Parallelisierung auf MPSoC-Plattformen

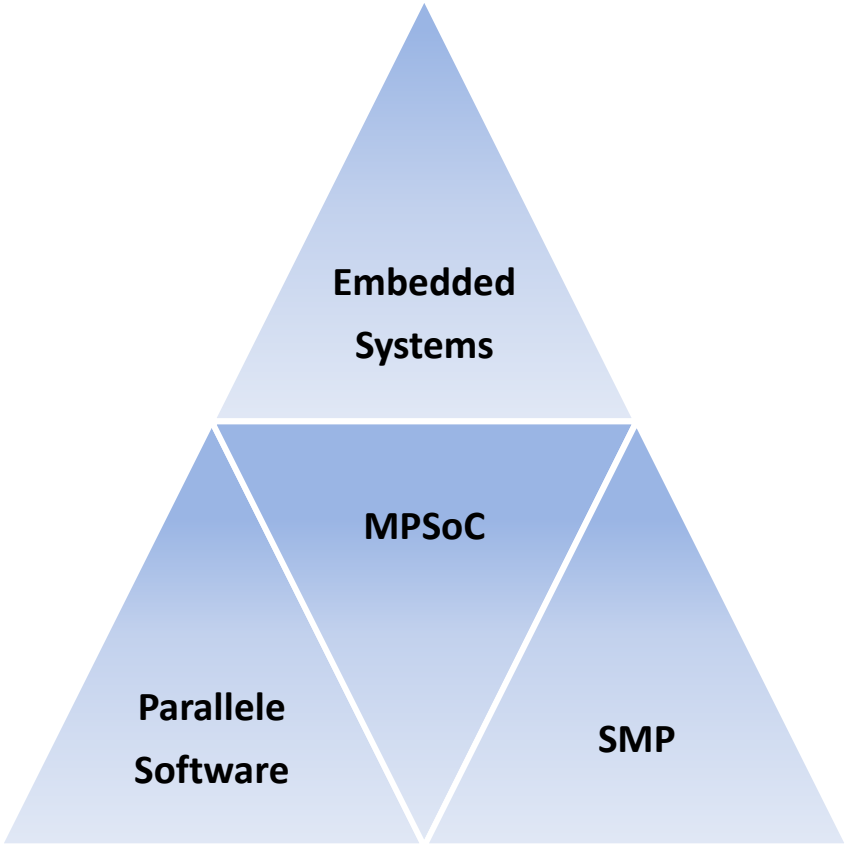
MINF 1, WiSe 2011
Anwendungen 1
17.11.2011
Steffen Rempp

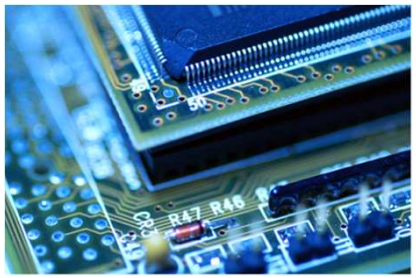
Betreuer: Prof. Dr. Schwarz

Übersicht

1.	Einleitung	Beschreibung des Themas MPSoC Motivation
2.	Multiprozessor-Architekturen	Softwarehierarchie Vergleich von MP Architekturen
3.	SMP im Linux Kernel	Kernel-Komponenten SMP Scheduling
4.	Entwurfstechniken für Parallelisierung	POSIX threads OpenMP
5.	Ausblick	

