André Kohn (Autor)

Virtualisierung und Software-Parallelisierung für Fail-Operational Multicore-Domänensteuergeräte in der Automobilindustrie

https://cuvillier.de/de/shop/publications/8855

1 Elektronikentwicklung und Trends in der Automobilindustrie

Die Verbreitung und Bedeutung von Elektronik als tägliches Hilfsmittel und bei der Verbesserung von bestehenden Technologien steigt in der Gesellschaft kontinuierlich an. Leistungsfähige, informatiklastige IT-Systeme finden vermehrt den Weg auch in die Automobilindustrie, woraus sich neue Herausforderungen für die eher auf Maschinenbau spezialisierten Entwickler ergeben. Zahlreiche unterschiedliche Domänen, die bisher von mechanischen Systemen dominiert wurden, erfahren dabei einen grundlegenden Wandel.

1.1 Anforderungen an zukünftige E/E-Fahrzeugarchitekturen

Fahrzeuge bestanden in der Vergangenheit zum großen Teil aus klassischen, mechanischen Systemen. Ein Wechsel zu mechatronischen Systemen erfolgte durch die Einführung von immer mehr Elektronik und softwarebasierten Systemen. Diese Richtung setzt sich derzeit weiter fort und spiegelt sich in immer komplexeren IT-Systemen, die Einzug in das Fahrzeug erhalten, wider. Heutige E/E-Architekturen verwenden häufig einen modularen Ansatz, bei dem Funktionen durch individuelle Electronic Control Units (ECUs) voneinander isoliert sind. Ein erster Ansatz in Richtung einer optimierten Architektur beinhaltet die Hochintegration von Funktionen auf leistungsfähige Multicore-Controller. Dabei werden bestehende Steuergeräte konsolidiert, wodurch sich eine Einsparung sowohl bei der Steuergeräte-Hardware als auch beim Vernetzungsaufwand ergeben kann. Die aktuelle Entwicklungstendenz verfolgt eine domänenorientierte Zentralisierung auf sogenannte Domain Control Units (DCUs). Diese repräsentieren eine Schnittstelle zu Domänen wie Fahrwerk, Antrieb, Infotainment und Karosserieelektronik, mit dem Ziel einer möglichst einheitlichen Steuergerätearchitektur. Bereits heute sind zahlreiche neue Konzepte in Entwicklung, die Umstrukturierungen der bestehenden Fahrzeugarchitektur beinhalten, um die neue Anforderungen zukünftiger Fahrzeugfunktionen zu erfüllen. Ein Ansatz für weitere Verbesserungen ist eine Fusion bzw. Hochintegration der DCUs zu leistungsfähigen, zentralisierten Fahrzeugcomputern, die in bestimmten Zonen des Fahrzeugs verbaut werden. Die steigende Vernetzung kann zukünftig dazu führen, dass ein Teil der Steuergerätefunktionen in das Backend des Fahrzeugherstellers ausgelagert

wird (Abbildung 1.1). Die grundlegende Herausforderung bei diesen Ansätzen ist die Verbindung der klassischen Fahrzeugeigenschaften mit den umfangreichen Möglichkeiten der IT-Branche.

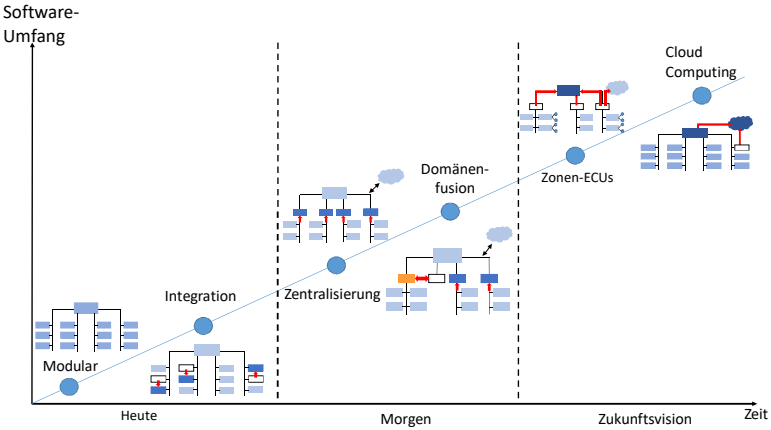

Abbildung 1.1: Evolution des Software-Umfangs in Fahrzeug-E/E-Architekturen

Im Rahmen dieser Arbeit werden unterschiedliche Herausforderungen adressiert, die sich durch eine DCU- und Fahrzeugcomputer-Architektur ergeben. Dazu werden entsprechend potenzielle Lösungsansätze von ausgewählten Themen aufgezeigt.

