EE382N:
Embedded System Design and Modeling

Lecture 1 – Introduction

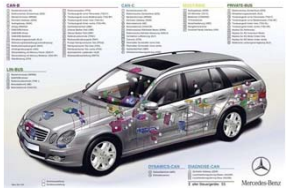
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

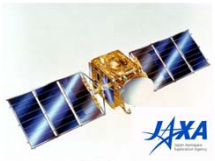
Lecture 1: Outline

- **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
    - Not general purpose
    - Known a priori
  - Tightly constrained
    - Guaranteed, not best effort
    - Performance, power, cost, reliability, security, …





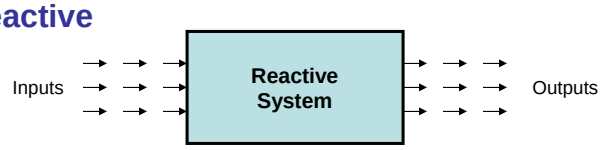
- **Ubiquitous**
  - Far bigger market than general-purpose computing (PCs, servers)
    - 98% of all processors sold [Turley02, embedded.com]

- **Growing complexities**
  - Application demands & technological advances
  - Increasingly networked and programmable
  - ➢ Internet of Things (IoT)

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 real time**

Embedded System Design

- **Correctly implement a specific set of *functions***
- **While satisfying *constraints***
  - Performance, cost, energy, power, thermal, …

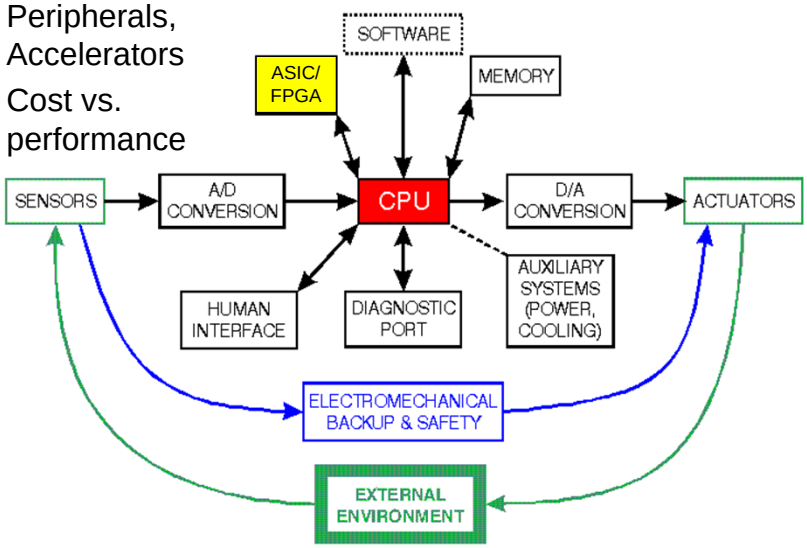
➢ **Specialization and Optimization**
  - Choice of *system architecture* and *application mapping*
  ➢ Large design spaces, and growing

➢ **General-purpose computing seeing similar needs**
  - Power, thermal, … constraints
  - Application/architecture specialization & optimization
  ➢ The two worlds are merging…

