ECE382N.23: Embedded System Design and Modeling

Lecture 1 – Introduction

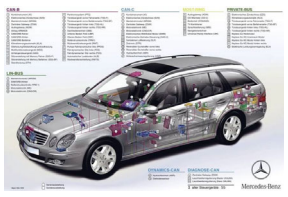
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - Design challenges
  - Formal methods and models
  - 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
    - Real time/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
  - ➢ Cyber-Physical Systems (CPS), Internet of Things (IoT)

Embedded System Design

- **Correctly implement a specific set of *functions***
  - Hardware and software
- **While satisfying *constraints***
  - Real-time, cost, energy, power, thermal, …

➢ **Application-/domain-specific & resource-constrained**
  - Opportunity and need to optimize
  - Choice of *system architecture* and *application mapping*
  - ➢ Large design spaces, and growing

➢ **General-purpose computing seeing similar needs**
  - Physical limits of scaling w/ power, thermal, … constraints
  - Application/architecture specialization & optimization
  - ➢ The two worlds are merging…

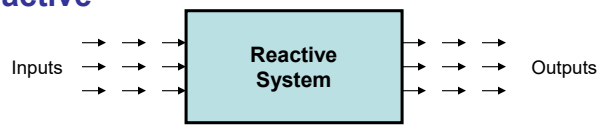
Source: M. Jacome, UT Austin

Cyber-Physical Systems (CPS)

- **Not transformative**

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

- **But reactive**

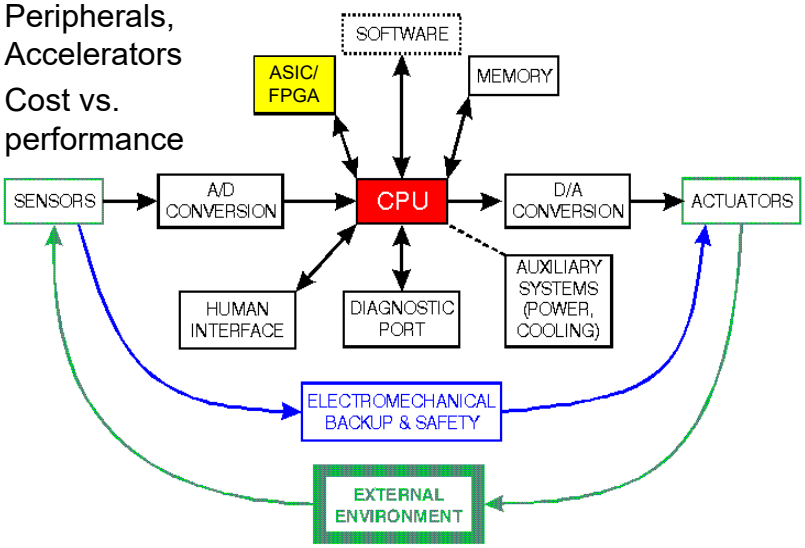


  - Continuous interaction with environment
  - ➢ Sense and act on the physical world

