technische universität dortmund

CoaCh

Car on a Chip: Neue Steuergeräte-Architekturen mit Systems-on-Chip im Automobilbereich

Olaf Spinczyk
Horst Schirmeier
Jochen Streicher
Michael Engel

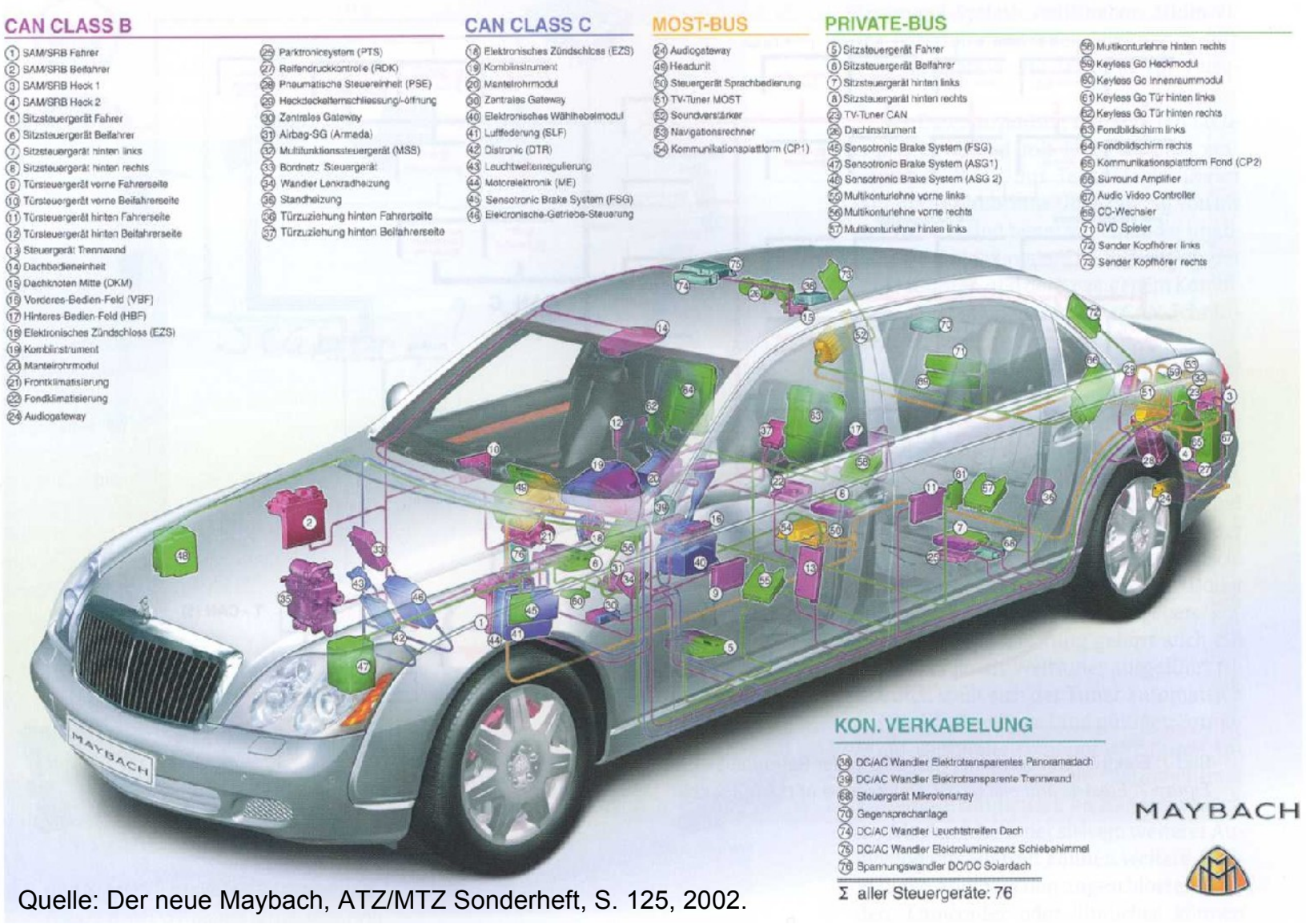
Lehrstuhl XII
AG Eingebettete Systemsoftware
Fakultät für Informatik

http://ess.cs.uni-dortmund.de/DE/Teaching/PGs/coach/

Agenda

- **Einführung**
- Vision und Minimalziele
- Durchführung
- Seminarthemen
- Nächste Schritte
- **Fragen**

Steuergeräte im Automobil



Quelle: Der neue Maybach, ATZ/MTZ Sonderheft, S. 125, 2002.

Steuergeräte im Automobil

Ein modernes Auto ist ein
heterogenes, verteiltes, eingebettetes System
auf Rädern **mit Dutzenden von Steuergeräten**.

Steuergeräte...

- kontrollieren einzelne Fahrzeugkomponenten
- sind häufig „kleine“ 8- und 16-Bit Mikrocontroller

Probleme:

- Gewicht, Energieverbrauch und Kosten
- Ersatzteilversorgung über die Lebenszeit eines Autos
- Unzureichende Ausnutzung von Rechenleistung
- Hohe Fehlerhäufigkeit (viele Steckkontakte und Bauteile)
- Fehlertolerante Systeme schlecht realisierbar

Lösungsansatz der KFZ-Industrie

- Integration von Software-Komponenten auf wenige, leistungsstärkere Mikrocontroller
  - AUTOSAR
    (AUTomotive Open System Architecture)

- Isolation durch Betriebssystem-Mechanismen
  - Einfacher hardwareunterstützter Speicherschutz
  - Zeitliche Isolation durch echtzeitfähiges Scheduling

- Probleme:
  - Anpassung/Neuentwicklung vorhandener Software notwendig
  - Geändertes Zeitverhalten
  - Energiesparmechnismen ganz anders
  - Kosten durch neue HW/SW und Anpassungen

Unser Lösungsansatz: CoaCh

- Nachbildung kompletter Steuergeräte-netzwerke („**Network-on-Chip**")
  - CPU, Speicher, Netzwerk, Peripherie
  - Einsatz **rekonfigurierbarer Hardware**
- **Isolation** durch
  - Komplett getrennte und parallel arbeitende ***Soft Cores***
- **Integration** durch
  - Vernetzung mehrerer Systeme auf einem Chip
  - Nachbildung konventioneller KFZ-Bussysteme
    - CAN- und LIN-Bus
  - Anbindung an die „Außenwelt“
    - Schnittstellen zu realen physikalischen Geräten und Bussen

