Software-Verfahren für die funktionale Sicherheit

Software Coded Processing (SCP) ist ein Software-Verfahren, mit dem fehlende und existierende Sicherheitsmaßnahmen ersetzt werden können. SCP erkennt transiente, permanente und systematische Hardware-Ausführungsfehler mit hohen Fehlererkennungsraten bis ASIL D während der Laufzeit einer sicherheitskritischen Funktion. In diesem Artikel werden die Funktionsweise von SCP erläutert, experimentell gewonnene Ergebnisse präsentiert sowie eine Implementierungsform vorgeschlagen.

Sicherheitsrelevante Innovationen in der Automobiltechnik, z. B. der Einsatz von Domain-Controllern oder das automatisierte Fahren, fordern eine signifikant höhere Rechenleistung als auf heutigen Steuergeräten verfügbar, die für sicherheitskritische Anwendungen eingesetzt werden. Der Einsatz rechenstarker Controller, die nur eine begrenzte Sicherheitsintegrität bieten, stellt sowohl Automobilhersteller als auch -zulieferer vor eine anspruchsvolle Herausforderung.

Zukünftige Fahrzeuge werden als ein zusammenhängendes und verteiltes Netzwerk komplexer Software-Systeme gesehen (Bild 1). Zukunftsvisionen einer intelligenten Vernetzung von Fahrer, Fahrzeug und Umfeld erfordern neue Ansätze für die Plattformen der Steuereinheiten und deren Schnittstellen im Gesamtsystem. Nicht zuletzt aufgrund der beschränkten Erweiterungsmöglichkeit der E/E-Architekturen im Fahrzeug muss die steigende Software-Last in hochintegrierte Domain-Controller verlagert werden. Fahrerassistenzsysteme bis zum autonomen Fahren erfordern außerdem die Ausführung datenintensiver Funktionen und komplexer Algorithmen.

Moderne eingebettete Rechner müssen nicht nur hohe Anforderungen an Rechenleistung und Kosteneffizienz, sondern auch hohe Sicherheits- und Zuverlässigkeitsanforderungen erfüllen, denn die steigende Komplexität und Autonomie eingebetteter E/E-Systeme verschärft die Notwendigkeit wirksamer Konzepte und Mechanismen zur funktionalen Sicherheit.

Zuverlässige und kosteneffiziente Rechner

Verfügbare Controller auf dem Markt eingebetteter Systeme verfügen entweder über hohe Rechenleistung oder hohe Sicherheitsintegrität. Verschiedene Halbleiterhersteller bestätigen aktuell das Fehlen ausfallsicherer Produkte mit der Rechenleistung, die von künftigen Software-intensiven Anwendungen gefordert wird. »

Bilder: Silistra Systems ; © 123RF.com/Adrienn Orbanhegyi, Denis Tevekov

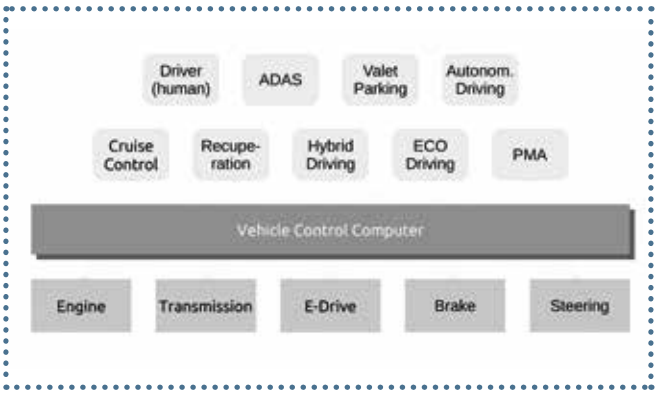

Bild 1: Architektur und Vernetzung zukünftiger Fahrzeugsysteme.

