System-Level Design (and Modeling for Embedded Systems)

Lecture 1 – Introduction & Overview

Kim Grüttner kim.gruettner@dlr.de
Henning Schlender henning.schlender@dlr.de
Jörg Walter joerg.walter@offis.de

System Evolution and Operation
German Aerospace Center (DLR)
&
Distributed Computation and Communication
OFFIS

Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Who we are

Dr. Kim Grüttner
DLR, EVO Department

Henning Schlender
DLR, EVO Department

Dr.-Ing. Jörg Walter
OFFIS, DCC Group

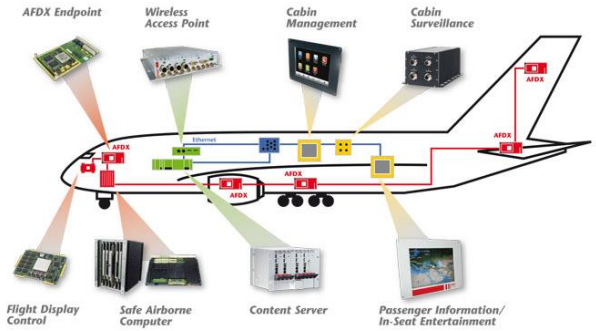
Sven Mehlhop
Uni Oldenburg, EHS

Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - Abstraction levels, design flow
  - System-level design
  - Design tasks, challenges and tools
- **Course information**
  - Administration
  - Topics
  - Materials
  - Policies
  - Timetable

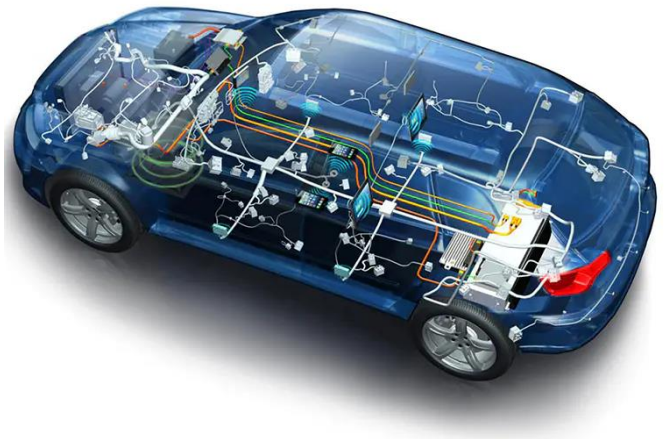
Embedded Systems

- **Systems that are part of a larger system**
  - Application-specific
    - Diverse application areas
  - Tight constraints
    - Real-time, performance, power, size
    - Cost, time-to-market, reliability



- **Ubiquitous**
  - Far bigger market than general-purpose computing (PCs, servers)
    - $86 billion in 2020, >$116 billion by 2025, 6.1% annual growth

[https://www.marketsandmarkets.com/Market-Reports/embedded-system-market-98154672.html]

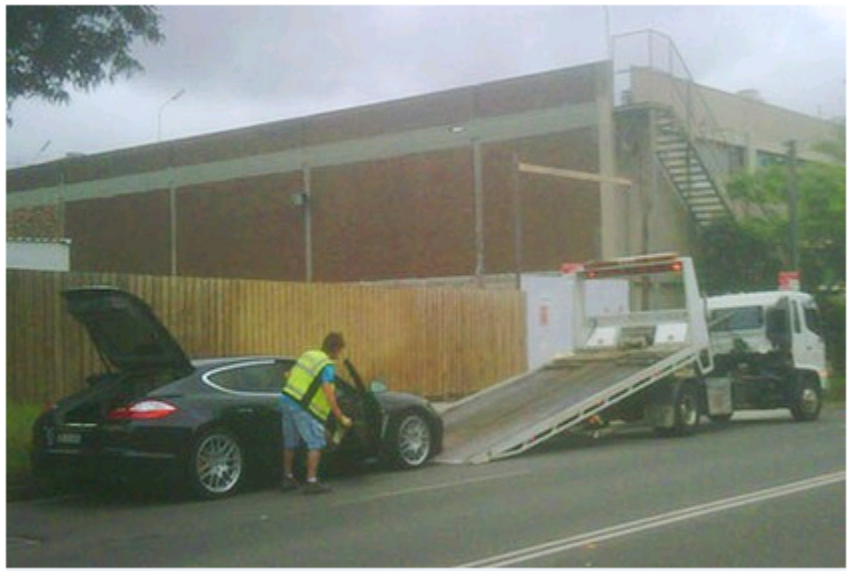


Embedded System Design is hard…

When car electronics go wrong

Toby Hagon
February 15, 2012

| Comments 96



Drive editor, Toby Hagon, got a nasty surprise when the electronics in a Porsche Panamera locked him and his family in the car.

They can make modern motoring more comfortable, but electronics also have the potential to endanger lives.

Modern cars may be more comfortable, cleaner and connected than ever, but they also have the potential to cause significant inconvenience, major traffic jams - or even endanger your life.

This morning an **Audi A8's entire electrical system shut down** while it was travelling south in the Eastern Distributor. It took two tow trucks to move it and caused extensive traffic jams.

Two years ago I discovered how badly things can go wrong with the electronics in modern cars when I was stuck in **a $400,000 Porsche that locked me and my family in the parked car.**

The Bug That Destroyed a Rocket

Mordechai Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
<moti.ben-ari@weizmann.ac.il>

Author's Note

In the 2000 December issue of *inroads*, Michael Williams suggested that the failure of the Ariane 5 rocket launch could be used as a case study in teaching programming concepts. Here is an article I wrote several years ago in which I present the story of the Ariane 5 in terms used to teach introductory computer science.

The morning of the 4th of June 1996 was partially cloudy at Kourou in Guyana as the European Space Agency (ESA) prepared for the first launch of the French-built Ariane 5 rocket. The rocket lifted off at 09:34. Just 37 seconds later, the rocket veered on its side and began to break up. The range safety mechanism identified the impending catastrophe and initiated explosive charges that blew up the rocket to prevent further damages and possible casualties. An investigation by the ESA determined that the accident was caused by a software 'bug . This is the story of that bug.

Sorry - a bit of physics

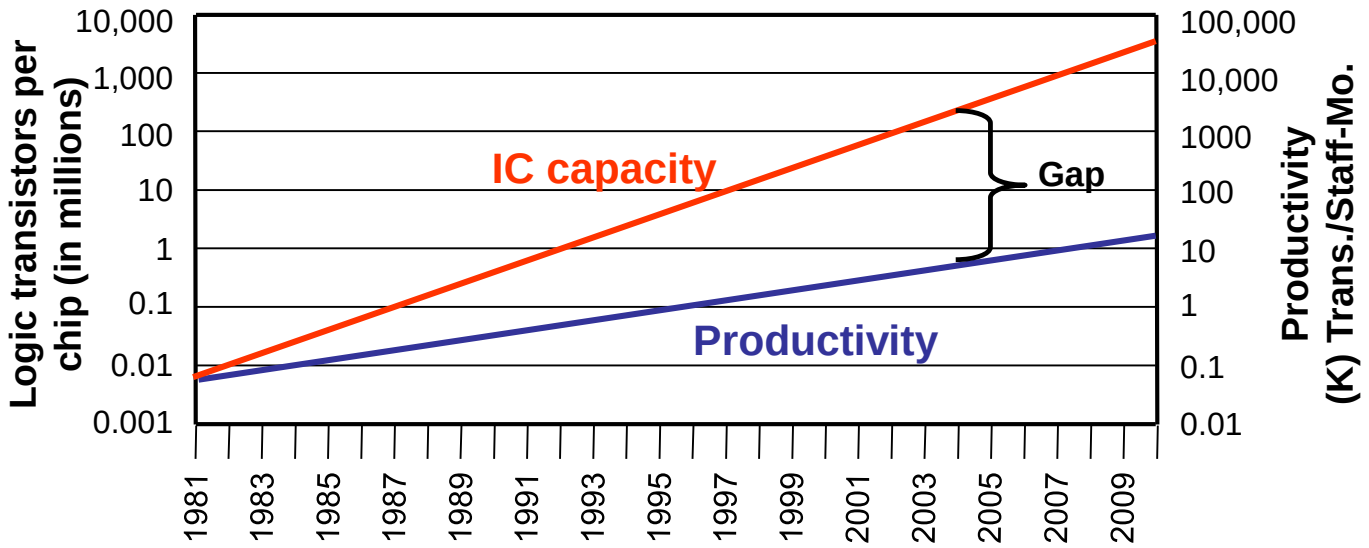
A short description of the physics involved will make the explanation of the bug more intelligible. Students whose health would be damaged by reading physics can skip this section.

