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

Institut für Technik der Informationsverarbeitung (ITIV)

Hardware/Software Co-Design

Informationstechnische Systeme

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







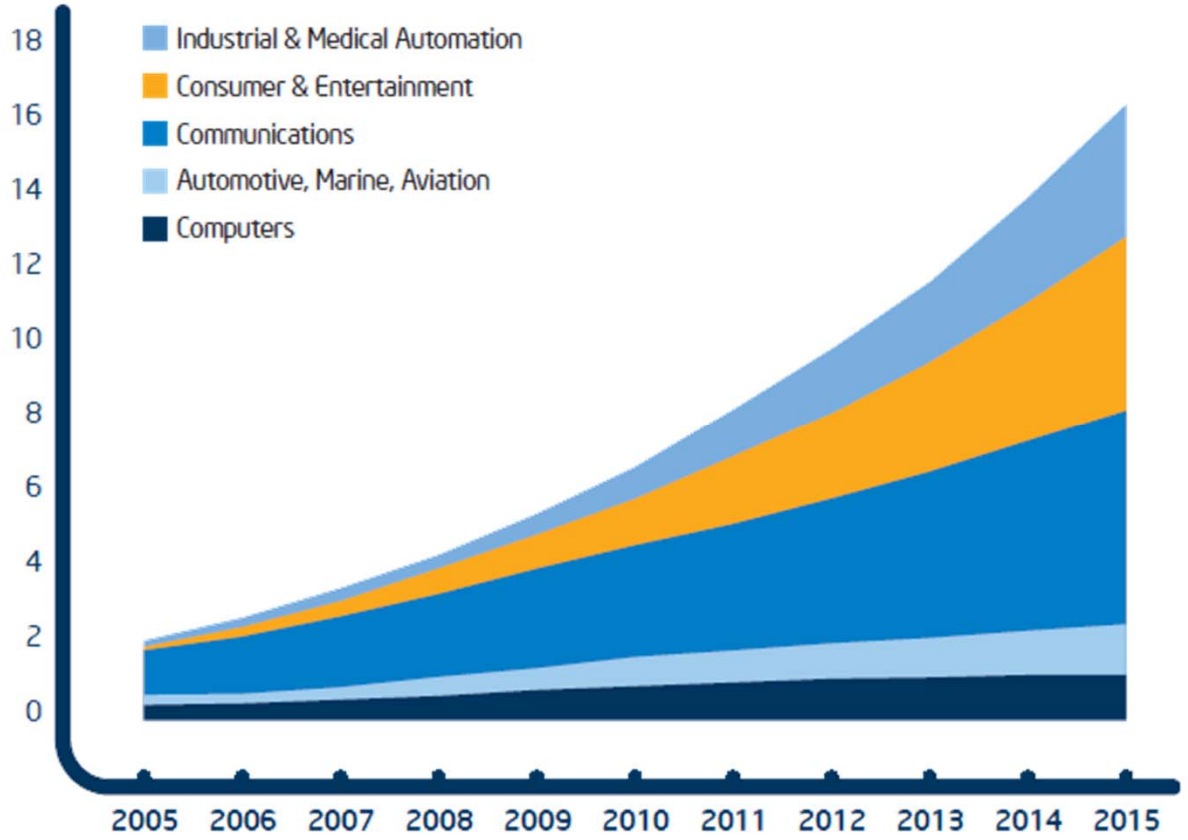







Oliver Sander
Hardware/Software Co-Design

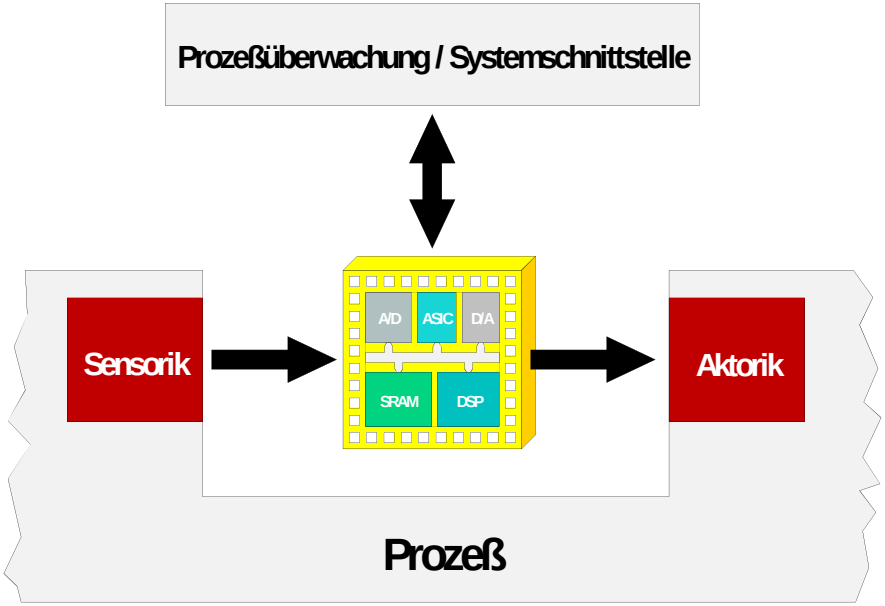
Prozessorstückzahlen



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

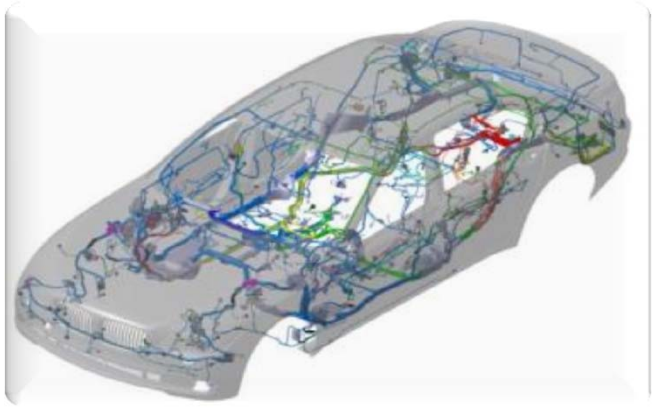
Eingebettetes System (Embedded System)

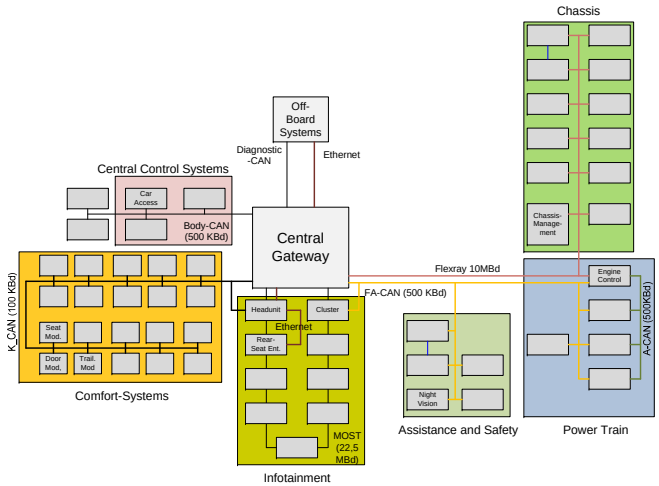
- Der Ausdruck eingebettetes System bezeichnet einen elektronischen Rechner oder auch Computer, der in einen technischen Kontext eingebunden (eingebettet) ist. Dabei übernimmt der Rechner entweder Überwachungs-, Steuerungs- oder Regelfunktionen oder ist für eine Form der Daten- bzw. Signalverarbeitung 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

