EE382V: Embedded System Design and Modeling

Lecture 1 – Introduction

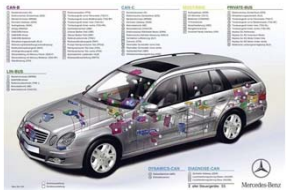
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

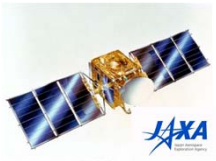
Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - System-level design
- **Course information**
  - Topics
  - Logistics
  - Projects
- **Design methodology**
  - System-level design flow
  - Models and methodologies

Embedded Systems

- **System-in-a-system**
  - Application-specific
  - Tightly constrained





- **Ubiquitous**
  - Far bigger market than general-purpose computing (PCs, servers)
    - 98% of all processors sold [Turley02, embedded.com]

- **Exponentially growing complexities**
  - Application demands & technological advances
  - ➢ Increasingly networked and programmable

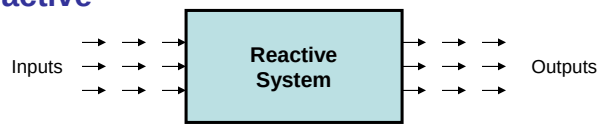
EE382V: Embedded Sys Dsgn and Modeling, Lecture 1 © 2014 A. Gerstlauer 3

Cyber-Physical Systems (CPS)

- **Not transformative**

  - Output = F(Input)
  - ➢ Procedural/batch processing

- **But reactive**



  - Continuous interaction with environment
  - ➢ Sense and act on the physical world

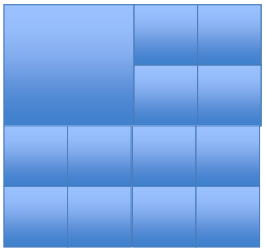
- ➢ **Concurrency and time (order)**

EE382V: Embedded Sys Dsgn and Modeling, Lecture 1 © 2014 A. Gerstlauer 4

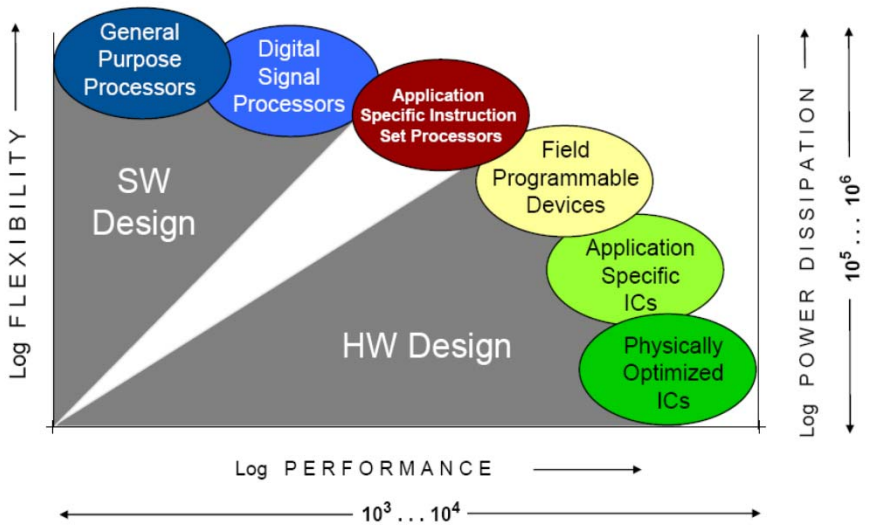
General-Purpose Computing

- **We are reaching physical limits of technology scaling**
  - Dark silicon: power/utilization/… walls
  - Mobile devices and no leap in battery technology
  - ➢ Efficiency is the new performance

- ➢ **Opportunity and need for specialization**
  - ➢ Heterogeneous multi-core / Asynchronous CMP
    - ➢ Big/little architectures
    - ➢ GP-GPUs
  - ➢ Flexibility vs. specialization
    - ➢ Right mix of cores?
    - ➢ How to "program"?