Unser Lösungsansatz: CoaCh

- Nachbildung kompletter Steuergeräte-
  ne
  - C
  - E
- Is
  - K
    p
- In
  - V
  - N
    -
  - A
    - Schnittstellen zu realen physikalischen Geräten und Bussen

Vorteile des CoaCh-Ansatzes

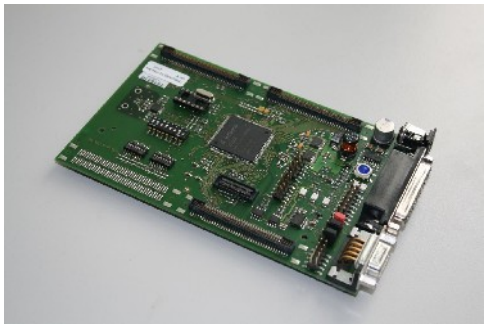
- Gewicht, Energieverbrauch und Kosten reduziert ✓
- Ersatzteilversorgung dauerhaft gewährleistet ✓
- Geringe Fehlerhäufigkeit ✓
  (weniger Steckkontakte und Bauteile)
- Fehlertolerante Systeme realisierbar ✓

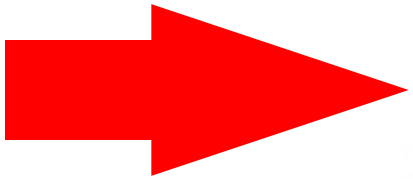
Darüber hinaus ...

- Software kann unverändert übernommen werden
- Besseres Zeitverhalten durch Parallelisierung
- Beschleunigung durch angepasste Hardwarestrukturen
- ...

Idee der Projektgruppe

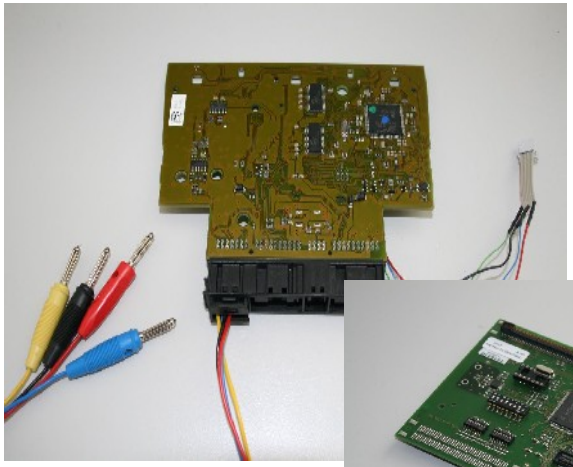
Entwicklung von eingebetteten **Systems-on-Chip** für den Automobilbereich