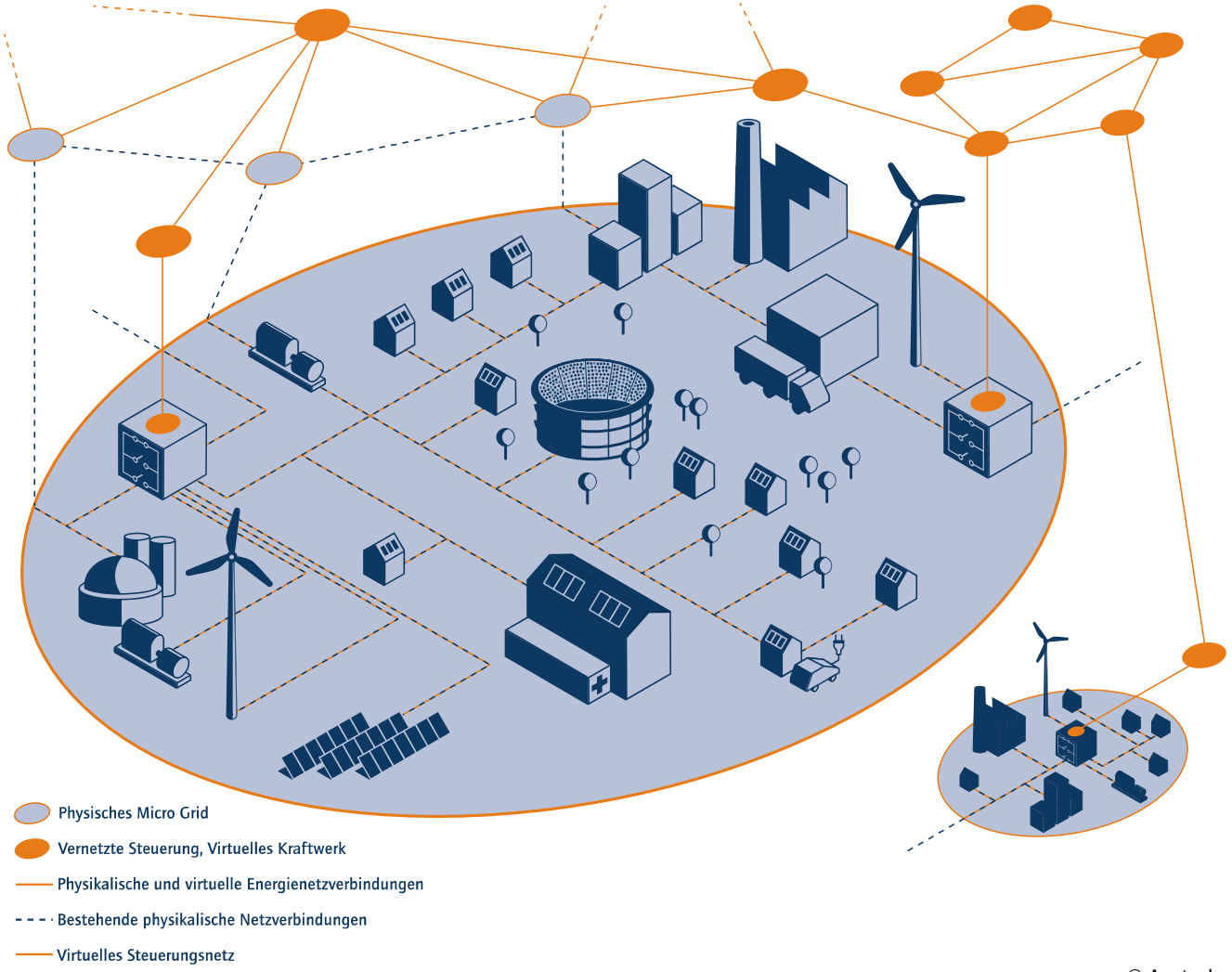
Other

Software	3-5 GB Application 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)

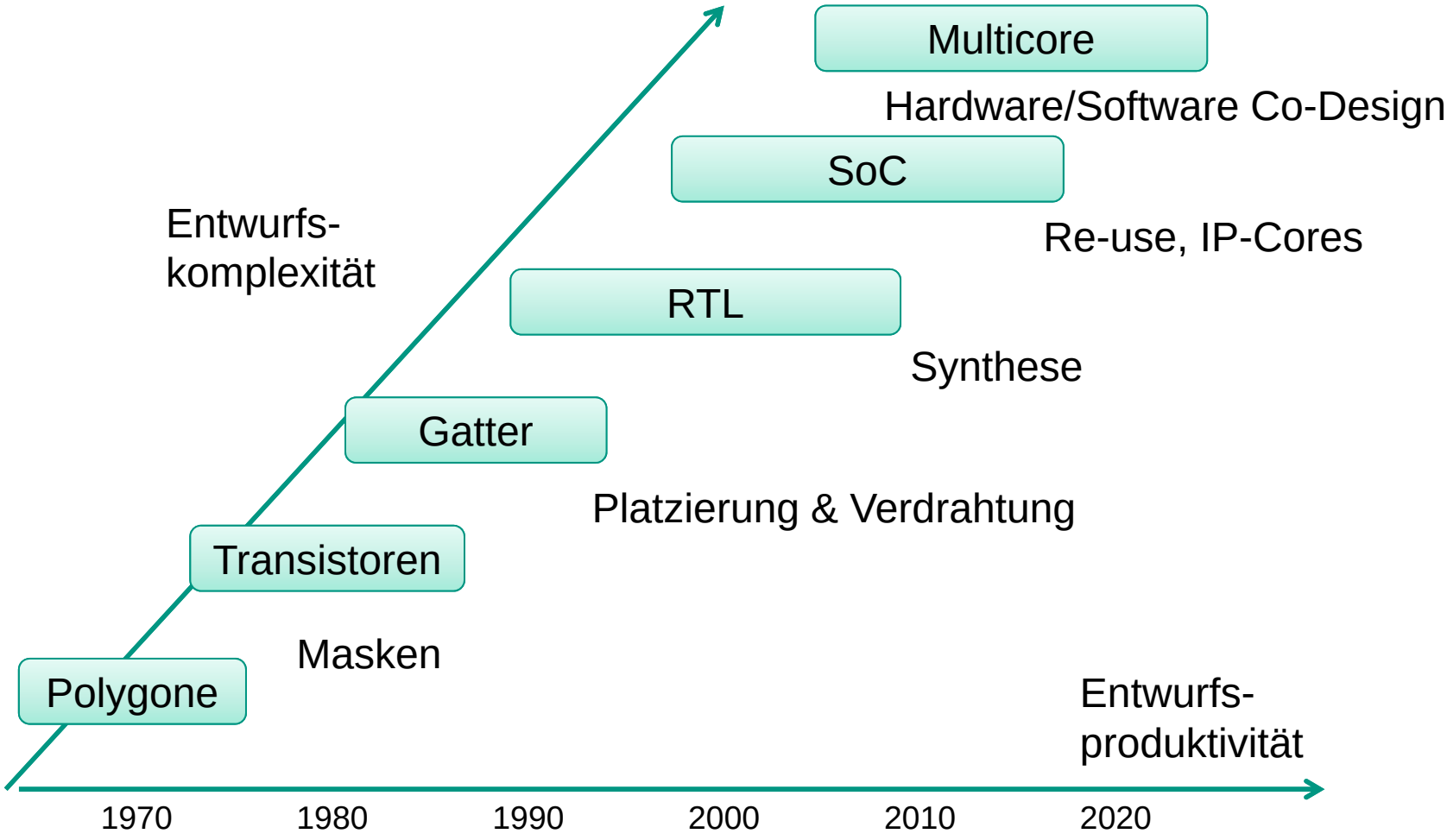
- Erfahrener Systementwickler kann für manche Komponenten oftmals gezielt entscheiden, ob HW- oder SW-Implementierung besser ist
  - spezielles Anwendungswissen
- Interaktive Strategie notwendig und vorteilhaft
  - Erfahrungswissen
- Spezielle Partitionierungsstrategien für unterschiedliche Anwendungsbereiche 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

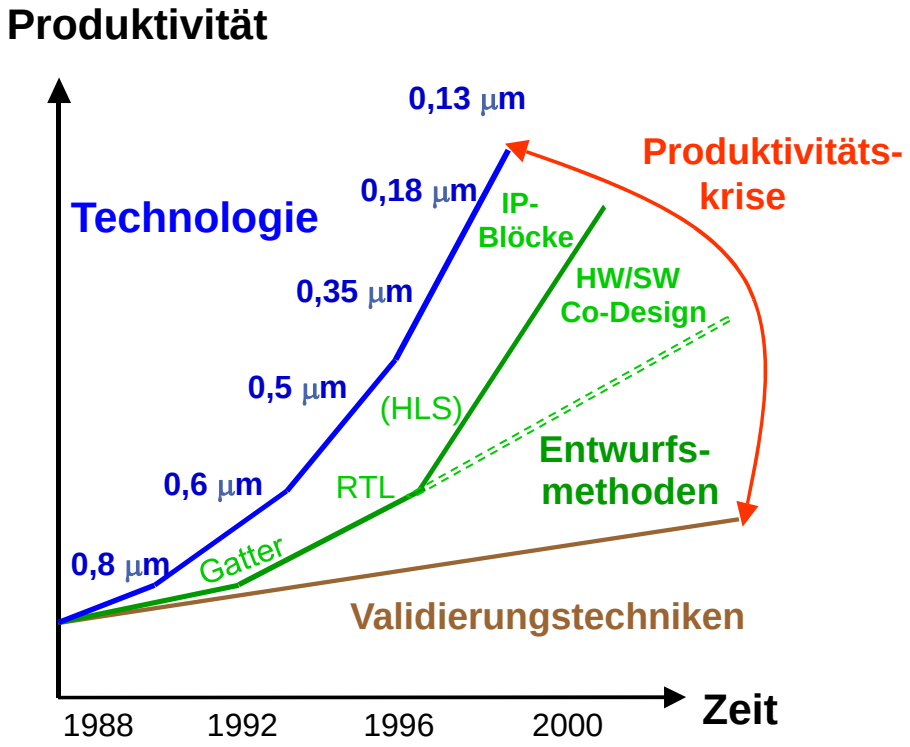


Produktivitätsgrenze (II)