Newton s third law explains how a rocket is launched. The backwards force of the jet stream from the nozzle is balanced by a forward force on the body of the rocket. However, the rocket, which is just a long narrow tube, is very unstable. The rocket must be steered by changing the

https://www.researchgate.net/publication/220612984_The_bug_that_destroyed_a_rocket

… and getting harder

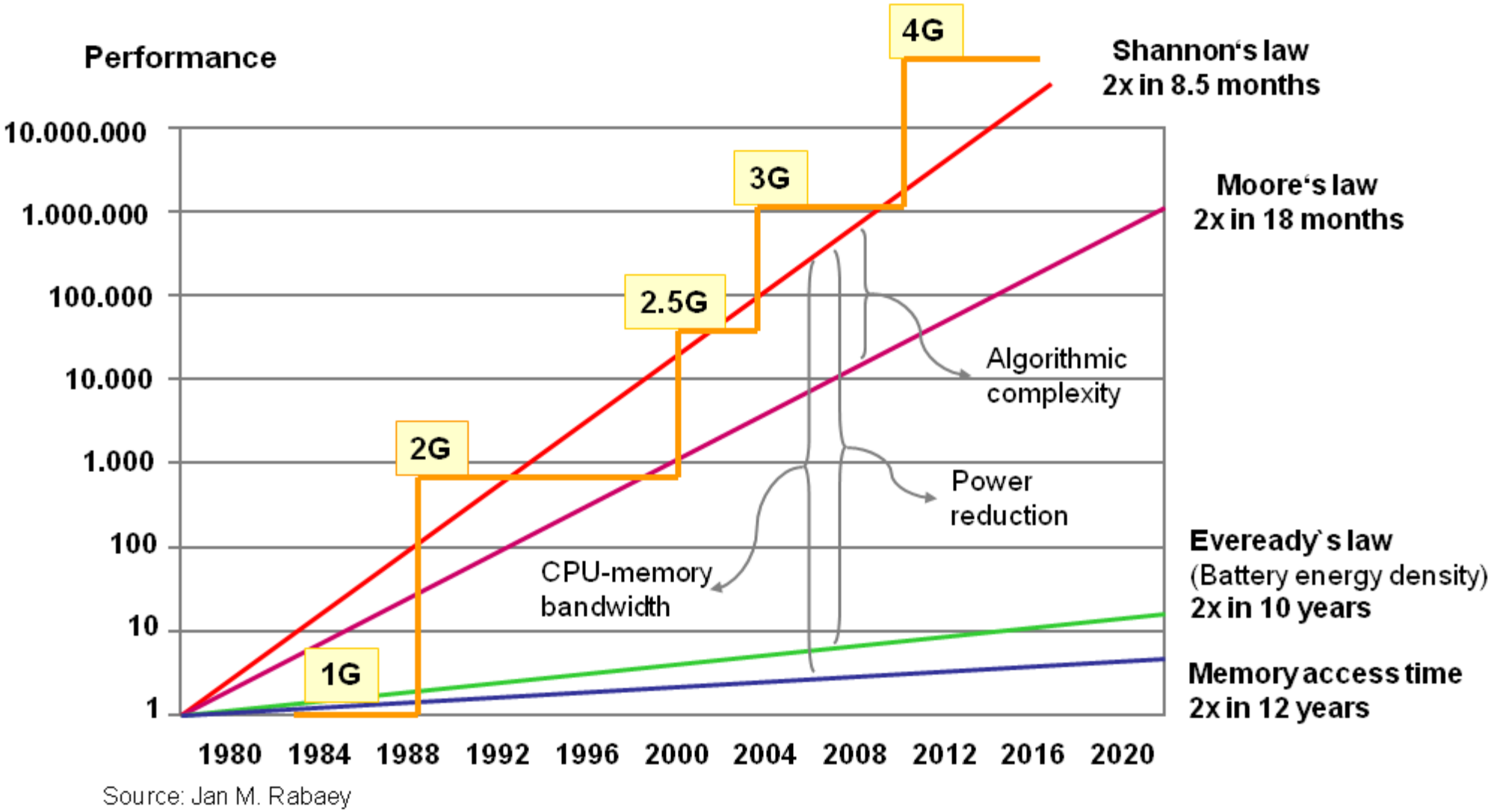
- **Growing system complexities**
  - Increasing application demands
    - Device convergence (multimedia, infotainment, GPS, …)
  - Technological advances
    - Multi-Core/Multi-Processor System-On-Chip (MPSoC)



Source: SEMATECH; Courtesy of: T. Givargis, F. Vahid. "Embedded System Design", Wiley 2002.

➢ **Raising the level of abstraction**

… and even harder for extra-functional properties



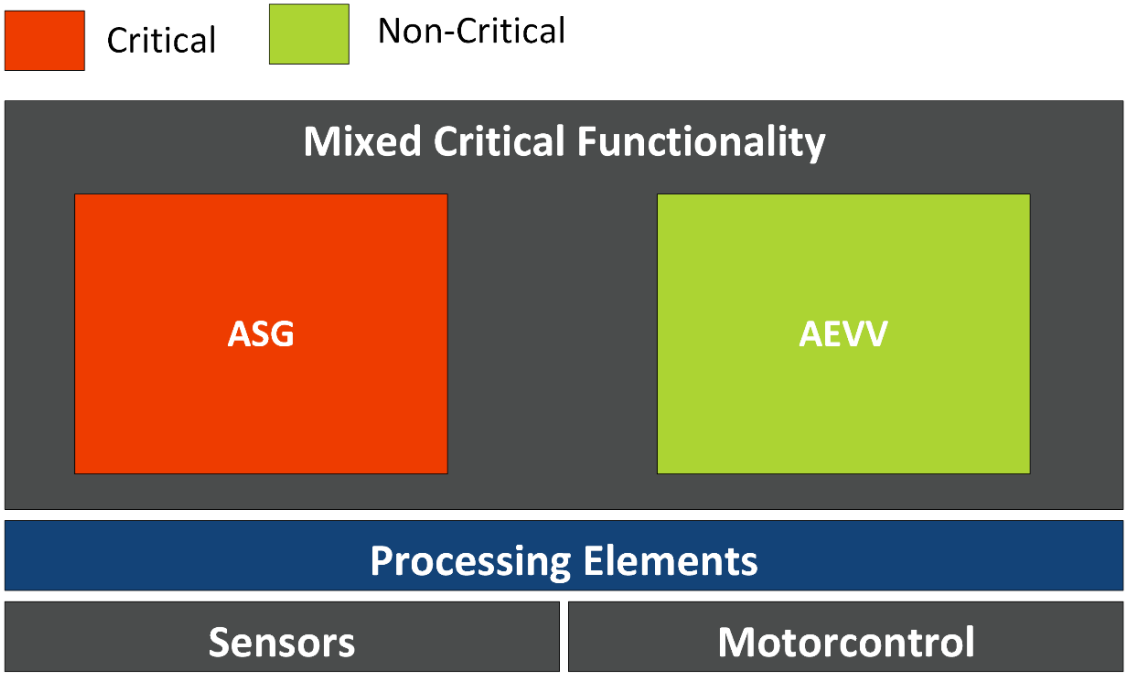
Source: Jan M. Rabaey

Multirotor Demonstrator

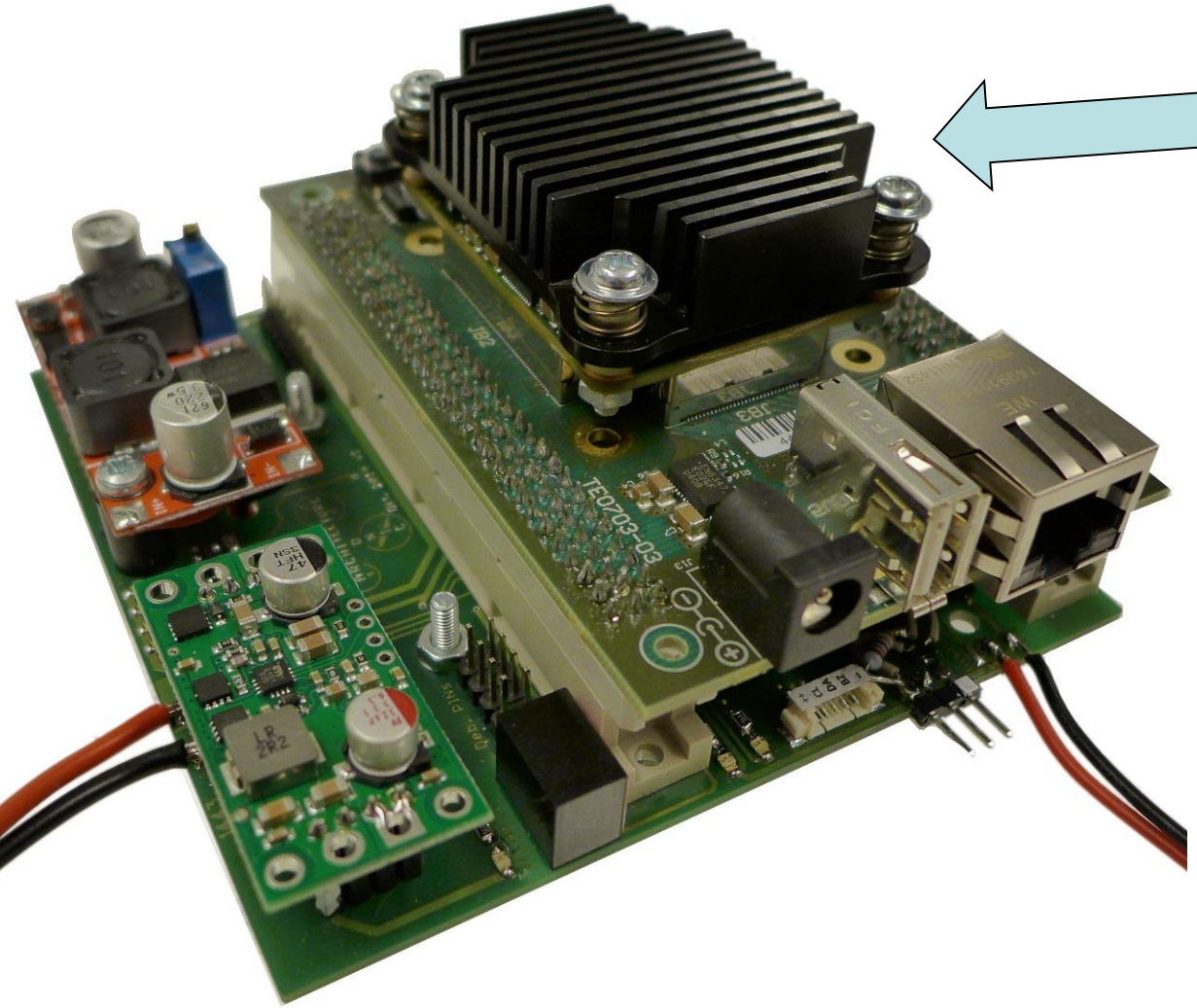
Mixed-Critical System

- Development of a **Mixed-Critical Systems on a single Chip**
- Basis: Quadrocopter
- **Safety Critical Application**: Avionics
  - Altitude control
  - Attitute control
- **Non-safety-critical** application:
  - Objekt tracking (image processing)

