Juli 2015 € 12,80

elektroniknet.de

funktionale sicherheit

powered by

Elektronik
Fachmedium für industrielle Anwender und Entwickler

Elektronik automotive
Fachmedium für professionelle Automobilelektronik

Anforderungsbasiertes Testen – aber richtig

>> Seite 26

dSPACE

„In vielen Industriezweigen werden Geräte sorglos und ohne große Sicherheitsvorkehrungen mit dem Internet verbunden."

>> Seite 6

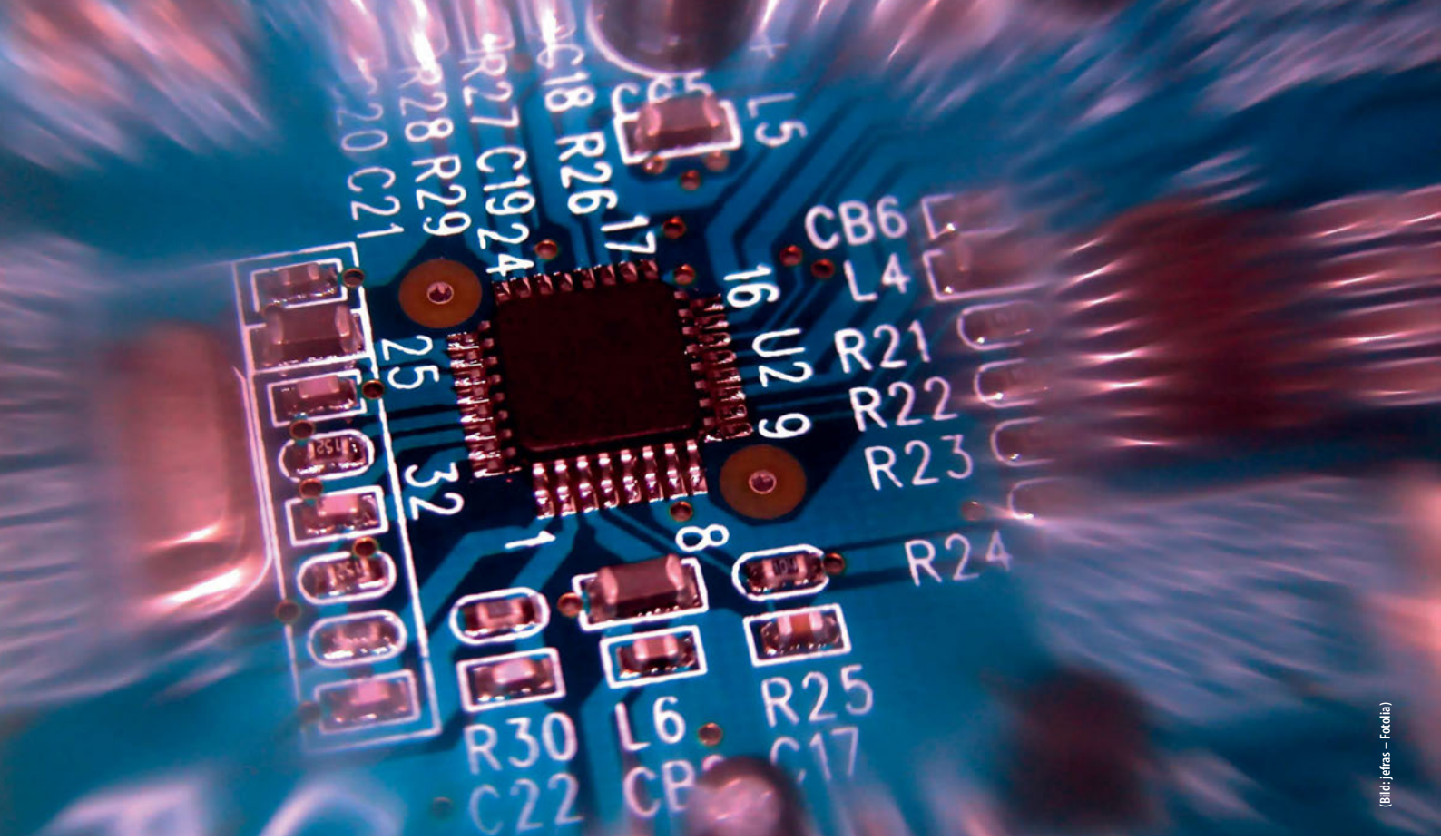
Dan Smith, Principial Engineer bei der Barr Group