Themenbeschreibung



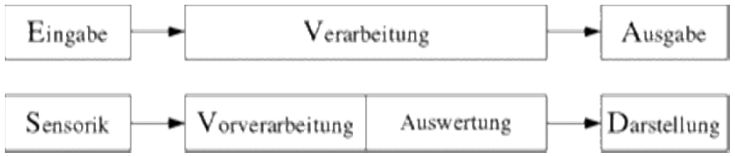


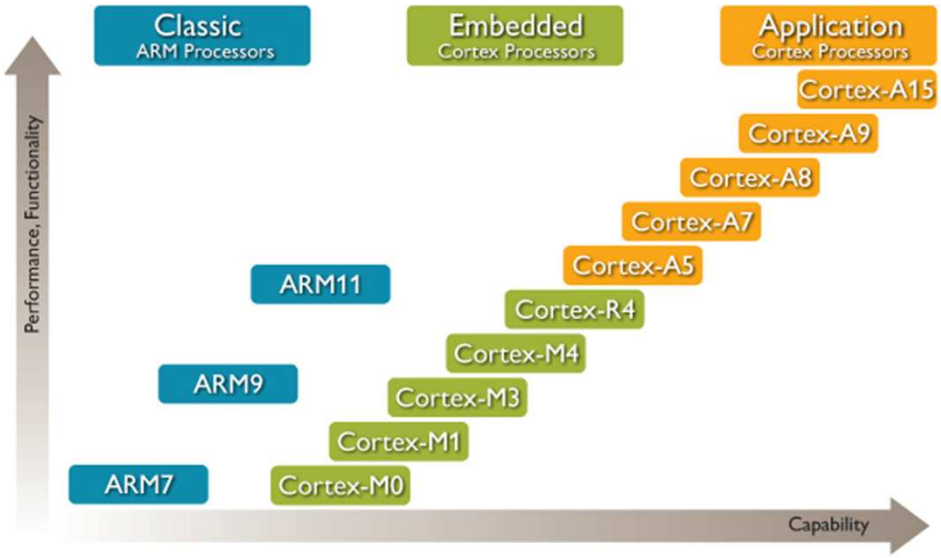
Anwendungsfelder:

- Automotive Systems
- Avionics
- Consumer Electronics
- Robotics
- Mobile Devices
- Medical Healthcare

Kernaspekte:

- Steuerung / Regelung eines technischen Kontextes
- Sicherheitskritische Anwendungen
- Echtzeitfähigkeit
- Starke Kopplung an Hardware: Sensorik – Verarbeitung – Aktorik
- Anpassung auf spezielle Aufgaben
- Verarbeitung großer Datenmengen
- Hohe Performance Ansprüche
  - Hohe Rechenleistung bei geringem Stromverbrauch
- Stärker eingeschränkte Speicherhierarchie als bei PC
- Vielzahl an Architekturen und Plattformen
- (Meistens) unsichtbar für Benutzer





MPSoC – Multiprozessor System on Chip

„on Chip“:

- Alle Komponenten/Funktionen sind auf einem IC (Chip) integriert
  - Bsp: FPGA

System: Kombination unterschiedlicher Elemente (IP-Cores):

- Logische Schaltungen
- Microcontroller / Microprozessoren
- Taktgebung
- Speicherhierarchie
- Sensoren
- (Hardware-) Peripherie: Schnittstellen, I/O
- Bussysteme

