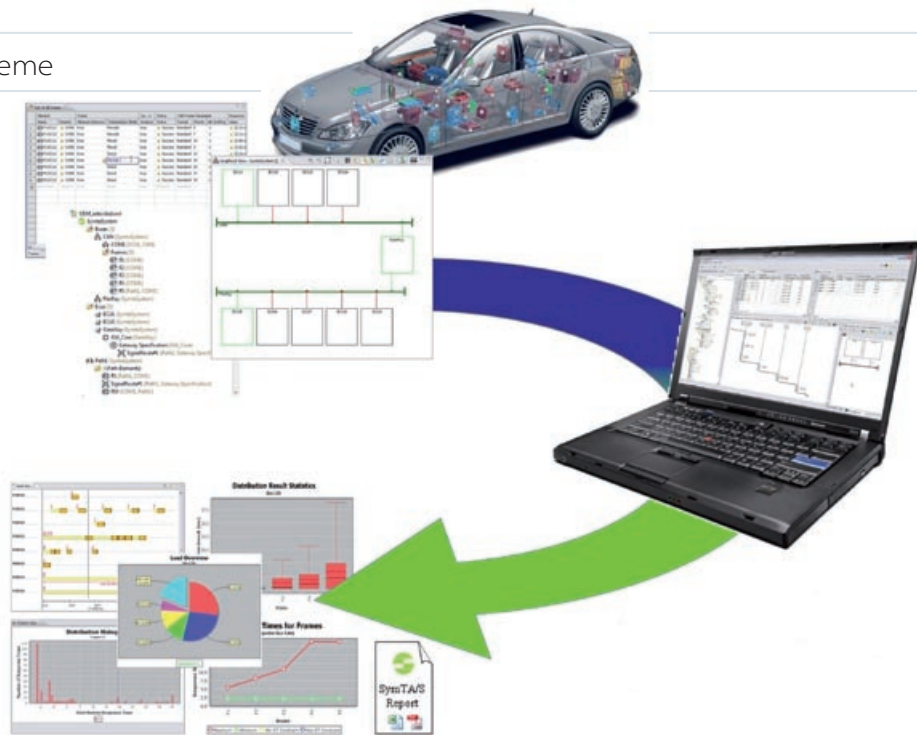


Echtzeit in AUTOSAR in Serie

Methoden und Prozesse für die Steuergerätekonfiguration in AUTOSAR-Anwendungen



AUTOSAR ist als Standard für Steuergeräteentwicklungen mittlerweile eingeführt und wird in Serienprojekten, je nach OEM-Vorgabe, in der Version 3.2 oder 4.0 eingesetzt; letztere inklusive der Timing Extensions.

Von Christoph Ficek und Dr. Kai Richter

aus Sicht der Software-Entwicklung hat AUTOSAR durch diese Standardisierung bereits eine enorme Produktivitätssteigerung ermöglicht. In der System-Entwicklung tauchen allerdings, speziell im Kontext einer Echtzeit-Fähigkeit, weitere Fragen auf:

- ▶ Realisierbarkeit: Wie viel Software kann auf einem Steuergerät ablaufen, damit noch alle Echtzeit-Anforderungen erfüllt werden?
- ▶ Sicherheit, Verfügbarkeit, Erweiterbarkeit: Welche technischen und kommerziellen Auswirkungen haben Design-Änderungen?
- ▶ Dokumentation der Echtzeit-Fähigkeit und Anforderungen: Wie müssen und/oder lassen sich Echtzeit-Anforderungen dokumentieren bzw. spezifizieren?

Wunschgemäß sollte am Ende der Entwicklung eine Systemintegration zur Verfügung stehen, welche die oben genannten Anforderungen erfüllt, die Steuergeräte-Ressourcen optimal nutzt und Platz für Erweiterungen hat. Die Schritte bis zur dieser Systemintegration sind unter anderem die Umsetzung von Funktionsarchitekturen in Software-Architekturen, die Erstellung

der RTE- und Betriebssystemkonfiguration (Schedule) und die Integration der Basis-Software-Elemente.

