EE382N.23: Embedded System Design and Modeling

Lecture 1 – Introduction

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

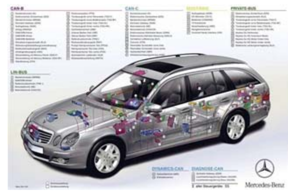
The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

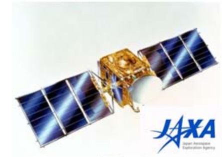
Lecture 1: Outline

- **Introduction**
  - Embedded systems
  - Design challenges
  - Formal methods and models
  - System-level design
  - Network-level design
- **Course information**
  - Topics
  - Logistics
  - Projects

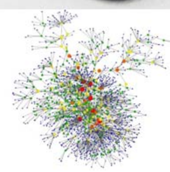
Embedded Systems

- **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, …





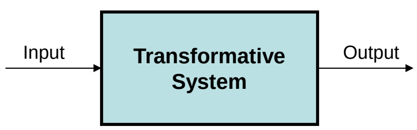
- **Ubiquitous**
  - Far bigger market than general-purpose computing (PCs, servers)
    - 98% of all processors sold [Turley02, embedded.com]



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

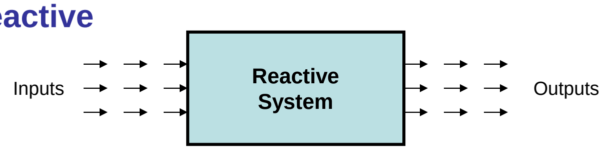
Cyber-Physical Systems (CPS)

- **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

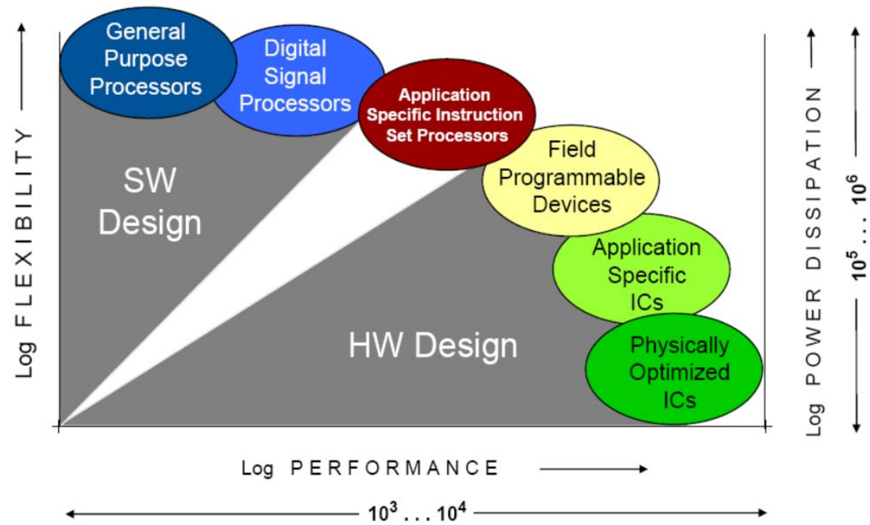
- **Correctly implement a specific set of *functions***
- **While satisfying *constraints***
  - Real-time, cost, energy, power, thermal, …

➢ **Application-specific, resource-constrained system design**
  - 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