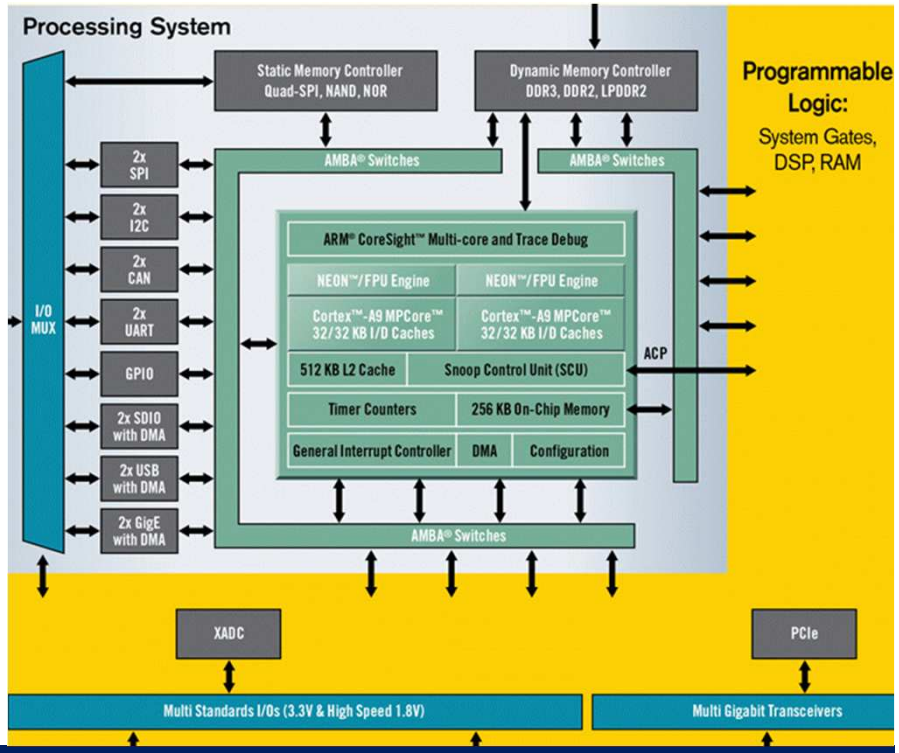
SoC: Ein Prozessor Subsystem

MPSoC: Mehrere Prozessor Subsysteme

Beispiel:

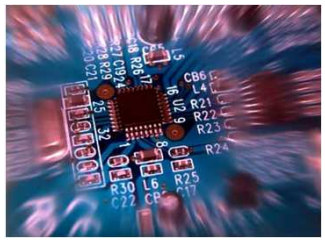
XILINX Zynq 7000 Extensible Processing Plattform [1]

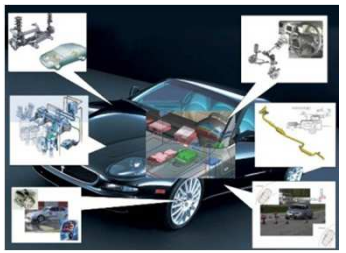
- ARM dualcore Cortex A9 MPCore [2]
- XILINX 28 nm FPGA
- AMBA AXI Interface zw. Prozessor und FPGA
- Prozessor-zentrale Architektur
- Software-Entwicklung, System-Architektur und programmierbare Logik parallel
- FPGA Konfiguration über Prozessor



Motivation:

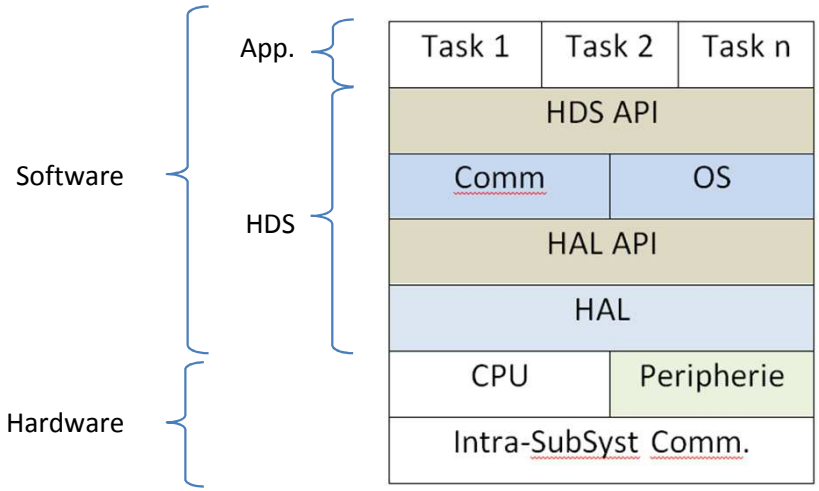
- Software-Entwicklung in Embedded Systems
- Migration von bereits etablierten Software-Entwurfstechniken in den Embedded Bereich
- Wunsch innerhalb der Branche: Standardisierung
- Herausforderung: Parallelisieren von sicherheitskritischen und echtzeitfähigen Anwendungen
- Verwendungsmöglichkeiten von *Symmetric Multiprocessing* (SMP) im Embedded Bereich



Übersicht

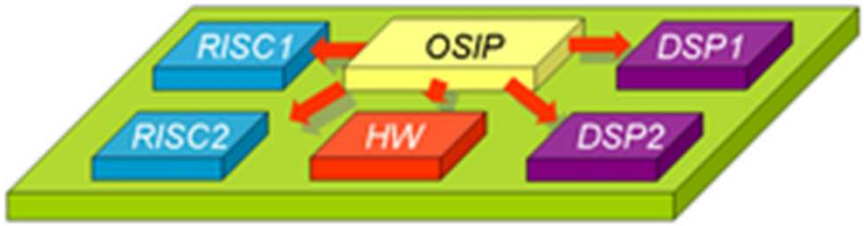
Softwarehierarchie auf Processing System [3]