- Avionics control unit (ASG)Various hardware and software components that are prerequisites for controlling the flight behavior of the quadrocopter.
- Acquisition, detection, tracking and dispatch system (AEVV system)Functionalities and hardware components of the non-safety-critical application.



Hardware



- Xilinx Zynq 7020
  - ARM dual-core Cortex-A9 (866MHz)
  - Artix-7 FPGA (85k Logik Zellen)
- Ein- / Ausgänge
  - 152 FPGA I/Os
  - 14 Schnittstellen (I²C, UART, CAN, etc.)

Xilinx Zynq 7000

Xilinx Zynq

Dual ARM-Core

Objekt-verfolgung

FPGA

Flug- und Lageregelung

Processing System

Static Memory Controller
Quad-SPI, NAND, NOR

Dynamic Memory Controller
DDR3, DDR2, LPDDR2

Programmable Logic:
System Gates, DSP, RAM

I/O MUX

2x SPI

2x I2C

2x CAN

2x UART

GPIO

2x SDIO with DMA

2x USB with DMA

2x GigE with DMA

AMBA® Switches

AMBA® Switches

AMBA® Switches

ARM® CoreSight™ Multi-core and Trace Debug

NEON™/FPU Engine

NEON™/FPU Engine

Cortex™-A9 MPCore™ 32/32 KB I/D Caches

Cortex™-A9 MPCore™ 32/32 KB I/D Caches

ACP

512 KB L2 Cache

Snoop Control Unit (SCU)

Timer Counters

256 KB On-Chip Memory

General Interrupt Controller

DMA

Configuration

Multi Standards I/Os (3.3V & High Speed 1.8V)

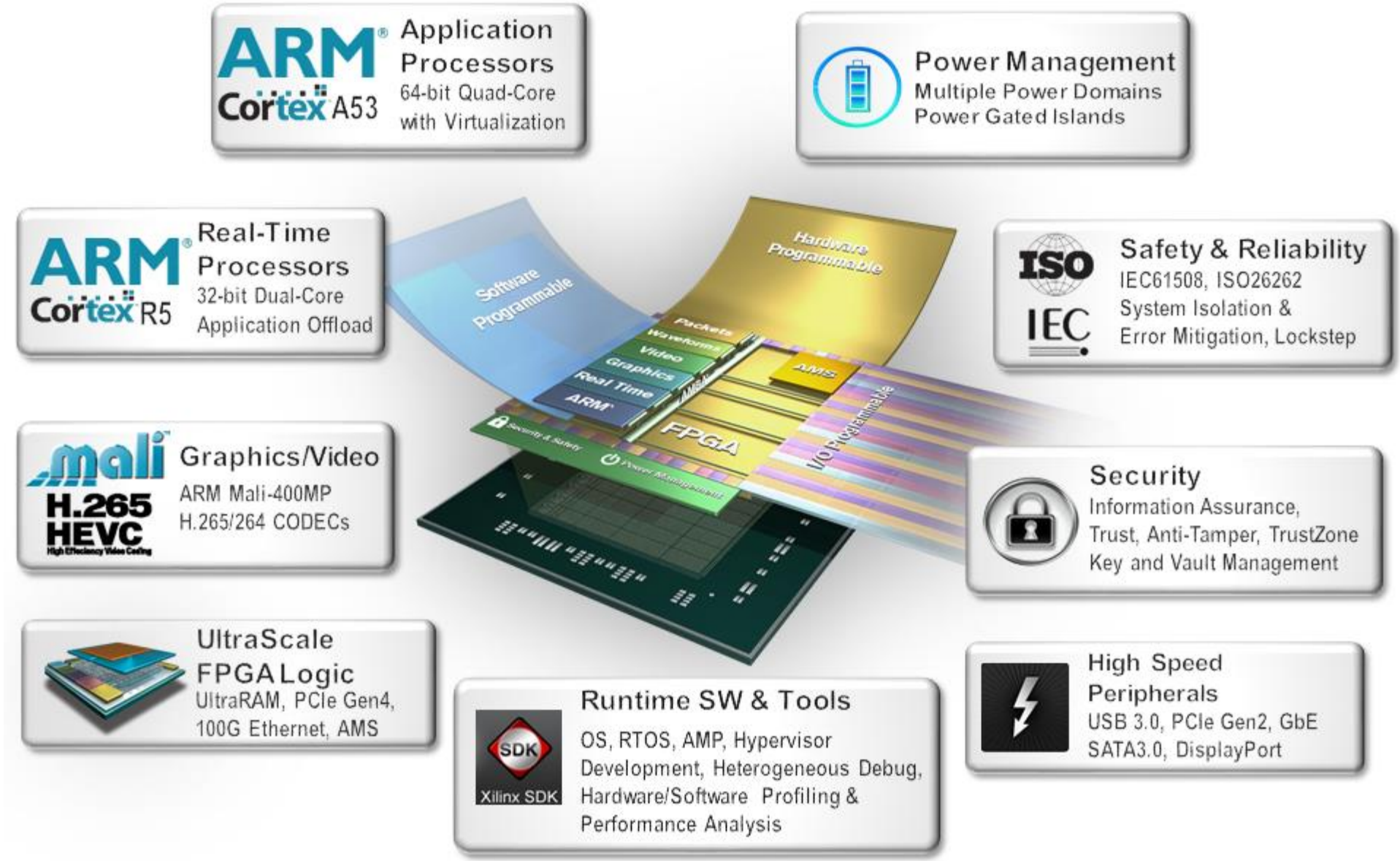
XADC

PCIe

Multi Standards I/Os (3.3V & High Speed 1.8V)

