Entwurf und Implementierung einer Prozessinterkommunikation für Multi-Core CPUs

Workshop Echtzeit 2013

Manuel Strobel

[1]Hochschule Furtwangen University
Fakultät CEE

[2]Embedded Office GmbH & Co. KG

21. November 2013

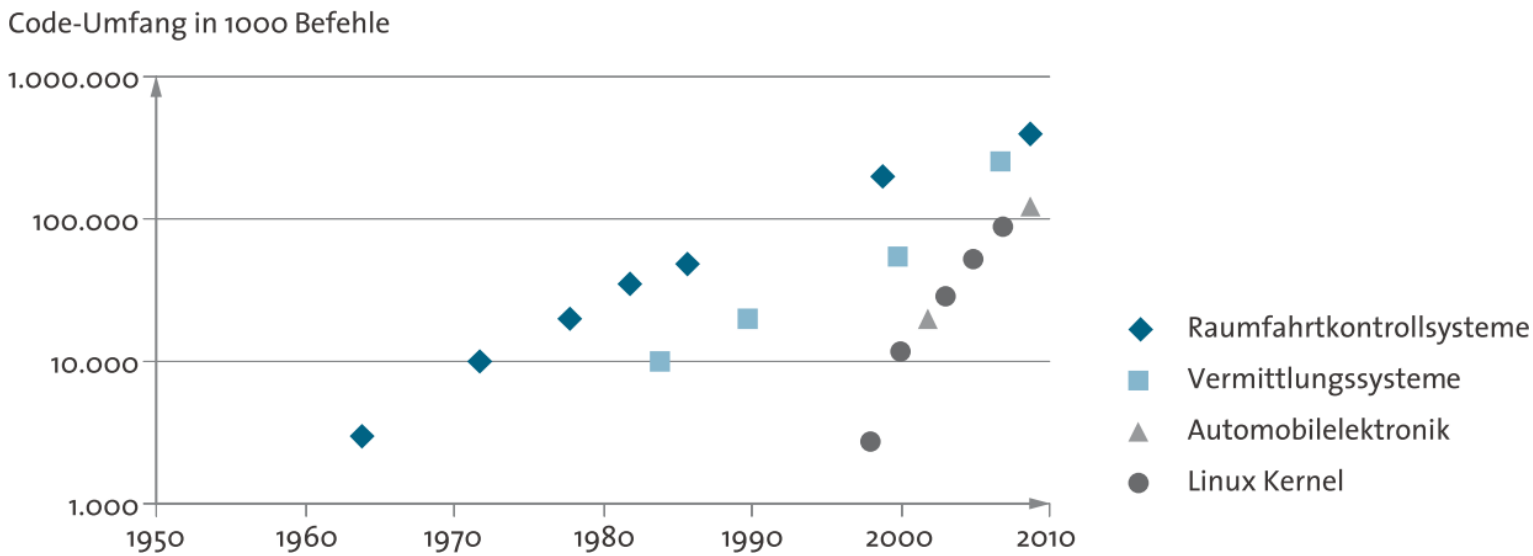

Abbildung: Komplexität eingebetteter Systeme und ihre Entwicklung (BIT10, S. 7)

- Komplexere Systeme erfordern mehr Rechenleistung.
- Trend hin zur Parallelisierung auf Prozessorebene.
- Verfügbarkeit von Multi-Core Systemen im Embedded Bereich.

- Ein Kernel mit Echtzeiteigenschaften (RTOS).
- Ermöglicht die parallele Bearbeitung mehrerer Aufgaben (Multitasking).

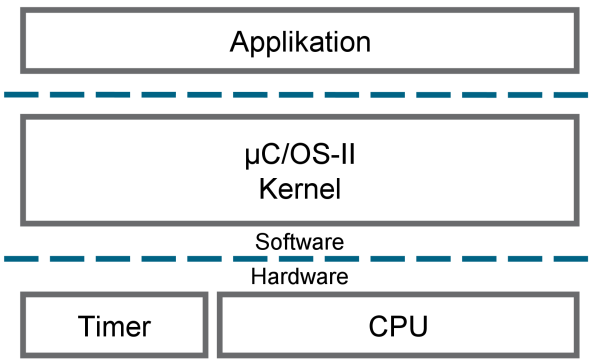

Abbildung: µC/OS-II
Hardware/Software Architektur
(nach Lab02, S. 288)

- Begrenzte Anzahl an Tasks.
- Jeder Task hat eine eindeutige Priorität.
- Zur Kommunikation zwischen Tasks stehen Mailboxes, Queues, Mutexe, Semaphore und Flags zur Verfügung.