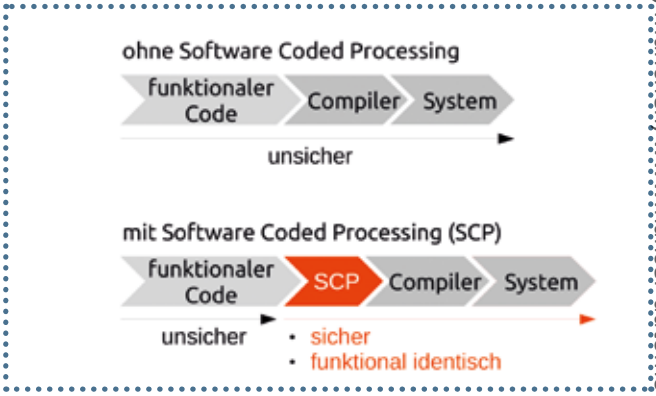

Bild 2: Code-Transformation mittels Software Coded Processing (SCP).

Standard-Prozessoren bieten zusätzlich zu ihrer hohen Rechenleistung deutlich bessere Kosten pro Leistung als Controller, die speziell für automobile Anwendungen entwickelt wurden. Die hohe Leistung und die Kosteneffizienz motivieren die Untersuchung solcher Prozessoren für den Einsatz im Automobilbereich. Eine signifikante Herausforderung besteht darin, dass solche Bauelemente beschränkte Möglichkeiten zur Selbstdiagnose und Fehlererkennung haben und dadurch nicht ohne weitere Maßnahmen für den Einsatz in sicherheitskritischen Systemen geeignet sind.

Die Ausführungsintegrität aktueller Automotive-Controller wird in der Regel durch Selbsttests und Hardware-Redundanz gewährleistet. Verwendet wird hier der Begriff „Ausführungsintegrität", um die Erkennung und Behandlung von systematischen, permanenten und transienten Hardware-Fehlern zu bezeichnen, die zur Verletzung eines Sicherheitsziels der ausgeführten Funktion führen können.

Eine für die Ausführungsintegrität etablierte Technik bei Mikroprozessoren sind Zweikern-Architekturen, die im Lock-Step-Modus betrieben werden. Von jedem Kern wird erwartet, dass mit den gleichen Eingangswerten die gleichen Ausgangswerte generiert werden. Lock-Step-Prozessoren sind allerdings anfällig für Nicht-Determinismus, der durch eine Anzahl von Mechanismen in aktuellen CPUs (z. B. Pipe-Lining) erhöht wird, sodass solche Prozessoren ohne aufwendige Anpassungen von den jüngsten Entwicklungen in der Prozessortechnologie entkoppelt werden. Zusätzlich sollen Standard-Prozessoren möglichst ohne Änderungen an der Hardware-Architektur eingesetzt werden, um deren wirtschaftlichen Vorteile zu nutzen.

Vorgeschlagen wird daher, Software-implementierte und Hardware-unabhängige Fehlererkennungsmechanismen für zukünftige sicherheitskritische Systeme einzusetzen. Im Gegensatz zu aufwendigen Software-Implementierten Ansätzen, z. B. diversitäre Software-Redundanz, wird hier ein alternativer Ansatz basierend auf Software Coded Processing eingeführt, der gleichzeitig niedrigeren Entwicklungsaufwand und hohe Diagnose für Ausführungsfehler bietet.

Software Coded Processing

Software Coded Processing (SCP) ist ein Software-Verfahren, das auf der arithmetischen Kodierung von Variablen, Konstanten und Operationen basiert und Ausführungsfehler aufdeckt. Im Ergebnis erhält man durch SCP eine Ende-zu-Ende-Absicherung eines gegebenen Programm-Codes, die – weil es sich um ein Software-Verfahren handelt – unabhängig von der verwendeten Hardware ist und einen während der Laufzeit kontinuierlichen Schutz bietet.