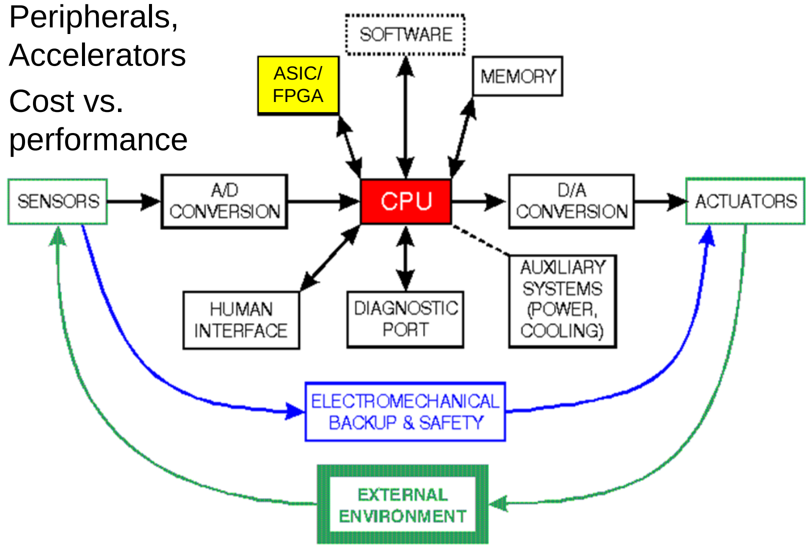
Implementation Options



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

Traditional Embedded System

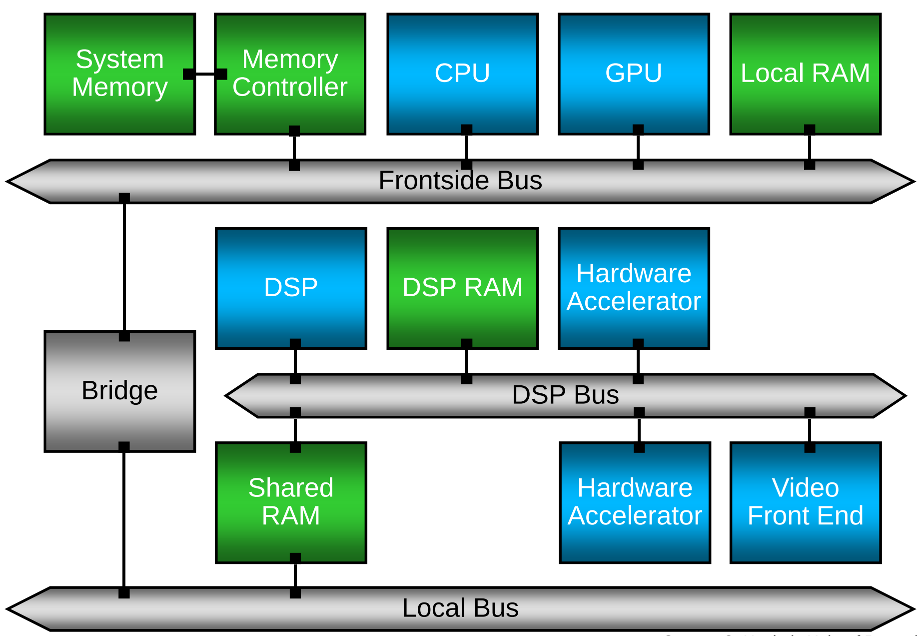
- **CPU-centric design**
  - Peripherals, Accelerators
  - Cost vs. performance



Source: M. Jacome, UT Austin

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 7

Multi-Processor System-on-Chip (MPSoC)

