(Bild: Kritsana Noisakul | Shutterstock)

Timing-Analyse in Multicore-Systemen:

Auf den Spuren des Worst Case

Sonderdruck

Der längst möglichen Ausführungszeit eines Programms – Worst-Case Execution Time (WCET) – kann man mittels Messungen oder statischer Analyse auf die Spur kommen. Beide Methoden haben aber Nachteile. Eine hybrider Ansatz verspricht bessere Ergebnisse.

Von Simon Wegener

Die Korrektheit des Gesamtsystems in eingebetteten Echtzeitsystemen hängt nicht nur davon ab, ob eine Berechnung das richtige Ergebnis liefert, sondern auch davon, ob das Ergebnis rechtzeitig vorliegt. Daher schreiben alle Normen zur funktionalen Sicherheit (z. B. ISO 26262 im Automobilbereich, DO-178C im Luftfahrtbereich) einen Nachweis der sogenannten WCET (worst-case execution time) vor – der maximalen Ausführungszeit einer Berechnung im schlimmsten Fall.

Messung der WCET

Die für diesen Nachweis notwendigen Zeitinformationen können entweder durch Messung auf realer Hardware ermittelt werden oder mittels statischer Programmanalyse aus einem abstrakten Prozessormodell. Neben dem Nachweis wichtiger Sicherheitseigenschaften (in diesem Fall: der Nachweis, dass eine Berechnung rechtzeitig terminiert) sind Zeitanalyse-Werkzeuge auch zur allgemeinen Einschätzung der Performance eines Systems nützlich.

Im einfachsten Fall wird die Laufzeit eines Programms ermittelt, indem vor Beginn und nach Ende der Berechnung jeweils ein Zeitstempel genommen wird. Die Differenz dieser Zeitstempel ergibt dann eine Ende-zu-Ende-Laufzeit des Programms. Diese Art der Messung ist zwar relativ einfach umzusetzen, birgt aber gravierende Nachteile:

➜ Im Allgemeinen ist es nicht möglich, mittels Ende-zu-Ende-Messung eine sichere obere Schranke der Laufzeit zu ermitteln, da üblicherweise nur eine begrenzte Menge an Testfällen vermessen werden kann. Selbst wenn der kritische Pfad im Programm durch einen Testfall stimuliert wurde, kann nicht garantiert werden, dass sich die Hardware in einem Zustand befand, der die längstmögliche Ausführungszeit hervorruft. Dies ist aber insbesondere bei modernen Multicore-Prozessoren wichtig, da dort tiefe Pipelines, Caches, Sprungvorhersage und andere Techniken, die die Performance im Durchschnitt verbessern, einen großen Einfluss auf das Zeitverhalten des Prozessors haben. Eine Stimulation des schlimmstmöglichen Anfangszustands ist nicht ohne weiteres möglich.

➜ Ende-zu-Ende-Messungen ergeben nur einen Datenpunkt pro Ausführung des Programms, aus dem nicht gefolgert werden kann, in welchem Teil des Programms die meiste Ausführungszeit verstreicht. Insbesondere ist nicht zu erkennen, welchen Anteil an der Laufzeit die eigentliche Berechnung hat, und welcher Anteil durch mögliche Unterbrechungen (z. B. Interrupts) oder Interferenzen (Warten auf belegte Ressourcen, z. B. gemeinsamer Speicher im Multicore-System) verursacht wird. In der industriellen Praxis werden daher oft zuerst Ende-zu-Ende-Messungen des gesamten Programms und später Messungen für die Teilbereiche des Programms durchgeführt, in denen man Hotspots vermutet. Dieser Prozess ist sehr aufwendig.

