ESL Design in the context of embedded systems education

Mladen Berekovic

Technische Universität Braunschweig

WESE 2009

Overview

- introduction – electronic system level design today
- new software challenges in microelectronic systems
- example: automotive electronics
- consequences for ESL design education
- curriculum in computer engineering & embedded system platforms
- conclusion

Introduction - ESL Design Today

- ESL (*Electronic System Level*) design is typically understood as microelectronic design raised to higher levels of abstraction
  - start with a coherent system function specification
    - specification might be adapted during the design process
  - validate system function
  - implement in HW and SW components
  - integrate components and test system
    - larger systems: V-Model
- design goal
  - correct implementation of function specification
  - system optimization
    - objectives: cost, power, …
    - constraints: physical, reliability, …

Introduction – Designer Qualification

- required skills
  - hardware architectures
  - hardware system design incl. partitioning, accelerators, …
  - system modeling and verification technology – TLM, SystemC, …
  - hardware design processes
- helpful knowledge
  - application domain knowledge, e.g. signal processing, control,
  - compiler technology
  - fundamentals of basic software
    - e.g. exception handling, drivers, …

Software in Microelectronic Systems

- a growing number of microelectronic circuits are not designed for a single final product
  - no coherent initial specification
    - a large part of the final system specification is delayed to a later development process, including upgrades/updates
    - software is used for end product diversification
  - software is more than interfaces and drivers, but controls hardware system function and adaptation
  - software has developed into an integral part of microelectronic system design
  - ® Multiprocessor Platforms (MpSoC)
- examples:
  - wireless, consumer, automotive, industrial, …

MpSoCs

Software in ESL Design

- software development follows different rules and processes
  - comes with its own layered architecture
  - follows different design objectives of
    - flexibility
    - systems evolution
    - dynamic adaptation
  - uses different organization
    - resource sharing strategies (processors, buses, memories, …)
    - component protocols and signaling

« system design effort has to a large part moved to software development

  - just count number of people involved

Software Efficiency and Hardware Organization

- software efficiency very much depends on the underlying hardware and its organization
  - so, embedded software developers should (finally) be educated in hardware organization?
  - yes, but that is only one side

Software Depends on Hardware - Consequences

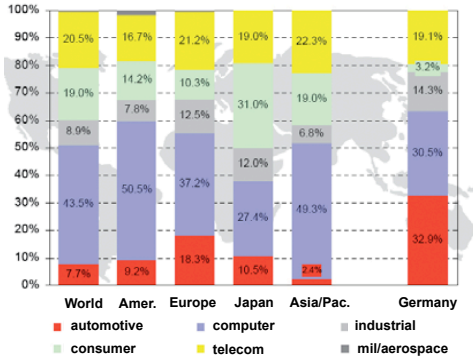
- hardware architectures that effectively support emerging software architectures have become an (inherent) new design objective
- software architecture support influences
  - platform (re-)usability
  - design space for diversification
  - quality of the final hardware-software product

® ESL hardware designer should have a basic understanding of software to support system design

How important is all this for systems engineering?

- embedded systems and their software have become the dominant driver for European microelectronics and systems engineering
- can be seen in market and cost shares