Source: M. Jacome, UT Austin

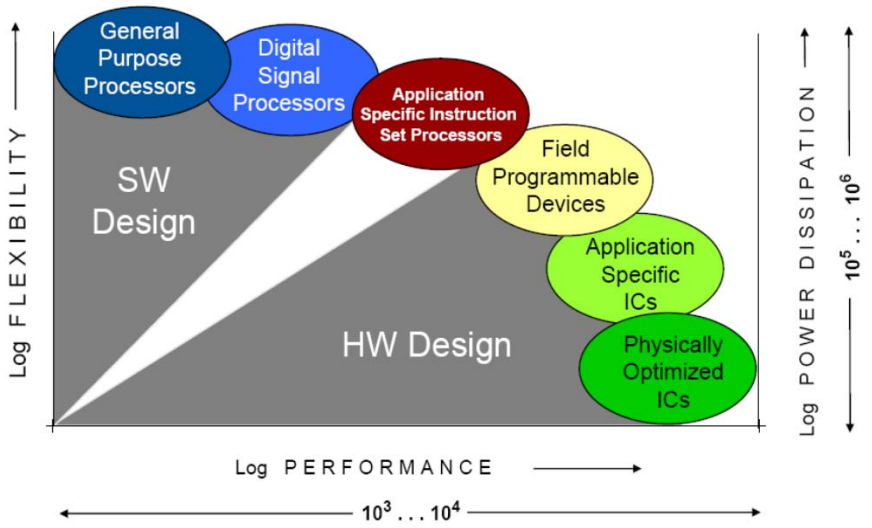
Traditional Embedded System

- **CPU-centric design**
  - Peripherals, Accelerators
  - Cost vs. performance



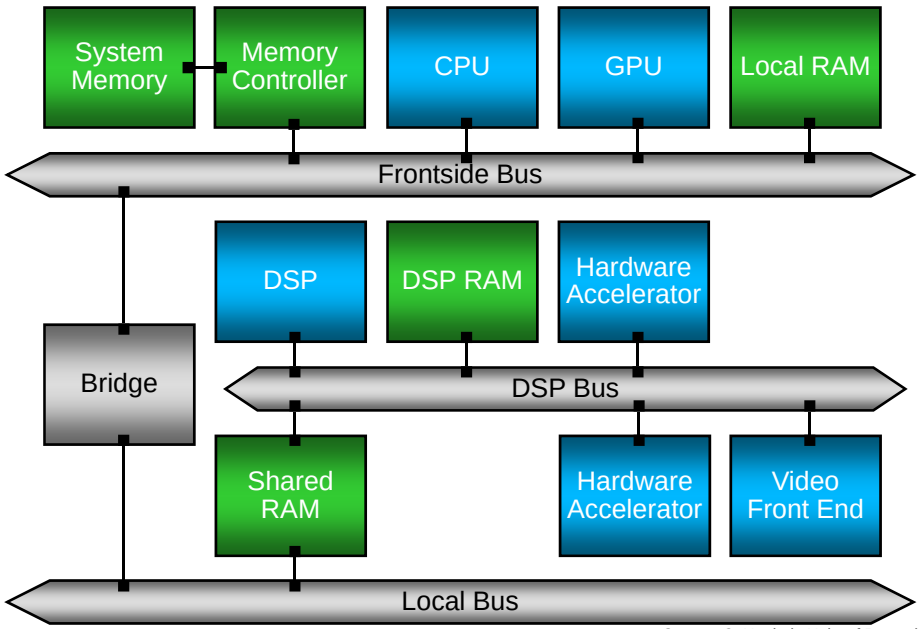
Source: M. Jacome, UT Austin

Implementation Options



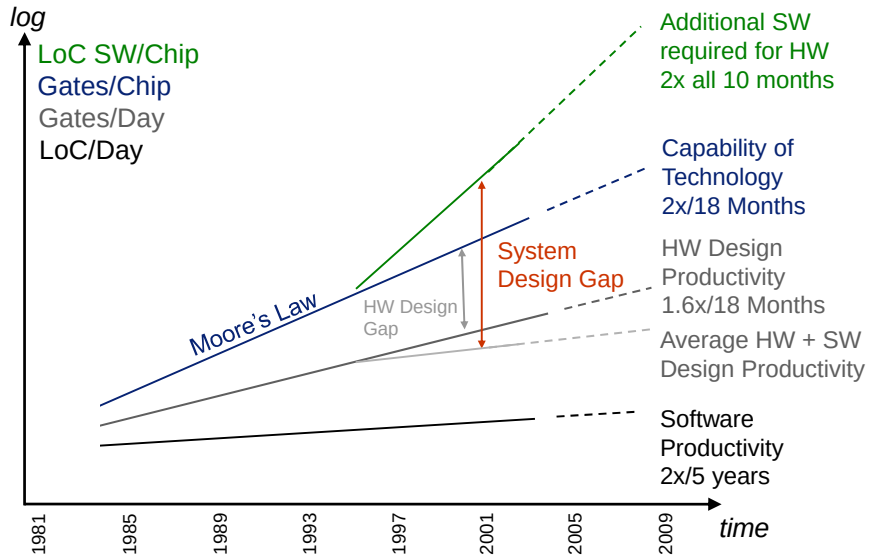
Source: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

Multi-Processor System-on-Chip (MPSoC)



Source: C. Haubelt, Univ. of Rostock

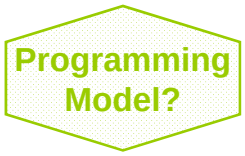
Productivity Gaps

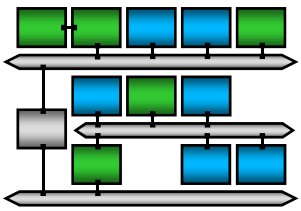


Source: W. Ecker, W. Müller, R. Dömer, *Hardware-dependent Software - Principles and Practice*, Springer 2009.

Design Challenges

- **Complexity**
  - High degree of parallelism
  - High degree of design freedom
  - Multiple optimization objectives & design constraints
    - Cost, performance, power, …
    - Reliability, safety
- **Heterogeneity**
  - Of components
    - Processors, memories, busses
  - Of design tasks
    - Architecture design
    - Application mapping





Source: C. Haubelt, Univ. of Rostock

Heterogeneity, Complexity

- **Managing complexity and heterogeneity challenge**
  - Mix of hardware design with software design
  - Mixes design styles within each of these categories
  - Mix of abstraction/detail/specificity
- **Systematic specification, modeling and design techniques**
  - Rigorous and unambiguous specification
  - Automated analysis & synthesis

➢ **Formal methods for analysis and synthesis are key**
  - It requires reconciling
    - Simplicity of modeling required by verification and synthesis
    - Complexity and heterogeneity of real world design

Key need ⇒ Formal models to capture/express the various types of behavior at different abstraction levels, and how those diverse formal models interact and can be analyzed and synthesized.

Source: M. Jacome, UT Austin

(Engineering) Models vs. Reality

- **"You can't strike oil by drilling through a map" [Solomon'68]**
  - Yet, maps are incredibly useful

➢ **We can make definitive statements about models from which we can *infer* properties of system realizations [Kopetz]**
  - ➢ Validity of inference depends on model fidelity
  - ➢ Always approximate

➢ **Assertions about (predicted) properties are always assertions about a model of the system**
  - ➢ Never truly properties of the final implemented system

Source: E. Lee, CEDA Keynote, DAC'13.