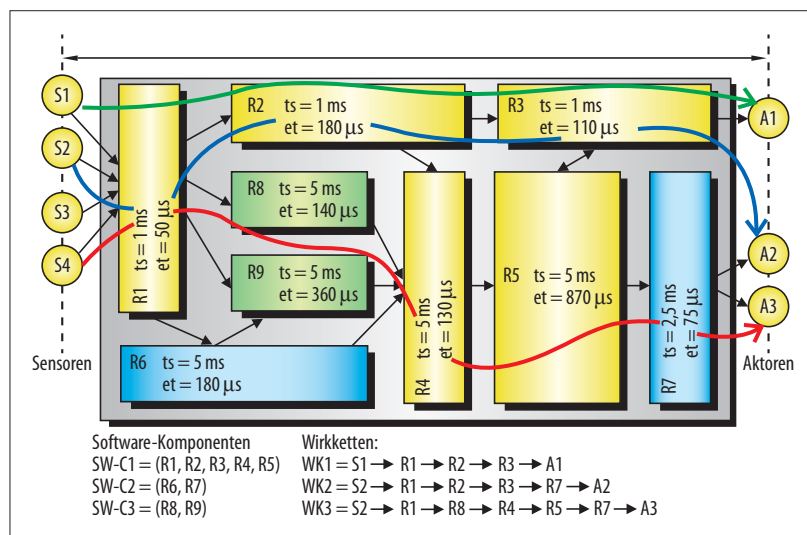
In Hinblick auf die Sicherstellung der Echtzeit-Fähigkeit sind die Erzeugung und die Überprüfung der Steuergerätekonfiguration (mit Runnable-Mapping, Task-Layout und Schedule-Konfiguration) besonders wichtig. Hier helfen Timing-Analysen dabei, die Konfigurationen zu bewerten, zu dokumentieren und ggf. Anforderungen an die Zulieferer zu formulieren.

Weiterhin liefern sie die Grundlage für anstehende Entwurfsentscheidungen.

Geeignete Timing-Analyse-Methoden existieren bereits und sind vielfach im Einsatz. Die wichtigsten Aspekte dabei sind die CPU-Auslastung – aufgeschlüsselt nach Software-Komponenten, Tasks und Runnables – und die Zusammenhänge zwischen Wirkketten und Software-Architektur. Weiterhin wichtig sind die Zeitbedingungen (Event Chains) für Wirkketten und das Scheduling der Systemkonfiguration, also der Summe aller zu integrierenden Funktionen.

■ Last, Zyklus- und Ausführungszeiten

Die prozentuale Auslastung einer CPU ergibt sich aus der Summe der Last



■ Bild 1. Software-Architektur mit Timing-Informationen und Wirkketten.

aller ausgeführten Funktionen und Komponenten. Die Last einer einzelnen Funktion wiederum ergibt sich aus dem Quotienten von Laufzeit und Zykluszeit der Funktion.

Die Zykluszeit einer Funktion ist in der Funktionsmodellierung, z.B. in Matlab/Simulink oder Ascet, bekannt und als Periode oder Sample Time beschrieben. Bei nicht periodischen Prozessen können die Aktivierungshäufigkeiten aus der Funktionssimulation abgeleitet werden.

Für die Ermittlung der Laufzeit einer Funktion stehen je nach Fortschritt der Entwicklung verschiedene Methoden zur Verfügung. Aus früheren Projekten können Laufzeiten-Schätzungen am Anfang der Entwicklung verwendet werden. Budgetierung ist ebenfalls eine häufig eingesetzte Methode. Stehen im Laufe des Projektes erste Implementierungen zur Verfügung, können diese auf Prozessorsimulatoren (z.B. von Vast), auf Prototypen-Hardware oder später auf der echten Ziel-Hardware ausgemessen werden (z.B. mit Gliwa T1). Eine statische Laufzeitanalyse (z.B. von AbsInt) liefert zudem zuverlässige obere Laufzeitschranken.

In **Bild 1** ist ein Beispiel mit drei Software-Komponenten (SW-C) mit jeweils mehreren Runnables und den entsprechenden Laufzeiten (execution time, et) und Zykluszeiten (time sample, ts) zu sehen. Die **Tabelle** zeigt die Auswertung einer Lastanalyse für das gesamte System (71 Prozent) sowie für die einzelnen, dazu beitragenden Teile.

