

# EE382N.23: Embedded System Design and Modeling

---

## Lecture 12 – System-Level Design Tools

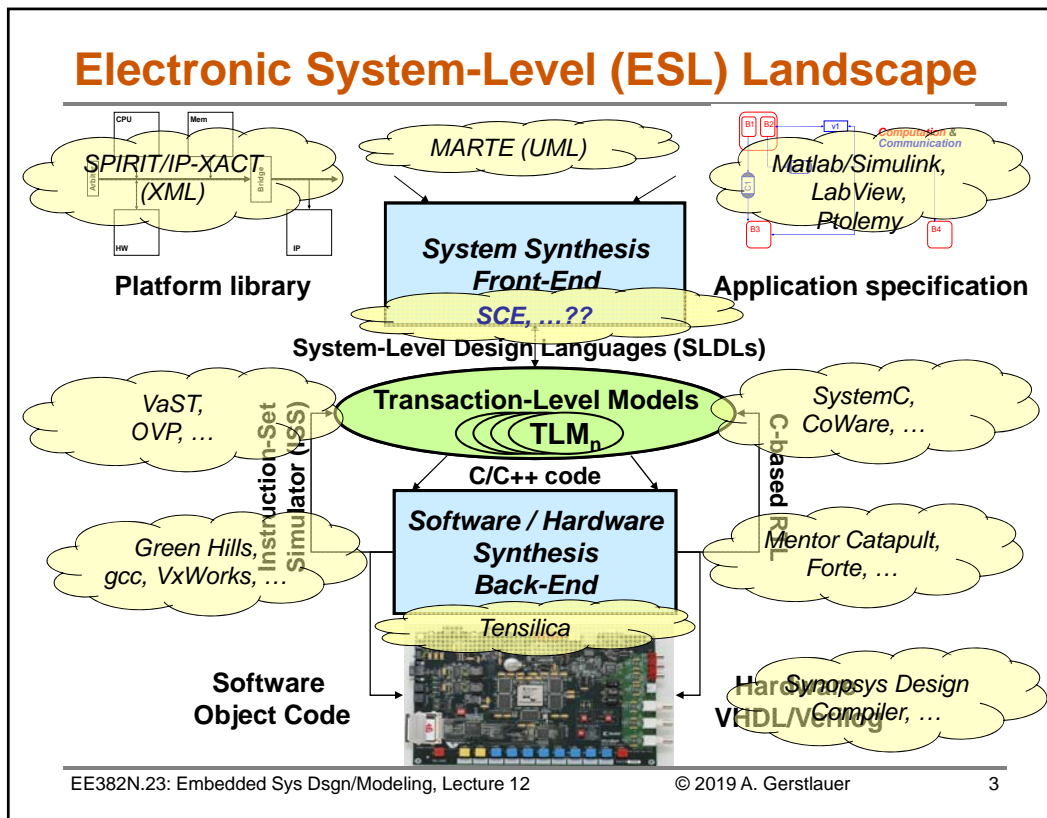
Andreas Gerstlauer  
Electrical and Computer Engineering  
University of Texas at Austin  
gerstl@ece.utexas.edu



## Lecture 12: Outline

---

- **System-level design tools**
  - Tool landscape
  - Commercial & academic tools
- **Outlook**
  - Beyond system-level design
  - Network-level design



## ESL Tools

- **Electronic System-Level (ESL) terminology misused**
  - Often single hardware unit only (high-level HW synthesis)
- **System-level has to span across hardware and software**
  - System-level frontend
  - Hardware and software synthesis backend
- **Commercial tools for modeling and simulation**
  - Algorithmic modeling (MoC) [UML, Matlab/Simulink, Labview]
  - Virtual system prototyping (TLM) [Caware, VaST, Virtutech]
  - *Only horizontal integration across models / components*
- **Academic tools for synthesis and verification**
  - MPSoC synthesis [SCE, Metropolis, SCD, PeaCE, Deadalus]
  - *Vertical integration for path to implementation*