Die Ausführungsintegrität eines Programms wird durch Ausführungsfehler verletzt. Diese besonders schwierig zu greifenden Fehler kommen zustande, indem Fehler in der Hardware in die Programmausführung propagieren und dabei Fehler verursachen. Da durch das Programm die Eigenschaften und das Verhalten eines Systems definiert werden, resultiert aus einem Ausführungsfehler ein Fehlverhalten des Systems resp. der gesamten Anwendung, woraus eine Gefahrensituation entstehen kann.

Ausführungsfehler treten in drei Formen auf: als transiente, permanente und systematische Ausführungsfehler. Daneben gibt es Fälle von Interferenz zwischen Programmen, also eine unerwünschte Wechselwirkung zwischen Anwendungsprogrammen oder einem Anwendungsprogramm und Middle-Ware respektive dem Betriebssystem, die sich ebenfalls negativ auf die korrekte Ausführung der Software auswirken kann. Auf dreifache Weise können Ausführungsfehler und Interferenz die korrekte Ausführung eines Programms verfälschen: innerhalb des Datenflusses, des Kontrollflusses und des Zeitverhaltens. SCP erkennt transiente, permanente und systematische Ausführungsfehler und Interferenz innerhalb des Daten- und Kontrollflusses. Die geschützte Überwachung des Zeitverhaltens kann über eine Anpassung des Sicherheitskonzepts durchgeführt werden.

Arithmetische Kodierung

Die Grundlage des SCP bildet die arithmetische Kodierung, durch die ein gegebener, funktionaler Programm-Code in eine neue Form transformiert wird. Diese Transformation kann manuell, teilautomatisiert oder vollständig automatisiert mit Unterstützung der Entwicklungswerkzeuge SIListra Safety Tools erfolgen. Bild 2 visualisiert das

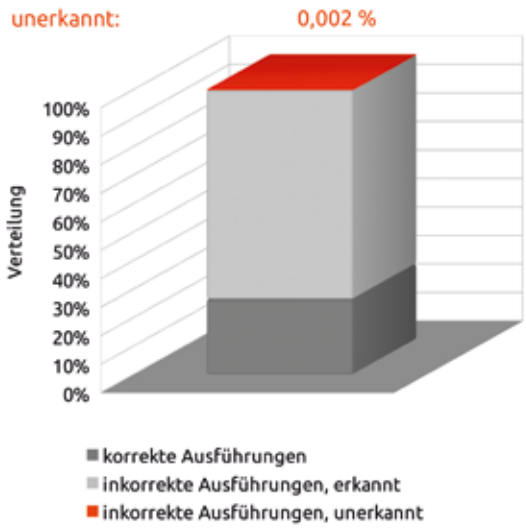

Bild 3:Experimentelle Ergebnisse (über 300 Millionen Fehlerinjektionen).

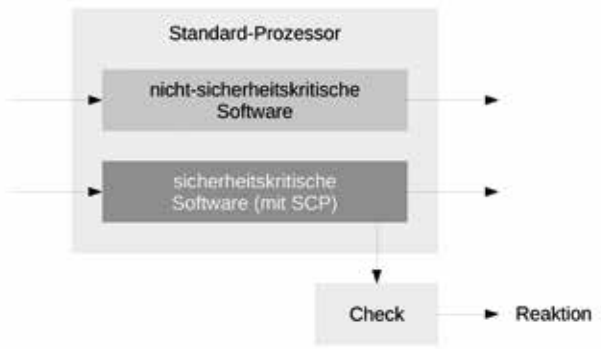

Bild 4: Integration sicherheitskritischer Software mit SCP auf einem Standard-Prozessor.

Funktionsprinzip der Code-Transformation: Ein funktionaler Programm-Code dient als Input für den Transformationsschritt (SCP). Das Ergebnis des Transformationsschritts wird übersetzt (Compiler) und auf die Ziel-Hardware (Teil des Systems) hochgeladen. Dadurch erhält man im Ergebnis ein System, das neben seiner originären Funktion – basierend auf dem Original-Programm-Code – zusätzlich SCP enthält und dadurch Ausführungsfehler offenbart.