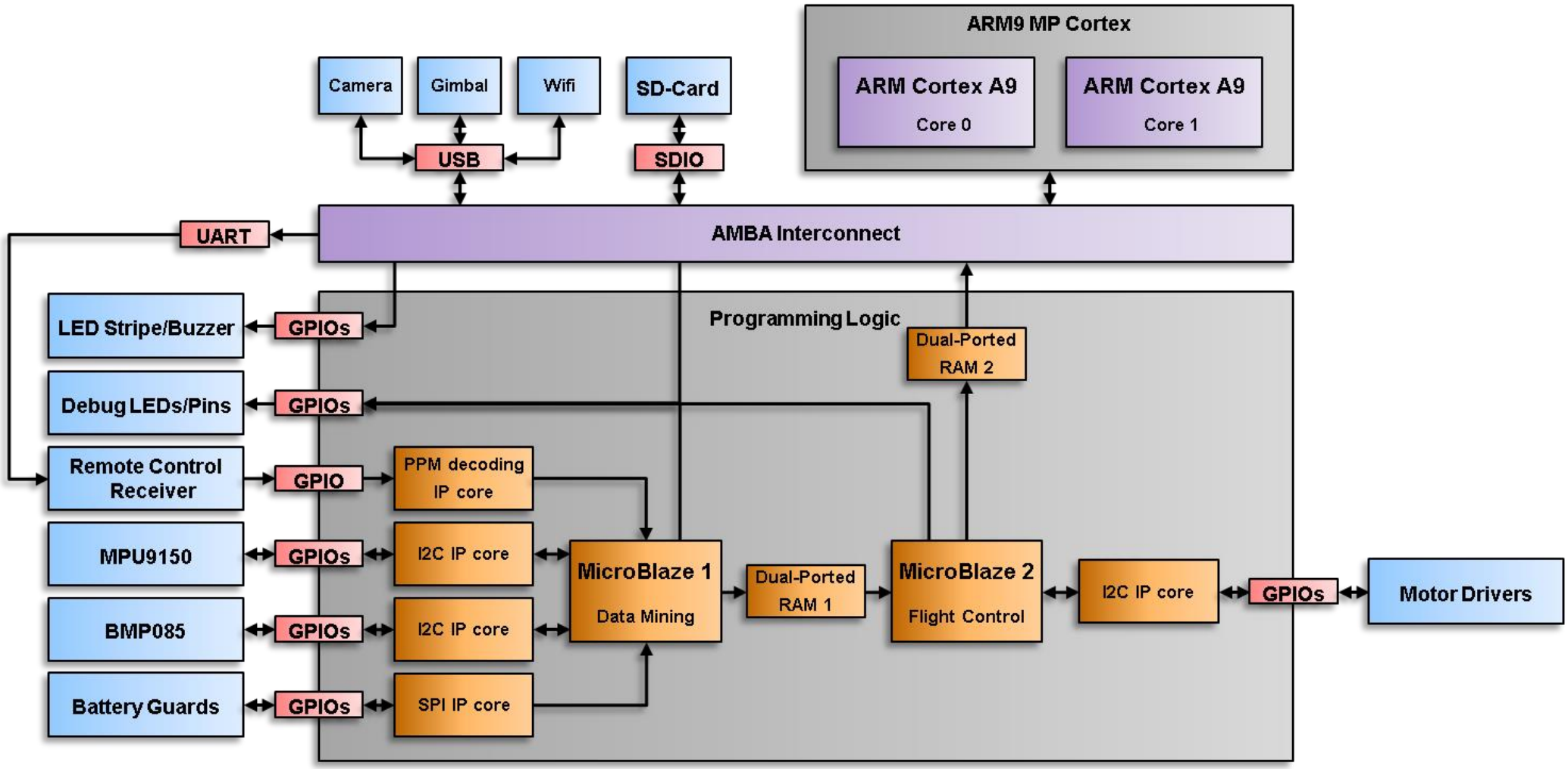
Multi Gigabit Transceivers

Multi-Processor System-On-Chip (MPSoC)
The Xilinx ZYNQ – UltraScale+ platform



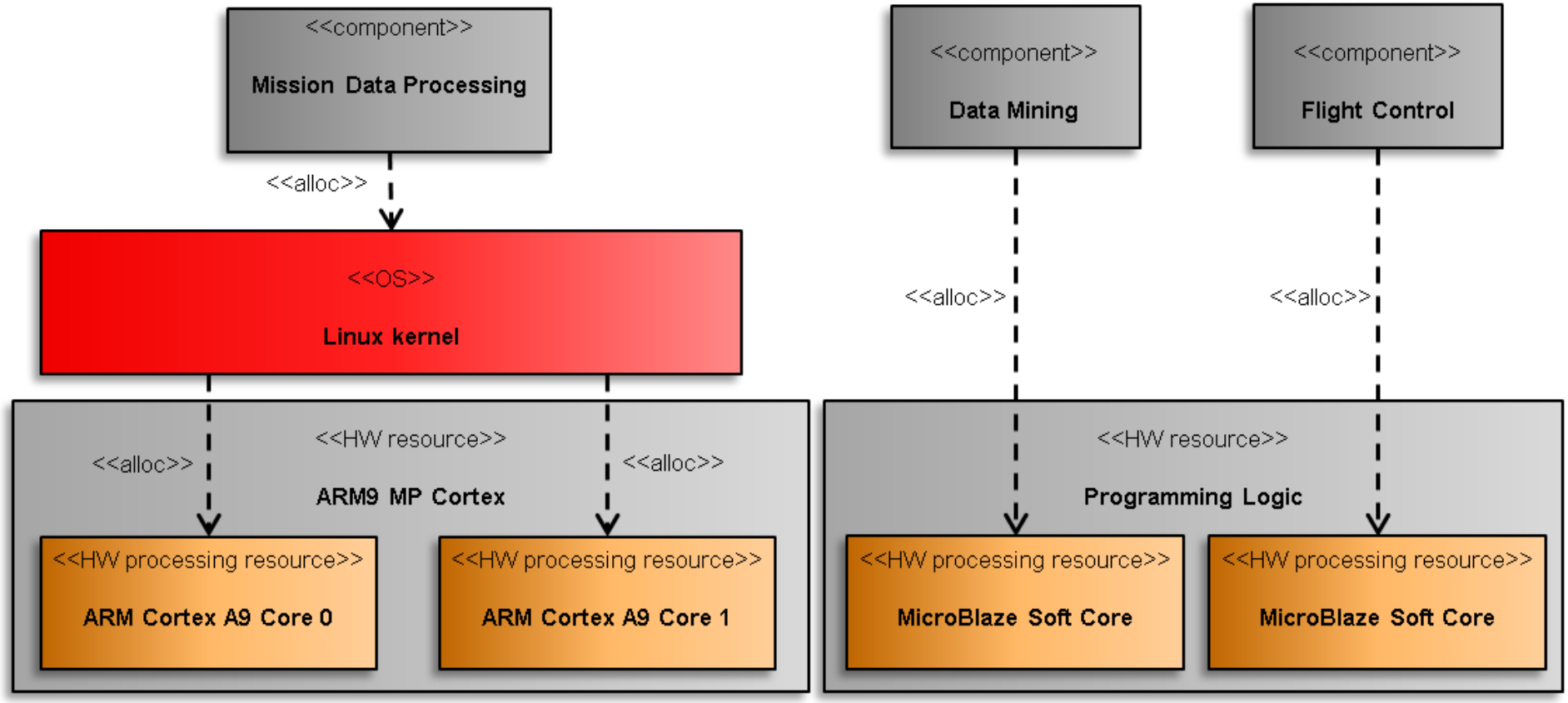
Source: http://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

HW Architecture

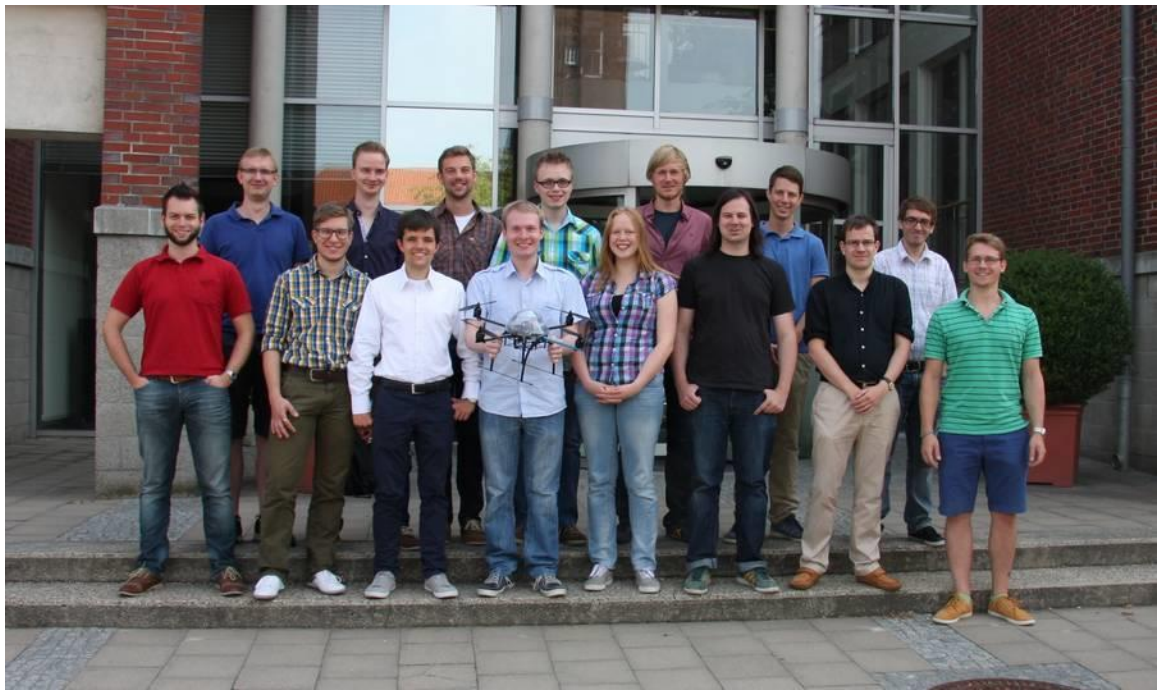


Mapping/Allocation

- Flight and attitude control
  - Execution on soft cores in the FPGA
  - Without support of an operating system
  - Requires interfaces for I2C, PPM and GPIO
- Object tracking
  - Runs on Dual ARM Core
  - Requires Linux as operating system
  - Multimedia libraries
  - Requires interfaces to USB and network



More info



Final Presentation:
https://uol.de/f/2/dept/informatik/ag/avionicarchitecture/PGAA_Praesentation_Rev4_20150331.pdf

Final Report:
https://uol.de/f/2/dept/informatik/ag/avionicarchitecture/PGAA_Dokumentation_final_20150414.pdf

Schlender, Henning; Stemmer, Ralf; Grüttner, Kim; Ehmen, Günter; Westphal, Bernd; Bruns, Friederike (2023): **Teaching Cyber-Physical Systems in Student Project Groups**. SEUH 2023. DOI: https://doi.org/10.18420/seuh2023_05. Gesellschaft für Informatik, Bonn. ISSN: 1617-5468. ISBN: 978-3-88579-727-2


S. Becker, C. Gerth. (Hrsg.): SEUH 2023,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 43

Teaching Cyber-Physical Systems in Student Project Groups: How Do Alumni Assess the Experience in Retrospective?

Henning Schlender[1] Ralf Stemmer[1] Kim Grüttner[1] Günter Ehmen[1] Bernd Westphal[1] Friederike Bruns[2]

Abstract: The development of Cyber-Physical Systems (CPS) requires software engineering competences in an interdisciplinary context including physics, mechatronics, and electrical engineering. In addition, social skills and teamwork competences are necessary since the current complexity of CPS exceeds the capacity of a single human. The Computer Science faculty at the University of Oldenburg offers project groups (PGs) in the Master's curriculum to particularly strengthen these competences. To this end, small teams of students work jointly during two semesters on large software or systems engineering problems, which are often inspired by research projects with industrial partners.

