Embedded Software Engineering Kongress

ESE Kongress 2015

Vortragsskript:

Ein Multicore-Referendum
Die Qual der Wahl oder nur die Wahl der Qual?

Dipl.-Ing. (TU) Marcus Gößler, MicroConsult GmbH

Die Auswahl des richtigen Mikrocontrollers bleibt eine Herausforderung für jedes Unternehmen. Das Aufkommen von Multicore-Architekturen in Embedded-Systemen bringt zusätzliche Aspekte für die Auswahl-Betrachtung mit sich und erfordert eine Neubewertung etablierter Kriterien. Worauf ist hinsichtlich verfügbarer Multicore-Ansätze besonders zu achten, welche Lösungen existieren und wie beeinflussen sie die Entwicklungskette?

Etablierte Auswahlprozesse und -kriterien

Sieht man sich traditionelle Auswahlkriterien für Singlecore-Controller an, wird schnell klar, dass dieselben Kriterien eine Lösung basierend auf Multicore-Systemen erstrebenswert machen. Sowohl im industriellen Umfeld als auch in der Automobilindustrie sehen sich Ingenieure mit den typischen Forderungen nach höherer Integrationsdichte, geringerer Leistungsaufnahme und zusätzlichen Funktionen bei gleichzeitig gesenkten Herstellungskosten konfrontiert. Neuere Anforderungen im Bereich Wartung und autonomes Fahren zusammen mit der Applikation von Normen funktionaler Sicherheit gemäß ISO 26262 oder IEC 61508 geben den Multicore-Ansätzen entsprechend Vorschub, sind sie doch angetreten, genau diese Aspekte kostengünstig zu adressieren. Es gilt, unterschiedliche Perspektiven im Entscheidungsprozess abzubilden bzw. über Gewichtung der verschiedenen Faktoren für eine ausgewogene Auswahl Sorge zu tragen.

Typische Ausgangslage für die Neuentwicklung:

- Forderung nach allgemein höherer Performanz
- Senkung der Stromaufnahme
- Gesteigerte Sicherheit (funktional und Intellectual Property)
- Einbindung neuer Funktionen
- Bauteilreduktion
- Schnellere Produkteinführung
- Größen- und Gewichtsreduktion
- Kostensenkung allgemein

Adaption auf Multicore

Aus Sicht der Chiphersteller stellen Multicore-Controller eine kosteneffiziente und relativ leicht zu realisierende Option hinsichtlich der Hardware dar, diese treibenden Faktoren zu adressieren. Ein Paradigmenwechsel von Singlecore auf Multicore bringt allerdings neue Herausforderungen und demnach Risiken mit sich, die sich vermehrt in Richtung System-Design und Software verlagern.

Es ist daher angebracht, relevante Betrachtungen auch verstärkt in den Auswahlprozess einfließen zu lassen sowie plakative Versprechen auf den Prüfstand zu stellen und mit harten realen Bemessungskriterien zu belegen. Nur auf diese Art und Weise kann eine Auswahl, die schlussendlich auf einem Vergleich von Optionen basiert, faktengetreu durchgeführt werden.

Propagierte Multicore-Vorteile

- · Rechnerische Vervielfachung der Rechenleistung
- · Minimale Latenzen
- · Höhere Integration
- · Geringere Boardfläche
- · Niedrigerer relativer Stromverbrauch
- · Höhere Taktraten bei on-chip Signalen
- · Einfachere Kern-zu-Kern-Kommunikation
- · Bessere Kosteneffizienz
- · Skalierbarkeit

Neue Betrachtungswinkel und Aspekte

Bei der Auswahl einer geeigneten Multicore-Mikrocontrollerfamilie ist neben der typischen Vorgehensweise und den üblichen Auswahlkriterien besonderes Augenmerk auf den aktuellen "Ist"-Zustand zu legen:

- Treffen einer Grundsatzentscheidung
- Portierung (auch graduell)
- Bedingt unter Umständen neue Hardware- und Software-Architektur

