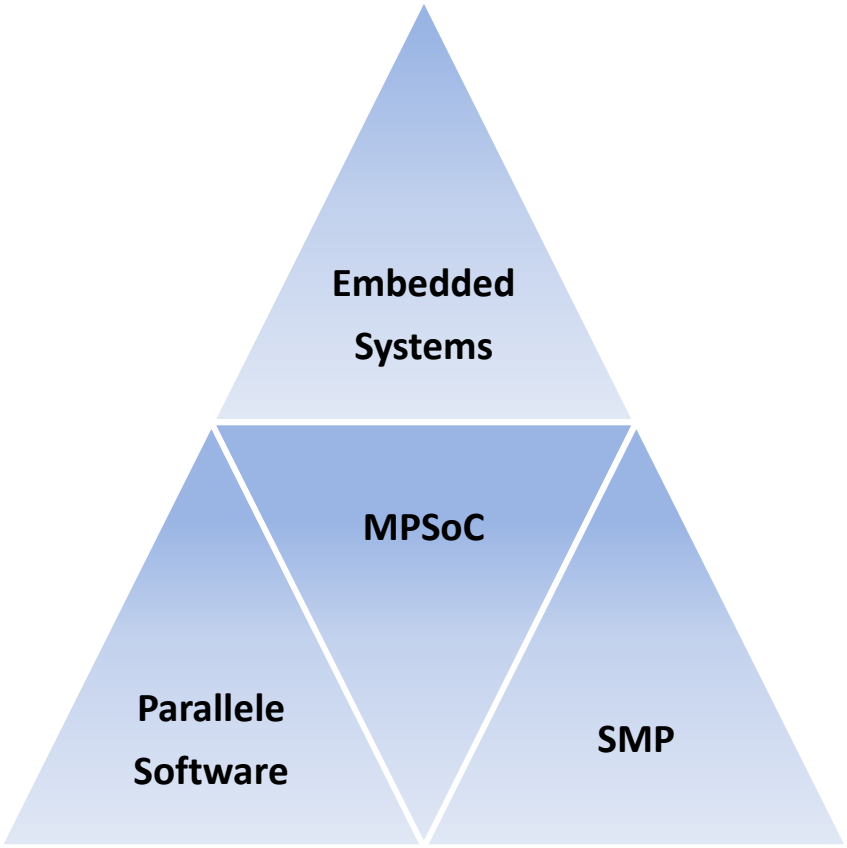
Entwurfstechniken für Parallelisierung Auf MPSoC-Plattformen

MINF 2, SoSe 2011
Anwendungen 2
23.06.2011
Steffen Rempp

Betreuer: Prof. Dr. Schwarz

Übersicht

Themenfelder



Multiprozessor-Architekturen etablieren sich zunehmend in Embedded Systems

MPSoC als Lösungsansatz für:

- Wachsende Komplexität
- Anwendungen fordern mehr Rechenleistung
- Energieeffizienz
- Kosteneffizienz

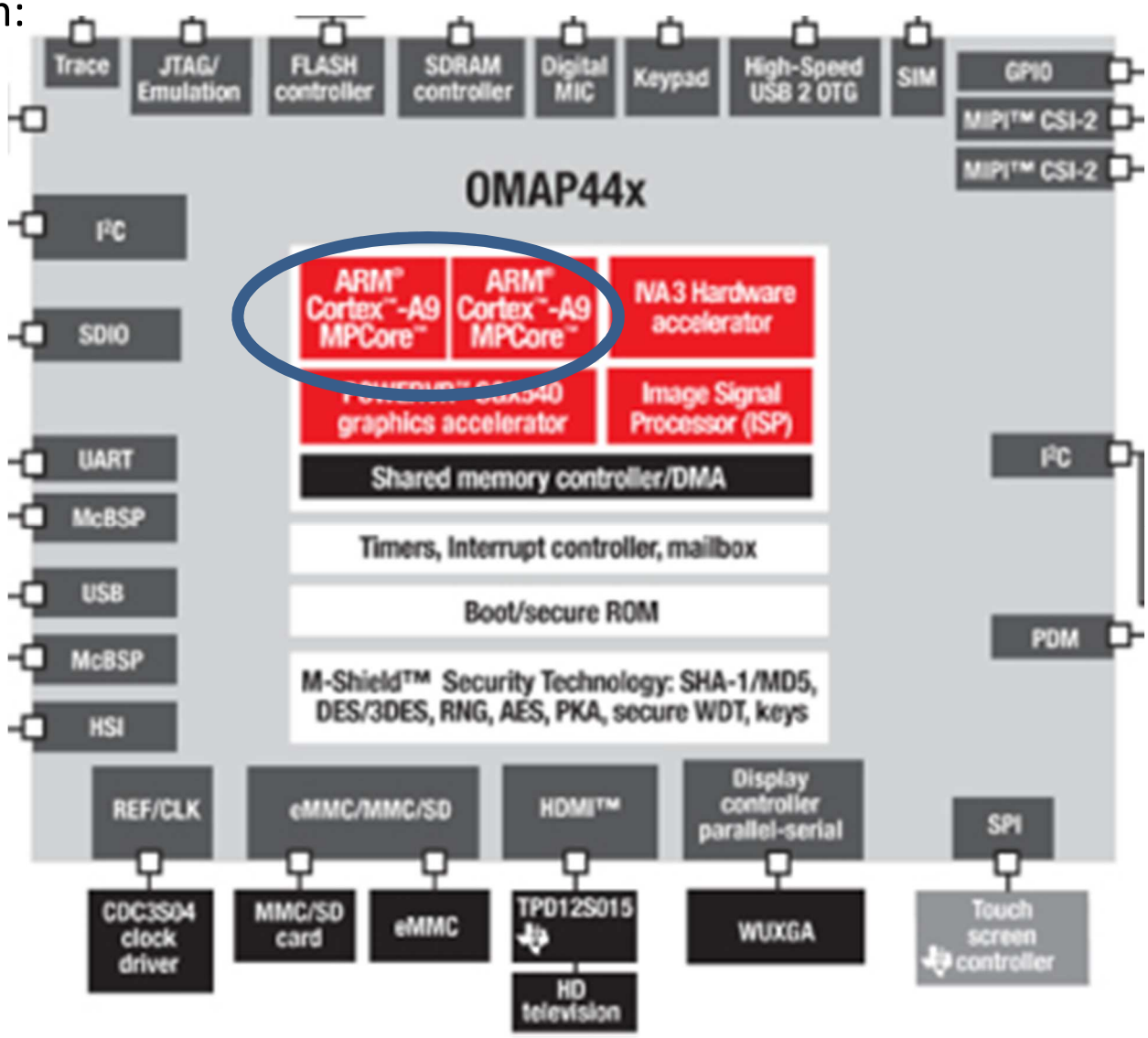
Thema in AW 1:

