Effizienter Steuergerätetest mit Real-Time Plugins

Dipl.-Ing. Thomas Schmerler
Dr. Ing. Ulrich Lefarth

ETAS GmbH
Borsigstraße 14, 70469 Stuttgart
Tel.: +49 (711) 89661-103, Fax: +49 (711) 89661-330
E-Mail: thomas.schmerler@etas.com, ulrich.lefarth@etas.com

Abstract: Bei vielen HiL-Tests spielt die Berücksichtigung harter Echtzeitbedingungen eine wichtige Rolle. Im praktischen Einsatz läuft die Software zur Testautomatisierung jedoch auf einem Host-PC – und damit nicht synchron zum Simulationsmodell auf dem Echtzeitrechner. Dies führt dazu, dass die Echtzeittests oft direkt in das Simulationsmodell eingefügt werden. Der Echtzeit-Testfall kann dann per Schalter aktiviert und so synchron zum Echtzeitmodell abgearbeitet werden. Dies hat jedoch zur Folge, dass das eigentliche Simulationsmodell und der Test nicht mehr klar getrennt sind – was dazu führen kann, dass die Weiterentwicklung und die Ableitung von Testvarianten sehr kompliziert werden. Zudem wird die Pflege des Simulationsmodells erschwert, da neben dem Basismodell noch viele Varianten mitmodelliert werden müssen, um diese später zur Laufzeit zu aktivieren. In Summe entstehen so gigantische monolithische Gesamtsimulationsmodelle, die das Basismodell, die Varianten und die Echtzeittests enthalten.

Eine Lösung bietet der modulbasierte Ansatz. Der Simulationscode besteht dabei aus einzelnen Modulen, die zu einem Testfall verbunden und zur Laufzeit dynamisch erweitert werden können. Der große Vorteil liegt darin, dass nun die Funktionen des Simulationsmodells und der Echtzeittest separat erstellt und versioniert werden können. Dies führt zu einer besseren Übersichtlichkeit und einem „sauberen“ Echtzeitmodell, dass effizienter gepflegt und umfassender eingesetzt werden kann. Durch standardisierte C-Schnittstellen ist auch der Einsatz von Modulen möglich, die in verschiedenen domänenspezifischen Werkzeugen erstellt wurden.

1 Einführung

1.1 Die Herausforderung

Dass die Komplexität der Simulationsmodelle in vielen Bereichen stetig zunimmt, ist hinlänglich bekannt. Dabei sind nicht nur die Modelle selbst betroffen. Auch die Anzahl der Testfälle steigt unaufhaltsam. Erschwerend kommt hinzu, dass in der Praxis die Echtzeittests oft direkt in das Simulationsmodell eingefügt werden, um synchron zum Simulationsmodell auf dem Echtzeitrechner zu laufen. Der Echtzeit-Testfall kann dann bei Bedarf mit einem Softwareschalter aktiviert werden. Dies hat jedoch zur Folge, dass der Test und das Modell nicht mehr klar getrennt sind, was die Weiterentwicklung und die Ableitung von Testvarianten sehr erschweren kann.

Doch damit nicht genug: In immer mehr Fällen liegt der Simulationscode in unterschiedlichen Sprachen vor (zum Beispiel ASCET-, Simulink®- oder C-Code) welche zum Teil sehr schwer zu verbinden sind. Auch die Pflege des Simulationsmodells wird immer komplexer, da neben dem eigentlichen Simulationsmodell noch viele Varianten, wie zum Beispiel Sensorvarianten, CAN-Kommunikationsmatrizen oder Steuergerätevarianten, mitmodelliert werden müssen, um diese später zur Laufzeit zu aktivieren. In Summe entstehen so gigantische monolithische Gesamtsimulationsmodelle, die aus dem Basismodell, den Varianten und den Echtzeittests bestehen. Besonders schwierig wird es, wenn das Basismodell zugekauft oder von einem anderen Bereich im Unternehmen zugeliefert wird. In diesem Fall ist der Eingriff in das Originalmodell meist problematisch, da das Modell für den Anwender oft als „Black-Box“ vorliegt.