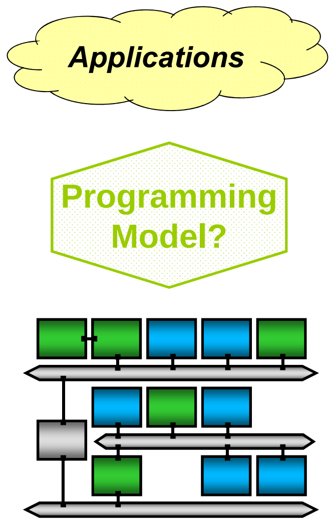


Source: C. Haubelt, Univ. of Rostock

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 8

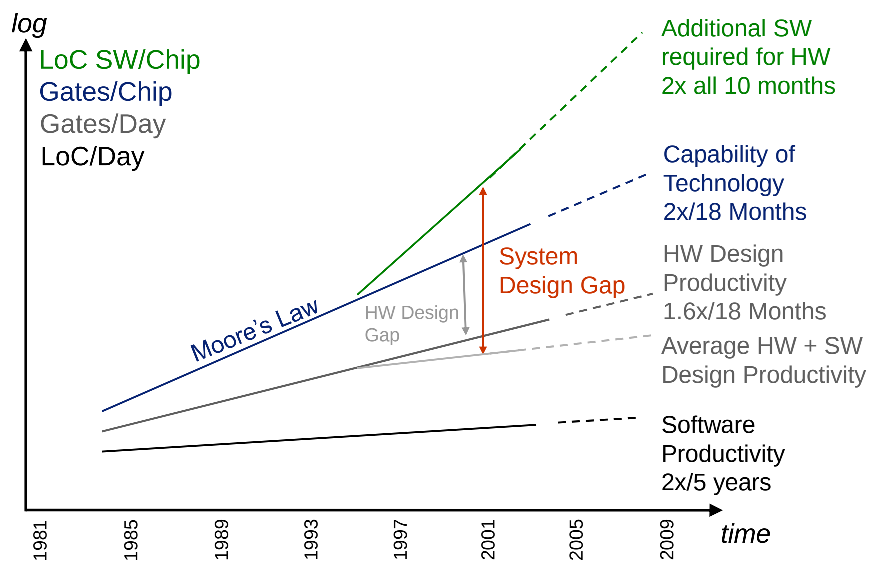
Design Challenges

- **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

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

➢ **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" [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

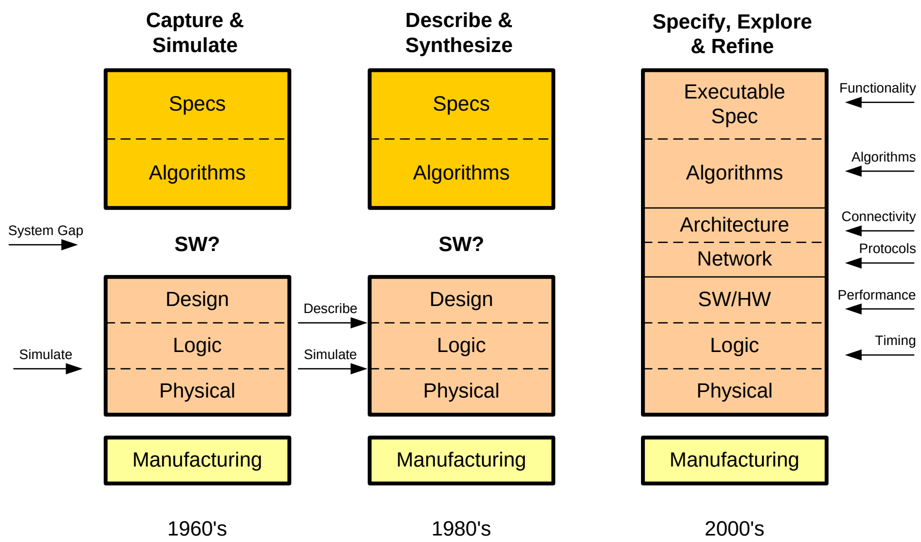
Source: E. Lee, CEDA Keynote, DAC'13.

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

Evolution of Design Flows

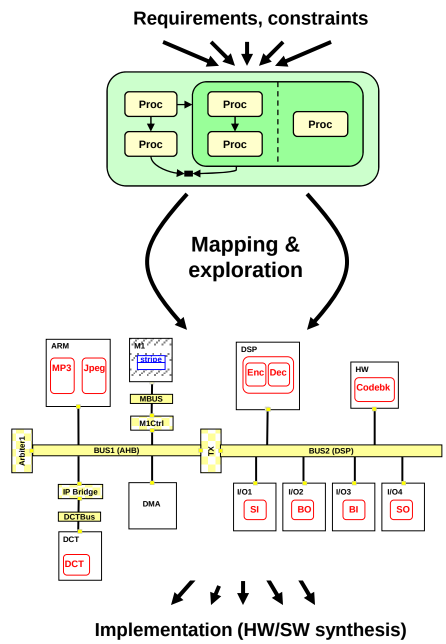


Source: D. Gajski, UC Irvine

System-Level Design

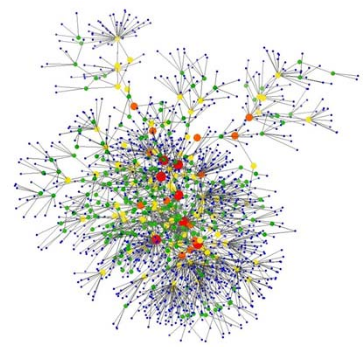
- **From system specification**
  - Functionality, behavior
    - Concurrency, order
    - Constraints
- **To system implementation**
  - MPSoC architecture
    - Spatial and temporal order
    - Components and connectivity
    - Hardware and software