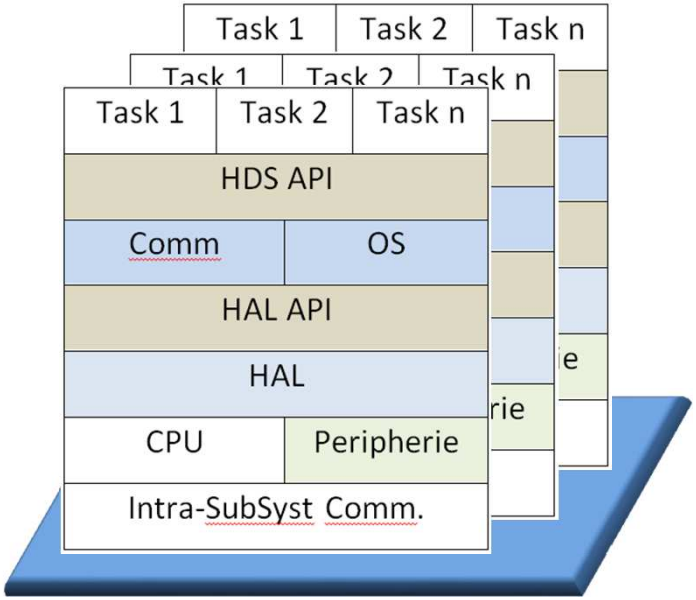
Gliederung in zwei Teile:

- **Application Layer**
- **HDS Layer:**
  - Gliederung in drei Teile:
    - (RT-)OS
    - Comm, I/O
    - HAL
  - Dienste für Application Layer
    - Task Scheduling → (RT-OS): SMP
    - IPC
    - Ext. Kommunikation mit anderen Subsystemen
    - Hardware Resource Management
  - Zusammenfassung aller Dienste in HDS API

Heterogene MPSoC



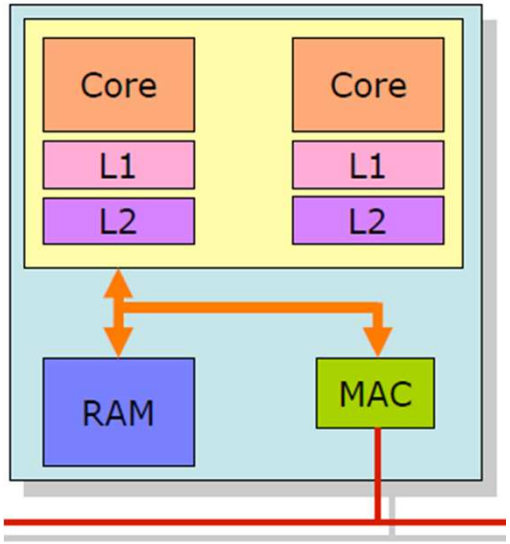
- Dedes Prozessor Subsystem führt seinen eigenen Software Stack aus
- Aufgabenverteilung
- Verschiedene Prozessortypen/Befehlssätze
- Kein gemeinsamer Speicher
- Jedes Subsystem hat seinen eigenen Adressraum
- Kommunikation zw. Subsystemen über Message Passing (Bussysteme)



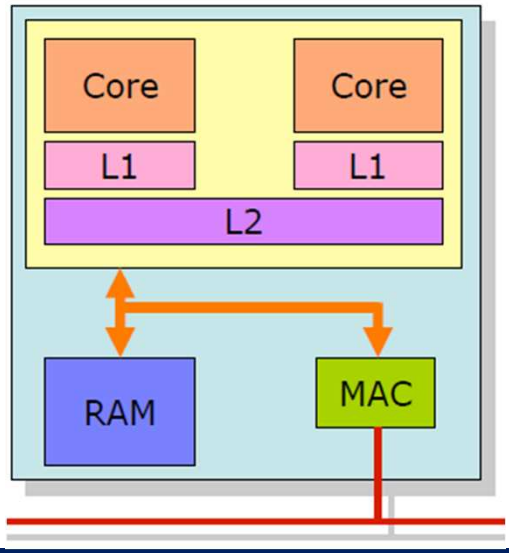
Symmetric Multiprocessing (SMP)