1.2 Das LABCAR-System

Um die wesentlichen Begriffe einzuführen, soll zu Beginn kurz auf das Hardware-in-the-Loop (HiL)-System LABCAR von ETAS eingegangen werden. Eine wesentliche Stärke des LABCAR ist seine offene und modulare Architektur. Dadurch kann das Testsystem sehr einfach an zukünftige Anforderungen angepasst und funktional erweitert werden. Die klare Trennung zwischen der systemspezifischen Hardware und der Berechnungseinheit auf Standard-PC-Basis bietet die Möglichkeit, gezielt und ohne weitere Zusatzinvestitionen einzelne Hardware auszutauschen oder zu ergänzen.

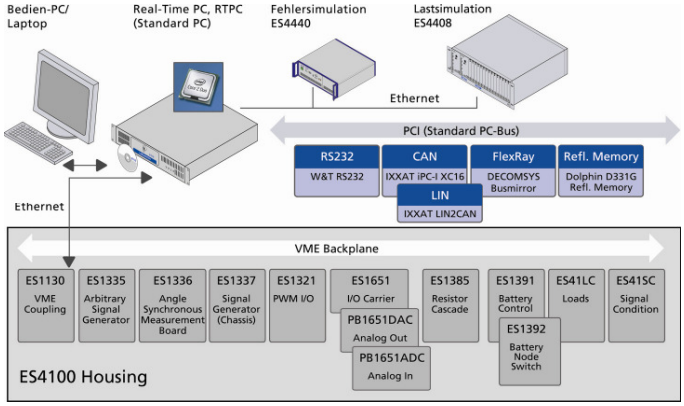

Bild 1: Komponenten des HiL-Systems LABCAR

Der Real-Time PC (RTPC) bildet das Herzstück des Testsystems. Mit Hilfe der LABCAR-RTPC-Software entsteht aus einem PC ein Hochleistungs-Simulationstarget, welches komplexeste DVE[1)]-Modelle mit Zykluszeiten um 20 µs in Echtzeit rechnet. So können selbst hochdynamische physikalische Regelprozesse simuliert werden. Die Kommunikation erfolgt über Standardschnittstellen, bevorzugt PCI, PCIe oder Ethernet.

Die LABCAR-OPERATOR Software auf dem Bedien-PC ist die elektronische Bedienschnittstelle für LABCAR – sowohl für das Erstellen der Experimente, als auch für deren Ausführung.

2 Modularer Simulationscode

Bei der neuesten LABCAR-Generation kann der Simulationscode, wie die Steuergerätesoftware, in einzelne LABCAR-Module aufgeteilt werden. Diese beschreiben jeweils einen Teil der Gesamtsimulation. Ein einzelnes LABCAR-Modul besteht aus der Schnittstelleninformation und der Funktionalität an sich, die als Standard C-Code beschrieben wird. Die Schnittstelleninformation beinhaltet die Definition der Ein- und Ausgänge des Moduls, Informationen darüber, auf welche Mess- und Verstellgrößen zugegriffen werden kann und in welche Prozesse das Modul unterteilt ist.

---

[1] DVE-Modelle: Driver-Vehicle-Environment-Modelle [SZ06]:
Diese mathematischen Modelle bilden Fahrer, Fahrzeug und Umwelt für die Simulation im Labor ab.