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



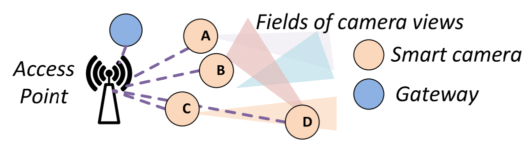
Beyond System-Level

- **Increasingly networked embedded systems (NES)**
  - Application-specific
  - Resource-constrained
  - Heterogeneous
  - Distributed



➢ **Cyber-physical systems (CPS)**
  - Real-time sensing & acting
  - Coordinated interactions

➢ **Internet-of-things (IoT)**
  - Edge computing at/near sink/source
  - Open public networks

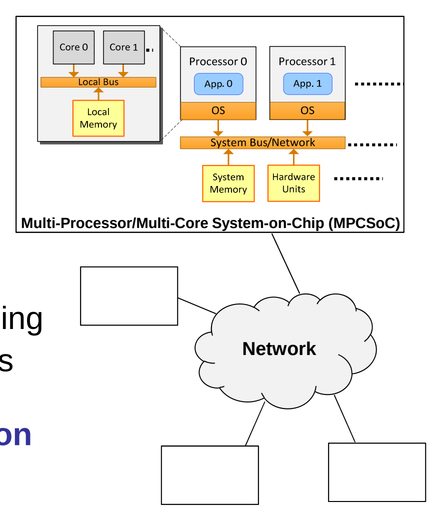


EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 17

Network-Level Design

- **Networks-of-Systems (NoS)**
  - Computation & communication
  - Network & system interactions
- **Network/system co-design**
  - Programming & mapping
  - Accelerator, fog, cloud offloading
  - Middleware & runtime systems

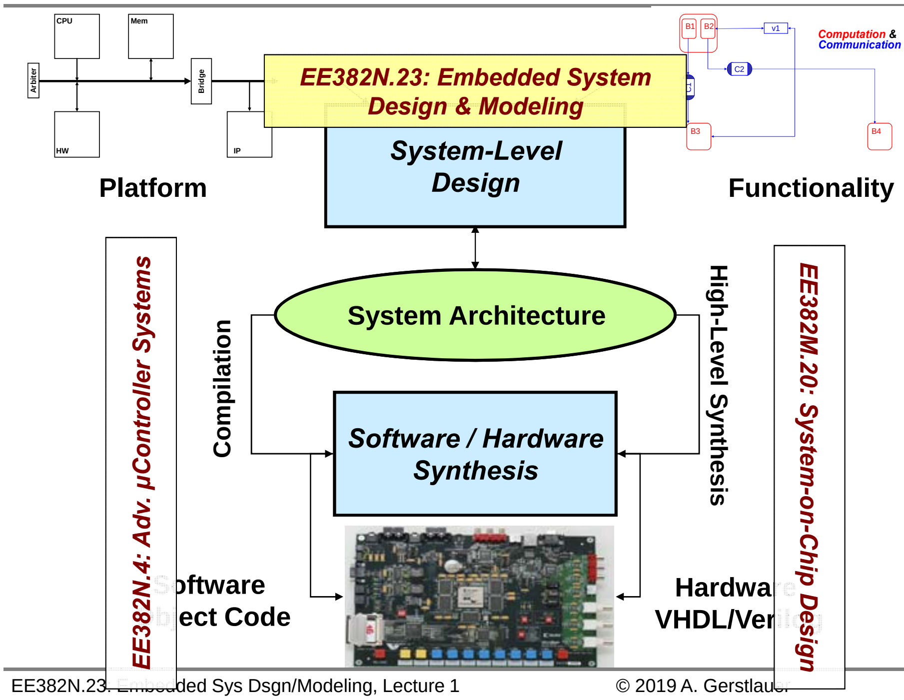
➢ **Network-level design automation**
  - Specification, analysis
  - Modeling, simulation
  - Synthesis, verification



Network-of-Systems (NoS)

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 18

UT ECE Courses



EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 19

Lecture 1: Outline

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

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 20

Course Topics

- **System-level design**

1. **Specification modeling**
   - Formal Models of Computation (MoC)
     - Parallel programming models, threads, dataflow, process networks
     - Hierarchical and concurrent finite state machine (FSM) models
2. **Implementation modeling**
   - Performance estimation and simulation (virtual prototyping)
     - Hardware/software 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