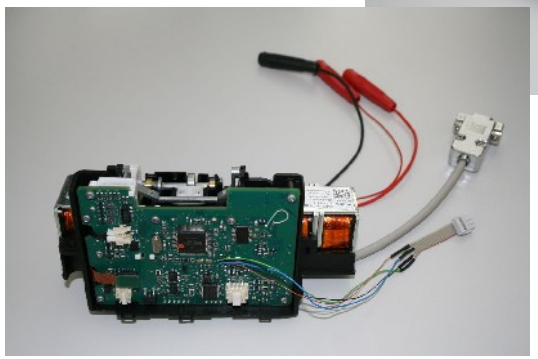


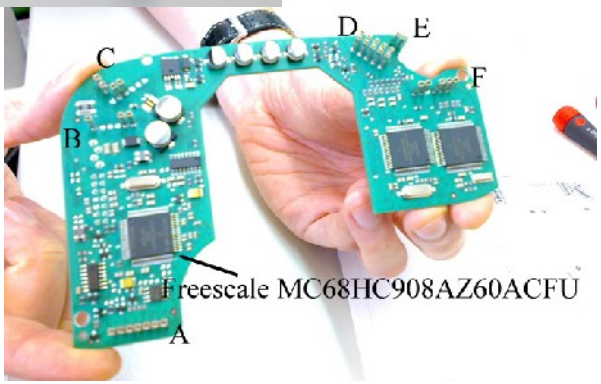


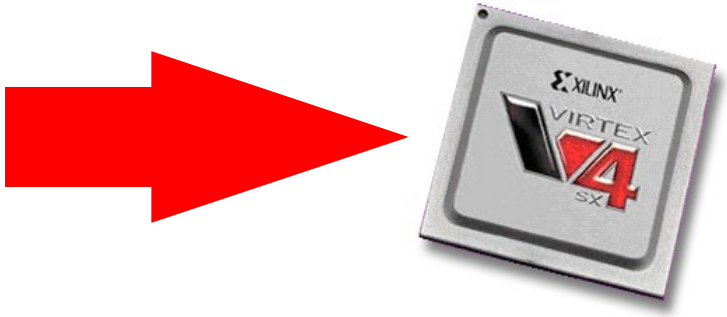
Idee der Projektgruppe

Integration heterogener Systeme auf einem FPGA als **Network-on-Chip**

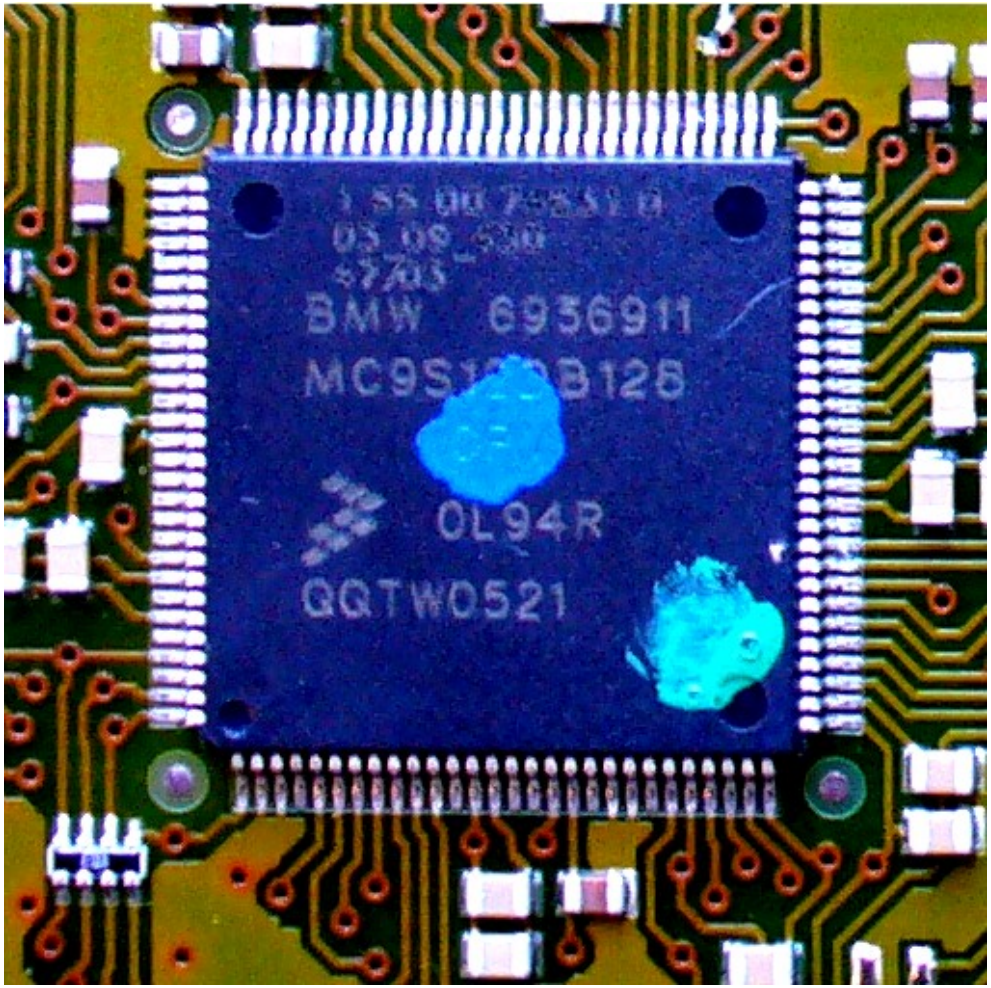






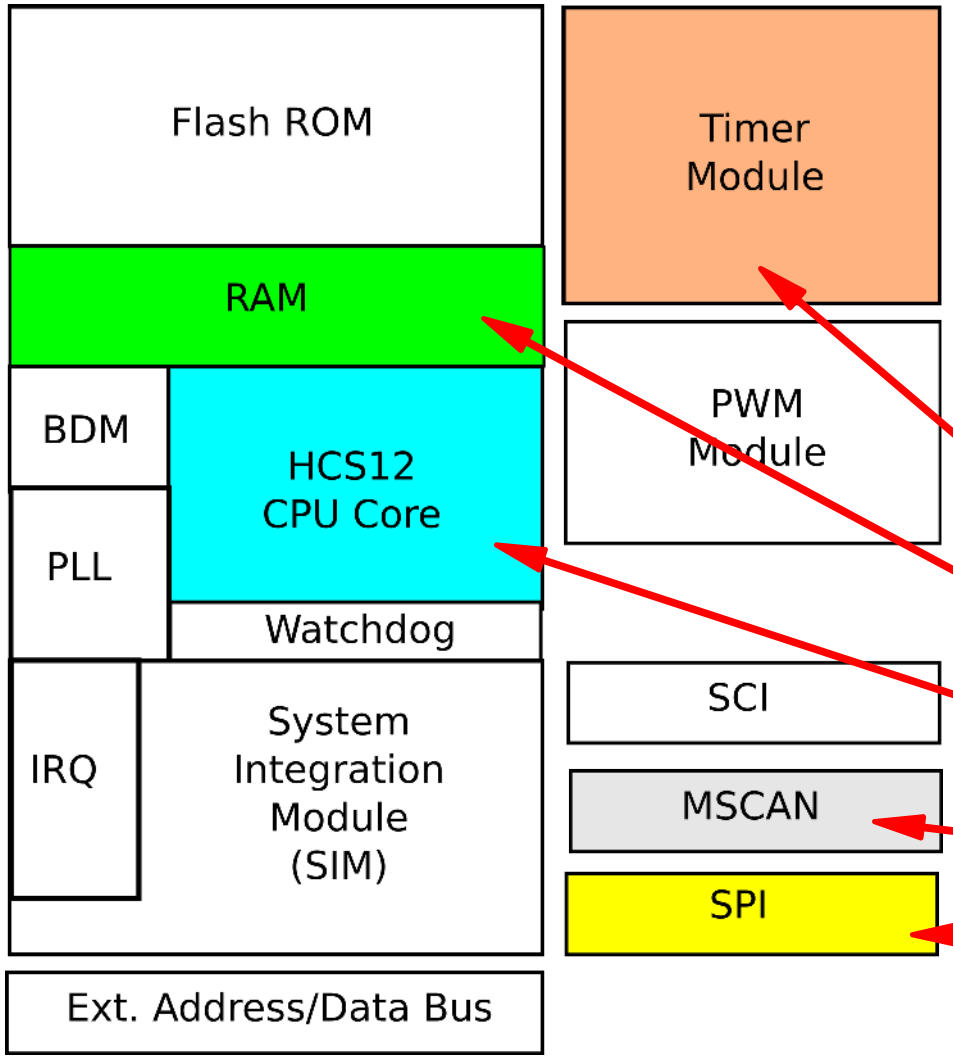


Ziele der Projektgruppe (Semester 1)



- Einarbeitung in existierende Steuergeräte-Architekturen

Ziele der Projektgruppe (Semester 1)



- Einarbeitung in existierende Steuergeräte-Architekturen
- Bau eines **System-on-Chip** mit Open Source Hardware-Beschreibungen