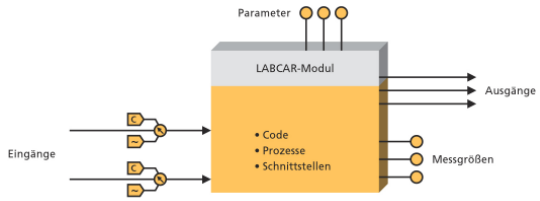

Bild 2: LABCAR-Modul

Bild 3 zeigt exemplarisch einen einfachen Aufbau aus mehreren Modulen. Jedes dieser Module kann auf Signalebene frei mit anderen Modulen verbunden werden. Auf diese Weise werden Signalpfade definiert und Regelkreise geschlossen. Jeder der Eingänge kann zur Laufzeit aufgebrochen werden, um einen Konstantwert, eine synthetische Signalform oder einen ausgezeichneten Signalverlauf (z. B. von einer Testfahrt) einzuspeisen.

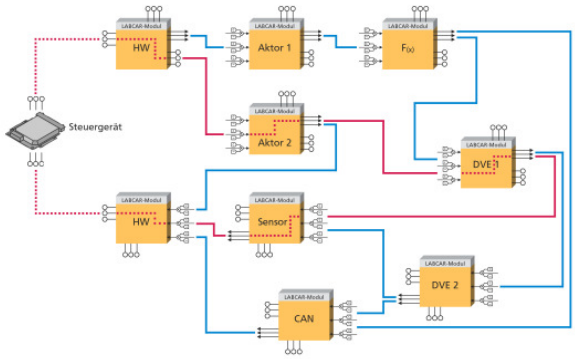

Bild 3: Modular aufgebauter Simulationscode

Aus jedem der Module wird bei der Codegenerierung zunächst C-Code generiert, sofern diese nicht bereits in C-Code vorliegen. Die einzelnen C-Code-Teile werden danach zum Gesamt-Simulationscode integriert, der später auf dem Simulationstarget (RTPC) ausgeführt wird.

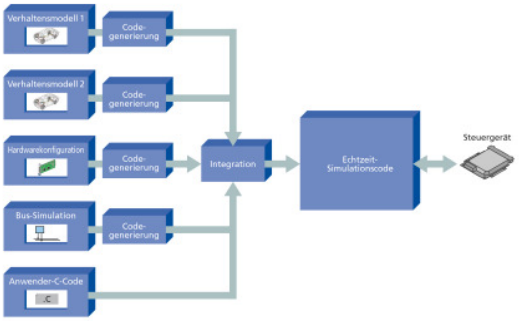

Bild 4: Integration des Simulationscodes

Dieser Ansatz bietet maximale Flexibilität, da jedes verhaltensbeschreibende Werkzeug eingesetzt werden kann, das in der Lage ist C-Code zu erzeugen. Auf diese Weise können die domänenspezifischen Vorteile diverser Werkzeuge gezielt zum Einsatz kommen, wie zum Beispiel die Verwendung verschiedener Modellierungssprachen, Restbussimulatoren oder C-Code Bibliotheken.

3 Multicore-Berechung des Simulationscodes

Die Echtzeitberechnung erfolgt auf dem Real-Time PC (RTPC). Dieses auf der INTEL Prozessorarchitektur basierende Simulationstarget bietet generische Unterstützung für Multicore-Prozessoren an. Dadurch ist es sehr leicht möglich, die einzelnen Module auf mehrere CPU Kerne verteilen. Diese Teile können dann parallel berechnet werden. So kann die Berechung schnell den wachsenden Anforderungen an Exaktheit, Umfang und Dynamik der Simulationsmodellteile angepasst werden.

Die RTPC Software erkennt automatisch, ob ein Multicore-Prozessor vorliegt und reserviert in diesem Fall einen der Kerne für administrative Aufgaben, Debugging, Logging und die Just-In-Time-Kompilierung (JIT). So bleiben beispielsweise bei einem modernen Quad-Core Prozessor drei Prozessorkerne, die ausschließlich die Echtzeitsimulation berechnen können. Diese Partitionierung erhöht die Qualität und Konstanz der Gesamtsimulation, da alle anderen Aufgaben getrennt laufen und nicht die Gesamtausführung beeinträchtigen können.