Die Laufzeit einzelner Funktionen und Komponenten wird im Projektverlauf durch eine immer detailreichere Systembeschreibung zunehmend genauer; so wird auch das Lastmodell im Projektverlauf verfeinert. Damit kann

man kritische Lastsituationen frühzeitig erkennen und gezielt entgegenwirken, beispielsweise durch Reduzierung der Laufzeit-Budgets und/oder Erhöhung der Zykluszeiten.

Die Aussage über die erwartete Last stellt einen ersten Anhaltspunkt für die Realisierbarkeit eines Systems dar, jedoch keine hinreichende Garantie. Integrations- und Scheduling-Effekte wie Jitter, Wirkketten-Latenzen oder temporäre Lastspitzen werden innerhalb einer reinen Lastbetrachtung nicht berücksichtigt und müssen gesondert untersucht werden.

Task-Generierung und Scheduling-Analyse

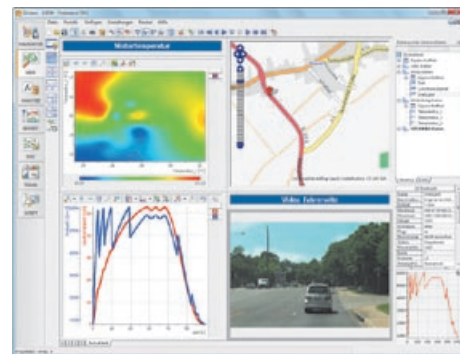
Aus den Runnables kann ein erster virtueller Task-Schedule aufgebaut werden, wobei für jedes Projekt ein geeignetes Scheduling-Konzept zugrunde gelegt werden muss:

- ▶ Runnables gleicher Zykluszeit werden in einem Task gruppiert.
- ▶ Je kürzer die Zykluszeit, desto höher die Task-Priorität.
- ▶ Die Task-Deadline entspricht der Zykluszeit (keine Mehrfachaktivierungen).

Diese Informationen werden in AUTOSAR innerhalb der RTE- und Steuergerätekonfiguration nachgehalten. Der Schritt der virtuellen Task-Generierung lässt sich somit weitgehend automatisieren und um projektspezifische Regeln erweitern.

Anhand des virtuellen Task-Schedules wird nun eine Scheduling-Analyse durchgeführt, welche die Antwortzeit der Tasks, d.h. die Zeit zwischen der Aktivierung und dem Ende des Tasks, ermittelt. Dabei wird mit den Unterbre-

NI DIAdem



Von Fahrzeugdaten zum technischen Wissen mit industrieller Standardsoftware

- Interaktive Analyseroutine, einschließlich Crash-Analysen, nutzen
- Professionelle, druckreife Ergebnisberichte mit Grafen und Diagrammen erstellen
- Daten aus beliebigen Dateien oder Datenbanken, z.B. ASAM ODS, verwalten und auswerten

>> Sparen Sie Zeit mit DIAdem unter:

ni.com/diadem/d

089 7413130



National Instruments Germany
 Ganghoferstraße 70 b • 80339 München
 Tel.: +49 89 7413130 • Fax: +49 89 7146035
ni.com/germany • info.germany@ni.com

©2011 National Instruments. Alle Rechte vorbehalten. CompactRIO, LabVIEW, National Instruments, NI und ni.com sind Warenzeichen von National Instruments. Andere erwähnte Produkt- und Firmennamen sind Marken oder Handelsbezeichnungen der jeweiligen Unternehmen. Druckfehler, Irrtümer und Änderungen vorbehalten.