- Mehrere gleichberechtigte Prozessoren greifen über einen gemeinsamen Speicherbus auf denselben Speicher zu
- Gemeinsamer Adressraum (Shared Memory)
- Ein einziger Software Stack für alle CPUs
- Dynamische Verteilung von Prozessen auf verfügbare CPU
- Vorteil: Direkter und gleich schneller Zugriff auf die Daten von allen Prozessoren (UMA)
- Nachteile:
  - Gemeinsamer Speicherbus ist Flaschenhals
  - Kaum Systeme mit mehr als 16 CPUs

AMD:

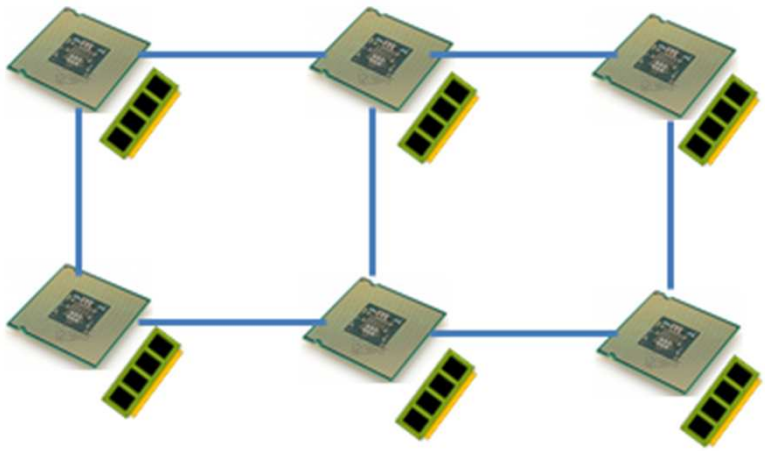


Intel:



Non-Uniform Memory Access (NUMA)

- SMP Variante / Erweiterung
- „P2P“ Netzwerk von Prozessor Subsystemen
- Jedes Subsystem hat seinen lokalen Speicher
- Gleicher Adressraum, unterschiedliche Speicherblöcke
- Distributed Shared Memory
- Vorteil: Speicherbus zum lokalen Speicher wird weniger belastet
- Nachteile:
  - Unterschiedliche Antwortzeiten bei Daten in verschiedenen Adressbereichen
  - Overhead durch Cache Kohärenz

