Virtualisierung bei Eingebetteten Multicore Systemen: Integration und Isolation sicherheitskritischer Funktionen

Robert Hilbrich (robert.hilbrich@first.fraunhofer.de)
Matthias Gerlach (matthias.gerlach@opensynergy.com)

Abstract: Virtualisierung ist nicht nur ein aktuelles Thema für Rechenzentren und deren Server. Auch bei sicherheitskritischen, eingebetteten Systemen werden Virtualisierungstechnologien eingesetzt. Dieser Trend ist auch das Ergebnis einer zunehmenden Verwendung von Mehrkern Prozessoren zur Erhöhung der Funktionsdichte und zur weiteren Senkung der Kosten. Diese Arbeit beschreibt die Verwendung von Virtualisierung im Engineering-Konflikt zwischen dem Streben nach gleichzeitiger *Isolation* und *Integration* eines Gesamtsystems. Dies führt zu einer Spezialisierung des Begriffs *Virtualisierung*, um den Anforderungen sicherheitsgerichteter, eingebetteter Systeme stärker Rechnung zu tragen. Anhand eines aktuellen Fallbeispiels aus dem Automotive-Kontext werden diese neuen Anforderungen in einem konkreten Anwendungskontext illustriert und diskutiert. Als Ausblick auf zukünftige Herausforderungen werden abschließend die zukünftigen Entwicklungen bei Manycore Architekturen vorgestellt und die wesentlichen Potentiale und Auswirkungen für das Systemdesign herausgearbeitet.

1 Einführung und Hintergrund

Die Entwicklung leistungsfähiger Prozessoren war in den letzten Jahren durch die Abkehr von der Leistungssteigerung durch eine weitere Erhöhung der Taktfrequenz gekennzeichnet. Der Grund dafür liegt in der *Power Wall*, die eine physikalische und ökonomische Grenze bei der Entwicklung der Leistungsdichte von Prozessoren beschreibt.

Der einzig aktuell verfügbare Weg zur Steigerung der Leistungsfähigkeit liegt daher nicht in der Erhöhung der *Peak-Performance*, sondern viel mehr in der Steigerung des *Durchsatzes* Prozessors mit Hilfe von mehreren, parallel und unabhängig voneinander arbeitenden Rechenkernen.

Multicore Prozessoren, als Dual- oder Quad-Core, gehören mittlerweile im Server-Bereich und auch im Desktop-Bereich zum Austattungsstandard. Die Durchdringung dieser neuen Prozessorgeneration im Bereich komplexer, Software-intensiver, eingebetteter Systeme verläuft dagegen wesentlich langsamer. Während bereits zertifizierte Steuerungen für Gelenkarmroboter auf der Basis von Dual-Core Prozessoren verfügbar sind, konnten sich Mehrkernprozessoren in sicherheitskritischen Domänen, wie der Avionik oder im Automotive Bereich, trotz ihrer Vorteile noch nicht in der Breite durchsetzen.

Bei der Betrachtung des theoretischen Durchsatzes von aktuellen Multicore Prozessoren wird deutlich, dass damit mittlerweile auch eingebettete Systeme mit einer Rechenleistung ausgestattet werden können, die bis vor wenigen Jahren noch Servern und *Number-*

Crunchern vorbehalten war. Während dort jedoch die parallele Verarbeitung von Algorithmen von Anfang an ein fester Bestandteil der Architektur und des Software-Designs war, wird der Bereich der eingebetteten Systeme nun an vielen Stellen erstmalig mit Parallelverarbeitung und den damit verbundenen Herausforderungen konfrontiert. Viele Entwicklungsmuster im Umgang mit komplexen Steuerungsaufgaben eingebetteter Systeme basieren auf einer seriellen Ausführung mit einem zumindest temporär exklusiven Zugriff auf Hardware-Ressourcen, so dass beispielsweise kritische Code-Abschnitte durch ein Abschalten von Interrupts vor unerwünschten Unterbrechungen geschützt werden können.

Aus diesem Grund müssen erprobte Techniken und Entwicklungsstrategien mit der Einführung von Mehrkernprozessoren erneut auf den Prüfstand. Gleichzeitig werden dabei aber auch Technologien, die bereits seit vielen Jahren erfolgreich im Server-Bereich für parallele Architekturen eingesetzt wurden, hinsichtlich ihrer Tauglichkeit für eingebettete Systeme evaluiert. Dazu zählt insbesondere auch *Virtualisierung* (siehe [Ros04, RG05] für einen Überblick zu aktuellen Entwicklungen im Bereich der Virtualisierung).

Die Entwicklung von sicherheitskritischen, eingebetteten Systemen kann damit auf fundierte Erfahrungen aus der Entwicklung von effizienten Hypervisoren und Mechanismen zum Schutz von Speicherzugriffen zurückgreifen. In Verbindung mit Multicore Prozessoren bietet sich damit das Potential zur signifikanten Erhöhung der Funktionsdichte bei geringerem Platz-, Gewichts- und Energieverbrauch. Eine *zertifizierbare* Nutzung mehrerer Rechenkerne wird damit zu einem relevanten Wettbewerbsvorteil.

Als direkte Folge der Evaluation von Virtualisierungstechniken im Embedded-Bereich, wird diese aber auch mit neuen Anforderungen konfrontiert, die sich speziell aus der neuen Anwendungsdomäne ergeben. Dazu gehört die Adressierung von erforderlichen Systemeigenschaften wie *Echtzeitfähigkeit* und *Determinismus*, aber auch eine wesentliche größere Bedeutung von *Energiesparmechanismen*, sowie die Fähigkeit zum adäquaten Umgang mit permanent wechselnden Umgebungskontexten.

Virtualisierung in der Form, wie sie bisher in ihren klassischen Anwendungsgebieten zum Einsatz kam, kann daher *nicht nahtlos* in den Bereich eingebetteter Systeme übernommen werden. Es ist vielmehr erforderlich, eine neue Spezialisierung dieses Oberbegriffs herauszuarbeiten, die auch den neuen Anforderungen Rechnung trägt.