Neben der flexiblen, parallelen Berechnung der einzelnen Module und der Verwendung unterschiedlicher Expertentools zur Generierung der Module entstehen bei der hier gezeigten Vorgehensweise noch weitere Vorteile: Die verteilte Modellierung entspricht auch dem Vorgehen bei der Entstehung der Software. Dies erleichtert das Versions- und Variantenmanagement sowie die verteilte Entwicklung verschiedener Teile. Die gesamte Struktur wird übersichtlicher und nachträgliche Änderungen und Erweiterungen sind einfacher und damit sicherer zu handhaben. Es ist nun möglich, an günstigen Stellen zwischen den einzelnen Modulen manipulativ einzugreifen.

4 Plugin Hooks

ETAS hat für die Software des PC-basierten Echtzeitrechners (Real-Time PC, RTPC) ein neues echtzeitfähiges Plugin-Framework entwickelt, das es erlaubt, die Module der Gesamtsimulation zur Laufzeit punktuell zu erweitern. In das laufende Simulationsmodell wird dabei an frei definierbaren Stellen zusätzlicher Code zur Laufzeit integriert.

Das Simulationsmodell, welches die Grundregelschleife darstellt, erhält „Plugin Hooks“, die selbst keine Laufzeit benötigen. An diesen können später zur Laufzeit die in C-Code erstellten Plugins eingehängt und dadurch ausgeführt werden. Sie laufen zeitsynchron zum Modell und im Adressraum des Simulationscodes – d. h. der Zugriff auf alle Variablen, Parameter, Messgrößen, CAN-Botschaften, Hardware-I/O-Größen usw. ist möglich.Die einzelnen Prozess-Module werden als Tasks aneinander gereiht. Dabei kann jedem Task eine Zeit oder eine Bedingung mitgegeben werden, wann dieser ausgeführt werden soll. Innerhalb der Prozesskette, welche die Task-Reihenfolge festlegt, werden an geeigneten Stellen die Plugin Hooks eingefügt. Bild 5 zeigt exemplarisch eine Prozesskette in einem CPU-Kern eines RTPC.

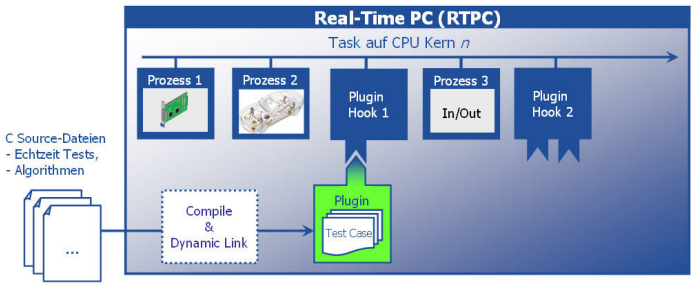

Bild 5: Einhängen eines Plugins in der Prozesskette

Die RTPC Software nimmt den Plugin-Code entgegen und kompiliert diesen „Just-In-Time“ (JIT-Kompilierung). Anschließend wird das neue Modul an den entsprechenden Plugin Hook in die Prozesskette eingehängt. Solange, bis das Modul wieder entfernt wird, wird es auf gleicher Ebene und im identischen zeitlichen Kontext wie das restliche Simulationsmodell ausgeführt. Der Vorteil dieses Vorgehens ist, das die Plugins beliebig und ohne Interpreter ein- und ausgehängt werden können. Ist kein Plugin eingehängt, läuft die Prozesskette einfach ohne Zeitverzug weiter.

Die Plugin Hooks können an jeder Position und beliebig oft innerhalb der Prozesskette in das Gesamtsimulationsmodell gesetzt werden. Die Definition der Position der Hooks erfolgt vor dem Kompilieren des Gesamtsimulationscodes. Dadurch ist die Reihenfolge der Plugins untereinander und der Plugins im Kontext der Gesamtprozesse eindeutig vorgegeben.