Einheitliche Sicherheitsbegriffe in der Normung

>> Seite 12

Prozessoren fit für ADAS-Funktionen

>> Seite 44



(Bild: jefas – Fotolia)

Vorhersage von Rechenleistung unabdingbar:

Multicore-CPUs in sicherheitskritischen Echtzeitsystemen

Viele Multicore-Designs weisen Eigenschaften auf, die ihren Einsatz in sicherheitskritischen Echtzeitsystemen erschweren. Um ein Scheitern in der Integrationsphase zu vermeiden, müssen die Architektureigenschaften frühzeitig auf Echtzeitfähigkeit untersucht und der Prozessor im Hinblick auf vorhersagbare Rechenleistung konfiguriert werden.

Von Dr. Daniel Kästner

Multicore-Prozessoren sind schon seit Jahren im PC-Bereich etabliert. Die Parallelisierung durch mehrere Rechenkerne auf einem Chip ermöglicht eine deutliche Beschleunigung von Systemen, die aus vielen unabhängigen Threads bestehen. Zudem tragen die niedrigeren Taktraten zu einer höheren Energieeffizienz bei. Nicht zuletzt wegen stark gesunkener Preise werden Multicores zunehmend auch in eingebetteten Systemen verwendet.

Im Desktop-Bereich ist eine hohe durchschnittliche Leistung ausschlaggebend; ein sicherheitskritisches, eingebettetes Echtzeitsystem hingegen muss nicht nur logisch korrekt sein, sondern auch ein korrektes Zeitverhalten aufweisen. Überschreitet eine Echtzeit-Task ihre Deadline, kann dies zu schweren Fehlern führen. Zum Nachweis, dass eine Task vor ihrer Deadline beendet ist, muss ihre längstmögliche Ausführungszeit bekannt sein, die sogenannte Worst-Case Execution Time (WCET). In einem Multitasking-System können Tasks auch unterbrochen oder blockiert werden. Dies wird bei der Antwortzeit der Task berücksichtigt, der Worst-Case Response Time (WCRT), bei der Unterbrechungseffekte zur WCET hinzuaddiert werden.

Alle aktuellen Sicherheitsnormen, darunter DO-178B/DO-178C, ISO 26262, IEC 61508 und EN 50128, erfordern den Nachweis, dass genügend Ressourcen vorhanden sind, um korrekte Systemfunktion zu gewährleisten. Hierzu zählen der Nachweis sicherer oberer Schranken des maximalen Speicherverbrauchs und der maximalen Ausführungs- und Antwortzeiten. Bei Anwendungen gemischter Kritikalität unterliegt zudem die gesamte Anwendung der höchsten vorkommenden Sicherheitsanforderungsstufe, es sei denn, dass nachgewiesen werden kann, dass alle Sicherheitsfunktionen räumlich und zeitlich voneinander unabhängig sind.

Methodisch bestimmen

Wie kann nun das Zeitverhalten methodisch bestimmt werden? Bei direkten Zeitmessungen mit Logikanalysatoren bzw. Debuggern oder Hardware-Simulation werden Zeiten nur für jeweils eine konkrete Eingabe ermittelt. Eine vollständige Testabdeckung ist in der Regel

nicht zu realisieren, ein sicheres Testende-Kriterium nicht vorhanden. Hinzu kommt, dass – z.B. aufgrund von Caches – bei allen modernen Prozessoren das Zeitverhalten einer Instruktion von den zuvor ausgeführten Instruktionen abhängt, sodass selbst eine MC/DC-Abdeckung zur Bestimmung der längstmöglichen Ausführungszeit unzureichend ist. Verfahren auf Basis von Code-Instrumentierung verändern den Code, was das Verhalten der Cache-Speicher signifikant verändern kann. Die gemessenen Zeiten der instrumentierten Software können weit vom Zeitverhalten des ursprünglichen Systems abweichen.

