

# Echtzeit-Methodik für AUTOSAR-Serienentwicklungen

Im ersten Teil dieses Beitrags wurde eine Echtzeit-Methodik aus Prozesssicht beschrieben. Es wurde dargestellt, welche elementaren Timing-Analyse- und Optimierungsschritte in welchen Entwicklungsphasen sinnvoll sind. Diese Methodik wird nun anhand des bereits eingeführten Beispiels aus Teil 1 im Detail angewendet und insbesondere die verfügbaren Daten sowie die Ergebnisse unter die Lupe genommen.

Die prozentuale CPU-Auslastung jeder Funktionskomponente bzw. jedes Runnables entspricht dem Quotienten aus Laufzeit und Zykluszeit. Die Zykluszeit ist aus der Funktionsmodellierung bekannt, z. B. als „sample time“ in Matlab/Simulink- oder ASCET-Funktionsmodellen. Im Fall von nicht-periodischen Funktionen lassen sich geeignete Aktivierungshäufigkeiten aus der funktionalen Simulation ableiten.

Die Laufzeit ist abhängig von der Software-Implementierung und der Ziel-Hardware. Ist eine Komponente bereits implementiert, kann man die Laufzeiten auf dem Target oder einem Evaluationsboard mit entsprechenden Debug- und Tracing-Werkzeugen (z. B. Gliwa T1) ausmessen. Ist die Software implementiert, aber die Hardware nicht verfügbar, liefern Prozessorsimulatoren (z. B. von der Firma Vast) oder die statische Laufzeitanalyse (z. B. von der Firma AbsInt) die genauesten Ergebnisse. In allen anderen Fällen führen Schätzung und Budgetierung am schnellsten zum Ziel. In der Regel ist dies der pragmatische Ansatz für die

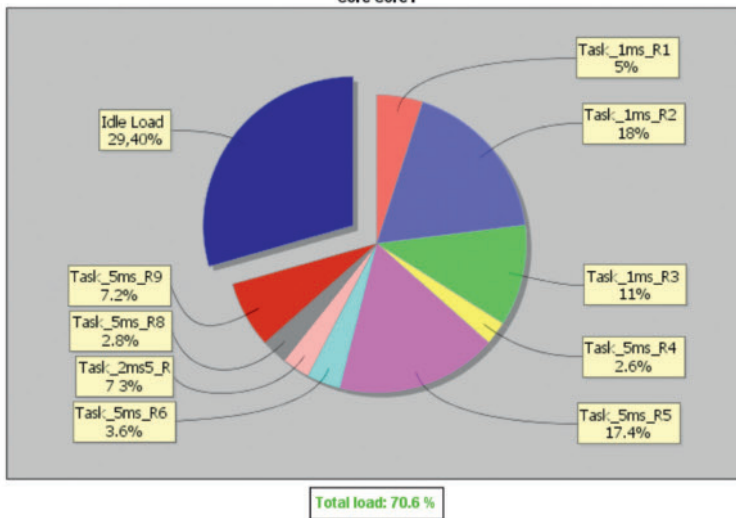
System-Design-Phase. Die CPU-Auslastung einer Software-Komponente mit mehreren Runnables ergibt sich dann aus der Summe der Werte für die einzelnen Runnables. Für das im ersten Teil des Beitrags gezeigte Beispiel ergibt sich eine CPU-Auslastung je Software-Komponente von: SW-C1 - 54%, SW-C2 - 6,6% und SW-C3 - 10%. Die Gesamtlast von etwas über 71% erscheint zunächst nicht kritisch. Die Lastdetails sind in **Bild 1** tabellarisch zusammengestellt, daneben die grafische Darstellung in SymTA/S.

