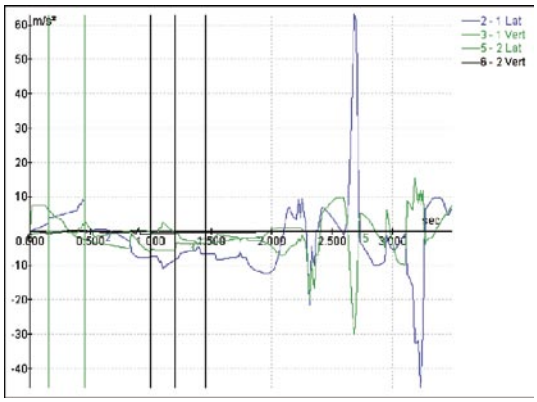
Überschläge überleben

Überschläge spielen für die Fahrzeugsicherheit eine große Rolle, denn oft haben sie einen verhängnisvollen Ausgang. Jährlich sind für etwa ein Viertel der tödlich endenden Verkehrsunfälle in den USA Fahrzeugüberschläge verantwortlich. Der von Delphi patentierte Algorithmus WinGAMR erkennt Überschläge und löst Schutzmechanismen aus. Für die Implementierung von WinGAMR nutzte Delphi den modellbasierten Entwurf und setzte TargetLink für die automatische Seriencode-Generierung ein. Die Code-Profiling-Techniken von TargetLink halfen bei der Code-Analyse und verbesserten nachhaltig die Effizienz des Seriencodes.

- **Delphis Entwicklungen für Insassenschutzsysteme**
- **Algorithmus für Überrollerkennung implementiert und nun in Serie**
- **Entscheidende Code-Optimierungen durch Einsatz von TargetLink erreicht**



▲ *Simulierte Sensorsignale zeigen Längs- und Querbeschleunigung, die der Algorithmus analysieren muss.*

Hintergrund

Ein Überschlag ist ein sehr komplexer Vorgang, der die unterschiedlichsten Ursachen haben kann, zum Beispiel plötzliches Reißen am Lenkrad durch einen übermüdeten, abgelenkten Fahrer oder wenn ein Fahrzeug ins Schleudern gerät und dabei ein flaches Hindernis passiert. Auch Fahrzeugkollisionen können zu Überschlägen führen.