## Commercial Tools (1)

---

- **CoFluent (now Intel)**
  - SystemC-based modeling and simulation
    - Networks of timed processes
    - Communication through queues, events, variables
  - Early, high-level interactive design space exploration
    - Graphical application, architecture and mapping capture
    - Fast TLM simulation with estimated timing
- **Space Codesign**
  - Graphical application, architecture and mapping capture (Eclipse)
    - Process network with message-passing or shared-memory channels
  - SystemC TLM simulation
    - Annotated, host-compiled or cycle-accurate ISS models
  - FPGA-based prototyping
    - Cross-compilation and third-party hardware synthesis (Forte/Catapult)

## Commercial Tools (2)

---

- **CoWare (now Synopsys)**
  - Virtual platforms
    - SystemC TLM capture, modeling and simulation
    - Extensive library of IP, processor and bus models
    - Application-specific processor ISS models (LISAtek acquisition)
  - Proprietary SystemC simulation framework
    - Optimized SystemC kernel
    - Graphical debugging, visualization and analysis capabilities
- **Soc Designer (Carbon Design Systems)**
  - Proprietary, C++ based modeling and simulation
    - Fast, statically scheduled cycle-accurate simulation
    - Special cycle-callable component models
- **VaST (now Synopsys), Simics (Virtutech, now Intel), OVP**
  - Proprietary SW-centric virtual platform modeling/simulation
    - Fast, cycle-approximate binary translated or compiled ISS + peripherals
    - SystemC wrappers

## Commercial Tools (3)

---

- **Software tools**
  - Cross-compilers, debuggers, IDEs
    - Eclipse, GreenHills, Esterel SCADE, Mathworks, UML/SysML, ...
  - Real-time operating systems
    - uCOS, VxWorks, RTLinux, ...
  - Timing analysis
    - aIT/AbsInt [Saarland Univ.], SymTA/S [Univ. Braunschweig]
- **Hardware tools**
  - High-level synthesis
    - Xilinx Vivado [UCLA], Mentor Catapult, Bluespec [MIT], ...
  - Application-specific instruction-set processor (ASIP) design
    - Tensilica Xtensa, Synopsys ASIP Designer/LISA [RWTH Aachen]

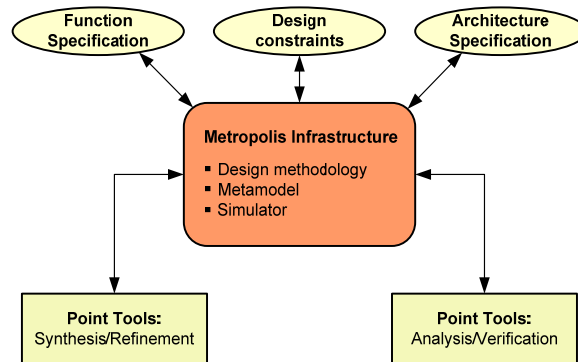
## Academic Tools

---

- **Metropolis**
  - Platform-based design (PBD)
- **SystemCoDesigner**
  - Dynamic dataflow MoC
  - Automated design space exploration
- **Daedalus**
  - KPN MoC for streaming, multi-media applications
  - IP-based MPSoC assembly
- **PeaCE**
  - “Ptolemy extension as a Codesign Environment”
  - Recent extensions for software development (HoPES)
- **SCE**
  - SpecC-based “System-on-Chip Environment”
  - Successive, stepwise Specify-Explore-Refine methodology

## Academic Tools: Metropolis

- **Platform-based**
  - Pre-defined target architecture
  - Reuse
- **Meet-in-the-middle**
  - Platform mapping and configuration
- **General, proprietary meta-modeling language**
  - Capture function, architecture and mapping
- **Modeling framework**
  - Built-in parsing and simulation
  - Back-end point tool integration