Eine Methode, Deadline-Überschreitungen auszuschließen, ist die sichere statische Analyse. Statische Analysen werden als sicher bezeichnet, wenn mathematisch bewiesen wurde, dass sie keine potenziellen Fehler übersehen. Solche Beweise sind durch eine abstrakte Interpretation möglich, eine formale Methode für statische Programmanalysen [1]. Statische-Analyse-Tools auf Basis der abstrakten Interpretation haben sich in den letzten Jahren zunehmend verbreitet und können als Stand der Technik zum Nachweis nichtfunktionaler Korrektheitseigenschaften angesehen werden. Räumliche Unabhängigkeit kann durch sichere statische Analysatoren nachgewiesen werden, die die Abwesenheit von Stack-Überläufen, ungültigen Zeigerzugriffen oder anderen Laufzeitfehlern beweisen können [2, 3]. Garantierte obere Schranken für die längstmögliche Ausführungszeit von Tasks können beispielsweise durch den Analysator aiT von AbsInt berechnet werden [4]. **Bild 1** zeigt die Kontrollflussgraph-Visualisierung von aiT mit kritischem Ausführungspfad.

Statische Analyse liefert präzise Ergebnisse

Statische WCET-Analyse ist für komplexe Prozessoren einsetzbar und liefert präzise Resultate [5]. Eine Voraussetzung ist jedoch, dass das Zeitverhalten des Prozessors vorhersagbar ist [6]. Bereits bei Single-Core-Prozessoren wird die Vorhersagbarkeit durch spekulative Hardware-Mechanismen wie Caches, Out-of-Order Pipelining oder Sprungvorhersage-Mechanismen beeinträchtigt. Wird eine Task T1 auf einem Prozessor mit Caches von einer anderen Task unterbrochen, können von T1 benötigte Speicherblöcke aus dem Cache verdrängt werden. Wird T1 nach der

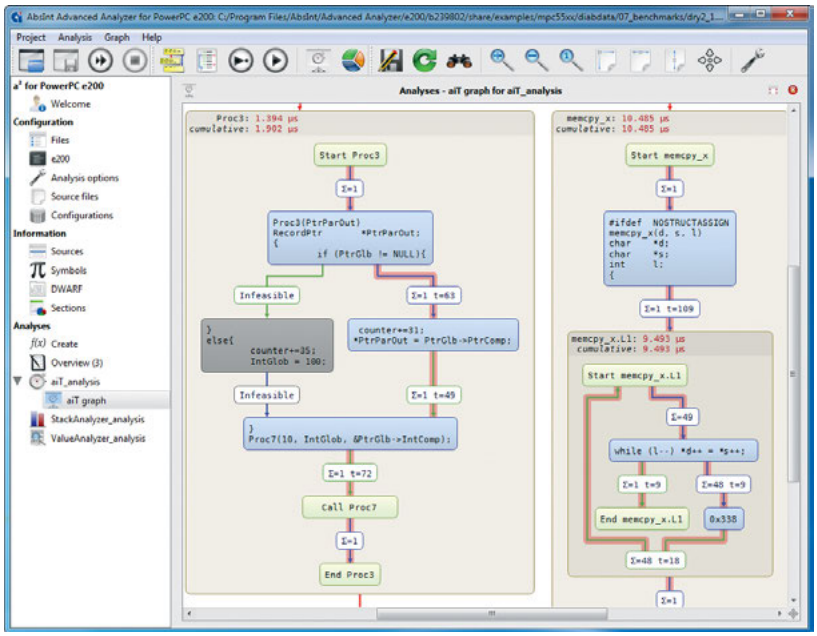


Bild 1. Kontrollflussgraph-Visualisierung von aiT mit kritischem Ausführungspfad und WCET-Annotationen. (Quellen: Absint)

Unterbrechung fortgesetzt, können dadurch zusätzliche Cache Misses auftreten. Eine Berücksichtigung aller möglichen Unterbrechungsszenarien ist unmöglich durch Messungen zu realisieren. Cache-bezogene Unterbrechungskosten können von statischen Analysen wie aiT berücksichtigt werden, aber der Unterschied zwischen durchschnittlicher und maximaler Ausführungszeit wächst. Durch Cache-Partitionierung oder Cache Locking lässt sich die Vorhersagbarkeit verbessern; die mögliche Varianz der Ausführungszeit sinkt. An diesem Beispiel wird deutlich, dass durch Zugriffe auf gemeinsam genutzte Hardware-Ressourcen Interferenzen entstehen können, die die Vorhersagbarkeit beeinträchtigen, da die Abfolge der Zugriffe stark variieren kann.