Grundsätzlich wird die Automobilindustrie mit immer mehr Kundenbedürfnissen und -anforderungen konfrontiert. Die Änderung der E/E-Architektur zu einem Domain Control Unit (DCU)-Ansatz, ist eine erste Maßnahme zur Erfüllung dieser Kundenanforderungen. Als zentraler Kommunikationsknoten kann ein Gateway dienen, das sowohl den Datenaustausch mit dem Backend, als auch zwischen den DCUs ermöglicht. Aufgrund des hohen Datenaufkommens ist eine Anbindung der DCUs an das Gateway über Automotive Ethernet notwendig. Die DCUs übernehmen dabei die Funktion eines zweiten Gateways, das die Kommunikation zwischen dem Ethernet und den entsprechenden Domänenbussen koordiniert. Die Funktionsweise und die Architektur der Steuergeräte innerhalb der Domänen sollen dabei möglichst unverändert übernommen werden. Folglich ist ein Signal-Routing zwischen Ethernet und FlexRay oder CAN(-FD) notwendig. Ein Beispiel für eine domänenbasierte E/E-Architektur ist in Abbildung 1.2 dargestellt.

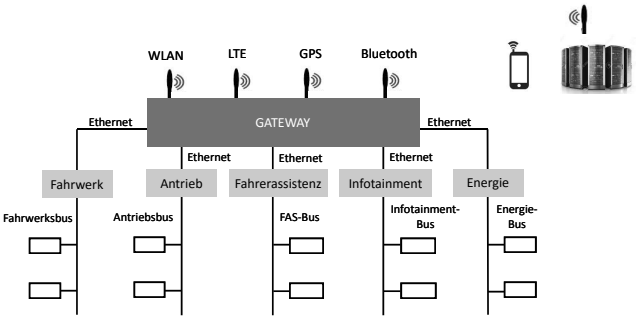

Abbildung 1.2: Beispiel für eine domänenbasierte E/E-Architektur

Die mit dieser E/E-Architektur zu erfüllenden Anforderungen werden in den folgenden Abschnitten beschrieben.

Online-Aktualisierung von Steuergerätesoftware

Eine wesentliche Neuerung der nächsten Fahrzeuggeneration ist die Möglichkeit, zusätzliche Funktionen nachträglich ins Fahrzeug einzubringen. Dies erfordert eine Infrastruktur mit einem ausgereiften OEM-Backendkonzept und der Kommunikationsanbindung über Mobilfunkstrecken. Zudem müssen die heutigen Steuergerätearchitekturen sowie die Vernetzungsarchitektur des Fahrzeugs überarbeitet werden, so dass der Kunde Funktionen bei Bedarf nachladen kann. Diese können zum einen bei der Fahrzeugauslieferung an den Kunden in Entwicklung sein oder noch nicht existieren. Zum anderen ist es vorstellbar, dass Funktionen im Fahrzeug bei der Fahrzeugauslieferung an den Kunden bereits vorinstalliert sind und nachträglich entgeltlich für einen definierten Zeitraum aktiviert werden. Denkbar sind Assistenzfunktionen oder eine gesteigerte Motorleistung für ein komfortableres oder sportlicheres Fahrerlebnis. Ein weiterer Anwendungsfall für das nachträgliche Einbringen von Software ist eine Softwareaktualisierung, wodurch potenzielle Security-Lücken geschlossen werden, um das Fahrzeug gegen unerlaubte Zugriffe und Manipulation zu schützen. Insbesondere die Anbindung von externen Geräten zur Kommunikation mit dem Fahrzeug oder zur direkten Fahrzeugsteuerung eröffnet Hackern viele neue Möglichkeiten, Zugriff auf das Fahrzeug zu erhalten.