## Virtuelle Integration

Als nächstes erstellt man einen ersten Task-Schedule. Dabei folgen wir in diesem Beispiel im ersten Schritt einem einfachen Schedule-Konzept. Jeweils Runnables mit derselben Zykluszeit werden in einer Task zusammen gefasst. Für die Task-Priorisierung folgt man der Regel: Je kürzer die Zykluszeit, desto höher die Task-Priorität. Daraus ergibt sich der im **Bild 2** dargestellte Schedule (ts: sampling time,

Name	SW-C	Zyklus	Laufzeit	Runnable	Last-Analyse			
					SW-C i	SW-C1	SW-C1+2	SW-C1-3
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%

**Load Overview**  
Core Core1



**Bild 1: Lastanalyse tabellarisch und grafisch.**

© automotive

Berücksichtigung der Basis-Software (z. B. COM) und zukünftiger Erweiterungen ausreichend ist, ist noch offen.

**Wirkketten-Analyse**

Die maximal erlaubten Latenzen sind aus der Funktionsentwicklung (Regelungstechnik) vorgegeben und entsprechen den Zykluszeiten der langsamsten Elemente, d. h., 1 ms für die Wirkkette WK1, 2,5 ms für WK2, und 5 ms für WK3. Nun ist zu überprüfen, ob diese Constraints eingehalten werden. Dazu werden die Wirkketten mit SymTA/S durch Verbinden der bereits modellierten Runnable- und modelliert. WK3 führt durch alle drei Zykluszeiten und Tasks: vom 1-ms-Task über den 5-ms-Task zum 2,5-ms-Task.

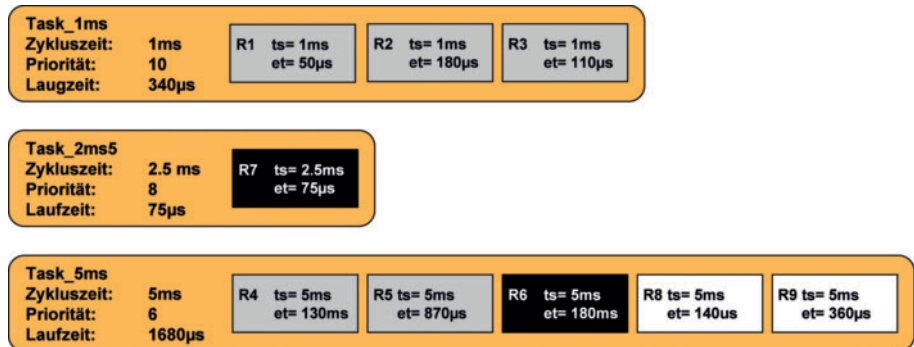
**Bild 5a** zeigt grafisch, wie sich die Gesamlatenz von 5,4 ms zusammensetzt: zur reinen Netto-Laufzeit der beteiligten Tasks wird der 5-ms-Task noch zweimal vom 1-ms-Task unterbrochen. Er beendet seine Ausführung bei den bereits erwähnten 2,85 ms und verpasst somit gerade den 2,5-ms-Task. Daraus

et: execution time). Aus diesen (und zunächst keinen weiteren) Informationen lässt sich nun mit dem Werkzeug SymTA/S (**Bild 3**) ein Scheduling-Modell erstellen und eine Schedulinganalyse durchführen. Als Ergebnis liefert SymTA/S einen Einblick in die zu erwartenden zeitlichen Abläufe im Zielsystem und bestimmt die so genannten Antwortzeiten der Tasks inklusive der Integrationseffekte wie Verdrängungen und Unterbrechungen. Dieser Schritt entspricht einer virtuellen, modellbasierten Integration bezüglich des Echtzeitverhaltens.