„Plattformanforderungen für Parallelisierung auf Multiprozessor Systemen on Chip (MPSoC)“

- Vergleich von Multiprozessor-Architekturen
  - SMP, AMP, Heterogene MPSoC,…
- Generisches Strukturmodell einer MPSoC Plattform
  - HW/SW - Ebenen
- SMP: Task Scheduling und Lastverteilung
  - Load Balancing
- Konzepte und Mechanismen für paralleles Programmieren
  - MPI, OpenMP, POSIX Threads, Unified Parallel C

Ziele in Projekt 1:

- Bereitstellen einer MPSoC-Plattform:
  - OMAP TI 4430 Prozessor [1]
    - ➡ ARM Cortex A9 DualCore
-
- Volle Nutzung der SMP-Fähigkeit
  - ➡ Parallelisierung
- Aufbau von KnowHow:
  - Embedded Systems
  - MPSoC
  - SMP



Übersicht


J.Y. Mignolet

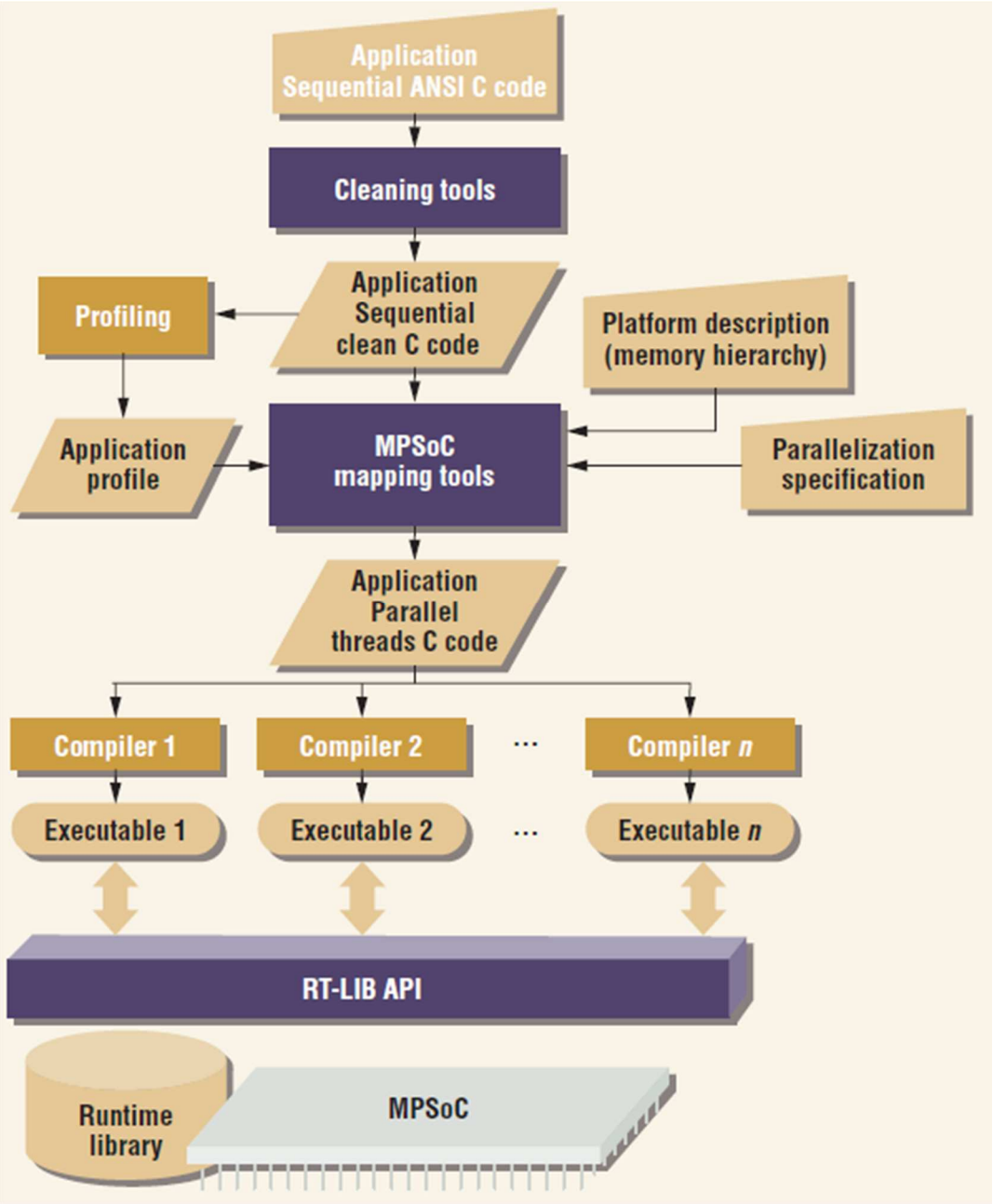

R. Wuyts

IMEC MPSoC Programming Approach, 2009 [2]