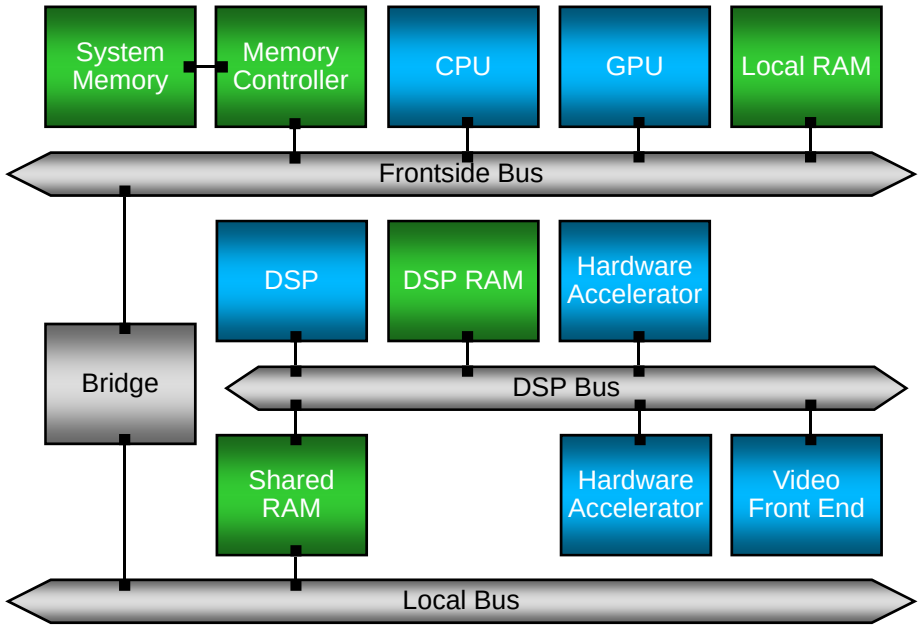


Implementation Options



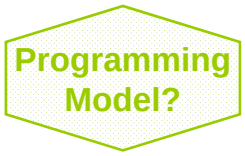
Source: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

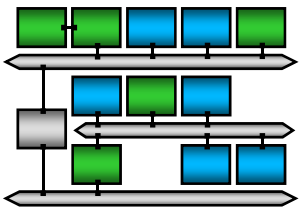
Multi-Processor System-on-Chip (MPSoC)



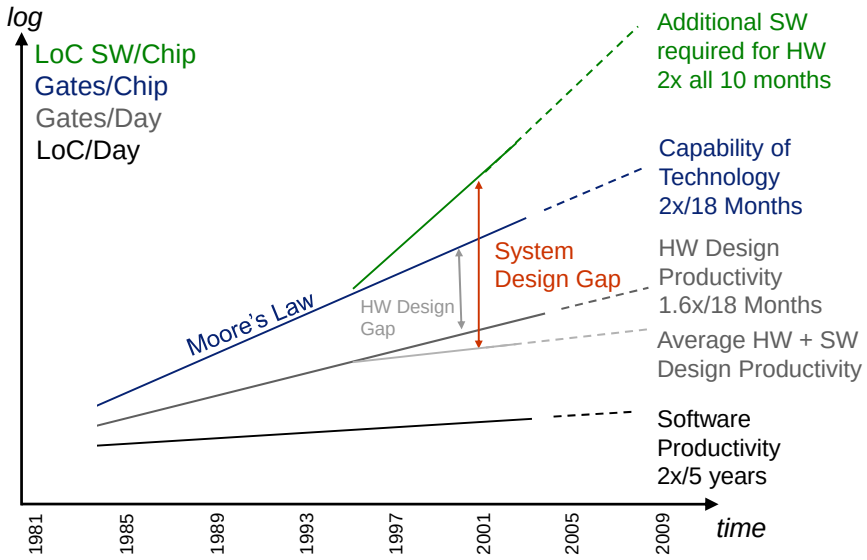
Design Challenges

- **Reactivity**
  - Concurrency & time
  - Safety, reliability, robustness
- **Heterogeneity**
  - Of components
    - Processors, memories, busses
  - Of design tasks
    - Architecture, mapping, scheduling
- ➢ **Complexity**
  - High degree of parallelism
  - High degree of design freedom
  - Multiple optimization objectives & design constraints





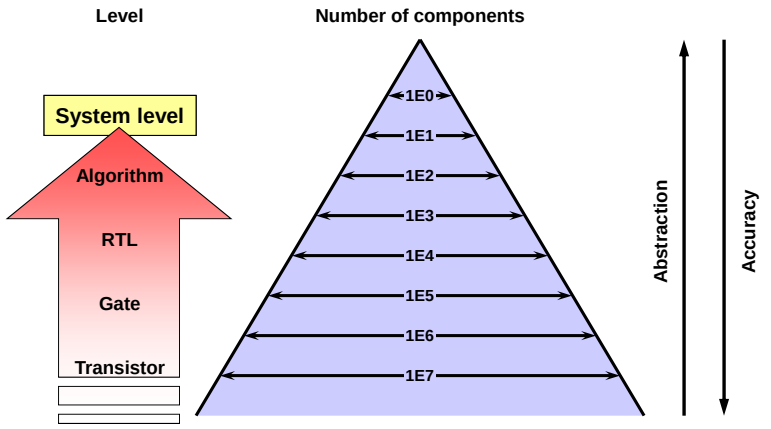
Complexity Trends



Source: W. Ecker, W. Müller, R. Dömer, *Hardware-dependent Software - Principles and Practice*, Springer 2009.

