Karlsruher Institut für Technologie

Institut für Prozessdatenverarbeitung und Elektronik

Institut für Technik der Informationsverarbeitung

# 1. Einführung & Motivation

**Dr.-Ing. Oliver Sander**
**Dipl.-Inform. Leonard Masing**



# Hardware/Software Co-Design

# Informationstechnische Systeme

**Wir sind von mehr informationstechnischen Systemen umgeben als wir gemeinhin sehen. „Eingebettete elektronische Systeme“**





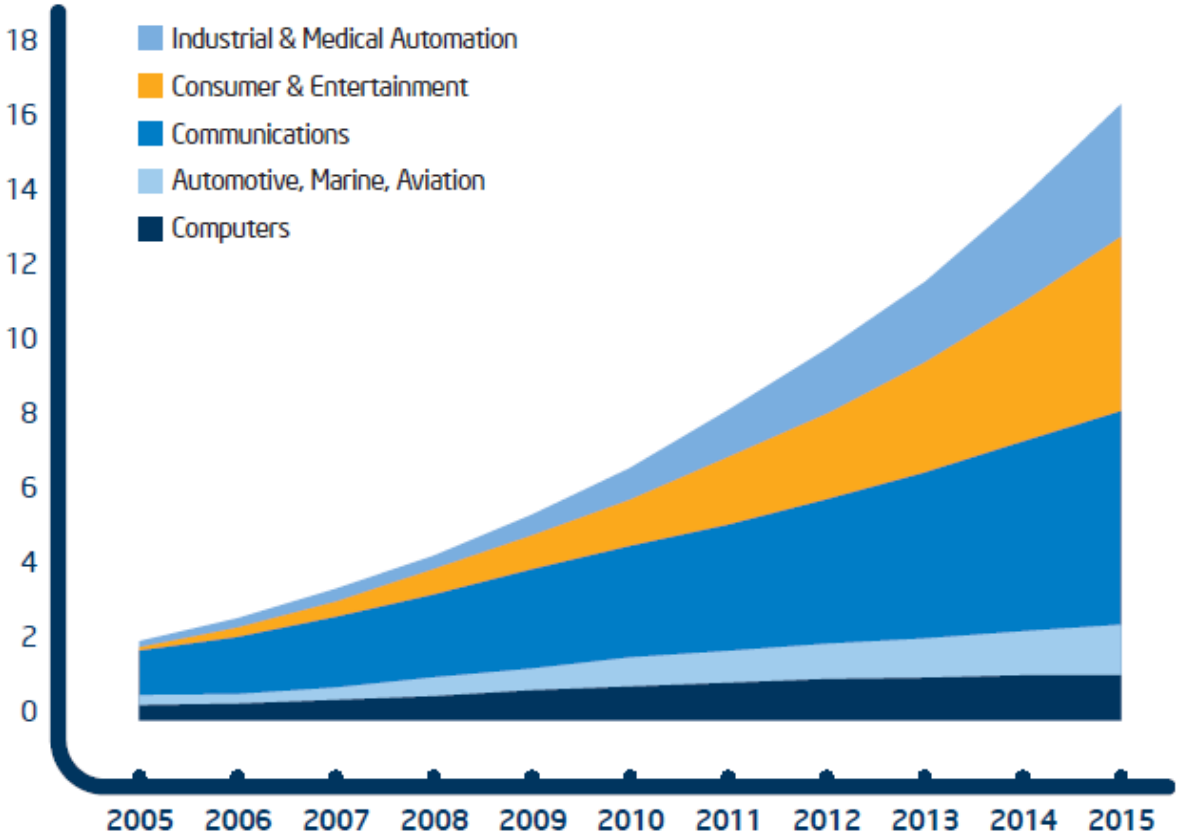







Oliver Sander
Hardware/Software Co-Design

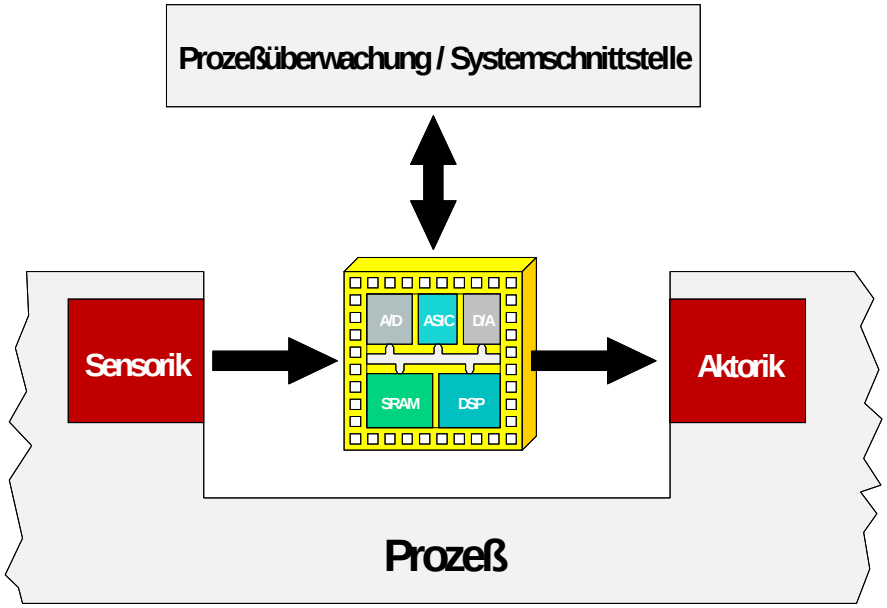
# Prozessorstückzahlen




- Entwicklung der weltweiten Anzahl an Prozessoren in den verschiedenen Domänen (in Milliarden Stück) [Quelle: Intel Developer Conference Jan. 2009]

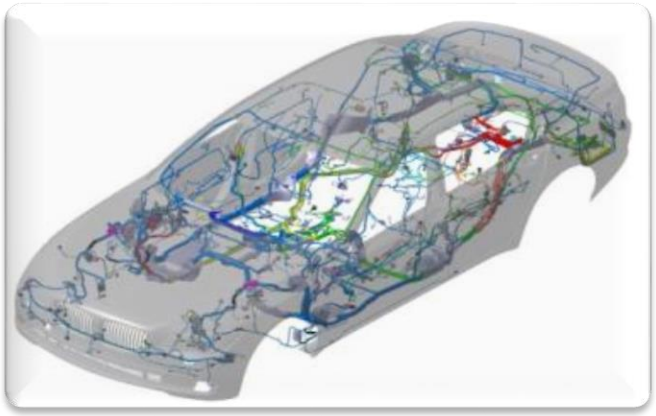
# Eingebettetes System (Embedded System)

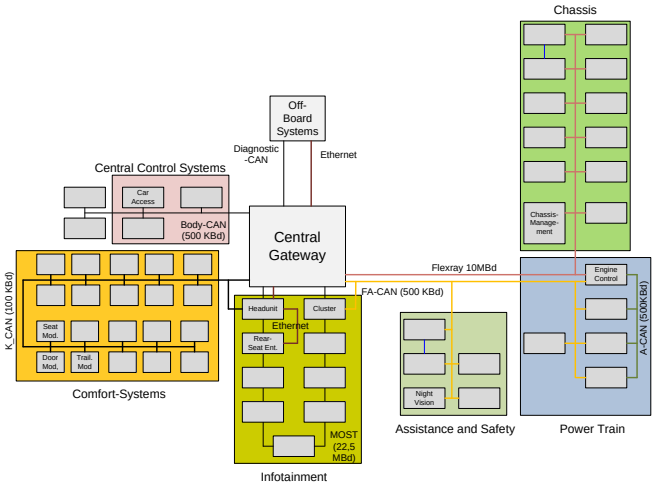
- Der Ausdruck eingebettetes System bezeichnet einen elektronischen Rechner oder auch Computer, der in einen technischen Kontext eingebunden (eingebettet) ist. Dabei übernimmt der Rechner entweder Überwachungs-, Steuerungs- oder Regelfunktionen oder ist für eine Form der Daten- bzw. Signalverarbeitung zuständig.
- Eingebettete Systeme verrichten – **weitestgehend unsichtbar für den Benutzer** – den Dienst.
- Beispiele: Mobiltelefone, Automobil- und Industriesteuerungen, Unterhaltungselektronik, Digitales Fernsehen, Netzwerkrouter, Multimedia-Anwendungen, Ubiquitäres Computing, etc.



# Komplexe eingebettete Systeme

- Im Fall von komplexen Gesamtsystemen handelt es sich dabei meist um eine Vernetzung einer Vielzahl von ansonsten autonomen, eingebetteten Systemen (z. B. im Fahrzeug oder Flugzeug).






## Wiring harness

| | |
|---|---|
| Length | ca. 2700 m |
| Connecting Plugs | ca. 500 |
| Weight | ca. 40 kg |

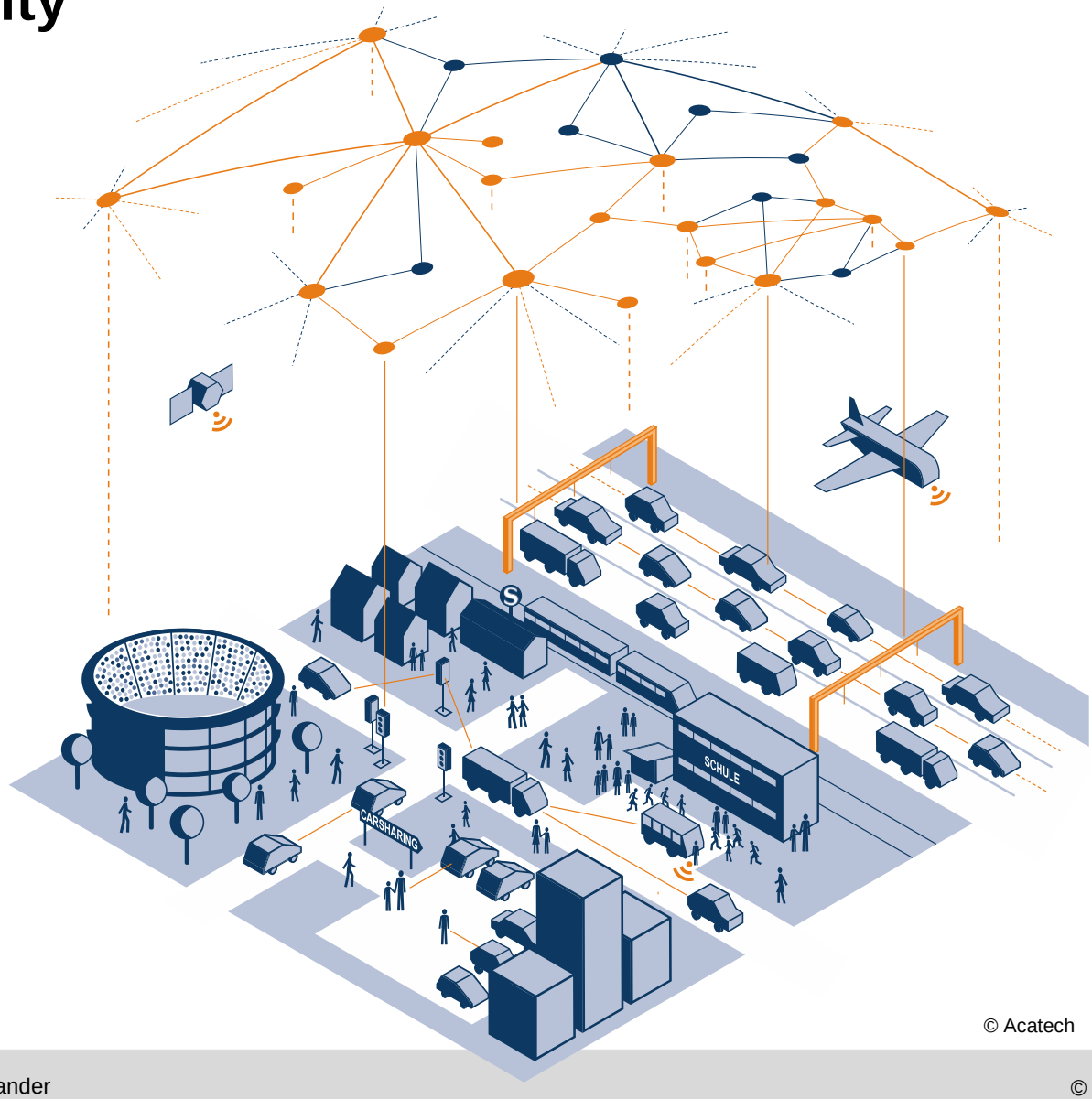
## Electronic Control Units (ECU)

| | |
|---|---|
| No. ECUs | 28 … 74 |
| CPUs | ca. 230 |
| GPUs | > 5 |
| Power PCs | 3 |
| Busses | CAN, LIN, MOST*, Flex Ray, Ethernet |