Reliability and Safety

- **Embedded systems often are used in life critical situations, where reliability and safety are more important criteria than performance**
  - Today, embedded systems are designed using a somewhat ad hoc approach that is heavily based on earlier experience with similar products and on manual design
- **Formal verification and automated synthesis are the surest ways to guarantee safety**
  - Both, formal verification and synthesis from high levels of abstraction have been demonstrated only for small, specialized languages with restricted semantics
  - ➢ Insufficient, given the *complexity* and *heterogeneity* found in typical embedded systems

Source: M. Jacome, UT Austin

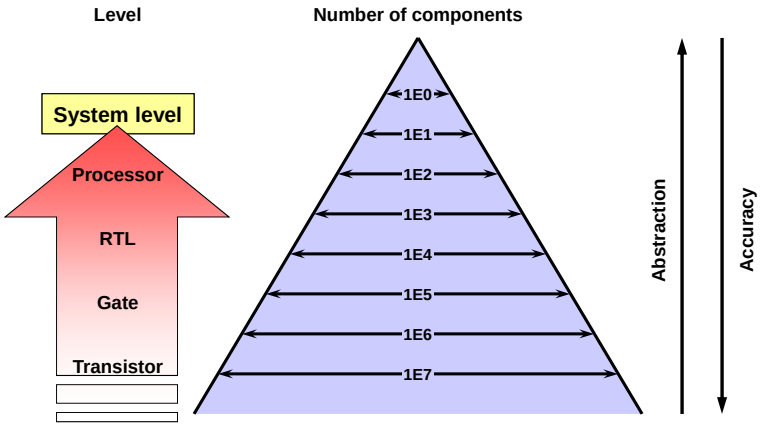
Desirable Design Methodology

- **Design should be based on the use of one or more *formal models* to describe the *behavior* of the *system* at a high level of abstraction**
  - Such behavior should be captured on an unbiased way, that is, before a decision on its decomposition into hardware and software components is taken
- **The final implementation of the system should be generated as much as possible using *automatic synthesis* from this high level of abstraction**
  - To ensure implementations that are “correct by construction”
- **Validation (through *simulation* or *verification*) should be done as much as possible at the higher levels of abstraction**

Source: M. Jacome, UT Austin

Abstraction Levels

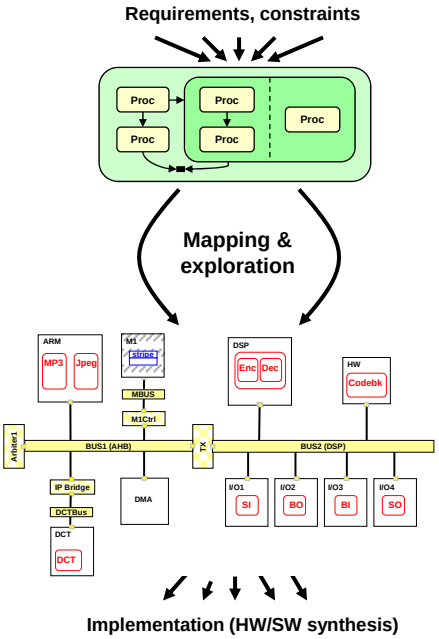
- **Move to higher levels of abstraction [ITRS07, itrs.net]**
  - ➢ System-level design



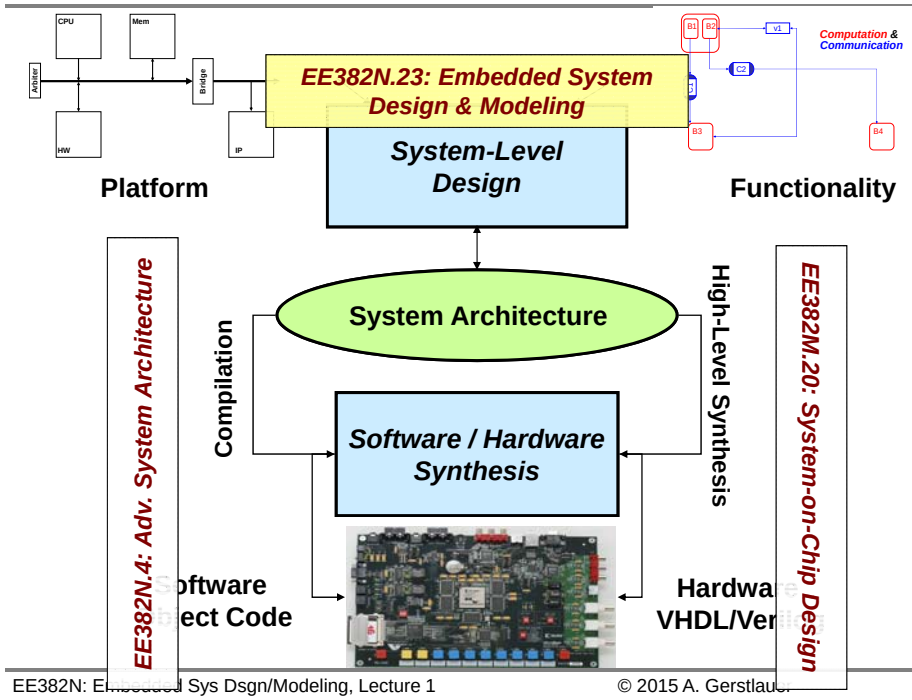
Source: R. Doemer, UC Irvine

System-Level Design

- **From specification**
  - Functionality, behavior
    - Concurrency, order
    - Constraints
- **To implementation**
  - MPSoC architecture
    - Spatial and temporal order
    - Components and connectivity
- ➢ **Design automation**
  - Modeling
  - Synthesis
  - Verification