Abstraction Levels

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

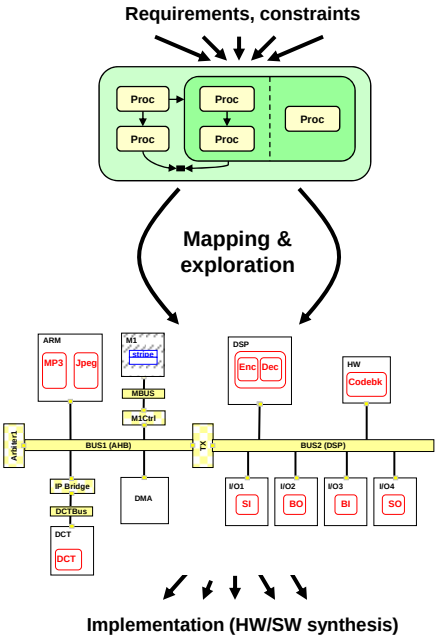


Source: R. Doemer, UC Irvine

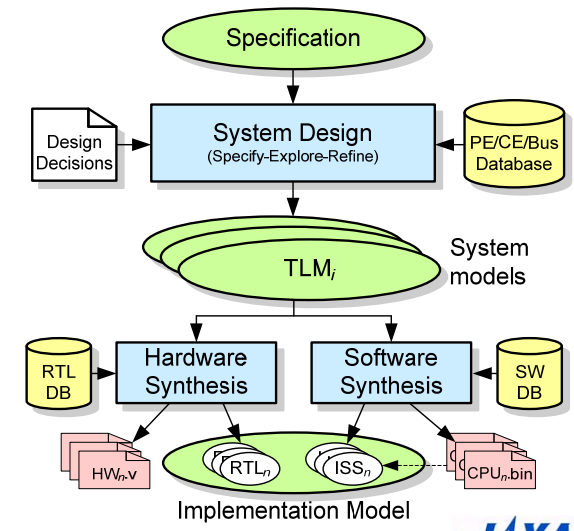
System-Level Design

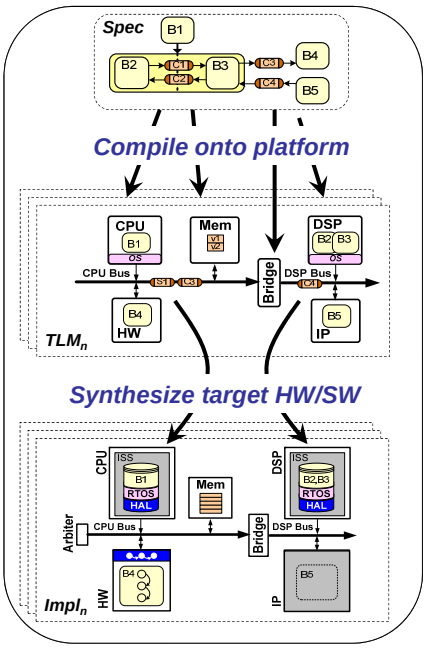
- **From specification**
  - Functionality, behavior
    - Concurrency, order
    - Constraints
- **To implementation**
  - Architecture
    - Spatial and temporal order
    - Components and connectivity

➢ **Design automation**
- Modeling
- Exploration
- Synthesis



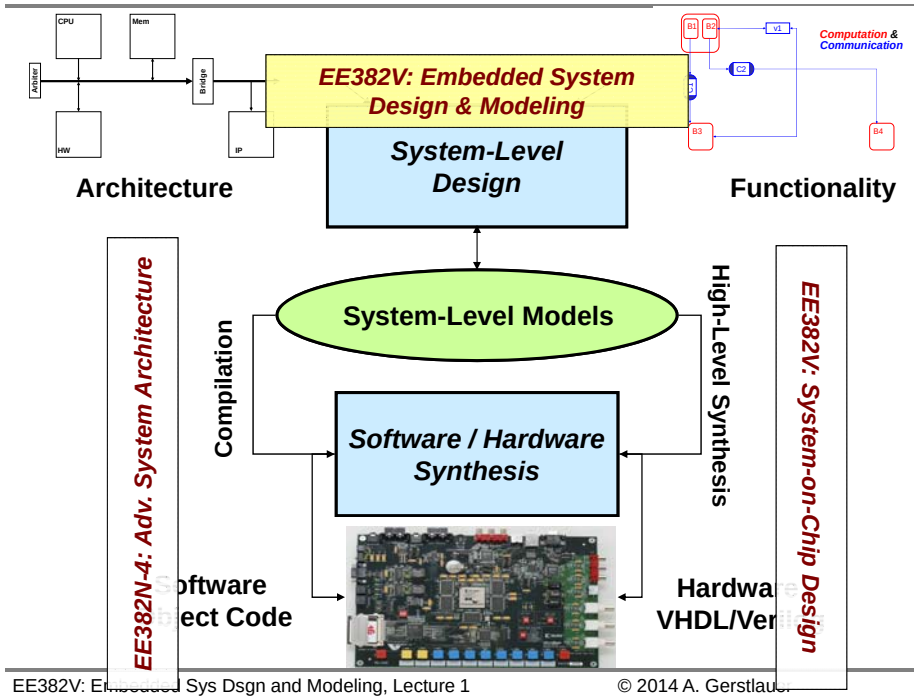
System-On-Chip Environment (SCE)