- ➢ **Concurrency and real time**

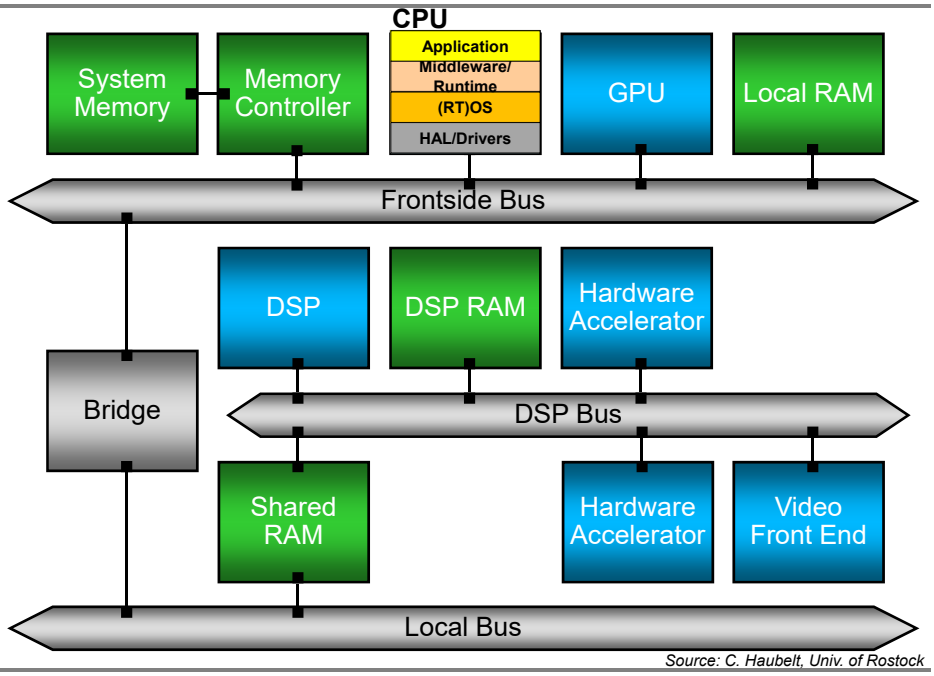
Traditional Embedded System

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



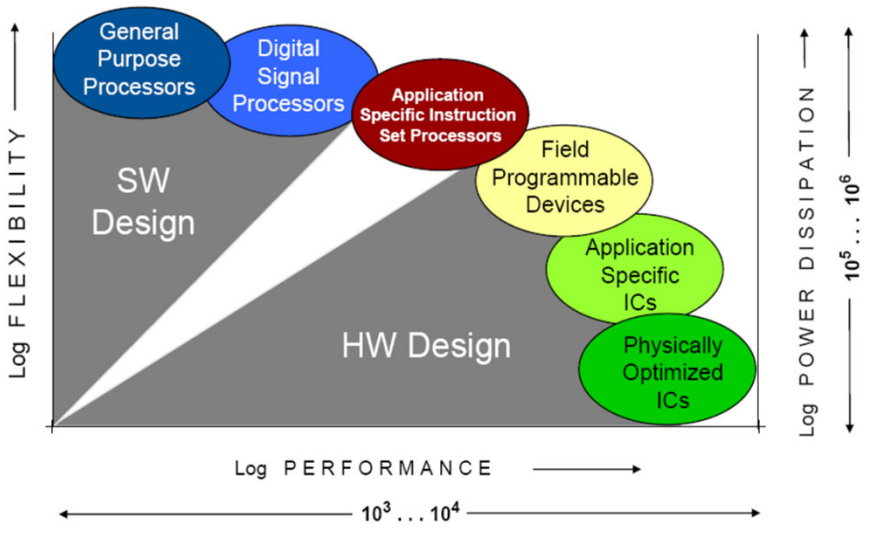
Source: M. Jacome, UT Austin

Multi-Processor System-on-Chip (MPSoC)



Source: C. Haubelt, Univ. of Rostock

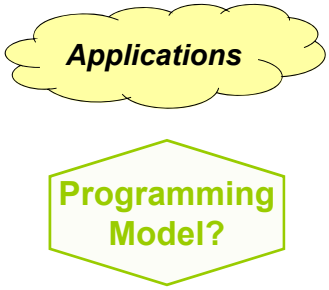
Implementation Options

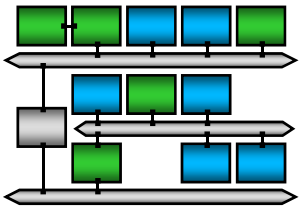


Source: T. Noll, RWTH Aachen, via R. Leupers, "From ASIP to MPSoC", Computer Engineering Colloquium, TU Delft, 2006

Design Challenges

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

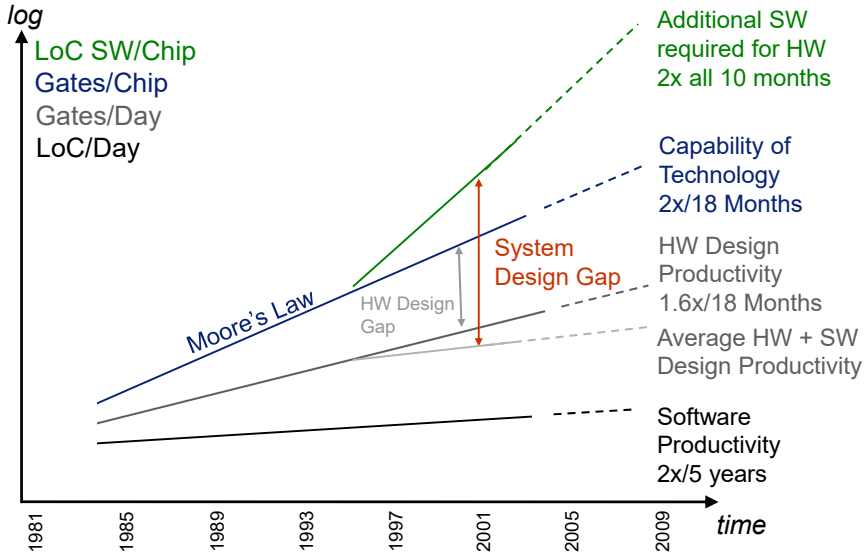




Domain-specific system-on-chip (DSSoC)

Source: C. Haubelt, Univ. of Rostock

Productivity Gaps



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

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

Formal Design Methods

- **Managing complexity, heterogeneity, correctness challenges**
  - 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 & design space exploration
- ➢ **Formal methods for analysis, synthesis & exploration are key**
  - It requires reconciling
    - Simplicity of modeling required by verification and synthesis
    - Complexity and heterogeneity of real world design

Key need ⇒ Abstractions into 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" [Golob'68]**
  - Yet, maps are incredibly useful
- **"All models are wrong, some are useful" [Box'76]**
  - Abstraction of reality

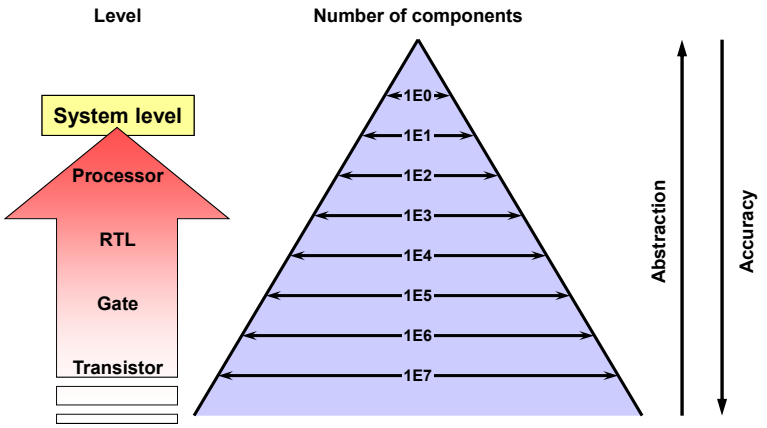
➢ **We can make definitive statements about models from which we can *infer* properties of system realizations [Kopetz]**
  - ➢ Validity of inference depends on model fidelity
  - ➢ Always approximate

➢ **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.