2 Zuverlässigkeit erfordert die Reduktion von Komplexität

Eine der maßgeblichen Ursachen für den stetig steigenden Entwicklungsaufwand von sicherheitskritischen, eingebetteten Systemen liegt in der stark ansteigenden Komplexität. Diese ist das Ergebnis von steigenden Anforderungen an die Funktionalität, die neben den reinen Kosten ein wesentliches Distinktionsmerkmal zu konkurrierenden Produkten am Markt darstellen.

Im Bereich der Flugelektronik (Aviation Elektronik - *Avionik*) können die hoch-komplexen Interaktionsmuster zwischen den beteiligten Steuergeräten mittlerweile nur noch mit Hilfe von Software gesteuert werden.

Auch im Automotive Bereich ist ein vergleichbarer Trend zur weiteren Steigerung der Komplexität der entwickelten Systeme zu erkennen. Fahrerassistenzsysteme werden auch bei Kleinwagen zum Standard und entwickeln sich von ehemals reinen Komfortfunktionen zu Safety-relevanten Komponenten, die an immer mehr Stellen in den Prozess des Fahrens eingreifen. Beispielsweise sind die verbauten Radar-Einheiten in neueren Oberklasse-Fahrzeugen nicht nur in der Lage, den Abstand zum nächsten Hindernis zu bestimmen. Mittlerweile benötigen diese Geräte Dual-Core Prozessoren und können damit mehrere vorausfahrende Fahrzeuge einzeln identifizieren und deren Positionsveränderung verfolgen. Der Schritt zur automatischen Erkennung von Gefahrensituationen und dem Einleiten entsprechender Maßnahmen ist dann nur noch ein kleiner.

Bei genauerer Analyse des Komplexitätsbegriffs wird deutlich, dass diese das Resultat vom Zusammentreffen zweier Parameter ist: *Funktionsvielfalt* und *Dynamik* . Umfangreiche Funktionalitäten können problemlos realisiert werden, wenn dafür ausreichend zeitliche Reserven zur Verfügung stehen. Andererseits führt die Forderung nach schnellen Reaktionen bei einfachen Aufgaben auch nicht zu hohen Komplexitätsniveaus. Erst wenn komplizierte Entscheidungsprozesse und Steuerungsaufgaben sehr schnell durchgeführt werden müssen, ergibt sich ein hohes Maß an Komplexität (siehe Abbildung 1).

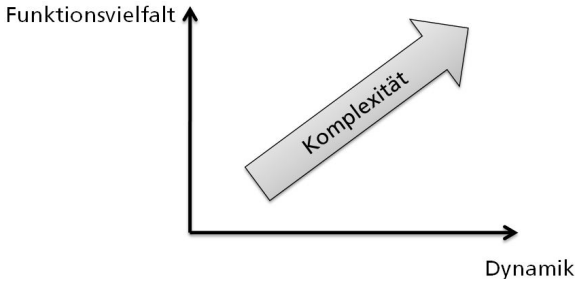


Abbildung 1: Erst das Zusammentreffen einer hohen Dynamik mit einer hohen Funktionsvielfalt führt zu einem Anstieg der Komplexität des Gesamtsystems.

Kurz gesagt: Als Konsequenz einer verstärkten Verwendung von Mehrkernprozessoren im Embedded Bereich wird die Funktionsdichte weiter ansteigen ohne das dabei Echtzeiteigenschaften vernachlässigt werden dürfen. Damit wird sich die Komplexität zwangsläufig erhöhen und damit auch zu einer Steigerung der Wahrscheinlichkeit für unentdeckte Fehler im Systemdesign, einer höheren Anfälligkeit für zufällige Fehler zur Laufzeit, aber zu mehr Fehlern in der Bedienung führen.

3 Virtualisierung - ein technischer Kompromiss im Konflikt zwischen Isolation und Integration

Das Design von sicherheitskritischen, eingebetteten Systemen stellt sich - abstrakt gesehen - als die gezielte Suche nach einem ökonomisch vertretbaren Ausgleich zwischen den

konfligierenden Designzielen *Isolation* und *Integration* dar. Die Softwarearchitektur eines eingebetteten Systems muss einerseits ein hinreichende Maß an Isolation zwischen den einzelnen Bestandteilen bieten, um eine unerwünschte und intransparente Fehlerpropagation zu verhindern. Dies ist das Ergebnis von Anforderungen zur funktionalen Sicherheit des Gesamtsystems. Neben einem hohen Maß an Isolation muss die Architektur auch eine starke Integration der einzelnen Komponenten in Form einer engen Vernetzung aufweisen. Notwendig ist dies für die Realisierung von komplexen Steuerungs- und Interaktionsmustern.

In der Vergangenheit wurde dieser Konflikt bereits generisch im Kontext von verteilten Betriebssystemen für Mehrprozessorsysteme unter dem Stichwort *Partitioning* und *Kooperation* untersucht [Ous80]. In dieser Arbeit wird dieser Konflikt spezifisch für den Bereich der sicherheitsgerichteten, eingebetteten Systeme herausgearbeitet.

Dieser Grundkonflikt hat signifikante Auswirkungen auf weitere Architektureigenschaften und Entwicklungsentscheidungen (siehe Abbildung 2). So erhöht eine verstärkte Integration zwischen den Einzelkomponenten zwar das Level der damit realisierbaren Funktionsvielfalt und senkt in der Regel auch die Materialkosten, gleichzeitig geht dies aber auf Kosten der Zuverlässigkeit, da die Wahrscheinlichkeit für eine intransparente Fehlerpropagation steigt, und erhöht zudem auch den Entwicklungsaufwand durch eine aufwändige Integrationsphase.