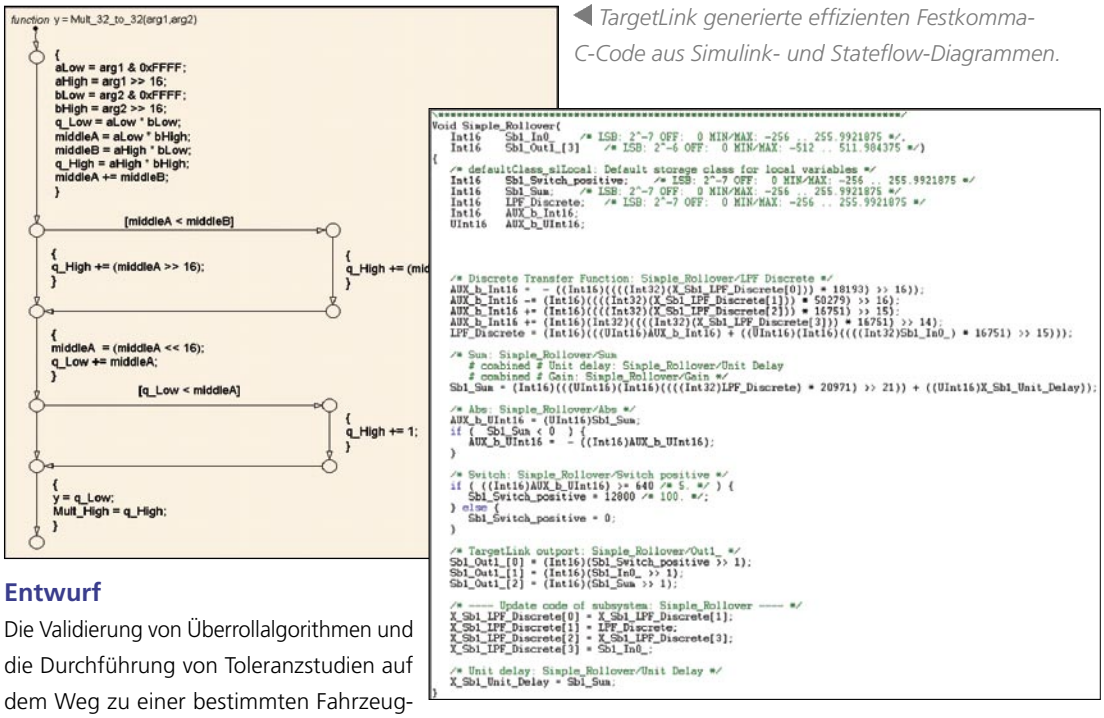
Insassenschutz

Nach Front-Airbags für Fahrer und Beifahrer und in vielen Fahrzeugen mittlerweile auch Seiten-Airbags bildet der Überschlagschutz das kommende große Marktpotenzial im Bereich der Insassenschutzsysteme. Auf dem Gebiet sind drei Möglichkeiten üblich: Verringern des Spiels beim Rückhaltegurt (aktive Gurtstraffung), um das Risiko des Herausschleuderns von Insassen zu reduzieren; Ausfahren von Überrollbügeln bei Cabriolets sowie Seiten-Airbags, um Kopfverletzungen und Herausschleudern vorzubeugen. Zudem sind auch Maßnahmen nach dem Überschlag erforderlich, zum Beispiel Unterbrechung der Kraftstoffzufuhr und Absetzen eines Notrufes.

Überrollerkennung

Den richtigen Zeitpunkt für das Auslösen solcher Sicherheitsmaßnahmen zu erkennen, ist die Aufgabe des Algorithmus für die Überrollerkennung. Daten von internen Sensoren wie Gyroskopen und Beschleunigungsmessern werden üblicherweise von einem Modul zur Überrollerkennung verarbeitet, um zu entscheiden, ob die Maßnahmen ausgelöst werden müssen oder nicht. Der Entwurf des Überrollalgorithmus muss einen weitläufigen, dynamischen Bereich an Ereignissen abdecken, vom schrittweisen Abkommen in Richtung Straßengraben bis hin zum unplanmäßigen Bordsteinkontakt mitten in der Stadt. In Anbetracht von Komplexität und Variantenreichtum der Überschlagszenarien besteht in der Unterscheidung zwischen auslösenden und nicht auslösenden Ereignissen eine enorme Herausforderung.

▲ *In solchen Situationen löst der WinGAMR Algorithmus zur Überrollerkennung die Sicherheitsmaßnahmen aus.*



◀ *TargetLink generierte effizienten Festkomma-C-Code aus Simulink- und Stateflow-Diagrammen.*

Entwurf

Die Validierung von Überrollalgorithmen und die Durchführung von Toleranzstudien auf dem Weg zu einer bestimmten Fahrzeugplattform werden durch mathematisch basierte Entwurfsmethoden stark vereinfacht. Sobald ein Algorithmus validiert ist, kann er für die Implementierung auf einem Festkomma-Mikroprozessor autocodiert werden. Um die Zuverlässigkeit des endgültigen Systems zu garantieren, muss sichergestellt werden können, dass sich C-Code und mathematisches Modell exakt gleich verhalten.

Projekt und Prozess

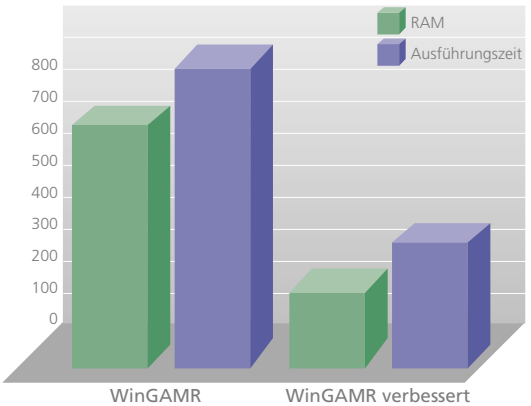
Der patentierte Algorithmus WinGAMR, entwickelt von Delphi Electronics & Safety, wurde vorab mit Simulink®- und Stateflow®-Modellen implementiert. Anschließend wurden mit TargetLink Wertebereich und Auflösung für jede Variable eingestellt (Skalierung). Die Simulationsresultate des Festkomma-Target-C-Codes wurden mit denen des ursprünglichen Modells verglichen, um etwaige Einbußen bezüglich Speicherbedarf und Ausführungszeit zu erkennen und das System zu optimieren. Durch den Vergleich von Target-C-Code (PIL-Simulation) mit den Originaldaten des Modells (MIL-Simulation) wurde schließlich die Leistung des finalen Produkts verifiziert.

Profiling und Optimierung

TargetLink wurde zusammen mit einer Zielprozessor-Evaluierungskarte eingesetzt, um eine Metrik für die Ausführungszeit zu erstellen. Eine suboptimale Implementierung des Algorithmus konnte damit innerhalb von Minuten identifiziert und die Ursache für den hohen RAM-Bedarf und die lange Ausführungszeit gefunden werden. Sowohl RAM als auch Ausführungszeit wurden durch geringe Änderungen an der Implementierung des Algorithmus um 75 % reduziert. Mit dem herkömmlichen Ansatz wären diese enormen Verbesserungsmöglichkeiten bestenfalls in Wochen oder gar Monaten erkannt worden. Die Funktionsentwickler lernten schnell aus den Erkenntnissen der Algorithmusimplementierung, was unmittelbar zur Leistungssteigerung führte. Neben der beschleunigten Markteinführung ließen sich so unnötige Aufwendungen in der Entwicklung vermeiden.