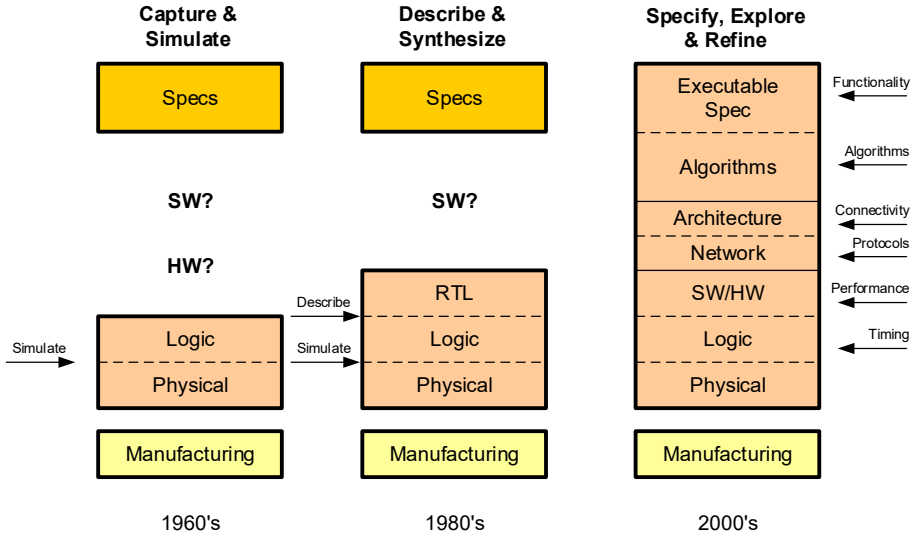
Abstraction Levels

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



Source: R. Doemer, UC Irvine

Evolution of Design Flows



Source: D. Gajski, UC Irvine

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**

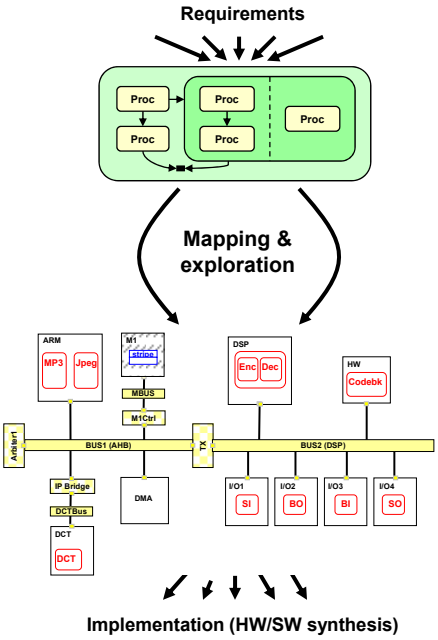
➢ **Model-based design**

Source: M. Jacome, UT Austin

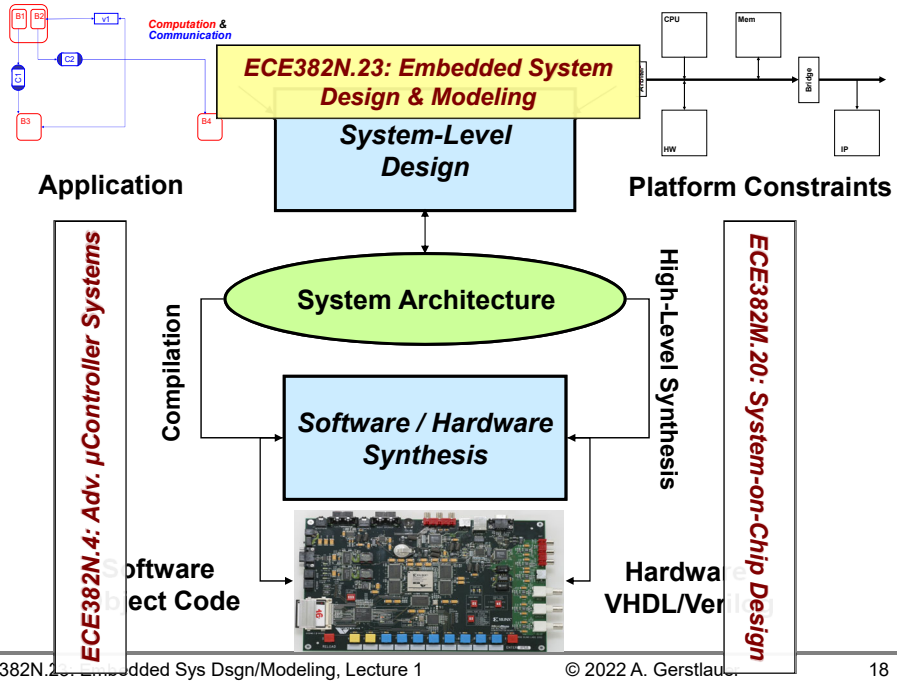
System-Level Design

- **From system specification**
  - Functionality, behavior
    - Concurrency, order
    - Constraints
- **To system implementation**
  - MPSoC architecture
    - Spatial and temporal order
    - Components and connectivity
    - Hardware and software

➢ **Design automation**
- Modeling
- Synthesis & exploration
- Verification



UT ECE Courses



Lecture 1: Outline

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

Course Topics

➢ **System-level design**

1. **System specification**
   - Application models
     - Parallel programming models, threads, dataflow, process networks
     - Hierarchical and concurrent finite state machine (FSM) models
2. **System modeling**
   - Performance estimation and evaluation
     - Analytical and simulation-based modeling
     - Machine learning based models
3. **System synthesis**
   - Design space exploration and optimization
     - Mapping, partitioning and scheduling algorithms
     - Design space exploration & optimization heuristics