➢ **Commercial derivative for Japanese Aerospace Exploration Agency**

UT ECE Courses



Lecture 1: Outline

- ✓ Introduction
  - ✓ Embedded systems
  - ✓ System-level design
- **Course information**
  - Topics
  - Logistics
  - Projects
- Design methodology
  - System-level design flow
  - Models and methodologies

Course Topics

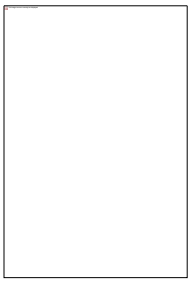
- **System-level design**
  - Methodologies and languages: SpecC, SystemC
- **Specification modeling**
  - Formal Models of Computation (MoC)
    - Parallel programming models, threads, dataflow, process networks
    - Hierarchical and concurrent finite state machine (FSM) models
- **System synthesis**
  - Design space exploration and optimization
    - Mapping and scheduling algorithms, exploration heuristics
    - System-level design tools: SCE
- **Architecture modeling**
  - Implementation and simulation (virtual prototyping) models
    - Host-compiled OS and processor models for computation
    - Transaction-level modeling of communication
- **Prerequisites**
  - Software: C/C++ (algorithms and data structures)
  - Hardware: VHDL/Verilog (digital design)
  - Embedded systems and embedded software

Class Administration

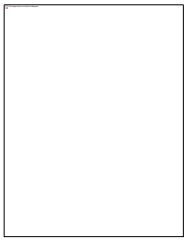
- **Schedule**
  - Lectures: TTh 11:00am-12:30pm, WEL 3.402
- **Instructor**
  - Prof. Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: ACE 6.118, TW 2-3pm, or after class/by appt.
- **Teaching Assistant**
  - Parisa Razaghi <parisa.r@utexas.edu>
    - Office hours: MF 3-4pm, ENS 110
- **Information**
  - Web page: http://www.ece.utexas.edu/~gerstl/ee382v_s14
  - Announcements, assignments, grades: Blackboard
  - Questions, discussions: Blackboard

Textbooks (1)

- **Recommended**
  - D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner, *Embedded System Design: Modeling, Synthesis, Verification*, Springer, 2009 ("*orange book*")



- **Optional**
  - E. A Lee, S. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2011
    - Models of computation (MoCs)
    - http://leeseshia.org



Textbooks (2)

- **Background material**
  - 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
    - Electronic copy of selected chapters on Blackboard

  - T. Groetker, S. Liao, G. Martin, S. Swan, *System Design with SystemC*, Kluwer, 2002 *("black book")*
    - Reference for SystemC language and methodology

Policies

- **Grading**
  - Homeworks: 20%
  - Labs: 20%
  - Midterm: 25%
  - Project: 35%
  - ➢ No late submissions!
- **Academic dishonesty**
  - Homeworks are independent
    - Discuss questions and problems with others
    - Turn in own, independently developed solution
  - Labs and project are teamwork
    - Teams of up to 3 students
    - One report and presentation

Homeworks and Labs

- **Two to three homeworks and one exam**
  - Cover theoretical aspects of system design
    - Languages
    - Models
    - Exploration and optimization
  - Some practical implementation
    - Exposure to general language and modeling concepts
- **Two to three labs**
  - Real-world system design
    - Design example using SpecC and System-on-Chip Environment (SCE)
  - From specification to implementation
    - Modeling
    - Design space exploration
    - Hardware/software synthesis

Project

- **Two options**
  - Research project
    - System design research problem
    - Literature survey on system design research area
  - Implementation project
    - Non-trivial system design example/case study
    - Specification, exploration, implementation
- **Project timeline (tentative)**
  - Abstract: March 4 (email)
  - Proposal, literature survey: March 18
  - Presentations: last week of classes (Apr. 29 & May 1)
  - Report: finals week (May 8)
  - ➢ Final report and presentation in publishable quality

Some Possible Projects

- **Design projects**
  - (Embedded) system design example
    - Specify, model, simulate, explore, synthesize using SCE
      - » Existing examples: MP3 Decoder, AC3 Decoder, Jpeg Encoder, GSM Vocoder
      - » Backend synthesis down to ARM+FPGA prototyping board
- **Research projects**
  - Modeling
    - Specification modeling
      - » Develop/modify a language or MoC: data parallel extensions of SpecC/SystemC
      - » Translation between MoCs & languages: from Matlab/SDF/… to SpecC/SystemC
    - Architecture modeling
      - » Component modeling: QEMU-SpecC/SystemC integration, bus modeling
      - » Automatic model generation: generate bus TLMs from abstract protocol descriptions
      - » OS modeling: OS-internal timing estimation and back-annotation
      - » Performance estimation and modeling (timing, power, reliability, …): statistical simulation, parallel or hardware/software co-simulation of functional & performance models
      - » Assertion-based verification in a TLM environment
  - Synthesis
    - Pick an implementation problem and solve it
      - » Decision making: machine learning for optimization (allocation, partitioning, scheduling), design space exploration for dataflow models/signal processing systems
      - » OS scheduling for power, performance, reliability
      - » Hardware or software synthesis for new OS/processors: targeting Linux in SCE

