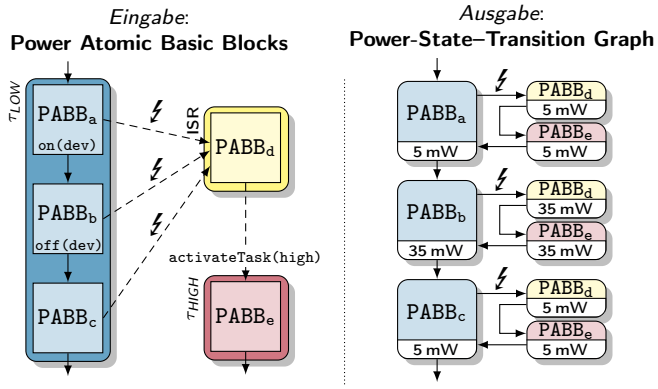

Abb. 4: Der *SysWCEC*-Ansatz verwendet die Untergliederung des Systems in Blöcke mit gleicher Leistungsaufnahme für eine Aufzählung aller möglichen Systemzustände.

tem: Die ursprüngliche niederpriore Aufgabe τ_{LOW} besteht jetzt aus einzelnen *Power Atomic Basic Blocks (PABBs)*, welche sich unter anderem durch eine gemeinsame Menge an aktiven Geräten kennzeichnen und eine Erweiterung der Atomic Basic Blocks darstellen [SSP11]. Die Aufgabe τ_{LOW} besteht somit aus den drei Teilen $\texttt{PABB}_\texttt{a}$ (Code vor Aktivierung), $\texttt{PABB}_\texttt{b}$ (Code während Aktivierung) und $\texttt{PABB}_\texttt{c}$ (Code nach Aktivierung).

Die rechte Seite von Abbildung 4 zeigt das Ergebnis der Pfadanalyse, den *Leistungszustandsgraphen (engl. Power-State–Transition Graph, PSTG)*. Der Algorithmus der Pfadanalyse beginnt mit dem initialen Zustand der geringen Leistungsaufnahme (5 mW). Unter Beachtung der Betriebssystemsemantik und der potenziell auftretenden Interrupts werden Transitionen im PSTG eingefügt. Dabei wird bei einem gerätebezogenen Aufruf (z. B. Aktivierung des Geräts) der neue Leistungszustand propagiert. Ansonsten gibt die Analyse den vorherrschenden Kontext der Geräte und deren Leistung über die möglichen Systempfade weiter, solange bis alle Pfade exploriert wurden. Das Wissen über alle Systempfade löst das Analyseproblem der wechselseitigen Beeinflussung der Aufgaben. *SysWCEC* verwendet das Wissen über Pfade des PSTG für die Nebenbedingungen des ILPs. Die Zielfunktion des ILPs ist die Maximierung des Energiebedarfs über alle möglichen Systemzustände, in denen die zu analysierende Aufgabe ausgeführt werden kann. Das Lösen des ILPs mittels eines mathematischen Optimierers liefert die obere WCEC-Schranke.

Die Formulierung der Kosten für die Zielfunktion des ILPs verwendet den maximalen (kontextsensitiven) Leistungszustand P_{max} der Knoten im PSTG. Um nun wiederum eine obere Schranke für einzelne Knoten des PSTG zu erhalten, bestimmt *SysWCEC* die WCET jedes Knotens. Der Energiebedarf einzelner Knoten berechnet sich folglich durch $WCEC = P_{max} \cdot WCET$. Durch diese Art der WCEC-Analyse ist es möglich, *sichere* Schranken von Knoten zu bestimmen. Jedoch ist die *Genauigkeit* der WCET-Schranke unbekannt. Für diese Frage nach dem Analysepessimismus von *SysWCEC* muss der Pessimismus von WCET-Werten abgeschätzt werden. Das Vorgehen bei der Abschätzung von Pessimismus ist die zentrale Fragestellung des folgenden Abschnitts.

3 Validierung von Worst-Case-Analysen

Der folgende Abschnitt 3.1 erläutert das grundsätzliche Problem bei der Bestimmung der Genauigkeit von Worst-Case-Analysen. Anschließend beschreibt Abschnitt 3.2 die Lösung dieses Problems durch den Benchmark-Generator *GenE*.

