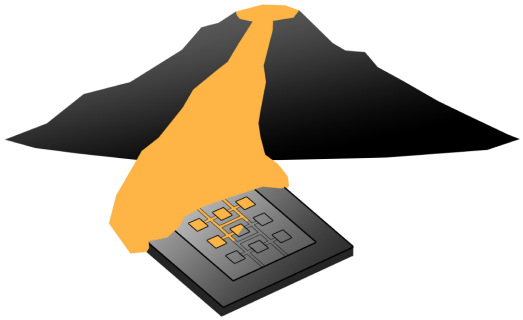
technische universität
dortmund

LavA



Betriebssysteme für konfigurierbare eingebettete Rechnersysteme von morgen

Michael Engel, Matthias Meier, Horst Schirmeier,
Jochen Streicher und **Olaf Spinczyk**
Arbeitsgruppe Eingebettete Systemsoftware

Lehrstuhl für Informatik 12
TU Dortmund

olaf.spinczyk@tu-dortmund.de
http://ess.cs.uni-dortmund.de/~os/

ESS

Überblick

- Konfigurierbare Hardware
  - Vorzüge und Probleme
- Das LavA-Projekt
  - Ansatz
  - Erste Systembausteine
- Fazit und Ausblick

Überblick

- **Konfigurierbare Hardware**
  - **Vorzüge und Probleme**
- Das LavA-Projekt
  - Ansatz
  - Erste Systembausteine
- Fazit und Ausblick

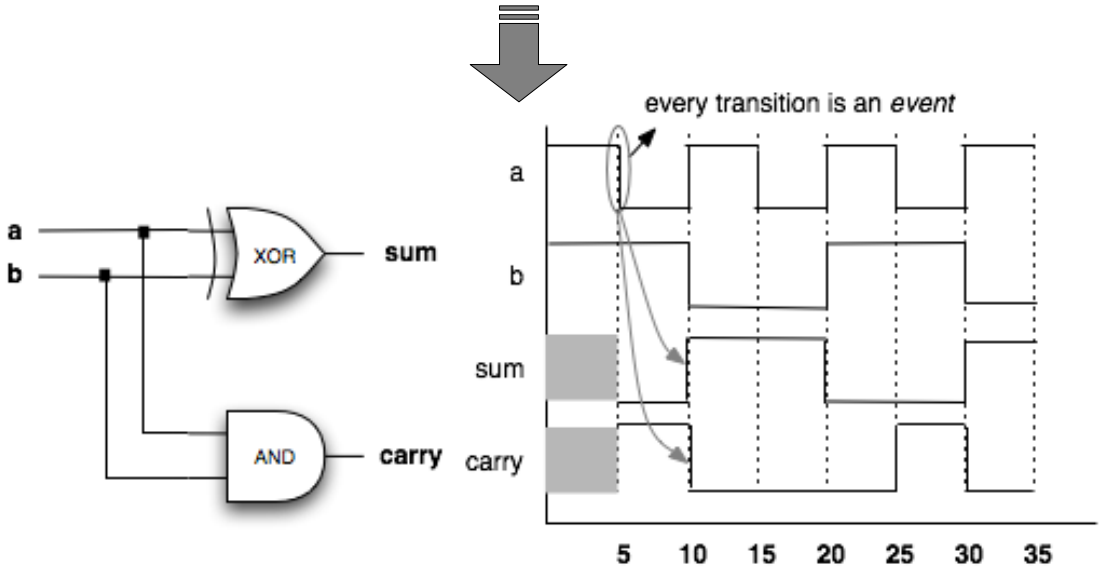
(Re-)Konfigurierbare Hardware

- Hardwarestrukturen werden heutzutage „programmiert“
  - Sprachen wie VHDL, Verilog oder SystemC

VHDL-Quelltext

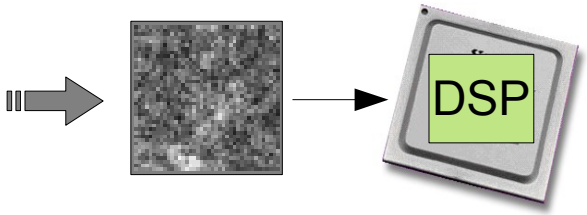
```
entity half_adder is port(
  a, b: in bit;
  sum, carry: out bit);
end half_adder;
```

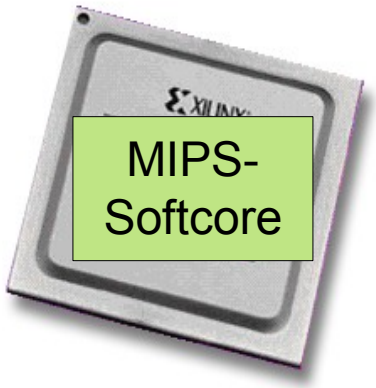
```
architecture half_adder_arch of
half_adder is
begin
  sum <= (a xor b) after 5 ns;
  carry <= (a and b) after 5 ns;
end half_adder_arch;
```