Timer

Memory Controller

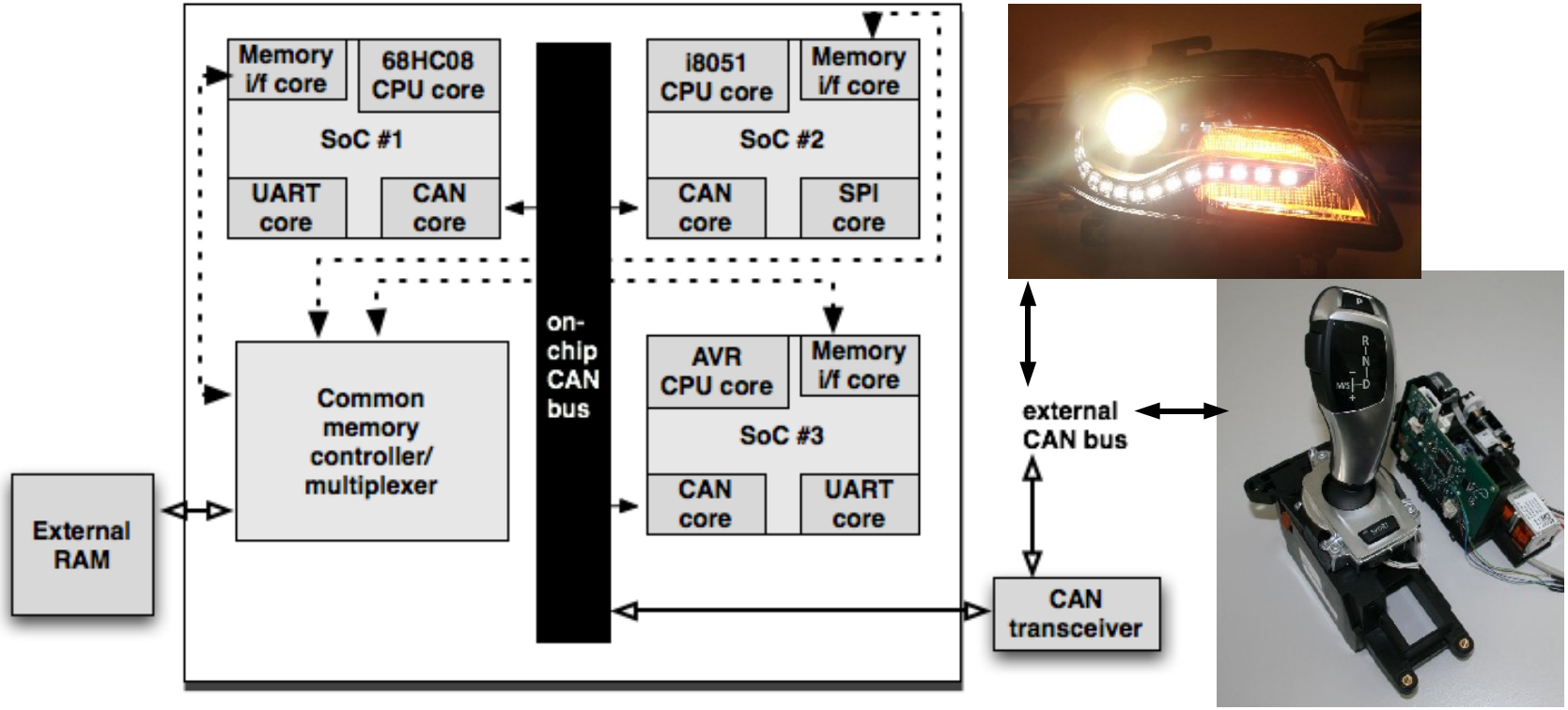
68HC11 CPU

CAN-Core

SPI-Core

Ziele der Projektgruppe (Semester 2)

Vernetzung der verschiedenen entwickelten Systems-on-Chip zu einem **Network-on-Chip** sowie mit physikalischen Geräten (Scheinwerfer, Gangwahlschalter, ...) mit emulierten Bustechnologien aus dem Automobilbereich (CAN, LIN)



Die Highlights

- komplexe Aufgabe im Team lösen

+ aktuelle Standards der KFZ-Industrie
+ Bauen von Systems on Chip auf FPGAs
+ Vernetzung auf Chip-Ebene
+ Industriekontakte
+ Messebesuch
+ praktische Arbeiten
+ unterschiedlichste Aufgabenbereiche
  - Hardware, Systemsoftware, Peripherie, Kommunikation, Test, ...

unterstützt von ...

KOSTAL

Audi

infineon

Teilnahmevoraussetzungen

- **notwendig**
  - VL *Rechnerarchitektur*, *Prozessrechnertechnik* oder *Eingebettete Systeme*
  - Bereitschaft zu hardware- und systemnaher Programmierung
    - keine Angst vor Hardware!
  - Verständnis englischsprachiger Artikel und Handbücher
- **wünschenswert**
  - Kenntnisse in VHDL und/oder Verilog
  - Programmiererfahrung in C und/oder C++

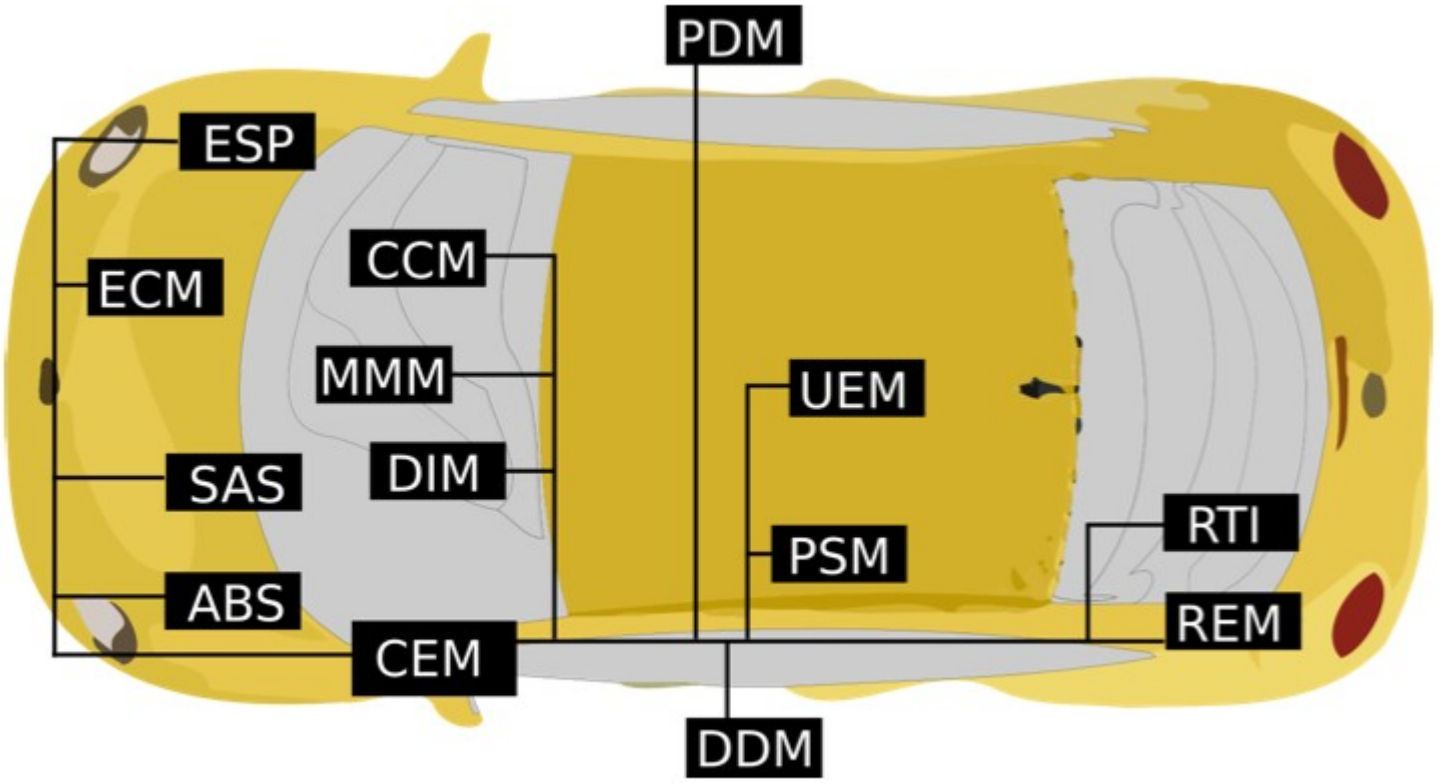
Agenda

- Einführung
- **Vision und Minimalziele**
- Durchführung
- Seminarthemen
- Nächste Schritte