➜ Besondere Vorsicht ist auch geboten, falls die Software zur Vermessung instrumentiert wird. Die zusätzlich eingefügten Instruktionen verändern das Zeitverhalten des zu vermessenden Programms derart, dass aus den Messungen kein Rückschluss auf das Zeitverhalten der ursprünglichen Software getroffen werden kann. Dieses Verhalten ist als „probe effect“ bekannt. Si-

cherheitsnormen wie die ISO 26262 im Automobilbereich verbieten die Instrumentierung, falls der Instrumentierungscode die zu beobachtenden Eigenschaften verändern kann. Dieses Problem kann umgangen werden, wenn die Hardware die nicht-intrusive Erstellung von Programm-Traces unterstützt.

Statische WCET-Analyse

Im Gegensatz zur Messung wird bei der statischen WCET-Analyse das Zeitverhalten eines Programms mit Hilfe eines mathematischen Modells der Ziel-Hardware ermittelt. Dabei wird nicht ein konkreter Lauf des Programms analysiert, sondern die Gesamtheit aller möglichen Programmzustände und aller möglichen Hardware-Zustände untersucht. Damit ist es möglich, eine sichere obere Schranke der Ausführungszeit zu bestimmen.

Statische WCET-Analyse ist das Mittel der Wahl für Singlecore-Prozessoren und wird zum Beispiel von Airbus eingesetzt, um das Zeitverhalten der Flight-Control-Software des A380 zu verifizieren. Selbst komplexe Prozessoren können präzise modelliert werden – inklusive Cache, Pipeline etc. Diese Modellierungen beruhen einerseits auf verfügbaren Prozessordokumentationen, andererseits werden die einzelnen Instruktionen des Prozessors extensiv vermessen, um daraus ein Modell des Zeitverhaltens abzuleiten. Dieses Verfahren findet seine Grenzen, wenn Teile des Prozessors, welche einen Einfluss auf das Zeitverhalten haben, nicht oder nur ungenau dokumentiert wurden, oder wenn das Verhalten von Teilen des Prozessors nicht vorhergesagt werden kann, z. B. bei Random-Replacement-Caches. In diesem Fall kann das Zeitverhalten nur konservativ abgeschätzt werden, was meist zu einer hohen Überschätzung der WCET führt.

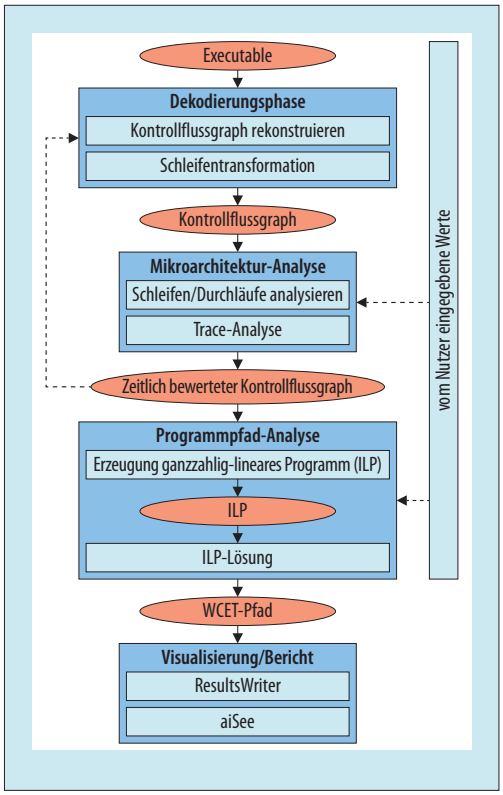
Bei Multicore-Prozessoren ist zusätzlich zu beachten, dass die parallel laufenden Kerne sich gegenseitig im Zeitverhalten beeinflussen können. Diese sogenannten Interferenzen entstehen dann, wenn mehrere Kerne gleichzeitig auf eine gemeinsam genutzte Ressource zugreifen möchten, z. B. den Speicher oder den Bus. Abhängig von den jeweiligen Arbitrierungsregeln wird einer der Zugriffe abgearbeitet, während die anderen warten. Die vereinfachende Annahme, dass jeder Zugriff auf eine gemeinsam genutzte Ressource warten muss, führt üblicherweise zu unrealistisch überschätzten Ergebnissen. Ziel muss daher sein, Interferenzen möglichst zu vermeiden, und falls dies nicht möglich ist, sie geeignet abzuschätzen. In diesem Fall bietet sich die hybride WCET-Analyse an.