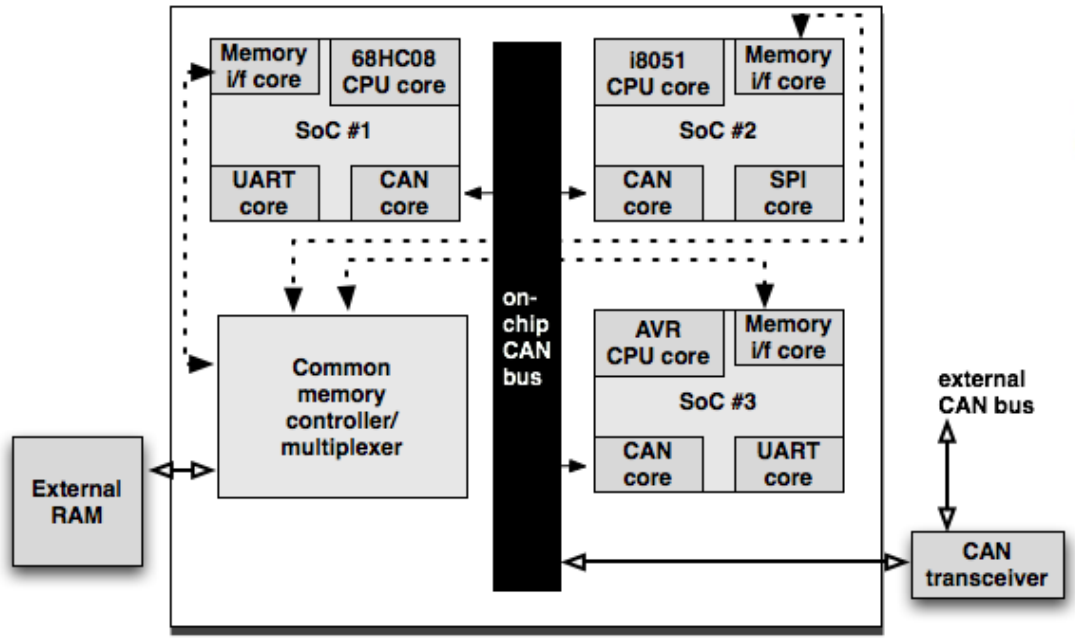


Maßgeschneiderte Hardware

FPGAs: universelle Maschinen mit programmierbaren Hardwarestrukturen
ASICs: für die Serienfertigung „in Maske“





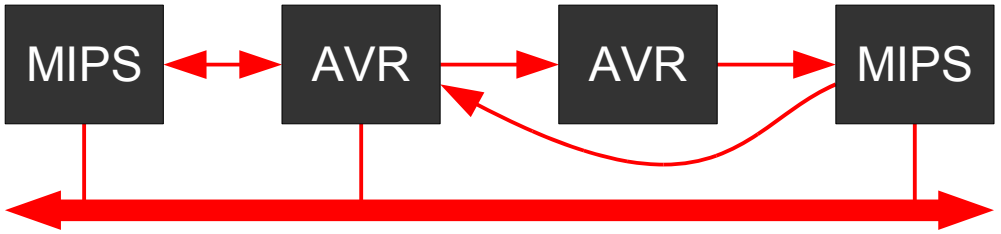


MPSoCs: Selbst heterogene Multiprozessorsysteme lassen sich mit heutiger Technik „programmieren“.

Vorzüge konfigurierbarer Hardware

… gegenüber klassischen CPUs/MPUs

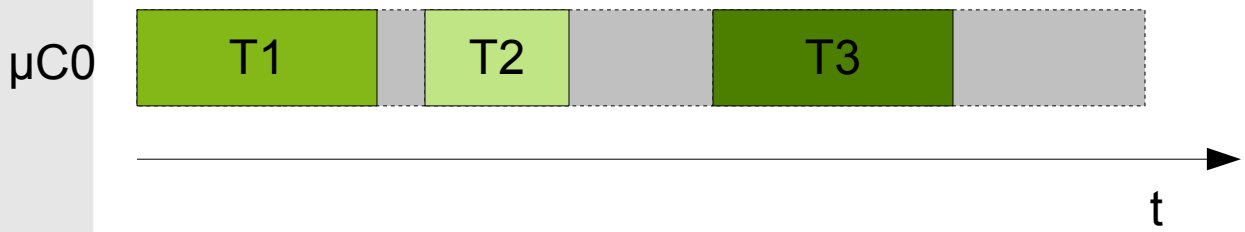
- Mehrere oder viele CPUs
  - Extremfall: Eine CPU pro Anwendungskontrollfluss
  - Viel Rechenleistung bei niedrigerem Takt
- Dedizierte Kommunikationsstrukturen
  - Vermeidung von Kollisionen



- *„Soft Cores“*
  - Langfristige Verfügbarkeit
  - Programmierung mit gewöhnlichen Compilern
  - Heterogene Systeme möglich
- Integration anwendungsspezifischer E/A-Controller
- Anwendungsspezifische Funktionseinheiten
  - Hohe Rechenleistung durch feingranulare Parallelität

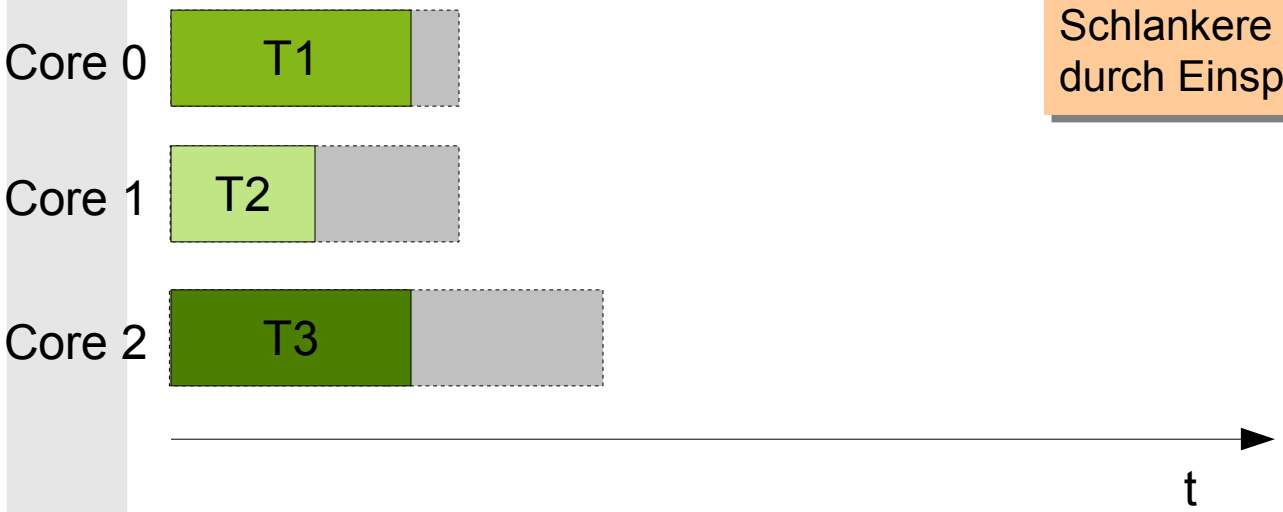
Vorzüge zusammengefasst

- Hohe Performance
  - Parallelverarbeitung
- Beste Echtzeiteigenschaften
  - Z.B. ein Task pro *Core* → minimaler *Jitter*, WCET gut bestimmbar
- Hohe Energieeffizienz
  - geringere Taktraten möglich