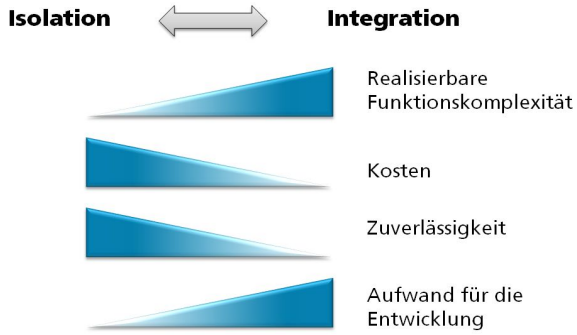


Abbildung 2: Der Grundkonflikt zwischen Isolation und Integration findet sich auch in weiteren Architektureigenschaften und Entwicklungsentscheidungen wieder.

Bisher wurde der beschriebene Grundkonflikt bei Entwicklung von sicherheitskritischen, eingebetteten Systemen verstärkt zugunsten der Isolation aufgelöst. Besonders deutlich wird dies an den *Federated Architectures* im Avionik Bereich, bei denen jede Avionik-Funktion durch separate Hardware-Komponenten und Kommunikationskanäle realisiert wurde. Bei diesen Architekturen ergibt sich eine *inhärente* Isolation, da durch die Verwendung einer großen Bandbreite von *heterogenen* Hardware-Komponenten die Wahrscheinlichkeit für ein vollständiges Systemversagen als Folge eines unentdeckten Design-Fehlers - genauer gesagt: der Propagation eines Fehlers über gemeinsame Ressourcen hinweg - minimiert wird.

Mittlerweile bekommt der Aspekt der Integration allerdings ein stärkeres Gewicht. Durch standardisierte und von mehreren Applikationen gemeinsam verwendete Hardwarekomponenten, können die Entwicklungs-, Beschaffungs- und Wartungskosten spürbar gesenkt werden. Deutlich wird diese Entwicklung im Luftfahrtbereich am Konzept der *Integrated Modular Avionics (IMA)*. IMA steht für eine gemeinsam genutzte Computer-Plattform für verschiedene Avionik-Systeme, die zwar logisch gesehen als *zentrale* Plattform konzipiert ist, aber aus Gründen der Zuverlässigkeit physisch auf verschiedene Zonen im Flugzeug verteilt ist.

Da nun eine automatische Isolation zwischen den Avionik-Systemen auf der Hardware-Ebene durch die Verwendung *homogener* Hardware-Komponenten nicht mehr gegeben ist, muss der Forderung nach Isolation im Bereich der Software Rechnung getragen werden. An dieser Stelle präsentiert sich Virtualisierung als vielversprechender Ansatz, um *Safety-Mechanismen* zu realisieren, so dass trotz einer hoch integrierten Hardware-Basis ein hinreichendes Maß an Isolation zwischen den Software-Komponenten erreicht werden kann.

Dies zeigt, wie vielfältig Virtualisierung eingesetzt werden, aber auch welche neuen Anforderungen sich durch den Einsatz bei sicherheitskritischen, eingebetteten Systemen ergeben. Eine wichtige Herausforderung, die den breiten Einsatz von Virtualisierungslösungen bei eingebetteten Systemen verhindert, ergibt sich aus dem Umgang mit stark Hardware-nah implementierten Komponenten, die die Verwendung von Abstraktionsschichten erschweren und teilweise auch unmöglich machen. Während dies im Server-Bereich nur eine untergeordnete Rolle spielte, ist dieser Aspekt gerade bei eingebetteten Systemen von hoher Bedeutung.

Beispielsweise nutzen die Steuergeräte im Auto hochgradig Hardware-spezifische Funktionen und teilweise auch spezielle ASICs, um die hohe Komplexität der Signalverarbeitung in einem engen Zeitraster realisieren zu können. Ein weiteres Beispiel ist der Zugriff auf 3D-Beschleuniger, wie sie in Navigationsgeräten zur graphischen Anzeige der Routeninformationen verwendet werden. Auch hier fordert die Nutzung von Virtualisierungslösungen ihren Preis.

Kurz gesagt: Virtualisierung bietet hinsichtlich der damit verwendbaren Isolationstechniken ein großes Einsatzpotential. Allerdings führt die Einführung zusätzlicher Abstraktionsschichten von physischen Ressourcen zu neuen Herausforderungen bei Echtzeitanforderungen und der leichten Migration von Legacy-Software.

4 Sichere Partitionierung in Raum und Zeit ist das Ziel der Virtualisierung bei sicherheitskritischen, eingebetteten Systemen

Die klassischen Ziele, die den Einsatz von Virtualisierungstechnologien im Server- und teilweise aber auch im Desktop-Bereich motivieren, sind mit der Möglichkeit eines effizienteren *Multiplexens* von Hardware Ressourcen verbunden oder liegen in der Nutzung der Vorteile von zusätzlichen Hardware-Abstraktionsschichten, zum Beispiel für eine Reduzierung der Aufwände in der Software-Entwicklung.

Bei eingebetteten Systemen verschiebt sich dieser Fokus aufgrund anderer Anforderungen. Mit dem Einsatz von Virtualisierungstechnologien soll hier vielmehr eine nachweisbar, zuverlässige Aufteilung und Isolation der gemeinsam genutzten Hardware-Ressourcen, wie zum Beispiel CPU Zeit oder Speicher, erreicht werden. Damit Echtzeitanforderungen zertifizierbar erfüllt werden können, muss trotz zusätzlicher Abstraktionsschichten von der Hardware ein deterministischer Zugriff auf die physischen Ressourcen realisiert werden. Eine Verschleierung der physischen Ressourcen ist daher nicht als Vorteil, sondern viel mehr als unerwünschter Nebeneffekt und eventueller Kostentreiber zu sehen.

Der Einsatz von Virtualisierungstechnologien bei sicherheitskritischen, eingebetteten Systemen wird daher verstärkt durch den Begriff *sichere Partitionierung* charakterisiert, um trotz der gemeinsamen technischen Basis, die unterschiedlichen Zielstellungen zu unterstreichen.

Die genauen Anforderungen, die durch eine sichere Partitionierung für Steuergeräte im Bereich der Avionik erreicht werden müssen, werden unter *Time and Space Partitioning (TSP)* (siehe [RTC94] Seite 9) zusammengefasst. Damit wird ein Grad an Isolation zwischen einzelnen Partitionen auf einem Prozessor bezeichnet, der sowohl eine räumliche Dimension (*Space*) aber auch eine zeitliche Dimension (*Time*) besitzt.