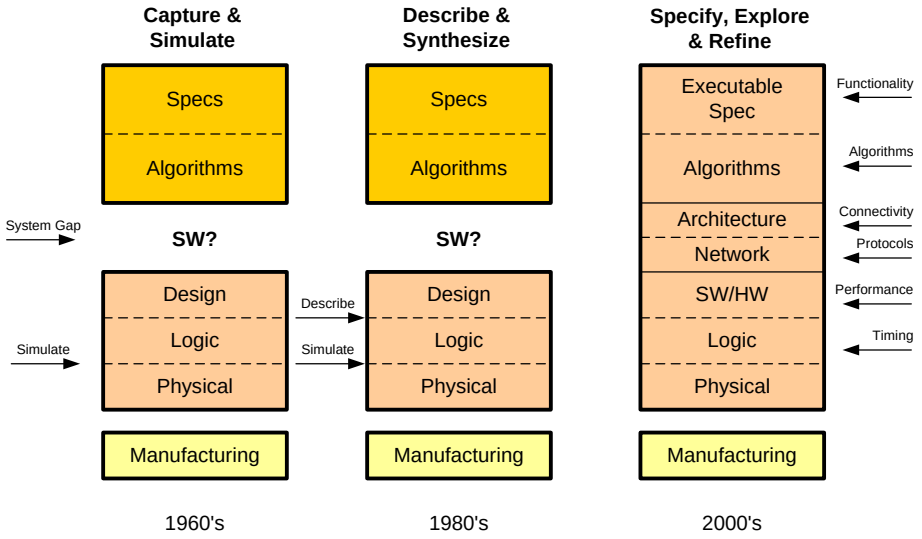
Successful Past Projects

- **Modeling**
  - A. Abdel-Hadi, J. Michel, "Real-Time Optimization of Video Transmission in a Network of AAVs," *VTC* 2011.
  - A. Pedram, C. Craven, T. Amimeur, “Modeling Cache Effects at the Transaction Level,” *IESS* 2009 **(best paper runner-up)**
  - A. Banerjee, “Transaction Level Modeling of Best Effort Channels for Networked Embedded Devices”, *IESS* 2009.
- **Exploration and synthesis**
  - J. Lin, A. Srivatsa, “Heterogeneous Multiprocessor Mapping for Real-Time Streaming Systems,” *ICASSP* 2011.
  - S. Lee, K. Saleem, J. Li, "Fine Grain Word Length Optimization for Dynamic Precision Scaling in DSP Systems," *VLSI-SoC* 2013 **(best paper candidate)**

Lecture 1: Outline

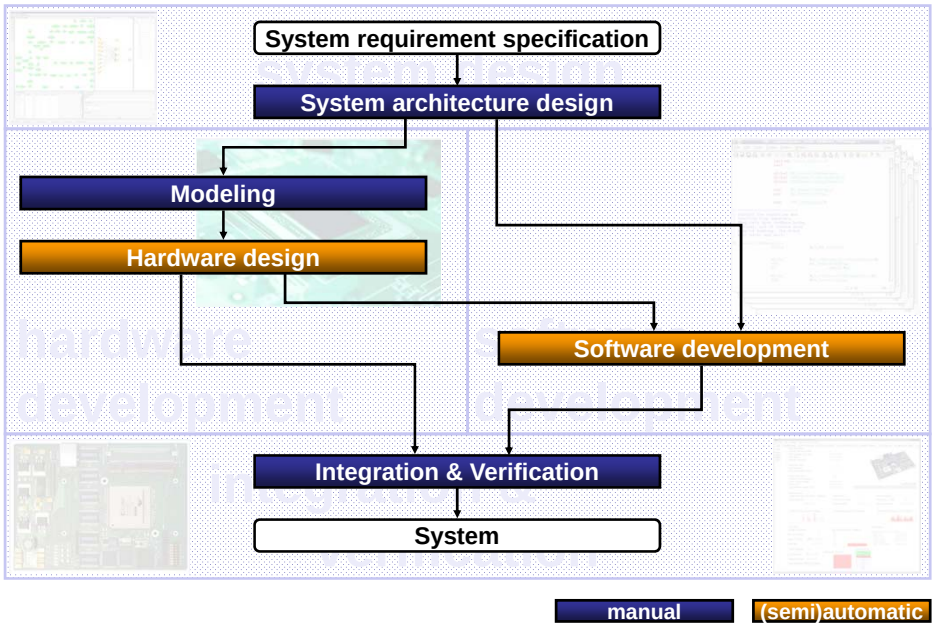
- ✓ Introduction
  - ✓ Embedded systems
  - ✓ System-level design
- ✓ Course information
  - ✓ Topics
  - ✓ Logistics
  - ✓ Projects
- **Design methodology**
  - System-level design flow
  - Models and methodologies