Vorzüge zusammengefasst

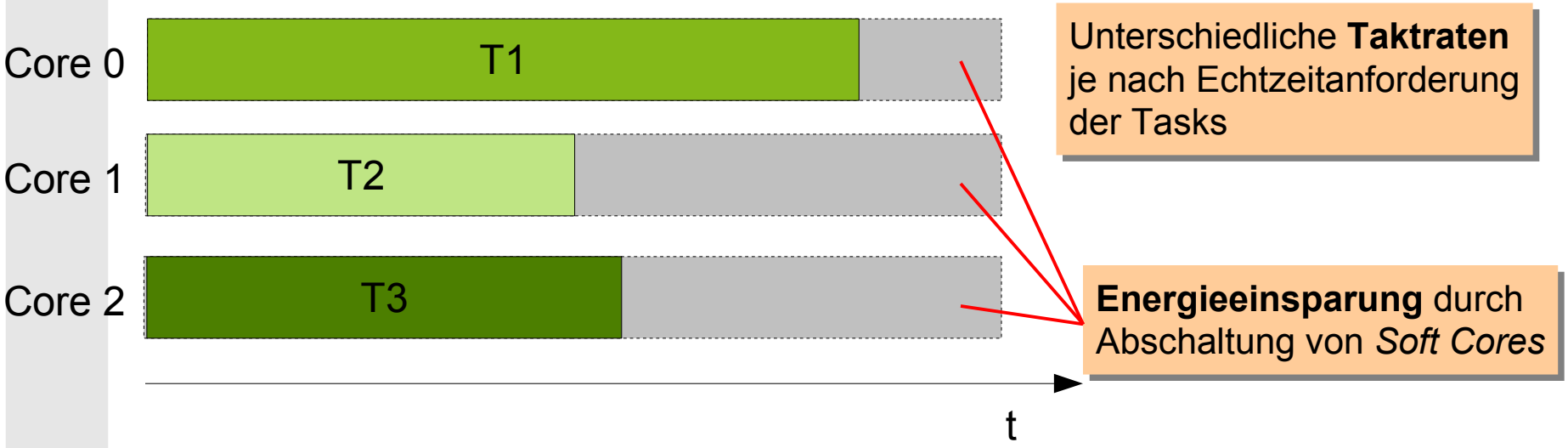
- **Hohe Performance**
  - Parallelverarbeitung
- **Beste Echtzeiteigenschaften**
  - Z.B. ein Task pro *Core* → minimaler *Jitter*, WCET gut bestimmbar
- Hohe Energieeffizienz
  - geringere Taktraten möglich



Schlankere **Systemsoftware**, z. B. durch Einsparung eines **Schedulers**

Vorzüge zusammengefasst

- Hohe Performance
  - Parallelverarbeitung
- Beste Echtzeiteigenschaften
  - Z.B. ein Task pro *Core* → minimaler *Jitter*, WCET gut bestimmbar
- **Hohe Energieeffizienz**
  - geringere Taktraten möglich



Probleme auf mehreren Ebenen

- **Hardware**
  - Hardware und Software: Getrennte Welten
  - Hardwareanforderungen werden direkt aus Anwendung abgeleitet
  - Betriebssystem bleibt außen vor
  - Nur Rechnerarchitekten verstehen Kunst konfigurierbarer Hardware
- **Betriebssystem**
  - Kein Betriebssystem passt so richtig auf die heterogene Multiprozessor-Hardware
  - Klassische Betriebssystemdienste oft unnötig
- **Anwendung**
  - Applikationen greifen direkt auf Hardware zu → keine Portabilität
  - Aufwändige Hardware-Simulationsumgebungen
  - Viel *Know-How* erforderlich