Name	SW-C	Zyklus	Laufzeit	Lastanalyse				
				Runnable	SW-C 1	SW-C 1	SW-C 1+2	SW-C 1-3
R1	SW-C 1	1	0,05	5,0 %				
R2	SW-C 1	1	0,18	18,0 %				
R3	SW-C 1	1	0,11	11,0 %				
R4	SW-C 1	5	0,13	2,6 %				
R5	SW-C 1	5	0,87	17,4 %	54 %	54 %		
R6	SW-C 2	5	0,18	3,6 %				
R7	SW-C 2	2,5	0,075	3,0 %	6,6 %		61 %	
R8	SW-C 3	5	0,14	2,8 %				
R9	SW-C 3	5	0,36	7,2 %	10,0 %			71 %

I Tabellarische Darstellung der Lastanalyse

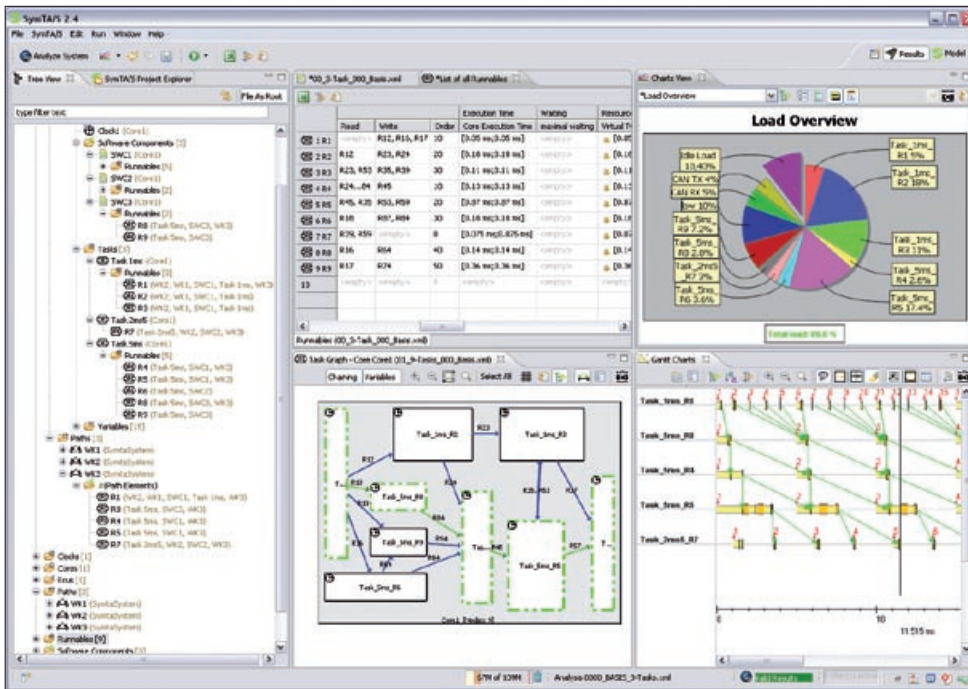


Bild 2. SymTA/S-Timing-Modell mit Last-, Schedule- und Wirkkettenanalyse.

chungen durch Tasks höherer Priorität ein essentieller Integrationseffekt berücksichtigt. Entsprechend der aufgestellten Bedingung muss ein Task vor seiner nächsten Aktivierung vollständig ausgeführt sein (Deadline). Ist dies nicht der Fall, dann können zur Laufzeit Task-Aktivierungen verlorengehen, und die korrekte Funktion der Applikation kann nicht garantiert werden. In frühen Phasen der Entwicklung sollten die Antwortzeiten 30 bis 50 Prozent unter der Deadline liegen, um genügend Platz für Erweiterungen zu haben.

Wird eine Deadline verletzt, so müssen entsprechende Maßnahmen getroffen werden, falls möglich beispielsweise die Wahl einer höheren Zykluszeit für ein Runnable. Durch Code-Optimierung können gegebenenfalls die Laufzeiten von Runnables reduziert werden, was die eigene Laufzeit, aber auch die der unterbrechenden Tasks vermindert und die Antwortzeit verkürzt. Ebenso kann die Wahl einer schnelleren CPU für das gesamte System Abhilfe schaffen. Eine Automatisierung dieser Scheduling-Analyse hilft dabei, mehrere Optionen zu explorieren. Das Scheduling-Analyse-Werkzeug SymTA/S bietet diese Automatisierung und reduziert das Risiko von späten und teuren Design-Änderungen. Sind die individuellen Antwortzeiten der Runnables im erlaubten Bereich, gilt es, die Wirkketten auf ihre Latenzen hin zu untersuchen.