In this paper we report on a survey that we conducted among alumni of a set of nine PGs that worked on different challenging CPS problems. The main questions were how alumni who already started their professional career assess the learning experience and different aspects of the design of the CPS PGs in hindsight. The responses indicate a good learning success on the main learning objectives of the considered CPS PGs, and a large majority of respondents considers the course to be comparably relevant or very relevant in the overall Master's curriculum.

Keywords: Software Engineering, Systems Engineering, Cyber-Physical Systems, Project Group, Alumni-Survey

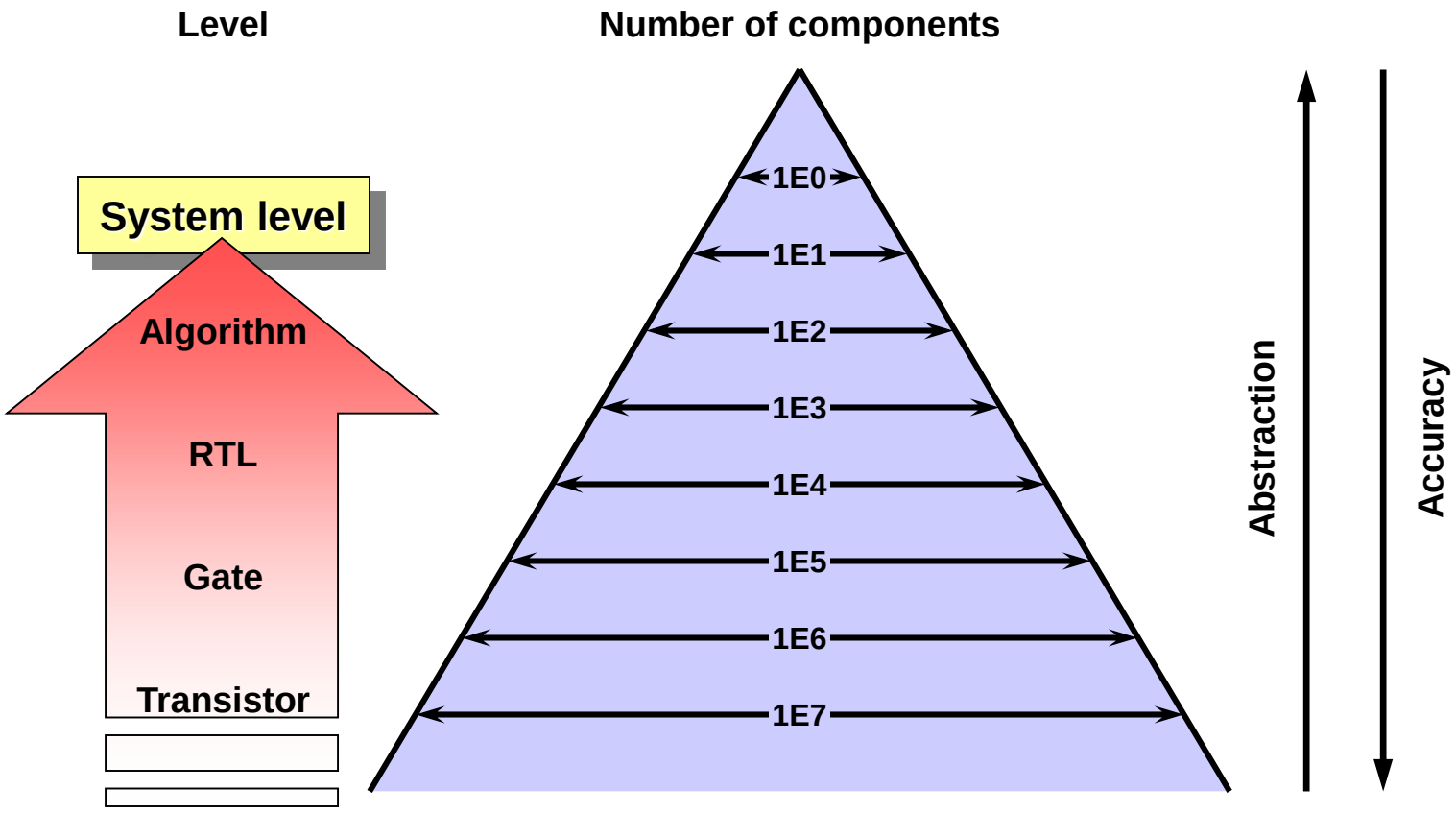
1 Introduction

Cyber-Physical Systems (CPS) are already omnipresent in our daily lives and are still increasingly entrusted with safety-critical tasks in domains such as transportation systems, medical devices, or factory automation. Among the characteristics of CPS are the aspects that they need to interact with a physical environment in a reliable, timely, and safe manner. Hence, the development of CPS requires competences from many different disciplines including physics, mechatronics, electrical engineering, and of course computer science. The latter contributes competences from areas such as embedded HW/SW design, real-time systems, control theory, and software and systems engineering methods that deal with the high quality demands, the particular care that is needed for functional and extra-functional properties, as well as mixed criticality components and Safety Integrity Levels (SIL). Last

[1] German Aerospace Center, Institute of Systems Engineering for Future Mobility, Oldenburg, Germany
[2] Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany
[1] henning.schlender@dlr.de, ralf.stemmer@dlr.de, kim.gruettner@dlr.de, guenter.ehmen@dlr.de, bernd.westphal@dlr.de, friederike.bruns@uol.de

Abstraction Levels

- **Growing system complexities**
- ➢ **Move to higher levels of abstraction [ITRS07, itrs.net]**
  - ➢ Electronic system-level (ESL) design