Personalisierung und Profilierung

Bereits heute ist es durch vorgegebene Profile möglich das Fahrverhalten des Fahrzeugs von komfortabel, energieeffizient bis zu sportlich einzustellen. Durch die neue Fahrzeugarchitektur kann der Fahrer einerseits ein eigenes Profil im Fahrzeug individuell erstellen. Dabei gilt es genau abzuwägen, welche Fahrzeugparameter einstellbar sein sollen und mit welchen Möglichkeiten der Fahrer überfordert sein könnte. Andererseits können mit Hilfe der Daten, die während der Fahrt gesammelt worden sind, personalisierte Fahrzeugeinstellungen für ein verbessertes Fahrerlebnis vorgeschlagen werden. Die Personalisierung beinhaltet außerdem zusätzliche Funktionsempfehlungen, die für den Fahrer interessant sein könnten.

Datensammlung für vernetzte Fahrzeugfunktionen

Die Datenerfassung und -sammlung über den Fahrzeugzustand sowie des Fahrers ist ein weiteres Ziel in zukünftigen Fahrzeugen. Ein Großteil kommender Fahrzeugfunktionen sind abhängig von Daten die entweder durch andere Fahrzeuge, durch die Verkehrsinfrastruktur oder vom Backend bzw. der Cloud zur Verfügung gestellt werden. Häufig sind diese Funktionen nur möglich, wenn die Verkehrsinfrastruktur ebenfalls durch intelligente Elektronik erweitert wird (Parkhauspilot, Ampelassistent). Auch Assistenzfunktionen, die ein automatisiertes Fahren ermöglichen, erfordern zahlreiche Daten aus der Fahrzeugsensorik, die mit Backenddaten fusioniert werden. Weitere Konzepte beschreiben die Sammlung von Umgebungsdaten wie Straßenbeschaffenheit oder Luftqualität durch die Fahrzeugsensorik. Mit Hilfe dieser von dem Backend gelieferten Daten können nachfolgende Fahrzeuge beispielsweise ihre Dämpferregelung oder Innenraumklimatisierung prädiktiv anpassen.

Security-Mechanismen

Nahezu jede der zukünftigen Fahrzeugfunktionen benötigt Security-Mechnismen, die bis heute nur durch wenige Funktionen wie beispielsweise Wegfahrsperre oder Tuningschutz realisiert werden. Dazu zählt unter anderem die Kommunikation zwischen mobilen Geräten, die entweder mit einer direkten Verbindung oder über ein Backend und Mobilfunknetz erfolgen kann. Ein Steuergerät dient hier als Schnittstelle zwischen den Kommunikationsbussen des Fahrzeugs und dem Backend, so dass die Software- und Hardwarearchitektur entsprechend angepasst werden muss. Ein weiterer Anwendungsfall für Security ist die Sicherheit von Flashdaten für die Steuergeräte. Dabei existiert ein deutlicher Unterschied bei der Vorgehensweise im Infotainment und den klassischen ECUs. Während ein leistungsfähiges Infotainmentsteuergerät häufig über externe Medien geflasht wird,

ist dies bei ECUs mit harten Echtzeitanforderungen derzeit nur über die Diagnoseschnittstelle möglich. Bei Mikrocontrollern mit sicherheitsrelevanten Funktionen wird die Steuergerätesoftware aufgrund des geringen Speichers zunächst geflasht und danach verifiziert. Da durch das flexible Funktionsnachladen auch ein Rückflashen möglich sein muss, sind hier neue Konzepte notwendig. Weitere Security-Mechanismen für den Manipulationsschutz sind die Absicherung der Fahrzeugdiagnose sowie die Onboard-Kommunikation. Zur sicheren Schlüsselablage und Zertifikatsverwaltung wird außerdem ein Key Management System (KMS) benötigt.

Generell bestehen die Schutzklassen im Fahrzeug aus den Punkten *Safety*, *Geld & Geschäft*, *Datenschutz & Gesetze* und *Qualität* (Abbildung 1.3).