Hybride Zeitanalyse

Statische Programmanalyse und Messwerte können zu einem hybriden Ansatz kombiniert werden, der die Vorteile der statischen Analyse (vollständige Codeabdeckung bei der Bestimmung von Schleifengrenzen und nichtausführbaren Programmpfaden) mit den Vorteilen der Messung auf der Ziel-Hardware (Beobachtbarkeit des Zeitverhaltens und der Interferenzen) kombiniert. Das Zeitanalyse-Werkzeug TimeWeaver basiert auf solch einem hybriden Ansatz (**Bild 1**):

➜ Zuerst wird das Binärprogramm eingelesen, um daraus den Kontrollflussgraphen (CFG) zu rekonstruieren. Berechnete Sprünge und Funktionsaufrufe, deren Ziele nicht statisch bestimmt werden können, können mithilfe der späteren Trace-Analyse-Phase aufgelöst werden. Die so durchgeführte iterative Decodierung des Binärprogramms reduziert die nötigen Benutzerannotationen auf ein Minimum.

➜ Auf dem so erzeugten CFG werden in der nächsten Phase verschiedene statische Programmanalysen durchgeführt. Neben Schleifengrenzen werden so die möglichen Inhalte von Registern und Speicherzellen bestimmt. Diese Information wird genutzt, um nicht ausführbare Pfade von vornherein von der späteren Pfadanalyse auszuschließen und damit die Präzision des Ergebnisses zu erhöhen.



(Quelle: AbsInt)

Bild 1. Struktur des Zeitanalysewerkzeugs TimeWeaver.