IMEC (Interuniversity Microelectronics Centre), Belgien [3]

- Software-Werkzeuge für automatische Parallelisierung von sequentiellem C Code
- Basiert auf Benutzerspezifikationen
- Kontrolle über Interprozesskommunikation
- Transition des sequentiellen Codes in paralleles Programmiermodell

Vergleichbare Arbeiten – IMEC Toolchain



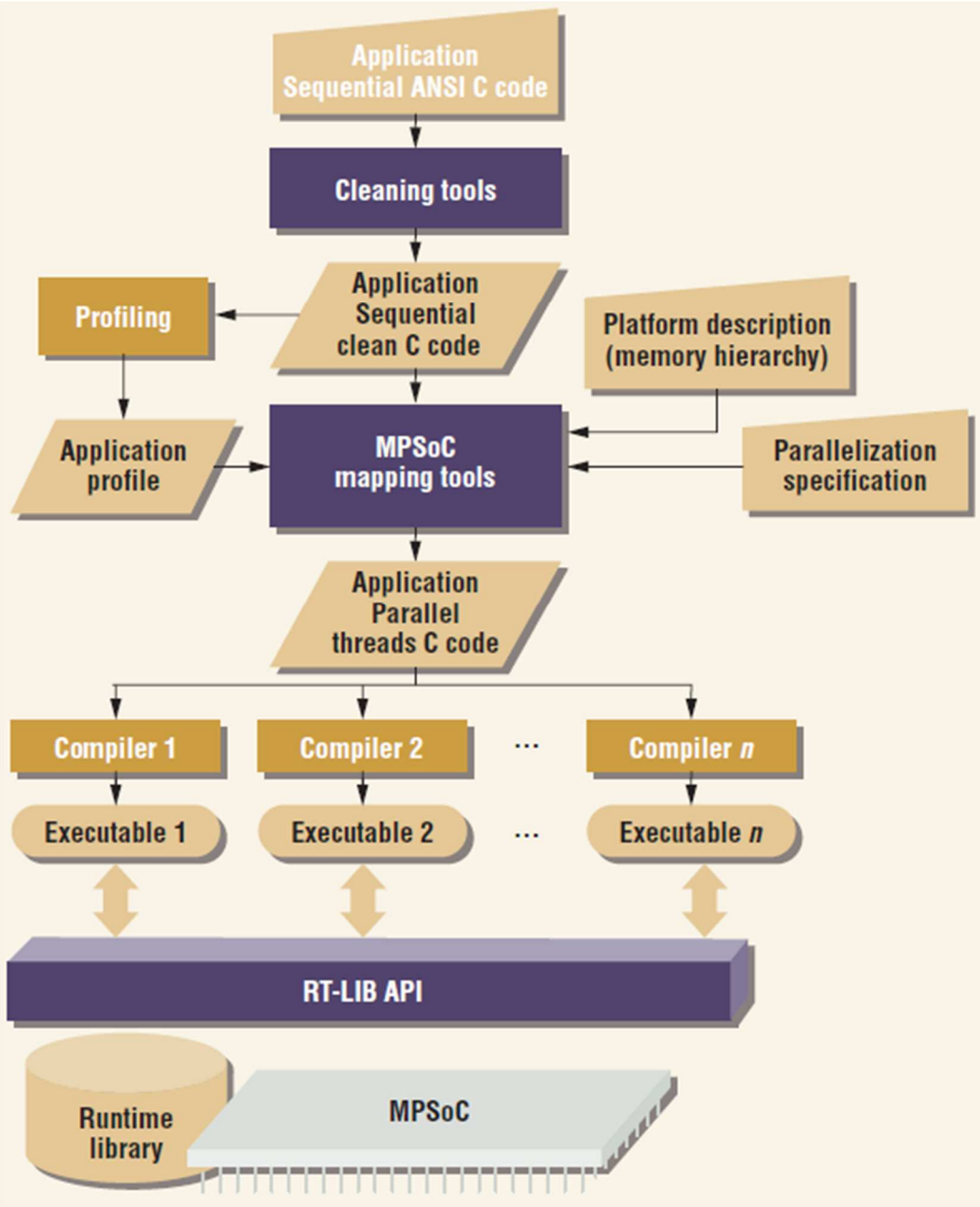
2 Arten von Software-Werkzeugen:

- **Cleaning tools**
  - CleanC
- **Mapping tools**
  - Memory Hierarchy (**MH**) tool
  - MPSoC Parallelizing Assistant (**MPA**)

Runtime Library:

- HW-Abstrahierung
- Plattformunabhängigkeit
- Für ADRES und ARM11 MPCore

Vergleichbare Arbeiten – IMEC Toolchain



MH tool:

- Management der Speicherbelegung
- Transferieren von Datenblöcken
- Lokale Kopien generieren
- Code-Analyse, um relevante Datenblöcke zu finden
- Vermeidung von wiederholtem Speicherzugriff

MPA:

- Identifizierung von parallelisierbaren Code-Abschnitten (parsections)
- Anzahl der Threads festlegen
- Functional parallism vs. Data parallelism
- Alle Parallelisierungs-Direktiven in Datei schreiben
- Hinzufügen von FIFOs für Synchronisation