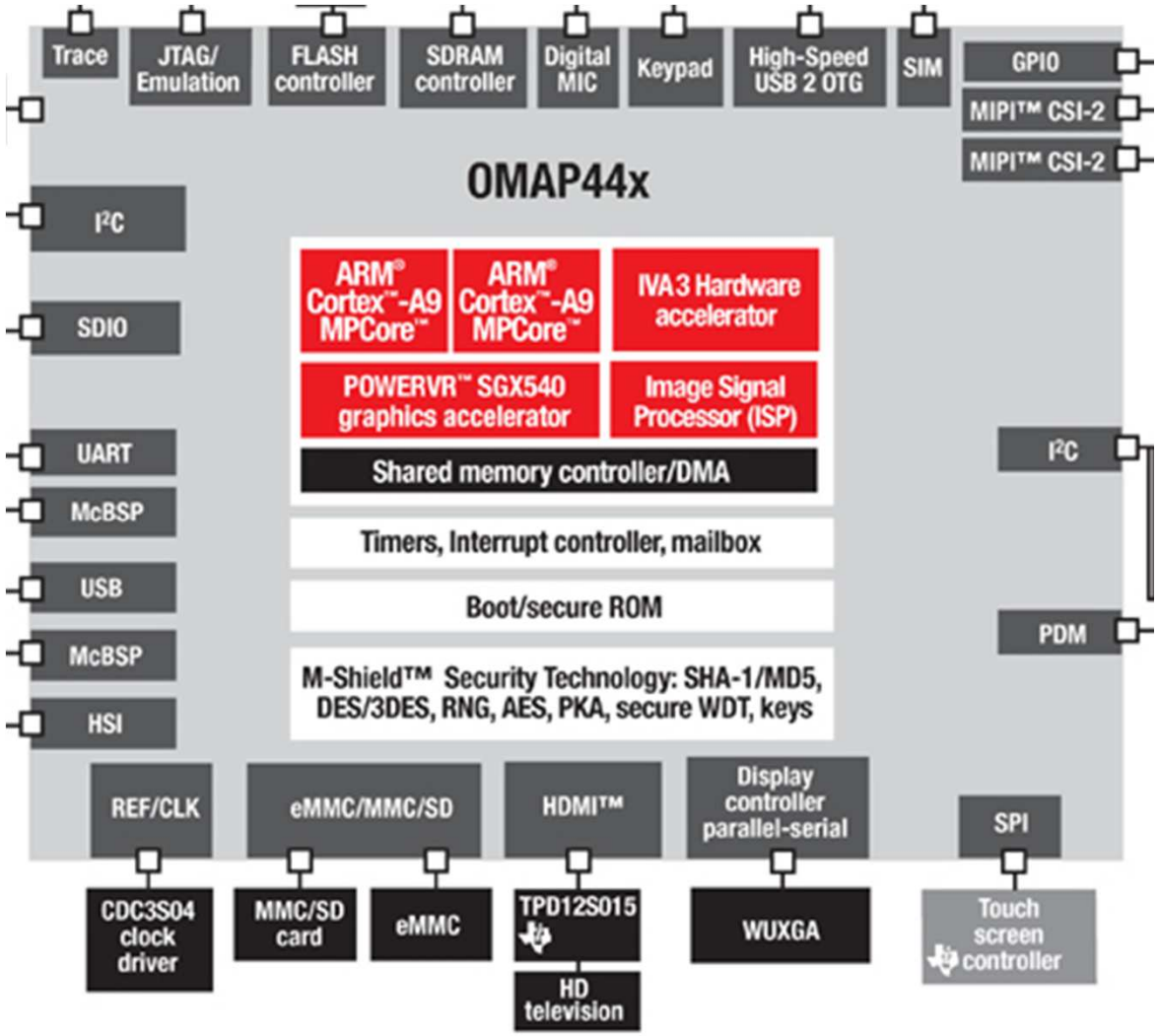


Kombination aus SMP und heterogenem MPSoC
→ SMP Subsystem in heterogenem MPSoC integriert

Bsp:

OMAP 4 Architektur [4]

- ARM Cortex A9 DualCore [2]
- ARM Cortex M3
- IVA3 Hardware-Beschleuniger
- Image Signal Prozessor (ISP)
- POWERVR Grafikbeschleuniger



Zusätzliche Hardware-Komponenten Zur Synchronisation:

- Shared Memory Controller (DMA)
- Memory Management Unit (MMU)
- Snoop Control Unit (SCU)

Übersicht

1.	Einleitung	Beschreibung des Themas Motivation Projekt 1
2.	Multiprozessor-Architekturen	Softwarehierarchie Vergleich von MP Architekturen
3.	**SMP im Linux Kernel**	**Kernel-Komponenten** **SMP Scheduling**
4.	Entwurfstechniken für Parallelisierung	POSIX threads OpenMP
5.	Ausblick	

Kernel Komponenten für SMP Scheduling [5]:

- **Prozesse:**
  - Threads, Tasks, Programme, Applikationen
  - Software Konstrukte
  - Eigener Adressraum
  - Sequentieller Ausführungsstrang von Befehlen

```
// definiert in /include/linux/sched.h
struct task_struct
{
    //...
};
```

- **Runqueue:**
  - Doppelt verkettete Listen von Prozessen
  - Unterteilt in Prioritäten
  - Jeder CPU wird eine eigene Runqueue zugewiesen

```
// definiert in /include/linux/sched.h
struct rq
{
    //...
};
```