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]
  - System-level design tools [SCE]

1. **Specification modeling**
   - Formal Models of Computation (MoC)
     - Parallel programming models, threads, dataflow, process networks
     - Hierarchical and concurrent finite state machine (FSM) models
2. **Performance modeling**
   - Estimation and simulation (virtual prototyping) models
     - Host-compiled OS and processor models for computation
     - Transaction-level modeling of communication
3. **System synthesis**
   - Design space exploration and optimization
     - Mapping, partitioning and scheduling algorithms
     - Design space exploration heuristics

- **Prerequisites**
  - Software: C/C++ (algorithms and data structures)
  - Hardware: VHDL/Verilog (digital design)
  - Embedded systems and embedded software

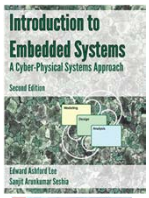
Class Administration

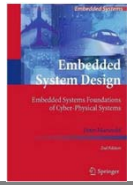
- **Schedule**
  - Lectures: MW 3-4:30pm, RLM 5.114
  - Midterm exam (tentative): December 2 (in class)
- **Instructor**
  - Prof. Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: POB 6.118, M 4:30-5:30pm, W 2-3pm, or after class/by appt.
- **Teaching Assistant**
  - Zhuoran Zhao <zhuoran@utexas.edu>
    - Office hours: TBD
- **Information**
  - Web page: http://www.ece.utexas.edu/~gerstl/ee382n_f15
  - Announcements, assignments, grades: Canvas
  - Questions, discussions: Canvas

Textbooks (1)

- **Recommended**
  - D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner, *Embedded System Design: Modeling, Synthesis, Verification*, Springer, 2009 ("*orange book*")
    - http://www.cecs.uci.edu/embedded-system-design-book/

- **Additional references**
  - E. A Lee, S. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed., 2015
    - Available for download at http://leeseshia.org
  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 2nd ed., Springer, 2011
    - http://ls12-www.cs.tu-dortmund.de/~marwedel/es-book/





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 Canvas

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

Policies

- **Grading**
  - Homeworks: 20%
  - Labs: 20%
  - Midterm: 20%
  - Project: 40%
  - ➢ 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

- **Three to four 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
- **Three labs**
  - Real-world system design
    - Design example using SpecC and System-on-Chip Environment (SCE)
  - From specification to implementation
    - Specification modeling
    - Performance 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: September 30 (Canvas)
  - Proposal, literature survey: October 28 (Canvas)
  - Presentations: November 23 & 25 (in class)
  - Report: finals week (December 15)
  - ➢ 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
    - Performance 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 optimization/exploration 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

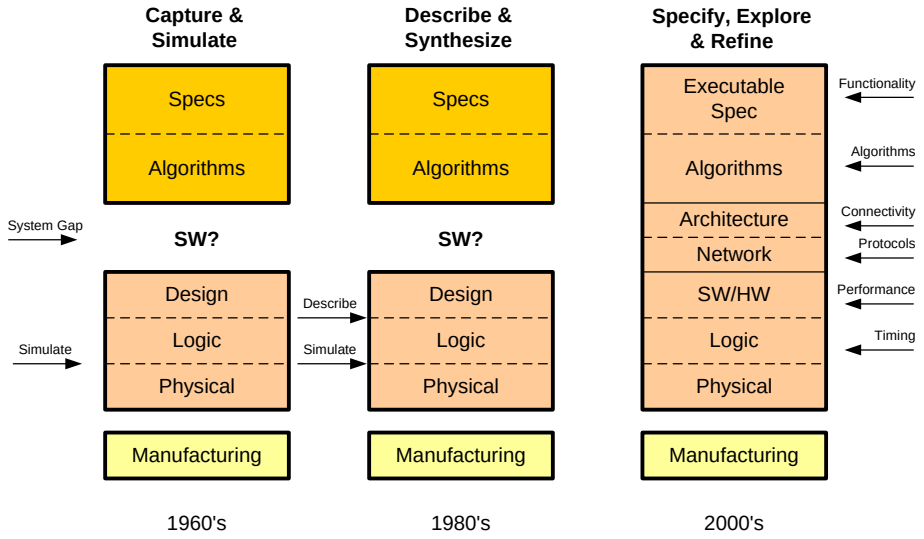
Successful Past Projects

- **Modeling**
  - X. Zheng, “Learning-Based Analytical Cross-Platform Performance Prediction,” *SAMOS* 2015 **(best paper award)**
  - 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**
  - S. Lee, K. Saleem, J. Li, "Fine Grain Word Length Optimization for Dynamic Precision Scaling in DSP Systems," *VLSI-SoC* 2013 **(best paper candidate)**
  - J. Lin, A. Srivatsa, “Heterogeneous Multiprocessor Mapping for Real-Time Streaming Systems,” *ICASSP* 2011.