Räumliche Partitionierung stellt in erster Linie sicher, dass die Software innerhalb einer Partition auf die privaten Daten einer Applikationen in einer anderen Partition nicht zugreifen kann. Dies bezieht sich nicht nur auf die Verwendung von speziellen Mechanismen bei der Virtualisierung die eine Isolation des gemeinsamen Speichers realisieren, sondern meint auch den Schutz des Zugriffs auf private Geräte einer Partition, zum Beispiel Aktuatoren.

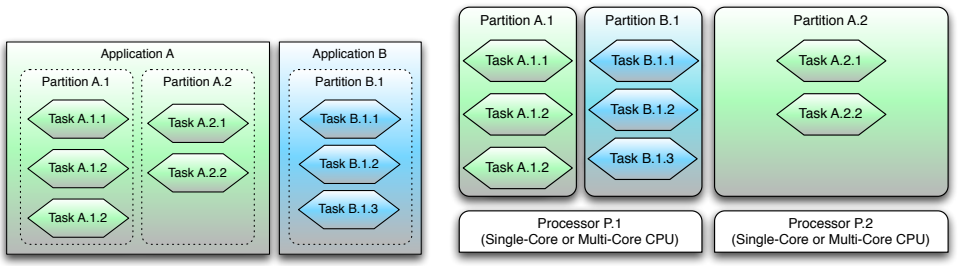
Zeitliche Partitionierung garantiert im Gegensatz dazu, dass das zeitliche Verhalten der Applikationen in einer Partition, zum Beispiel die *Worst Case Execution Time*, nicht durch die Ausführung von Applikationen in einer anderen Partition beeinflusst wird. Dies berührt den Aspekt der aufwändigen Erstellung von Betriebssystem-Schedules für Anwendungen in verschiedenen virtuellen Maschinen auf einer gemeinsamen Hardware-Plattformen. Dieses Problem wurde in der Arbeit von R. Kaiser [Kai09] untersucht.

Bei der Verwendung von Interrupts stellt die zeitliche Partitionierung eine besondere Herausforderung für die Virtualisierungsschicht dar, denn das *spontane* Auftreten von Interrupts und die Ausführung der erforderlichen Interrupt Service Routinen (ISRs) lässt sich zum Entwicklungszeitpunkt nicht sinnvoll einschränken. Damit besteht die Gefahr, dass durch fehlerhaft ausgelöste Interrupts, ein signifikanter Anteil der Rechenzeit zwangsläufig für die Behandlung der ISRs zur Verfügung gestellt werden muss.

Eventuell steht dann für die einzelnen Partitionen nicht mehr genügend Rechenzeit zur Verfügung, so dass diese ihre Aufgaben nicht mehr erfüllen können. Ein Ausweg bieten hier Verfahren, die zur Laufzeit auf der Basis von Ressourcen-Kontingenten die Behandlung von Interrupts ermöglichen. Überschreitet ein fehlerhafter Interrupt sein vorher festgelegtes Kontingent für die ISRs, wird dieser temporär deaktiviert, um die zeitliche Partitionierung sicherzustellen.

Das genaue Verhältnis zwischen *Tasks*, Applikationen und Partitionen auf der physischen Hardware mit den jeweiligen Virtualisierungsschichten wird in der Avionik im Standard

RTCA ARINC 653 [ARI05] definiert (siehe Abbildung 3). Eine Applikation kann dabei aus einer oder mehreren Partitionen bestehen. Jede Partition enthält einen oder mehrere Tasks, deren Ausführung innerhalb der Partition auch durch eigene Scheduling-Verfahren koordiniert werden kann. Partitionen werden statisch auf Prozessoren verteilt. Die Ausführung von verschiedenen Partitionen auf einem Prozessor wird durch ein statisches Scheduling koordiniert. In [HG11] wird ein modellbasierter Ansatz beschrieben, mit dem *korrekte Schedules* für sicherheitskritische Echtzeitsysteme auf der Basis einer Formalisierung der zeitlichen Anforderungen von den verschiedenen Applikationen automatisiert konstruiert werden können.



(a) Software Architektur (Logische Sicht)

(b) Verteilung der Partitionen auf die Hardware (Physische Sicht)

Abbildung 3: RTCA ARINC 653 Terminologie zur Partitionen.

Eine unmittelbarer Vorteil der sich durch die Verwendung von sicherer Partitionierung für die Entwicklung und Wartung von komplexen, sicherheitskritischen Systemen ergibt, liegt in der *inkrementellen Zertifizierbarkeit*. Software für sicherheitskritische Bereiche wird immer gemäß den Anforderungen einer bestimmten Kritikalitätsstufe entwickelt. Im Luftfahrtbereich werden dazu verschiedene *Design Assurance Level (DAL)* unterschieden, wohingegen im Automotive Bereich zum Beispiel *Automotive Software Integrity Levels (ASIL)* verwendet werden. Allgemein gilt: ein höheres Kritikalitätslevel ist immer mit einem erhöhten Entwicklungs- und Zertifizierungsaufwand verbunden.

Sichere Partitionierung in Raum und Zeit ermöglicht durch das hohe Maß an Isolation die Integration von Software-Komponenten mit unterschiedlichen Kritikalitätsstufen auf der gleichen Hardware-Plattform. Beispielsweise kann eine unkritische Funktion zusammen mit einer kritischen Funktion auf der selben Hardware eingesetzt werden. Inkrementelle Zertifizierbarkeit bedeutet nun, dass die Hardware-Plattform und die Virtualisierungsschicht gemäß den Anforderungen für die kritische Applikation entwickelt werden müssen, aber die unkritische Applikation dabei nicht betrachtet werden muss. Sie kann mit weniger Aufwand entwickelt und später auch aktualisiert werden, ohne dass dadurch eine aufwändige Rezertifizierung der kritischen Applikation notwendig wird.