Bei Multicore-Prozessoren sind nicht nur die Interferenzen innerhalb der einzelnen Cores zu berücksichtigen; hinzukommen sind nun auch Interferenzen durch Zugriffe auf gemeinsam genutzte Ressourcen von unterschiedlichen Cores. Greifen mehrere Cores auf denselben Cache zu und läuft auf jedem Core ein separater Scheduler, kann die Ausführungszeit einer Task auch durch eine völlig andere Applikation beeinflusst werden, die auf einem anderen Core läuft und nicht der Steuerung desselben Scheduler untersteht. Da verschiedene Applikationen meist erst in der Integrationsphase gemeinsam betrachtet werden können, besteht zudem die Gefahr, dass Timing-Probleme erst sehr spät erkannt werden. **Bild 2** zeigt den Nachweis der Abwesenheit von Laufzeitfehlern und Race Conditions mit Hilfe von Astrée. Astrée ist eine Software zur statischen Programmanalyse, die C-Programme automatisch auf Laufzeitfehler überprüft.

Astrée analysiert handgeschriebene oder automatisch erzeugte C-Programme mit komplexer Speichernutzung, aber ohne Rekursion oder dynamische Speicherallokation. Damit bietet sich Astrée vor allem zur Analyse von sicherheitskritischen eingebetteten Anwendungen an, insbesondere in den Bereichen Transport, Medizintechnik, Nuklearanlagen, Luft- und Raumfahrt.

Astrée überprüft, ob die C-Sprache korrekt eingesetzt wurde, und sucht nach Laufzeitfehlern in allen möglichen Ausführungsszenarien unter allen möglichen Bedingungen. Dabei meldet es jeden Gebrauch der Sprache, der laut ISO/IEC 9899:1999, der internationalen Norm für C, ein undefiniertes Verhalten aufweist, sowie jeden Verstoß gegen Hardware-spezifische Einschränkungen der Implementierung.

Im Einzelnen meldet Astrée Division durch Null, Feldzugriffe außerhalb der gültigen Feldgrenzen, ungültige Zeigermanipulationen und -dereferenzierungen (NULL-Zeiger, uninitialisierte und hängende Zeiger), ganzzahlige arithmetische Überläufe und Gleitkommaüberläufe, Verstöße gegen optionale benutzerdefinierte Annahmen zur Überprüfung von sonstigen Laufzeiteigenschaften (ähnlich der „assert“-Diagnostik), garantiert unerreichbaren Code sowie Lesezugriffe auf uninitialisierte Variablen. Gleitkommaberechnungen werden von Astrée präzise und sicher behandelt. Alle möglichen Rundungsfehler werden stets berücksichtigt.

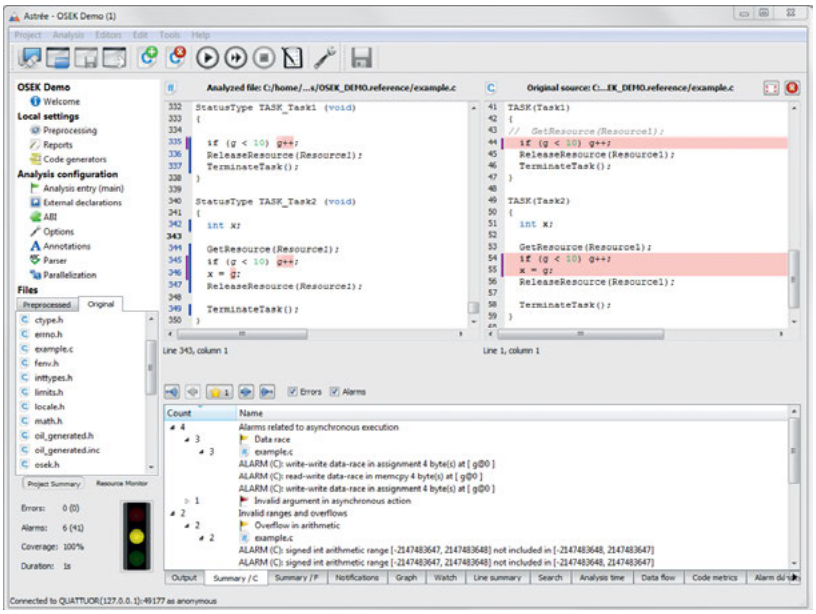


Bild 2. Nachweis der Abwesenheit von Laufzeitfehlern und Race-Conditions mit Astrée.