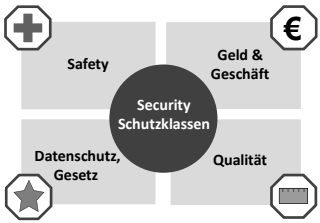

Abbildung 1.3: Security-Schutzklassen der Automobilindustrie

Safety zielt auf den Schutz der Fahrzeuginsassen und den anderen Verkehrsteilnehmern ab. Das Ziel ist die Gewährleistung der allgemeinen Betriebssicherheit und Zuverlässigkeit des Fahrzeugs und seiner Funktion. Bei einer Security-Lücke entsteht eine **Gefahr für Leib und Leben**.

Datenschutz & Gesetze konzentriert sich auf den Datenschutz der Kunden des Automobilherstellers. Ein unerlaubter Zugriff führt zu einer Verletzung der **Privatsphäre des Kunden sowie einem potentiellen Missbrauch der Kunden- und Herstellerdaten**.

Die Freischaltung von kostenpflichtigen Funktionen, beispielsweise bei Software as Product (SWaP), ist Bestandteil der Schutzkategorie *Geld & Geschäft*. Ein Auslesen von Fahrzeugdaten kann zu einem Verlust der Intellectual Property (IP) des Herstellers sowie zu einer generellen Gefährdung des Geschäftsmodells und folglich einem erheblichen **finanziellen Schaden des Herstellers** führen.

Bei einem Angriff auf die Produktqualität fallen einzelne Komponenten oder Fahrzeugfunktionen durch Manipulation aus oder liefern falsche Daten. Dies kann dazu führen, dass der Kunde die Qualität des gesamten Fahrzeugs anzweifelt. Ein Beispiel sind korrupte Anzeigen im Kombiinstrument oder ein Versagen des Fahrzeugs. Dadurch steigt die Wahrscheinlichkeit, dass der Kunde zukünftig auf

Fahrzeuge eines anderen Herstellers zurückgreift und so die Gefahr eines **Reputationsverlusts** besteht.

Weiterhin muss die Security-Architektur so umgesetzt werden, dass die funktionale Sicherheit bei einer Kompromittierung des Systems nicht beeinträchtigt wird. Dies ist besonders in Domänen mit sicherheitsrelevanten Funktionen von bedeutender Relevanz.

Aus diesen übergeordneten Zielen der Fahrzeug-E/E-Architektur ergeben sich spezifische Herausforderungen an die Steuergerätearchitekturen und insbesondere an die Hochintegration in Multicore-Systemen.

1.2 Herausforderung Multicore-Technologie in Domänensteuergeräten

Die umfangreichen, neuartigen Fahrzeugfunktionen und die damit verknüpfte gestiegene Kommunikation zwischen den Domänen und der Umgebung führen zu einer massiv erhöhten Softwarekomplexität. Daraus resultiert ein gesteigerter Bedarf an mehr Rechenleistung, der von den herkömmlichen Singlecore-Prozessoren nicht mehr erfüllt werden kann. In der IT-Domäne wurde diese Herausforderung lange Zeit durch eine Erhöhung der Taktfrequenz in den Prozessoren gelöst. Bei dieser Vorgehensweise wird aber gleichzeitig mehr Verlustleistung erzeugt, der mittels effizienter Kühlmechanismen entgegengewirkt werden muss. Gleichzeitig verdoppelt sich die Integrationsdichte bzw. die Anzahl der Transistoren. Eine Kühlung der Chips wird dadurch immer schwieriger und erreicht mit der sogenannten *Power Wall* das physikalische Limit (Abbildung 1.4).