- 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

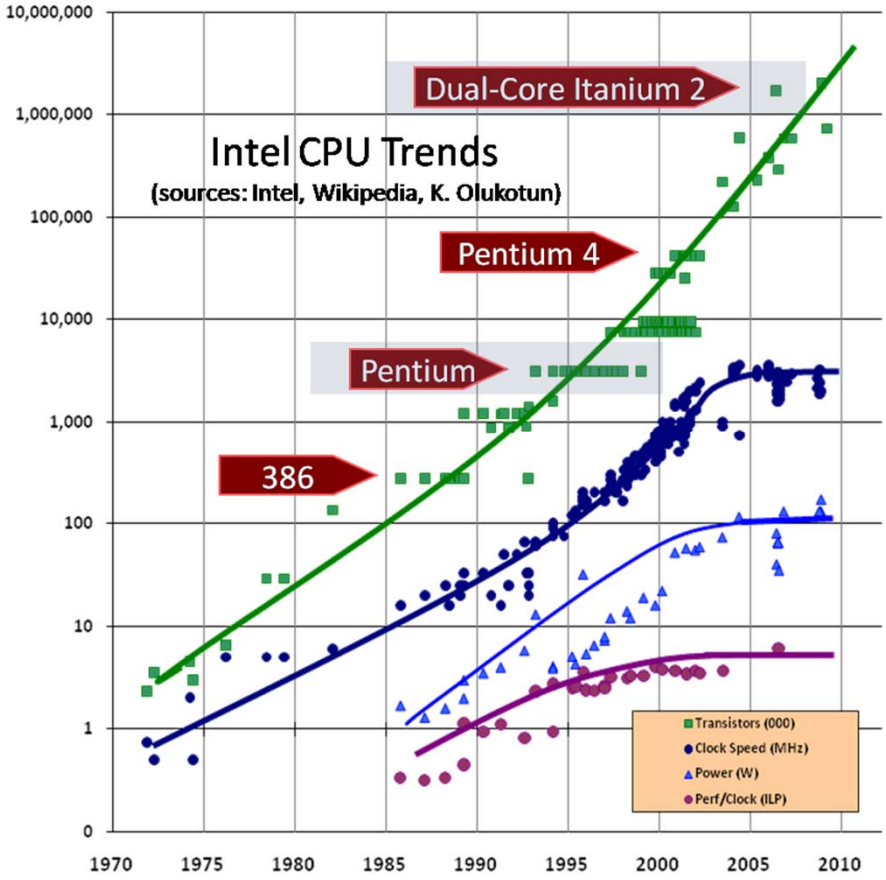


Free Lunch is over

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

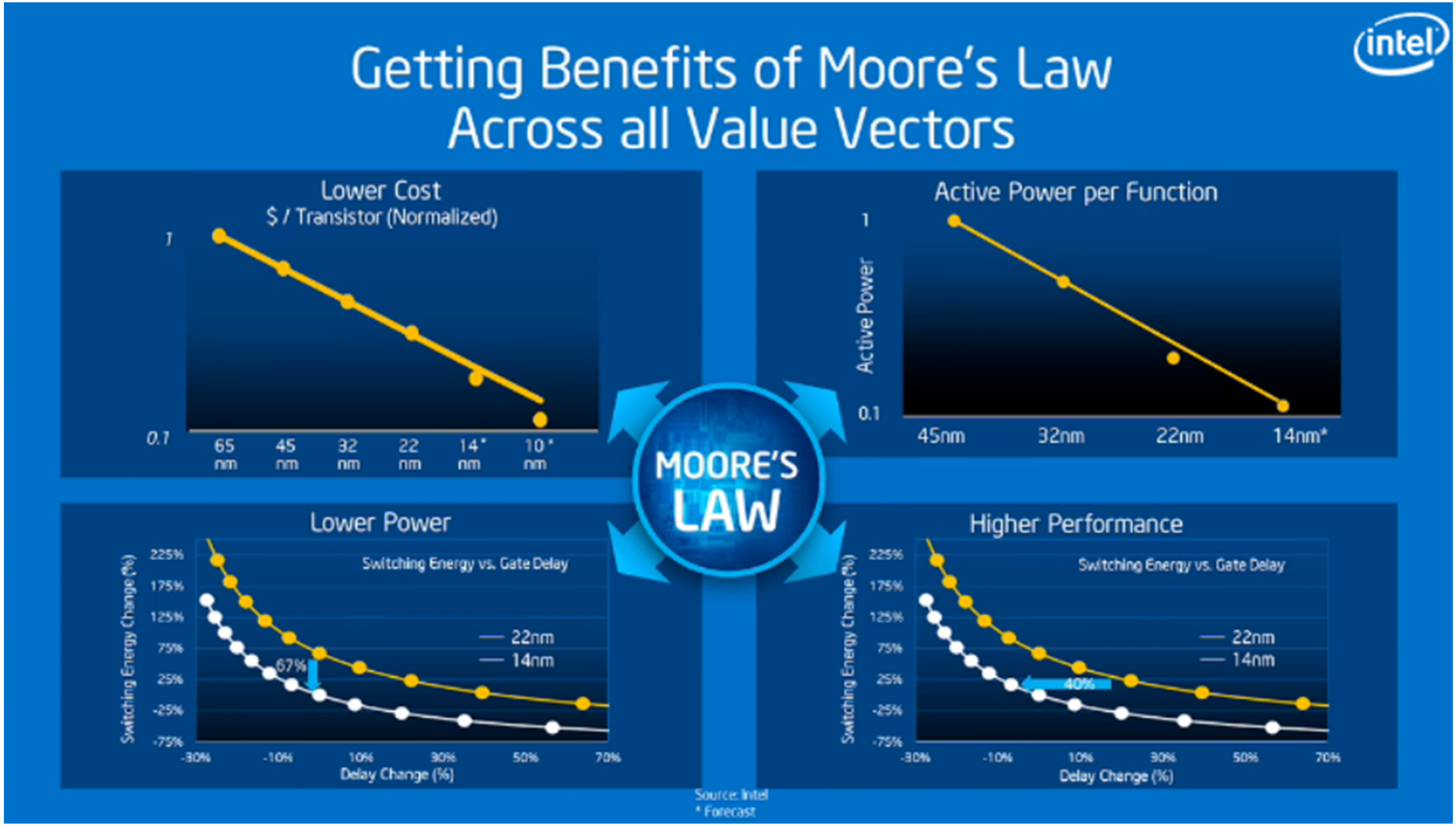
- Frequenz skaliert nicht mehr
  - Keine 10 GHz Systeme
- Fläche skaliert bald nicht mehr
  - Dark silicon
    - Deep sub-micron Effekte
- Verlustleistung
  - Statische und dynamische Verlustleistung vergleichbar

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



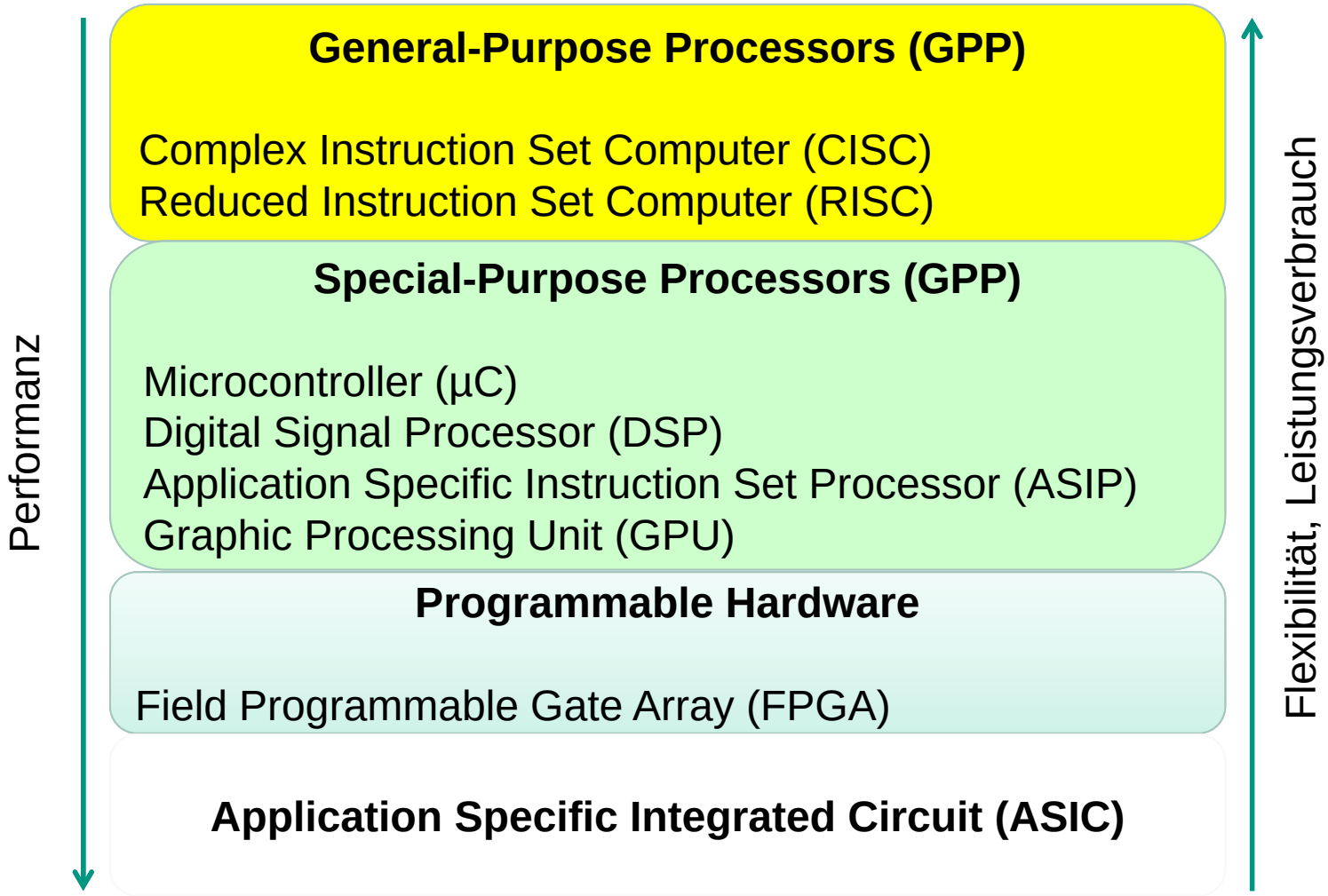
http://www.gotw.ca/publications/concurrency-ddj.htm