Implementierung

Der autogenerierte Code wurde dem für die Software-Integration zuständigen Ingenieur übergeben. Die Inte-



▲ *Die durch TargetLinks Analysewerkzeuge und eine Evaluierungskarte gewonnenen Einblicke in den Algorithmus führten zu deutlichen Verbesserungen beim RAM-Bedarf und der Ausführungszeit.*

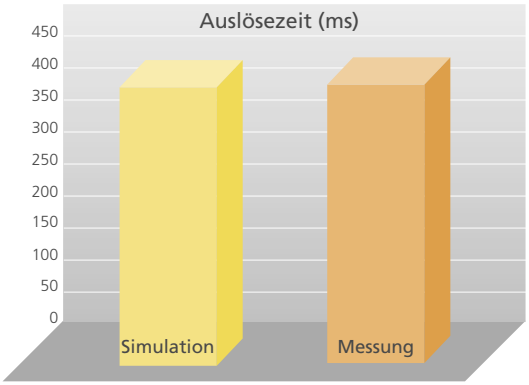
gration war in knapp 2 Tagen abgeschlossen. Stimmen dazu: „Der generierte Code war sehr verständlich. Jeder Kommentar und jeder Variablenname war eine echte Hilfe und meines Erachtens haben wir sehr viel Zeit eingespart." „Er ist eine gute Basis für die Entwicklung des Target-C-Codes. Der Großteil des Codes blieb beinahe unverändert." „beinahe" bezieht sich dabei auf einen digitalen Hochpassfilter, der mit 16-Bit-Variablen arbeitet. Der zuständige Integrationsingenieur erweiterte diese auf 32-Bit-Variablen, wodurch die Genauigkeit der finalen Ergebnisse verbessert werden konnte.

Handcodierung vs. Autocodierung

Bei dem autocodierten Algorithmus handelt es sich um eine Neuentwicklung, daher sind Vergleiche mit Handprogrammierung nicht vorhanden. Jedoch erforderte eine frühere handcodierte Version des Überrollalgorithmus längere Ausführungszeit, mehr RAM und mehr ROM als der autocodierte Algorithmus. Zudem verbesserte der autocodierte Algorithmus die Leistung in der Praxis. Zwar sind für diesen Vergleich weitere Faktoren notwendig, dennoch kann gesagt werden, dass der Autocode keine unerwünschten Metriken im neuerstellten Überrollalgorithmus verursachte.

Live-Tests

Um die Ergebnisse mit den Prognosen zu vergleichen, wurden Live-Tests mit dem Algorithmus durchgeführt. Insgesamt fanden drei Tests statt. Beide Tests mit einem Beinahe-Überschlag waren erfolgreich und es wurden keine Maßnahmen ausgelöst. Die gemessene Auslösezeit des Überrolltests entsprach exakt der Simulation (10 ms Abtastrate). Auch war ein professioneller Rennfahrer



▲ *Die für den autocodierten Algorithmus gemessene Auslösezeit entsprach haargenau der Simulation.*

selbst auf einer anspruchsvollen Teststrecke nicht in der Lage, unerwünschtes Verhalten zu provozieren. Aufgrund der erfolgreichen Ergebnisse wird dieser Algorithmus mittlerweile in Serienkomponenten eingesetzt.

Ausblick

Für die Weiterentwicklung von Überrollalgorithmen wird weiterhin MATLAB/Simulink/Stateflow eingesetzt. Zur automatischen Generierung von Seriencode kommt TargetLink zum Einsatz. Ausgehend von den Blöcken des WinGAMR Algorithmus wird die nächste Generation der Überrollalgorithmen deutlich schneller und effizienter entwickelt werden können.

Peter J. Schubert, PhD
Technical Fellow
Systems Methodology Advocate
Delphi Electronics & Safety
USA

Essenz

Erfolgsstory Autocode

- Code-Profiling führte zu Handcode-Effizienz; RAM/Ausführungszeit um 75 % reduziert
- In 1,5 Tagen integriert
- In Serie
- Aussagen der Software-Entwickler:
  „Der generierte Code war sehr verständlich. Jeder Kommentar und jeder Variablenname war eine echte Hilfe."
  „Meines Erachtens haben wir sehr viel Zeit gespart. Er ist eine gute Basis für die Entwicklung des Target-C-Codes. Der Großteil des Codes blieb unverändert."

Equipment und Methoden

- TargetLink, Target Optimization Module und Motorola HC12 Evaluation Board für Code-Profiling
  – Ausführungszeit
  – RAM (einschließlich Stack)
  – ROM
- Back-to-Back-Tests (MIL-, SIL- und PIL-Simulationen) zum frühest möglichen Zeitpunkt

Autocode reduziert Risiken

- Keine Transkriptionsfehler
- Keine Fehlinterpretationen der Spezifikation
- Entspricht der Leistung des Modells