Der Plugin Code hat Vollzugriff auf die Datenelemente der restlichen Simulationsmodule und kann zum Beispiel Parameter setzen, Messwerte analysieren und manipulieren, CAN-Botschaften senden, über XCP direkt auf ein Steuergerät zugreifen und vieles mehr. Über diesen Mechanismus ist es zum Beispiel auch sehr leicht möglich, in die laufende Echtzeitsimulation hinein CAN-Kommunikationsmatrizen auszutauschen oder Varianten eines Sensormodells durchzutesten ohne den Betriebspunkt verlassen zu müssen. Ein weiteres Anwendungsgebiet ist auch das Einfügen von Testfällen in das „saubere“ Simulationsmodell.

Plugins können durchaus voneinander abhängen. Daher ist es erforderlich, sich über die Position der Plugins innerhalb der Prozesskette im Klaren zu sein. Bei Abhängigkeit der Plugins sollten diese verschiedenen Hooks zugewiesen werden.

5 Anwendungen der Plugins

In Abhängigkeit von der Position des Plugins innerhalb der Prozesskette werden verschiedene Anwendungen eingesetzt. Dabei können die Pluginn Hooks im Modell an vielen Stellen vorgehalten werden (Bild 6). Das Arbeiten mit Plugins bietet einige große Vorteile. Zum einen können sehr schnell gezielt Erweiterungen durchgeführt werden. Zudem erfordert das Einfügen dieses Pluigins keine Neukompilierung des gesamten Modells, sondern nur des geänderten Plugin, was eine große Zeitersparnis darstellen kann. Zudem ist es möglich, die Plugins zusammen mit der Software zu versionieren. Im Folgenden einige Anwendungsmöglichkeiten:

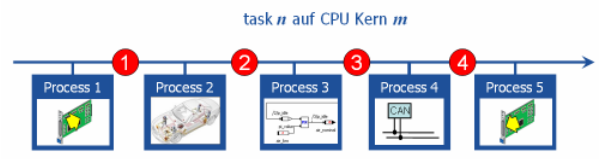

Bild 6: Prozesskette mit möglichen Positionen für Plugin Hooks (1- 4)

Beeinflussung der Eingangssignale

Ein Beispiel für die Beeinflussung der Eingangssignale: Bei der Versuchsfahrt tritt immer wieder folgendes Verhalten auf: Wird auf einer Schotterpiste gebremst, erkennt das Steuergerät den Fehler „Bremsleuchte defekt“ und nimmt einen Eintrag in den Fehlerspeicher vor. Eine Fehlfunktion des Bremslichtes kann aber nicht erkannt werden. Das Problem wird untersucht und es stellt sich heraus, dass der Bremslichtschalter durch die Erschütterungen länger als erwartet prellt. Die Entprellzeit in einem ESP-Steuergerät wird verlängert. Ein Test am HiL-System soll die Funktion über alle Varianten des Fahrzeuges absichern. Hierzu kann ein Plugin an Position 1 eingefügt werden. Eingangsgrößen sind die Signale der Drehwinkelsensoren und der Status des Bremslichts. Ein längeres Prellen des Schalters wird ausgegeben, wenn die Ableitung der Drehrate einen gewissen Schwellenwert überschreitet.

Einwirken auf die Simulation

Nach der Berechnung eines Simulationszustandes des Umgebungsmodells können gezielt Modellmanipulationen vorgenommen oder komplexe Stimulationen eingefügt werden (Plugin Hook 2 oder 3). Es können an dieser Stelle auch mehrere Sensormodelle simuliert werden, die nacheinander geladen und ausgeführt werden – ohne dass sie im Basismodell berücksichtigt werden müssen. Auch kann ein alternatives Fahrer-Modell zur Laufzeit geladen werden.