Evolution of Design Flows

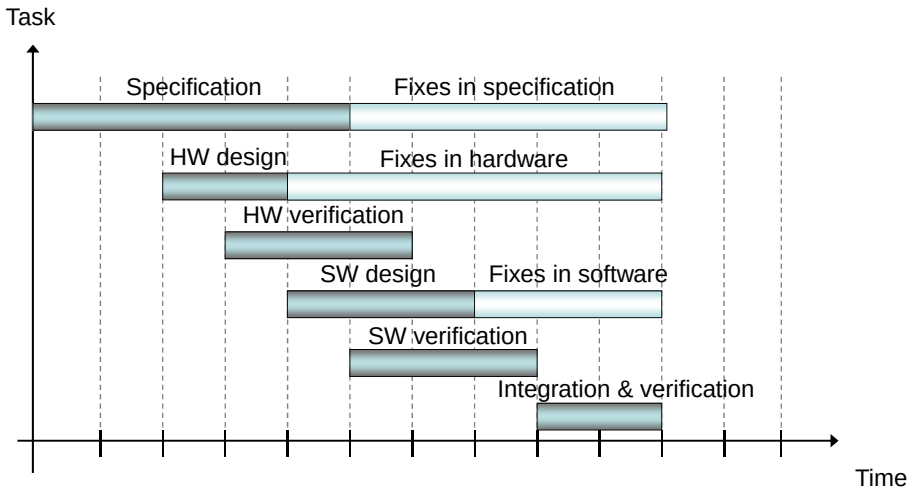


Source: D. Gajski, UC Irvine

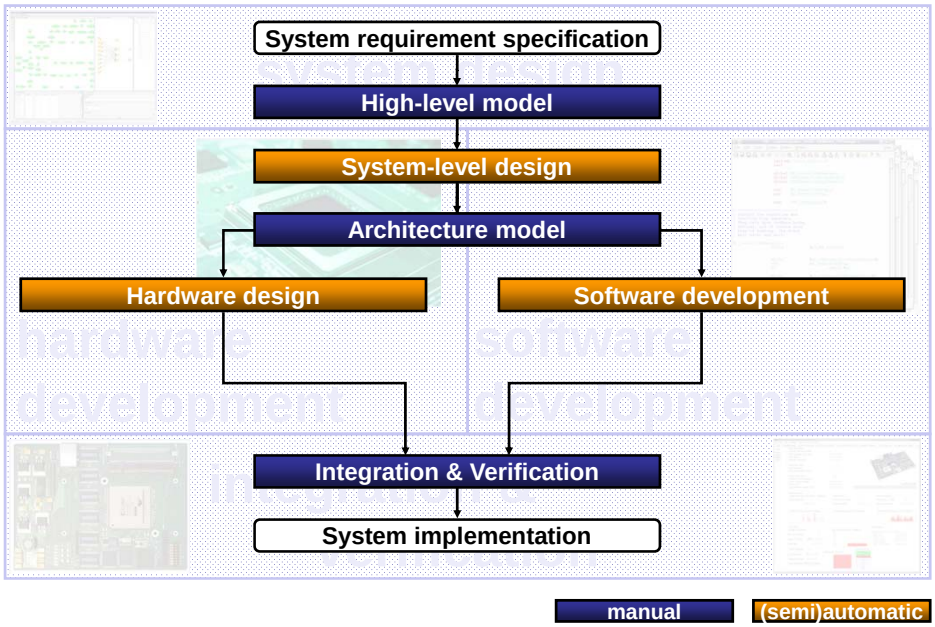
Classical System Design Flow



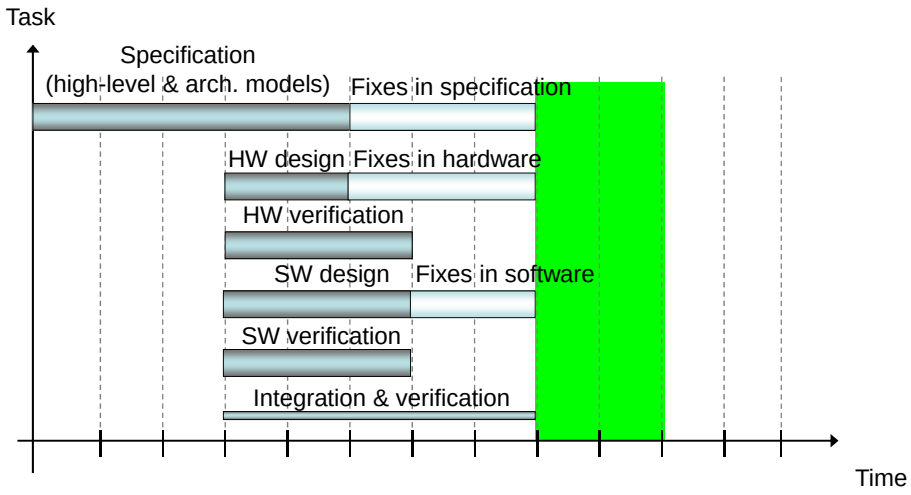
Classical Design Cycle



Electronic System-Level (ESL) Design Flow



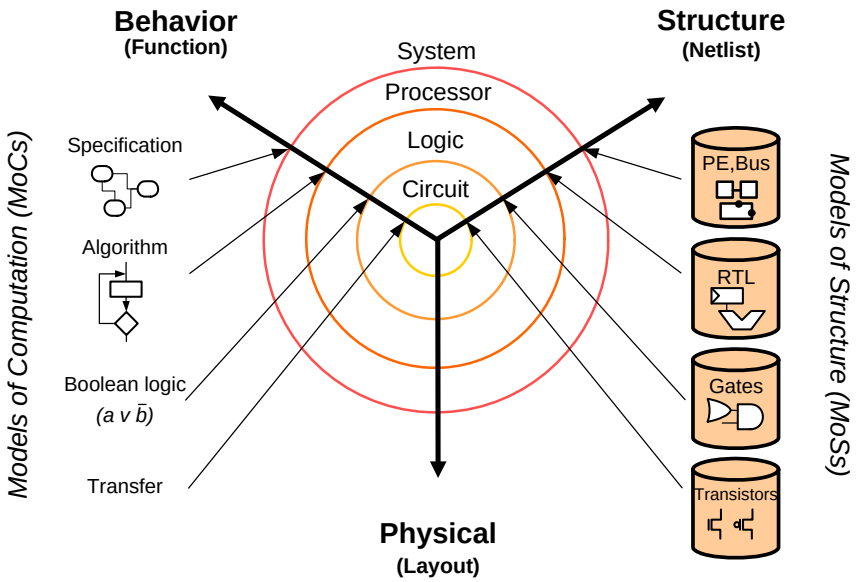
New ESL Design Cycle



Design Methodology

- **Formalization of a design flow**
  - Break into well-defined, repeatable steps
    - Set of models and transformations between them