Moore‘s Law & aktuelle Fertigungstechnologie

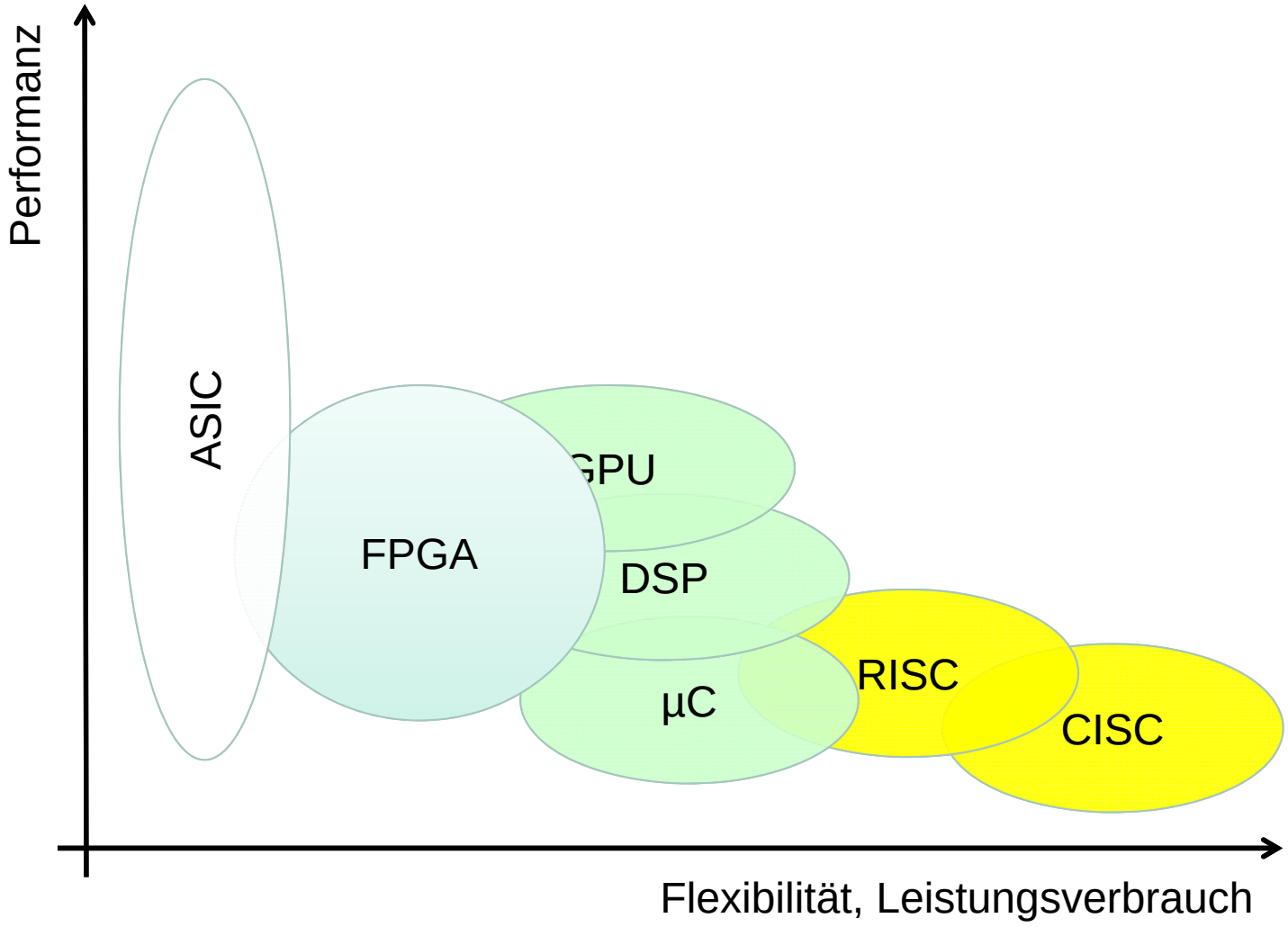


© Intel

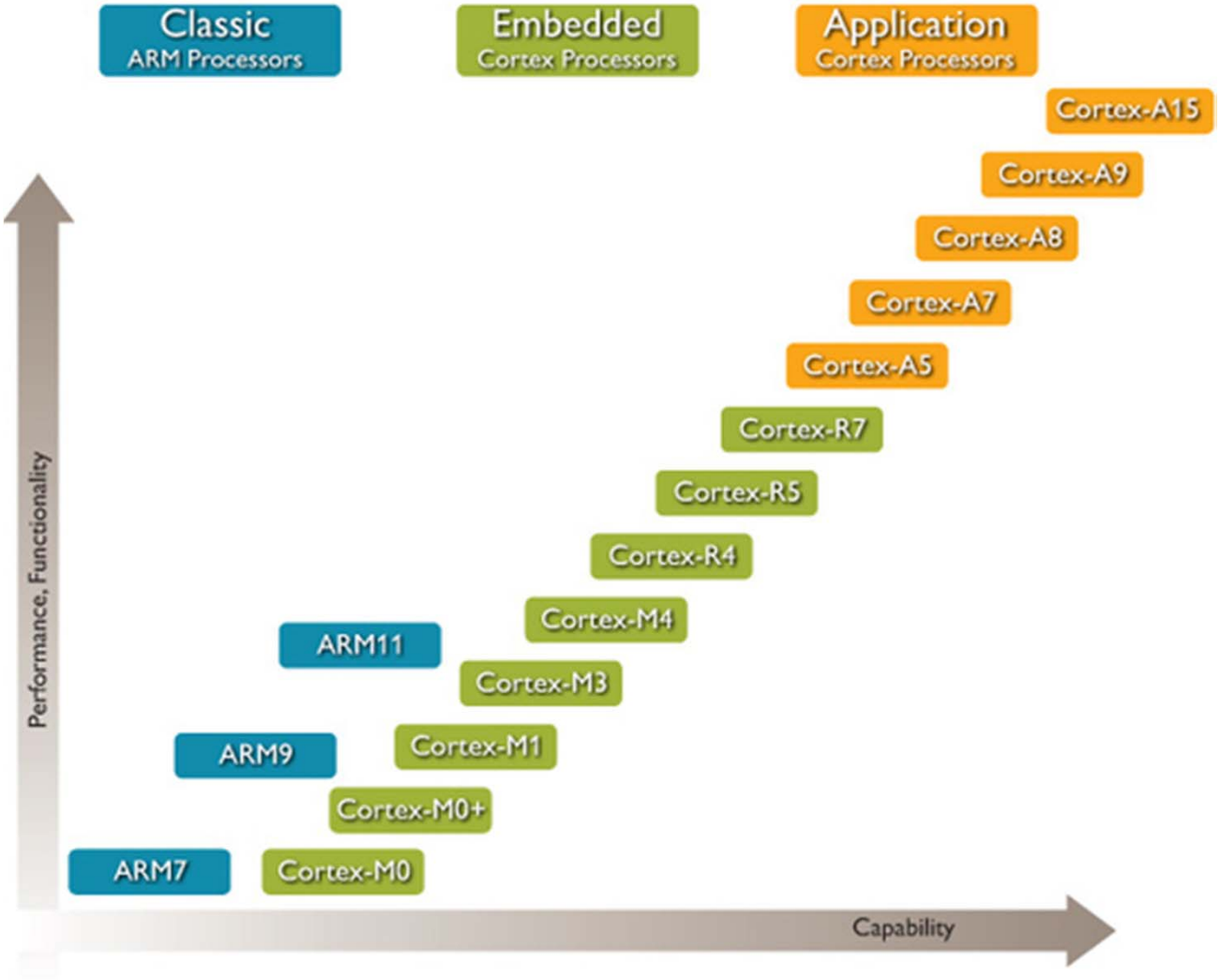
Vergleich der Zielarchitekturen



Vergleich der Zielarchitekturen



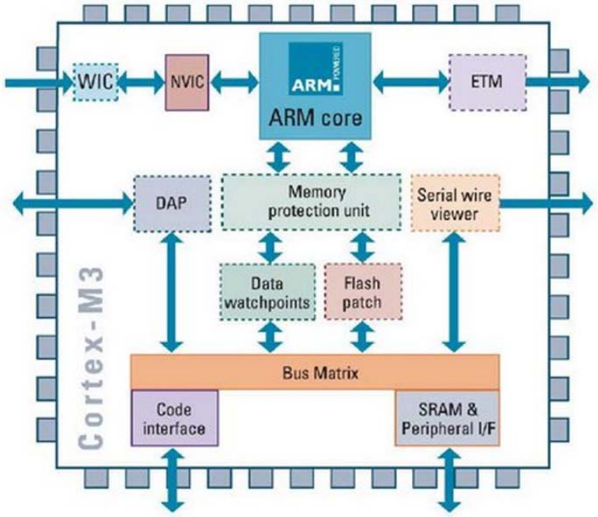
Beispiel RISC: ARM Prozessoren (I)

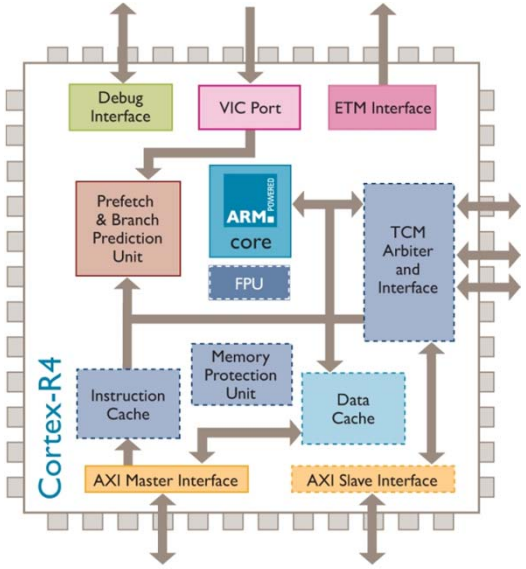


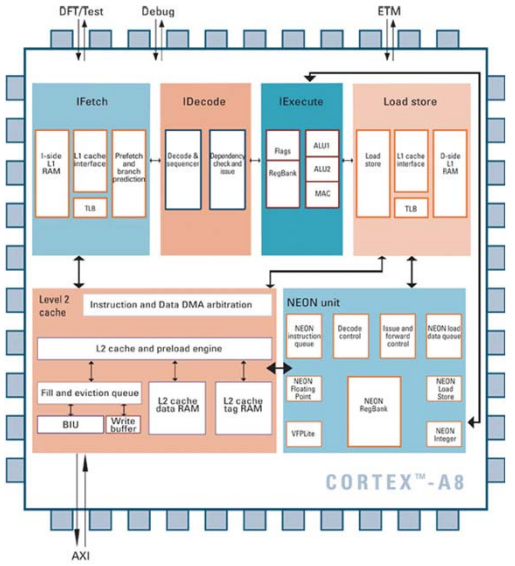
http://www.arm.com/products/processors/index.php