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

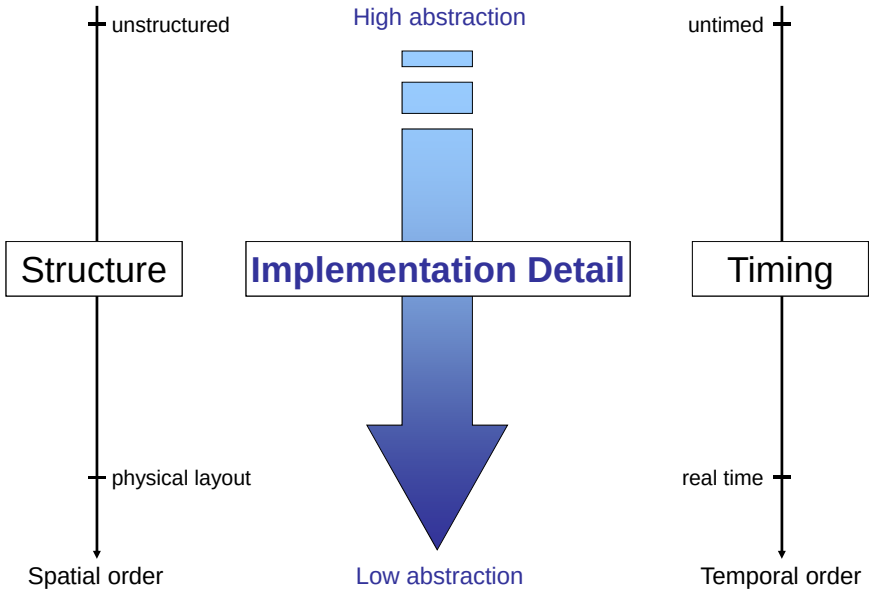
Design Process

- **Sequence of steps that transforms a set of requirements described informally into a detailed description that can be used for manufacturing**
  - Intermediate steps with transformation from a more abstract description to a more detailed one (*refinement*)
- **A designer can perform step-by-step refinement**
  - The "input" description is a *specification*
  - The final description of the design is an *implementation*

➢ **Take a model of the design at a level of abstraction and *refine* it to a lower one (level of detail ↑).**
  - Ensure that the properties at the lower level of abstraction are verified, and that the performance indices are satisfactory
  - Thus, refinement process involves mapping constraints, performance indices and properties to the lower level, so that they can be computed for the next level down

Source: M. Jacome, UT Austin

Abstraction Levels

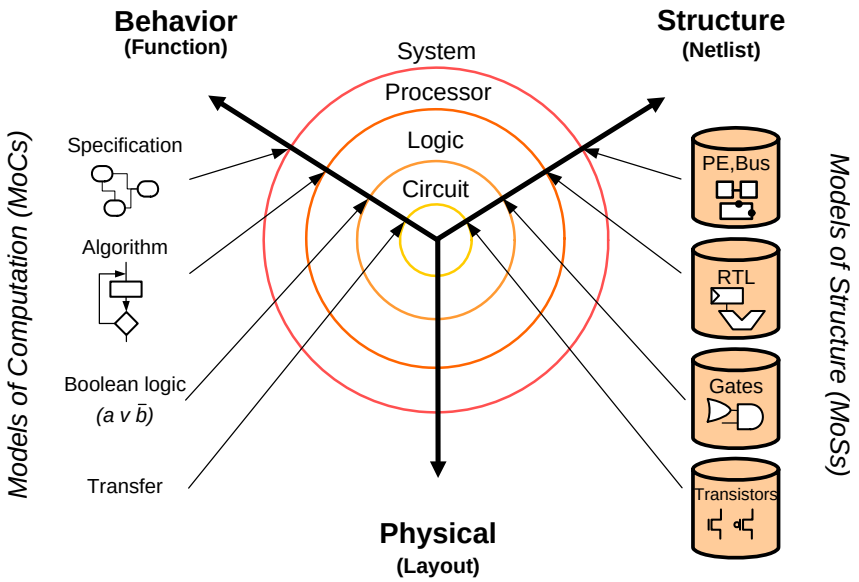


Design Methodology

- **Set of Models**
  - Design representations
    - Specification and documentation at interface between steps
- **Set of Transformations**
  - Design decisions and design steps
    - Refine input model into an output model reflecting decisions

➢ **Formalization of a design flow**
  - Break into well-defined, repeatable steps
  - ➢ Automate