## Other

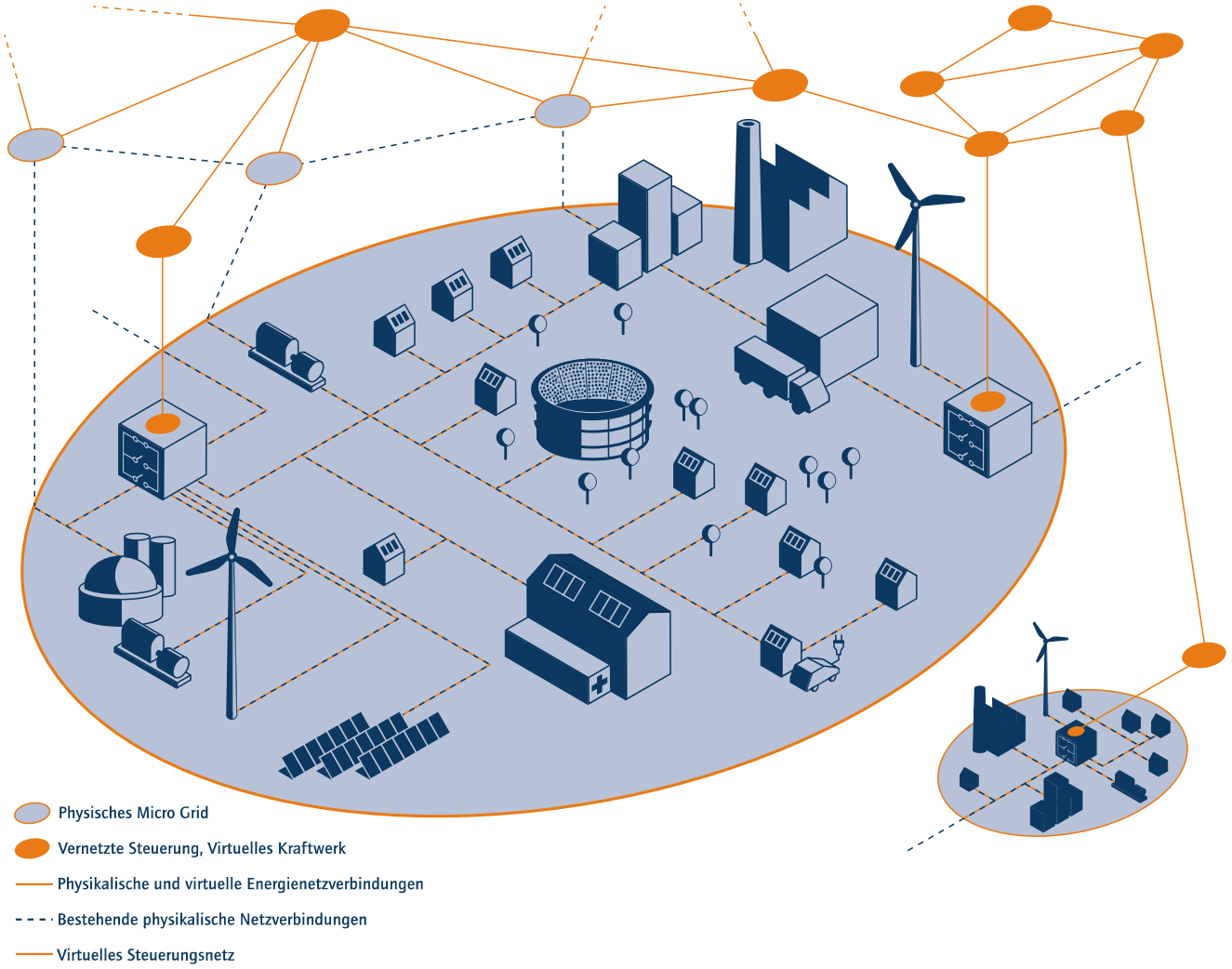
| | |
|---|---|
| Software | 3-5 GB Application<br>15-20 GB Data |

# Cyber Physical Systems (CPS) Scenario – Smart Mobility




© Acatech

# CPS Scenario - Smart Grid




© Acatech

# Begriff des Hardware-Software Co-Design (I)

- **Hardware-Software Co-Design ist der gleichzeitige und verzahnte Entwurf von Hardware- und Softwareteilen eines Systems**
  - Ziel: geringere Entwicklungszeit + Kosten
- Eingebettete Systeme bestehen aus kooperierenden Hardware- und Softwarekomponenten
  - unterschiedliche Zieltechnologien
- Bilden ein optimiertes System aus Hardware, Software und einer Kommunikationsstruktur.
- Erfordert Partitionierung der gegebenen Systemspezifikation, Kommunikationssynthese, Cosimulation sowie Compilierung der SW-Komponenten und Synthese der HW.

# Begriff des Hardware-Software Co-Design (II)

- Analyse von HW/SW-Grenzen sowie Bewertung von Entwurfsalternativen
  - Kosten (SW, HW, Entwicklung)
  - Performanz (Echtzeitanforderungen)
  - Verlustleistung (mobile Geräte)
  - Time-to-Market
  - Flexibilität (Multi-Purpose)

- Co-Simulation und Emulation (Rapid Prototyping) des entworfenen HW/SW-Systems (incl. HW/SW-Kommunikationsarchitektur) liefert realistische Performanz-Daten zur Optimierung (gegebenenfalls Re-Partitionierung) sowie zur Systemintegration

# Begriff des Hardware-Software Co-Design (III)

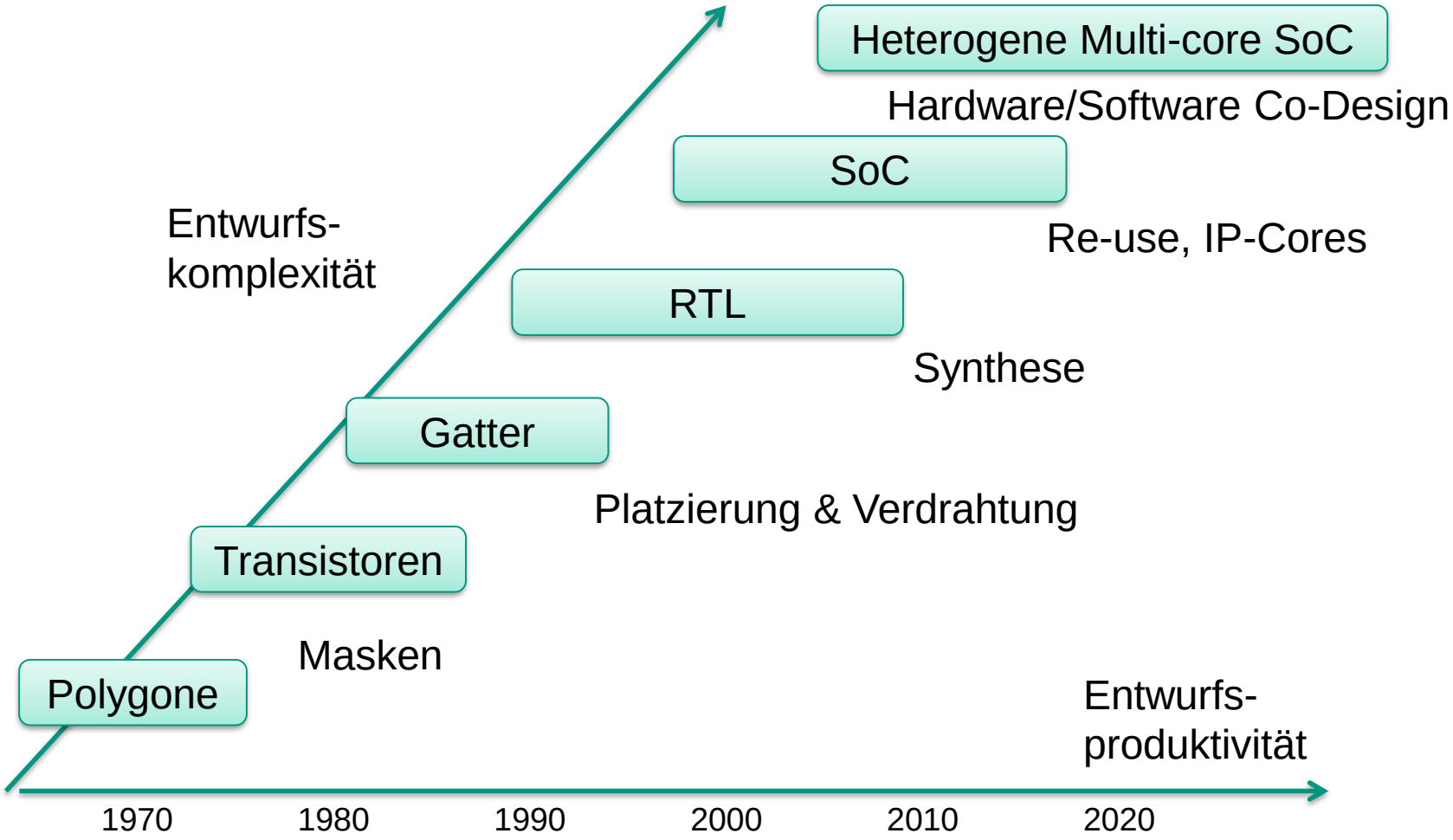
- Erfahrener Systementwickler kann für manche Komponenten oftmals gezielt entscheiden, ob HW- oder SW-Implementierung besser ist
  - spezielles Anwendungswissen
- Interaktive Strategie notwendig und vorteilhaft
  - Erfahrungswissen
- Spezielle Partitionierungsstrategien für unterschiedliche Anwendungsbereiche sinnvoll
  - Kriterien der Kostenfunktionen
  - Granularität, Intellectual Property (IP) Blöcke
  - Anwendungswissen ausnutzen!