Wirkketten-Analyse

Wirkketten beschreiben die Kommunikation zwischen Runnables und sind vergleichbar mit Signalpfaden (oder Signalwegen in Simulink bzw. Ascet). Die Kommunikation zwischen Runnables ist realisiert durch Datenkommunikation, die von einem Runnable erzeugt und vom nächsten Runnable in der Kette gelesen wird. Im dem Bei-

spiel aus Bild 1 sind die drei Wirkketten WK1, WK2 und WK3 gezeigt, die jeweils bei einem Sensor S beginnen und an einem Aktor A enden. In der Praxis unterliegen solche Wirkketten Zeitbedingungen, wie z.B. Reaktionszeiten und Regler-Totzeiten. Solche Wirkketten lassen sich in SymTA/S leicht spezifizieren.

Die Latenz einer Wirkkette ist abhängig von den Antwortzeiten der einzelnen Runnables, den Pufferzeiten zwischen Runnable-Ausführungen oder Up-/Down-Sampling-Effekten beim Wechsel von Zykluszeiten. Da die Antwortzeiten der Runnables selbst durch Unterbrechungen von Tasks höherer Priorität beeinflusst werden, sind auch die Wirkketten in höchstem Maße und direkt davon betroffen. Dies ist im Gantt-Chart rechts unten in Bild 2 für die Wirkkette WK3 zu sehen.

Die SymTA/S-Timing-Analyse ist in der Lage, die Latenz einer Wirkkette unter Betrachtung aller einwirkenden Faktoren zu bestimmen. Ein Vergleich mit der Deadline zeigt, ob die Latenz ausreichend ist, nahe dem Grenzwert oder gar darüber liegt. Ist letzteres der Fall, müssen wiederum entsprechende Maßnahmen ergriffen werden, um die Latenz zu senken.

Neben den bereits genannten Möglichkeiten (Zykluszeiten, Laufzeiten, Prioritäten) stehen zur Reduzierung der Wirkkettenlatenz noch weitere Mechanismen zur Verfügung. Der Parameter PositionInTask bestimmt die Reihenfolge der Runnable-Ausführungen innerhalb eines Tasks. Dieser Parameter kann für Wirkketten besonders wichtig werden, wenn zwei oder mehr Runnables der Wirkkette in demselben Task platziert sind. Schreibt ein Runnable, welches z.B. als letztes in dem Task ausgeführt wird, Daten, die wiederum vom ersten Runnable im Task gelesen werden, so wird dadurch ein ganzer Task-Zyklus verschwendet. Besser wäre es, die Runnables in umgekehrter Reihenfolge im Task anzuordnen. Die Wirkkettenanalyse zeigt solche Effekte unmittelbar auf und ermöglicht so eine manuelle oder automatische Optimierung.

Dasselbe gilt für die Task-Offsets. Mit deren Hilfe können die Aktivierungszeitpunkte von Tasks zeitlich gegeneinander verschoben werden. Dies

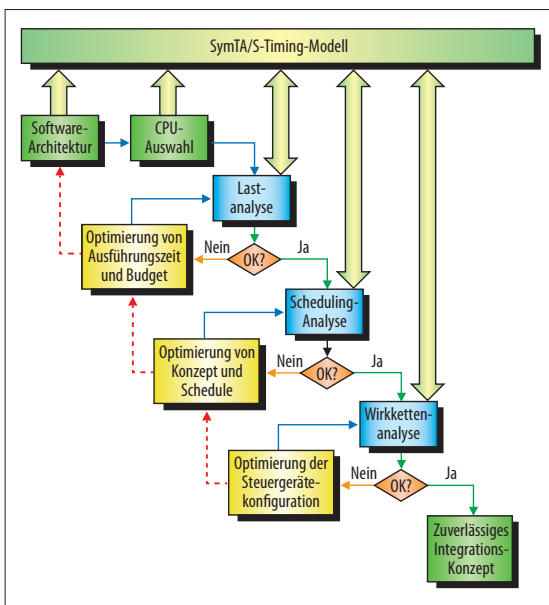


Bild 3. Ablauf der Methodik.

kann sowohl Lastspitzen als auch die Wirkkettenlatenzen weiter reduzieren. Für Multi-Core-Systeme können ebenfalls die einzelnen Cores synchron (einer Uhr folgend) oder individuell in den Schedule eingebunden werden. Beide Möglichkeiten sind sowohl in AUTOSAR als auch in SymTA/S verfügbar.