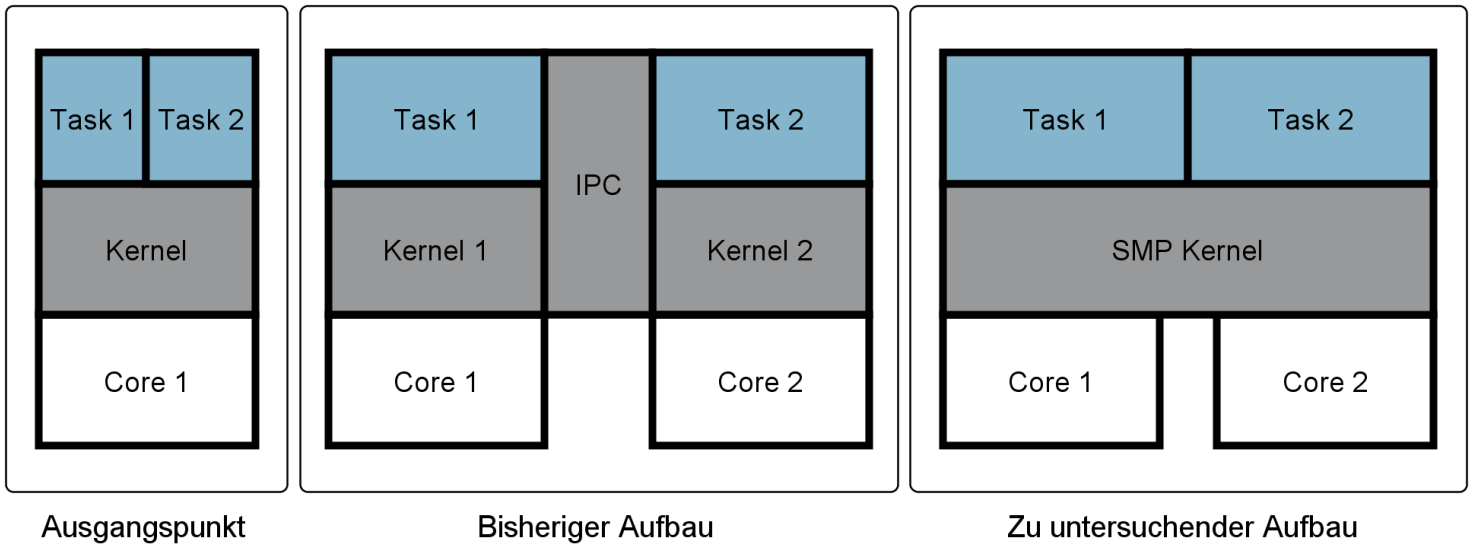

Abbildung: Varianten für den Systemaufbau im Vergleich

- Ausgangsversion unterstützt lediglich einen Prozessorkern.
- Die bestehende Multi-Core Lösung setzt auf ein asymmetrisches Konzept.
- Ziel dieser Arbeit ist ein **symmetrisches Multiprocessing.**

„Multiprozessor-Betriebssysteme unterscheiden sich nur in Sachen Prozesssynchronisation, Ressourcenverwaltung und Scheduling von gewöhnlichen Betriebssystemen“ (Moderne Betriebssysteme - Tanenbaum, S. 615).