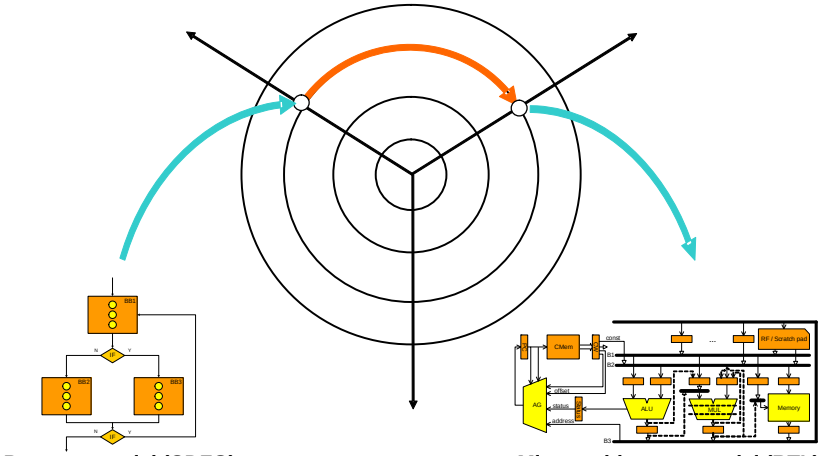
Y-Chart



Source: D. Gajski, UC Irvine

Processor Synthesis

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

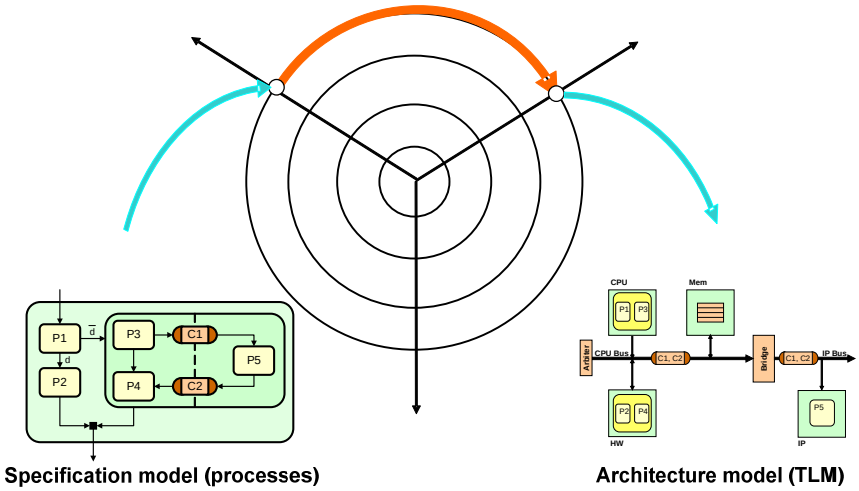


Program model (CDFG) **Microarchitecture model (RTL)**

Source: D. Gajski, UC Irvine

System Synthesis

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

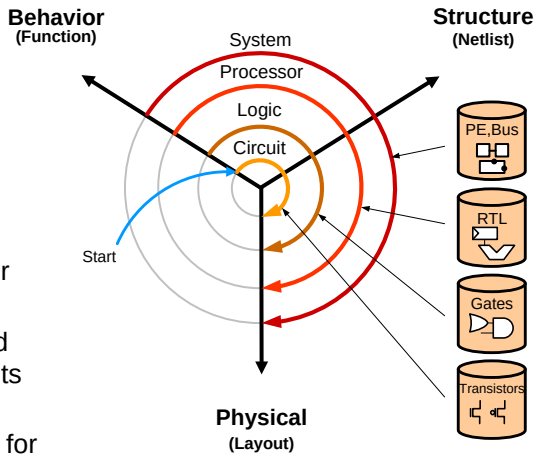


Specification model (processes) Architecture model (TLM)