3.1 Problem der Validierung von Worst-Case-Analysen

Durch die verwendeten Abstraktionen in den Analysealgorithmen liegt die ausgegebene obere Schranke – bei korrekter Implementierung – über dem tatsächlichen Worst-Case (wie in Abbildung 2). Existierende Verfahren zur Abschätzung des Analysepessimismus verwenden Benchmark-Suites auf der Ebene von Quellcode (z. B. C). Das grundlegende Problem ist hierbei, dass bei diesen Evaluationsverfahren das absolute Vergleichsmaß, also der tatsächliche Worst-Case, fehlt. Dieser tatsächliche Worst-Case sowie alle relevanten Programmfakten (wie Schleifenobergrenzen) sind jedoch nicht automatisiert bestimmbar [KKZ13]. Auch eine manuelle Extraktion dieser Fakten ist aufwendig und fehleranfällig. Durch dieses fehlende Wissen haben existierende Evaluationsansätze nur eine geringe Aussagekraft, weil sie den absoluten Grad der Über- oder (im Fall eines Softwarefehlers) Unterschätzung nicht bestimmen können.

3.2 *GenE* – Benchmark-Generator für Worst-Case-Analysen

Der Generator *GenE* ist eine Lösung für das Problem der Evaluation und Validierung von Worst-Case-Analysen [Wä15b, Wä17]. *GenE* generiert Benchmark-Programme auf eine Art, sodass alle Programmfakten bekannt sind. Mit diesem Wissen ist der tatsächliche Worst-Case bestimmbar, der wiederum als Vergleichsmaß für Analysewerkzeuge dient. Die Funktionsweise von *GenE* lässt sich anhand einer Metapher veranschaulichen: Benchmarks sind wie ein Irrgarten für Analysatoren, welche einen Weg durch den Irrgarten finden müssen. Selbst wenn ein Analysator möglicherweise einen Weg (Lösung) finden würde, so ist unbekannt, ob dieser der optimale Weg ist. *GenE* verfolgt den genau umgekehrten Ansatz: Zunächst wird ein Pfad bestimmt und darauf folgend werden Verzweigungen entlang dieses Pfades eingefügt. Dadurch wird der Irrgarten um den vorab bestimmten Pfad gebaut. Durch den generativen Ansatz von *GenE* ist die optimale Lösung (tatsächlicher Worst-Case) des Irrgartens konstruktionsbedingt bekannt.

Programm-Patterns Für den Generierungsprozess verwendet *GenE* eine Vielzahl von verschiedenen *Programm-Patterns*. Diese haben die Eigenschaft, dass sie Wissen über die Programmfakten (wie ihren Worst-Case-Pfad) besitzen. Weiterhin sind Programm-Patterns miteinander kombinierbar für die Erzeugung von neuen Benchmarks. Dadurch generiert *GenE* beispielsweise Schleifen innerhalb von eingabedatenabhängigen Verzweigungen. *GenE* implementiert eine Vielzahl von unterschiedlichen Pattern für Pfade, Schleifen, arithmetische Operationen oder Variablen. Um die Generierung von unrealistischem Code für Worst-Case-Analysen zu vermeiden, verwendet *GenE* Pattern aus realen Industrieanwendungen sowie aus existierenden Benchmark-Suites.

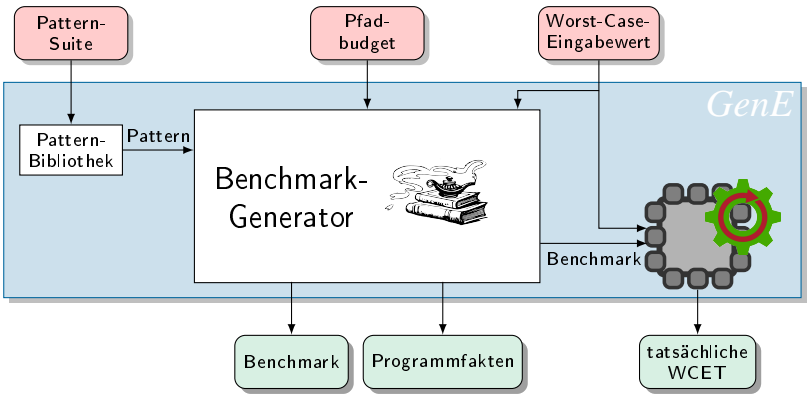

Abb. 5: Der *GenE*-Generator erzeugt Benchmarks so, dass der schlimmste Fall des Ressourcenbedarfs bekannt ist. Dieser Fall dient als Vergleichsmaß für statische Worst-Case-Analysewerkzeuge.