5 Multicore und Funktionale Sicherheit - auch ohne Virtualisierung?

Der Einsatz von Virtualisierungstechnologien gerade im Umfeld von eingebetteten Systemen - ein traditionell sehr kostensensitiver Markt - wird in der Diskussion oft hinterfragt. Gerade in Hinblick auf die statisch zur Entwicklungszeit konfigurierten *Electronic Control Units (ECUs)* im Automobil, stellt sich die berechtigte Frage, ob Virtualisierung in Anbetracht dieser Randbedingungen mit einem zu hohen *Overhead* durch die Einführung einer weiteren Abstraktionsschicht in der Software verbunden ist.

(a) Feste Zuordnung der Funktionen zu den *Cores*

(b) Nutzung einer Virtualisierungsschicht zur Zuordnung der Funktionen zu den *Cores*

Abbildung 4: Architekturansätze zur Konsolidierung von Funktionen auf Mehrkernprozessoren

Virtualisierung muss sich hier mit alternativen Architekturansätzen (siehe Abbildung 4) messen lassen, bei denen Funktionen statisch zu den einzelnen Kernen eines Multicore Prozessors zugeordnet und dann zur Laufzeit parallel ausgeführt werden. Der Gedanke der Konsolidierung von Steuergeräten wird dabei an sich konsequent weitergeführt, wobei nun ein einzelner *Core* die Funktion eines Prozessors der "alten" ECU übernimmt. Vereinfacht gesprochen können damit auf einem Prozessor mit n Kernen bis zu n ECUs konsolidiert werden.

Während dieser Ansatz durch die geringere Komplexität der Softwarearchitektur erkennbare Vorteile bietet, greift er hinsichtlich verschiedener Aspekte im Rahmen der zukünftigen Entwicklung von Mehrkernprozessoren zu kurz. Zunächst beschränkt er die Anzahl der aktuell konsolidierbaren Funktionen auf die Menge an Rechenkernen, so dass aktuell "nur" ca. zwei bis acht Funktionen auf einem Prozessor integriert werden können. Falls eine Funktion einen Rechenkern nicht vollständig ausnutzt, weil sie beispielsweise *memory-bound* implementiert ist, ergibt sich hier eine Ressourcen-Unterauslastung. Mit Hilfe einer Virtualisierungsschicht können signifikant mehr Software-Funktionen in einzelnen Partitionen auf einem Prozessor integriert werden, so dass damit eine optimale Ressourcennutzung gewährleistet wird.

Weiterhin ist absehbar, dass die einzelnen Kerne eines Mehrkernprozessors in Zukunft langsamer als ihre Vertreter der Singlecore Prozessoren arbeiten werden, um den rapiden Anstieg der Leistungsaufnahme zu reduzieren (siehe [Bor07]). Eine einfache 1:1 Übertragung einer Software-Funktion von einem Singlecore auf einen Multicore Prozessor wird daher bei dieser Architektur zu einer Verschlechterung der Performance führen.

Im Zuge einer stärkeren Bedeutung von Energiesparmechanismen, zum Beispiel im Rahmen der aktuellen Forschungsprojekte im Bereich *Elektro-Mobilität*, erscheint die Verwendung einer Virtualisierungsschicht vorteilhaft, da hier eine *flexiblere Ressourcenzutei-*

lung möglich ist. Unkritische Funktionen können dann leichter zwischen verschiedenen Kernen migriert werden, so dass ungenutzte Teile des Mehrkernprozessors in einen Energiesparmodus versetzt werden können. Falls zur Laufzeit mehr Leistung für eine Funktion benötigt wird und diese die Verwendung mehrere Kerne unterstützt können ebenfalls dynamisch ungenutzte Ressourcen zur Verfügung gestellt werden, um eine Leistungssteigerung zu erreichen.

Der größte Nachteil dieser vereinfachten Architektur liegt jedoch im Bereich der Zuverlässigkeit und der Toleranz von Laufzeitfehlern. Einzelne Cores bieten durch ihren Befehlssatz die Möglichkeit der Trennung von privilegierten und nicht-privilegierten Instruktionen und damit einen gewissen Schutz vor den Auswirkungen von Laufzeitfehlern auf andere, parallel arbeitende Softwarekomponenten. Für eine sichere Partitionierung in Raum und Zeit (siehe Abschnitt 4) genügt eine Trennung von Instruktionen jedoch nicht, um eine Propagation von Fehlern über gemeinsam genutzte Ressourcen zu verhindern. Die Nutzung einer zusätzlichen Virtualisierungsschicht in der Softwarearchitektur ermöglicht nicht nur eine dynamische und damit flexible Zuordnung von Ressourcen zu Software-Funktionen, sie kann auch zur sicheren Partitionierung von gemeinsam genutzten Kommunikationskanälen und anderen I/O Komponenten genutzt werden (siehe dazu auch [Rus99]).

Je weiter die Entwicklung von Mehrkernprozessoren voranschreitet und je stärker damit eine höhere Integration von Komponenten realisiert wird, umso mehr wird die Partitionierung von gemeinsam genutzten Ressourcen zusätzlich zur Prozessorzeit in den Fokus geraten. Eine höhere Integration ist letztlich das Ergebnis einer gemeinsamen Nutzung von bisher exklusiv genutzten Komponenten. Damit führt eine höhere Integrationsdichte automatisch zu einem Anstieg der Wahrscheinlichkeit für eine Fehlerpropagation, so dass sichere und zertifizierbare Schutzmechanismen in der Software in Zukunft von essentieller Bedeutung sind.

Die Nutzung einer Virtualisierungsschicht, um eine sichere Partitionierung in Raum und Zeit zu realisieren, bietet ein großes Potential zur Kosten- und Gewichtsminimierung sowie zur Steigerung der Zuverlässigkeit bei sicherheitskritischen, eingebetteten Systemen. Insbesondere können beliebig viele voneinander isolierte Partitionen auf einem Prozessor integriert werden. Damit können einerseits die angebotenen Ressourcen eines Systems optimal genutzt werden sowie bei einer Vollvirtualisierung die Software in den Partitionen weitgehend unabhängig von der konkreten Prozessorarchitektur entwickelt werden.