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 21

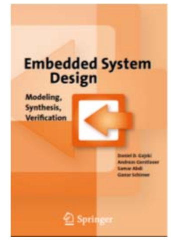
Class Administration

- **Schedule**
  - Lectures: TTh 12:30-3:00pm, ECJ 1.316
  - Midterm exam (tentative): November 14 (in class)
- **Instructor**
  - Prof. Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: EER 5.882, T 2-3pm, Th 2-4pm, or after class/by appointment
- **Teaching Assistant**
  - Kamyar Mirzazad Barijough <kammirzazad@utexas.edu>
    - Office hours: TBD
- **Information**
  - Web page: http://www.ece.utexas.edu/~gerstl/ee382n_f19
  - Announcements, assignments, grades: Canvas
  - Questions, discussions: Canvas

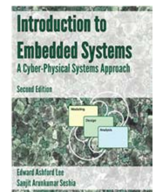
EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 1 © 2019 A. Gerstlauer 22

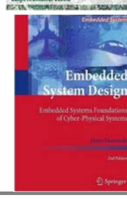
Textbooks (1)

- **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/



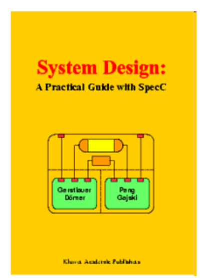
- **Additional references**
  - E. A Lee, S. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2^{nd} ed., 2017
    - Available for download at http://leeseshia.org
  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 3^{rd} ed., Springer, 2018
    - http://ls12-www.cs.tu-dortmund.de/~marwedel/es-book/

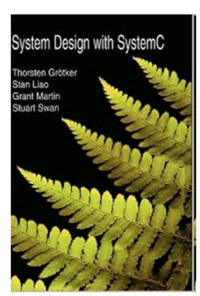




Textbooks (2)

- **Further reading**
  - A. Gerstlauer, R. Doemer, J. Peng, D. Gajski, *System Design: A Practical Guide with SpecC*, Springer, 2001 ("*yellow book*")
    - Practical, example-driven introduction to the SpecC system-level design language & methodology
  - T. Groetker, S. Liao, G. Martin, S. Swan, *System Design with SystemC*, Springer, 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

- **Homeworks and exam**
  - Cover theoretical aspects of system design
    - Specification modeling
    - Implementation modeling
    - Synthesis and exploration
  - Some practical implementation
    - Exposure to general language, modeling and optimization concepts
- **Labs**
  - Real-world IoT edge computing system design example
    - Deep learning based visual object recognition in a smart camera network
  - From specification to implementation
    - Specification modeling
    - Implementation modeling
    - Design space exploration

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, optimization
- **Project timeline (tentative)**
  - Abstract: September 30 (Canvas)
  - Proposal: October 31 (Canvas)
  - Presentations: December 3 & 5 (in class)
  - Report: Finals week (December 14)
  - ➢ Final report and presentation in publishable quality

Some Possible Projects

- **Design projects**
  - (Embedded) system design example
    - Specify, model, simulate, explore, synthesize using choice of tools
      - » Further optimize or extend deep learning example from the labs
      - » Other applications (RNNs, recommender systems), other target platforms
      - » Real-time training and learning on the edge (beyond just inference)?
- **Research projects**
  - Modeling
    - Specification modeling
      - » Develop a new/extend an existing MoC and associated analysis techniques
      - » Develop a new specification language for capturing existing/new MoC
    - Implementation 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
    - Pick an optimization/exploration problem and solve it
      - » Network-level mapping and exploration
      - » System-level allocation, partitioning, scheduling and design space exploration
      - » Hardware or software synthesis for different optimization targets
      - » Approximate computing

Successful Class Projects

- **Modeling**
  - K. Punniyamurthy, B. Boroujerdian, “GATSim: Abstract Timing Simulation of GPUs,” *DATE* 2017.
  - 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**
  - K. Mirzazad, Z. Zhao, A. Gerstlauer, "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* 2013 **(best paper candidate)**
  - J. Lin, A. Srivatsa, “Heterogeneous Multiprocessor Mapping for Real-Time Streaming Systems,” *ICASSP* 2011.

Lecture 1: Summary

✓ **Introduction**
- ✓ Embedded systems
- ✓ Design challenges
- ✓ Formal methods and models
- ✓ System-level design
- ✓ Network-level design

✓ **Course information**
- ✓ Topics
- ✓ Logistics
- ✓ Projects