Probleme auf mehreren Ebenen

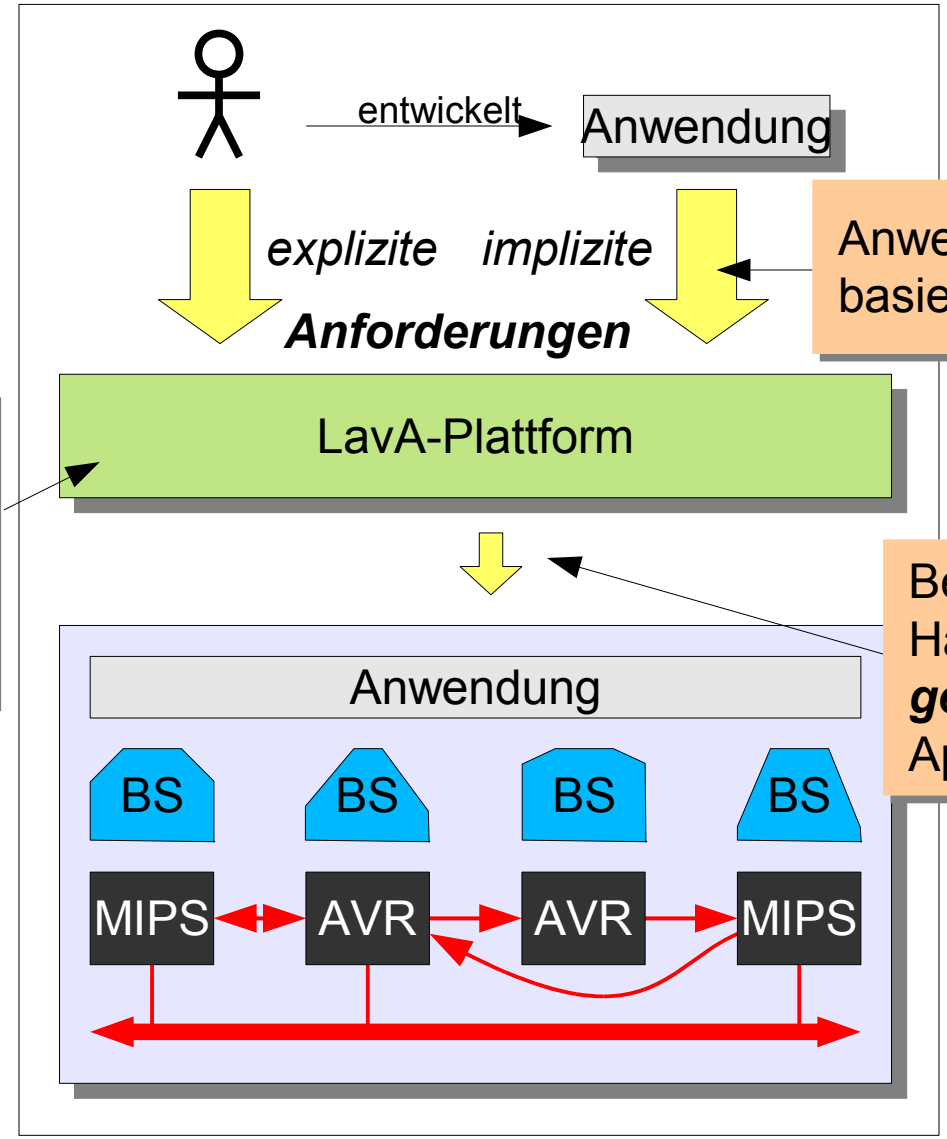
- **Hardware**
  - Hardware und Software: Getrennte Welten
  - Hardwareanforderungen werden direkt aus Anwendung abgeleitet
  - Betriebssystem bleibt außen vor
  - Nur Rechnerarchitekten verstehen Kunst konfigurierbarer Hardware
- **Betr**
  - Kei
    auf die heterogene Multiprozessor-Hardware
  - Klassische Betriebssystemdienste oft unnötig
- **Anwendung**
  - Applikationen greifen direkt auf Hardware zu → keine Portabilität
  - Aufwändige Hardware-Simulationsumgebungen
  - Viel *Know-How* erforderlich

Unnötig lange und teure Entwicklungsprozesse

Überblick

- Konfigurierbare Hardware
  - Vorzüge und Probleme
- **Das LavA-Projekt**
  - **Ansatz**
  - **Erste Systembausteine**
- Fazit und Ausblick