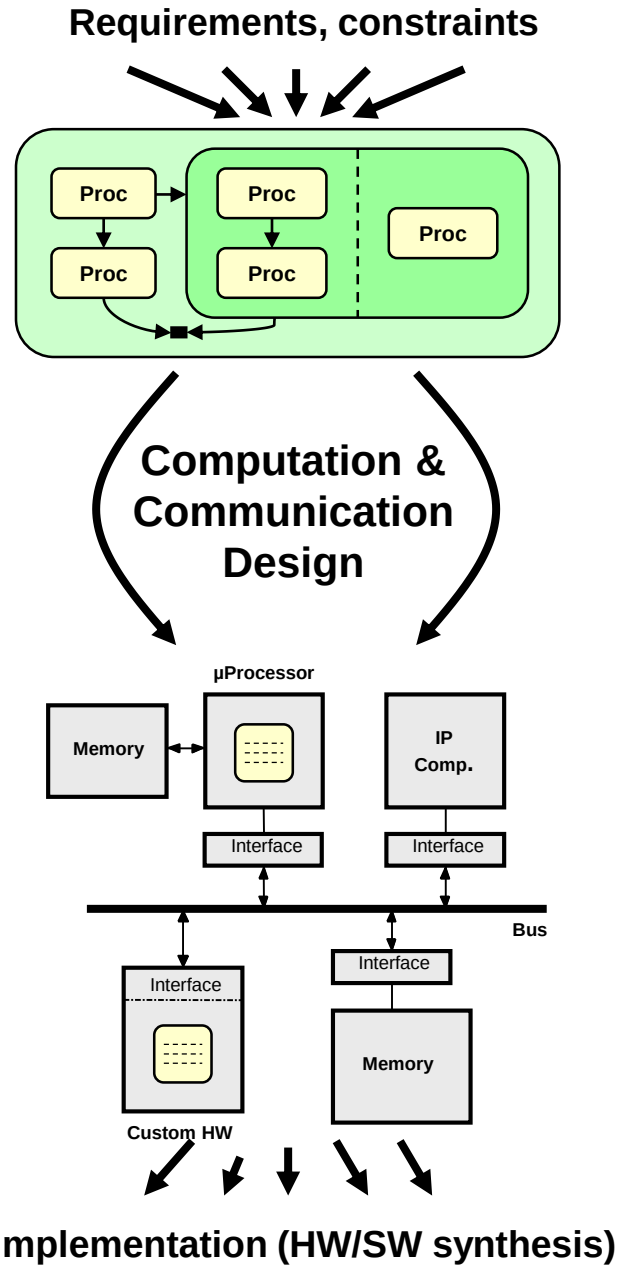


Source: R. Doemer, UC Irvine

System-Level Design

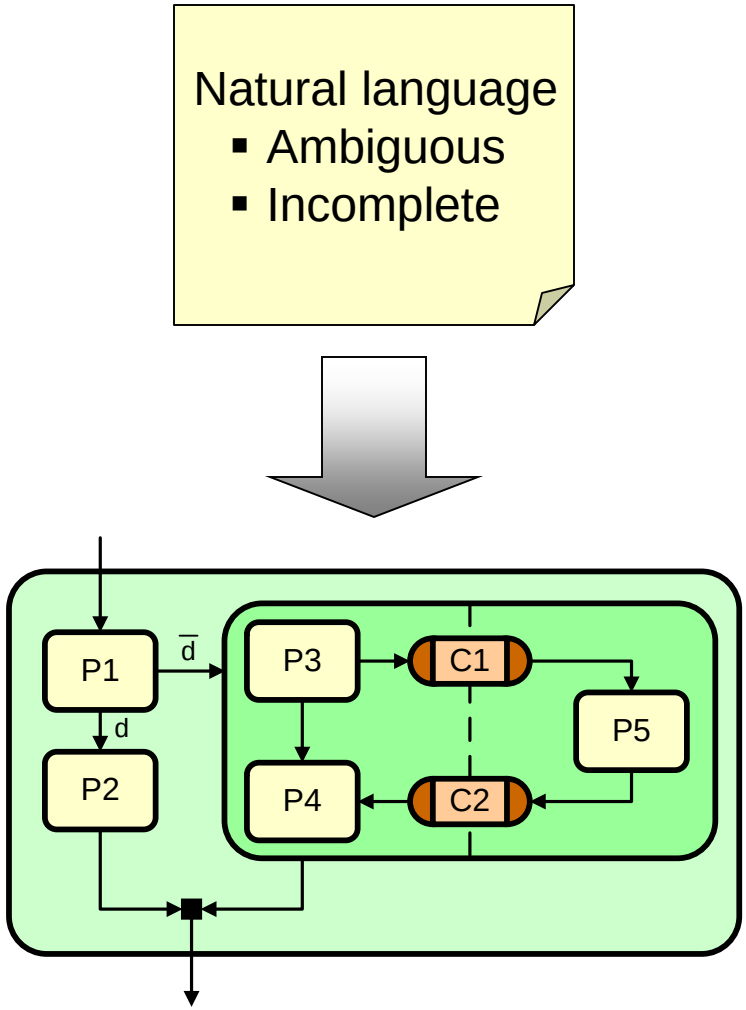
- **From system specification**
  - Functionality, behavior
    - Application algorithms
    - Constraints
- **To system architecture**
  - Structure
    - Spatial and temporal order
    - Components and connectivity
    - Across hardware and software

➢ **Design automation at the system level**
  - Modeling and simulation
  - Synthesis
  - Verification



System Specification

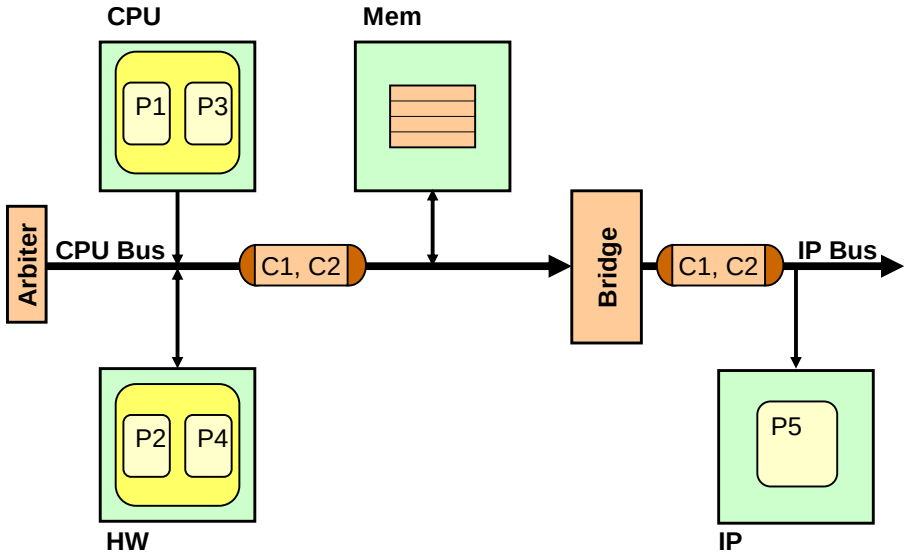
- **Capture requirements**
  - Functional
    - Free of any implementation details
  - Non-functional
    - Quality metrics, constraints
- **Formal representation**
  - Models of computation
    - Analysis of properties
  - Executable
    - Validation through simulation



- **Application development**
  - Precise description of desired system behavior

System Architecture

- **Processing elements (PEs)**
  - Processors
    - General-purpose, programmable
    - Digital signal processors (DSPs)
    - Application-specific instruction set processor (ASIP)
    - Custom hardware processors
    - Intellectual property (IP)
  - Memories



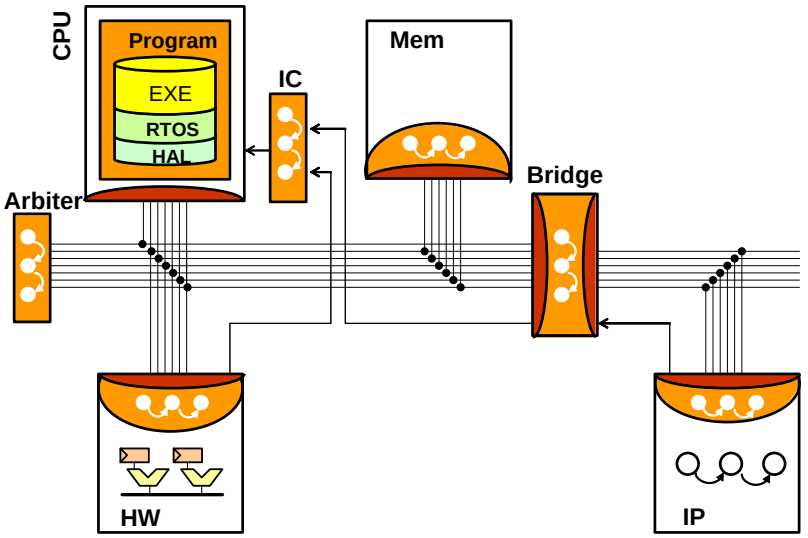
- **Communication elements (CEs)**
  - Transducers, bus bridges
  - I/O peripherals
- **Busses**
  - Communication media
    - Parallel, master/slave protocols
    - Serial and network media

➢ **Heterogeneous multi-processor systems**

➢ Multi-Processor System-on-Chip (MPSoC)

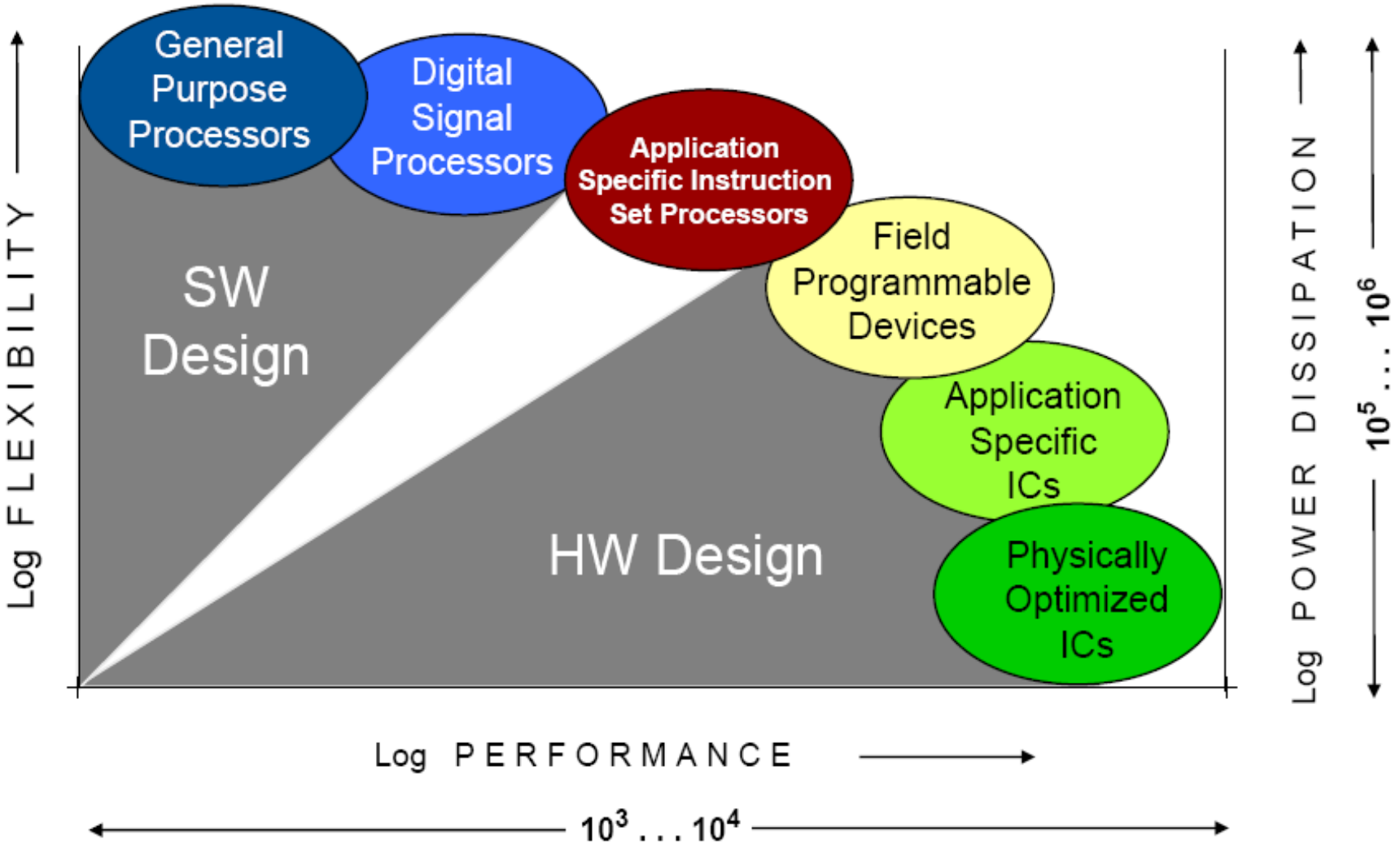
System Implementation

- **Hardware**
  - Microarchitecture
  - Register-transfer level (RTL)
- **Software**
  - Application (object) code
  - Real-time operating system (RTOS)
  - Hardware abstraction layer (HAL)
- **Interfaces**
  - Pins and wires
  - Arbiters, muxes, interrupt controllers (ICs), etc.
  - Bus protocol state machines