- **Fragen**

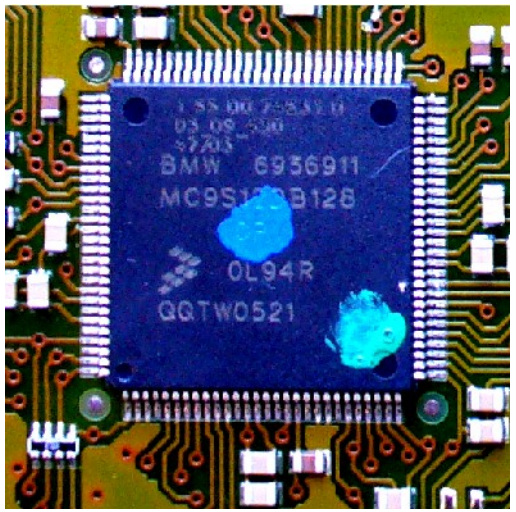
Ist-Zustand

Aktuelle Struktur eines KFZ mit vernetzten Steuergeräten



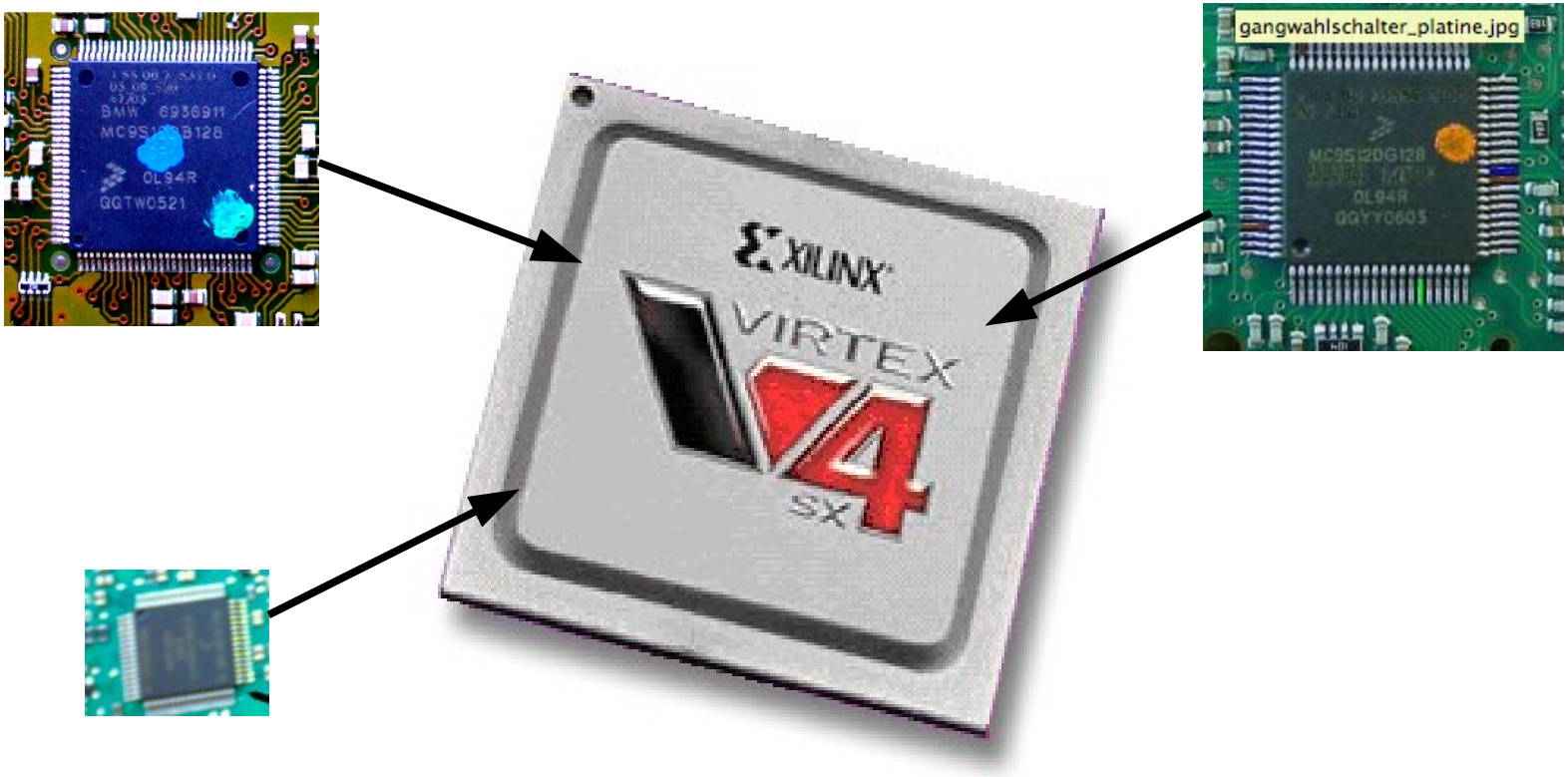
Struktur von Steuergeräten

- Steuergeräte erfüllen meist **eng begrenzte Aufgaben**
  - Vergleichsweise geringe Rechenleistung erforderlich
  - Kleine Programme, einfache Betriebssysteme
- Einsatz von **CPU-Architekturen** der **70er/80er Jahre**
  - Motorola/Freescale 68XX → HC08, 9S12
  - Intel 8051
- Einfache **neue CPU-Architekturen**
  - Atmel AVR
  - Texas Instruments TMS430
- Viel **Peripherie** auf dem Chip
  - Seriell (RS232, SPI)
  - „einfache“ Port-Pins: digitale I/O
  - Bus-Schnittstellen: CAN, LIN
- Einfache Peripherie off-Chip
  - Watchdogs, Leitungstreiber



Vision

Mehrere Steuergeräte sind auf einem FPGA als „Soft Cores“ in einer Hardware-Beschreibungssprache integriert