**Bild 4** zeigt ein von SymTA/S generiertes Gantt Chart der drei Tasks inklusive der Unterbrechungen. Man erkennt, dass die Antwortzeiten aller Tasks unterhalb der Zykluszeiten liegen, d. h., man verliert keine Task-Aktivierungen. Allerdings „verbraucht“ die Task im 5-ms-Zyklus bereits einen signifikanten Teil ihrer Zykluszeit; die Antwortzeit beträgt 2,85 ms, das entspricht 57% der gesamten Zykluszeit. Ob das unter

resultiert die lange Wartezeit zwischen dem 5-ms-Task und dem 2,5-ms-Task. Diese frühen Abschätzungen der Wirkkettenlatenz basieren ausschließlich auf den genannten Parametern (Laufzeit und Zykluszeit der beteiligten Runnables bzw. Funktionsblöcke) und berücksichtigen dabei – dank der virtuellen Integration – bereits die essenziellen Einflüsse auf das Zeitverhalten.

Es ist somit festzustellen, dass die Latenzschätzung der Wirkkette WK3 (5,4 ms) die Anforderung (5 ms) leicht ver-



**Bild 2: Task-Layout nach projektspezifischem Schedule-Konzept.**

© automotive

letzt, d. h., um knapp 10% überschreitet. Derartige Überschreitungen in der Frühphase zu erkennen, ermöglicht eine einfache Lösung, denn in der ECU-Konfiguration sind noch Optimierungsmöglichkeiten.

### ECU-Konfiguration

Hier wird nun für den bislang nur grob vorkonfigurierten virtuellen Task-Schedule die Details der RTE- und OS-Konfiguration festgelegt, und zwar so, dass insbesondere die kritischen Wirkketten optimiert werden, in unserem Fall WK3. Zunächst legt man den Parameter „Position-in-Task“ fest. Die Reihenfolge im 5-ms-Task im

ursprünglichen virtuellen Schedule lautet: R4, R5, R6, R8, R9. Für eine gute Effektivität der Wirkkette WK3 ist es jedoch sinnvoll, die Runnables im 5-ms-Task so anzuordnen, wie sie in der Wirkkette durchlaufen werden, also R8 vor R4 und R5. Die **Bilder 5b** (ursprüngliche Reihenfolge) und **5c** (optimierte Reihenfolge) zeigen, wie groß der Unterschied beider Konfigurationen ist. Zu sehen ist auch, dass die Latenz durch die Optimierung auf unter 3 ms gesunken ist.

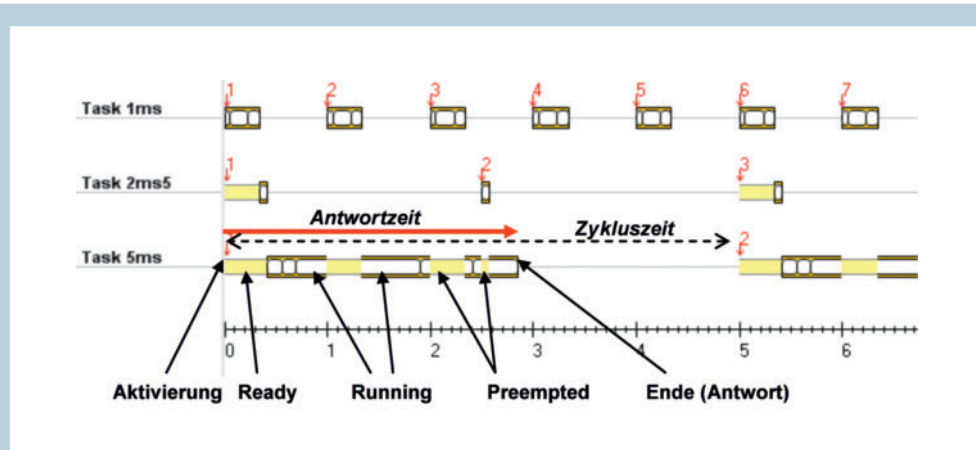
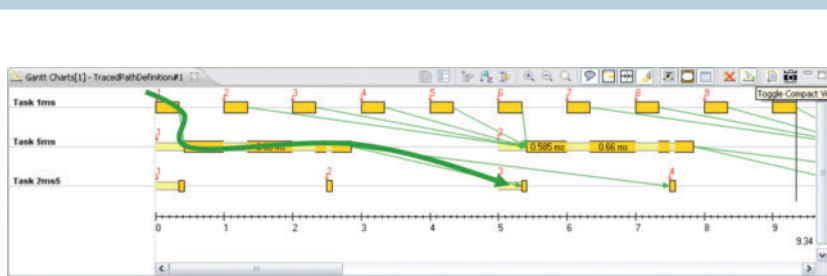


Bild 4: Timing-Anforderungen und Scheduling-Einflüsse.