# Begriff des Hardware-Software Co-Design (IV)

- Rasante Fortschritte in der Mikroelektronik machen Eingebettete Systeme zunehmend komplexer
  - zukünftig deep-submicron Technologien
- Einsatz von entsprechenden rechnergestützten Entwurfswerkzeugen ist notwendig
  - IP-basiert
    - um zunehmende Komplexität handhaben zu können
    - um die Entwurfskosten und die Entwurfszeit (time-to-market) zu senken
- Fortschritte in formalen / automatisierbaren Entwurfsmethoden
  - Automatisierung auf Systemebene wird möglich
  - Formale Validierung auf Systemebene
    - “gültige“ Entwurfsraumregionen
- Vorlesung Hardware/Software Co-Design:
  - notwendige multi-kriterielle Methoden und Verfahren
  - mögliche Hardware/Software Zielarchitekturen.

# Produktivitätsgrenze (I)

- Komplexe eingebettete Systeme erfordern neue Entwurfsmethoden

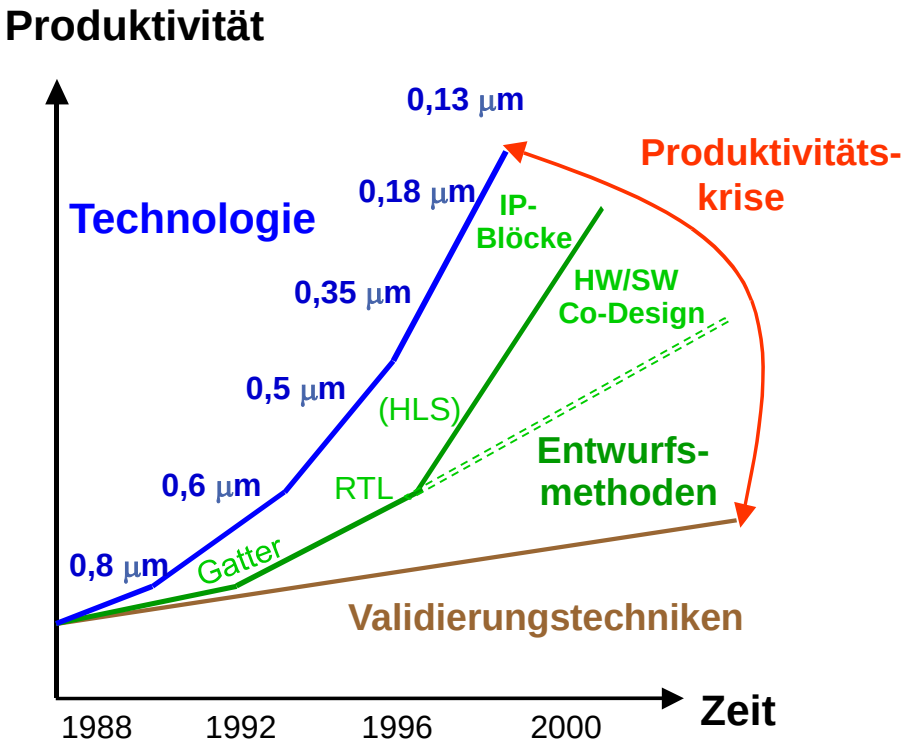


# Produktivitätsgrenze (II)

- Komplexe eingebettete Systeme benötigen neue Entwurfsmethoden
  - > 50% Entwicklungszeit für Simulation
  - > 25% Validierung/Verifikation

- Komplexität größer als Produktivitätssteigerung
  - Rapid Prototyping
  - HW/SW Co-Simulation
  - HW/SW Co-Verifikation
  - IP-basierter Entwurf

- Neue effiziente CAD-Methoden
  - Entwurfsraums Exploration

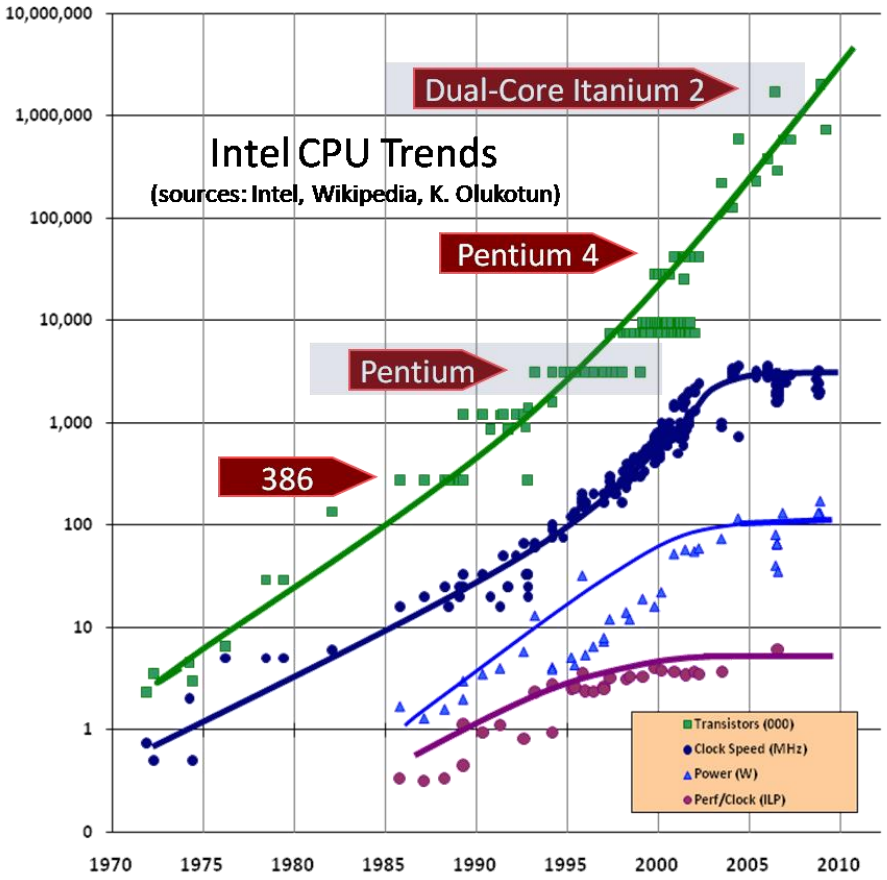


# Free Lunch is over

- Power, Performance & Area (PPA) Trade-Offs im Chip Design

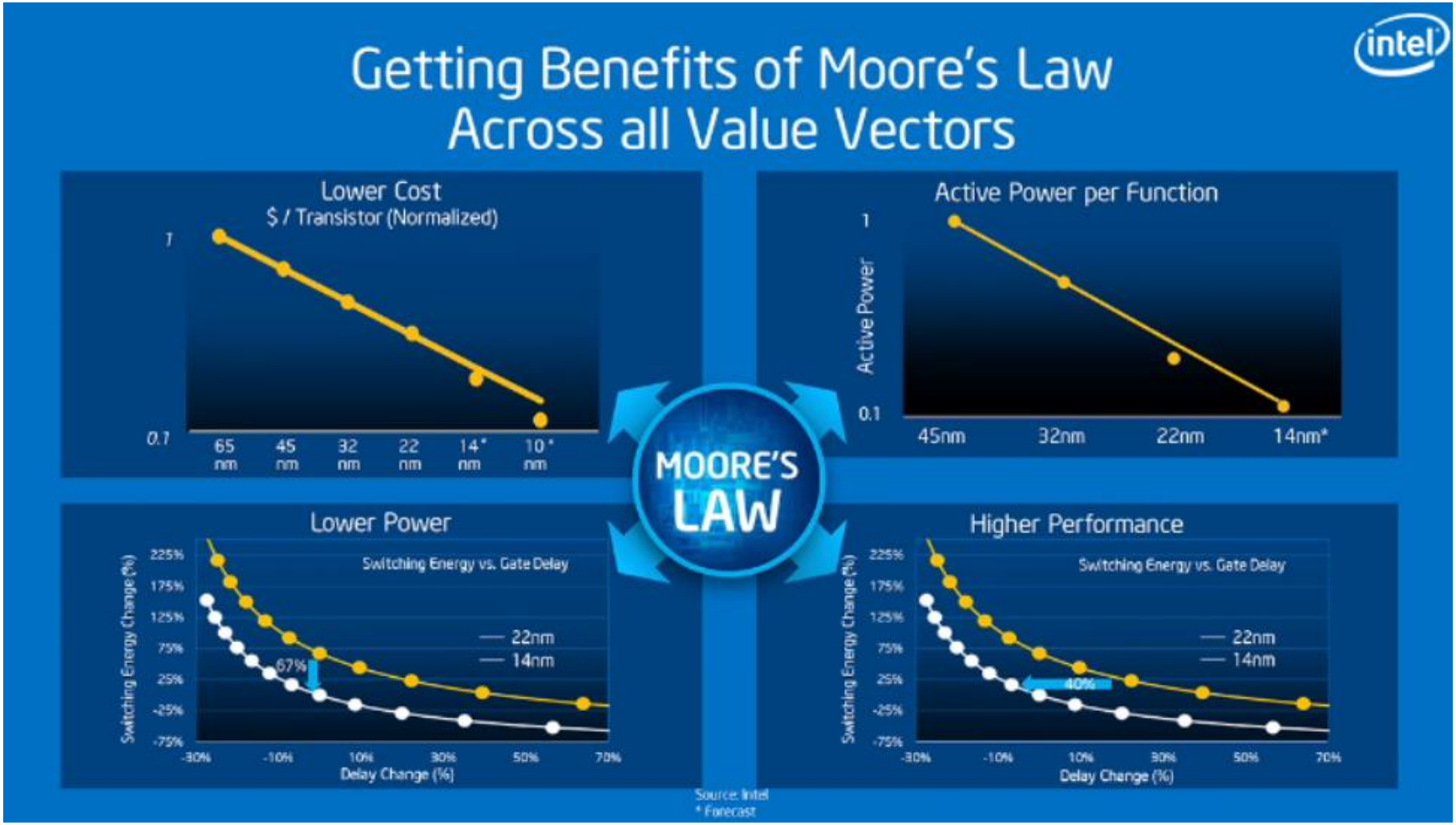
- Frequenz skaliert nicht mehr
  - Keine 10 GHz Systeme
- Fläche skaliert bald nicht mehr
  - Dark silicon
    - Deep sub-micron Effekte
- Verlustleistung
  - Statische und dynamische Verlustleistung vergleichbar