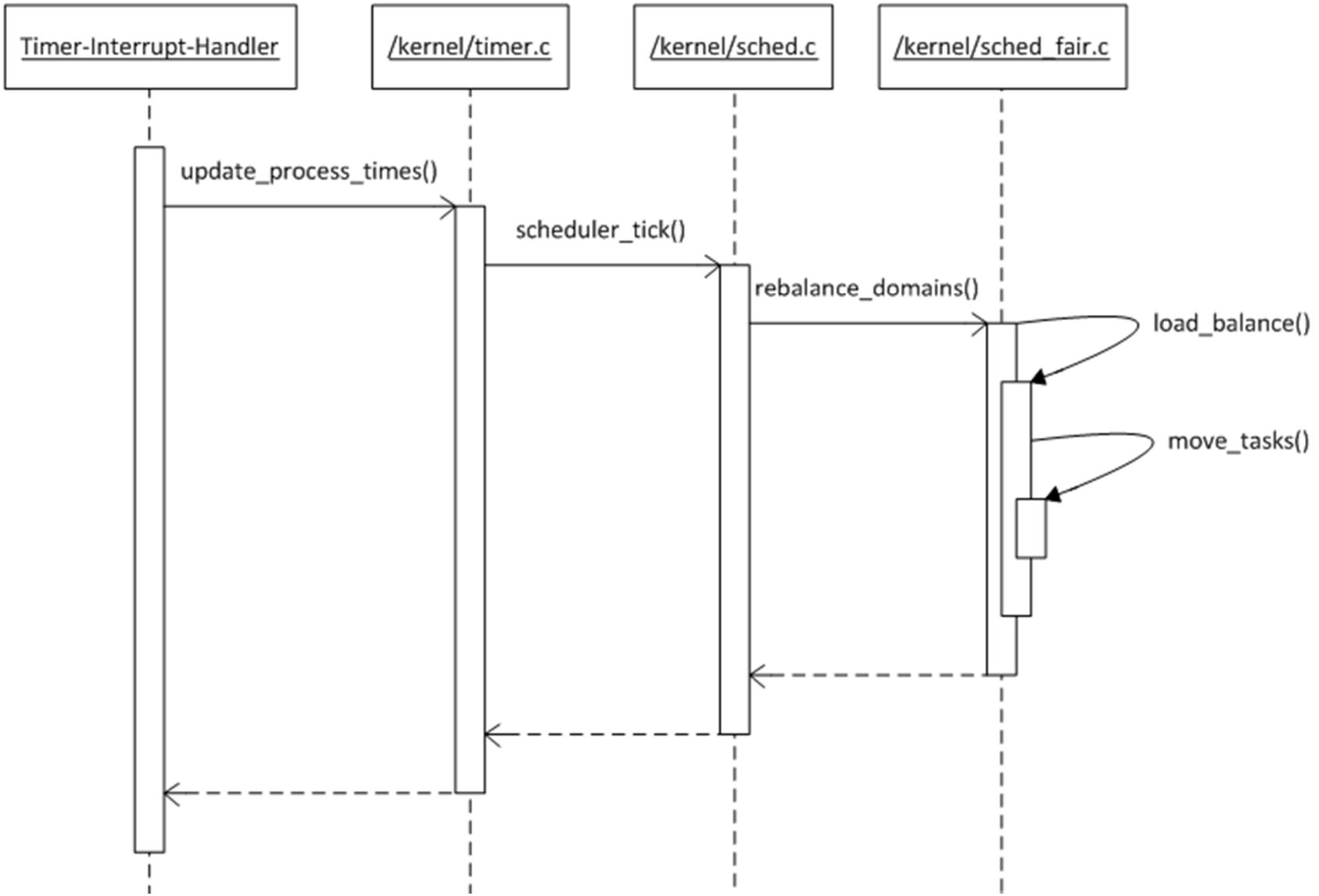
- **Scheduling Domain**
  - Gruppe von zusammenhängenden CPUs
  - Hyperthreading:
    - Eine physikalische CPU
    - Zwei logische CPUs
    - Eine Scheduling Domain

```
//definiert in /kernel/schedule.c
struct sched_domain
{
    //...
};
```