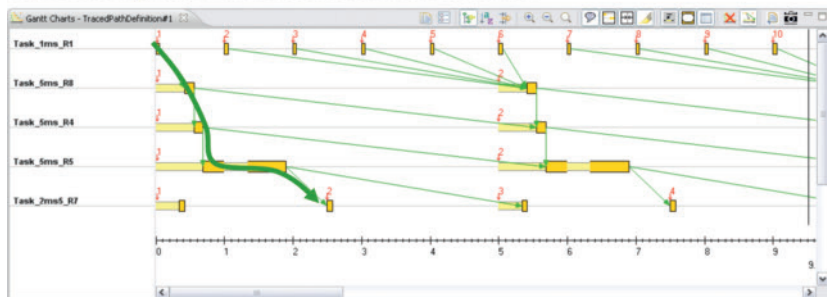
© automotive



a) Frühphasen-Abschätzung der Wirkkettenlatenz, basiert auf Funktionsmodell und Laufzeiten



b) RTE-Konfiguration mit schlechter Wirkketteneffektivität



c) RTE-Konfiguration mit guter Wirkketteneffektivität, optimierte Position-in-Task

Bild 5: SymTA/S-Darstellung der Wirkkettenlatenzen bei unterschiedlichen Scheduling-Konfigurationen.

© automotive

### Basissoftware

Schließlich wird überprüft, ob auch für die Basis-Software noch ausreichend Ressourcen zur Verfügung stehen. Man beschränkt sich dabei auf die größten „Zeitverbraucher“. Diese sind die CAN-COM Treiberschicht (zyklische TX-Task, dynamische RX-ISR) sowie der OS-Overhead zum Starten und Beenden von Tasks. Die weiteren Interrupts fasst man zusammen und wendet dabei wieder das Prinzip der virtuellen, modellbasierten Integration an, d. h., man modelliert die „Zeitverbraucher“ der Basissoftware genauso, wie dies zuvor mit den Funktionsbausteinen gemacht wurde: als Tasks und Runnables.

Neben den typischerweise höheren Prioritäten der COM-ISRs besteht der größte Unterschied zwischen ISRs und Tasks darin, dass die meisten ISRs azyklisch von externen Quellen aktiviert werden, z. B. durch den Empfang einer CAN-Botschaft. Das bedeutet,