➢ **Prerequisites**

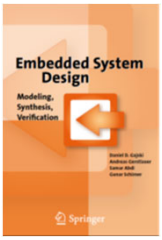
- ➢ Software: C/C++ (algorithms and data structures)
- ➢ Hardware: VHDL/Verilog (digital design)
- ➢ Embedded systems and embedded software

Class Administration

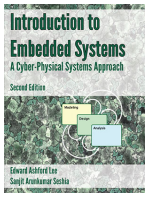
- **Schedule**
  - Lectures: TTh 11:00am-12:30pm, ECJ 1.318
  - Midterm exam (tentative): November 17 (in class)
- **Instructor**
  - Prof. Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: EER 5.882, TTh 1-2pm, or after class/by appointment
- **Teaching Assistant**
  - Erika Susana Alcorta (Susy) <esalcort@utexas.edu>
    - Office hours: TBD
- **Information**
  - Web page: http://www.ece.utexas.edu/~gerstl/ece382n_f22
  - Announcements, assignments, grades: Canvas
  - Questions, discussions: Piazza

Textbooks

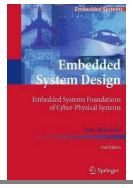
- **Recommended (but optional)**
  - D. Gajski, S. Abdi, A. Gerstlauer, G. Schirner, *Embedded System Design: Modeling, Synthesis, Verification*, Springer, 2009 ("*orange book*")
    - http://www.cecs.uci.edu/embedded-system-design-book/



  - E. A Lee, S. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed., 2017
    - Available for download at http://leeseshia.org



  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 4th ed., Springer, 2021
    - https://link.springer.com/book/10.1007/978-3-030-60910-8



Policies

- **Grading**
  - Homeworks: 30%
  - Midterm: 20%
  - Project: 50%
  - ➢ No late submissions!
- **Academic dishonesty**
  - Homeworks are independent
    - Discuss questions and problems with others
    - Turn in own, independently developed solution
  - Project is teamwork
    - Teams of up to 3 students
    - One report and presentation per teams

Homeworks and Exam

- **Homeworks**
  - Cover theoretical aspects of system design
    - System specification
    - System modeling
    - Synthesis and exploration
  - Some practical use of tools
    - Exposure to general language, modeling and optimization concepts
- **Exam**
  - Theoretical knowledge
    - Very similar to homework questions
    - Many homework questions have been old exam problems

Project

- **Two options**
  - Research project
    - System-level design research problem
  - Implementation project
    - Non-trivial embedded system design example/case study
- **Project timeline (tentative)**
  - Proposal: September 2 (email/office hours)
  - Part 1: September 30 (Canvas)
    - Research project: literature survey / related work
    - Implementation project: system specification
  - Part 2: October 31 (Canvas)
    - Research project: methods & preliminary results
    - Implementation project: system modeling and exploration
  - Presentations: November 29 & December 1 (in class)
  - Final Report: December 5
    - Research project: research paper in publishable quality
    - Implementation project: working system (in simulation or physical)

Some Possible Projects

- **Design projects**
  - (Embedded) system design example
    - Specify, model, explore, and implement
      - » Embedded machine learning, TinyML, IoT, e.g. in health care, robotics, smart *X*…
      - » Your choice of tools and target hardware/software platform
- **Research projects**
  - Modeling
    - System specification
      - » Develop a new/extend an existing MoC and associated analysis techniques
      - » Develop a new specification language for capturing existing/new MoC
    - System modeling
      - » Component modeling: CPU, GPU, accelerator power/performance models/simulators
      - » Machine learning-based power/performance/…. estimation and prediction
      - » Parallel or FPGA-based simulation of hardware/software/network systems
  - Synthesis & exploration
    - Pick an optimization/exploration problem and solve it
      - » Network-/system-level mapping, scheduling and design space exploration
      - » Machine learning-based design space exploration and optimization methods
      - » Hardware or software synthesis for different optimization targets
      - » Trading off functionality and performance/energy (approximate computing)

Successful Past Class Projects

- **Modeling**
  - A. K. Ananda Kumar, “Learning-Based CPU Power Modeling,” MLCAD’19.
  - K. Punniyamurthy, B. Boroujerdian, “GATSim: Abstract Timing Simulation of GPUs,” *DATE’*17.
  - X. Zheng, “Learning-Based Analytical Cross-Platform Performance Prediction,” *SAMOS’*15 **(best paper award)**
  - A. Abdel-Hadi, J. Michel, "Real-Time Optimization of Video Transmission in a Network of AAVs," *VTC’*11.
  - A. Pedram, C. Craven, T. Amimeur, “Modeling Cache Effects at the Transaction Level,” *IESS’*09 **(best paper runner-up)**
  - A. Banerjee, “Transaction Level Modeling of Best Effort Channels for Networked Embedded Devices”, *IESS’*09.
- **Exploration and synthesis**
  - K. Mirzazad, Z. Zhao, "Quality/Latency-Aware Real-time Scheduling of Distributed Streaming IoT Applications," *ACM TECS 2019.*
  - S. Lee, K. Saleem, J. Li, "Fine Grain Word Length Optimization for Dynamic Precision Scaling in DSP Systems," *VLSI-SoC’*13 **(best paper candidate)**
  - J. Lin, A. Srivatsa, “Heterogeneous Multiprocessor Mapping for Real-Time Streaming Systems,” *ICASSP’*11.