Beispiel RISC: ARM Prozessoren (II)

- ARM Cortex Prozessor Familie





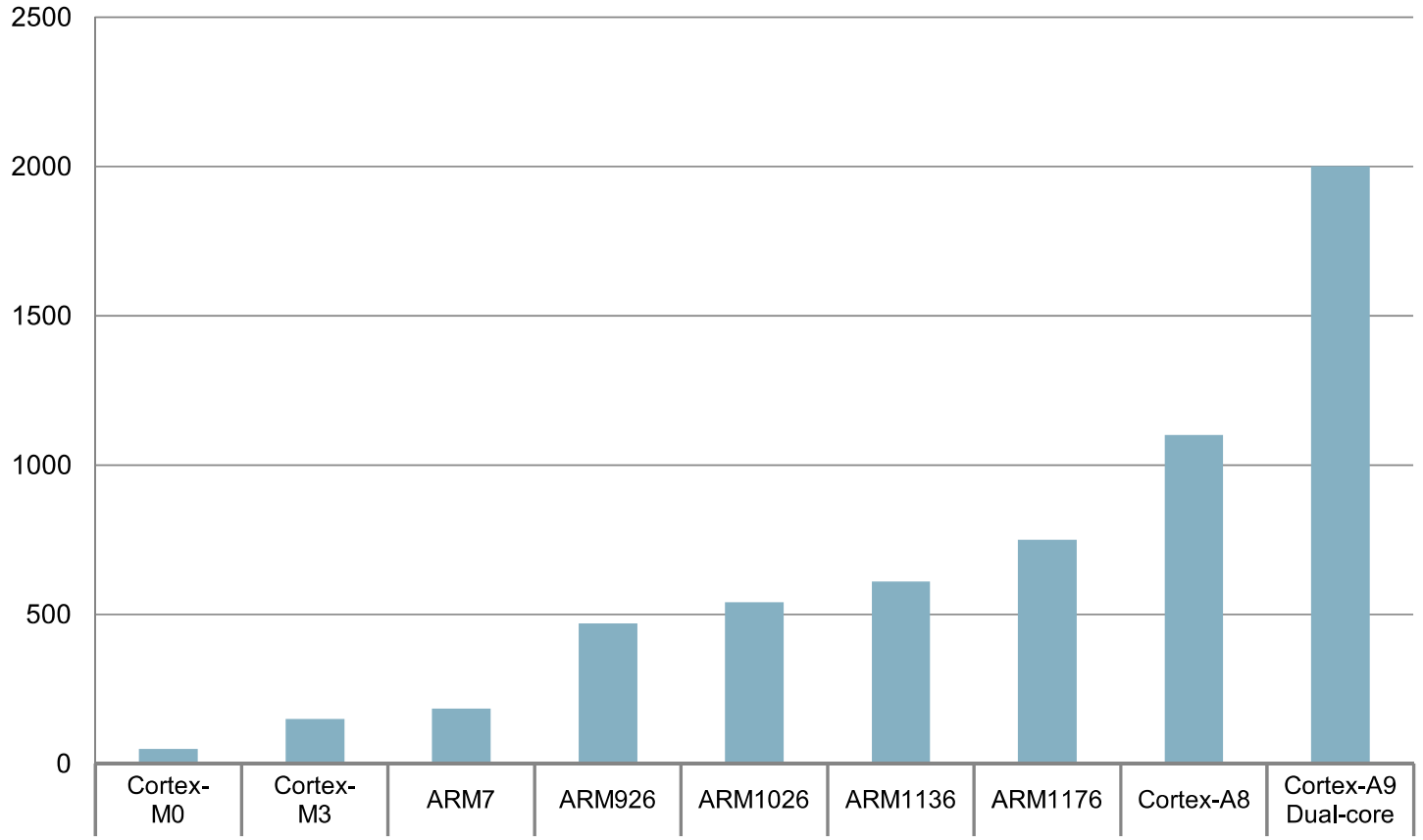


Cortex M3	Cortex R4	Cortex A8
Architektur v7M	Architektur v7R	Architektur v7A
MPU (optional)	MPU (optional)	MMU
AHB Lite & APB	AXI	AXI
	Dual Issue	VFP & NEON

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

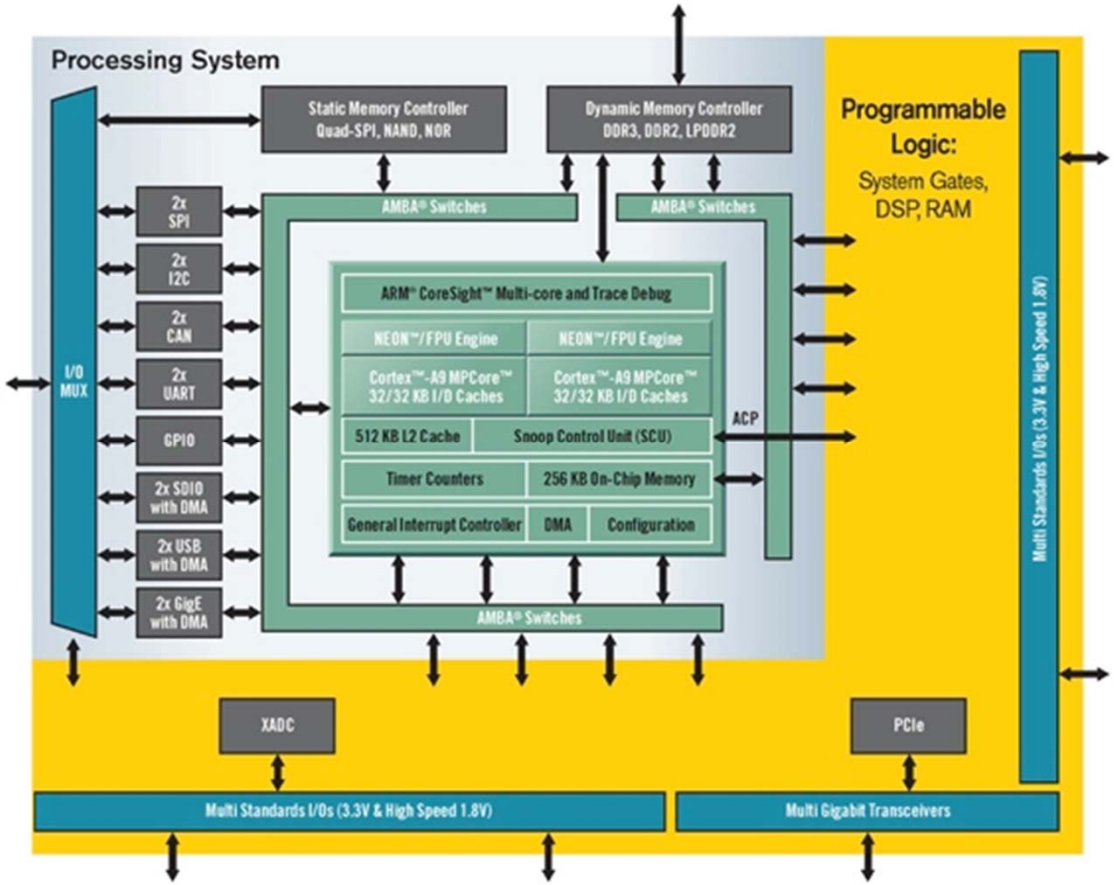
Beispiel RISC: ARM Prozessoren (III)

- Relative Performanz*



Beipiel SoC: Xilinx Zynq-7000

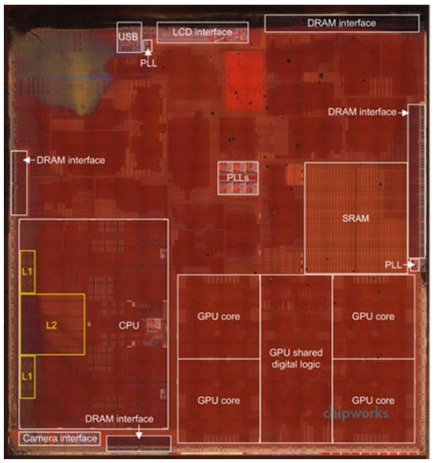
- 28nm Low Power programmierbare Logik
  - 28k bis 350k Logik Zellen
  - 240KB bis 2180KB Block RAM
  - 80 bis 900 DSP Slices
- Processing System
  - Dual Core ARM Cortex A9
    - Up to 800 MHz
    - 32kB I-&D-Cache
  - 512kB L2 Cache
  - 256kB on-chip Speicher
- Dual 12bit 1Msps ADC
- 4 bis 16 12.5Gbps Transceivers
- PCI Express, USB, CAN, I2C, UART, SPI, GPIO