6 Virtualisierung - Herausforderung für die Automotive-Domäne?

Während Virtualisierung in der Softwarearchitektur im Luftfahrtbereich bereist seit vielen Jahren eingesetzt wird (siehe Abschnitt 4), ist dieser Ansatz in der Automotive-Domäne noch nicht sehr verbreitet und trifft auf verschiedene Hürden, die nicht nur im Bereich der Softwaretechnik liegen.

Aktuelle Hardware-Architekturen im Automobilbereich sind durch eine gezielte Verteilung verschiedener Prozessoren auf das gesamte Fahrzeug gekennzeichnet, um an geeigneten Stellen eine lokale Vorverarbeitung durchzuführen und damit zentrale Steuerungs-

logiken zu entlasten. Gleichzeitig sind die Kabelbäume, die das Rückgrat einer derartigen Architektur darstellen, bereits heute für jedes Fahrzeug entsprechend der vorhandenen Funktionen konfektioniert, um Materialkosten zu senken. Eine fiktive Konzentration *alle* Funktionen auf *einem* Steuergerät würde daher vermutlich zu höheren Kosten führen. Gleichzeitig sind die Prozessoren von bestimmten ECUs speziell auf ihre Funktion angepasst, um die Herstellungskosten zu reduzieren. Ein prägnantes Beispiel für diese Anpassung stellen Fahrerassistenzsysteme dar, die mit speziellen Digitalen Signalprozessoren (DSPs) eine effiziente Verarbeitung der Kamera-Daten ermöglichen (siehe [Weu11]).

Vor diesem Hintergrund ist die Frage berechtigt, ob – und wenn ja wo – eine Funktionsintegration auf homogenen Mehrkernprozessoren mit Hilfe von Virtualisierungstechnologien im Automobil sinnvoll ist. Die *Head-Unit* ist daher eine der Komponenten, die bei aktuellen Arbeiten in diesem Gebiet im Vordergrund steht. Hier trifft einer hoher Bedarf an Rechenleistung, zum Beispiel für ein Navigationssystem, auf verschiedene Automotive-Funktionen im Bereich des *Instrument-Clusters*, so dass sich eine Integration auf der Basis von sicherer Partitionierung in Raum und Zeit anbietet. Diese Konsolidierung von *Automotive*- und *Infotainment*-Funktionen auf einem Gerät profitiert von flexibleren Ressourcenallokationen und einem Anstieg der realisierbaren Funktionskomplexität. Dabei treffen *statisch* konfigurierte Software-Architekturen für automotive Funktionen (AUTOSAR) auf ihre dynamischen agierenden Pendants im Infotainment-Bereich, die ihrerseits oft auf Basis von Linux realisiert sind. Die Anpassung der unterschiedlich etablierten Entwicklungsmethoden, sowie der vorhandenen Entwicklungswerkzeuge stellen neue Herausforderungen für die Zukunft bereit. Dieses Vorgehen ist jedoch nicht nur mit technischen Herausforderungen verbunden, sondern erfordert auch neue "Integrationsprozesse" zwischen unterschiedlichen Entwicklergruppen in langjährig gewachsenen Organisationsstrukturen.

7 Projekt VirtuOS: Virtualisierung im Automobil

Im VirtuOS Projekt [Vir10] untersuchen die Autoren die Einsatzmöglichkeiten von Virtualisierungstechnologien in sicherheitsrelevanten Anwendungsfällen anhand eines konkreten Fallbeispiels aus dem Automobil-Bereich. Angenommen wird die Nutzung des automotiven Betriebssystems *COQOS* auf einem *Instrument-Cluster* zur Integration von Infotainment Applikationen (*"car-apps"*) und sicherheitskritischen Automotive-Funktionen.

Schon allein durch die Öffnung der verwendeten Plattform für die Nutzung durch Anwendungen von Drittanbietern, ergibt sich die Forderung nach einer sicheren Partitionierung, um nicht nur Zuverlässigkeitsaspekte sondern auch Security-Aspekte geeignet zu adressieren. In Hinblick auf den neuen Automotive-Sicherheitsstandard ISO 26262 [Int09], wird in dieser Fallstudie auch die Realisierbarkeit von inkrementeller Zertifzierbarkeit und die Übertragbarkeit von bereits zertifizierten Komponenten aus anderen Domänen, wie zum Beispiel der Avionik, untersucht. Erste Ergebnisse dazu finden sich in [GHW11].

Die Software-Architektur des *Instrument-Clusters* ist in Abbildung 5 dargestellt. Auf einer gemeinsamen Hardware-Plattform werden die verfügbaren Ressourcen mit Hilfe eines

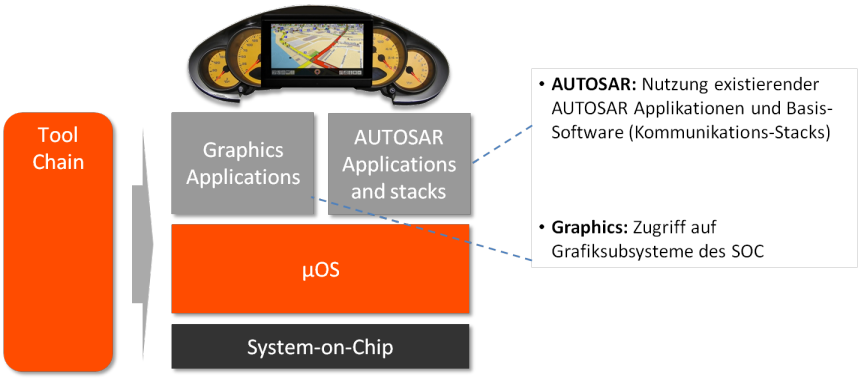


Abbildung 5: Architektur der Fallstudie und wichtige Eigenschaften

Mikrokernels in verschiedene Partitionen unterteilt. Die konfigurierten Partitionen sind dabei voneinander isoliert, so dass Fehler in einer Partitionen nicht die Ausführung von sicherheitskritischen Funktionen beeinflussen können.