■ Berücksichtigung der Basis-Software

Nach der Timing-Analyse auf Applikations-Ebene kann in einem nächsten Schritt das bestehenden Timing-Modell um Basis-Software-Elemente ergänzt werden, da diese ebenfalls CPU-Zeit verbrauchen und den bisherigen Schedule beeinflussen können. Hierfür wurde die anfangs erwähnte Lastreserve vorgehalten. Der Fokus für diese Elemente liegt auf eigenständigen Tasks (z.B. TX-COM) oder lang laufenden Interrupts (z.B. RX-CAN oder Kurbelwellen-Interrupts), da diese den größten Einfluss haben. Auch Reserven für einen Task-Wechsel werden betrachtet.

Nachdem alle Elemente eingebracht sind, gibt die Scheduling-Analyse erneut Aufschluss über Last, Task-Schedule sowie die Wirkkettenlatenzen. Nach eventuell noch notwendigen Korrekturen ist die Konfiguration (Software-Architektur, RTE-/OS-Konfiguration, Basis-Software) fertiggestellt und kann mit einem minimierten Projektrisiko in die Implementierung übernommen werden.

■ Durchgängigkeit und Methodik

Bild 3 zeigt die zusammengefasste Methodik. Sie ist sowohl für neue als auch existierende Systeme einsetzbar. Die Anwendung der Methodik hält zu jedem Zeitpunkt in der Steuergeräteentwicklung ein Timing-Modell bereit. So kann über Integrationsstufen oder Produktaktualisierungen hinweg die zeitliche Auswirkung von Softwar-Änderungen immer zuerst virtuell integriert, überprüft und gegebenenfalls optimiert werden. Die vorgestellte Methodik orientiert sich vollständig am AUTOSAR-Standard, und das Werkzeug SymTA/S bietet eine automatisierte Unterstützung aller genannten Phasen.

Als Ergebnis entsteht eine optimierte Steuergerätekonfiguration nebst einer fundierten, quantitativen Bewertung der Realisierbarkeit des gesamten Projektes. Diese Bewertung minimiert das Integrationsrisiko und kann zudem zur Kommunikation mit und Argumentation gegenüber dem Projekt-Management, den Software-Zulieferern und den OEM-Kunden genutzt werden. Dabei bilden die aufgestellten Timing-Modelle ideale Entwicklungsartefakte zum Austausch zwischen verschiedenen Entwicklungsgruppen und entlang der Zulieferkette. *sj*



Dipl.-Ing. Christoph Ficek

hat Informations-Systemtechnik an der Technischen Universität Braunschweig studiert. Er ist bei Symtavision als technischer Projekt-Manager für die Anwendung und Konzeptionierung von Analyselösungen für Multi-Core- und AUTOSAR-ECU-Systeme in Kundenprojekten verantwortlich.
ficek@symtavision.com



Dr. Kai Richter

hat Elektrotechnik an der Technischen Universität Braunschweig studiert und 2004 promoviert. Er ist ein anerkannter Experte für Timing-Analyse im Bereich eingebetteter Echtzeit-Systeme und hat in diesem Umfeld zahlreiche Beiträge verfasst bzw. Vorträge gehalten. Seit 2005 ist er in der von ihm mitgegründeten Symtavision GmbH als Geschäftsführer für Technologie und Forschung verantwortlich.
richter@symtavision.com



Mit Sicherheit auf der Überholspur

Sichere Hard- und Softwarelösungen – vom Prototyp bis zur Serie

- AUTOSAR – Sicherheitssoftware nach ASIL-D
- Modulare Steuergeräte nach ISO 26262
- Sichere Vernetzung verteilter Elektroniksysteme
- Zuverlässige Testprodukte
- Safety Consulting und onsite Support

TTTech Automotive GmbH
www.tttech-automotive.com
 Tel.: +43 1 585 65 38-5000