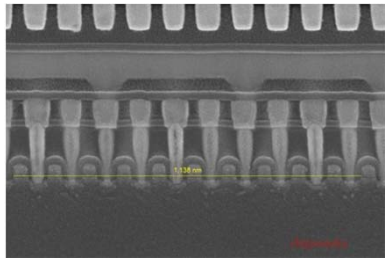


http://www.xilinx.com/products/silicon-devices/soc/index.htm

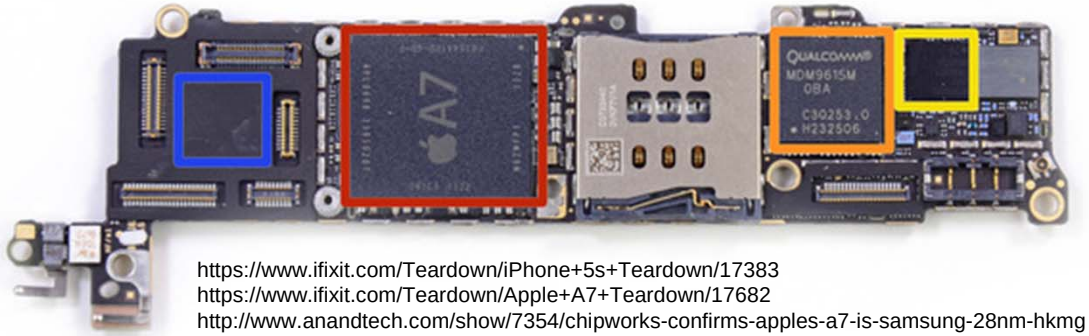
Product SoC: Apple A7 (iPhone5s)

- Erster 64-bit Prozessor im Smartphone
  - 2 x Apple Cyclone ARMv8 64-bit cores
    - ARM Architektur-Lizenz
  - IMG PowerVR G6430 GPU
  - 1GB LPDDR3
  - Samsung 28nm HK+MG Prozess



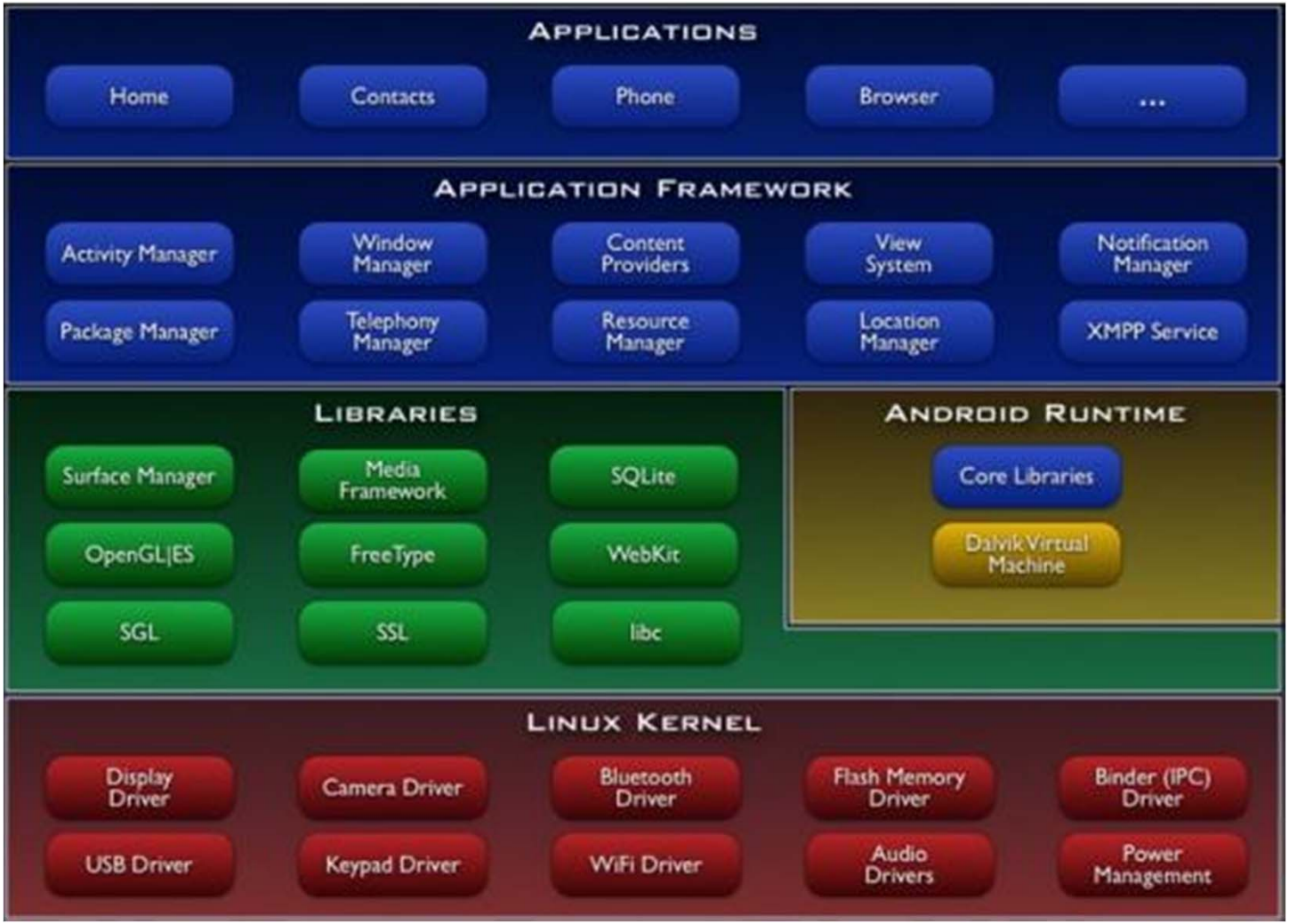


- Qualcomm WTR1605L LTE/HSPA+/CDMA2K/TDSCDMA/EDGE/GPS
- Qualcomm MDM9615M LTE Modem
- Apple A7 APL0698 SoC
- Apple M7 NXP LPC1800



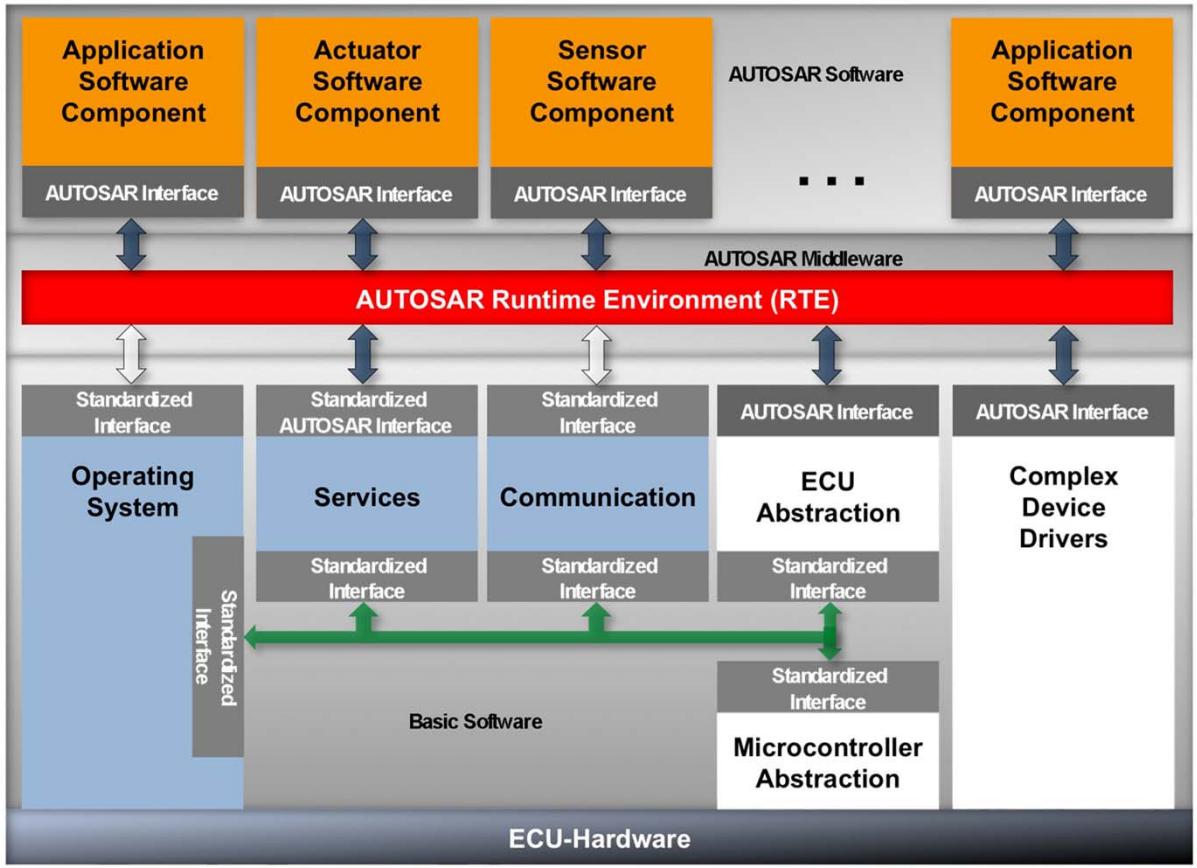
https://www.ifixit.com/Teardown/iPhone+5s+Teardown/17383
https://www.ifixit.com/Teardown/Apple+A7+Teardown/17682
http://www.anandtech.com/show/7354/chipworks-confirms-apples-a7-is-samsung-28nm-hkmg