Lecture 1: Outline

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

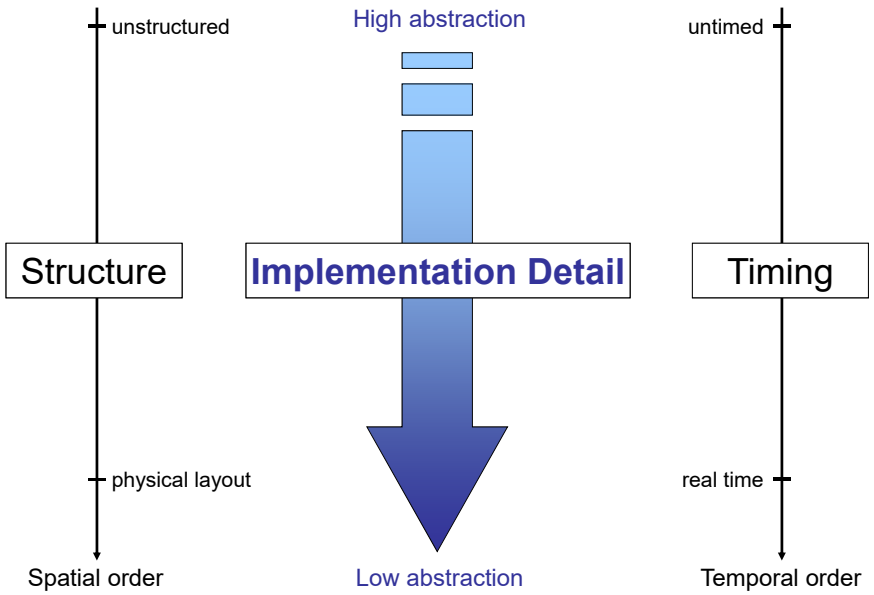
Design Process

- **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



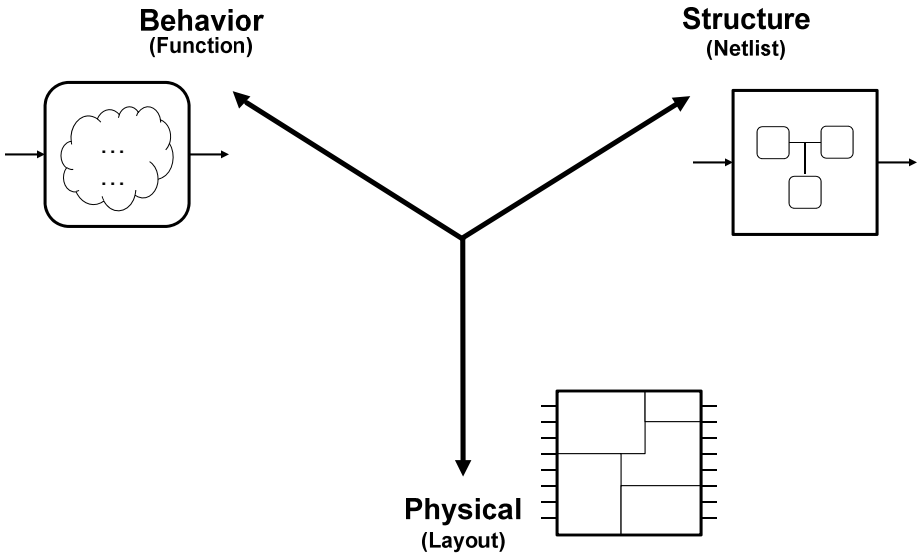
Design Methodology

- **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

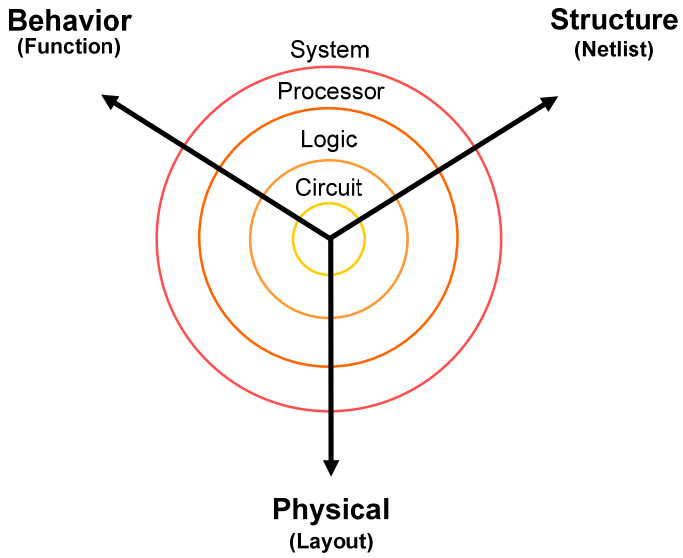
- **3 Design Views**



Source: D. Gajski, UC Irvine

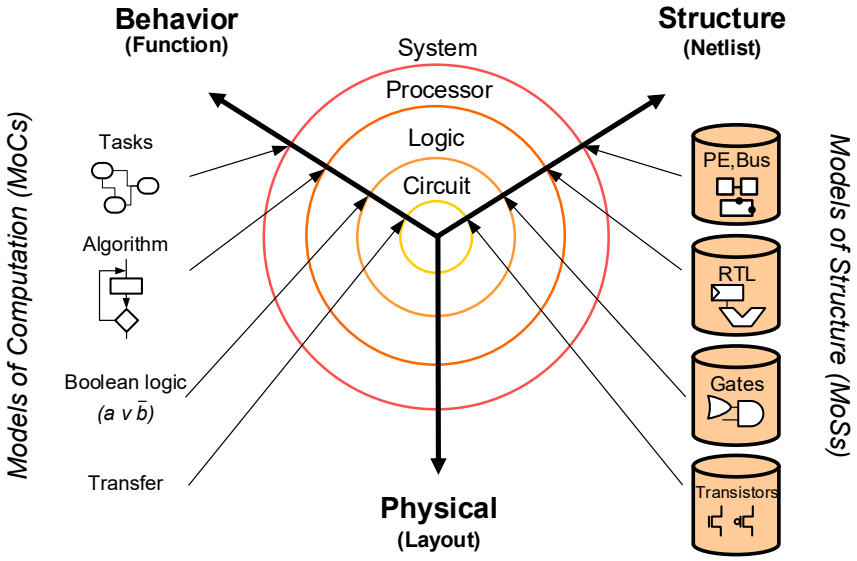
Y-Chart

- **4 Abstraction Levels**



Source: D. Gajski, UC Irvine

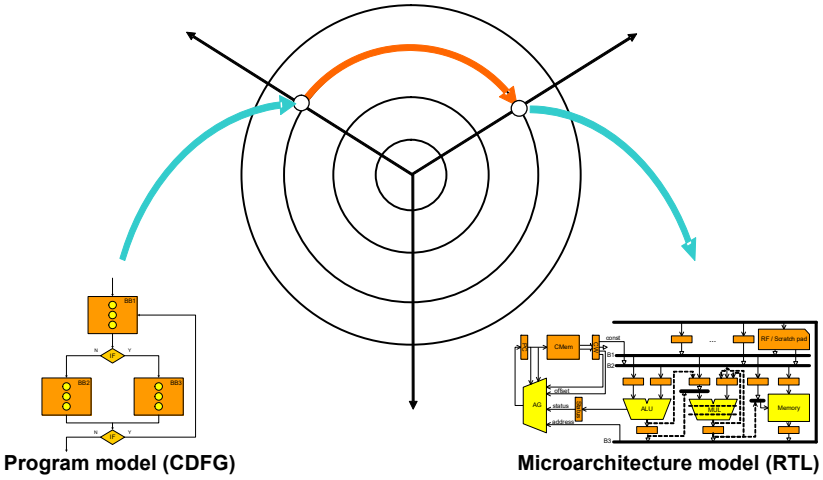
Y-Chart



Source: D. Gajski, UC Irvine

Processor Synthesis

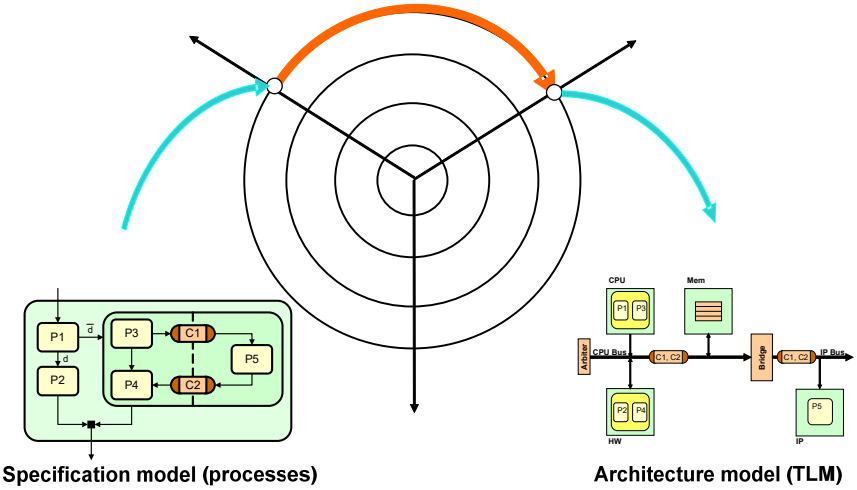
- **Software processor**
  - Compilation and linking
- **Hardware processor**