Ansatz: BS/HW-*Co-Konfiguration*

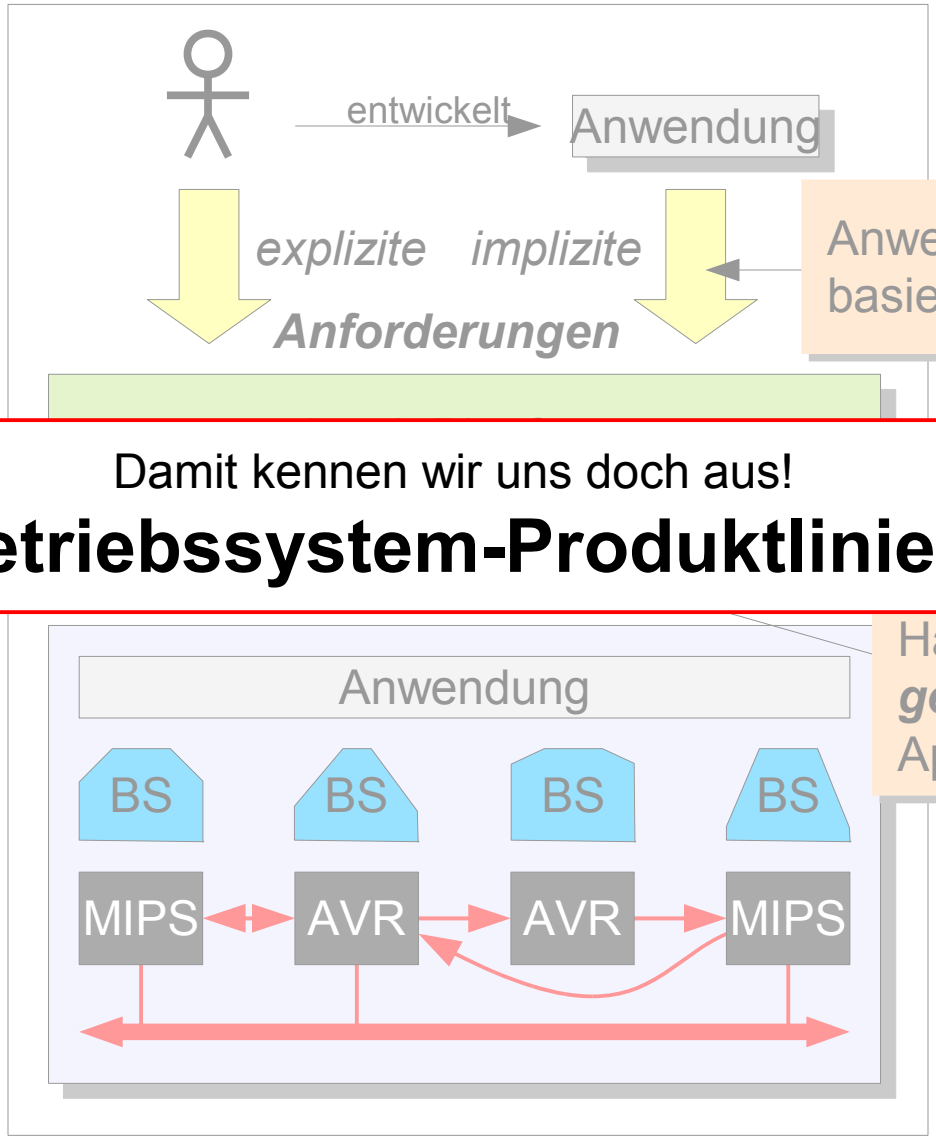


Anwendungsentwicklung basiert auf ***BS-API***

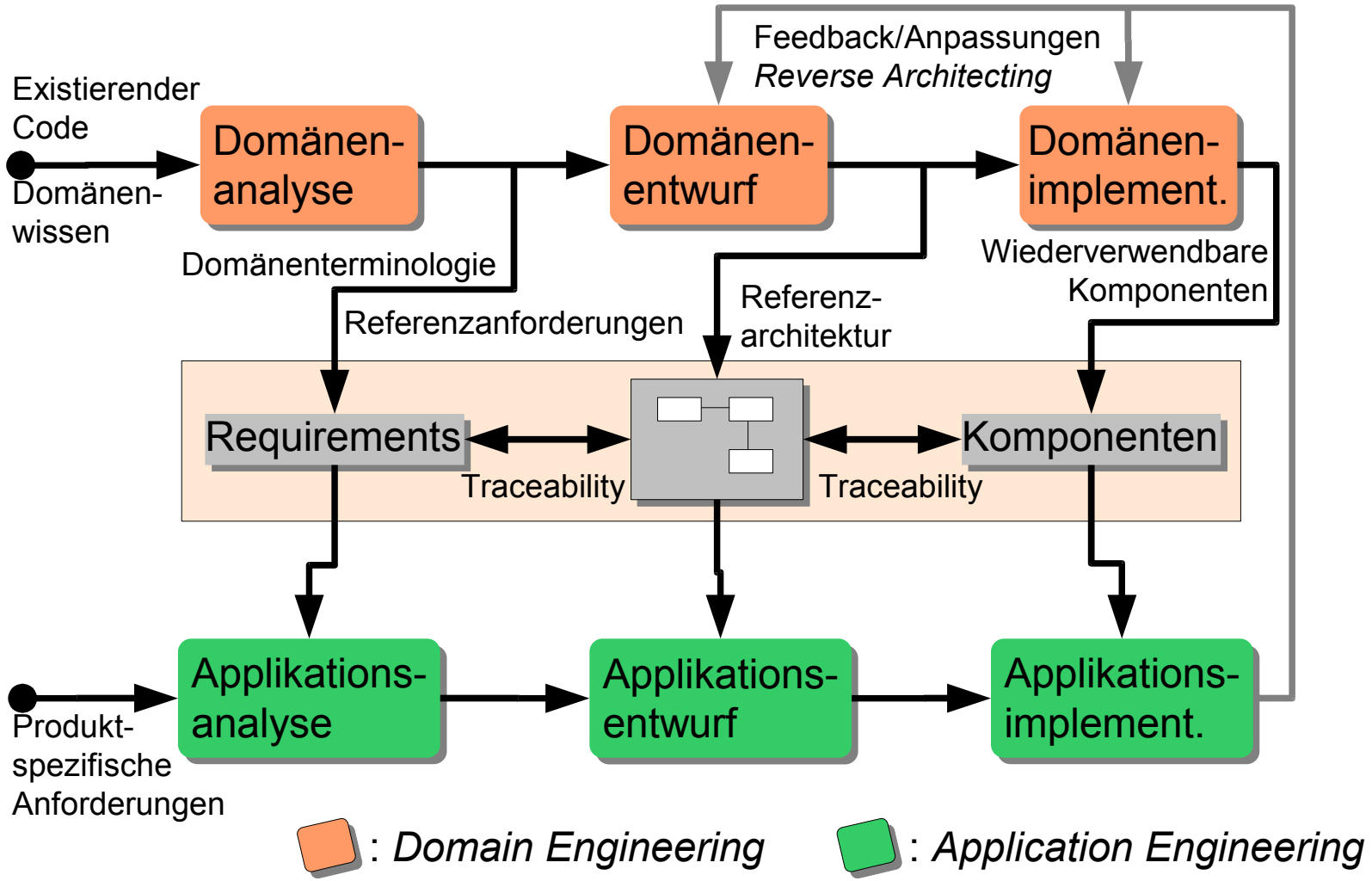
Generische Hardware- ***und*** Betriebssystem-komponenten bildendie Plattform

Betriebssystem und Hardware werden ***gemeinsam*** aus der Applikation abgeleitet