Source: D. Gajski, UC Irvine

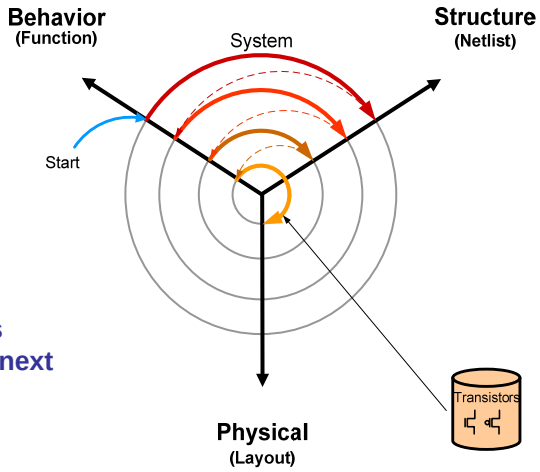
Bottom-Up Methodology

- **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

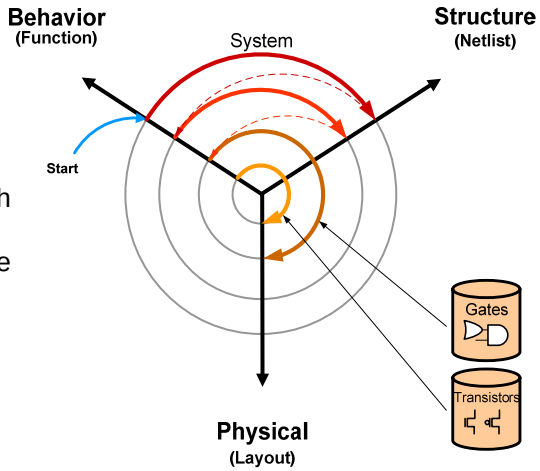
Top-down Methodology



- 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

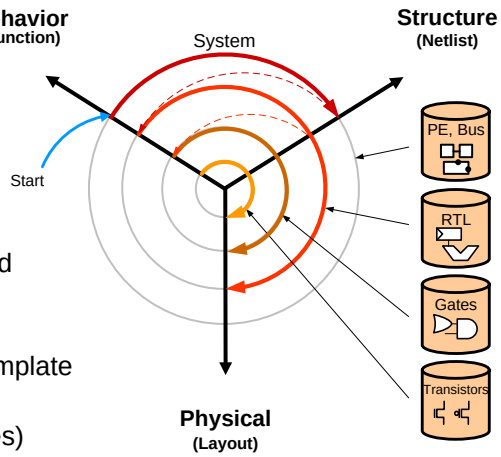
Meet-in-the-Middle Methodology



- 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

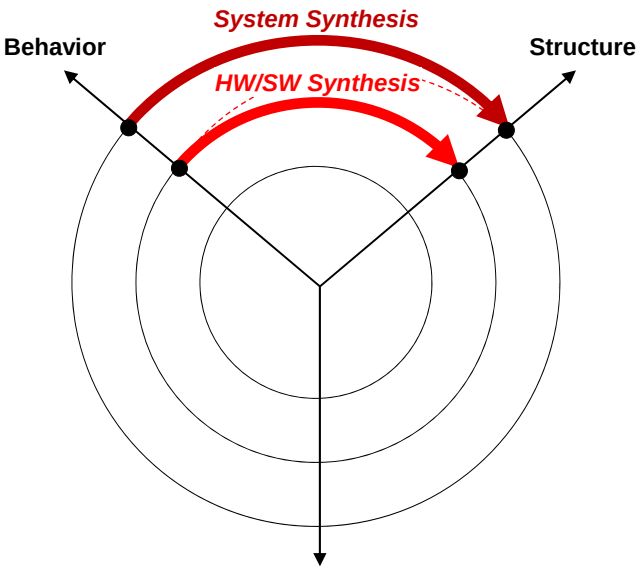
Platform-Based Design



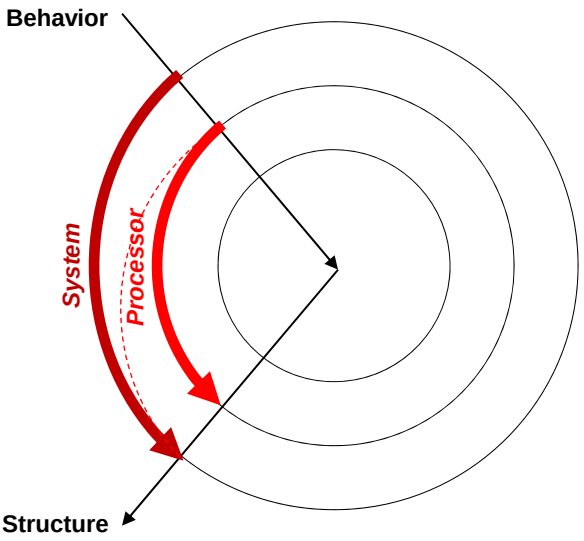
- **Meet-in-the-middle at the system level**
  - System platform with standard components
  - System synthesis to map specification onto platform template
  - Some custom processor are synthesized (to RTL and gates)
  - Other (programmable) processors are pre-synthesized and just need software compilation
  - Layout and floorplanning at the SoC level