Hardware-Beschreibungssprachen

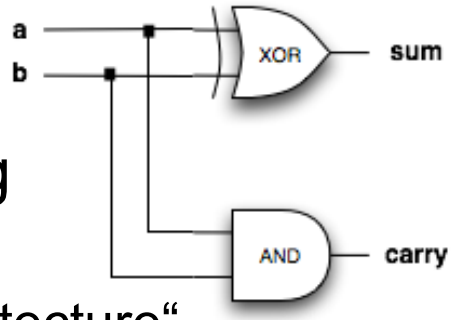
- Hardware-Entwurf nicht mehr „traditionell“
- Beschreibung des Verhaltens in **domänenspezifischer Sprache** (HDL, Hardware Description Language)
  - VHDL, Verilog
  - Aktueller Trend: SystemC
- **Komponentenbasierter** Entwurf
  - Wiederverwendung von Beschreibungen
- Trend: **Open Source Hardware**
  - Analog zu Open Source Software
  - http://www.opencores.org u.v.a.m.
  - CPU-, Peripherie-, Bus-, Beschleunigerkomponenten

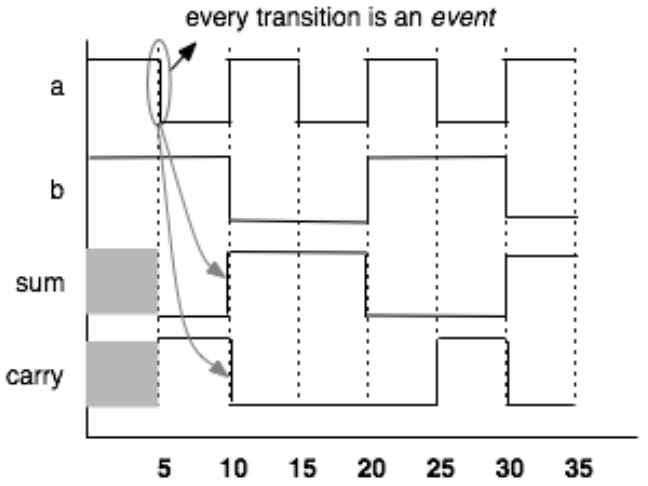
Open Source Hardware-Komponenten

- Kein Entwurf komplexer neuer CPU- und Peripherie-Soft Cores
- Im Netz als **Open Source** verfügbar:
  - CPUs: 68HC08, AVR, T51 (8051), S430 (TMS430)
  - Peripherie: UART (RS232), SPI
  - Bus-Systeme: CAN-Bus, I2C
- **Anpassung** der vorhandenen Cores
  - Erweiterung um nicht vorhandene Befehle
  - Anpassung der Registerstrukturen/Schnittstellen der Peripherie-Cores
  - Entwicklung einfacher Cores (z.B. LIN)

VHDL-Beispiel: Halbaddierer

- Addiere zwei Bits a,b
- Erzeuge Summe und Carry
- VHDL-Beschreibung
  - Schnittstelle: „entity“
  - Implementation: „architecture“






```
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;
```

Prozesse in VHDL

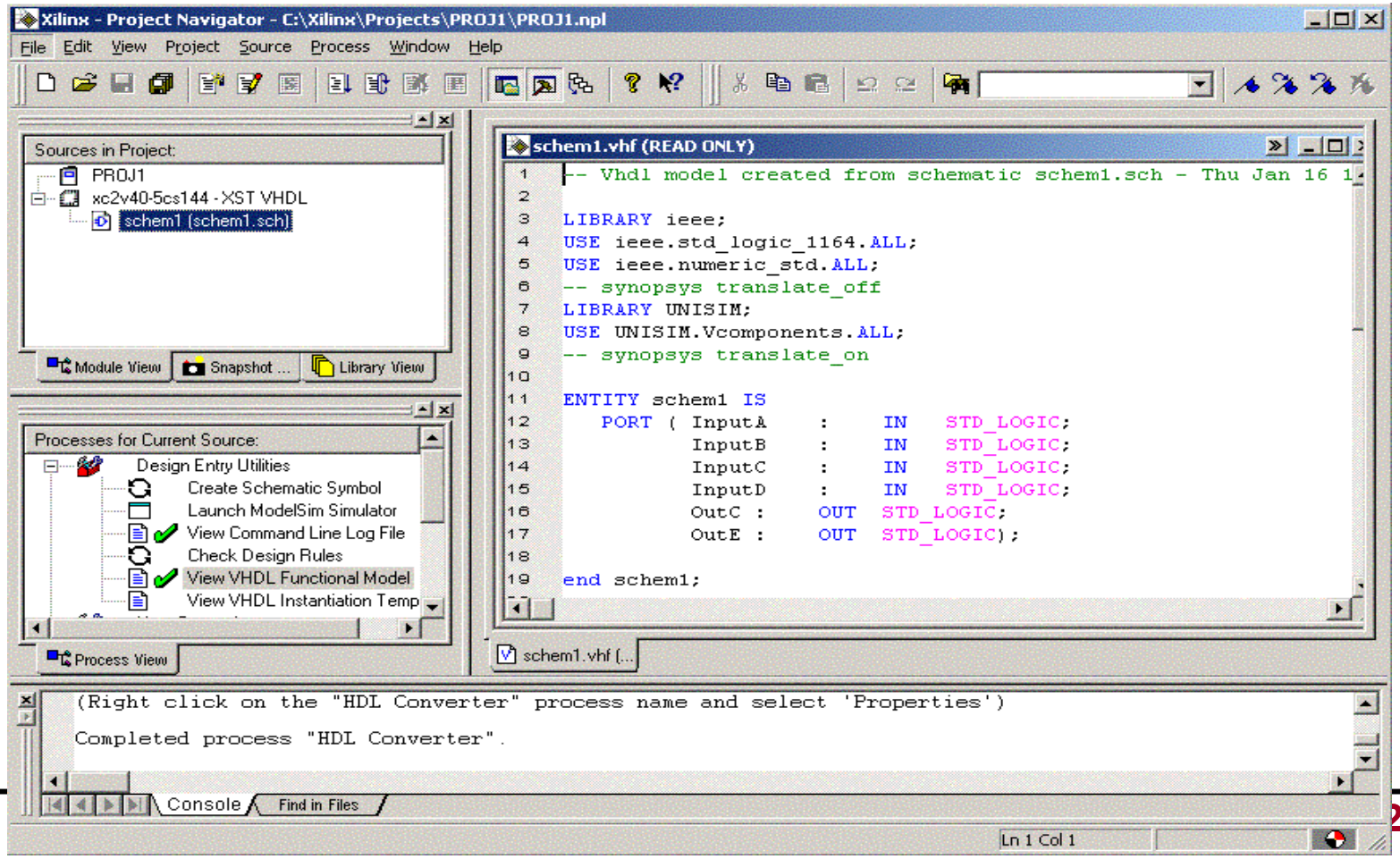
- Parallele Ausführung *unterschiedlicher* Prozesse
- Sequentielle Ausführung *innerhalb* jedes Prozesses

```
architecture behavior of half_adder is
  begin
    sum_proc: process(a, b) begin
      if (a = b) then
         sum <= '0' after 5 ns;
      else
         sum <= (a or b) after 5 ns;
      end if
    end process;
    carry_proc: process(a, b) begin
      case a is
        when '0' => carry <= a after 5 ns;
        when '1' => carry <= b after 5 ns;
        when others => carry <= 'X' after 5 ns;
      end case;
    end process;
end behavior;
```