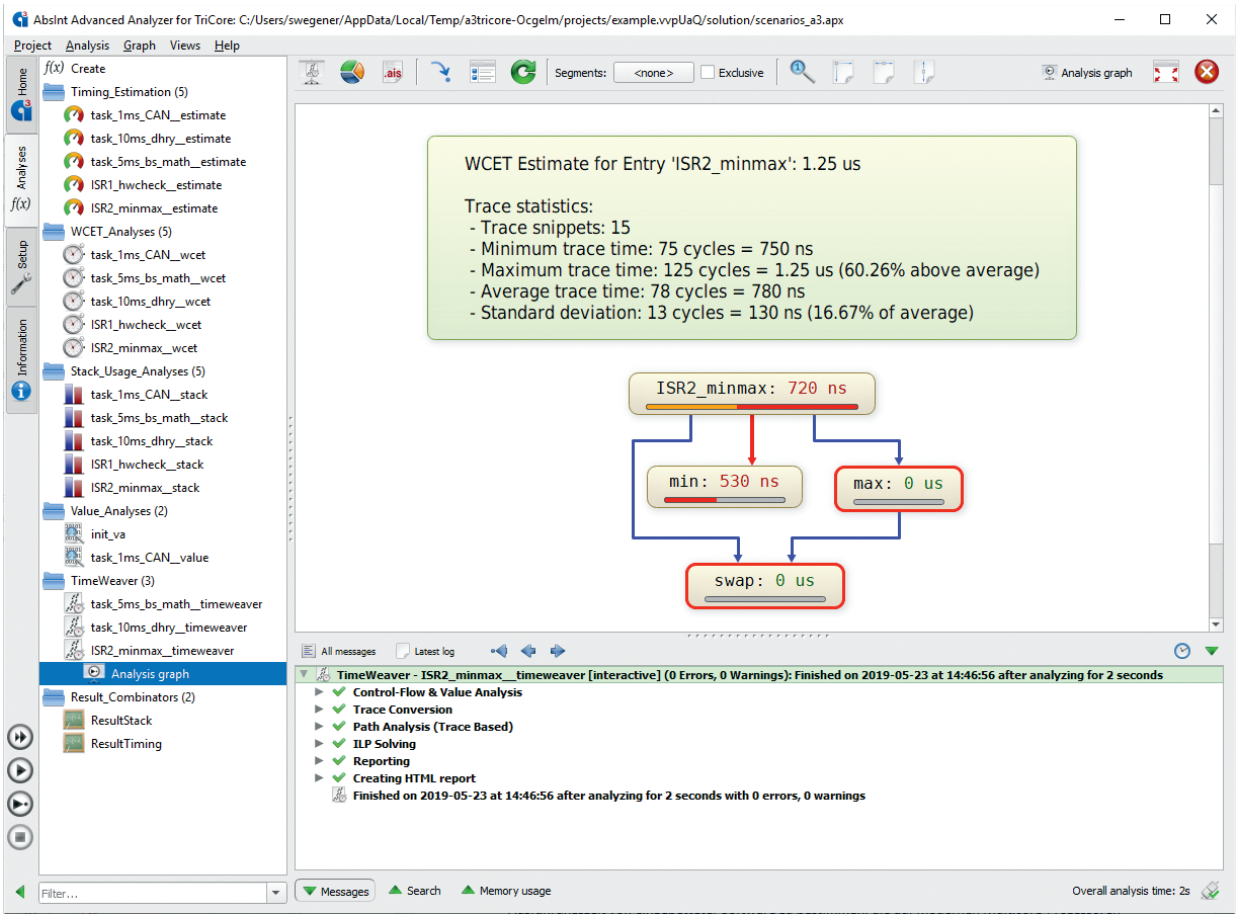


Bild 2. Visualisierung des Analyse-Ergebnisses. (Bild: AbsInt)

➜ Danach werden die Eingabe-Traces verarbeitet. Hierbei werden die Traces in kleine Teilstücke zerlegt, die sogenannten Trace-Segmente. Die zu den jeweiligen Trace-Segmenten gehörige Zeitinformation wird auf das jeweils passende Teilstück des CFG abgebildet. Wenn mehrere Trace-Segmente Zeitinformation für das gleiche Teilstück des CFG enthalten, wird die gespeicherte Ausführungszeit maximiert. Außerdem können auf Wunsch auch Schleifengrenzen aus den Traces extrahiert werden.

➜ Aus dem mit Zeitinformationen angereicherten CFG wird ein Optimierungsproblem in Form eines Ganzzahlig-Linearen-Programms (ILP) erzeugt. Die Lösung dieses ILP ergibt die maximierte Ausführungszeit sowie den dazugehörigen Pfad durch das Programm, der die längste Ausführungszeit verursacht.

➜ In einem letzten Schritt werden die Ergebnisse der verschiedenen Analysephasen für den Benutzer grafisch aufgearbeitet (**Bild 2**) und Berichte in verschiedenen Formaten erzeugt.

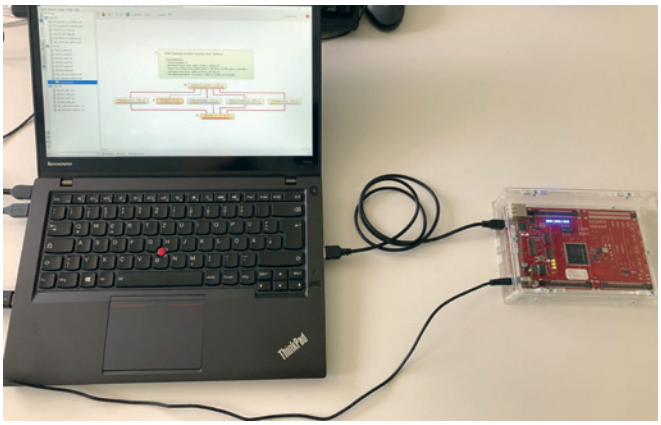


Bild 3. Direkte Anbindung eines Infineon Aurix-Testboards ohne zusätzlichen Debugger. (Bild: AbsInt)