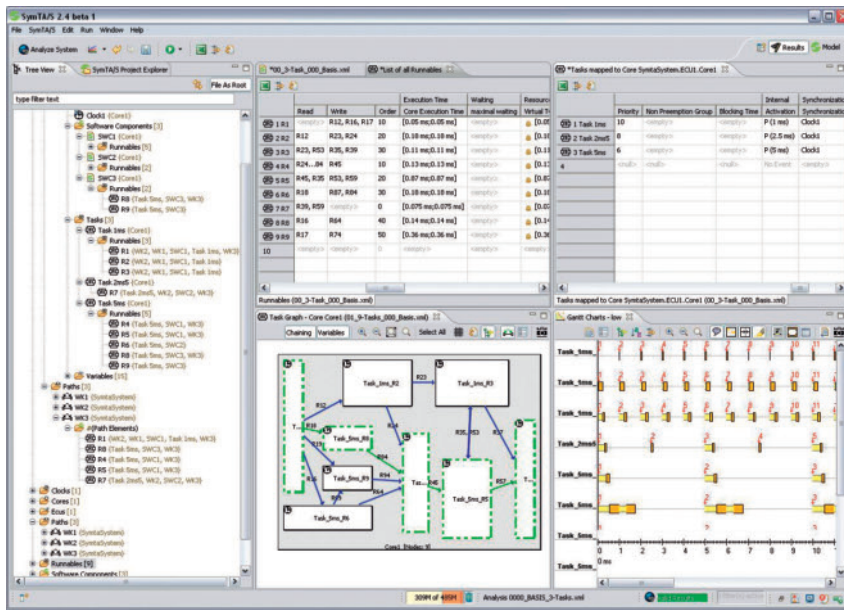


Bild 3: SymTA/S System-Timing-Modell.

© automotive

dass man an das für die virtuelle Integration notwendige Aktivierungsmodell anders herangehen muss. Für die CAN-RX-ISR ist im Projekt beispielsweise bekannt, dass nie mehr als sieben Frames unmittelbar hintereinander für dieselbe ECU adressiert sind, diese Botschaften lösen also schlimmstenfalls einen ISR-Burst mit einem Abstand von  $\sim 0,5 \dots 1$  ms aus (1-Mbit-CAN), danach kann mit einer mittleren Empfangsrate von einer Botschaft alle 2 ms gerechnet werden. SymTA/S unterstützt derartige nicht-periodische Aktivierungsmuster mit Hilfe von Ereignismodellen. So lassen sich auch das dynamische Verhalten des CAN-

Jetzt kostenlosen Eintrittsgutschein sichern:  
[www.sensor-test.com/gutschein](http://www.sensor-test.com/gutschein)

Willkommen zum

# Innovationsdialog!



## SENSOR+TEST

DIE MESSTECHNIK - MESSE

Nürnberg, 7. – 9. Juni 2011

Effizient und persönlich - Wissenschaftlich fundiert - Vom Sensor bis zur Auswertung

Veranstalter: AMA Service GmbH - Postfach 2352 - 31515 Wunstorf, Germany - Tel. +49 5033 96390 - info@sensor-test.com

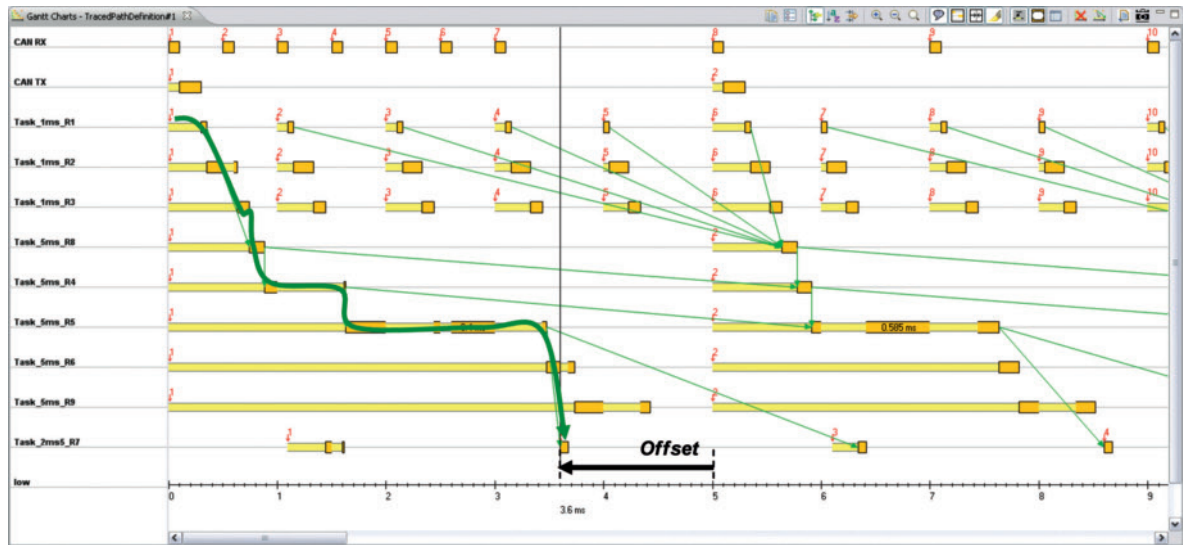


Bild 6: Optimierte Wirkketteneffektivität im Gesamt-ECU-Modell (mit BSW).