sensitivity lists

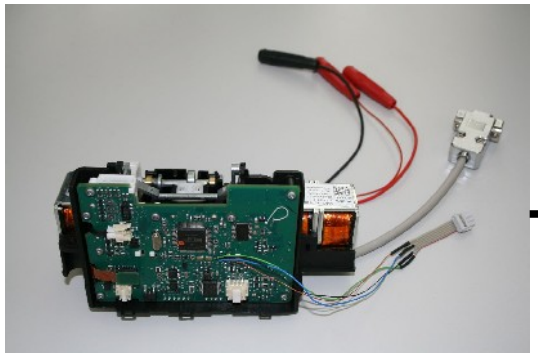
HDL-Entwicklung

- Proprietäre Entwicklungswerkzeuge erforderlich
  - Xilinx: WebPack ISE (kostenlos für Linux und Windows)
- Editor, Compiler für VHDL/Verilog, Placement, ***Simulator***



Vision

Integriertes System: Auf FPGA realisierte Steuergeräte integriert mit physikalischen Komponenten via Standard-Bussystemen (CAN, LIN)

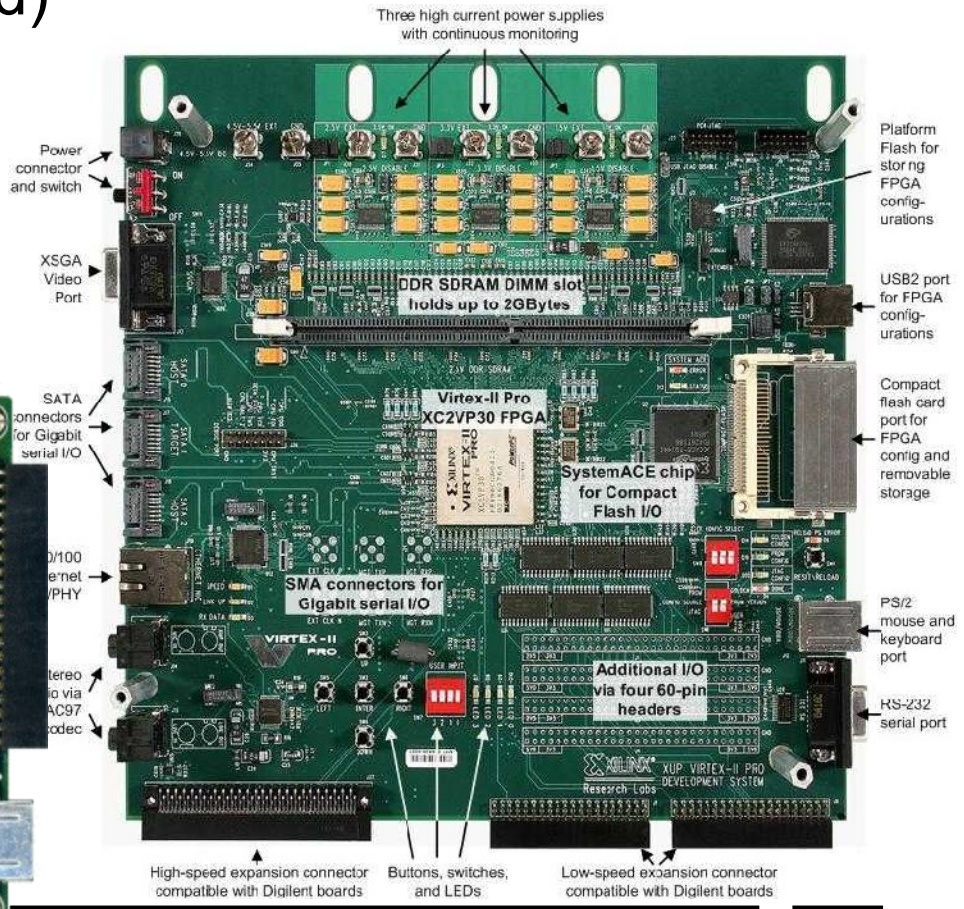


Entwicklungsplattform

FPGA-Entwicklungsboards von Xilinx/Digilent

- Virtex IIpro FPGA (High-End)
- Spartan 3 FPGA (Low-End)

- On-Board-Schnittstellen
  - Port Pins,RS232
  - Ethernet, Audio, PS/2...



Passt das denn wirklich?!?

Ja! Aktuelle FPGAs haben hohe Kapazitäten...

	T51	AVR	68HC08	CAN (DMA)
Spartan 3S400				
Size (LUTs)	1057	978	877	943
Size (Percent)	29	26	24	25
Speed (MHz)	40	31	34	—
Virtex IIpro 2VP30				
Size (LUTs)	1109	989	901	966
Size (Percent)	8	7	7	7
Speed (MHz)	62	48	50	—

Minimalziele

Keine Panik!

Minimalziele

- bis hier: Vision
  - was aus unserem Projekt werden *könnte*
  - sprengt vielleicht den Rahmen
- jetzt: Minimalziele:
  - müssen definiert werden (Scheinvergabe)
  - sind realistisch
  - höchstwahrscheinlich kommen wir zu mehr...

Minimalziele

- **Entwurf** von mind. zwei minimalen 8-Bit Steuergeräten
  - CPU, minimale Peripherie, CAN-Anbindung
  - Aufbau in kooperierenden Teams, die jeweils an einem System-on-Chip arbeiten
- **Simulation** in Xilinx ISE
- **Implementierung** auf jeweils eigenem FPGA
- **Test** mit real existierender Software

- **Integration** der entwickelten SoCs auf einem FPGA
- **Vernetzung** der verschiedenen SoCs mit CAN
- **Anbindung** des CAN-Bus im SoC an die Außenwelt
- **Test** mit realen physikalischen Komponenten

Optionale Ziele

Wenn uns langweilig werden sollte...