Die Vorteile der hybriden Zeitanalyse mit TimeWeaver zeigen sich insbesondere gegenüber rein messbasierten Verfahren. Dank der Aufteilung in Trace-Segmente wird eine zufriedenstellende Pfadabdeckung schneller erreicht. Nicht jeder mögliche Pfad im Programm muss einzeln vermessen werden, da die Pfadanalyse den global schlimmsten Pfad aus den Trace-Segmenten extrapoliert. Zur Berechnung einer sicheren oberen Schranke genügt die Stimulation von Worst-Case-Hardware-Zuständen pro Segment anstatt pro kompletten Pfad. Gleichzeitig wird, dank der Maximierung der beobachteten Zeiten für jedes einzelne Trace-Segment, jeder beobachtete Hardware-Zustand berücksichtigt. Auch undokumentierte oder unvorhersagbare Hardware wird korrekt und so präzise wie möglich behandelt. Die auf der echten Ziel-Hardware beobachteten Zeiten bilden die Grundlage der WCET-Abschätzung. Auch der „probe effect“ wird umgangen, da TimeWeaver die Hardware-Unterstützung moderner Mikroprozessoren zur nicht-intrusiven Aufzeichnung von Traces ausnutzt.

Die umfangreichen Visualisierungs- und Reporting-Möglichkeiten erlauben eine feingranulare Analyse: TimeWeaver berechnet nicht nur die eine Abschätzung der WCET für das Gesamtprogramm, sondern zeigt auch, wo in den einzelnen Routinen die meiste Zeit verbraucht wird.

Hotspots müssen nicht mehr geraten, sondern können einfach gefunden werden. Interferenzen durch parallel laufende Programme auf anderen Kernen sind sichtbar und werden bei der Berechnung des Ergebnisses berücksichtigt. Unterbrechungen durch Task-Wechsel und Interrupts werden herausgerechnet. Damit ist TimeWeaver insbesondere für Multicore-Prozessoren geeignet. Zusätzlich berechnet TimeWeaver die Codeabdeckung der Eingabe-Traces. Damit ist eine Einschätzung möglich, ob weitere Messungen durchgeführt werden müssen. Routinen ohne Messabdeckung werden hervorgehoben.

Unterstützte Plattformen und Trace-Formate

TimeWeaver unterstützt verschiedene Prozessor-Architekturen mit eingebauten Trace-Einheiten (PowerPC mit Unterstützung für IEEE-ISTO 5001 NEXUS-Traces, ARM mit Unterstützung zyklenakkurater ETM-Traces, TriCore/Aurix mit Unterstützung für MCDS-Traces) sowie verschiedene Trace-Formate führender Debugger-Hersteller (iSystem, Lauterbach, PLS). Aurix Emulation Devices können auch direkt mittels des Infineon DAS Trace Servers angebunden werden, ohne dass ein zusätzlicher Debugger nötig ist (**Bild 3**).

Transparente Performance

Die hybride WCET-Analyse mit TimeWeaver ermöglicht es, Schranken für die maximale Ausführungszeit von eingebetteter Software zu bestimmen, die auf modernen Multicore-Prozessoren ausgeführt wird. Die Ergebnisse basieren dabei auf Zeiten, die aus Traces von der realen Hardware extrahiert wurden, ohne den Code hierfür instrumentieren zu müssen. Neben der berechneten Laufzeitschranke inklusive des kritischen Pfads durch die Software präsentiert TimeWeaver auch die Codeabdeckung der genutzten Messungen. Die Ergebnisse werden feingranular dargestellt, sodass die Performance der Software beurteilt und Hotspots einfach identifiziert werden können. *jk*

Simon Wegener schloss sein Informatik-Studium an der Universität des Saarlandes mit dem Master-Abschluss ab. Seit 2011 arbeitet er in der Produktentwicklung der AbsInt Angewandte Informatik GmbH. Hier spezialisierte er sich auf die statische Binärcode-Analyse. Er ist Autor mehrerer wissenschaftlicher Veröffentlichungen zum Thema Zeitanalyse und hat in verschiedenen deutschen und europäischen Forschungsprojekten mitgearbeitet.