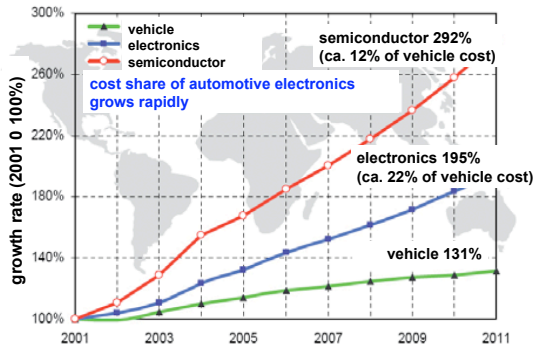
Microelectronics – Market in 2006



embedded systems components clearly dominate computer components in Europe

source: German ZVEI 2007

Growth Rates in Automotive Systems Cost



source: German ZVEI 2007

Example: Current Automotive Design Chain

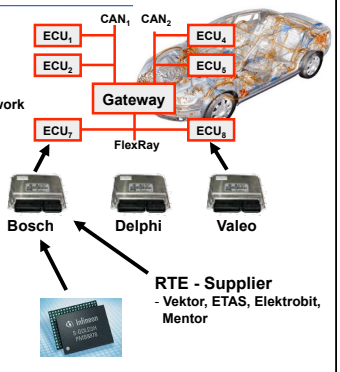
OEM
- BMW, Daimler, GM, PSA, Toyota, …
- global system, integration and network

specs ↓ ↑ ECUs

ECU - Supplier
- Bosch, Delphi, Valeo, …
- ECU responsibility

specs ↓ ↑ SoCs

HW Component - Supplier
- Infineon, Freescale, ST, Toshiba, …



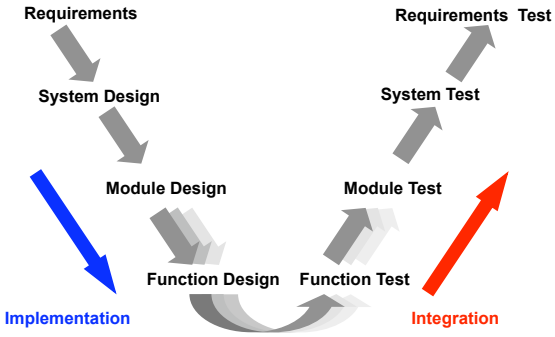
RTE - Supplier
- Vektor, ETAS, Elektrobit, Mentor

„Traditional“ ECU Design

- OEM defines, feature set, network, and specifies ECU function (using text, MatLab and other models, …), defines diagnosis and test procedures
- supplier develops and integrates ECU HW components, RTE and application function provides ECU
- HW component supplier designs IC specifically for the needs of the ECU function which is defined by the OEM specification
  - ® single source „coherent“ specification
- supplier integrates and tests ECU and local RTE
- OEM integrates and tests ECUs, networks and final car

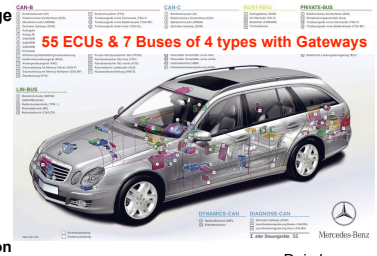
® V-Model

V Model



Automotive Design - Result

- complexity challenge
  - hundreds of functions
  - 50+ ECUs
  - networked control
  - many suppliers
  - heterogeneous
- design challenges
  - supply chains
  - systems integration
  - verification



source: Daimler

- software architecture standards: single ECU => network

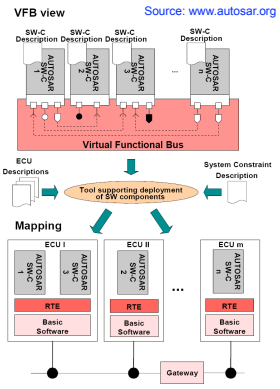
OSEK => AUTOSAR

AUTOSAR Standard Goals (selection)

- Modularity
  - tailor the SW-components according to the individual requirements
- Scalability
  - adaptability of SW-components to different vehicle platforms
  - avoid proliferation of software with similar functionality
- Transferability
  - remapping of SW-components among different HW-components
- Re-usability
  - adaptability of SW-components across different product lines
  - shorten design process
  - improve quality and reliability of E/E systems

AUTOSAR Methodology

- SW-Components (SW-C)
  - encapsulate the applications
- Virtual Functional Bus (VFB)
  - communication mechanisms
  - interface to Basic SW
- Mapping
  - configuration and generation of RTE and Basic SW
- Runtime Environment (RTE)
  - VFB implementation on a specific ECU
- Basic Software (BSW)
  - infrastructural functionality on an ECU