Pattern-Suites Ein weiteres Problem bei der bisherigen Verwendung von Benchmarks ist die Tatsache, dass existierende Benchmarks im Bereich der Worst-Case-Analyse üblicherweise mehrere verschiedene Pattern enthalten. Daraus resultiert das Problem, dass verschiedene Faktoren zum Pessimismus in der ausgegebenen Worst-Case-Schranke beitragen. *GenE* löst dieses Problem, indem die Auswahl der verwendeten Programm-Pattern konfigurierbar ist. Durch diese spezifischen *Pattern-Suites* kann *GenE* die individuellen Stärken und Schwächen von Analysatoren evaluieren.

Ein- & Ausgaben Diese Konfiguration der Pattern-Suite ist eine der Eingaben für *GenE*, wie in Abbildung 5 dargestellt. Das Pfadbudget konfiguriert die Komplexität des generierten Benchmarks (durch die Angabe der Anzahl von Instruktionen entlang des Worst-Case-Pfades). Der Worst-Case-Eingabewert ist ein entscheidender Wert für den generierten Benchmark: Wird das Programm mit diesem Wert ausgeführt, so wird der Worst-Case-Pfad abgelaufen. Die Vermessung dieses Pfades ergibt letztendlich den tatsächlichen Worst-Case-Ressourcenbedarf. *GenE* zielt auf die Evaluation von WCET-Analysatoren ab. Basierend auf *GenE* entwickelten Eichler et al. einen Generator für vollständige Echtzeitsysteme [Ei18] sowie einen speziellen Generator für WCEC-Analysatoren [EWSP19].

Evaluation & Validierung von aiT WCET-Werkzeug Evaluationen mit dem WCET-Analysator aiT der Firma AbsInt zeigten einige Benchmarks, bei denen der ausgegebene Wert geringer als die tatsächliche WCET war. AbsInt konnte mithilfe der Benchmarks von *GenE* dieses Verhalten bestätigten, welches durch ein fehlerhaftes Hardwaremodell verursacht wurde. Nachdem die Unterschätzungen bekannt wurden, korrigierte AbsInt diese Fehler in einer überarbeiteten Version von aiT, in der keine Fehler mehr gefunden werden konnten. In Anbetracht der Tatsache, dass statischen WCET-Werkzeuge für äußerst sicherheitskritische Echtzeitsysteme (wie den Airbus A380) verwendet werden, ist die strukturierte Evaluation und Validierung durch einen Testfall-Generator wie *GenE* von hoher Relevanz, um die Qualität der Analysatoren zu verbessern.

GenE ermöglicht durch die spezifischen Pattern-Suites das Aufspüren von individuellen Schwächen in der WCET-Analyse. Jedoch ist bei der Verwendung von sicheren Schranken

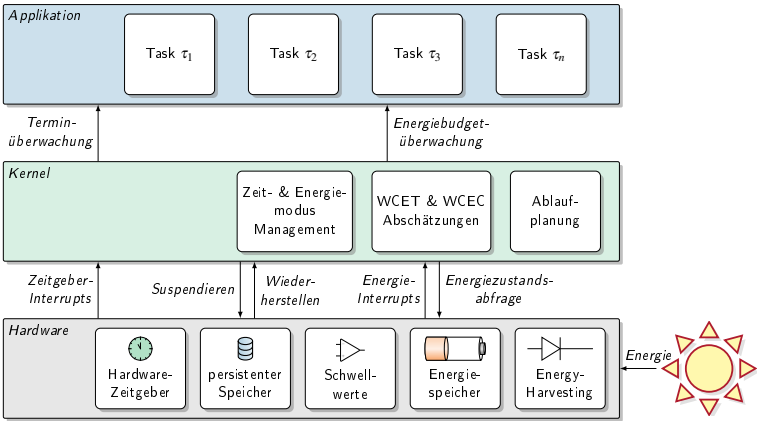

Abb. 6: *EnOS* ermöglicht Echtzeitgarantien während dedizierter Phasen und sorgt durch die vorausschauende Ablaufplanung für sichere Zustände während Stromausfällen.

Pessimismus zur Laufzeit unvermeidbar. Wenn allerdings der Betriebssystemkern Wissen über den erwarteten Grad des Pessimismus besitzt, so kann der Kern diesen effizient ausnutzen. Der nächste Abschnitt über den *EnOS*-Kern erläutert diesen Zusammenhang näher.

4 Betrieb energiebeschränkter Echtzeitsysteme

Beim Betrieb von energiebeschränkten Echtzeitsystemen ergeben sich mehrere Herausforderungen (siehe Abschnitt 4.1). Basierend darauf stellt Abschnitt 4.2 *EnOS* vor.