Ansatz: BS/HW-*Co-Konfiguration*

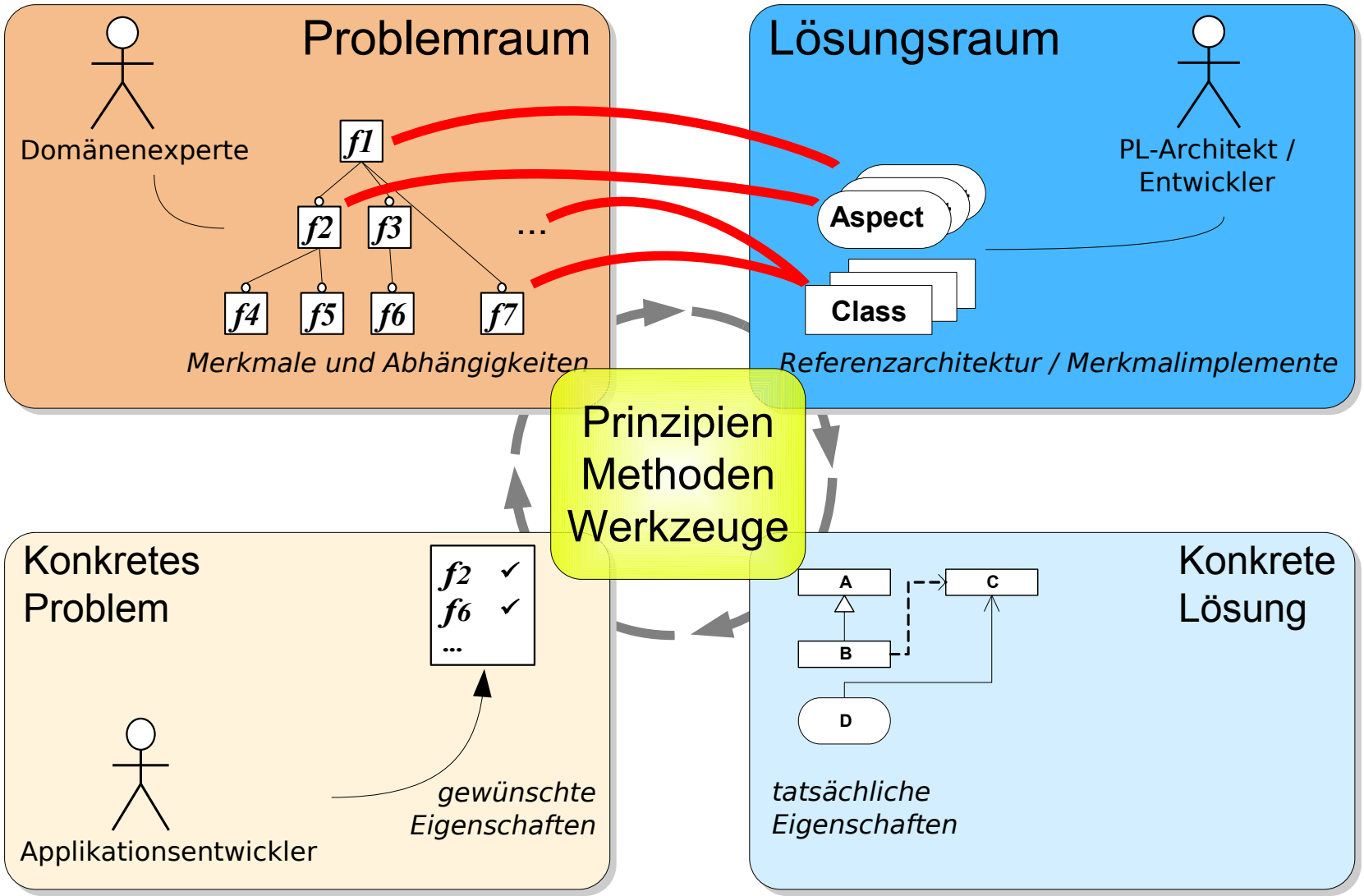


Software-Produktlinienentwicklung



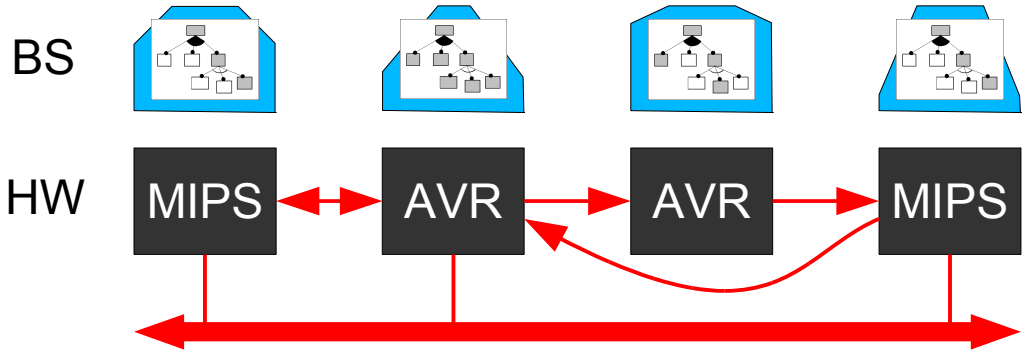
Referenzprozess für die Software-Produktlinienentwicklung [1]