- Ausweg:
  - Nebenläufigkeit der Software
    - Multi-/Many-Core



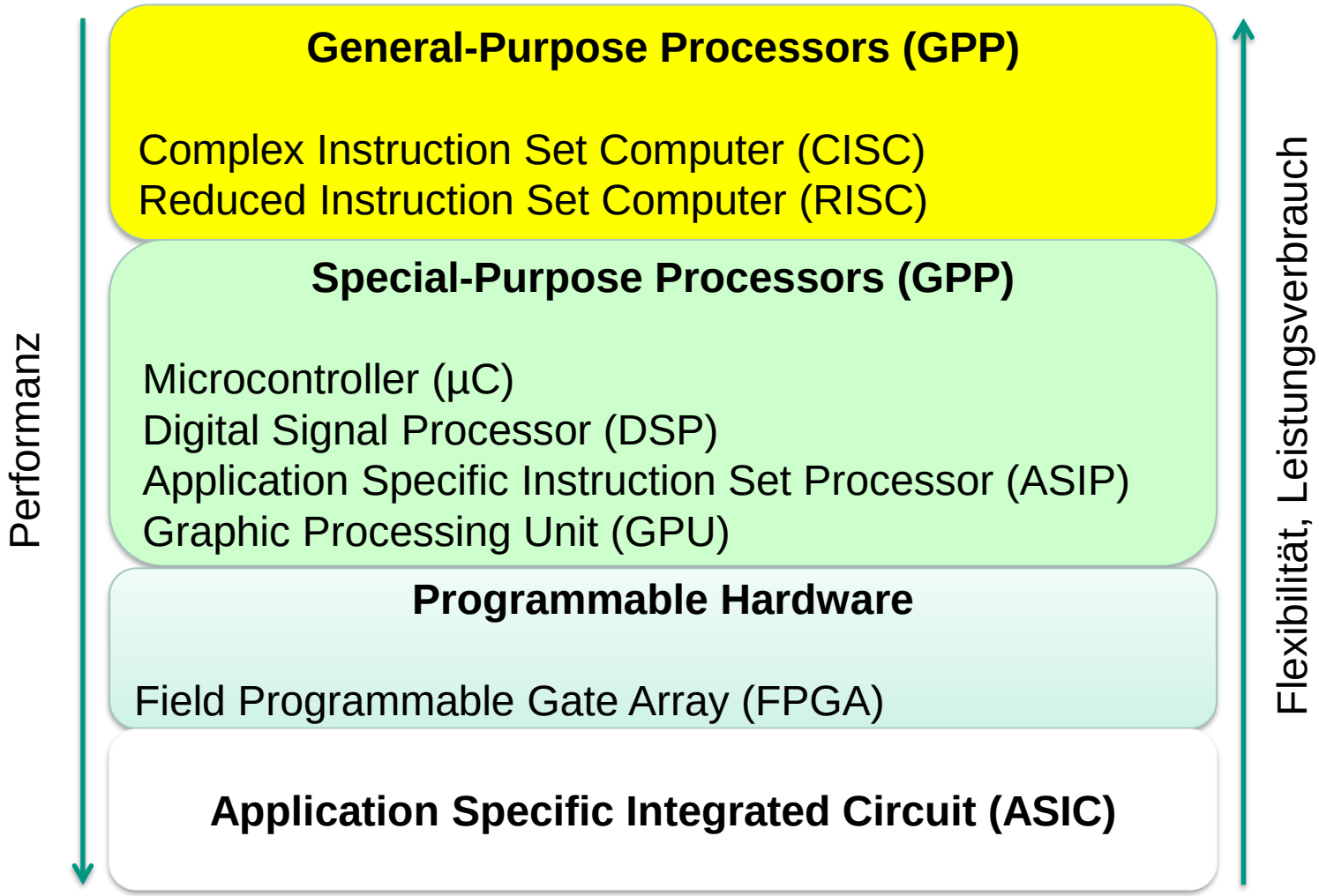

http://www.gotw.ca/publications/concurrency-ddj.htm

# Moore‘s Law & aktuelle Fertigungstechnologie

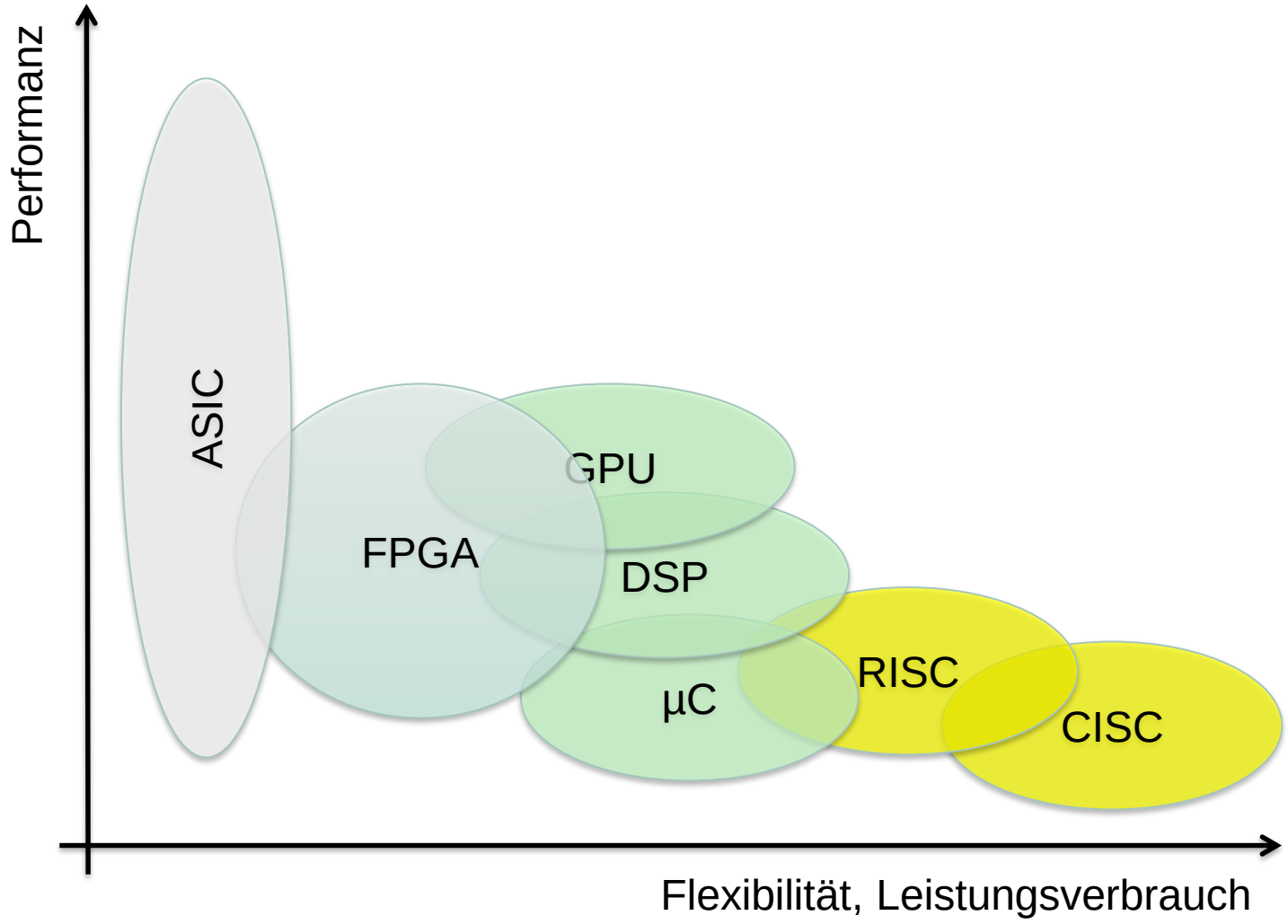



© Intel

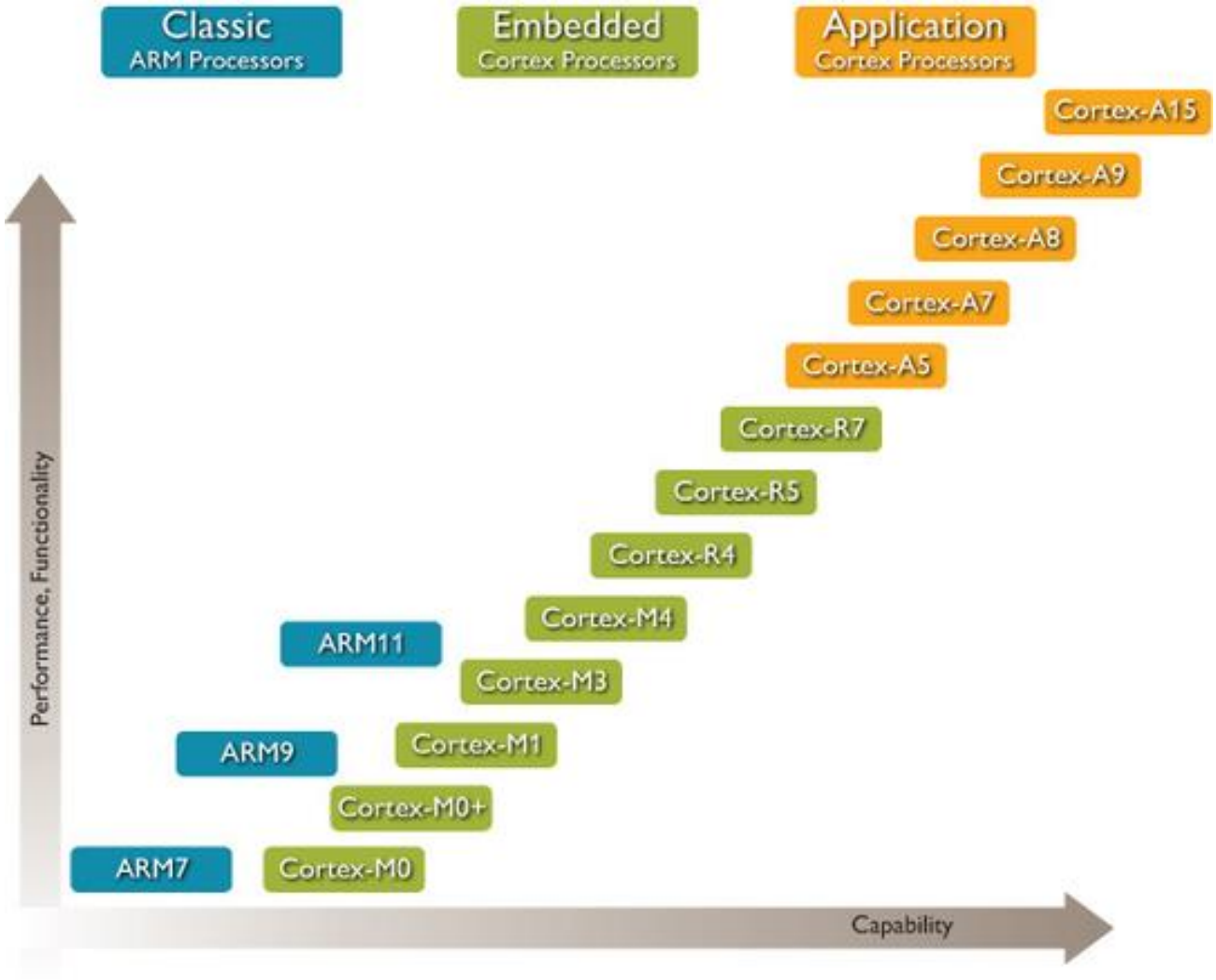
# Vergleich der Zielarchitekturen



# Vergleich der Zielarchitekturen



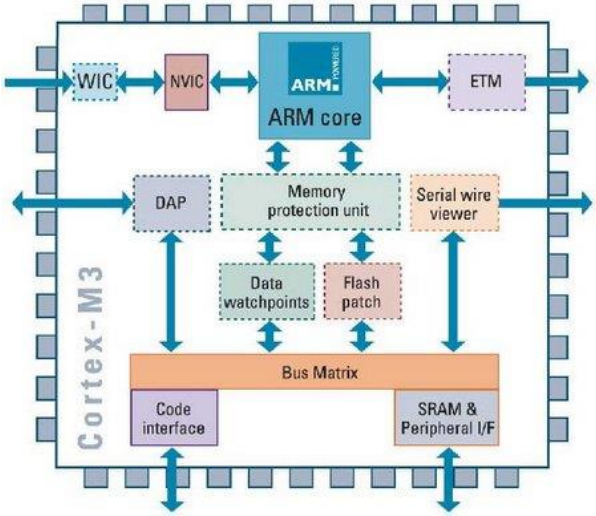
# Beispiel RISC: ARM Prozessoren (I)