© automotive

RX-Interrupts erfassen. Auf dieselbe Art können weitere Interrupts virtuell integriert werden. Der COM-TX-Task verschickt alle 5 ms seine Botschaften und wird als periodischer Task modelliert. Dabei wird der OS-Overhead automatisch von SymTA/S berücksichtigt, lediglich die Laufzeiten der ActivateTask()- und TerminateTask()- Funktionen gibt man ein. Diese zusätzlichen Zeitinformatoren werden ebenfalls gemessen (z. B. ISR-Laufzeit) oder plausibel hergeleitet (z. B. CAN-RX-Aktivierungen) und das SymTA/S-Modell mit diesen zusätzlichen Informationen angereichert. SymTA/S bestimmt erneut Last, Schedule und Wirkketten-Latenzen des nun um BSW-Komponenten ergänzten virtuellen Integrationsmodells. Leider zeigt das Ergebnis, dass die zusätzlichen hochpriorigen Interrupts den 5-ms-Task derart verzögern, dass die zweite Ausführung des 2,5-ms-Task (aus Bild 5c) verpasst wird und dadurch die WK3-Latenz erneut über die kritische Marke von 5 ms steigt. Auch dieses Problem kann leicht behoben werden. Über den Parameter TaskOffset verlegt man den Start des 2,5-ms-Task um 1,1 ms nach hinten, sodass er den 5-ms-Task unmittelbar nach Beendigung des Runnable R5 bei 3,6 ms unterbricht und somit die Kette beschleunigt. Eine neuerliche SymTA/S-Analyse bestätigt die Wirksamkeit dieses Schrittes (Bild 6).

### Zusammenfassung

Es wurde eine vollständige virtuelle Integration einer kompletten ECU mit Echtzeit-kritischen Wirkketten durchgeführt, ausgehend vom einfachen Lastansatz über die Erstellung und Prüfung des RTOS-Schedule, bis hin zur Optimierung der Wirkkettenlatenzen. Dies geschah mittels virtueller ECU-Konfiguration und Timing-Analyse in SymTA/S. Die Modellierung beschränkte sich dabei auf die echtzeitrelevanten „Zeitverbraucher“ in der Applikations-

und Basissoftware. Der Schedule ist abgesichert, Reserven sind vorgesehen, und somit kann man nun mit minimalem Projektrisiko die Implementierung fortsetzen. In gleicher Weise lassen sich nun weitere Software-Komponenten und Wirkketten virtuell integrieren, um beispielsweise die nächste Integrationsstufe oder ein Produkt-Update zu planen. Der Einsatz von SymTA/S als Timing-Design-Werkzeug hilft dabei, stets einen Überblick über das zu erwartende Echtzeitverhalten zu haben, und die grundlegenden Integrationsentscheidungen so früh wie möglich auf Basis der realistisch verfügbaren Daten zu optimieren und virtuell abzusichern. (oe)



Dr. Kai Richter ist anerkannter Experte im Bereich der Zeit- und Leistungsanalyse verteilter, eingebetteter Systeme. Seit 2005 ist er in der von ihm mitgegründeten Symtvision GmbH als Geschäftsführer für Technologie und Forschung verantwortlich.



Dr. Marek Jersak arbeitete als Ingenieur und Projektleiter für Conexant Systems im Bereich DSP Compiler-Design und Optimierung. Seit 2005 ist er Geschäftsführer (CEO) der Symtvision GmbH, die er mitgründete. Seine Aufgaben umfassen Unternehmensstrategie und Business Development.