Dabei dient das automotive Betriebssystem COQOS (μOS in Abbildung 5) der Firma OpenSynergy als Grundlage zur sicheren Partitionierung des Systems. COQOS adressiert Kernanforderungen aus der Automobilwelt, wie z.B. schnelles Hochstarten ausgewählter Applikationen, Echtzeitfähigkeit und die Integration von AUTOSAR Softwarekomponenten. COQOS basiert auf dem Mikrokern *PikeOS* der Firma SYSGO der bereits erfolgreich für verschiedene Avionik-Systeme zertifiziert worden ist, so dass sich hier die bereits angesprochene Herausforderung hinsichtlich der Übertragbarkeit von Domänen-spezifischen Zertifizierungsartefakten stellt (siehe auch [Ope11, SYS11]).

Weil Mikrokerne nur eine sehr kleinen Implementierungsumfang haben, lassen sie sich besser auf Fehler überprüfen. Deswegen bieten sich Mikrokernel als Teil der Trusted Computing Base (TCB) und damit den Einsatz als sicherer Hypervisor an. Deutlich wird dies unter anderem durch die Verfügbarkeit eines formal verifizierten L4-Mikrokerns[1]. Mit monolithischen Kernen ist dieses Maß an *Korrektheit* aufgrund des Umfangs ihres Quellcodes nur sehr schwer zu erreichen.

Neben den dynamischen Aspekten der Partitionierung zur Laufzeit, wie Scheduling, Isolationsmechanismen oder auch der Umgang mit spezieller Hardware, widmet sich das Projekt auch den statischen Aspekten zur Entwicklungszeit. Zur Steigerung der Zuverlässigkeit und auch als direkte Folge der Forderungen aus der ISO 26262, werden auch statische Analyse-Verfahren hinsichtlich ihres Beitrages zu einer effizienten Steigerung der Software-Qualität untersucht.

[1] siehe `http://ertos.nicta.com.au/research/sel4/`

8 Ein Ausblick: Manycore Prozessoren und Flexible Partitionierung

Bisher wurde deutlich, welche Bedeutung Virtualisierungstechnologien für eine weitere Erhöhung der Funktionsdichte bei Multicore Prozessoren besitzen. Durch eine sichere Partitionierung der Hardware-Ressourcen (CPU Zeit und Speicher) in Raum und Zeit kann ein hohes Maß an Software-Isolation trotz paralleler Rechenkerne sichergestellt werden. Dabei kann die Anzahl der voneinander isolierten Partitionen unabhängig von der Anzahl der Rechenkerne konfiguriert werden.

Beim Blick auf die aktuellen Entwicklungen im Prozessorbereich stellt sich die Frage, inwieweit die bisherigen Lösungsansätze für Multicore Prozessoren auch für Manycore Prozessoren mit mehr als 32 parallelen Rechenkernen eingesetzt werden können. Welche architektonischen Unterschiede in den Prozessoren werden zukünftig zu neuen Anforderungen für Virtualisierungstechnologien führen?

Hilfreich ist dazu ein Blick auf bereits heute erhältliche Manycore Prozessoren wie den Tilera Tile64Pro mit 64 Kernen. Jeder der 64 Rechenkerne ist über ein 6-faches 2D-Mesh Netzwerk - das *Network-On-Chip (NoC)* - mit allen anderen Kernen und auch den I/O Controllern zum Speicher und zur Peripherie verbunden. Innerhalb eines Taktes kann ein *Hop* zu einem benachbarten Kern realisiert werden. Diese Architektur verfügt nicht über *Shared Memory*, sondern basiert auf dem Ansatz von „privaten“ L2 Caches, auf die andere Kerne über das NoC zugreifen können (*Distributed Shared Memory*).

Diese Prozessorarchitektur ist mittlerweile mit bis zu 100 Kernen am Markt erhältlich und zeigt damit deutlich ihre Skalierbarkeit über die Multicore Grenze von 32 Kernen hinaus. Eine Leistungsaufnahme von etwa 25W für 64 Kerne mit 800 MHz unterstreicht auch die Tauglichkeit für eingebettete Systeme. In Abbildung 6 ist zur didaktischen Vereinfachung ein fiktiver 16-Core Prozessor mit einem entsprechenden NoC abgebildet.

Dieser Bruch mit klassischen Multicore Architekturen mit Shared-Memory wirft die Frage auf, welche Auswirkungen sich daraus für die Virtualisierung und Partitionierung ergeben. Bereits durch die Betrachtung des Verhältnisses von Rechenleistung zu Kommunikationsbandbreite wird deutlich, dass Rechenleistung im Gegensatz zu den Kommunikationsverbindungen keine knappe Ressource mehr ist. Manycore Prozessoren beginnen den Wandel vom *Computation-Centric Computing* zum *Communication-Centric Computing*.

Für die Partitionierung bedeutet dies, dass das NoC als inhärent gemeinsam genutzte Ressource identifiziert werden muss und der Zugriff darauf durch geeignete Scheduling-Verfahren (zum Beispiel mittels *Time-Division-Multiplexing (TDM)*) und geeigneten Isolationstechniken (zum Beispiel mit Hilfe einer *NoC-Firewall*) koordiniert werden muss. Dies stellt eine wesentliche Grundvoraussetzung für die Realisierung von *Time and Space Partitioning* auf dieser Architektur dar.