Produktlinienentwicklung



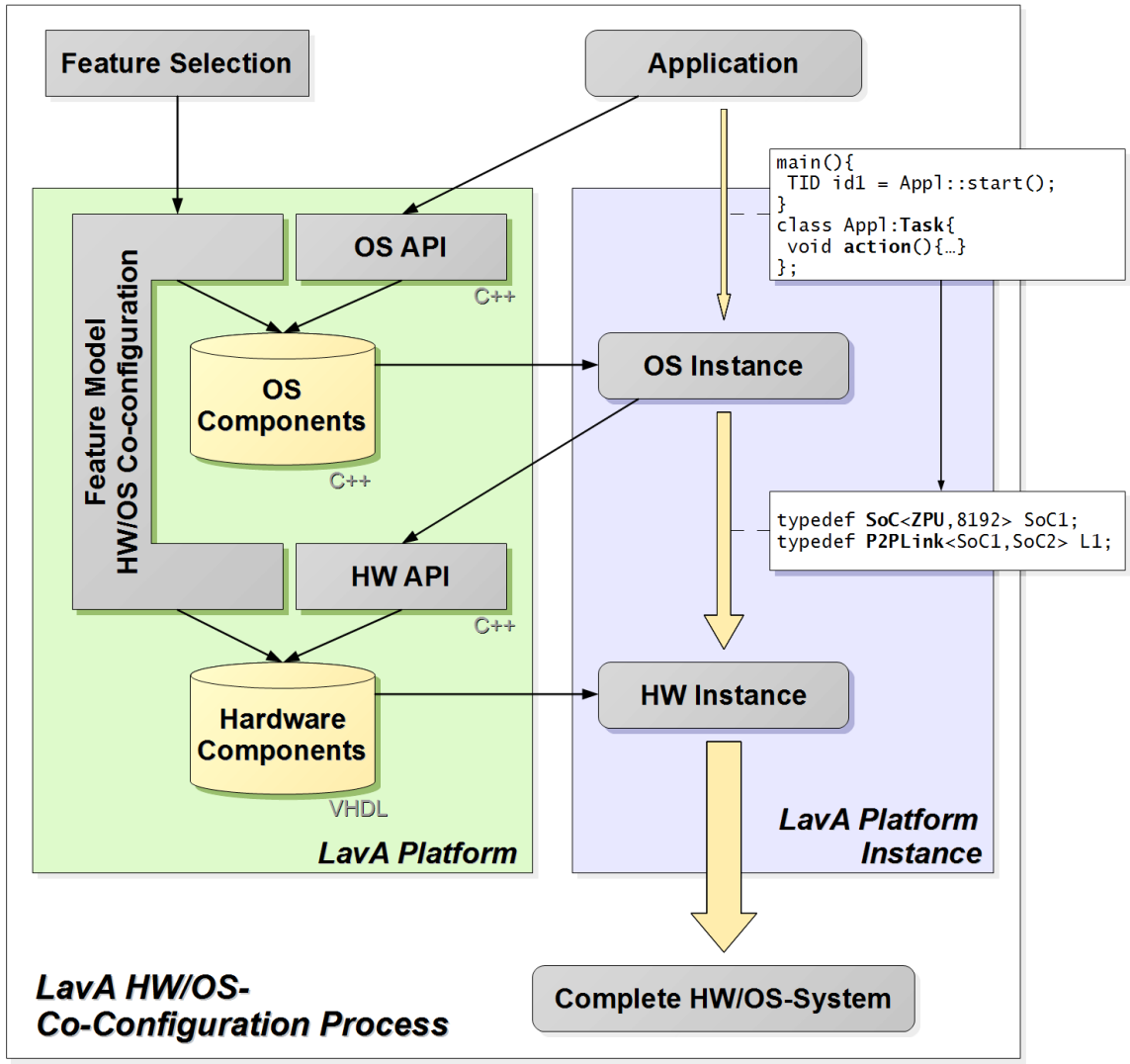
Herausforderungen bei LavA

- Komposition von Produktlinien
  - Vertikal (BS/HW)
  - Horizontal (BS_1..BS_N)



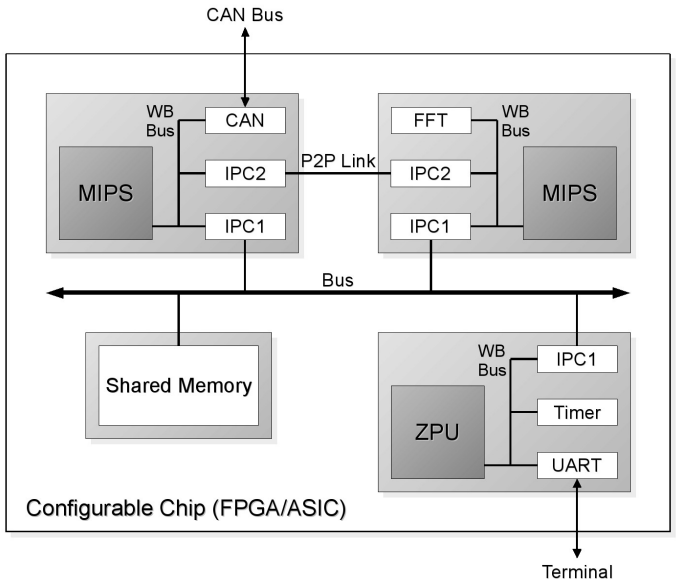
- Konfigurierung von Hardwarestrukturen
  - Merkmalmodelle sind nicht adäquat
  - Automatisierung ist nötig
- Generische Hardwarekomponenten
  - Bei reinem VHDL nur eingeschränkt möglich
- Ressourcenbeschränkungen und -optimierung
  - Neue Ressourcenarten („LUTs“, …)
  - Behindert *top-down* Konfigurierung

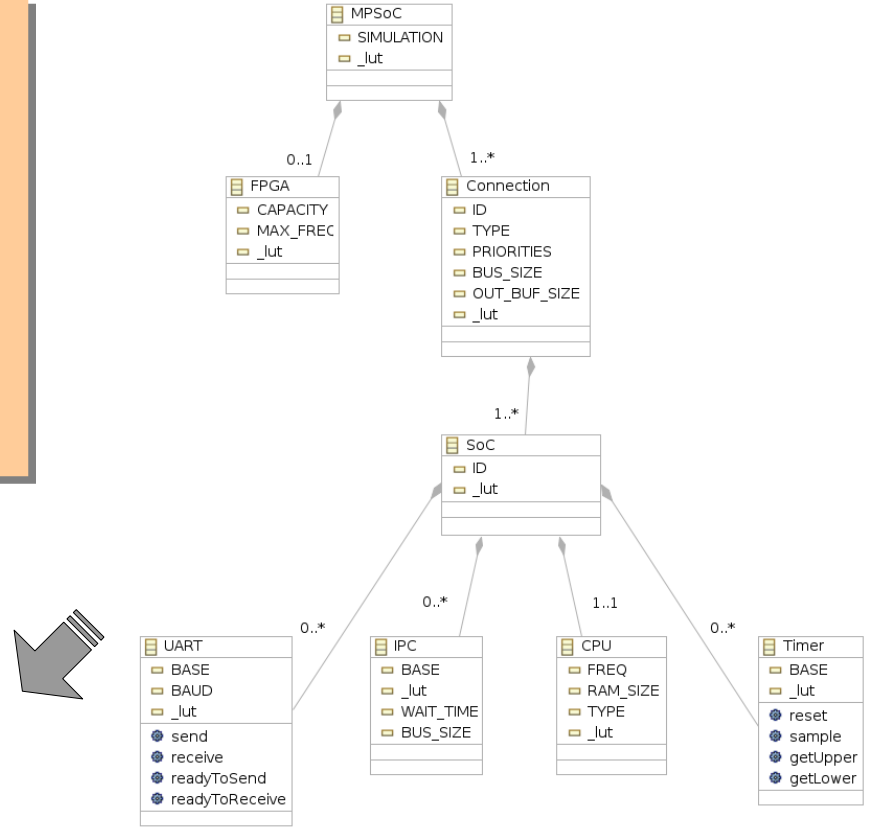
Der LavA-Prozess im Überblick



LavA-Hardware-Produktlinie

- Ein Metamodell beschreibt die Kompositionsregeln
- Diverse Komponenten
  - 3 *Softcore*-Typen
  - Busse, Ringe, Punkt-zu-Punkt
  - *Shared*- und *Local Memory*





