© Blue Planet Studio / Adobestock

Verschiedene elektronische Domänen auf einer Plattform zusammenführen

Sichere Virtualisierung

Die zentrale Herausforderung moderner Automobilanwendungen besteht darin, die erforderliche Hard- und Software so zu konsolidieren, dass die Rückwirkungsfreiheit kritischer Funktionen gewährleistet ist und gleichzeitig eine hohe Echtzeitleistung, niedrige Kosten und Design-Flexibilität für zukünftige Anpassungen geboten werden.

Carmelo Loiacono

Virtualisierungstechnologie ist allgemein bekannt und für High-End-Cores wie den Arm Cortex v8-A verfügbar. Die Unterstützung mehrerer Betriebssysteme, die auf derselben Hardware laufen, bringt Herausforderungen mit sich, wie den Schutz von Ressourcen vor Missbrauch und die Gewährleistung einer entsprechenden Bandbreite für alle. Diese Probleme werden bereits mit Prozessortechniken gelöst, die Hardware-Virtualisierung unterstützen. Kürzlich wurden Hardware-Virtualisierungsfunktionen auch in kleine Mikrocontroller-Kerne wie den Arm Cortex v8-R integriert. Die Isolierung und gemeinsame Nutzung von Ressourcen durch mehrere Betriebssysteme, die auf einem physischen System laufen, sind ebenfalls Bedingungen für einfache und kostengünstige Embedded-Systeme.

Die Wahl der spezifischen Virtualisierungstechniken hängt streng vom Prozessortyp, dem Hersteller und der Prozessor-Hardware ab. Eine flexible Hypervisor-Architektur ist erforderlich, um die in heutigen Mikrocontrollern, wie dem NXP S32Z, S32E und ST Stellar SR6, verfügbaren Hardware-Virtualisierungsfunktionen optimal zu nutzen.

Unterstützung für Virtualisierung

Der folgende Abschnitt behandelt die gängigsten Virtualisierungsfunktionen auf Hardware-Ebene für Arm-Architekturen.

Ausnahmestufen

Die Arm v8-Architekturen assoziieren Exception Levels (EL) mit Software-Ausführungsprivilegien und definieren einen Satz von vier Exception Levels – EL0 bis EL3 – wobei EL0 die niedrigste und EL3 die höchste privilegierte Ausführungsebene ist.

Die Armv8-R-Architektur führt Funktionen ein, die es Entwicklern ermöglichen, Hochleistungsanwendungen für sicherheitskritische Umgebungen zu entwerfen und zu implementieren. Dazu gehören:

- keine überlappenden Speicherbereiche,
- neues Exceptionmodell, das mit dem Modell Armv8-A kompatibel ist,
- Virtualisierung mit Unterstützung für

Gastbetriebssysteme.
- Optional: Unterstützung für Double-Precision Floating-Point und Advanced SIMD.

Außerdem können mehrere Protected-Memory-System-Architecture-Kontexte (PMSA) auf demselben Kern ausgeführt werden, indem sie mit Hilfe von Virtualisierungstechnologie eingegrenzt werden. Moderne Mikrocontroller ermöglichen es, die Echtzeitleistung verschiedener Kontexte einzuschließen, wodurch verhindert wird, dass ein Kontext die Reaktionszeit und den Determinismus eines kritischeren Kontexts beeinträchtigt. Einige verfügen auch über redundante Kopien der Logik- und Komparatorinstanzen für den Dual-Core Lock-Step-Betrieb (DCLS).