Source: D. Gajski, UC Irvine

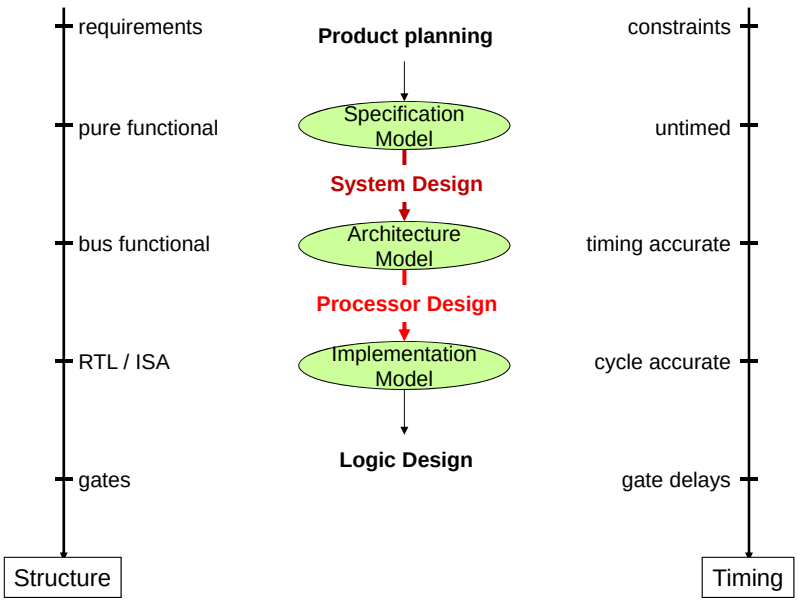
System-Level Design Methodology



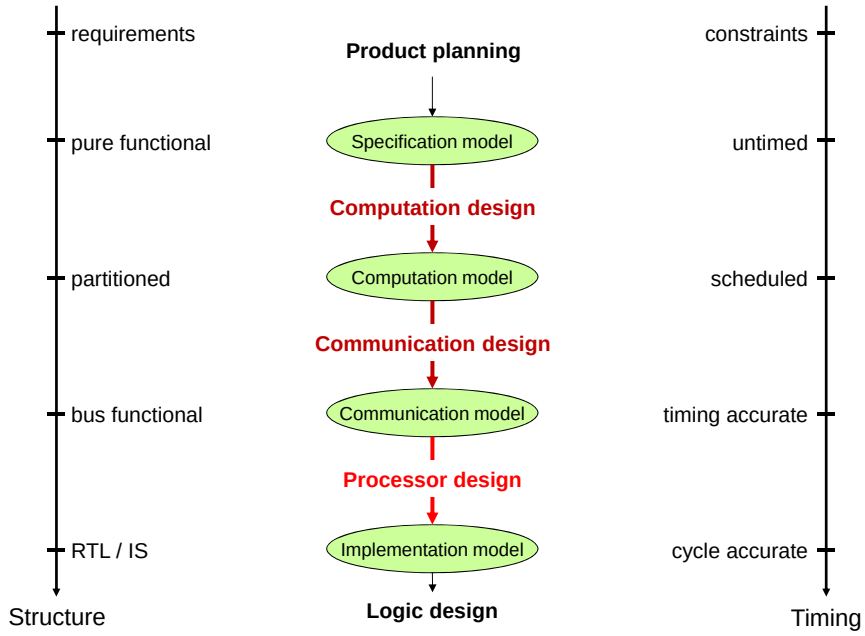
System-Level Design Methodology



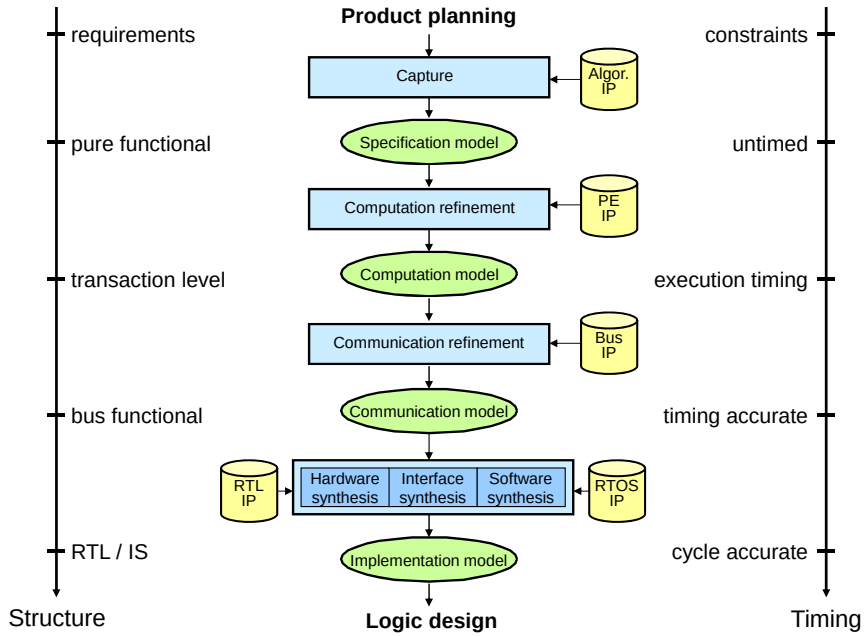
System-Level Design Flow



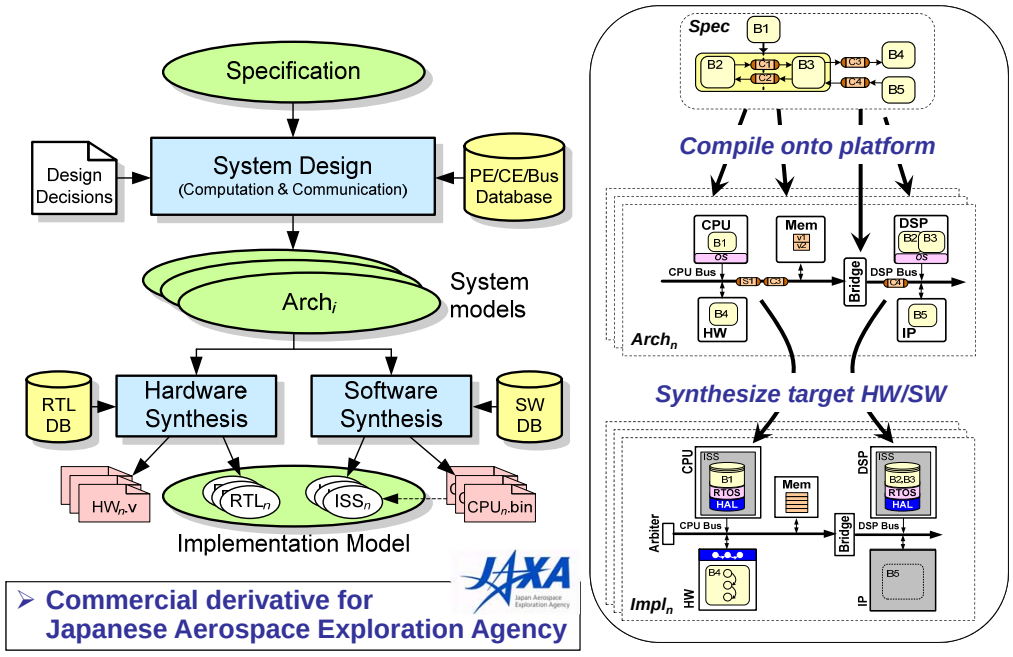
SpecC Design Flow



SpecC Design Methodology



System-On-Chip Environment (SCE)



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

Lecture 1: Summary

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