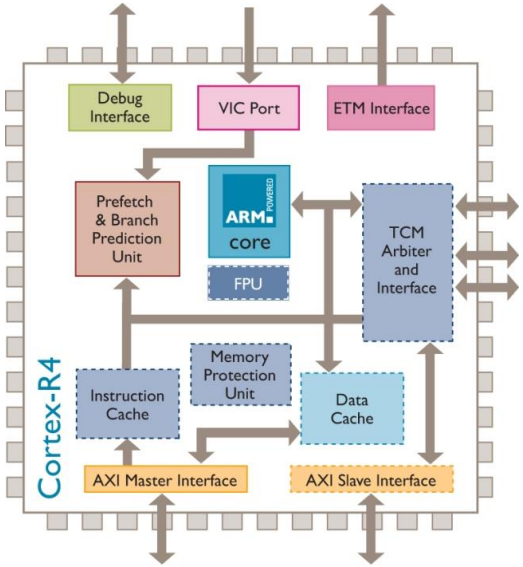


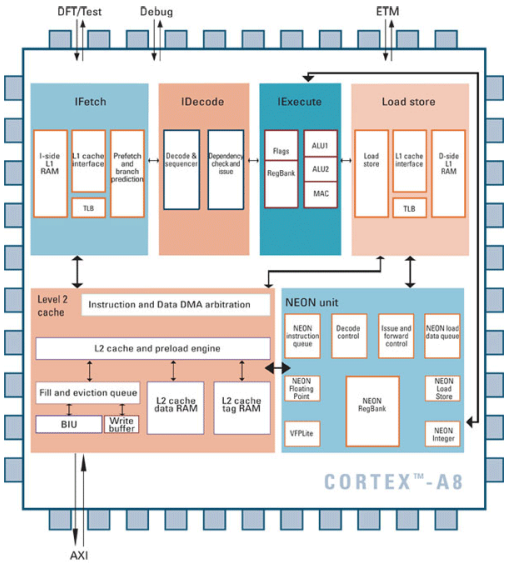

http://www.arm.com/products/processors/index.php

# Beispiel RISC: ARM Prozessoren (II)

- ARM Cortex Prozessor Familie



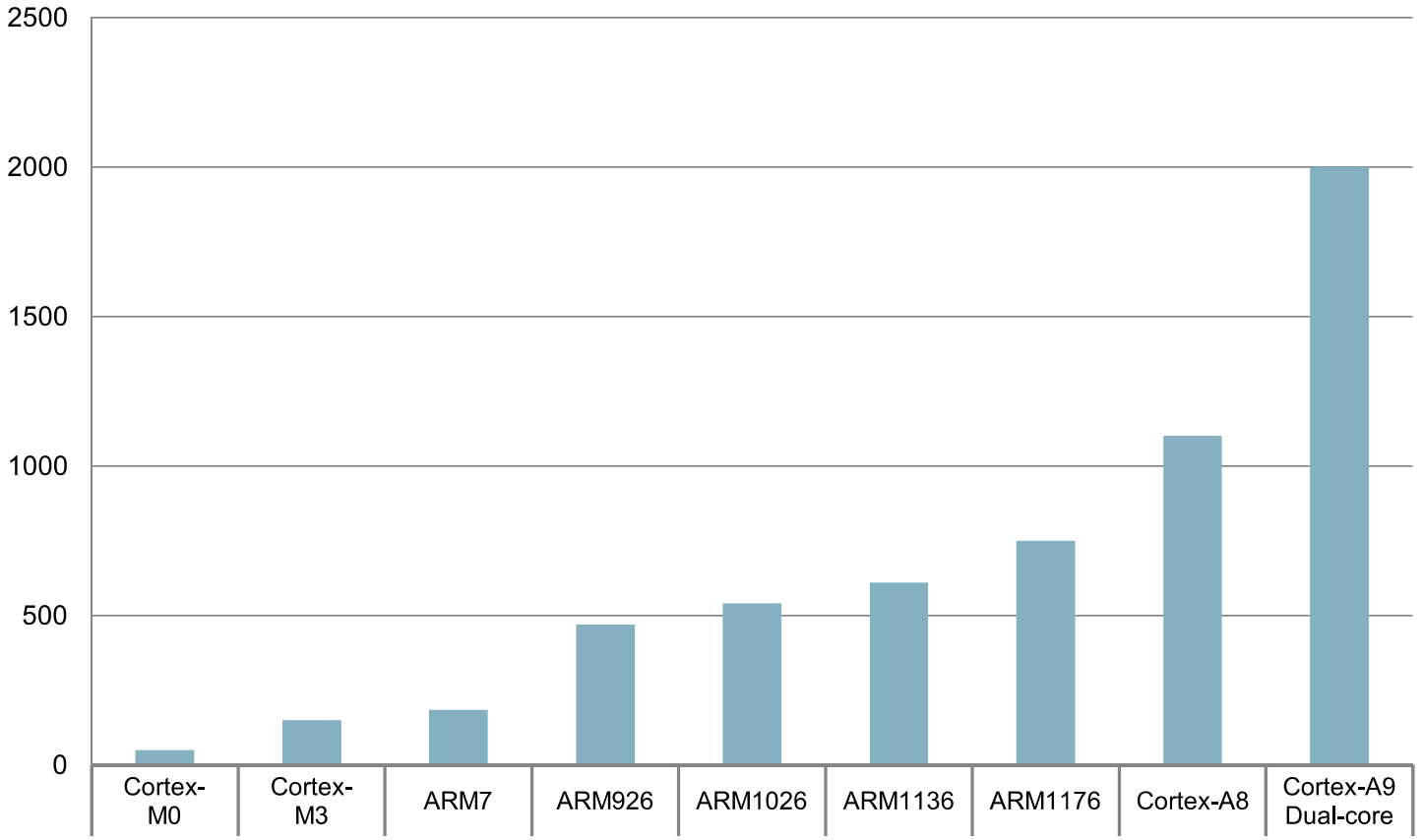




| Cortex M3 | Cortex R4 | Cortex A8 |
|---|---|---|
| Architektur v7M | Architektur v7R | Architektur v7A |
| MPU (optional) | MPU (optional) | MMU |
| AHB Lite & APB | AXI | AXI |
| | Dual Issue | VFP & NEON |

http://www.arm.com/products/processors/index.php

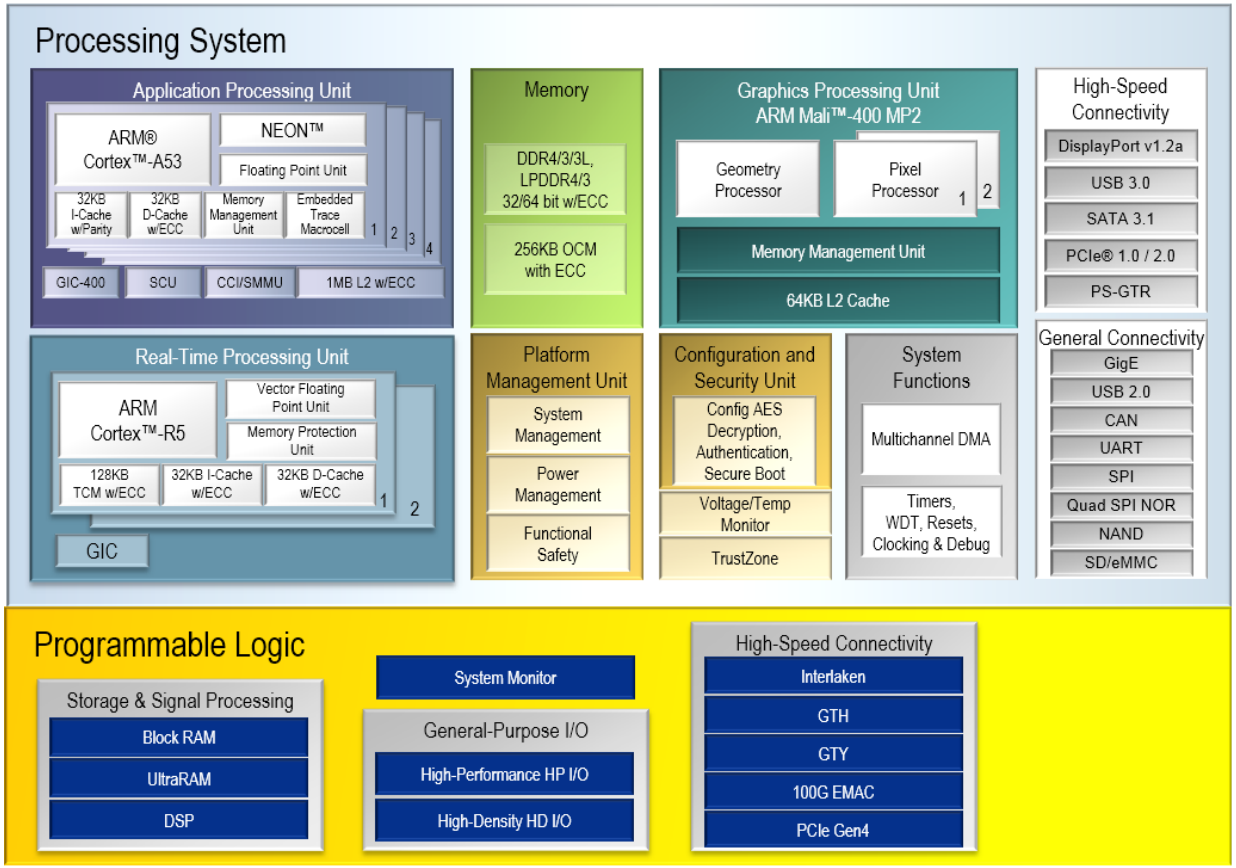
# Beispiel RISC: ARM Prozessoren (III)

- Relative Performanz*



# Beispiel SoC: Xilinx Zynq Ultrascale+ EG

- 16nm Finfet+ Low Power programmierbare Logik
  - 103k bis 1143k Logik Zellen
  - 5,3 MB bis 34,6 MB Block RAM
  - 240 bis 1968 DSP Slices
- Processing System
  - Quad Core ARM Cortex A53
    - Bis zu 1500 MHz
    - 32kB I-&D-Cache
    - 1 MB L2 Cache
    - 256kB on-chip Speicher
  - Dual Core ARM Cortex R5
    - 32 kB I-&D-Cache
    - 128 kB TCM
  - Mali-400 MP2 GPU
- High Speed Transceivers
  - 16 bis 44 16.3 Gbps
  - 0 bis 28 32,75 Gbps