Speicherschutz: MMU und MPU

Eine MMU ist eine „Speicherverwaltungseinheit", während eine MPU eine „Speicherschutzeinheit" ist. Beide sind spezialisierte Hardware, die von der CPU für die Speicherverwaltung verwendet wird. Die MMU wird für viele Funktionen verwendet, darunter in erster Linie für den virtuellen Speicher, also die Übersetzung der virtuellen Adresse in die physische Adresse und den Speicherschutz. Die MPU hingegen wird ausschließlich für den Speicherschutz verwendet. In diesem Sinne kann man sich die MMU als Obermenge einer MPU vorstellen. Je nach Komplexität des Prozessors oder SoCs können MMUs/MPUs an verschiedenen Stellen innerhalb der Architektur platziert werden, beispielsweise in Multi-

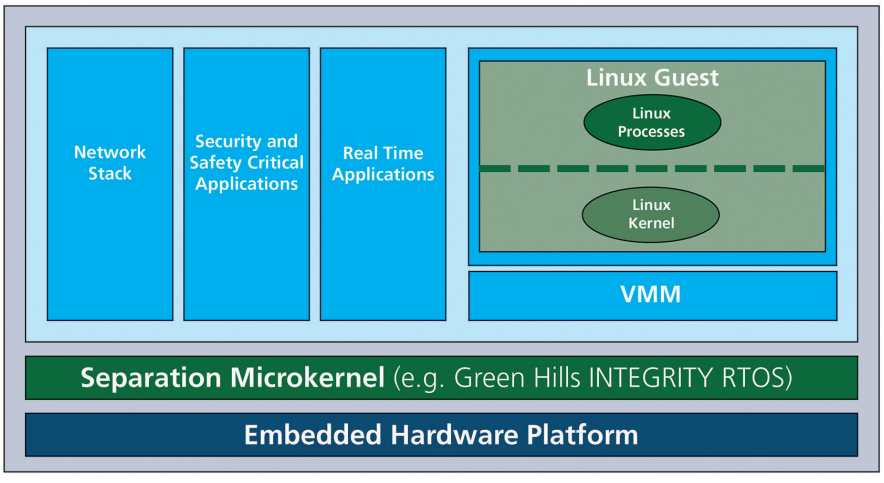

Bild 1: Der Separation Microkernel Hosted Hypervisor stellt sicher, dass sich Fehler innerhalb eines Prozesses nicht auf das gesamte System ausbreiten, indem der Prozessspeicher auf einen bestimmten Speicherbereich begrenzt wird. © Green Hills Software

Core-SoCs lokal für jeden Kern oder auf einer gemeinsamen Ebene.

Unterstützung der E/A-Virtualisierung

Ähnlich wie eine MMU steuert eine E/A-Speicherverwaltungseinheit (IOMMU) den Zugriff von E/A-Geräten auf den Hauptspeicher, auch als direkter Speicherzugriff (DMA) bezeichnet, und bietet Adressübersetzungsmechanismen und eine programmatische Schnittstelle zur Festlegung der Adressbereiche, auf die das Gerät zugreifen kann. In modernen Mikrocontrollern mit Virtualisierungsunterstützung wurde eine IOMPU oder Peripherie-MPU hinzugefügt, um den Zugriff auf E/A-Geräte vor den Gastbetriebssystemen zu schützen. Sie ähnelt der IOMMU, wurde aber speziell für die Mikrocontroller-Architektur entwickelt.

Virtualisierungstechniken

Bei der Verwendung von Anwendungsprozessoren wie Arm v8-A bietet die Virtualisierung in Kombination mit einem Separations-Kernel erweiterte Funktionen für Software-Entwickler, die sicherstellen müssen, dass die heterogenen Software-Komponenten frei von Interferenzen sind, der Informationsfluss geschützt ist und das Kommunikationssystem im Hinblick auf Sicherheitsaspekte verstärkt wird.

Ein Separationskernel besteht aus Partitionen oder virtuellen Adressräumen (VAS), die vom Kernel erstellt und über die MMU des Systems implementiert wurden. Innerhalb einer Partition ist die Software-Trennung nicht garantiert: Wenn der Code fehlerhaft ist oder kompromittiert wird, kann er weder etwas außerhalb der Partition beeinflussen noch den Betrieb oder das Verhalten anderer Partitionen oder des Separations-Kernel selbst ändern.