AUTOSAR – Consequences for ESL Design (1)

- automotive systems become software platforms
  - no complete ECU function specification at application level
  - partially defined and evolving system functionality
  - mapping of software to platform remains open
  - abstract requirements to robustness and scalability
- software is used for end product diversification
  - new types of resilient multicore architectures will become interesting
  - software only partially accessible to the hardware designer
    - IP protection
    - later upgrades must be planed in advance
    - ® ESL design process has to adapt

AUTOSAR – Consequences for ESL Design (2)

- so, why not leave software design to software people as in general purpose processor design?
  - benchmarking does not cover networked system functionality
  - embedded systems require performance guarantees
  - such guarantees require appropriate interplay of HW, RTE and application

« difference to general purpose processor design

AUTOSAR – Consequences for ESL Design (3)

- hardware has significant influence on predictability, performance and correct concurrent task execution
  - memory architectures, caches, multithreading, synchronization, power control, network topologies and protocols, …
- AUTOSAR is just a highly visible example for general trends in embedded systems design

Consequences for ESL Education (1)

- ESL design requires background in RTE and software platform architectures
  - resource organization principles
    - scheduling, arbitration, task activation and execution, QoS control
  - embedded operating system principles
    - memory assignment, communication and synchronization
  - software performance evaluation techniques
    - measurement, instrumentation, monitoring, WCET timing, …

Consequences for ESL education (2)

- ESL design requires background in application models
  - examples: Simulink, FSMs, signal flow graphs, task graphs, …
  - application models determine
    - task activation frequencies (timing or event driven)
    - task dependencies - causality
    - communication frequency and volume
    - buffer memory usage …
    - crucial to identify system load and application timing
  - explicit description lost in implementation language
    - examples: SystemC, VHDL, C, …
    - knowledge of implementation languages not sufficient

Consequences for ESL education (3)

- ESL design profits from software methodology background
  - modeling, verification, and analysis tools, synthesis (compiler and code generation)
  - software test strategies

What Software Skills are not in the ESL Focus ?

- general software engineering and programming skills
- software languages and standards with little influence on platform execution

® large part of software technology

Embedded Systems Education – Conflicting Goals

- education in CS tries to trade-off the growing importance of embedded systems software versus „classical“ CS topics
  - result is an emphasis on the overlapping segments of software engineering and software modeling technologies
  - emphasis off topics wrt. ESL design
- education in microelectronic design does not yet cover software topics
  - major differences in classical and embedded computing
  - software for MpSoC is an emerging field asking for course contents to be developed
  - software contents compete for shares in current microelectronic education
- compromise needed
  - tried to address in curriculum at TU Braunschweig

Computer & Communication Systems Engineering

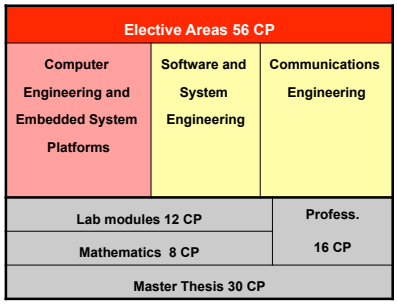
- curriculum CCSE introduced in fall 99 at Technische Universität Braunschweig (concurrently with TU Dresden)
  - German title “Informations-Systemtechnik”
- covers embedded system design - not restricted to MpSoC
- originally diploma, now BSc/MSc degrees
- cooperation of departments of CS and EE&IT
- 4 balanced parts in BSc and MSc program: EE, CS, mathematics, practical labs
- example for embedding ESL software education in embedded systems curriculum

BSc Mandatory Courses (selection)

• Hardware fundamentals (40 CP)		
– EE Fundamentals	9 CP	*classical CE*
– Circuits&Systems	13 CP	
– Computer Engineering	14 CP	
– Metrology	4 CP	
• Software courses (30 CP)		
– Algorithms & Data Structures	8 CP	
– Programming	10 CP	*general software technology*
– Software Engineering	4 CP	
– Operating Systems	4 CP	
– CS theory	4 CP	
• Systems engineering (16 CP)		
– Computer Networks	4 CP	*provides application foundation*
– Communications Engineering	8 CP	
– Signal Processing	4 CP	