Die Aufnahme des „Ist-Zustandes“

- Aktuell verwendete Plattform (z.B. 8-/16-/32-Bit Mikrocontroller-Typ, DSP, etc.) Erfahrungen mit dem Support der Anbieter
- Rechenleistung der aktuellen Architektur (MIPs, zusätzlich sind Aspekte wie Floating-Point Arithmetik wichtig, Datendurchsatz, etc.)
- Aktuelle Programmgröße und Speichertyp
- Kommunikationsplattformen und zu übertragende Datenmengen
- Zu generierende Signaltypen und Sensor-Signalauswertungen
- Interrupt-Reaktionszeiten
- Verwendete Tools/Tool-Ketten für die Software-Entwicklung und den Test
- Realzeit-Betriebssystem (RTOS) bzw. Scheduler
- Kostenaspekte

Die Auswahl der Multicore-Plattform hat anschließend basierend auf der Anforderungsanalyse zu erfolgen. Durch einen prozessorientierten Vorgang bleibt sichergestellt, dass alle ausschlaggebenden Anforderungen in die Entscheidung einfließen und nachvollziehbare, begründete Entscheidungen getroffen werden können. Der Umstieg auf Multicore-Derivate bringt zwangsläufig neue Herausforderungen mit sich und erhöht somit auch das Risiko von Fehl-und Rückschlägen. Es ist daher essentiell, sich gegenüber diesen neuen Herausforderungen entsprechend aufzustellen und Risiken abzuschätzen bzw. zu minimieren.

Neue Herausforderungen entstehen durch:

- Unbekannte Hardware-Plattform bzw. -Architektur
- Notwendigkeit einer neuen Software-Architektur (auch unter dem Gesichtspunkt der Skalierungsstrategie)
- Portierung des aktuell bestehenden Systems
- Einbezug neuer oder geänderter Projektanforderungen
- Festlegung der Portierungsstrategie und entsprechender Ziele
- Neue Tools/Entwicklungsumgebungen
- Geändertes Boot-/Startverhalten
- Nötige Schutzmechanismen
- Testanforderungen
- Trainingsbedarf (Single-Thread-Denken --> Multicore-Verarbeitung)
- Verstärkt zukunftsorientiertes Denken
- Risikobewertung und Minimierung

Tools

Bei Multicore-Systemen stellt sich die Frage nach der Steuerung von Zugriffen auf geteilte Ressourcen. Bekannte Mechanismen von Singlecore-Systemen greifen nur bedingt, so dass andere Mechanismen wie Atomic-Instruktionen, bei denen auf Maschinenebene eine Unterbrechungsfreiheit von Lese-Verändern-Schreib-Befehlen erreicht wird, herangezogen werden müssen, die wiederum entsprechende Toolunterstützung unabdingbar machen. Teilweise besteht sogar die Möglichkeit, auf Hardware zurückzugreifen, die explizit Objekte wie Semaphore oder Mutexes abbilden. Zwischen Compiler und Linker/Locator besteht bei eingebetteten Systemen eine enge Kopplung. Gerade bei Multicore-Systemen ist es wünschenswert, über einfache Schlüsselwörter das Lokatieren von Variablen oder Codeteilen zu gestalten.

Dafür stellen manche Toolhersteller eigene Schlüsselwörter zur Verfügung:

__privateN	// CoreN RAM but accessible by other cores
__share	// shared/global RAM
__clone	// cloned but non-shared RAM of cores
extern	// other core(s) use the same name and location

Der Möglichkeit des Tracens von Code und Daten kommt bei Multicore-Systemen erhöhtes Interesse zu. Zum einen, weil natürlich ein Vielfaches an Berechnungen pro Zeiteinheit durchgeführt wird (und das parallel), und zum anderen, weil die notwendige Datenrate ebenfalls um ein Vielfaches ansteigt. Zu diesem Zweck haben einige Chiphersteller neue bzw. zusätzliche Schnittstellen geschaffen, um Trace-Daten mit hoher Bandbreite zu übertragen. Dies kann von Chip-Varianten abhängen ebenso wie das Vorhandensein von zusätzlichem internen Trace-Speicher zum Zwischenpuffern von Trace-Daten.