➡ Generieren von parallelem Code

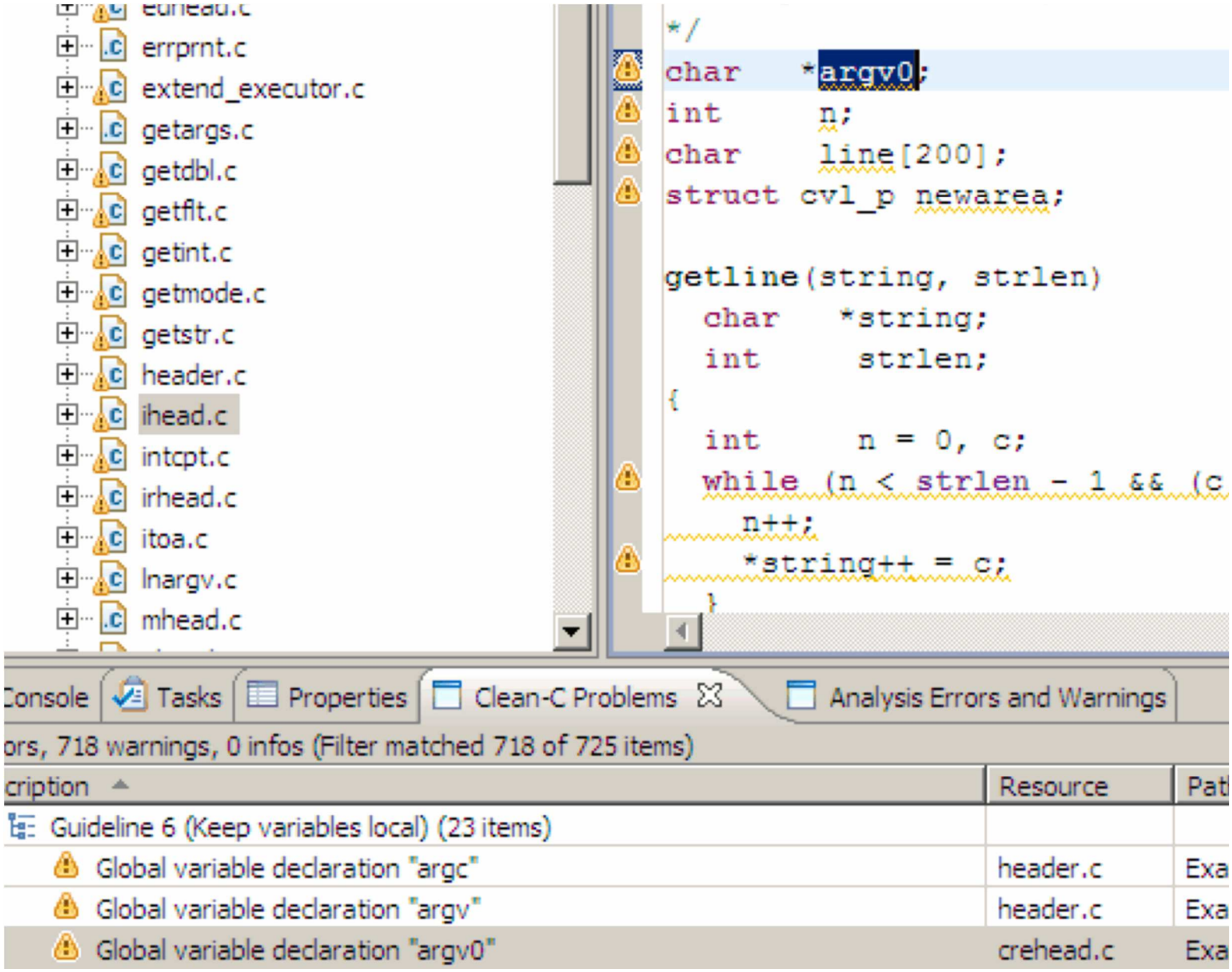
Vergleichbare Arbeiten – IMEC Toolchain

CleanC [4]:

- Eclipse Plugin
- Herausfiltern von „unerwünschten“ Konstrukten im C Code
  - Restrictions
  - Guidelines

Beispiele für Restrictions:

- Cast bei malloc()
- Keine varargs



Übersicht

Vergleichbare Arbeiten – OpenMP für MPSoC

Efficient OpenMP Support and Extensions for MPSoCs [5]

Universität Bologna
Andrea Marongiu
Luca Benini

- OpenMP[6] Implementierung für MPSoC ohne Cache-Kohärenz
- Erweiterungen mit explizitem Management der Speicherhierarchie
- Erweiterter CCC 4.3.2 OpenMP Compiler
- Erweiterung der OpenMP runtime library (libgomp)