- **Further logic and physical synthesis**
  - Manufacturing
  - Prototyping boards

Processor Implementation Options



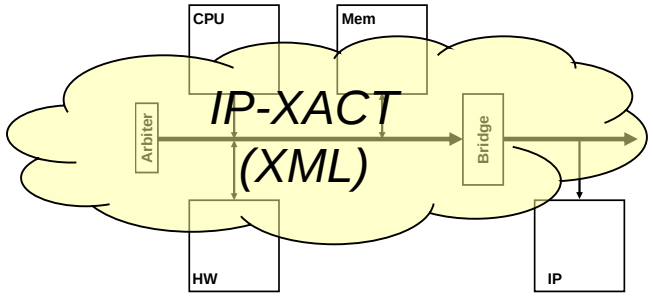
Source: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

Design Challenges

- **Design quality metrics and constraints**
  - Performance
    - Latency and throughput
  - Power
    - Static and dynamic power consumption
  - Cost
    - Unit and non-recurring engineering (NRE) costs
  - Dependability and reliability
    - Fault tolerance, safety, correctness, mean time between failure (MTBF)
  - Management
    - Time-to-market, maintainability, flexibility
  - …

- **Multi-objective optimization in vast design space**
  - Complexity
    - High degree of parallelism, large number of implementation options
  - Heterogeneity
    - Different types of components and applications, irregular architectures

Electronic System-Level (ESL) Flow

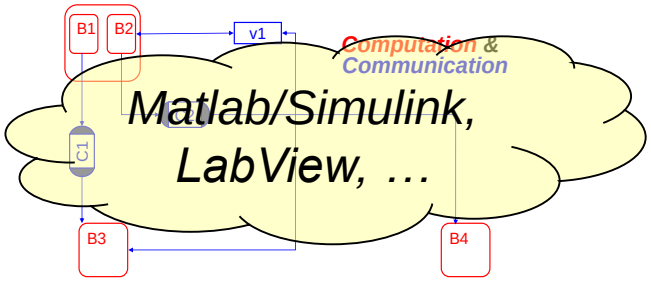


IP-XACT (XML)

Platform library

MARTE (UML)

System Synthesis Front-End



Matlab/Simulink, LabView, …

Application specification

System-Level Design Languages (SLDLs)

VaST, QEMU, Imperas OVP,…

Instruction-Set Simulator (ISS)