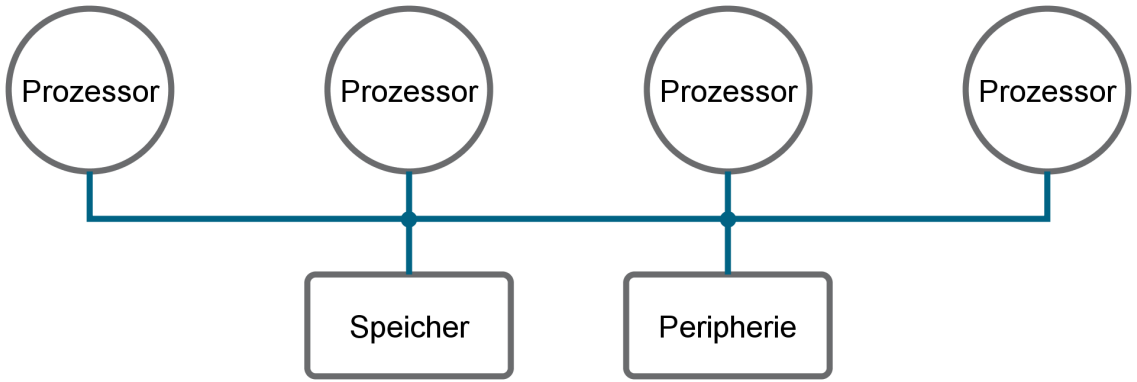

Abbildung: Basisstruktur eines Multiprozessors mit gemeinsamem Speicher (nach HPAD07, S. 200)

- Speicher und Peripherie werden gemeinsam genutzt.
- Durch zeitgleiche Zugriffe können Konflikte und Inkonsistenzen entstehen.
- Es werden folglich entsprechende Schutzmechanismen und Regeln benötigt.

- Der Scheduler ist zentraler Teil eines Betriebssystems.
- Weist der CPU abwechselnd rechenbereite Prozesse zu.
- Ziel ist es, die Prozessorzeit bestmöglich auszunutzen.
- Die bei µC/OS-II eingesetzte Schedulingstrategie basiert auf Prioritäten.

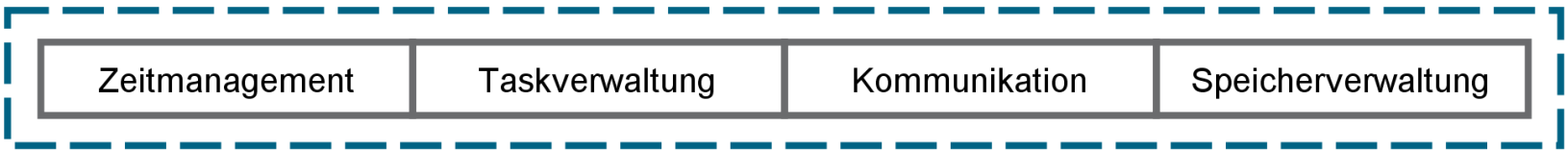

Abbildung: Einteilung der zu schützenden Datenstrukturen

- Aufteilung des Sourcecode in Abschnitte auf Basis der darin verwendeten Variablen.
- Jeder kritische Abschnitt muss durch ein Protokoll geschützt werden, das systemweit gültig ist.

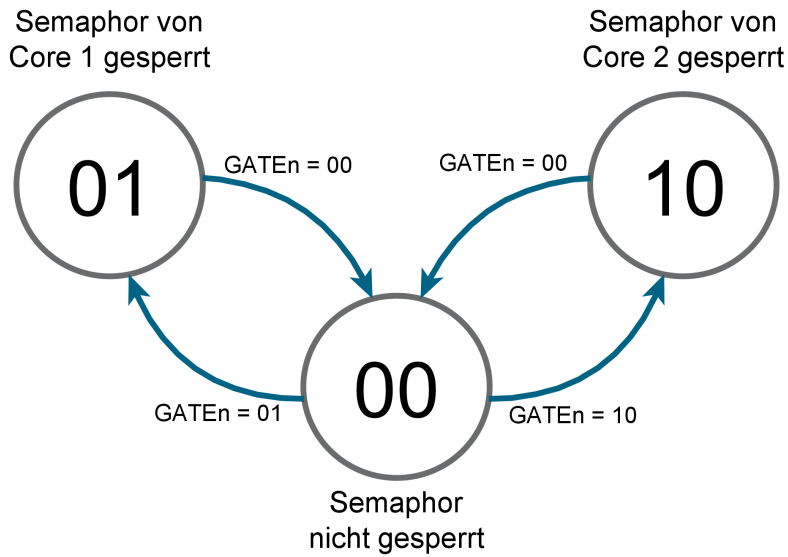

Abbildung: Semaphor Zustandsautomat (nach McK08, S. 8)

- Schutz auf Basis der vorgestellten Einteilung.
- Mechanismus der von allen Prozessorkernen gleichermaßen respektiert wird.