Vorteile des SCP sind die Automatisierbarkeit in Form eines Entwicklungswerkzeugs, das einfach in bestehende Entwicklungswerkzeugketten integriert und bedient werden kann, die Unabhängigkeit der erzielten Ausführungsintegrität von der verwendeten Ziel-Hardware, die Offenbarung von Interferenz und Ausführungsfehlern während der Laufzeit sowie die hohe Absicherung, die sich durch geringe Restraten unerkannter Interferenzen und Ausführungsfehler beziffern lässt. Letzteres, die hohe Erkennungsrate von Ausführungsfehlern und Interferenzen, lässt sich durch Fehlerinjektion quantifizieren. Bei den injizierten Fehlern handelt es sich z. B. um fehlerhafte Operanden oder Operationen, die in einen exemplarischen, für die Automobiltechnik repräsentativen Programm-Code injiziert wurden. Bild 3 zeigt die Ergebnisse einer exemplarischen Messung, die auf Standard-Prozessoren durchgeführt wurde: Nur ein kleiner Anteil von 0,002% aller Fehlerinjektionen blieben unentdeckt. Dies bedeutet, dass 99,998% aller injizierten Fehler aufgedeckt wurden.

Die in Bild 3 vorgestellten Ergebnisse sind reproduzierbar und erlauben es insbesondere, diejenigen leistungsstarken Prozessoren in sicherheitskritischen Anwendungen zu verwenden, die ursprünglich nicht für den Einsatz in solchen Anwendungen konzipiert wurden, d. h., über keine sicherheitsgerichtete Architektur verfügen.

Software Coded Processing in der Praxis

Software Coded Processing ermöglicht die Integration sicherheitskritischer und nicht-sicherheitskritischer Software auf einer Standard-Rechenplattform (Bild 4). Die sicherheitskritische Software kann je nach Auslegung des Sicherheitskonzepts eine Betriebs- oder eine Überwachungsfunktion sein. Die Integrität der Ausgangsdaten aus der kodierten Anwendung wird in der Regel durch eine unabhängige Hardware-Einheit überprüft (Bild 4), z. B. durch einen Watchdog, FPGA oder intelligenten Aktuator. Bei Erkennung eines Fehlers wird durch die externe Hardware-Einheit eine geeignete Fehlerreaktion ausgelöst.

Abhängig von der Ausfallrate der verwendeten Hardware, dem zu erreichenden Diagnosedeckungsgrad und den verfügbaren Rechenressourcen lässt sich für einzelne Anwendungsfälle das Optimum zwischen Leistungsbedarf und Sicherheitsintegrität konfigurieren. Der Sicherheitsnachweis für die betrachtete Anwendung kann unabhängig von der ausführenden Hardware erbracht werden.

Das Verfahren Software Coded Processing ermöglicht somit die Weiterentwicklung rechenintensiver Systeme im Automobil unter Einhaltung strengster Sicherheitsanforderungen an eingebettete Rechner. ■ (oe)

» **www.silistra-systems.com**

Referenzen

"Is Software Coded Processing an Answer to the Execution Integrity Challenge of Current and Future Automotive Software-Intensive Applications?"; Majdi Ghadhab, BMW AG; Jörg Kaienburg, SIListra Systems GmbH; Martin Süßkraut, SIListra Systems GmbH; Christof Fetzer, Technische Universität Dresden; Advanced Microsystems for Automotive Applications 2015, Berlin

Jörg Kaienburg ist Geschäftsführer der SIListra Systems GmbH, 01277 Dresden.

Majdi Ghadhab arbeitet in der Entwicklung Antrieb, System- und funktionale Sicherheit Antrieb, BMW Group, München.