Vergleichbare Arbeiten – OpenMP für MPSoC

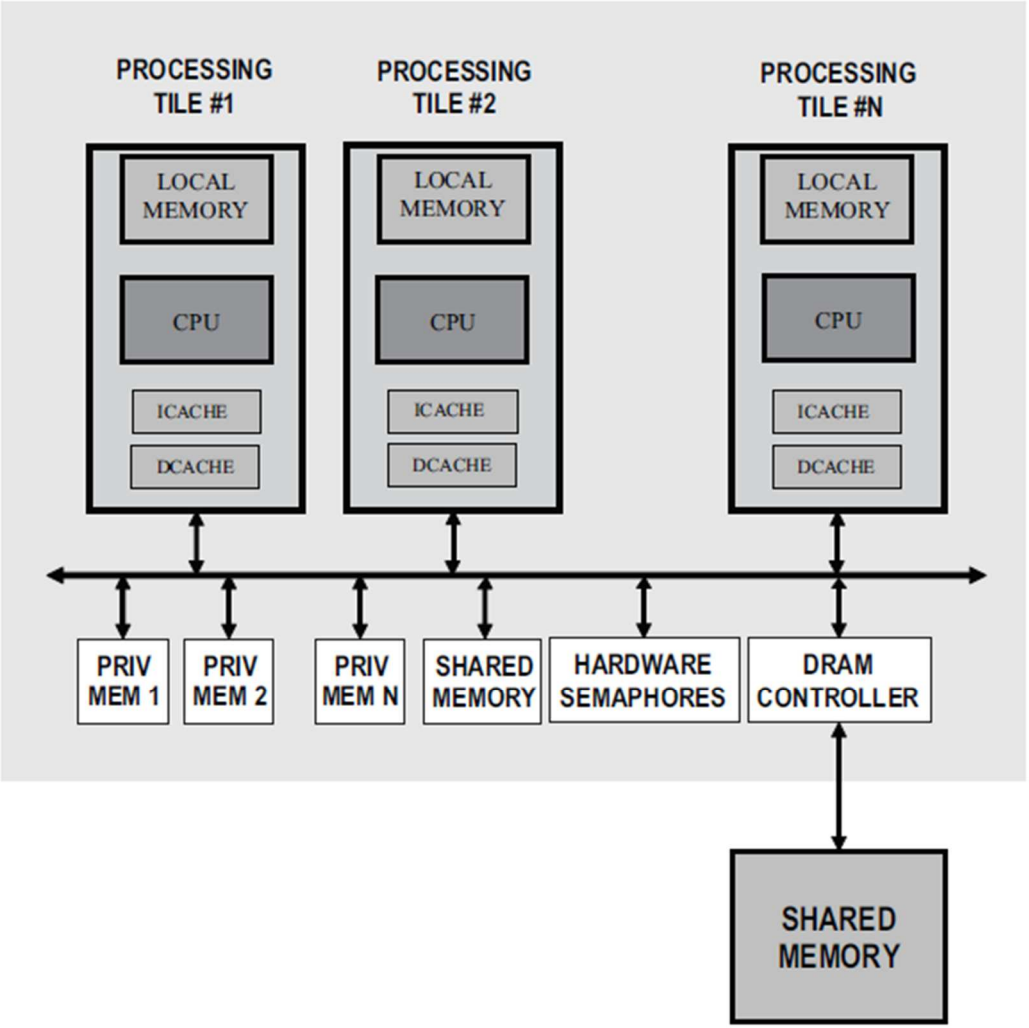
- Herkömmliches OpenMP ungeeignet für MPSoC, wenn:
  - Verschiedene Plattformen → Verschiedene Speicherhierarchien
  - MPSoC typischerweise heterogen → unterschiedliche Prozessor-Einheiten
  - Distributed shared memory → keine Replikation von Daten

- **Lokaler Scratchpad Memory (SPM)** statt Cache:
  - SPM befindet sich näher am Prozessor
  - Viel geringere Zugriffszeit (1 cyle)
  - Garantiert immer dieselbe Zugriffszeit im Gegensatz zu Caches

Vergleichbare Arbeiten – OpenMP für MPSoC

Muster-Plattform:

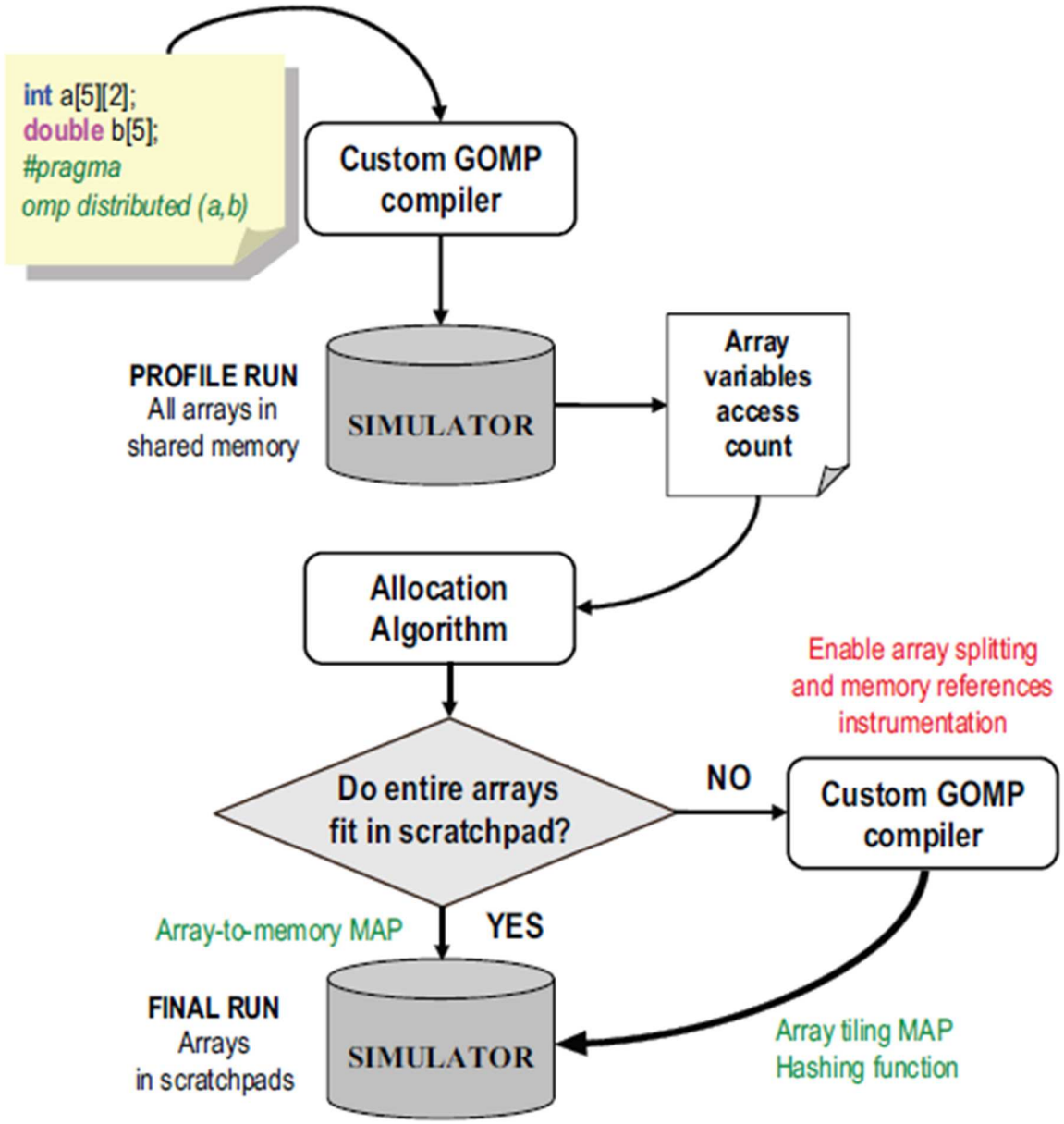
- Mehrere CPUs
  - L1 Data und Instruction Caches
  - Lokaler scratchpad memory (SPM)
- Shared memory on Chip
- Evtl. DRAM off Chip
- Keine Hardware Cache Kohärenz