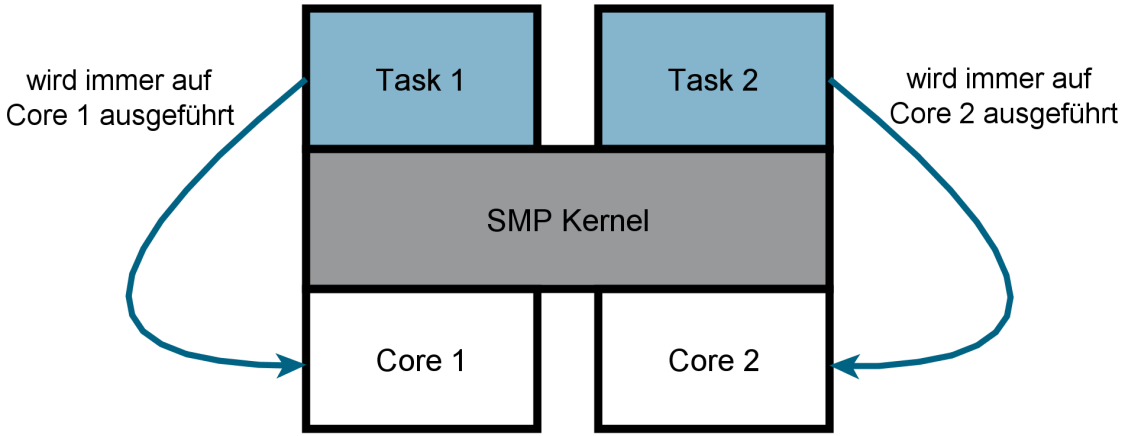

Abbildung: Zuordnung zwischen Tasks und Prozessor

- Zuteilung einzelner Tasks zu einem festen, zugehörigen Prozessor (Core Affinity).

- Welcher Task und somit welcher Prozessorkern sind von dem eingetretenen Ereignis betroffen?

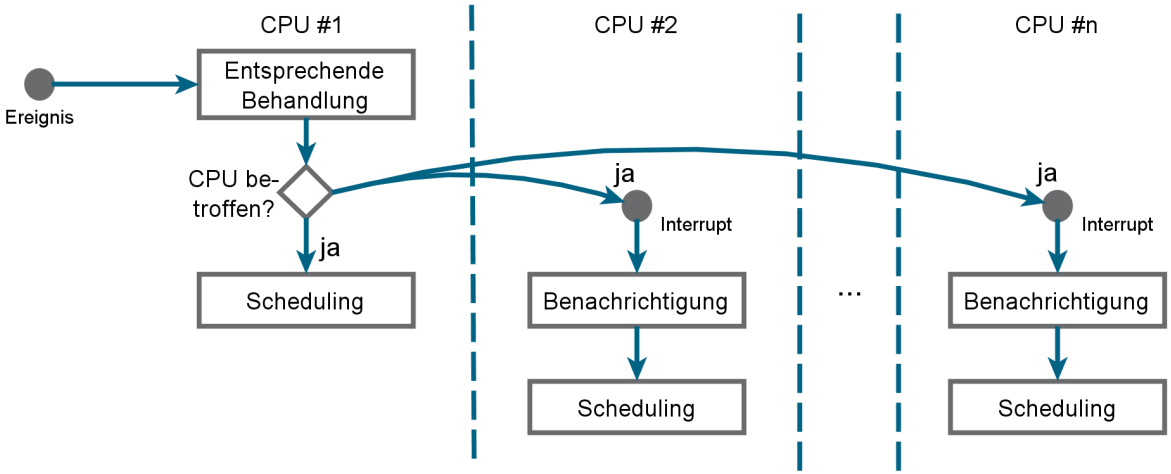

Abbildung: Behandlung eines Ereignisses

- Benachrichtigung entfernter Cores durch Software Interrupts.