Android SW Stack



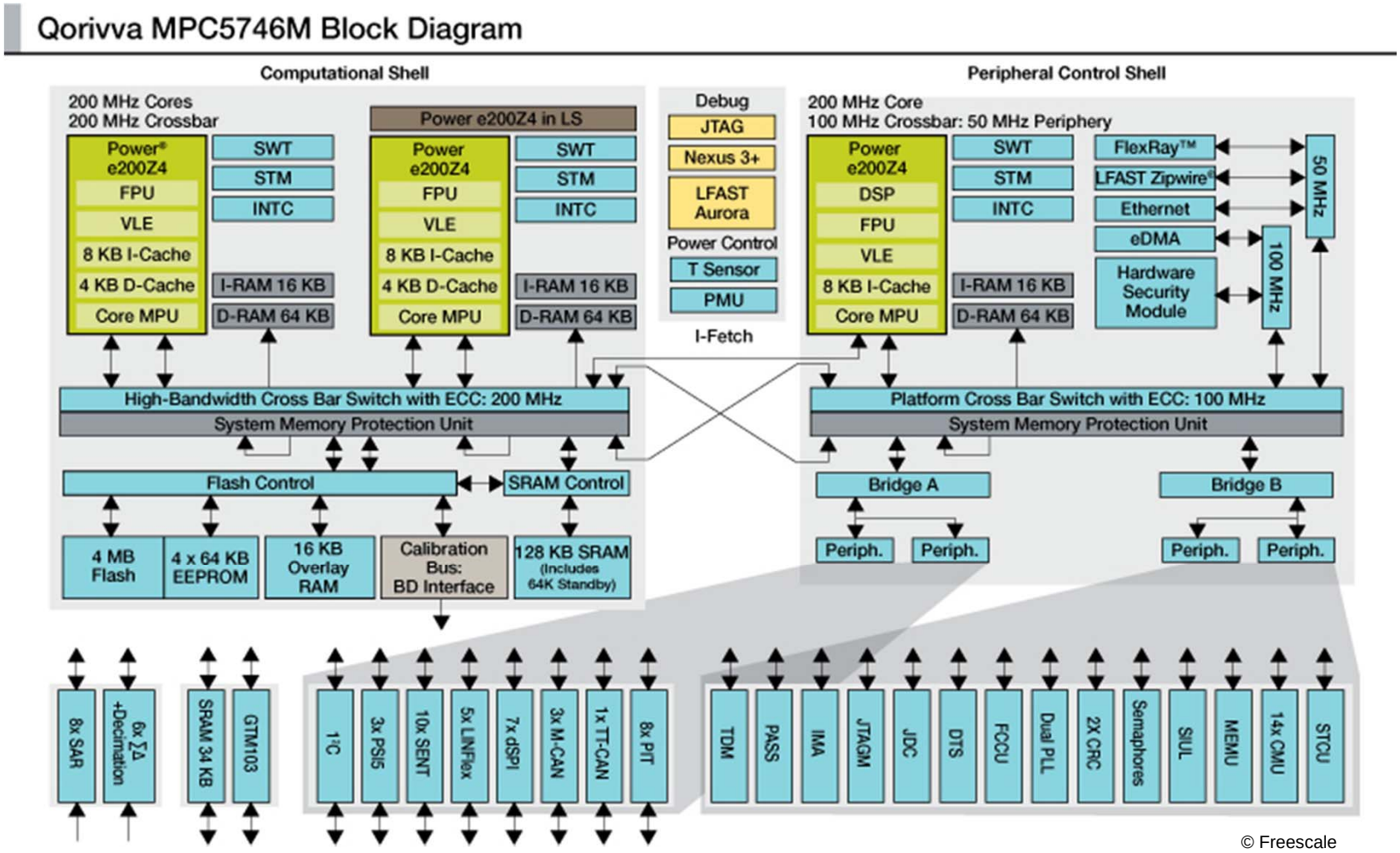
http://nikolaisammut.blogspot.de/2012/06/mobile-technologies.html

AUTOSAR SW Stack (Automotive)



www.autosar.org

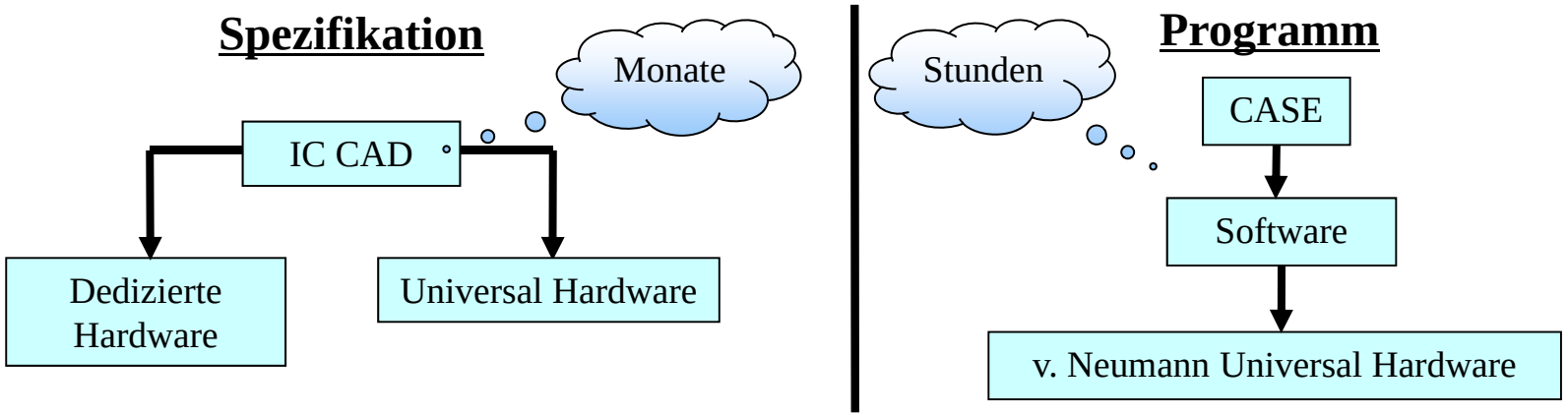
Rechnerarchitektur (Automotive)