- Umgang mit **externem Speicher**
- Detaillierte **Analyse** der integrierten SoCs
  - Monitoring-Komponenten für SoCs und Bus
    - Xilinx ChipScope, Visualisierung der Vorgänge im System
- **Energieverwaltung** auf FPGAs
- Weitere ...
  - SoCs entwerfen mit anderen CPUs
  - Bus-Systeme, z.B. LIN

Agenda

- Einführung
- Vision und Minimalziele
- **Durchführung**
- Seminarthemen
- Nächste Schritte

- **Fragen**

Zeitplanung WS 2008/2009

- **Wintersemester**
  - Seminar
  - Entwürfe
  - Implementierung verschiedener SoCs
- **Sommersemester**
  - Integration der SoCs
  - Anbindung an physikalische Hardware
  - Test

Zeitplanung detailliert

Woche	Inhalt	Arbeitsform
1-2	Einstieg in Literatur, Werkzeuge und Vorarbeiten	gemeinsames Seminar
3	Ideensammlung, Festlegung von Arbeitsaufträgen	Gruppen-/Plenarsitzungen
4-8	Entwürfe für die und Implementierung der einzelnen Systems-on-Chip (Minimalziel 1)	Gruppenarbeit
9-10	Bewertung der Implementierungen, Einarbeitung in die Software-Entwicklungsumgebung	Gruppen-/Plenarsitzungen
11-13	Erstellung einer Beispielanwendung für das jeweilige System-on-Chip (Minimalziel 2)	Gruppenarbeit
14-15	Überprüfung der bisherigen Ergebnisse, Zwischenbericht (Minimalziel 5)	Gruppen-/Plenarsitzungen

Woche	Inhalt	Arbeitsform
1	Rekapitulation des Wintersemesters, ggf. Anpassung des Entwurfs für die Systems-on-Chip und Festlegung von Arbeitsaufträgen für deren Implementierung	gemeinsames Seminar, Gruppen-/Plenarsitzungen
2-5	Integration der SoCs in ein Network-on-Chip: Ermittlung von Integrationsproblemen, Entwicklung von Konzepten für die Buskomponente und weitere Integrationskomponenten (Minimalziel 3)	Gruppenarbeit
6	Bewertung/Korrektur des Network-on-Chip-Entwurfs, Aufteilung von Arbeitsaufträgen	Gruppen-/Plenarsitzungen
7-11	Implementierung des Network-on-Chip (Minimalziel 4)	Gruppenarbeit
12-13	Test und Bewertung des Gesamtsystems	Gruppenarbeit
14-15	Abschlussbericht (Minimalziel 5)	Gruppen-/Plenarsitzungen

Dokumentation

- Ausarbeitung der Seminarthemen
- Kurzprotokolle in Gruppen- und Plenarsitzungen
- Entwurfsdokumente
- Zwischen- und Endbericht

... ganz schön viel Papier! ☹

Dokumentation

- Ausarbeitung der Seminarthemen
- Kurzprotokolle in Gruppen- und Plenarsitzungen
- Entwurfsdokumente
- Zwischen- und Endbericht

... ganz schön viel Papier! ☹

- Reduzierung auf das Nötigste
- kontinuierlich wachsende Berichte

Infrastruktur

- eigener **Laborraum** (OH-16 E06)
- Hard- und Software
  - **FPGA-Boards**
  - **Steuergeräte** und KFZ-Komponenten
  - **Entwicklungswerkzeuge** (Compiler, Debugger)
  - etc.
- **Kooperationswerkzeuge**
  - Mailingliste
  - Wiki
  - Versionsverwaltung (Subversion o.ä.)
  - Projektmanagement-Werkzeuge

Agenda

- Einführung
- Vision und Minimalziele
- Durchführung
- **Seminarthemen**
- **Nächste Schritte**
- **Fragen**

Seminarthemen

- konzeptionelle **Grundlagen**
  - Mikroprozessortechnik-Grundlagen
  - Systems-on-Chip, Networks-on-Chip
- grundlegende **Werkzeuge** und **Sprachen**
  - Xilinx ISE WebPack
  - VHDL und Verilog
- typische **Hardware**
  - Steuergeräteplattformen, Mikrocontroller-Architekturen
  - CAN und LIN-Bus
- typische **Entwicklungsmethoden**
  - Entwicklungswerkzeuge
  - Test und Diagnose
- **Forschungsthemen**
  - Dynamische Rekonfigurierung

Nächste Schritte

1. bis **14.5. 12:00**: Online Anmeldung!
2. optional: Mail bzgl. bevorzugter Mitstreiter an uns
3. eventuell Rückfragen von uns per Mail
4. Auswahlverfahren
5. ab 28.5.: Aushang der Listen
6. ab Juli: Erste Sitzung der Gruppe
   - Verteilung der Seminarthemen

Fragen?

Seeing is believing ;-)

Demonstration:

- FPGA-Board mit Spartan 3S400 FPGA
  - 1 MB RAM on-board
  - PS/2-Interface
  - VGA-Schnittstelle
- Und darauf... ein System-on-Chip
  - Apple][– kennt den noch wer? ;-)
  - 8-Bit-CPU 6502, Tastatur, Video, Sound...
- Komplexität...
  - ähnlich zu einem Steuergerät im Auto :-)

Aufgabenkomplexe: Wintersemester

→ **natürliche Grobeinteilung in Arbeitspakete**

- Paralleles Arbeiten an verschiedenen SoCs
  - Analyse existierender Steuergeräte
  - Selektion „passender“ Open Source-Hardware-Komponenten
  - Adaption der Komponenten auf Erfordernisse des zu emulierenden Steuergerätes
  - Simulation, Implementierung und Test
  - Je SoC ein Team aus 3-4 Studierenden

Aufgabenkomplexe: Sommersemester

- Integration der verschiedenen SoCs
  - Implementierung der gemeinsamen Bus-Komponente
  - Integration der einzelnen VHDL/Verilog-Komponenten in eine komplexe Beschreibung
  - Simulation und Test: FPGA alleine
  - Anbindung an externen CAN-Bus
    - Transceiver-Chip für „echten“ CAN-Bus anbinden
  - Test des Gesamtsystems
  - Ansteuerung phys. Geräte (Scheinwerfer etc.) via CAN-Bus von FPGA-basierten Steuergeräten
  - Aufbau eines Demonstrators für ein Gesamtsystem