  - High-level synthesis



Source: D. Gajski, UC Irvine

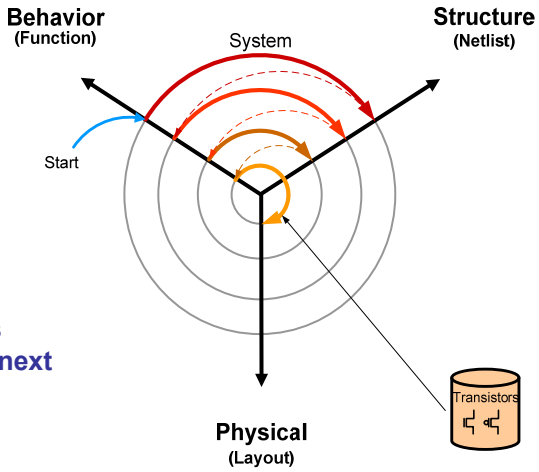
System Synthesis

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



Source: D. Gajski, UC Irvine

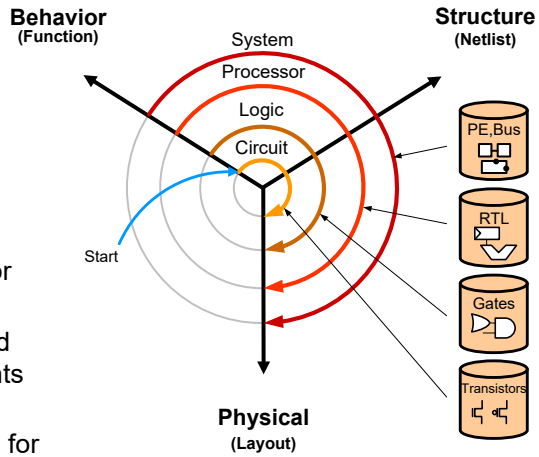
Top-down Methodology



- 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

Bottom-Up Methodology

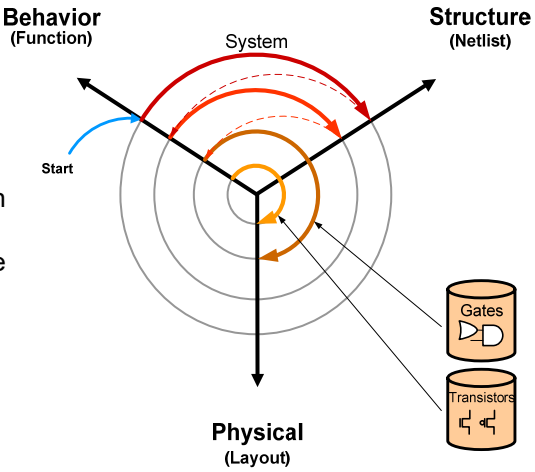


- Each level generates library for the next higher level
  - Circuit: Standard cells for logic level
  - Logic: RTL components for processor level
  - Processor: Processing and communication components for system level
  - System: System platforms for different applications
- Floorplanning and layout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