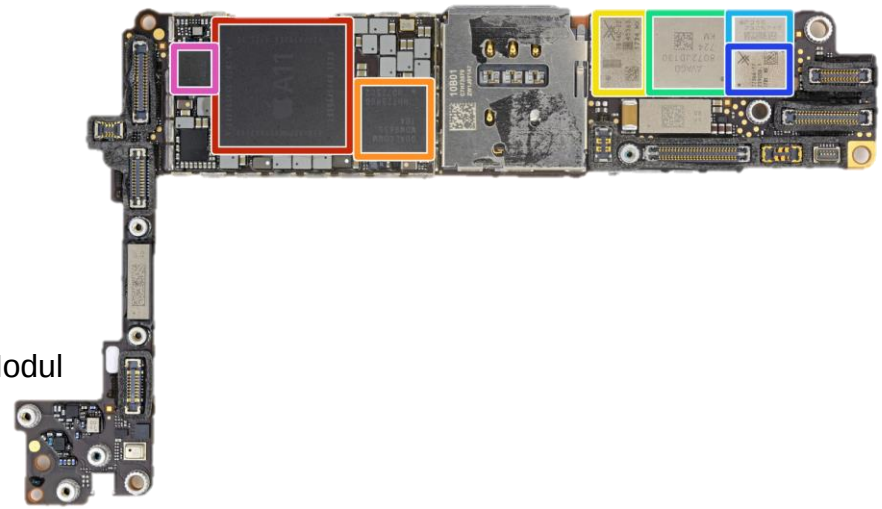



http://www.xilinx.com/products/silicon-devices/soc/index.htm

# Product SoC: Apple iPhone 8

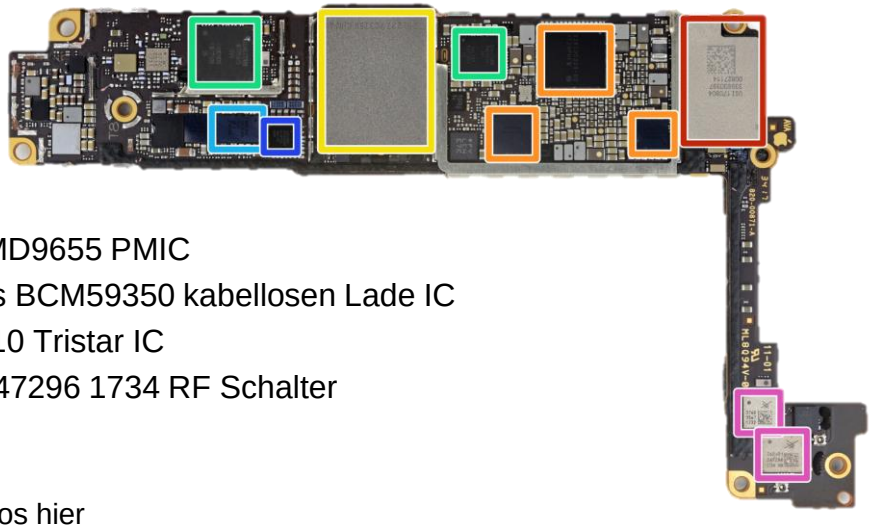
Oben:

- Apple 339S00434 A11 Bionic SoC
  über einem SK Hynix 2 GB LPDDR4x RAM
- Qualcomm MDM9655 Snapdragon X16 LTE modem
- Skyworks SkyOne SKY78140
- Avago 8072JD130
- P215 730N71T - wahrscheinlich ein envelope tracking IC
- Skyworks 77366-17 quad-band GSM Leistungsverstärker-Modul
- NXP 80V18 sicheres NFC-Modul



Unten

- Apple/USI 170804 339S00397 Wlan-/ Bluetooth-Modul
- Apple 338S00248, 338S00309 PMIC, und S3830028
- Toshiba TSBL227VC3759 64 GB NAND Flash-Speicher
- Qualcomm WTR5975 Gigabit LTE RF Transceiver und PMD9655 PMIC
- Broadcom 59355 – Wahrscheinlich eine Wiederholung des BCM59350 kabellosen Lade IC
- NXP 1612A1—Wahrscheinlich eine Wiederholung des 1610 Tristar IC
- Skyworks 3760 3576 1732 RF Schalter und SKY762-21 247296 1734 RF Schalter

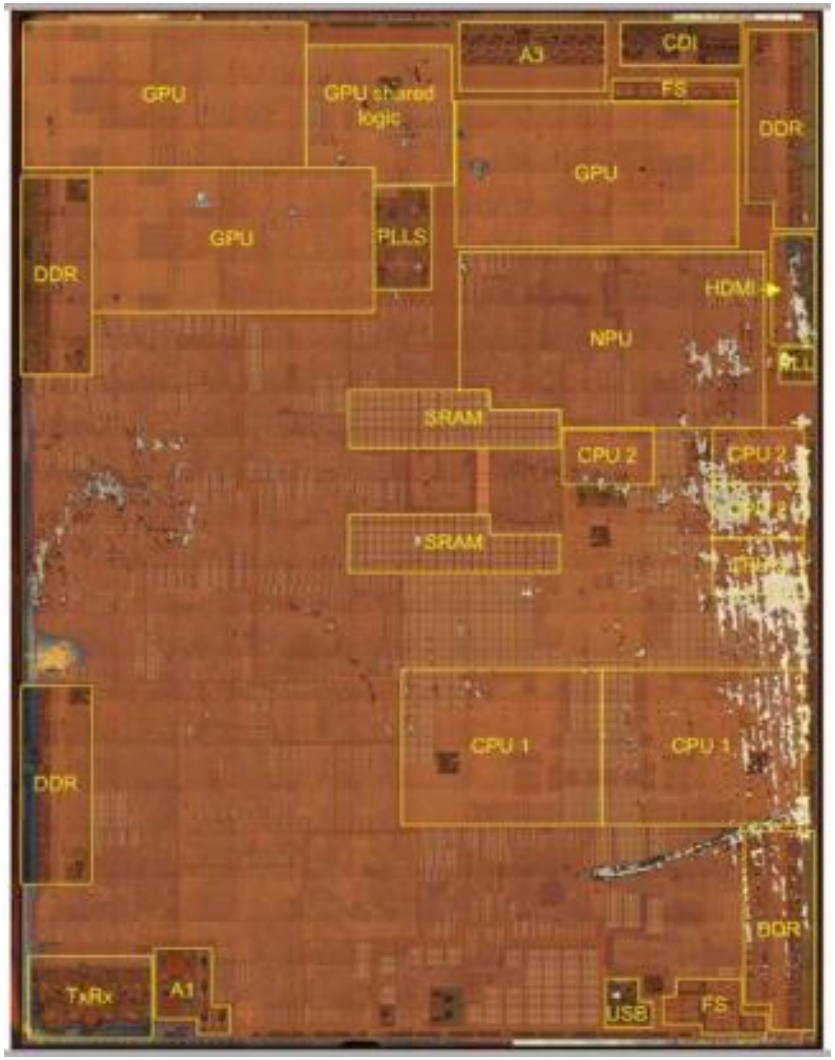


Quelle: de.ifixit.com
Notiz: Wer am Innenleben des XS oder XS Max interessiert ist findet infos hier
https://de.ifixit.com/Teardown/iPhone+XS+und+XS+Max+Teardown/113021

# Product SoC: Apple iPhone 8 – A11 SoC

- 64-Bit ARMv8-A CPU mit sechs Kernen
  - 2 Hochleistungskerne
  - 4 energieeffiziente Kerne
- Von Apple entwickelte GPU
- Bewegungs Coprozessor
- Neural Engine (maschinelles Lernen)

- Leistung laut Geekbench Benchmark auf Niveau mit Intel Core i7 MacBook Pro
- 10 nm TSMC
- 87,7 $mm^2$ , 41% kleiner als A10