    - Manual or automated
- **Models**
  - Design representations
    - Specification and documentation at interface between steps
- **Transformations**
  - Design decisions
    - Refine input model into an output model reflecting decisions

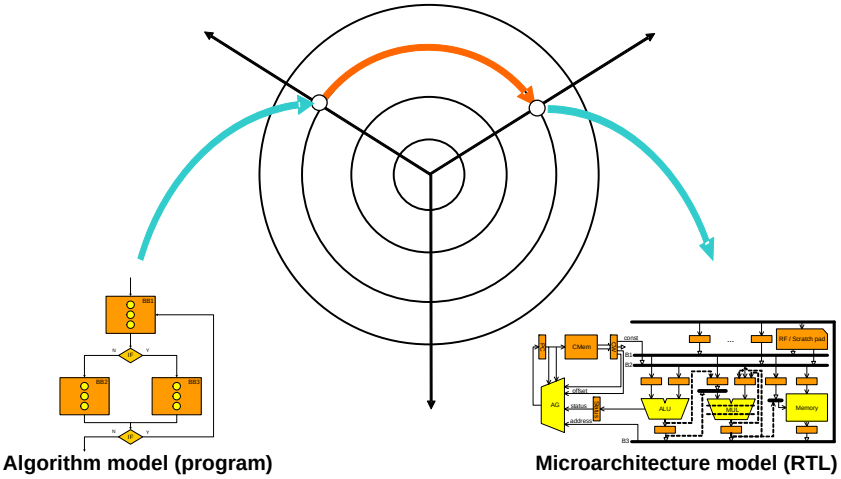
Y-Chart



Source: D. Gajski, UC Irvine

Processor Synthesis

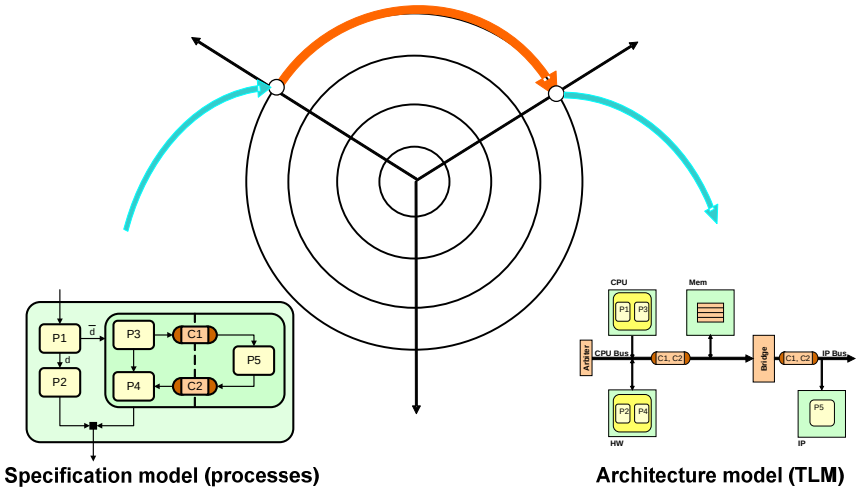
- **Software processor**
  - Compilation and linking
- **Hardware processor**
  - High-level synthesis