Im Gegensatz zu den bisherigen Prozessorarchitekturen, bei denen der Zugriff auf den Speicher die gleiche Zeit für alle Kerne erfordert, ist dies bei diesen Architekturen nicht gegeben. Mit einer größeren Entfernung zu den I/O Controllern steigt auch die Anzahl der notwendigen Hops und damit auch die Latenz für den Speicherzugriff. Das Zeitverhalten einer Applikation ist damit direkt von dessen Verteilung auf den einzelnen Rechenkernen abhängig. Bei der Zuordnung von Rechenkernen zu einzelnen Partitionen genügt

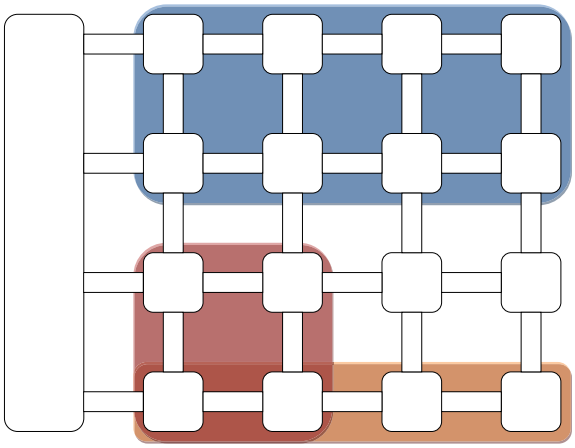


Abbildung 6: Ein (fiktiver) Prozessor mit 16 Kernen, die jeweils über ein Network-On-Chip (NoC) miteinander verbunden sind. I/O Anschlüsse befinden sich auf der linken Seite und können über das NoC angesprochen werden. Auf dem Prozessor sind als Beispiel drei Partitionen eingezeichnet.

es deshalb nicht mehr, lediglich kapazitive Angaben zu machen. Stattdessen müssen die verwendeten Cores genau identifiziert werden, so dass das Zeitverhalten nicht beeinflusst wird.

Eine weitere Herausforderung für Virtualisierungstechnologien, die sich aus der Verwendung von Manycore Prozessoren ergeben wird, ist die optimierte Nutzung der Ressourcen zur Laufzeit. Mit einer höheren Funktionsdichte, sowie einer weiteren Integration von sicherheitskritischen und unkritischen Applikationen auf der gleichen Plattform, wird sich dies auch in unterschiedlichen Partitionstypen niederschlagen.

Feste Partitionen definieren dann beispielsweise eine feste Zuordnung von Cores und NoC-Bandbreite, wohingegen *flexible Partitionen* je nach Auslastung des Prozessors auch ihre Ressourcenzuordnung variieren können. Denkbar ist zum Beispiel mit Blick auf Abbildung 6 die temporäre Nutzung nicht (mehr) benötigter Rechenzeit anderer Partitionen, so dass eine Partition ihre Form ändert (*Re-Shaping*). Im Falle von Fehlern könnte eine Partition auch innerhalb des Prozessors „umziehen“ (*Re-Locating*) und damit die Nutzung fehlerhafter Kerne oder NoC-Links vermeiden.

Es bleibt damit die wichtige Aufgabe für die angewandte Forschung im Bereich Virtualisierung für eingebettete Systeme, effiziente Modellierungsverfahren zu entwickeln, mit denen die genauen Anforderungen einzelner Partitionen spezifiziert werden können, so dass anschließend *automatisch korrekte Mappings*, *Schedules* für das Betriebssystem, sowie Konfigurationsinformationen für die verwendete Virtualisierungsschicht *generiert* werden können. Manuell ist dieser Prozess für zukünftige Prozessoren mit 1000 Kernen nicht mehr effizient durchführbar.

Acknowledgements

Teile dieser Arbeit wurden im Rahmen des VirtuOS Projektes erarbeitet. Das VirtuOS Projekt wird durch die TSB Technologiestiftung Berlin aus Mitteln des Zukunftsfonds des Landes Berlin gefördert, kofinanziert von der Europäischen Union - Europäischer Fonds für Regionale Entwicklung. Investition in Ihre Zukunft!

Literatur

[ARI05] ARINC. ARINC Specification 653P1-2: Avionics Application Software Standard Interface Part 1 - Required Services, Dezember 2005.

[Bor07] S. Borkar. Thousand Core Chips - A Technology Perspective. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, Seiten 746 –749, jun. 2007.

[GHW11] Matthias Gerlach, Robert Hilbrich und Stephan Weißleder. Can Cars Fly? From Avionics to Automotive:Comparability of Domain Specifc Safety Standards. In *Proceedings of Embedded World 2011*, Nuremberg, Germany, March 2011.

[HG11] Robert Hilbrich und Hans-Joachim Goltz. Model-based Generation of Static Schedules for Safety Critical Multi-Core Systems in the Avionics Domain. In *Proceedings of the 4th International Workshop on Multicore Software Engineering*, IWMSE '11, 2011.

[Int09] International Standards Organization. ISO DIS 26262: Functional Safety for Road Vehicles. Draft international standard, International Standards Organization, 2009.

[Kai09] Robert Kaiser. *Virtualisierung von Mehrprozessorsystemen mit Echtzeitanwendungen*. Dissertation, Universität Koblenz-Landau, June 2009.

[Ope11] OpenSynergy. www.opensynergy.com, 2011.

[Ous80] John Kenneth Ousterhout. *Partitioning and Cooperation in a Distributed Multiprocessor Operating System: Medusa*. Dissertation, Carnegie-Mellon University, Pittsburgh, PA, USA, 1980. AAI8020166.

[RG05] Mendel Rosenblum und Tal Garfinkel. Virtual Machine Monitors: Current Technology and Future Trends. *Computer*, 38:39–47, May 2005.

[Ros04] Robert Rose. Survey of System Virtualization Techniques. Bericht, Oregon State University (OSU), 2004.

[RTC94] RTCA Inc. DO-178B: Software Considerations in Airborne Systems and Equipment Certification. Bericht DO-178B, RTCA Inc., Washington, USA, 1994.

[Rus99] John Rushby. Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance, 1999.

[SYS11] SYSGO. www.sysgo.com, 2011.

[Vir10] VirtuOS Project Summary. http://www.technologiestiftung-berlin.de/data/files/tsb-zukunftsfonds/VirtuOS__Web_.pdf, 2010.

[Weu11] DSP-Weuffen GmbH. http://www.dsp-weuffen.de/de/automotive.html, 2011.