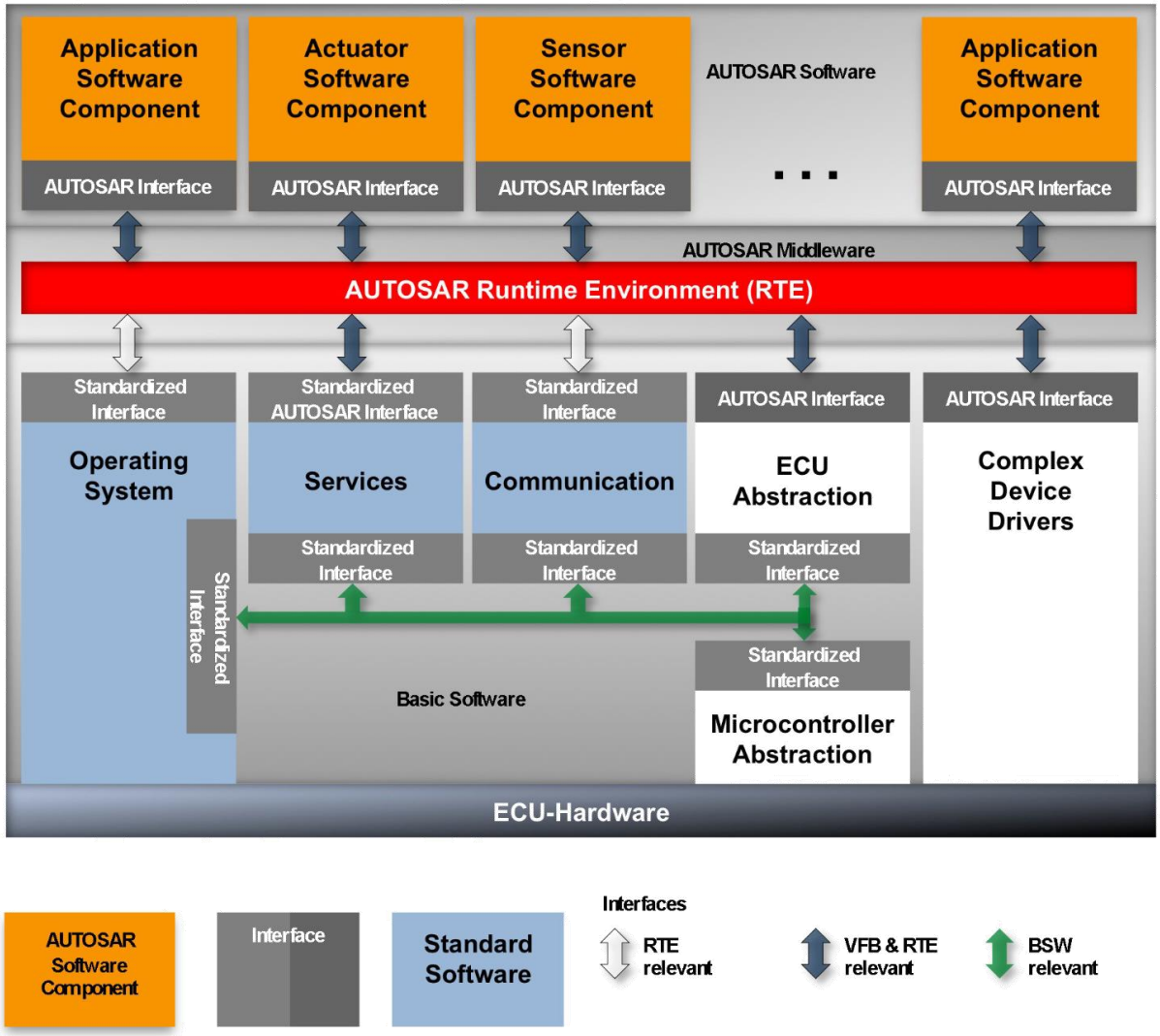



Oliver Sander
Hardware/Software Co-Design

# Android SW Stack

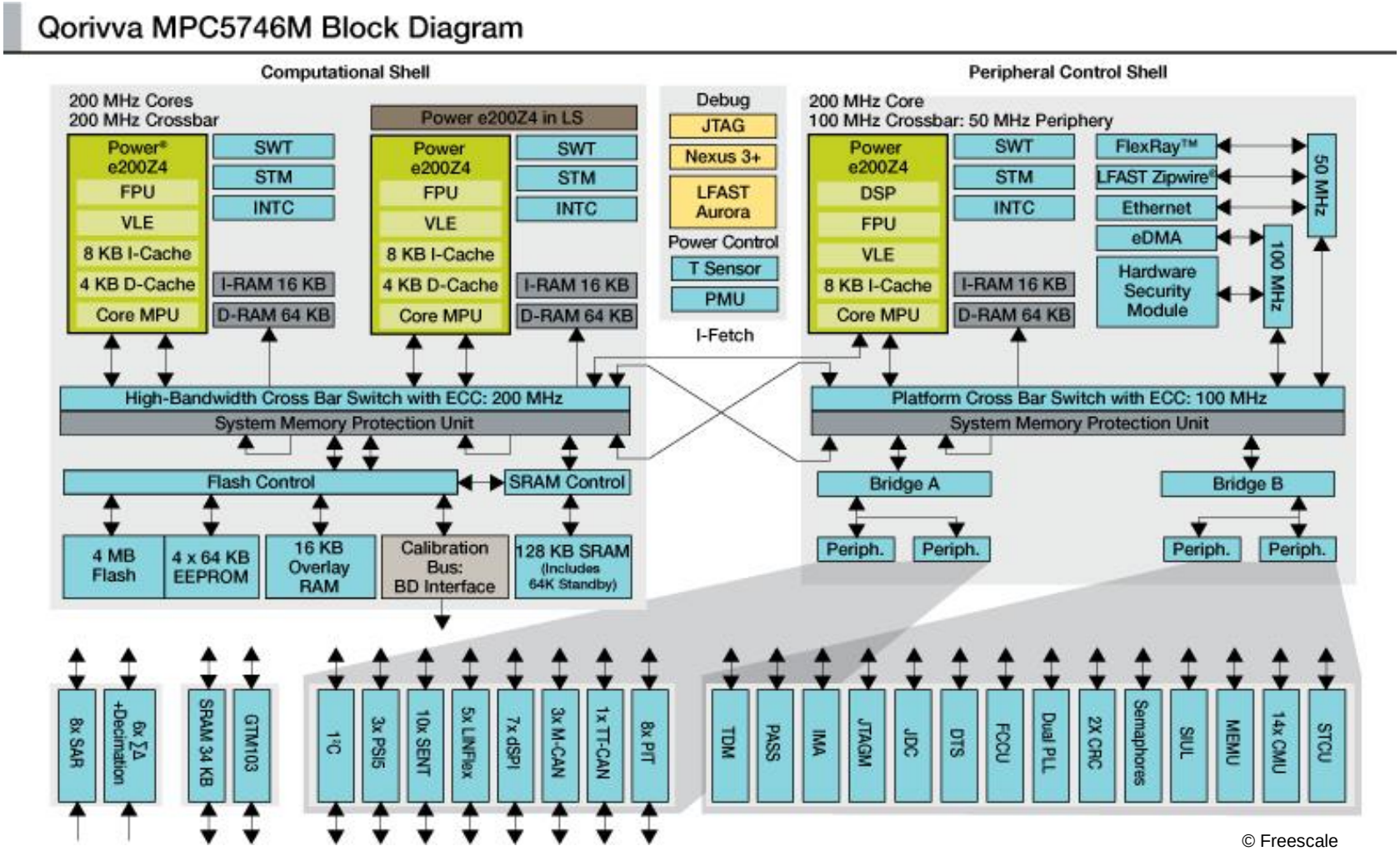

http://nikolaisammut.blogspot.de/2012/06/mobile-technologies.html

# AUTOSAR SW Stack (Automotive)




www.autosar.org

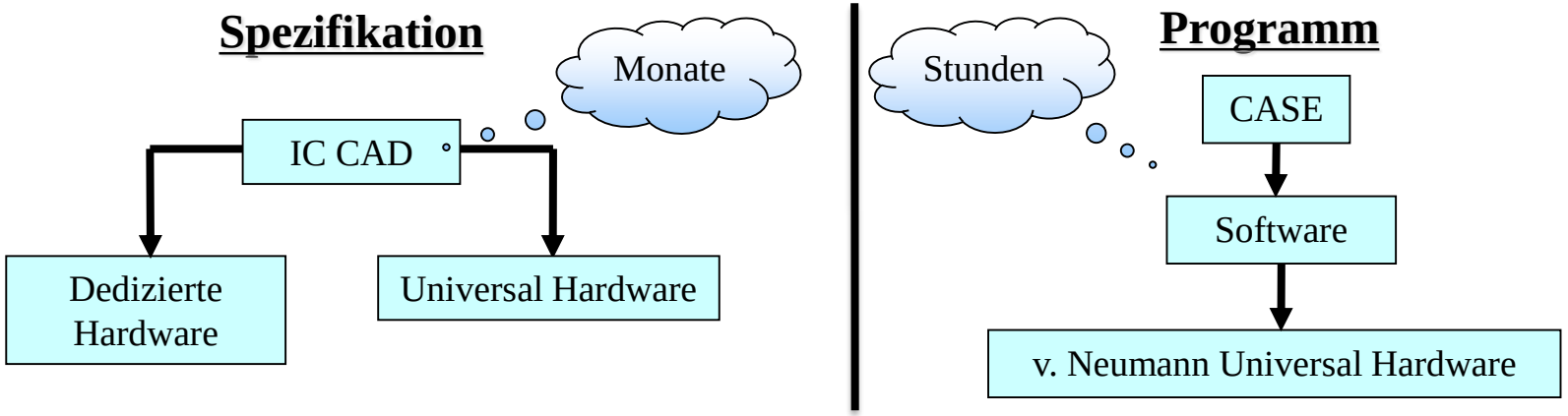
# Rechnerarchitektur (Automotive)




© Freescale

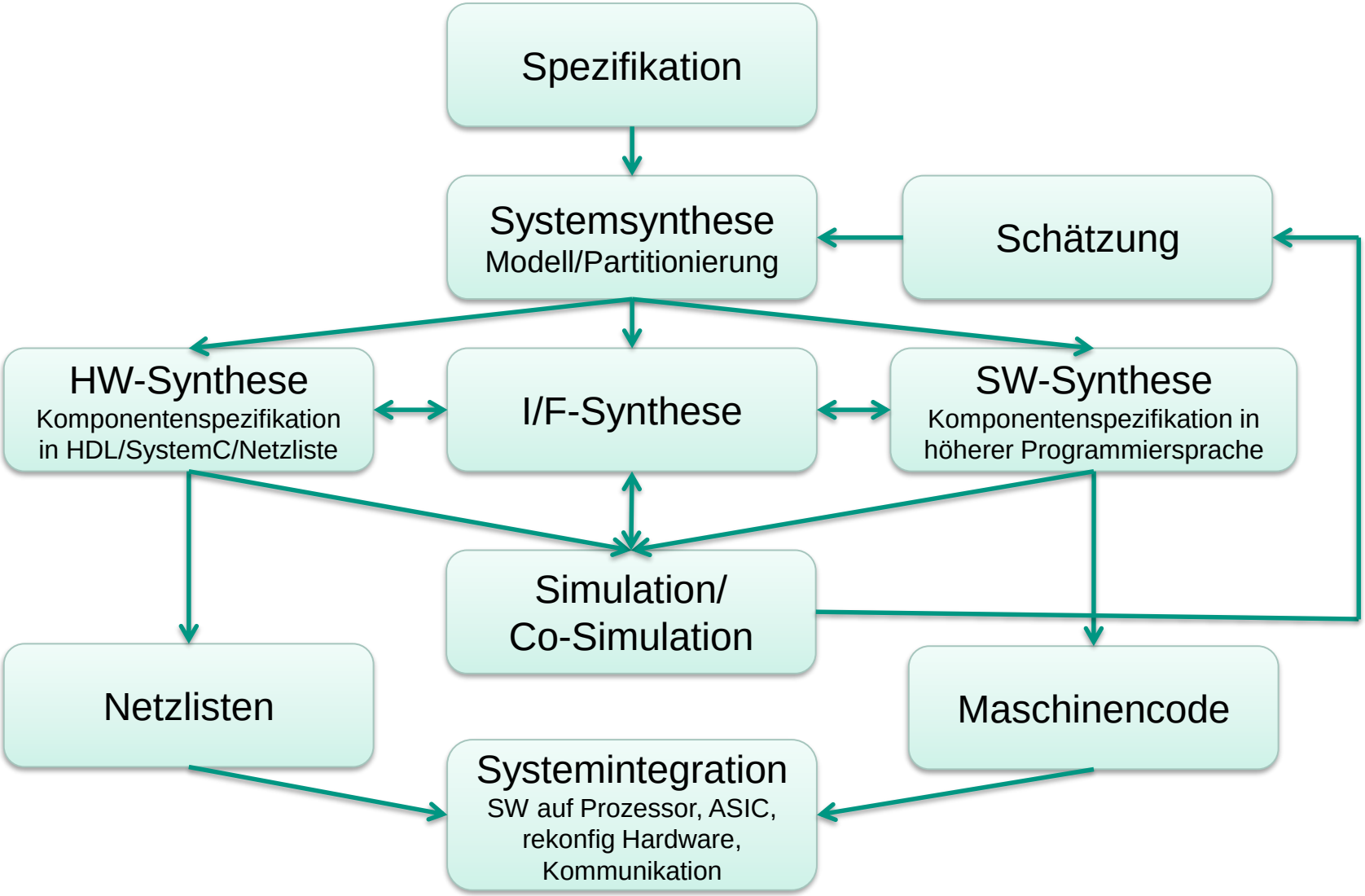
# Hardware/Software Dichotomie

| | Hardware | Software |
|---|---|---|
| Datendurchsatz | Hoch | Mittel |
| Leistungsaufnahme | Niedrig | Hoch |
| Flexibilität | Niedrig | Hoch |
| Entwicklungszeit | Hoch | Niedrig |
| Platzbedarf | Niedrig | Hoch |