- Ca. 100 ZPU Cores passen auf einen Virtex 5 FPGA

Generische Hardwarekomponten

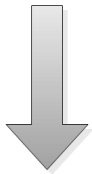
- Nutzung eines *Frame*-Prozessors (XVCL)

```
<x-frame name="SPC" outfile="MPSoC.vhd">
    <adapt x-frame="MPSoC">
        <insert break="SIGNALS">
            <select option="UART">
                <option value="1">
        uart_rx      : in  std_logic;
        uart_tx      : out std_logic;
                </option>
            </select>
        </insert>
    </adapt>
</x-frame>
```

```
<x-frame name="MPSoC">

entity MPSoC is
    port(
        clock        : in  std_logic;
        reset        : in  std_logic;
    <break name="SIGNALS"/>
    );
end MPSoC;

</x-frame>
```

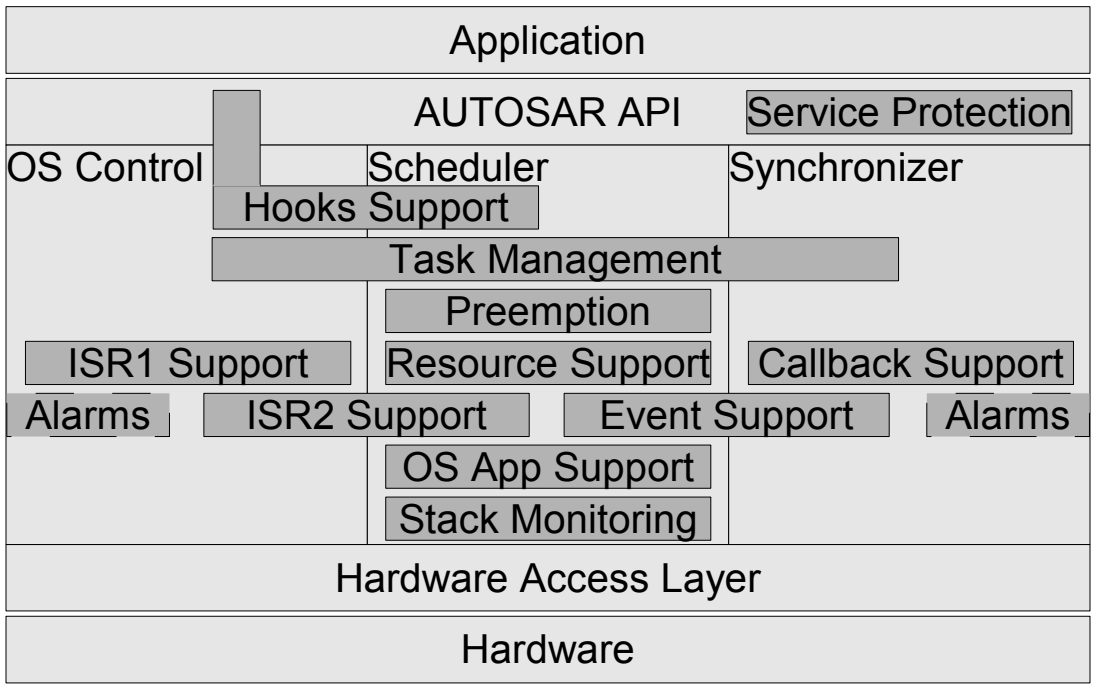


```
entity MPSoC is
    port(
        clock        : in  std_logic;
        reset        : in  std_logic;
        uart_rx      : in  std_logic;
        uart_tx      : out std_logic;
    );
end MPSoC;
```

- Aspektorientierte Hardwarestrukturen (AspectVHDL)
  - in Arbeit!

Das LavA-BS: CiAO (Uni ER/DO)

- **CiAO/AUTOSAR-Schnittstelle**
  - Systemressourcen zur Übersetzungszeit bekannt
  - Adressraumtrennung bei „*non-trusted applications*“
    - Explizite Kommunikation über Nachrichten
  - Standard im Bereich automotiver Systeme
  - Klein und effizient
- **Konfigurierbar**
  - Merkmalmodell
- **Generisch**
  - C++ Templates
  - Aspektorientierte Programmierung mit AspectC++



Fazit und Ausblick

- Ziel: **Vereinfachung**
  … der Entwicklung von anwendungsspezischen eingebetteten Multiprozessorsystemen auf einem Chip
- Ansatz: **Betriebssystemschnittstelle**
  … rückt (wieder) ins Zentrum der Betrachtung des Anwendungsentwicklers
- Weg: Know-How über **Systemsoftware-Produktlinien**
  … wird auf konfigurierbare Hardware übertragen
  - Plattformproduktlinie: Komposition von Hardware- und Systemsoftware
    - durchgängige Konfigurierung
  - Programmierung generischer (Hardware-)Komponenten
    - *Frame*-Prozessor, AspectVHDL
- … die Bausteine müssen nur noch zusammengefügt werden.

Links

- **Arbeitsgruppe eingebettete Systemsoftware der TU Dortmund**
  http://ess.cs.tu-dortmund.de
- **Projekt LavA**
  http://ess.cs.tu-dortmund.de/Research/Projects/LavA/
- **Projekt CiAO**
  http://www4.informatik.uni-erlangen.de/Research/CiAO/

Vielen Dank für die Aufmerksamkeit!