Passende Zielplattformen:

- IBM Cell BE
- TI OMAP [1]
- Cradle 3SoC

Vergleichbare Arbeiten – OpenMP für MPSoC



Beispiel: Allokieren von Arrays:

```
int i,j;
int a[100][100];
#pragma omp distributed (a)

#pragma omp parallel for

for(i=0; i< 100; i++)
{
    for (j=0; j< 100; j++)
    {
        a[i][j] ++;
    }
}
```

Prinzip:

- Array-Variable wird in SPMs geschrieben
- Ist sie zu groß, wird sie in Blöcke zerlegt

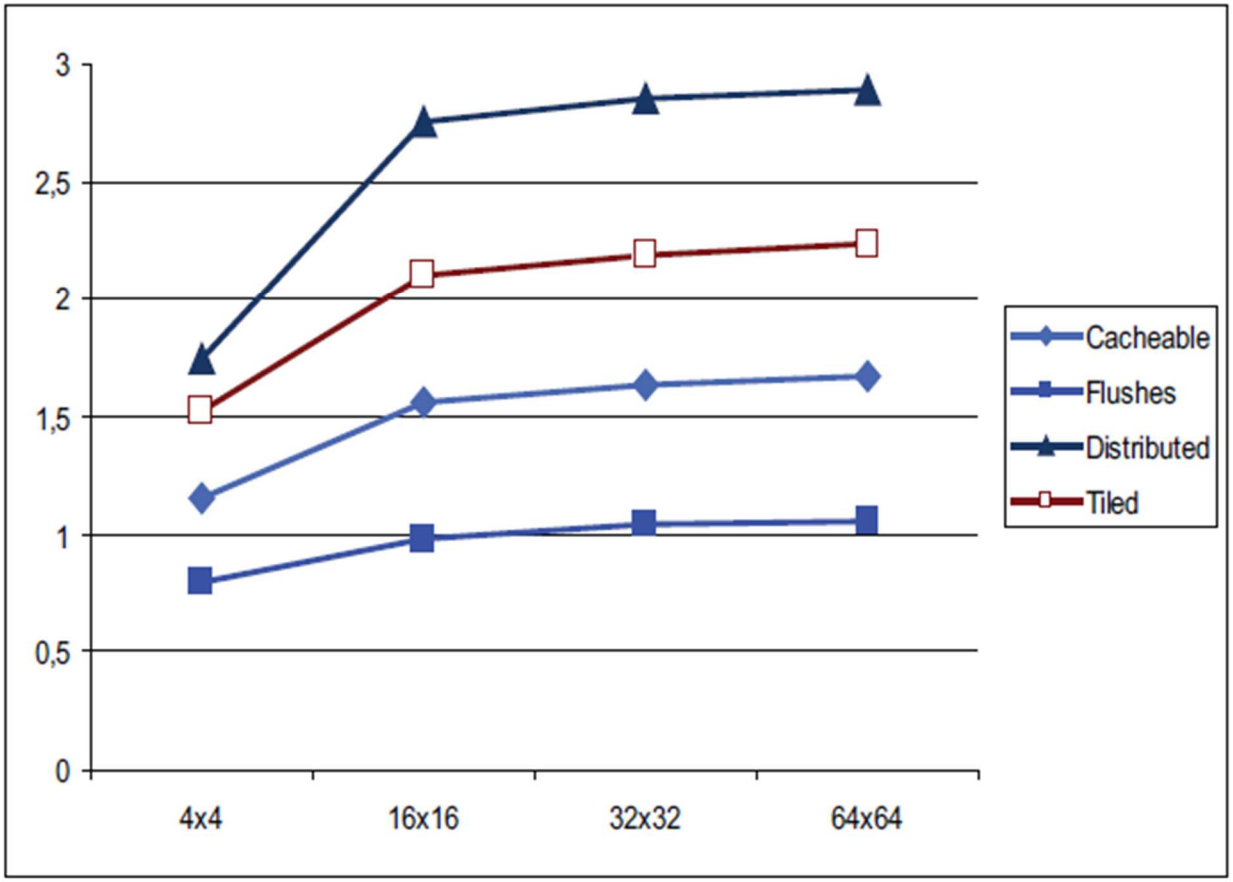
Vergleichbare Arbeiten – OpenMP für MPSoC