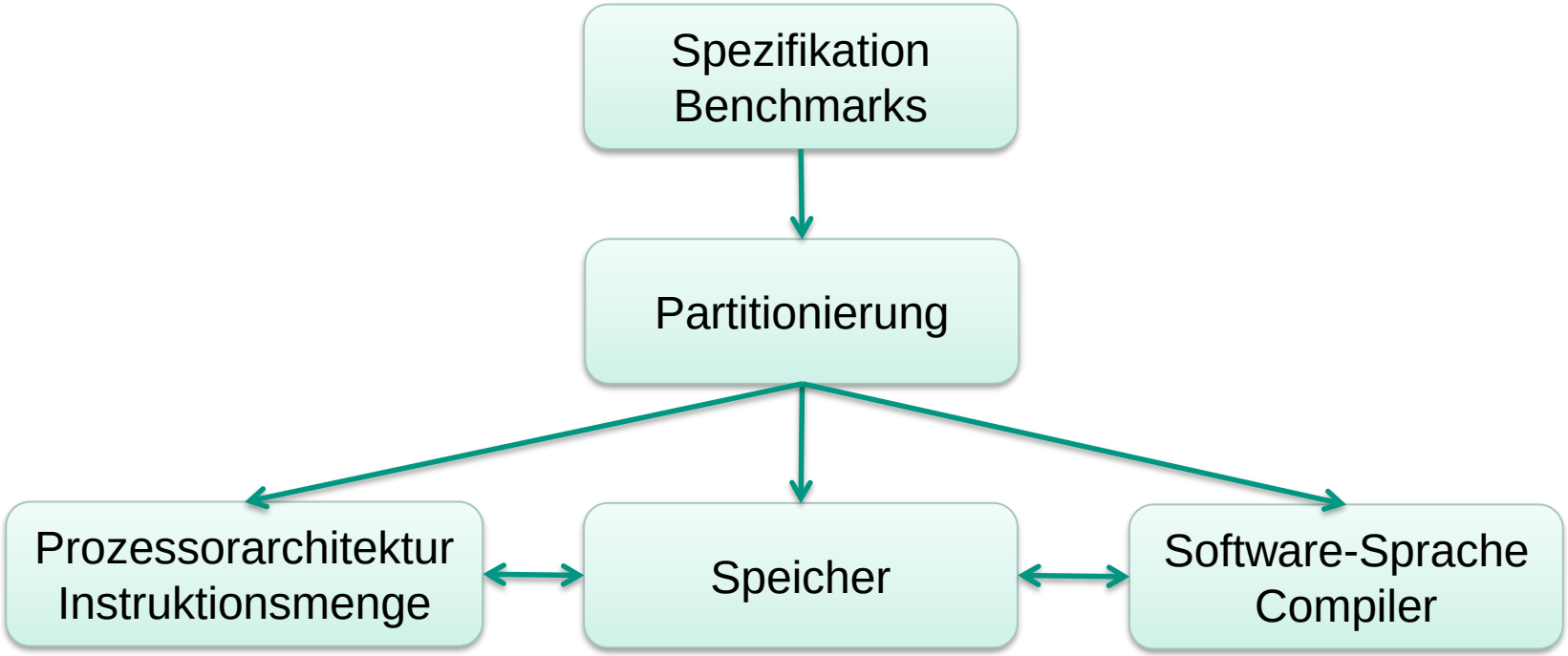

- Rekonfigurierbare Hardware zusammen mit Hardware/Software Co-Design durchbrechen diese Barriere

# Entwurf von gemischten HW/SW Systemen



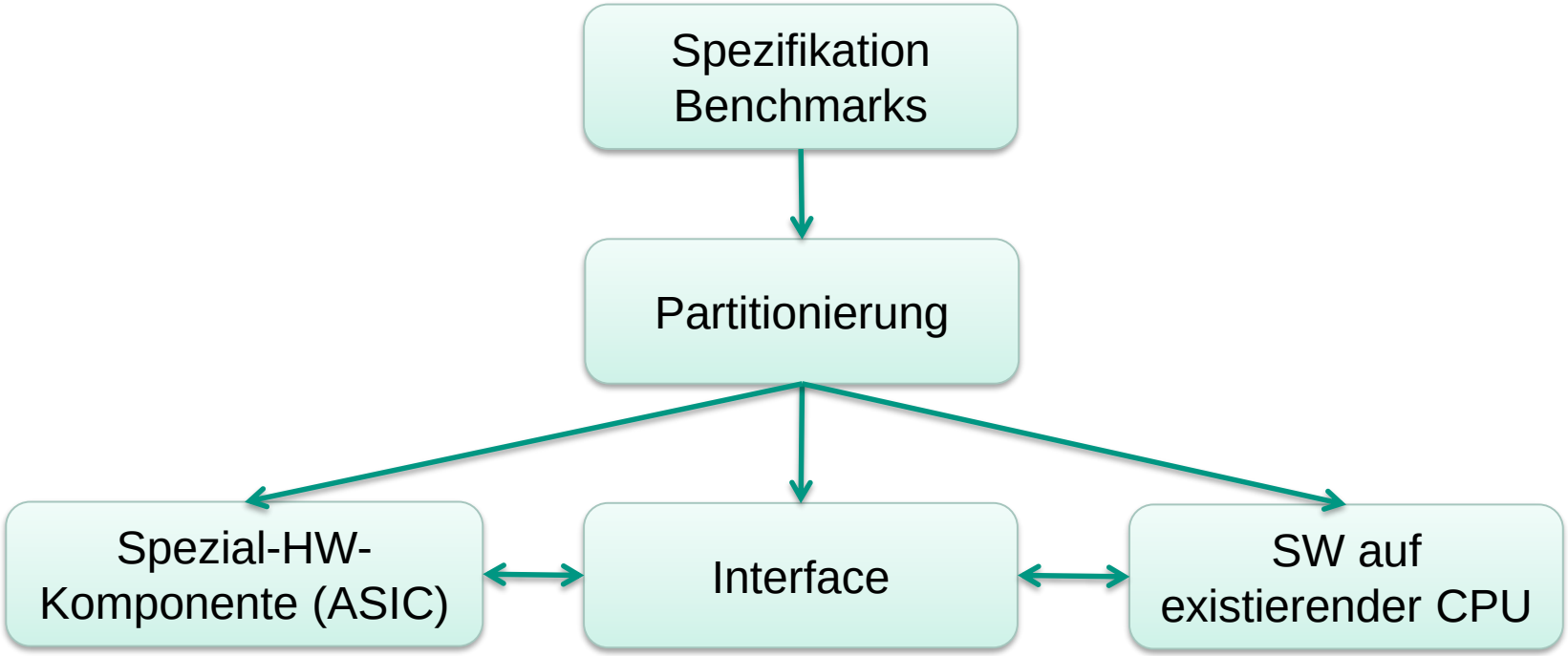
# Systemprinzip – Typ I

- logische Grenze zwischen Hardware und Software
  - Software wird auf dem gleichzeitig entwickelten Prozessor abgearbeitet
  - Optimierte Hardware-Compiler Schnittstelle (z.B. ASIP)



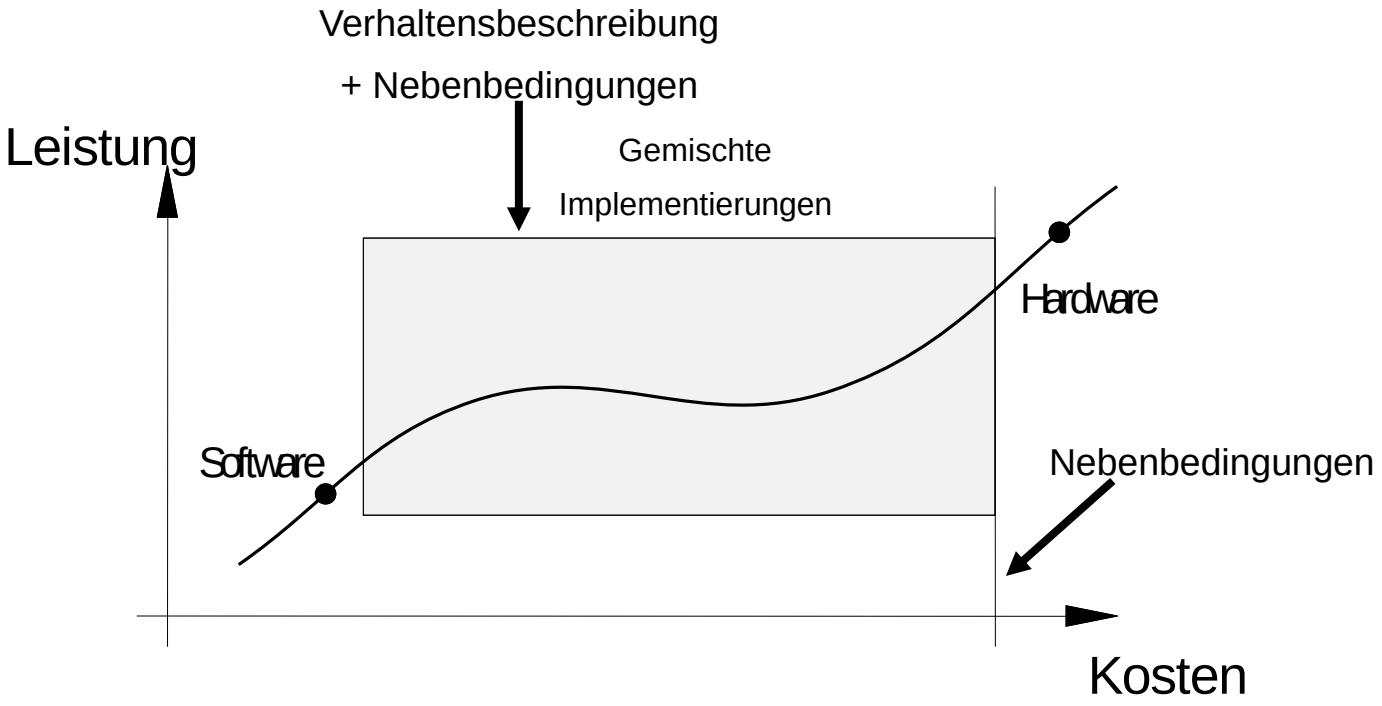
# Systemprinzip – Typ II

- physikalische Grenze
  - programmierbare Komponente und Hardwaremodule
  - effiziente Funktionalität durch Partitionierung in kooperierende HW/SW-Komponenten + Interfaces



# Entwurfsraum

- Systematische Untersuchungen des Entwurfsraumes werden bezüglich des Leistungs-/Kosten-Trade-offs durchgeführt.
  - Welche Komponente in Hardware, welche in Software?
  - Nebenbedingungen:
    - Systemkosten, Echtzeitanforderungen, Durchsatz, Fläche, Kosten



# System-Spezifikationssprachen (I)

- Frage: Wie kann man besonders effizient Hardware und Software Komponenten beschreiben?

- C/C++: universell, schwer synthetisierbar (dynamische Konstrukte)
- VHDL: gebräuchlich, Erweiterung OO–VHDL
- Verilog: gebräuchlich, Erweiterung SystemVerilog
- HardwareC / SystemC: Hardware-orientierte C/C++ – Erweiterungen

- Problem: Es gibt bisher keine Spezifikationsform, die sowohl gut zur Beschreibung von Software als auch gut für Hardware und deren Übersetzung & Synthese geeignet ist

# System-Spezifikationssprachen (II)

- Ansatz: Gemischte Hardware/Software Architektur-beschreibungssprachen (ADL)
  - z.B. Language for Instruction Set Architectures (LISA)

- Vorteile:
  - Standard-Vorgehen für Hardware- und Software-Zweig
  - Wiederverwendung von Komponenten
  - Leistungsfähige Werkzeuge zur Übersetzung & Synthese vorhanden

- Nachteile:
  - Übersetzung in andere Sprachen schwierig, z.B. Verschiebung von Komponenten von Hardware nach Software und umgekehrt
  - Hardware/Software Co-Design Experten werden benötigt

- Kennenlernen von Aufgabenstellungen, Ansätzen und Methoden im modernen Systementwurf
- Alternativen für Hardware/Software Implementierungen
- Vielseitig anwendbare Optimierungs-verfahren
- Einblicke in aktuelle Forschungsgebiete und deren Anwendungen bei
  - HiWi-Jobs
  - Abschlussarbeiten