4.1 Herausforderungen beim Betrieb energiebeschränkter Echtzeitsysteme

Analysepessimismus zeigt sich zur Laufzeit durch ungenutzte Ressourcen. Besonders in eingebetteten Systemen möchte man dies verhindern. Weiterhin muss die Ablaufplanung des energiebeschränkten Echtzeitsystems die beiden Ressourcen Zeit und Energie berücksichtigen, um Ausführungs- und Laufzeitgarantien für kritische Aufgaben zu gewährleisten. Der in Abschnitt 4.2 vorgestellte Betriebssystemkern *EnOS* adressiert die spezielle Klasse der *energieneutralen* Systeme, welche sich dadurch auszeichnet, dass die verwendete Energie vollständig aus der Umgebung entnommen wird und in aufladbaren Batterien gespeichert wird. Mechanismen für diese Art der Energieernte (engl. energy harvesting) umfassen beispielsweise Solarzellen. Durch die unregelmäßige Energiezufuhr muss das System mögliche Stromausfälle tolerieren. Wenn sich der Batteriezustand dem Ende entgegen neigt, muss das System rechtzeitig in einen sicheren Zustand übergehen, da dieser Übergang selbst Energie erfordert. Weiterhin benötigt das System Wissen darüber, wann ausreichend Energie geerntet wurde, um nach einem Stromausfall wieder sinnvolle Arbeit unterbrechungsfrei auszuführen und Phasen mit Echtzeitanforderungen zu ermöglichen.

4.2 *EnOS* – Betriebssystemkern für energieneutrale Echtzeitsysteme

Abbildung 6 zeigt eine Übersicht des Designs. Hardwareseitig verwendet *EnOS* eine Überwachung des Ressourcenbedarfs, welche beispielsweise beim Über-/Unterschreiten eines Schwellwerts der verfügbaren Energie dem Kern einen Interrupt zustellt. *EnOS* benutzt nicht-flüchtigen Speicher für das Sichern von Daten während eines Stromausfalls.

Auf der Softwareseite ist die offline berechnete Budgetierung von verschiedenen Betriebsmodi basierend auf WCEC- und WCET-Abschätzungen von zentraler Bedeutung. Diese Modi werden zur Laufzeit ausgeführt, entsprechend der verfügbaren Energie. Durch die vorausschauende Planung erlaubt *EnOS* Phasen mit Echtzeitgarantien. Nahe der leeren Batterie garantiert *EnOS* den Übergang in einen sicheren Zustand durch pessimistische (aber verlässliche) WCEC-Schranken. Dagegen verwendet *EnOS* in Phasen von hoher Energieverfügbarkeit optimistische Abschätzungen, welche den Einfluss des Analysepessimismus mildern und Ressourcen effizienter ausnutzen. Diese sind möglicherweise Unterschätzungen des tatsächlichen WCEC-Werts, jedoch garantiert die Ressourcenüberwachung unter allen Umständen die Ausführung von kritischen Aufgaben. Bei der Bestimmung von optimistischen Budgets und des erwarteten Grades an Analysepessimismus ist wiederum *GenE* hilfreich (siehe Abschnitt 3). Das Wissen über WCEC-Werte für Aufgaben ermöglicht *EnOS* die Beantwortung der Frage, wann ausreichend Energie geerntet wurde, um den Betrieb nach einem Stromausfall vorausschauend wieder aufzunehmen.

5 Zusammenfassung

Die Dissertation [Wä20] zielt auf den verlässlichen Betrieb von Systemen ab, die sich durch Energie- und Zeitbeschränkungen kennzeichnen. Der erste der drei Hauptbeiträge ist der Analysator *SysWCEC*, welcher Schranken des Energiebedarfs von Aufgaben bestimmt. *SysWCEC* beinhaltet eine Methodik zur Modellierung von temporär aktiven Leistungsverbrauchern. Weiterhin betrachtet diese Analyse die Ablaufplanung mit festen Prioritäten, die Verwendung von Betriebsmitteln, synchrone Aufgaben und asynchrone Interrupts. Die Bestimmung des Analysepessimismus ist ein fundamentales Problem bei Worst-Case-Analysen, welches die Dissertation mit dem Benchmark-Generator *GenE* löst. Das Aufdecken von Softwarefehlern in einem verbreiteten WCET-Werkzeug unterstreicht die Signifikanz eines solchen Generators zur Validierung von Analysatoren. Um zur Laufzeit einen Betrieb unter Zeit- und Energieschranken zu ermöglichen, stellt die Dissertation den Kern *EnOS* vor. Durch das Wissen über den erwarteten Analysepessimismus lässt sich dessen Einfluss mildern. *EnOS* verwendet zur Laufzeit ungenutzte Ressourcen für unkritische Aufgaben und garantiert schlussendlich die Ausführung kritischer Aufgaben unter Verwendung von verlässlichen Schranken des Ressourcenbedarfs.