© Freescale

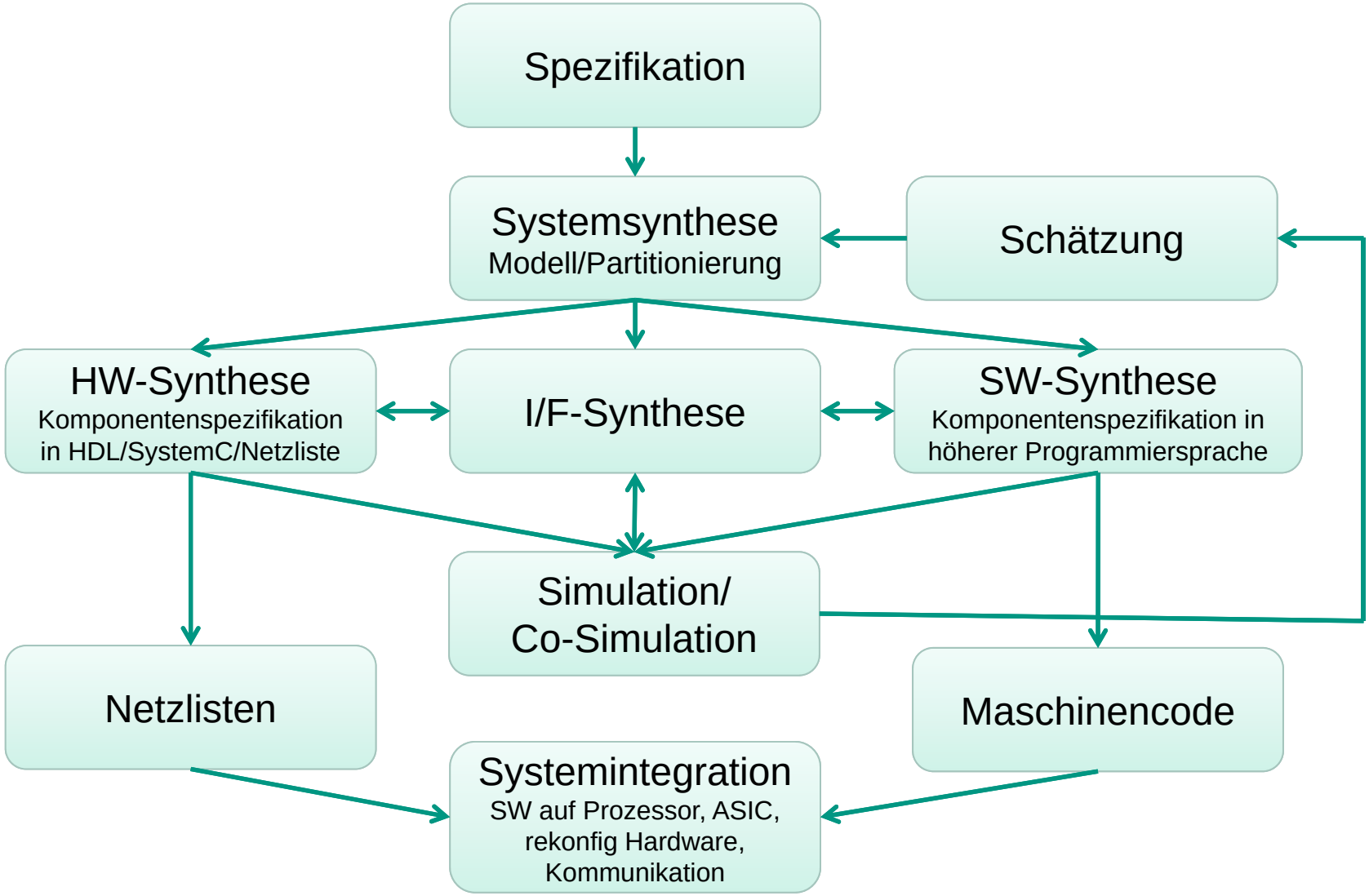
Hardware/Software Dichotomie

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



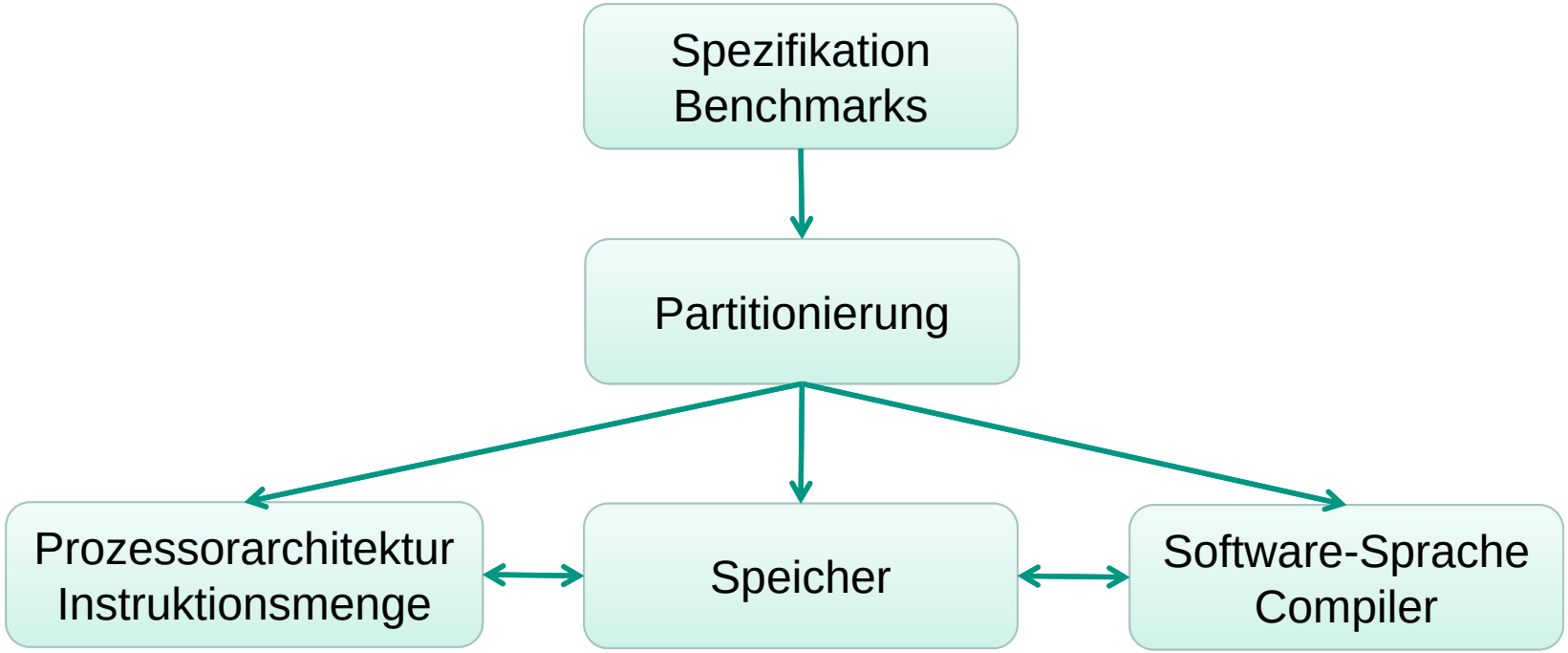
- Rekonfigurierbare Hardware zusammen mit Hardware/Software Co-Design durchbrechen diese Barriere

Entwurf von gemischten HW/SW Systemen



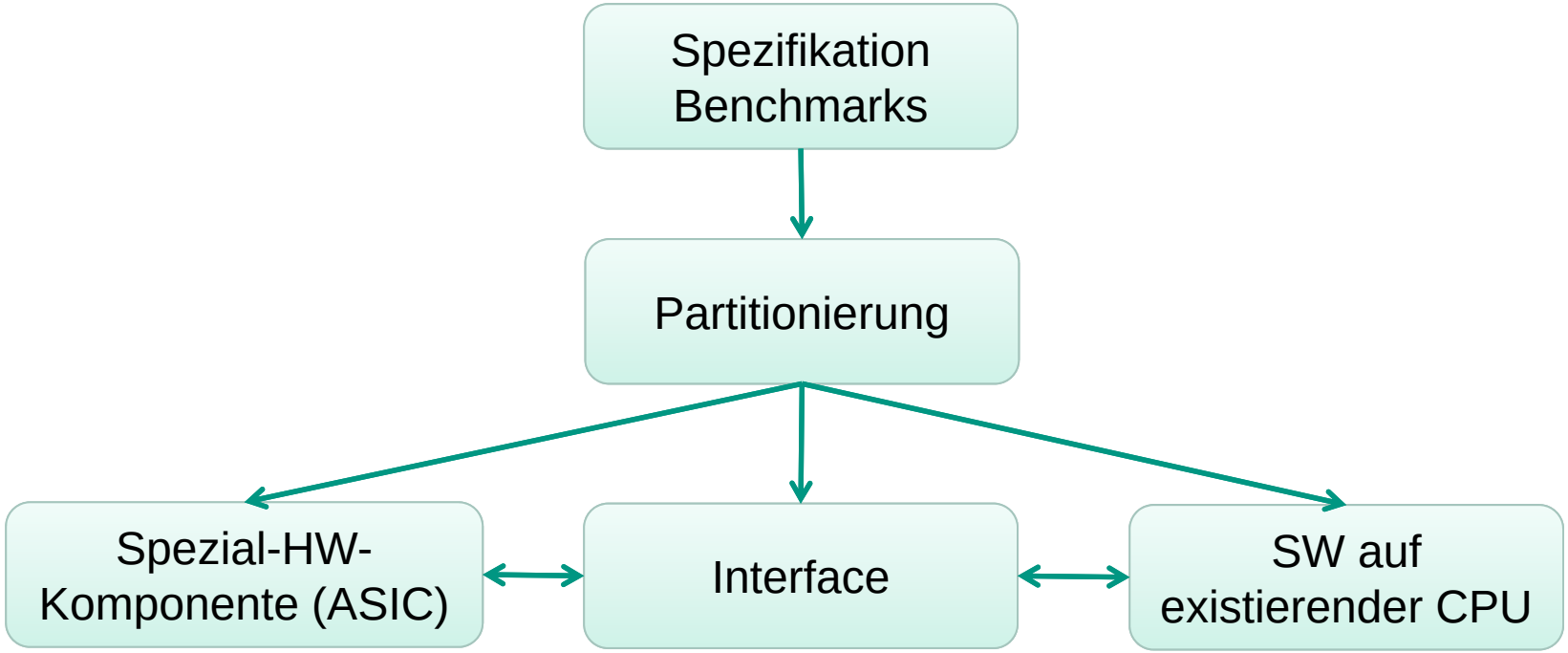
Systemprinzip – Typ I

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



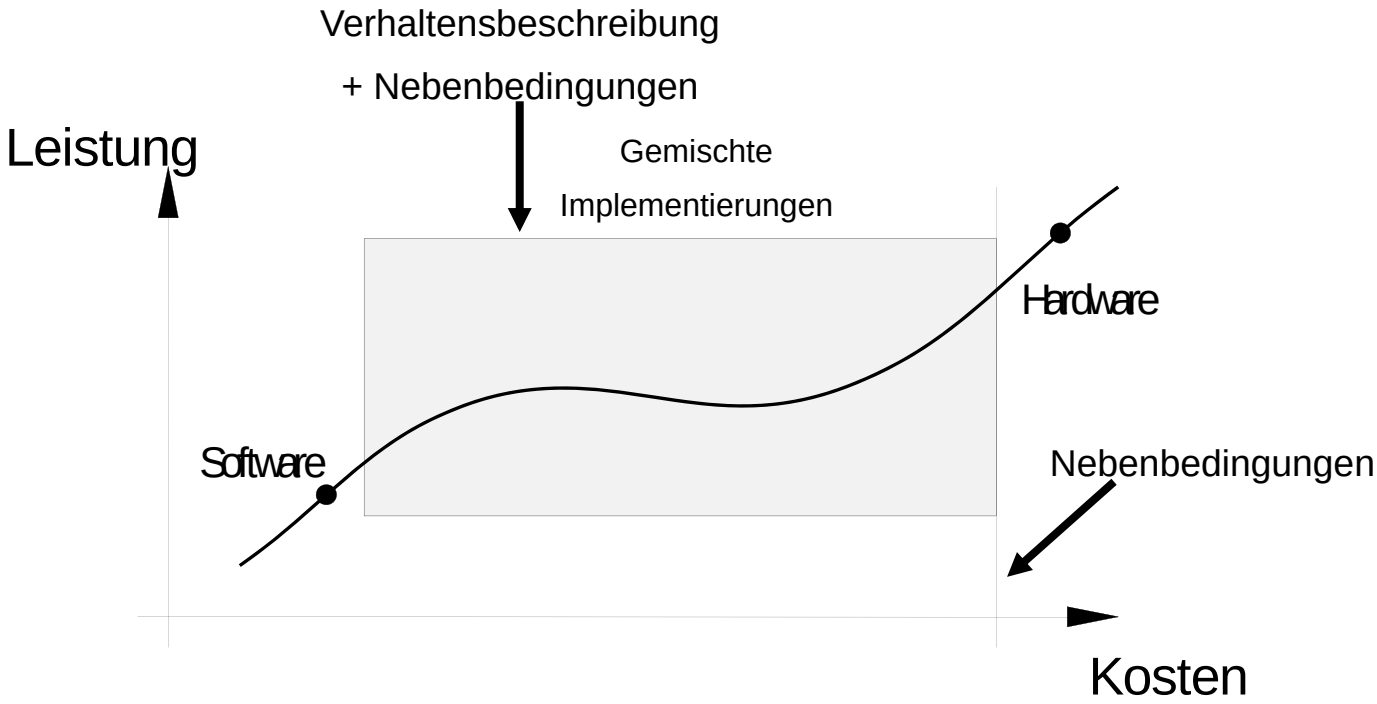
Systemprinzip – Typ II

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



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