Vorteile

- Funktionierende, gut skalierbare Lösung.
- Einfache Portierung bestehender Anwendungen möglich.
- Speicherbedarf für Code (ROM) ist besonders gering.
- Realisierung eines Prototyps auf einer Plattform der Power Architektur (Power PC).

Vorteile

- Funktionierende, gut skalierbare Lösung.
- Einfache Portierung bestehender Anwendungen möglich.
- Speicherbedarf für Code (ROM) ist besonders gering.
- Realisierung eines Prototyps auf einer Plattform der Power Architektur (Power PC).

Nachteile

- Zusätzlich nötige Schutzmechanismen sorgen für eine Zunahme bei der Laufzeit.
- Bei großer Anzahl an CPUs besteht die Gefahr der gegenseitigen Behinderung.

- Erweiterungen an µC/OS-II SMP wurden abgeschlossen.
- Hinzugekommen ist die Möglichkeit einzelne Tasks während der Laufzeit auf einen anderen Core zu verschieben.
- Abschluss weiterer Tests mit dem Ziel der Produktreife.
- Bereits eine erfolgreiche Evaluierung in einem Kundenprojekt.

Mein besonderer Dank gilt:

- der Firma Embedded Office für Betreuung und Unterstützung, in erster Linie durch Reinhard Baur und Michael Hillmann.
- Prof. Rainer Müller für die Übernahme der Erstbetreuung meiner Thesis.
- dem Fachausschuss „Echtzeitsysteme" der Gesellschaft für Informatik e.V. für die Prämierung und die Möglichkeit heute die Ergebnisse meiner Arbeit vorzustellen.

Mein besonderer Dank gilt:

- der Firma Embedded Office für Betreuung und Unterstützung, in erster Linie durch Reinhard Baur und Michael Hillmann.
- Prof. Rainer Müller für die Übernahme der Erstbetreuung meiner Thesis.
- dem Fachausschuss „Echtzeitsysteme" der Gesellschaft für Informatik e.V. für die Prämierung und die Möglichkeit heute die Ergebnisse meiner Arbeit vorzustellen.

Vielen Dank für die Aufmerksamkeit!

[BIT10] BITKOM (Hrsg.): *Eingebettete Systeme – Ein strategisches Wachstumsfeld für Deutschland: Anwendungsbeispiele, Zahlen und Trends*. http://www.iese.fraunhofer.de/content/dam/iese/de/documents/EingebetteteSysteme_web_tcm122-44787.pdf. Version: 2010

[Fre09] Freescale Semiconductor Inc.: *MPC5510 Microcontroller Family Data Sheet*. http://cache.freescale.com/files/32bit/doc/data_sheet/MPC5510.pdf. Version: 2009

[HPAD07] Hennessy, John L. ; Patterson, David A. ; Arpaci-Dusseau, Andrea C.: *Computer architecture: A quantitative approach*. 4. Amsterdam and Boston : Morgan Kaufmann, 2007. – ISBN 9780123735904

[Jon07] Jones, Tim M.: *Linux and symmetric multiprocessing: Unlock the power of Linux on SMP systems*. http://www.ibm.com/developerworks/library/l-linux-smp/. Version: 2007

[Lab02] Labrosse, Jean J.: *MicroC/OS-II: The real-time kernel*. 2. Lawrence and Kan : CMP Books, 2002. – ISBN 9781578201037

[McK08] McKenna, Daniel ; Freescale Semiconductor Inc. (Hrsg.): *Designing Code for the MPC5510 Z0 Core*. http://cache.freescale.com/files/microcontrollers/doc/app_note/AN3614.pdf. Version: 0.5, 2008

[Pla07] Platt, Christopher ; Freescale Semiconductor Inc. (Hrsg.): *Power Architecture™Demystified*. http://cache.freescale.com/files/32bit/doc/white_paper/PADEMYSTWP.pdf. Version: 2007

[Tan09] Tanenbaum, Andrew S.: *Moderne Betriebssysteme*. 3. München and Boston [u.a.] : Pearson Studium, 2009. – ISBN 3827373425