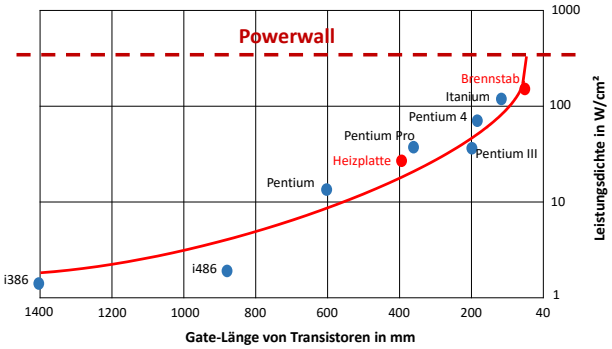

Abbildung 1.4: Leistungsdichte in Abhängigkeit von der Transistor Gate-Länge bei Prozessoren

Dies führte dazu, dass Singlecore-CPUs durch Multicore-Prozessoren ersetzt wurden. In eingebetteten Systemen setzt sich dieser Trend heutzutage fort. Für den DCU-Ansatz bedeutet dies, dass jede DCU eine leistungsfähige Hardwarearchitektur basierend auf Multicore-CPUs einsetzt. Die bestehende Software von älteren Singlecore-ECUs soll dabei nach Möglichkeit mit wenig oder gar keinen Anpassungen auf einem Multicore-System ausgeführt werden können.

Grundsätzlich ermöglichen Multicore-CPUs durch die erhöhte Rechenleistung eine Implementierung von komplexeren Funktionen zur Verbesserung des Fahrerlebnisses. Gleichzeitig darf bei hochintegrierten Steuergeräten in sicherheitsrelevanten Domänen die Sicherheit (Safety) nicht beeinflusst werden. Dazu sind diffizile Safety-Realisierungen notwendig, um möglichen kritischen Ausfällen vorzubeugen und entgegenzuwirken. Multicore-Prozessoren verwenden häufig für jeden Core sowohl individuelle, lokale Ressourcen als auch globale Komponenten, die von allen Cores gemeinsam genutzt werden können. Dazu zählen zum einen globaler Speicher und Peripherie wie I/O-Schnittstellen zum anderen auch der Interconnect für die Vernetzung der Cores untereinander. Zur Vermeidung eines zeitgleichen Zugriffs der Cores auf eine gemeinsame Ressource, muss diese räumlich und zeitlich partitioniert werden.

Für eine Konsolidierung von Software bieten Multicore-basierte DCUs sowohl Vorteile als auch neue Herausforderungen. Grundsätzlich müssen die Anforderungen des ISO 26262-Standards für funktionale Sicherheit eingehalten werden. Der Standard definiert mit den Automotive Safety Integrity Level (ASIL)[1] insgesamt vier Risikoklassen (A, B, C und D), die eine Adaption der Safety Integrity Level (SIL) nach dem internationalen IEC 61508 darstellen. Dabei beschreibt eine ASIL-D-Einstufung die höchsten und ASIL-A die niedrigsten Anforderungen an funktionale Sicherheit, wohingegen nicht sicherheitsrelevante Funktionen dem Quality Management (QM)-Level zugeordnet werden. Die wichtigsten Aspekte für die Funktionalität sicherheitsrelevanter Funktionen sind die Einhaltung harter Echtzeitanforderungen durch ein deterministisches Systemverhalten sowie statisch zugewiesene Speicherbereiche. Die Analyse des Timing-Verhaltens ist folglich ein essenzieller Bestandteil bei der Entwicklung und dem Design der Softwarearchitektur eines hochintegrierten Steuergeräts. Auf diese Weise ist ein frühzeitiges Identifizieren und Korrigieren von Fehlfunktionen als Resultat eines unpassendes Schedulings möglich. Zusätzlich werden Erkenntnisse zu den Kommunikationsbeziehungen gewonnen, die zu einem optimierten Softwaredesign führen. Die Timing-Analyse sorgt weiterhin für ein verbessertes Gesamtsystemverständnis, das für den Entwurf sicherheitsrelevanter Steuergeräte zwingend erforderlich ist.

[1] Der ASIL wird jeweils zu Beginn eines Entwicklungsprozesses bestimmt. Dabei werden die Systemfunktionen analysiert und in Bezug zu möglichen Risiken gestellt.