> Der Quellcode von *SysWCEC*, *GenE* und *EnOS* ist online verfügbar:
> `https://gitlab.cs.fau.de/{syswcec, gene, enos}`

Literaturverzeichnis

[Co18] Cohen, A. et al.: Inter-Disciplinary Research Challenges in Computer Systems for the 2020s. Bericht, USA, 2018.

[Ei18] Eichler, C.; Distler, T.; Ulbrich, P.; Wägemann, P.; Schröder-Preikschat, W.: TASKers: A Whole-System Generator for Benchmarking Real-Time-System Analyses. In: Proc. of WCET '18. S. 6:1–6:12, 2018.

[EWSP19] Eichler, C.; Wägemann, P.; Schröder-Preikschat, W.: GenEE: A Benchmark Generator for Static Analysis Tools of Energy-Constrained Cyber-Physical Systems. In: Proc. of CPS-IoTBench '19. 2019.

[KKZ13] Knoop, J.; Kovács, L.; Zwirchmayr, J.: WCET Squeezing: On-demand Feasibility Refinement for Proven Precise WCET-bounds. In: Proc. of RTNS '13. S. 161–170, 2013.

[Ou19] Ouyang, H. et al.: Symbiotic cardiac pacemaker. Nature Communications, 10, 2019.

[SSP11] Scheler, F.; Schröder-Preikschat, W.: The Real-Time Systems Compiler: Migrating event-triggered systems to time-triggered systems. Software: Practice and Experience, 41(12):1491–1515, 2011.

[Wä15a] Wägemann, P.; Distler, T.; Hönig, T.; Janker, H.; Kapitza, R.; Schröder-Preikschat, W.: Worst-Case Energy Consumption Analysis for Energy-Constrained Embedded Systems. In: Proc. of ECRTS '15. S. 105–114, 2015.

[Wä15b] Wägemann, P.; Distler, T.; Hönig, T.; Sieh, V.; Schröder-Preikschat, W.: GenE: A Benchmark Generator for WCET Analysis. In: Proc. of WCET '15. S. 33–43, 2015.

[Wä16a] Wägemann, P.; Distler, T.; Janker, H.; Raffeck, P.; Sieh, V.: A Kernel for Energy-Neutral Real-Time Systems with Mixed Criticalities. In: Proc. of RTAS '16. S. 25–36, 2016.

[Wä16b] Wägemann, P.; Distler, T.; Raffeck, P.; Schröder-Preikschat, W.: Towards Code Metrics for Benchmarking Timing Analysis. In: Proc. of RTSS WiP '16. 2016.

[Wä17] Wägemann, P.; Distler, T.; Eichler, C.; Schröder-Preikschat, W.: Benchmark Generation for Timing Analysis. In: Proc. of RTAS '17. S. 319–330, 2017.

[Wä18a] Wägemann, P.; Dietrich, C.; Distler, T.; Ulbrich, P.; Schröder-Preikschat, W.: Whole-System WCEC Analysis for Energy-Constrained Real-Time Systems (Artifact). Dagstuhl Artifacts Series (DARTS '18), 4(2):7:1–7:4, 2018.

[Wä18b] Wägemann, P.; Dietrich, C.; Distler, T.; Ulbrich, P.; Schröder-Preikschat, W.: Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems. In: Proc. of ECRTS '18. S. 24:1–24:25, 2018.

[Wä18c] Wägemann, P.; Distler, T.; Janker, H.; Raffeck, P.; Sieh, V.; Schröder-Preikschat, W.: Operating Energy-Neutral Real-Time Systems. ACM TECS, 17(1):11:1–11:25, 2018.

[Wä20] Wägemann, P.: Energy-Constrained Real-Time Systems and Their Worst-Case Analyses. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020.

[Wi08] Wilhelm, R. et al.: The Worst-case Execution-time Problem – Overview of Methods and Survey of Tools. ACM TECS, 7(3):1–53, 2008.

Peter Wägemann ist Post-Doktorand an der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). Im September 2020 promovierte Peter Wägemann mit Auszeichnung an der Technischen Fakultät der FAU im Bereich der Informatik. Neben einer Lehrtätigkeit im Bereich von Echtzeitsystemen sowie dem individuellen Prototyping, umfasst seine Arbeit die Forschung auf den Gebieten der eingebetteten Systeme und deren statischer Programmanalyse.