Startup

In Master-Slave-Implementationen startet typischerweise zuerst das Master Device, und dieses startet anschließend erst die Slave Devices. Ein Bootloader, der auf dem Master ausgeführt wird, lädt dabei den Speicher des Slave und gibt anschließend den Reset frei.

Andere Bootmöglichkeiten orientieren sich ebenfalls an einem Master-Slave-Konzept, laden allerdings den Gesamtspeicher des Systems bereits zu Beginn. Anschließend können die Program Counter der anderen Kerne vom Master initialisiert werden. Als letzten Vorgang gibt der Master die Slave-Kerne über spezielle Wakeup-Befehle oder Initialisierungen zur Ausführung von Programmcode frei.

Safety

Unter Umständen ist eine strenge Trennung bzw. Aufteilung von Safety-relevanten und nicht-relevanten Applikationseinheiten gefordert. Mit entsprechender Partitionierung kann dies auf Multicore-Plattformen effizient erreicht werden, sofern dies von der Architektur unterstützt wird. Sogenannte Lock-Step-Kerne führen mit einer Verzögerung von wenigen Taktzyklen dieselben Befehle aus wie auf dem "normalen" Kern. Bei Differenz der Resultate wird ein entsprechendes Alarmsignal generiert und entsprechend darauf reagiert.

Von gestiegener Wichtigkeit ist in solchen Systemen auch der Zugriffschutz auf die gemeinsamen Ressourcen. Als eine Möglichkeit haben Hersteller dazu den unterschiedlichen Kernen und anderen zugriffsberechtigten Teilnehmern (z.B. DMA) Identifikationsnummern zugeteilt, über die Zugriffe gesteuert werden.

Auch Speicherzugriffe fallen in diese Klasse. Allerdings sind diese so wichtig, dass häufig eigene Schutzeinheiten dazu implementiert sind.

Performance-Aspekte

- Grad der möglichen Parallelisierung
- Teilen von Ressourcen

Der Grad der Parallelisierung spiegelt sich direkt im Leistungszuwachs wider. Kann die geforderte Applikation nicht entsprechend auf mehrere Kerne aufgeteilt werden, wird es zu ungleichen Verteilungen kommen, in denen das Rechenpotential der Kerne nicht ausgenutzt wird.

In vielen Applikationen wird es ebenfalls nicht möglich sein, ein Teilen von Ressourcen (Speicher, Peripherie, interne Bussysteme, etc.) zu vermeiden. Um Teilen zur Laufzeit zu ermöglichen, müssen Synchronisationsmechanismen (Semaphore, Spinlocks) und Kommunikationsschnittstellen implementiert sein, deren Ausführung natürlich wiederum Rechenzeit kostet.

Zwei prinzipiell unterschiedliche Speicherarchitekturen haben abhängig von der Applikation und Software-Architektur ebenfalls großen Einfluss auf die Performance:

- · Große eng gekoppelte Speicher
- · Kleinere eng gekoppelte Speicher

Autor

Dipl.-Ing. (TU) Marcus Gößler schloss das Studium der Elektrotechnik an der Technischen Universität Graz ab. Seine berufliche Laufbahn begann als Field Application Engineer für analoge und digitale Produkte im Bereich Luft-und Raumfahrt. Weitere Applikationsfelder umfassten Audio/Video, portable Systeme und Infotainment im Automobil. Er leitete Applikationsorganisationen in Zentral- und Osteuropa und zeichnete Verantwortung für große Halbleiterhersteller im Vertriebskanal und Marketing. Bei MicroConsult ist er heute als Trainer und Coach im Bereich Embedded Systems tätig, mit Schwerpunkten in sicherheitsrelevanten Anwendungen und Multicore-Bausteinen.

Kontakt

Internet: www.microconsult.de
E-Mail: m.goessler@microconsult.de