Source: D. Gajski, UC Irvine

Meet-in-the-Middle Methodology

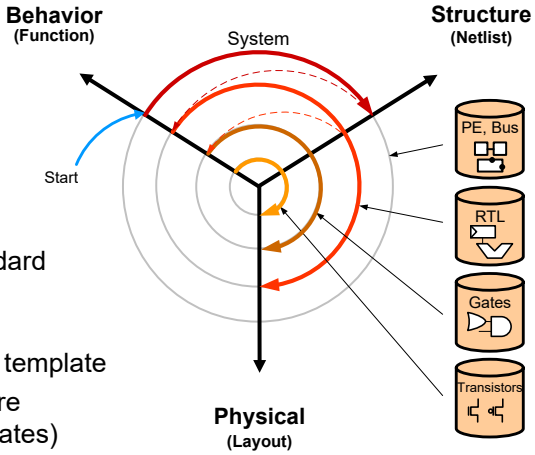
- **Gate netlist is hand-off**
- **Three levels of synthesis**
  - System is synthesized with processor components
  - Processor components are synthesized with RTL library
  - RTL components are synthesized with standard cells
- **Two levels of layout**
  - System layout is performed with standard cells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



Source: D. Gajski, UC Irvine

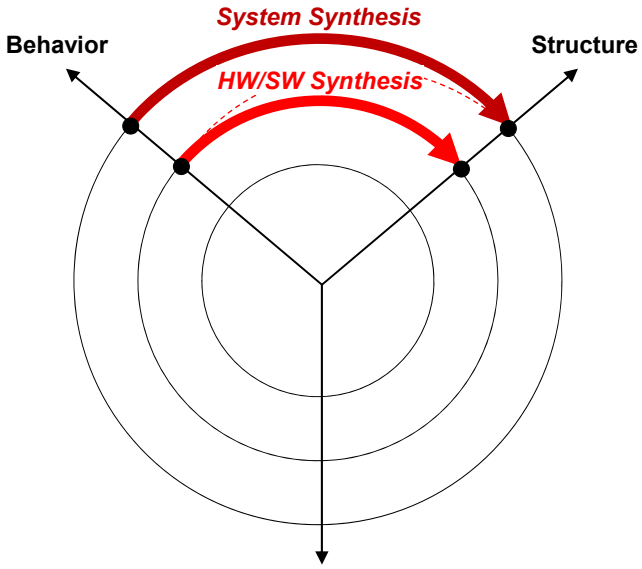
Platform-Based Design

- **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

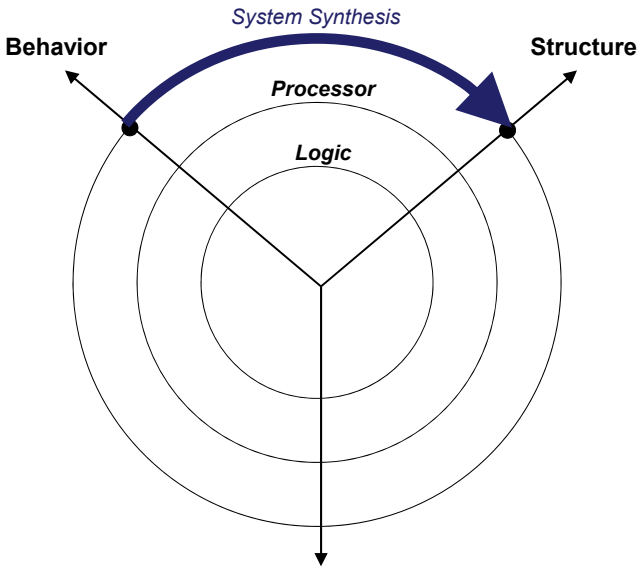


Source: D. Gajski, UC Irvine

System-Level Design Methodology

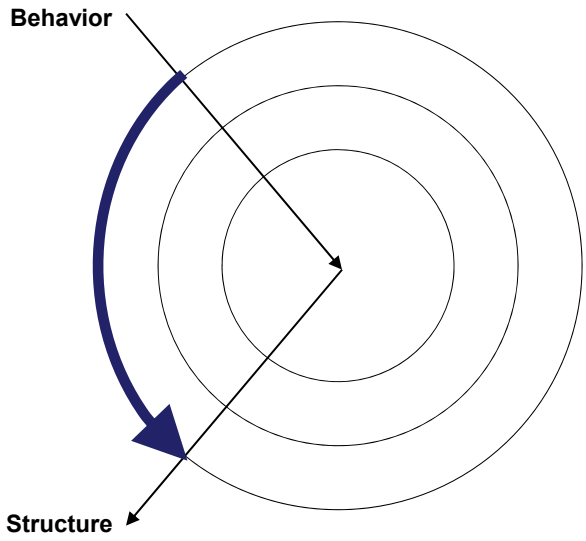