Master Curriculum

- 1 elective area as major (min. 20 CP)
- min. 1 area as minor (min. 12 CP)



Computer Engineering & Embedded System Platforms (1)

- includes introductory course to embedded system platforms as elective core course (6 CP)
  - application models and implementation
    - reactive and transformative system descriptions, automata networks, state diagrams, signal flow graphs, Kahn graphs, Petri-Nets,…
    - correct implementation, identifying design space, exploiting non determinism, …
  - embedded architectures
    - embedded system architecture principles
    - microcontroller and DSP – principles of specialization
    - example: different types of media processors (including failures)
    - complex multicore architectures: network processors, DSP, Emotion Engine (PS 2), Cell Processor (PS 3), reconfigurable architectures
  - platform organization
    - real-time platform organization, resource sharing,
    - software organization and architectures, …
    - real-time analysis and predictability
    - power control and optimization

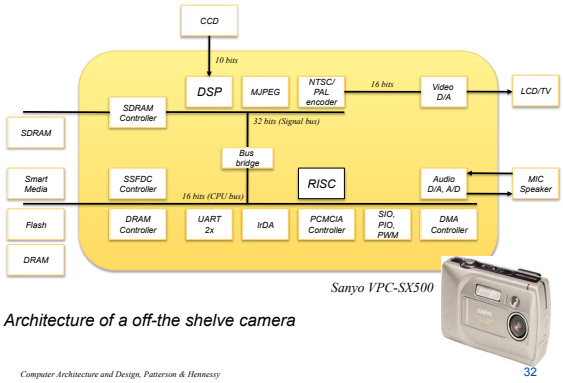
Computer Engineering & Embedded System Platforms (2)

- several in depth elective courses on (selection)
  - dependable computing
  - advanced computer architecture
  - advanced VLSI design
  - analog circuit design
  - avionics platforms
  - automotive electronics
  - system level EDA
  - …

Example Course: MPSoC Design

- **Module for ESL-VLSI Design: 6CPs**
- **Lecture MPSoC Design**
  - Electronic-System-Level Design Methodology, SystemC
  - Transaction-Level Modeling (TLM 2.0)
  - Networks, Busses & Protocols, Networks-on-Chip
  - Low-Power MPSoC System-Level Design Techniques
- **Accompanying Lab Course MPSoC Design**
  - SystemC based Design of a project: Digital Camera
  - Introduction to System-C based System Design
  - Coware (Multi-Core) Platform Creator
  - Component Design in System-C
  - Abstraction Models
  - Busses & Communication
  - Accelerator Design

VLSI-MPSoC Design Lab: Digital Camera



Architecture of a off-the shelve camera

Curriculum Summary

- **BSc provides broad insights in HW, SW, and application areas avoiding early bias**
  - emphasis on general software technology – will be the focus of most BSc embedded software developers
- **MSc includes ESL software in the context of microelectronics, application and in-depth software engineering**
  - concentrates special ESL software aspects in a single ECC (6 CP)
  - supports combination with in-depth microelectronic and embedded computer architecture courses
  - requires further in-depth education in application and systems engineering areas stressing the interdisciplinary role of embedded system platform design
- **specialization is taken by 30-40 students each year**
  - student evaluation very positive but introductory MSc course considered time consuming due to insufficient textbook material
  - very positive feedback from industry

Conclusion

- **trend towards software controlled MpSoC leads to separated hardware design and software configuration and update processes**
- **software architectures are moving fast and start to dominate the overall embedded system design**
- **software architectures impose new challenges that affect hardware design**
- **now, the hardware architect is not only working for the application developer, but also for the software architect**
- **with multicore and networked architectures, good hardware design is more than ever needed to reach system efficiency**
- **it is about time to include selected software topics in the curriculum**