Source: D. Gajski, UC Irvine

System Synthesis

- **Structure**
  - Partitioning, mapping
- **Timing**
  - Scheduling

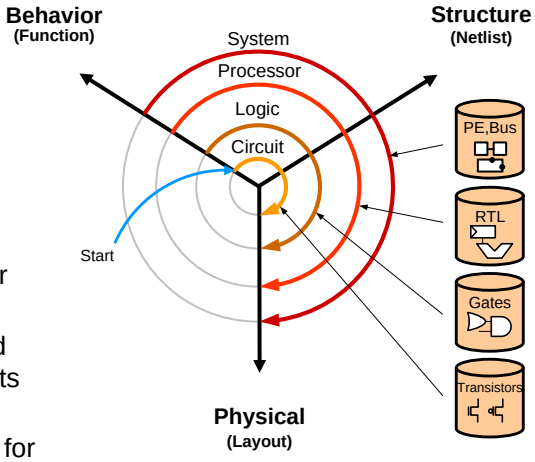


Specification model (processes) Architecture model (TLM)

Source: D. Gajski, UC Irvine

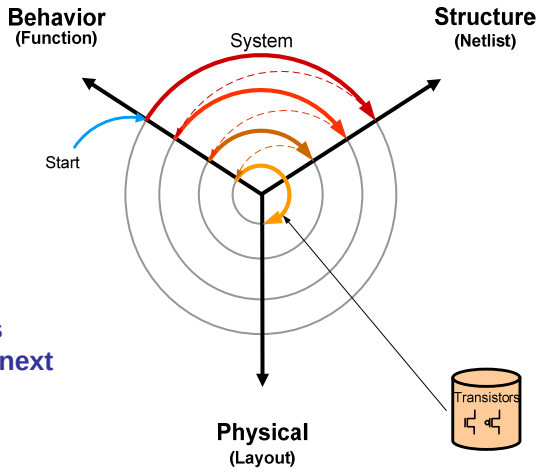
Bottom-Up Methodology

- **Each level generates library for the next higher level**
  - Circuit: Standard cells for logic level
  - Logic: RTL components for processor level
  - Processor: Processing and communication components for system level
  - System: System platforms for different applications
- **Floorplanning and layout on each level**



Source: D. Gajski, UC Irvine

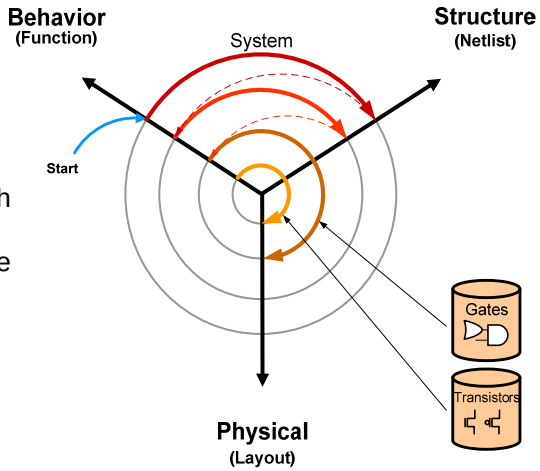
Top-down Methodology



- **Functional description is converted into component netlist on each level**
- **Each component function is decomposed further on the next abstraction level**
- **Layout is given only for transistor components**

Source: D. Gajski, UC Irvine

Meet-in-the-Middle Methodology

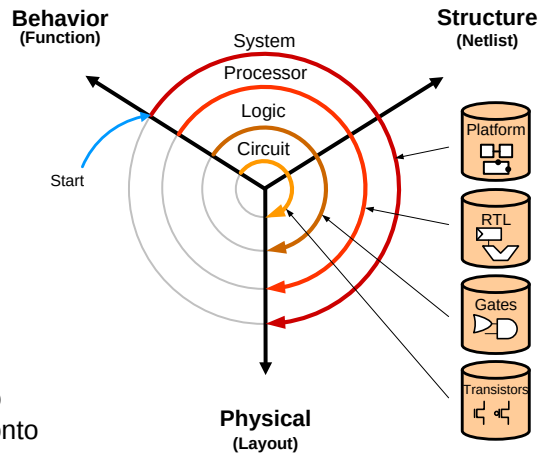


- **Gate netlist is hand-off**
- **Three levels of synthesis**
  - System is synthesized with processor components
  - Processor components are synthesized with RTL library
  - RTL components are synthesized with standard cells
- **Two levels of layout**
  - System layout is performed with standard cells
  - Standard cells layout with transistors

Source: D. Gajski, UC Irvine

Platform-Based Design

- **Meet-in-the-middle at the system level**
  - System platform with standard components
  - System design reduced to mapping of specification onto pre-defined platform



Source: D. Gajski, UC Irvine

Lecture 1: Summary

- ✓ **Introduction**
  - ✓ Embedded systems
  - ✓ System-level design
- ✓ **Course information**
  - ✓ Topics
  - ✓ Logistics
  - ✓ Projects
- ✓ **Design methodology**
  - ✓ System-level design flow
  - ✓ Models and methodologies