Virtuelle Adressräume sind physisch nicht vorhanden. Stattdessen werden ihre Adressen auf Adressen im Kernel Space abgebildet. Wenn sie ausgeführt werden, werden alle ihre Speicherzugriffe, ob Codeabrufe oder Datenzugriffe, von der MMU in physische Zugriffe übersetzt. In **Bild 1** befindet sich der Virtual Machine Monitor (VMM) nicht im privilegierten Modus, sondern wird als Anwendung auf dem Separations-Kernel ausgeführt, von wo aus er das nicht vertrauenswürdige Gastbetriebssystem innerhalb der Partition verwaltet. Der VMM sorgt für den Schutz der Speicherressourcen und die sichere Zugriffskontrolle für E/A und andere Systemobjekte. Zudem verteilt er die Arbeitslasten sicher und effizient auf die Kerne.

Der Gast-Kernel läuft im EL1-Modus und kümmert sich um die Hardware-Initialisierung, seine Gerätetreiber, die Planung von Gastprozessen, Interrupts und die Speicherverwaltung. Aus diesem Grund benötigt der Gast-Kernel Zugriffe im privilegierten Modus, die durch den VMM gemildert werden. Aufgrund der begrenzten Ressourcen von Mikrocontrollern ist eine leichtgewichtige Virtualisierung erforderlich. Bei dem implementierten Hypervisor handelt es sich in der Regel um einen Typ 1, der grundlegende Operationen wie die Konfiguration des Systems zur Aktivierung der Virtualisierung, das Einschalten und Zurücksetzen der Kerne, die Konfiguration der Taktgeber, die Einrichtung der MPU und das Zurücksetzen des Gastkontexts durchführt. Nach dieser Initialisierung überträgt der Hypervisor die Ausführung in den Gastmodus innerhalb des Gastbetriebssystems. Der Hypervisor befindet sich im Leerlauf und wartet auf Unterbrechungen oder Service-Anfragen von den Gästen.

Wie der Anwendungsprozessor-Hypervisor muss auch der Mikrocontroller-Hypervisor über eine robuste, durch Hardware erzwungene Software-Trennung, Unterstützung mehrerer Betriebssysteme und Echtzeit-Effizienz verfügen, um kritische Arbeitslasten sicher zu konsolidieren. Die meisten Anwendungen benötigen eine effiziente Architektur, die die Störungsfreiheit mehrerer Betriebssysteme, die auf derselben CPU laufen, gewährleistet und flexible Optionen zur vollständigen Nutzung mehrerer Kerne und begrenzter Prozessorressourcen bietet. **Bild 2** zeigt ein Blockdiagramm eines Typ-1-Hypervisors mit mehreren verschiedenen Gastbetriebssystemen, die auf einem Mikrocontroller laufen. Diese Art der Virtualisierung lässt sich je nach den Anforderungen der Anwendung auch für High-End-SoCs mit Arm v8-A-Kernen nutzen.

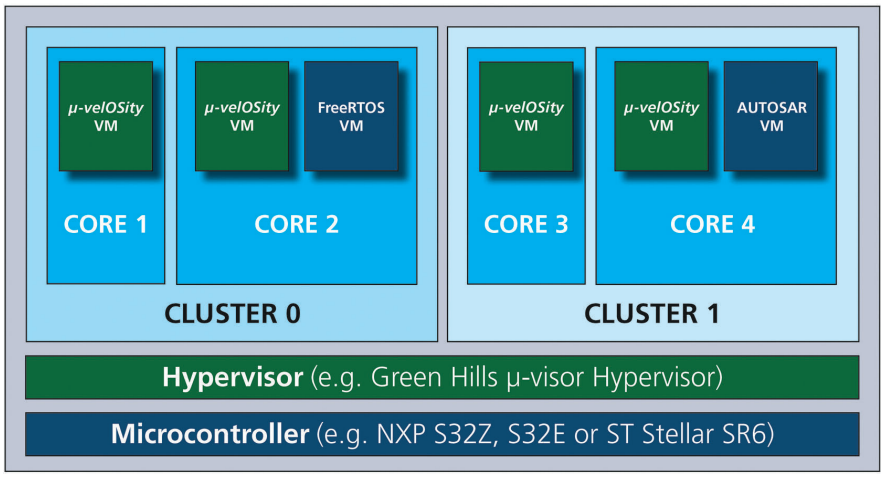

Bild 2: Ein Typ-1-Hypervisor, wie der µ-visor Hypervisor von Green Hills Software, mit mehreren virtualisierten Betriebssystemen, beispielsweise µ-velOSity RTOS, FreeRTOS und AUTOSAR © Green Hills Software