Bus-Manipulationen

Restbussimulationen für CAN und FlexRay sind wichtig für den HiL-Betrieb. Bevor die berechneten Botschaften an die Hardware übermittelt werden, können an dieser Stelle noch Manipulationen stattfinden, Botschaften verfälscht oder neue Botschaften aufgenommen werden (Plugin Hook 4). Zum Beispiel kann das Signal des Gierratesensors in Abhängigkeit des Weges manipuliert werden um das Ruckeln bei einem großen Stein zu simulieren.

Signalmanipulation

Bevor die Signale über die Hardware in die Closed-Loop Simulation zurückgespeist werden, besteht die Möglichkeit zur Fehlersimulation, indem beispielsweise Signale kurzzeitig zufällig auf 0 gezogen werden, um Aussetzer zu simulieren.

Flexible und übersichtliche Architektur

Um maximale Flexibilität zu erreichen, können die RT-Plugins direkt in der Architektur der Gesamtsimulation berücksichtigt werden. Das Basis-Simulationsmodell besteht dann nur noch aus dem „sauber modellierten Gut-Fall“. Jedwede Fehlermodelle, Tests oder Varianten werden an gezielt dafür vorgesehenen Positionen bei Bedarf eingehängt. Dies bringt den entscheidenden Vorteil, dass die Gesamtsimulation sauber partitioniert ist und die einzelnen Teile übersichtlich gehalten und gepflegt werden können. Da der Trend zu immer mehr Modellierungstiefe geht und die Anforderungen, zum Beispiel im Echtzeittest, stetig steigen, erleichtert diese Vorgehensweise das zukunftsorientierte Aufsetzen von HiL-Tests sehr.

6 Fazit

Die zunehmende Komplexität der Simulationsmodelle und die stetig steigende Zahl der Testfälle lassen sich durch einen modularen Aufbau erheblich besser beherrschen. Diese Trennung von Test und Modell ermöglicht die Erstellung eines sauberen, performanten und ggf. variantenunabhängigen Modells, ohne dass bedingt ausführbarer Code im Modell vorgehalten werden muss. Dabei können verschiedene Werkzeuge zum Einsatz kommen (zum Beispiel ASCET-, Simulink®- oder C-Code). Der Einsatz von C-Code Plugins erlaubt es dabei Funktionalität zur Laufzeit beliebig und ohne Interpreter ein- und auszuhängen. Der Simulationscode bleibt dabei unverändert. Dieses Konzept bietet viele neue Möglichkeiten, um komplexe, variantenreiche und in verschiedenen Werkzeugen erstellte HiL-Simulationen auch in Zukunft weiterzuentwickeln und sicher zu beherrschen.

Literaturverzeichnis

[SZ06] SCHÄUFELE, J.; ZURAWKA, T.: Automotive Software Engineering, 3. Auflage – Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen. Vieweg Verlag, Wiesbaden, 2006

[ETA09-ol] ETAS GROUP INTERNET www.etas.com/LABCAR: Steuergeräte-Test (HiL), 2009

Autoren

Dipl.-Ing. Thomas Schmerler, studierte Elektrotechnik an der Universität Karlsruhe (TH). Im Jahr 1999 trat er in die ETAS GmbH ein und ist seitdem in der Softwareentwicklung für die Produkte LABCAR-OPERATOR, LABCAR-AUTOMATION und LABCAR-PINCONTROL tätig. Sein Schwerpunkt sind Echtzeitsimulationen und Echtzeittests. 2005 übernahm Thomas Schmerler die Verantwortung für die Entwicklung der LABCAR Software und den Echtzeit-Simulationsrechner RTPC.

Dr. Ing. Ulrich Lefarth, studierte Maschinenbau an der Universität Paderborn und promovierte dort im Bereich Simulation mechatronischer Systeme. Im Jahre 1995 trat er in die ETAS GmbH ein. Nach der Leitung des Produktbereichs Embedded Systems war er als Geschäftsführer für die französische ETAS Niederlassung verantwortlich. Seit 2006 leitet die weltweite Entwicklung der ETAS Softwareprodukte.