SMP Scheduling [6]:
Symmetrisches Verteilen von Prozessen auf Runqueues

1. Überprüfung der Zeitlimit-Überschreitung von Prozessen
2. Überprüfung der Aktivität von Prozessen
3. Aktualisierung der dynamischen Priorität
4. Verschieben von Prozessen innerhalb der Runqueue
5. Arbeitsauslastung aller Runqueues überwachen
6. Migrieren von Prozessen zwischen Runqueues
7. Re-Scheduling

SMP – Sequenz im Linux-Kernel:



Übersicht

Entwurfstechniken für Parallelisierung

„Make use of Thread-Safe libraries wherever possible.“ (The Art Of Concurrency, Rule 4) [7]

POSIX Threads [8]

- Standard: POSIX 1c, Threads extensions (IEEE Std 1003.1c-1995)
- Threading API, Implementierungen in UNIX Betriebssystemen
- Thread-Operationen: Erzeugen, Terminieren, Synchronisieren von Threads
- Scheduling, data management und Prozess-Interaktion

„You cannot program a multiprocessor effectively without understanding your multiprocessor architecture.“ (The Art Of Multiprocessor Programming) [9]

Open Multi-Processing (OpenMP) [10]

- Plattformübergreifende API für Shared Memory Multiprozessor Programmierung
- Einfache und flexible Interfaces, um parallele Applikationen zu entwickeln
- OpenMP Support and Extensions für MPSoC with explicitly managed Memory Hierarchy [11]

Übersicht

Ziel (Projektarbeiten, Masterarbeit):

Integration von Embedded Software-Komponenten mehrerer Echtzeit Anwendungen auf einer MPSoC Plattform

Zielplattform:	XILINX Zync 7000 Extensible Processing Plattform [1]
Embedded Anwendungen:	SCV (FAUST) [12], Bsp: Fahrspurassistent, Bremsassistent, Ausweichassistent,…
Vorgehen:	➢ Entwickeln einer **Software-Architektur** für flexible Integration von Komponenten ➢ Entwickeln von **Basisfunktionen** (Plattform API), die von allen Applikationen genutzt werden ➢ Kommunikation ➢ Datenaustausch mit FPGA Blöcken ➢ Interrupt Controlling ➢ Speicherverwaltung ➢ Synchronisation Hardware-Zugriff ➢ Execution Controlling ➢ Gateway ➢ Memory Management

Danke für die Aufmerksamkeit

Fragen?

Quellen (1)

[1]: XILINX Zync 7000 Extensible Processing Plattform. Website.2011.
URL http://www.xilinx.com/products/silicon-devices/epp/zynq-7000/index.htm

[2]: ARM Cortex-A9 MPCore Processor. Website.2011
URL http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[3]: K. Popovici, F. Rousseau. *Embedded Software Design and Programming of Multiprocessor System-on-Chip*.
Springer, 2010. ISBN: 978-1-4419-5566-1

[4]: Texas Instruments: OMAP4430 Platform. Website. 2011. – URL http://focus.ti.com/general/docs/wtbu/
wtbuproductcontent.tsp?contentId=53243&navigationId=12843&templateId=6123

[5]: The Linux Kernel Archievs. Website. 2011.
URL http://kernel.org/

[6]: D.P. Bovet, M. Cesati. *Understanding the Linux Kernel*. O'Reilly Media; Auflage: 3rd ed. (2006).
ISBN: 978-0596005658

Quellen (2)

[7]: C. Breshears. *The Art of Concurrency*. O'Reilly Media, 2009. ISBN: 978-0-596-52153-0

[8]: POSIX Threads Programming. Website. 2011.
URL https://computing.llnl.gov/tutorials/pthreads

[9]: M. Herlihy, N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008.ISBN: 978-0-12-370591-4

[10]: The OpenMP API Specification for Parallel Programming. Website. 2011.
URL http://openmp.org/wp/

[11]: MARONGIU, Andrea ; BENINI, Luca: *Efficient OpenMP support and extensions for MPSoCs with explicitly managed Memory hierarchy*. In: ACM(2009)

[12]: FAUST-PROJEKT: Sensor Controlled Vehicle. Website. 2011.
URL http://www. informatik.haw-hamburg.de/faust.html