Treibermodell

In einem virtualisierten System ist es sehr wichtig zu definieren, welche Gäste Zugriff auf ein bestimmtes Peripheriegerät haben und ob ein Peripheriegerät nur einem Gast zugewiesen oder gemeinsam genutzt werden soll. Der nachfolgende Abschnitt beschreibt verschiedene Techniken beschrieben, die von der I/O-Virtualisierungsunterstützung moderner Prozessoren abhängen.

Direkter Zugriff

Der Gast hat vollen Zugriff auf E/A, einschließlich DMA. Das ermöglicht eine maximale Wiederverwendung vorhandener Gastbetriebssystemtreiber

ohne Leistungseinbußen. Peripheriegeräte, die auf Speicher zugreifen können, wie DMA, die nicht durch die MPU und MMU eingeschränkt sind, könnten jedoch für den Zugriff auf geschützte Speicherbereiche verwendet werden, wenn diese offenliegen. Eine Kernel-Schwachstelle kann E/A nutzen, um die Hypervisor-Isolierung vollständig zu umgehen und sogar den Hypervisor selbst zu untergraben. Dieses Problem lässt sich mit einer IOMMU und IOMPU – auch bekannt als Peripherie MPU – lösen.

Emulation

Das Gerät wird emuliert und alle E/A-Zugriffe werden vom Hypervisor verwaltet. Das emulierte Gerät kann sich von dem physischen Gerät unterscheiden. Bei dieser Methode werden emulierte Geräte über vertrauenswürdige Hypervisor-Treiber gemultiplext. Obwohl diese Methode eine sichere gemeinsame Nutzung von Geräten durch Gäste ermöglicht, ist der Leistungsverlust in der Regel sehr hoch, insbesondere bei Low-End-Prozessoren mit begrenzten Ressourcen.

Paravirtualisierung

Die Paravirtualisierung bietet speziell definierte „Hooks", die es dem Gast oder den Gästen ermöglichen, Aufgaben anzufordern und zu bestätigen, die andernfalls in der virtuellen Domäne ausgeführt würden – wo die Ausführungsgeschwindigkeit geringer ist. Eine erfolgreiche paravirtualisierte Plattform kann es ermöglichen, den VMM zu vereinfachen, indem die Ausführung kritischer Aufgaben von der virtuellen Domäne in die Host-Domäne verlagert wird und/oder die allgemeine Leistungsverschlechterung der Maschinenausführung innerhalb des virtuellen Gastes zu verringern. Bei Prozessoren wie Arm v8-A und Arm v8-R ist das die gängigste Technik zur gemeinsamen Nutzung von Geräten zwischen Gästen.

Schlussfolgerungen

Während viele Entwickler mit der Virtualisierung auf High-End-Cores wie dem Arm Cortex v8-A vertraut sind, ermöglichen Hochleistungs-Multicore-Mikrocontroller wie NXP S32Z, S32E und ST Stellar SR6, die mit dedizierten Virtualisierungs-Hardware-Funktionen ausgestattet sind, eine neue Klasse der Mikrocontroller-Konsolidierung. Um die Wiederverwendung von Software zu fördern und aufgrund der Vielfalt moderner SoCs muss eine Virtualisierungsplattform flexibel und konfigurierbar sein, um den spezifischen Anwendungsanforderungen gerecht zu werden. Man sollte daran denken, dass je nach Anwendung und verwendetem Prozessor verschiedene Virtualisierungstechniken sorgfältig evaluiert und implementiert werden müssen, um die erforderliche Effizienz und Leistung sowie eine sichere Virtualisierung für die Anwendung zu gewährleisten. ■ (eck)

www.ghs.com

Carmelo Loiacono ist Field Applications Engineer bei Green Hills Software. © Green Hills Software