Testergebnis bei 4 RISC CPUs (200 MHz),

Speedup / Array-Größe

4 verschiedene Konfigurationen:

- Cacheable
- Flushes
- Distributed
- Tiled



Übersicht

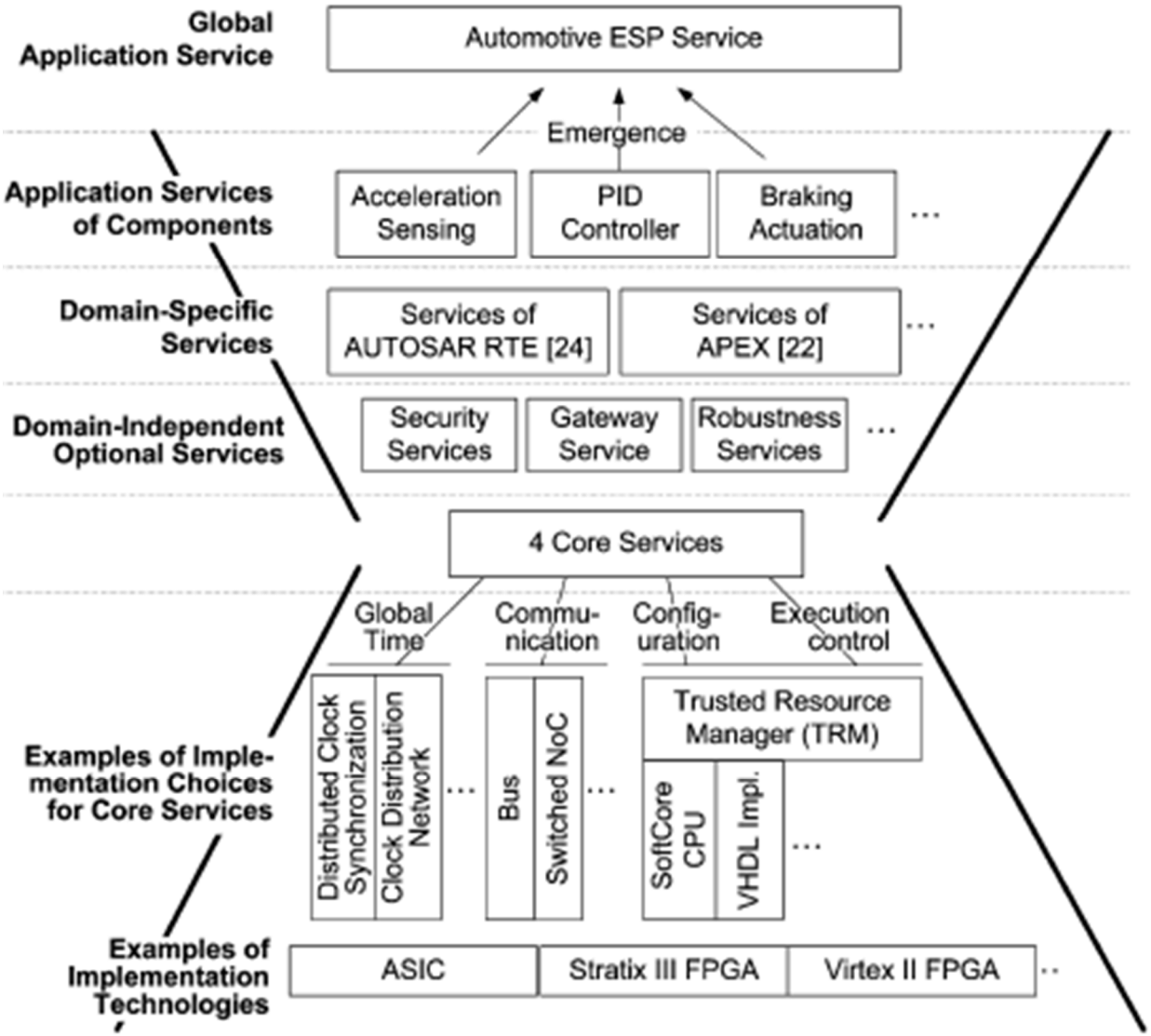
H.Kopetz C. Paukovits R. Obermaiser

A cross-domain MPSoC for Embedded Realtime Systems, 2010 [7]

TU Wien
Institut für Computer Engineering
GENESYS Projekt
(**GE**neric **E**mbedded **SYS**tem Plattform)

- „Bauanleitung“ für MPSoC Design
- Generische, branchenübergreifende (cross-domain) Architektur
- Problemstellung: „zu viele branchenspezifische Standards im Embedded Bereich“
- Motivation: Standardisierung, Konsolidierung im Embedded Bereich

Architektur-Modell:

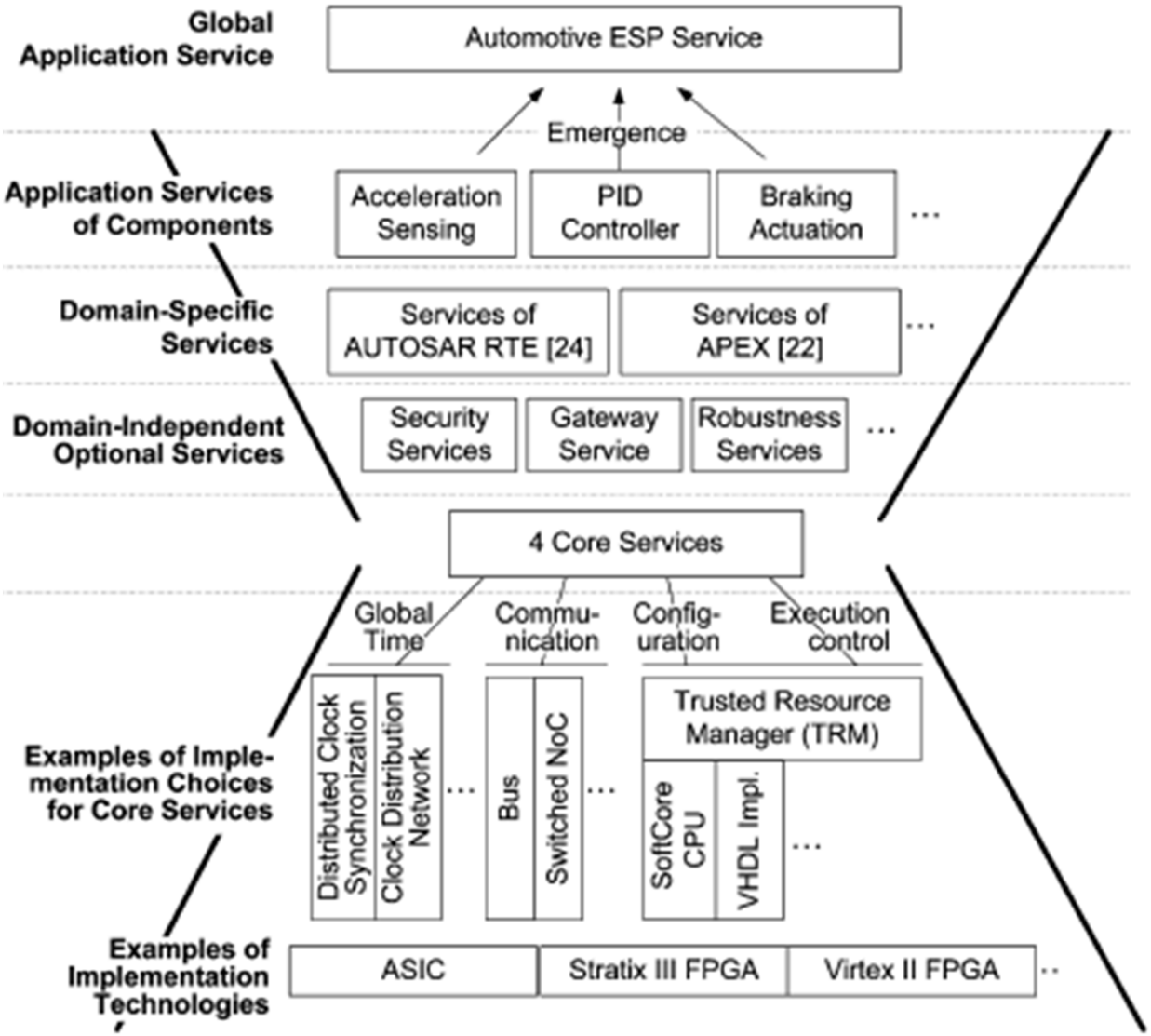


- Service-orientierte Architektur
- Service ←→ Komponente

- ESP-Komponenten/Services:
  - Sensor
  - Controller
  - Aktor

- 4 Core Services:
  - Global Time
  - Communication
  - Configuration
  - Execution control

⟹ Trusted Subsystem



- Fester domain-unabhängiger Kern
- Beliebig viele domain-unabhängige optionale Services
- Domain-spezifische Higher Level Services

Customizing „nach oben“

Ausblick:

- Portieren von domain-spezifischen Services aus anderen Architekturen (z.B. AUTOSAR [8])
- Vermeidung von Redundanzen
- Entwicklung von Mapping Tools

Übersicht

Abgrenzung zum Projekt 1

- Parallelisierung der Software:
  - Manuel parallelisieren (POSIX Threads, OpenMP, MPI) oder
  - Tools zur Parallelisierung benutzen

- Hierarchischer Aufbau der Plattform-Architektur:
  - Applikation
  - Entwurfswerkzeuge zur Parallelisierung
  - Betriebssystem:
    - Momentan Ubuntu[9], Pandroid [10]
    - Echtzeitfähigkeit mit ARTiS [11]
  - Evtl. Runtime Library
  - PandaBoard [12] (OMAP TI)

- Embedded Systems Design:
  - Freiheit beim Design der System- u. Plattformarchitektur
  - Orientieren an vorhandenen Architekturen
  - Von vorneherein gegebene Voraussetzungen
  - Gegebene Plattform und/oder Applikation

ZIEL:

- Bereitstellen einer MPSoC-Plattform für SCV-Projekt [13]
- Argumente pro/contra SMP formulieren

Danke für die Aufmerksamkeit

Quellen

[1] Texas Instruments: OMAP4430 Platform. Website. 2011. – URL http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?contentId=53243&navigationId=12843&templateId=6123. – Zugriffsdatum: 29.05.2011

[2] MIGNOLET, Jean-Yves ; WUYTS, Roel: Embedded Multiprocessor Systems-on-Chip Programming. In: IEEE Software (2009), Juni

[3] IMEC: URL http://www.imec.be

[4] CleanC: URL http://www.imec.be/CleanC

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

[6] OpenMP: URL http://www.openmp.org

[7] KOPETZ, Hermann; OBERMAISER Roman; PAUKOVITS, Christian: A Cross-Domain Multiprocessor System-on-a-Chip For Embedded Real-Time Systems. In IEEE Software (2010), November

[8] AUTOSAR: URL http://www.autosar.org

Quellen

[9] Ubuntu pre-built binaries: URL http://omappedia.org/wiki/Prebuilt_ubuntu_binaries

[10] Pandroid pre-built binaries: URL http://omappedia.org/wiki/Android_Pre-built_Binaries_Guide

[11] ARTis: Assymetric Real-Time Scheduling. URL http://gna.org/projects/artis

[12] PandaBord: URL http://pandaboard.org

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