Interferenzfreiheit und Fehlerisolation

Grundsätzlich ist die Separierung bzw. Isolation unterschiedlicher Software in Domänensteuergeräten eine Voraussetzung für die Konsolidierung von ECUs. Ein möglicher Anwendungsfall sind Fahrerassistenzsysteme, die Signale und Parameter aus der Fahrwerk- und Antriebsdomäne nutzen. Darüber hinaus nähern sich die Funktionen aus Antrieb und Fahrwerk, insbesondere durch das elektrifizierte Fahren, immer mehr an, so dass eine Konsolidierung dieser DCUs zu einem übergeordneten Fahrwerk/Antriebssteuergerät langfristig vorstellbar ist. Ein Kernelement in einem hochintegrierten Multicore-System ist folglich die Isolation von unterschiedlichen Applikationen. Dies kann zum einen aufgrund der Konsolidierung von Software aus verschiedenen Steuergeräten und Fahrzeugdomänen sein. In diesem Fall kann jede Software eine andere ASIL-Einstufung benötigen, so dass die Interferenzfreiheit sichergestellt werden muss. Zum anderen kann eine Konsolidierung dazu führen, dass auf dem entsprechenden Steuergerät ein fahrzeugspezifisches System wie AUTOSAR parallel mit einem benutzerorientierten System wie Android ausgeführt wird. Weiterhin spielen ausfallsichere, redundante Systeme in Fahrzeug- und Steuergerätearchitekturen eine immer größere Rolle. Diese müssen bei einem Ausfall eines Teilsystems passende Mechanismen zur Fehlerdetektion und Fehlerbehandlung mit einer entsprechenden Fehlertoleranz zur Verfügung stellen. Eine entscheidende Maßnahme ist die Isolation von Fehlern, um die Ausbreitung auf den funktionierenden Teil des Systems zu vermeiden. Grundsätzlich können alle Fehler, die bei Singlecore-Prozessoren auftreten auch in Multicore-Systemen entstehen. Die Komplexität von Multicore-Architekturen kann zudem zu weiteren Fehlern führen, die in Singlecore-Architekturen ausgeschlossen sind.

Für jede geteilte Ressource existieren zwei zu isolierende Fehlertypen. Dies sind zum einen Fehler, die eine zeitliche Separierung vermeiden, womit ein Teil der Software durch den Einfluss eines anderen Teils die Timing-Anforderungen nicht mehr erfüllen kann. Zum anderen können Fehler entstehen, die Ressourcen (z.B lokaler Speicher) eines anderen Softwareteils modifizieren oder diese unerlaubt nutzen. Ein Isolationsmechanismus muss demnach in der Lage sein sowohl einen Fehler einzuschließen als auch die Software vor einer potenziellen Beeinflussung durch einen externen Fehler zu schützen. Ein Ansatz zur Isolation der Systeme ist eine hypervisorbasierte Systemvirtualisierung.

Anwendungsfälle für Virtualisierung in DCUs

Die hypervisorbasierte Systemvirtualisierung wird bereits in Server- und Cloud-Umgebungen zur Verbesserung der Datensicherheit und Systemisolation erfolgreich eingesetzt. Die Möglichkeiten sind folglich auch für eingebettete Systeme in der Automobilindustrie interessant. Insbesondere die Konsolidierung von Soft-

ware aus mehreren Steuergeräten und unterschiedlichen ASIL erfordert eine Isolation und die Sicherstellung von Interferenzfreiheit (*Freedom-from-Interference*) nach dem ISO 26262-Standard für funktionale Sicherheit. Dazu gehört neben der räumlichen Speicherpartitionierung eine, durch ein geeignetes Scheduling ermöglichte, zeitliche Separierung sowie eine Ende-zu-Ende-Kommunikationsabsicherung. Daneben kann diese Separierung auch die Ausbreitung von möglichen Fehlern der Funktion vermeiden, wobei der Fehler sich entweder direkt auswirken oder sich über eine Wirkkette von Funktionen fortpflanzen kann. Ein Beispiel für den Einsatz einer Systemvirtualisierung ist demnach die Separierung nach Kritikalität (Abbildung 1.5).