Synthesis Process



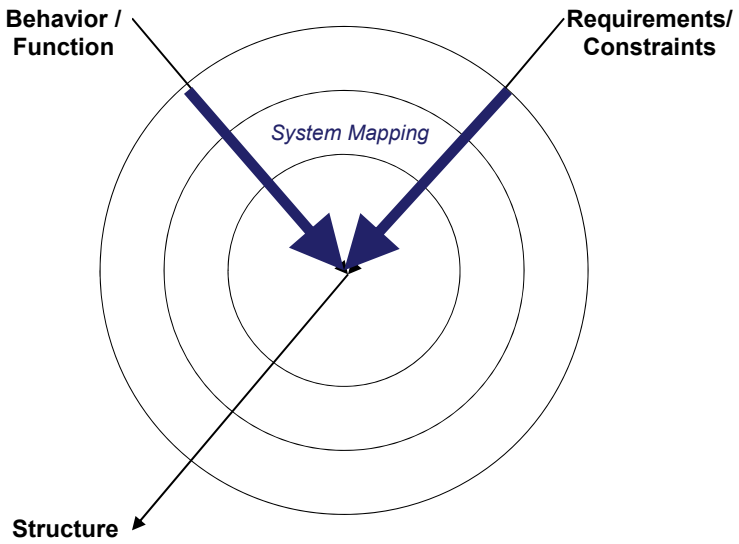
Synthesis Process

- **Gajski's Y-Chart**



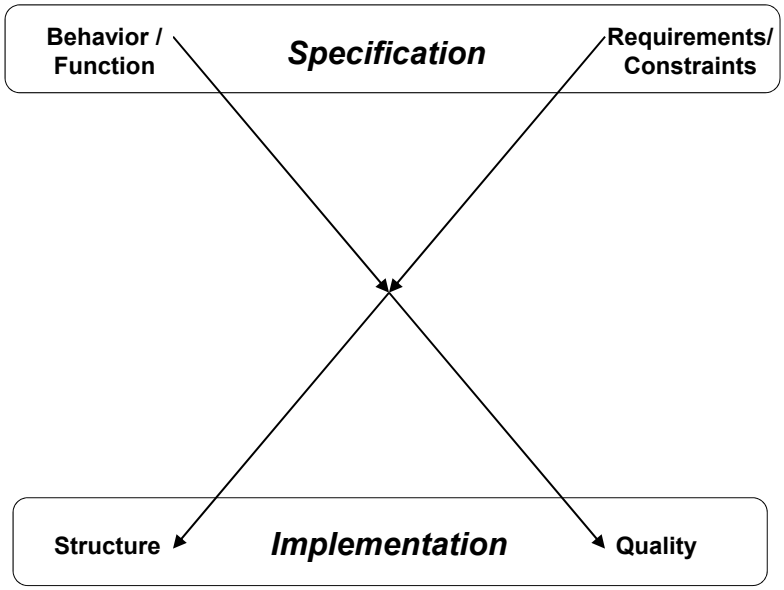
Synthesis Process

- **Platform-based design**



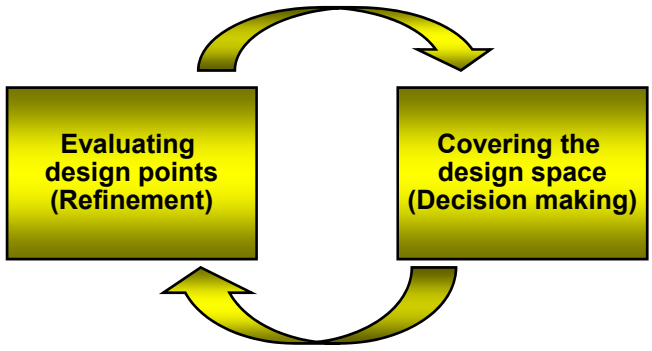
Synthesis Process

- **X-Chart**



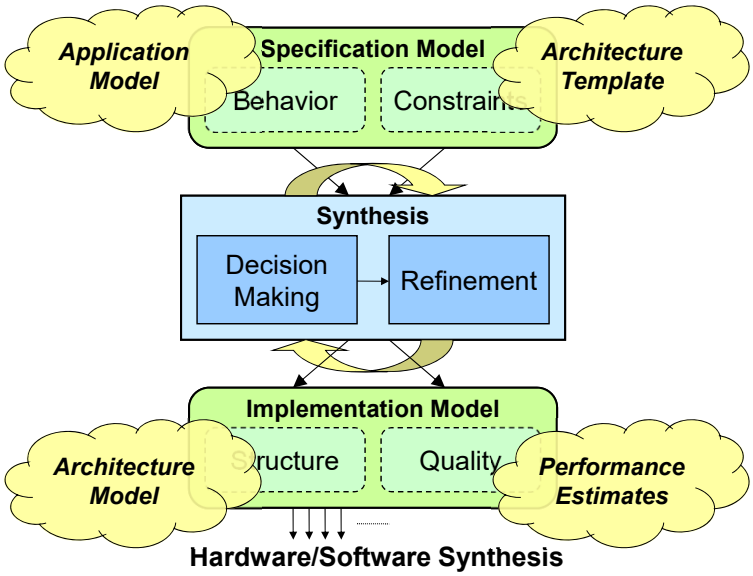
Design Space Exploration (DSE)

- **Iterative process**
  - Decision making
  - Evaluation
  - ➢ Cover design space during the exploration



Source: C. Haubelt, J. Teich, Univ. of Erlangen-Nuremberg

System-Level Synthesis Process



Source: A. Gerstlauer, C. Haubelt, A. Pimentel, et al., "Electronic System-Level Synthesis Methodologies," TCAD, 2009.

Lecture 1: Outline

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