Transaction-Level Models

TLM_n

SystemC, SNPS Platform Achitect, …

RTL-based

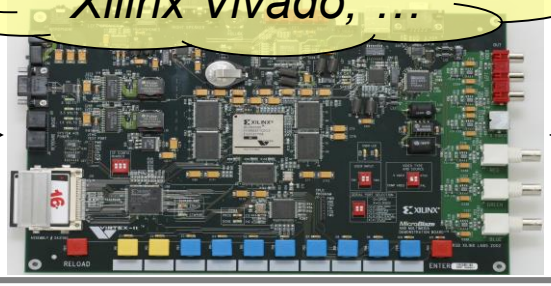
C/C++ code

Green Hills, gcc, VxWorks, …

Software / Hardware Synthesis Back-End

Mentor Catapult, Vivado HLS…

Xilinx Vivado, …

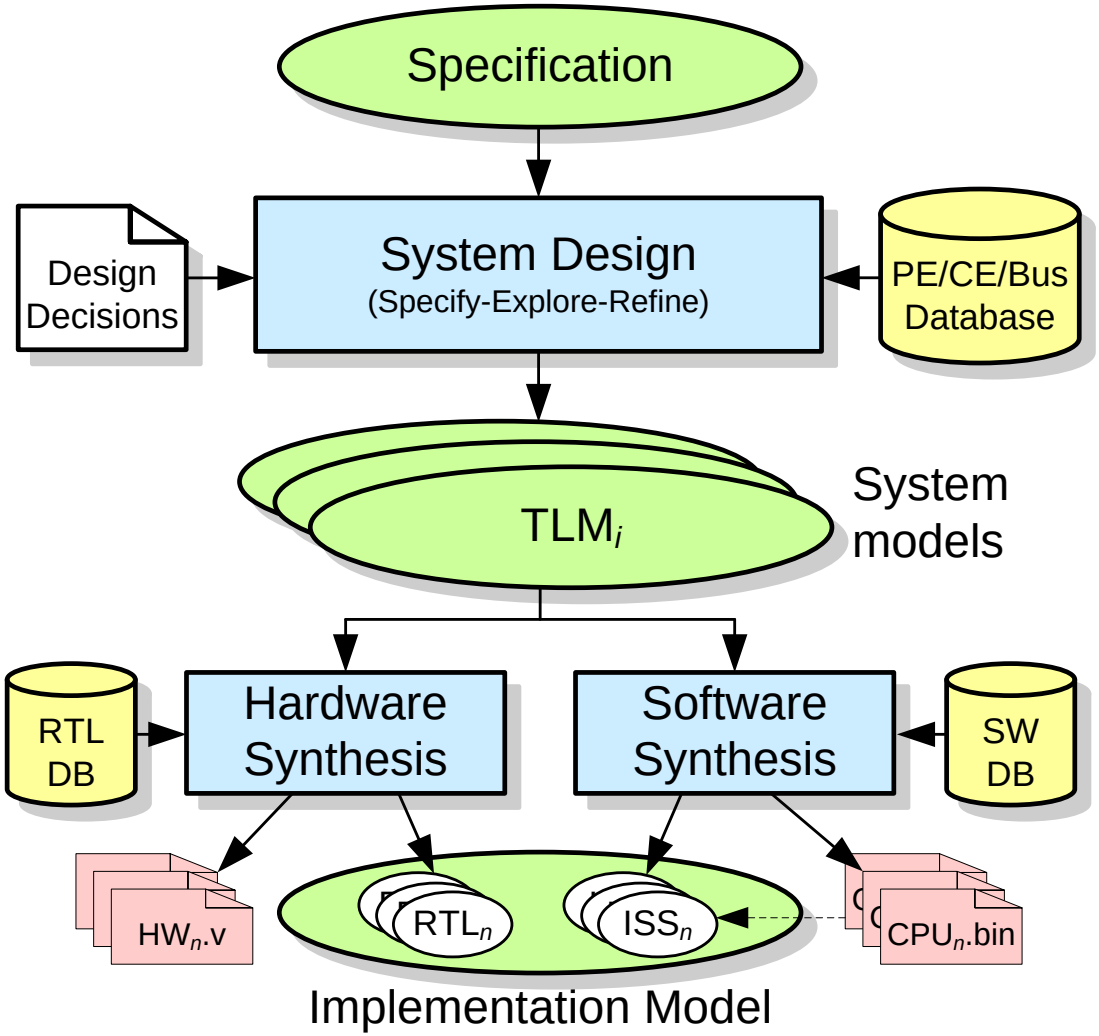


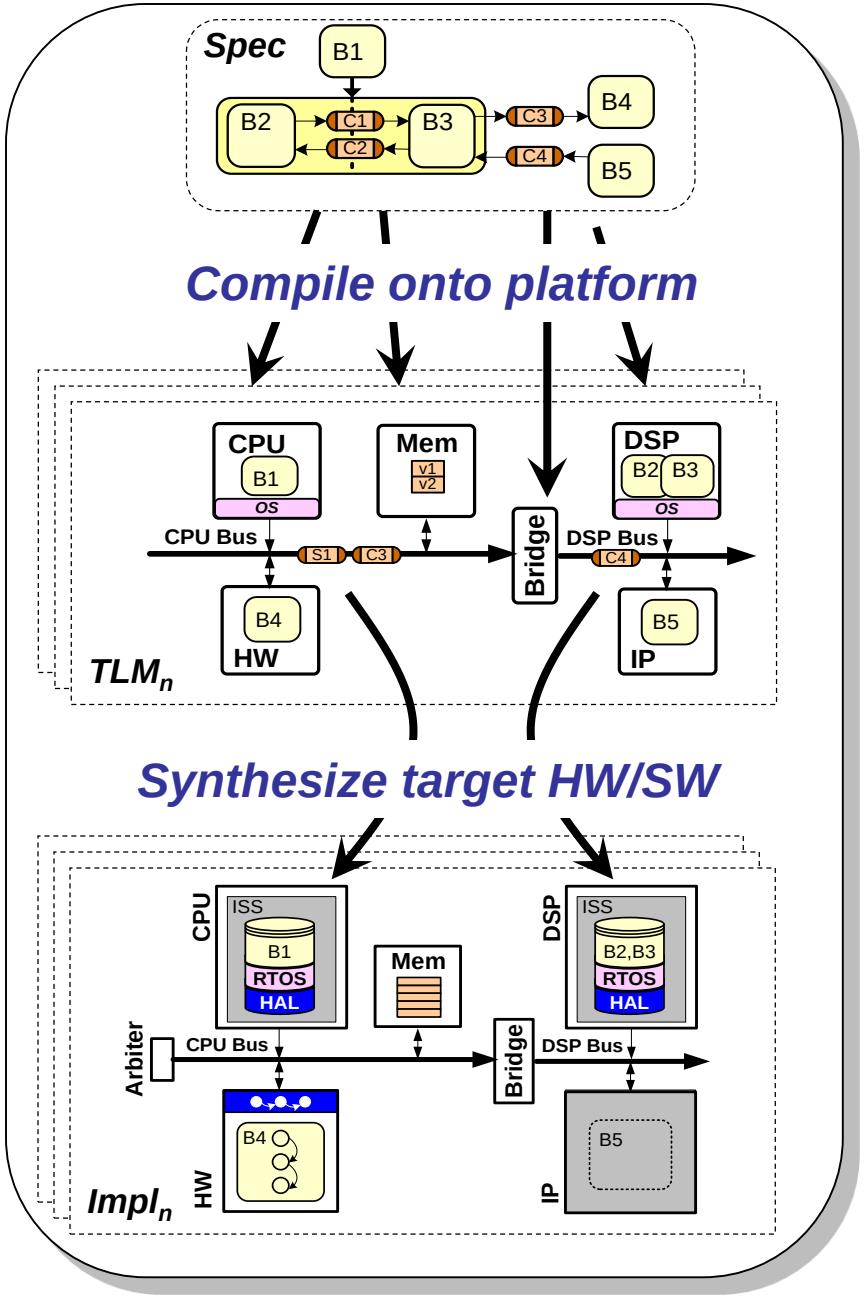
Software Object Code

Hardware VHDL/Verilog

SNPS Design Compiler, …

System-On-Chip Design Flow





Lecture 1: Outline

- ✓ **Introduction**
  - ✓ Embedded systems
  - ✓ Abstraction levels, design flow
  - ✓ System-level design
  - ✓ Design tasks, challenges and tools
- **Course information**
  - Administration
  - Topics
  - Materials
  - Policies
  - Timetable

Class Administration

- **Schedule**
  - See Timetable (next slide) and Stud.IP (will be updated)
- **Instructor(s)**
  - First part (SLD theory and SpecC)
    - Kim Grüttner (kim.gruettner@dlr.de) – main responsible
    - Henning Schlender (henning.schlender@dlr.de)
  - Second part (SystemC)
    - Jörg Walter (joerg.walter@offis.de) – main responsible
    - Sven Mehlhop (sven.mehlhop@offis.de)

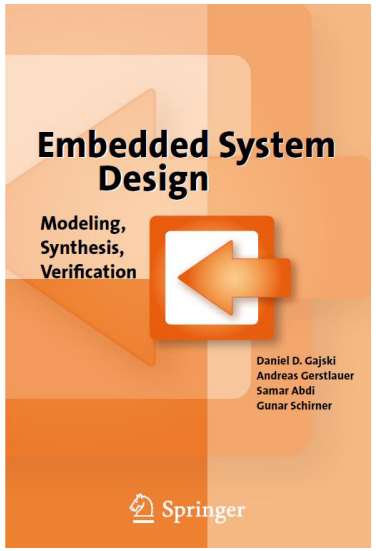
  Office hours: by appointment
- **Information**
  - Announcements, assignments: Stud.IP
  - Questions, discussions: Stud.IP
- **Examination**
  - Oral examinations
  - Duration: 30 min (15 min SLD theory and SpecC & 15 min SystemC)
  - Examinar: Kim Grüttner for SLD theory and SpecC, Jörg Walter for SystemC
  - Dates for oral examination: Between August and September 2024

Time Table

Day	Date	Star	End	Type	Topic	Lecurer(s)	Literature
Monday	08.04.2024	12:15	13:45	Lecture	Introduction	Kim Grüttner	lecture slides only
Monday	08.04.2024	14:15	15:45	Lecture	Methodologies, Models, Languages	Kim Grüttner	Orange book, Chapter 2, 3.2 - 3.3
Monday	15.04.2024	12:15	13:45	Excercise	Methodologies, Models, Languages	Kim Grüttner	Exercise 1
Monday	15.04.2024	14:15	15:45	Lecture	Models of Computation	Kim Grüttner	Orange book, Chapter 3.1
Monday	22.04.2024	12:15	13:45	Excercise	Models of Computation	Kim Grüttner	Exercise 2 (Task 1 - 4)
Monday	22.04.2024	14:15	15:45	Lecture	The SpecC Language	Kim Grüttner	Yellow book, Chapter 1
Monday	29.04.2024	12:15	13:45	Lecture	Specification & Refinement I	Kim Grüttner	Orange Book, Chapter 7 with focus on 7.4
Monday	29.04.2024	14:15	15:45	Lecture/ Excercise	Specification & Refinement II & The SpecC Language	Kim Grüttner	Orange Book, Chapter 7 with focus on 7.4, Setup of SpecC environment, Exercise 2 (Task 5)
Monday	06.05.2024	12:15	13:45	Lecture	System Synthesis and Exploration	Kim Grüttner	Orange book, Chapter 4, 5 & 6
Monday	06.05.2024	14:15	15:45	Excercise	Refinement in SpecC	Kim Grüttner	Exercise 3 (Task 1)
Monday	13.05.2024	12:15	13:45	Lecture	Computation Modeling & Refinement	Kim Grüttner	Orange book, Chapter 3.4
Monday	13.05.2024	14:15	15:45	Lecture	Communication Modeling & Refinement	Kim Grüttner	Orange book, Chapter 3.5
Monday	27.05.2024	12:15	13:45	Excercise	Communication Modeling & Refinement	Kim Grüttner	Exercise 3 (Task 2-4)
Monday	27.05.2024	14:15	15:45	Lecture	SystemC Part I	Jörg Walter	Red Book
Monday	03.06.2024	12:15	13:45	Lecture	Introduction to C++ for SystemC	Jörg Walter	lecture slides only
Monday	03.06.2024	14:15	15:45	Excercise	SystemC Part I (Excercise)	Jörg Walter	Excercise 4
Monday	10.06.2024	12:15	13:45	Lecture	SystemC Part II	Jörg Walter	Red Book
Monday	10.06.2024	14:15	15:45	Excercise	SystemC Part II (Excercise)	Jörg Walter	Excercise 5
Monday	17.06.2024	12:15	13:45	Lecture	SystemC TLM Part I	Jörg Walter	Red Book (2nd edition only)
Monday	17.06.2024	14:15	15:45	Excercise	SystemC TLM (Excercise)	Jörg Walter	Excercise 6 (Task 1+2)
Monday	24.06.2024	12:15	13:45	Lecture	SystemC TLM Part II	Jörg Walter	Red Book (2nd edition only)
Monday	24.06.2024	14:15	15:45	Excercise	SystemC TLM (Excercise)	Jörg Walter	Excercise 6 (Task 3)
Monday	01.07.2024	12:15	13:45	Lecture	SystemC SW Refinement: TLM	Jörg Walter	Red Book (2nd edition only)
Monday	01.07.2024	14:15	15:45	Excercise	SystemC SW Refinement: TLM (Excercise)	Jörg Walter	Excercise 7 (Task 1+2)
				Lecture	*SystemC SW Refinement: ISS*	*Jörg Walter*	*Red Book (2nd edition only)*
				Excercise	*SystemC SW Refinement: ISS (Excercise)*	*Jörg Walter*	*Excercise 7 (Task 3)*
TBD	*TBD*	*TBD*	*TBD*	*Q&A*	*Q&A Session*	*Kim Grüttner/ Jörg Walter*	*Q&A handout*

Textbooks (1)

- **Main textbooks**
  - D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner, *Embedded System Design: Modeling, Synthesis, Verification*, Springer, 2009 ("*orange book*")
  - E-Book uploaded at Stud.IP (only for personal usage in this lecture!)

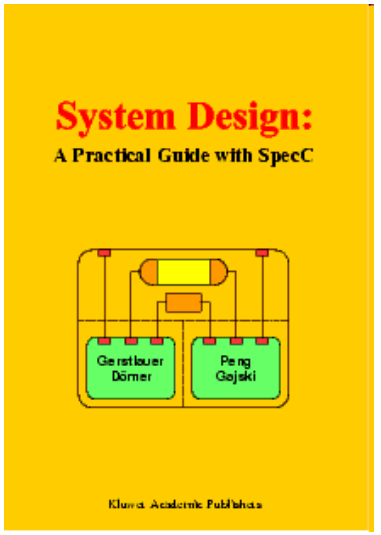


  - D. C. Black, J. Donovan, B. Bunton, A. Keist, *SystemC: From The Ground Up, Second Edition*, Springer 2010 (*"red book"*)
  - E-Book uploaded at Stud.IP (only for personal usage in this lecture!)

Textbooks (2)

- **Optional books (cont'd)**
  - F. Vahid, T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, Wiley, John & Sons, 2001 (*"blue book"*)
    - Background about embedded systems in general

  - A. Gerstlauer, R. Doemer, J. Peng, D. Gajski, *System Design: A Practical Guide with SpecC*, Kluwer, 2001 ("*yellow book*")
    - Practical, example-driven introduction using SpecC
    - Slides uploaded at Stud.IP



➢ **Additional reading material posted on Stud.IP**