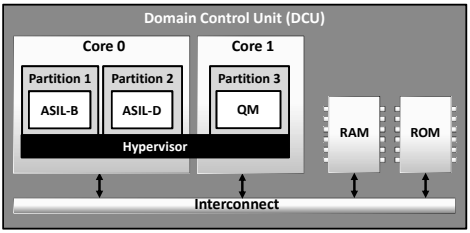

Abbildung 1.5: Beispiel für Separierung nach unterschiedlicher Kritikalität mittels Hypervisor

Die Implementierungsvariante in Abbildung 1.5 beinhaltet zwei Partitionen auf einem Core, jeweils mit unterschiedlicher Sicherheitseinstufung der Software. Auf dem zweiten Core befindet sich eine Partition ohne sicherheitsrelevanter Software (QM). Der dargestellte Anwendungsfall zeigt, wie eine Kapselung sowohl Core-intern durch mehrere Partitionen als auch durch eine Implementierung auf mehreren Cores umgesetzt werden kann.

Die Wiederverwendung von Steuergerätesoftware in neuen Hardwarearchitekturen ist ebenfalls ein essenzieller Bestandteil bei Domänensteuergeräten. Ein Ansatz für wiederverwendeten Code (Legacy Code) von Singlecore-Steuergeräten ist eine 1:1-Aufteilung, bei der jeweils ein Core auf dem Multicore-System für die jeweilige Singlecore-Software reserviert wird. Die direkte Core-Zuweisung führt zu einer physischen Isolation, wodurch die Softwarestacks den lokalen Core-Speicher exklusiv nutzen können. Die Kommunikation über den Fahrzeugbus muss mit Hilfe des Interconnects und der Verwendung des globalen Speichers entsprechend nachgebildet werden. Ein Beispiel mit zwei Cores und einer hypervisorbasierten Virtualisierung ist in Abbildung 1.6 dargestellt. Denkbar ist zudem ein redundantes System, bei dem im Fall eines Cores-Ausfalls die Funktionalität kurzfristig durch den zweiten Core ermöglicht wird. Unter der Voraussetzung, dass die

Hardwarearchitektur entsprechend redundante Komponenten verwendet, bietet dieser Ansatz Potenzial für die Realisierung eines Fail-Operational-Systems.

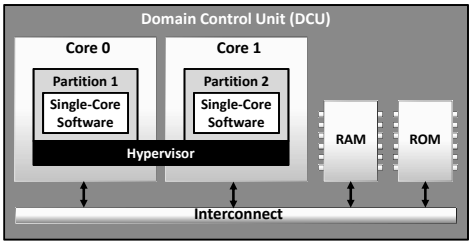

Abbildung 1.6: Beispiel für eine individuelle Core-Zuweisung mittels Hypervisor auf einer Multicore-DCU

Um Funktionen nachträglich ins Fahrzeug zu bringen, kann eine isolierte, separate Funktionspartition eingesetzt werden. Da Partitionen von einem Hypervisor neugestartet werden können, ist zudem eine Partition mit einer vorimplementierten Funktion denkbar, die von Fahrzeuginsassen bei Bedarf aktiviert werden kann. Bei dieser Implementierungsart können gleichzeitig unterschiedliche Betriebssystemen ausgeführt werden, wobei benutzerorientierte Systeme, wie beispielsweise Android, die eine Ausführung von Custom-Code erlauben, besonders kritisch sind. Dadurch können andere Funktionen beeinträchtigt werden, so dass sicherheitsrelevante Applikationen ihre Anforderungen nicht mehr erfüllen können. Dementsprechend ist in diesem Fall eine genau definierte Isolation notwendig. Alternativ kann die Funktionspartition einen Security-Stack beinhalten, um die Kommunikation vor unbefugten Zugriffen abzusichern (Abbildung 1.7). Diese Implementierung ist vergleichbar mit der *TrustZone*-Technologie, die in Abschnitt 2.2.6 beschrieben wird.

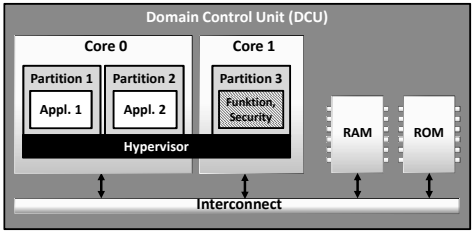

Abbildung 1.7: Beispiel für eine DCU-Architektur mit einer separaten Partition zum Nachladen von Funktionen oder Security-Mechanismen