Caches mit Konfliktpotenzial

Die Gewährleistung von Zeitschranken auf Multicore-Prozessoren wurde in zahlreichen nationalen und internationalen Forschungsprojekten untersucht – Beispiele sind Predator, Certainty oder Aramis. Dennoch stehen dem Einsatz von Multicore-Systemen in sicherheitskritischen Anwendungen auch heute noch Hürden entgegen, die in einem 2014 veröffentlichten Report der Luftfahrtsicherheitsbehörden zusammengefasst werden [7]. Aktuelle Multicore-Designs wurden nicht im Hinblick auf vorhersagbare Rechenleistung entwickelt. Konflikte können z.B. bei Zugriffen auf gemeinsame Caches, aber auch auf gemeinsame Speicherbänke oder gemeinsame Flash Prefetch Buffers auftreten. Für eine gegebene Multicore-Architektur müssen mögliche Interferenzen sorgfältig untersucht und eine Konfiguration bestimmt werden, die eine vorhersagbare Ausführung ermöglicht.

In [6] werden Konfigurationsbeispiele für zwei aktuelle Multicore-Prozessoren gegeben, die eine vorhersagbare Rechenleistung zulassen. Zu den Konfigurierungsmaßnahmen zählen die exklusive Zuordnung einzelner Speicherbänke oder Prefetch Buffers zu bestimmten Cores oder die Verwendung eines gemeinsamen L2-Cache als Scratchpad für einen einzelnen Core. Sind mehrere Cores durch ein gemeinsames Bussystem an den Hauptspeicher angebunden, müssen die Aktivitäten zwischen den Cores koordiniert werden, um Zugriffskonflikte zu kontrollieren [8]. Der Nachweis des korrekten Zeitverhaltens von Echtzeitsystemen ist von entscheidender Bedeutung für die Systemkorrektheit. Statische Analysatoren auf Basis der abstrakten Interpretation liefern garantierte obere Schranken für die Ausführungszeit von Tasks. Statische Analyseverfahren werden von aktuellen Sicherheitsstandards empfohlen. Bei Multicore-Architekturen für sicherheitskritische Echtzeitsysteme muss die Hardware sorgfältig im Hinblick auf vorhersagbare Performance untersucht und geeignet konfiguriert werden, um Integrations- und Stabilitätsprobleme zu vermeiden. *fr*

Literatur

[1] *Kästner, D.:* Applying Abstract Interpretation to Demonstrate Functional Safety. In Boulanger, J.-L., editor, Formal Methods Applied to Industrial Complex Systems, ISTE/Wiley, London, UK, 2014.

[2] *Kästner, D.; Ferdinand, C.:* Proving the Absence of Stack Overflows. Proceedings of the 33th International Conference on Computer Safety, Reliability and Security (SAFECOMP), Florence, 2014. Springer LNCS 8666, Springer, Heidelberg.

[3] *Kästner,D.; Wilhelm, S.; Nenova, S.; Cousot,P.; Cousot,R.;. Feret,J.; Mauborgne,L.; Miné,A.; Rival, X.:* Astrée: Proving the Absence of Runtime Errors. Embedded Real Time Software and Systems Congress ERTS^2, Toulouse, 2010.

[4] www.absint.com/ait.

[5] *Souyris,J.; Le Pavec,E.; Himbert,G.; Jégu,V.; Borios,G.; Heckmann,R.:* Computing the worst case execution time of an avionics program by abstract interpretation. Proceedings of the WCET Workshop, 2005.

[6] *Cullmann,C.; Ferdinand,C.; Gebhard,G.; Grund,D.; Burguière,C.; Reineke,J.; Triquet,B.; Wegener,S.; Wilhelm,R.:* Predictability Considerations in the Design of Multi-Core Embedded Systems. Ingénieurs de l'Automobile, volume 807, 2010.

[7] Certification Authorities Software Team (CAST). Position Paper CAST-32: Multi-core Processors. 2014. www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-32.pdf.

[8] *Nowotsch,J.; Paulitsch,M.; Bühler,D.; Theiling,H.; Wegener,S.; Schmidt,M.:* Multi-core Interference-Sensitive WCET Analysis Leveraging Runtime Resource Capacity Enforcement. 26th Euromicro Conference on Real-Time Systems (ECRTS^2 2014).

Dr. Daniel Kästner

arbeitet als CTO bei der AbsInt GmbH in Saarbrücken.

kaestner@absint.com