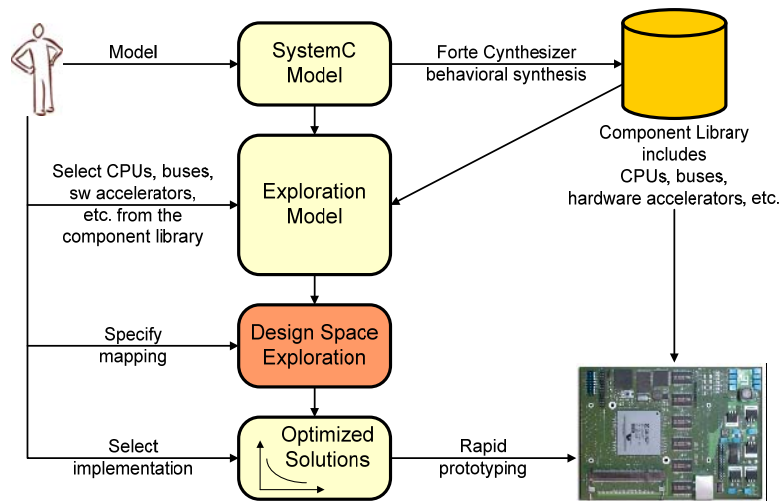
EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

9

## Academic Tools: SystemCoDesigner

- **SystemMoC input model**
  - Dynamic dataflow MoC (actors + FSMs) in SystemC
- **Fully automatic, multi-objective design space exploration**
  - Multi-objective evolutionary algorithms (MOEAs)



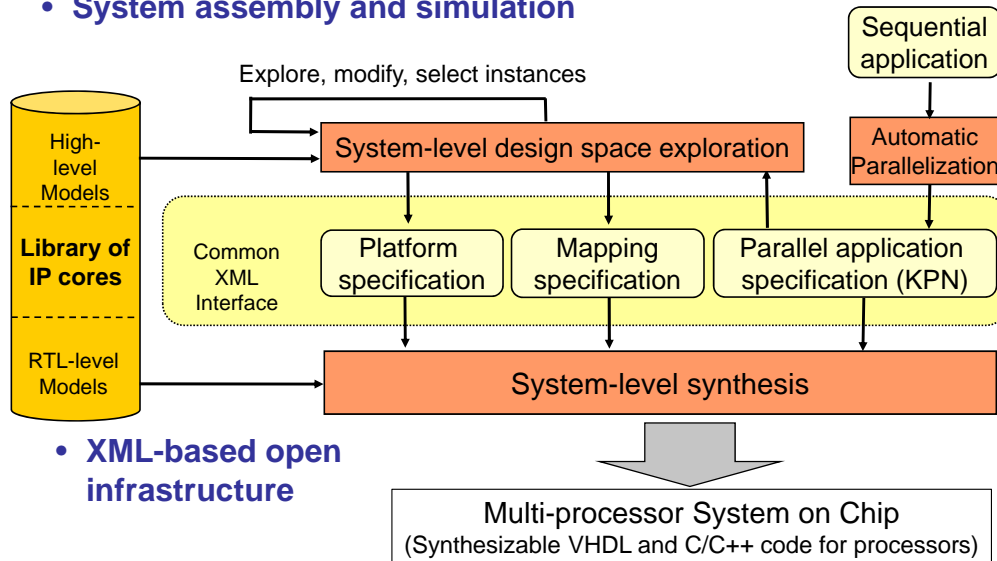
EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

10

## Academic Tools: Daedalus

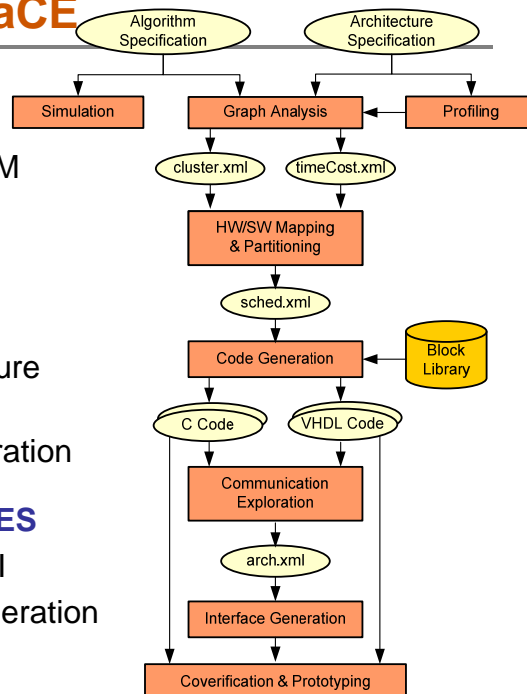
- KPN input model
- System assembly and simulation



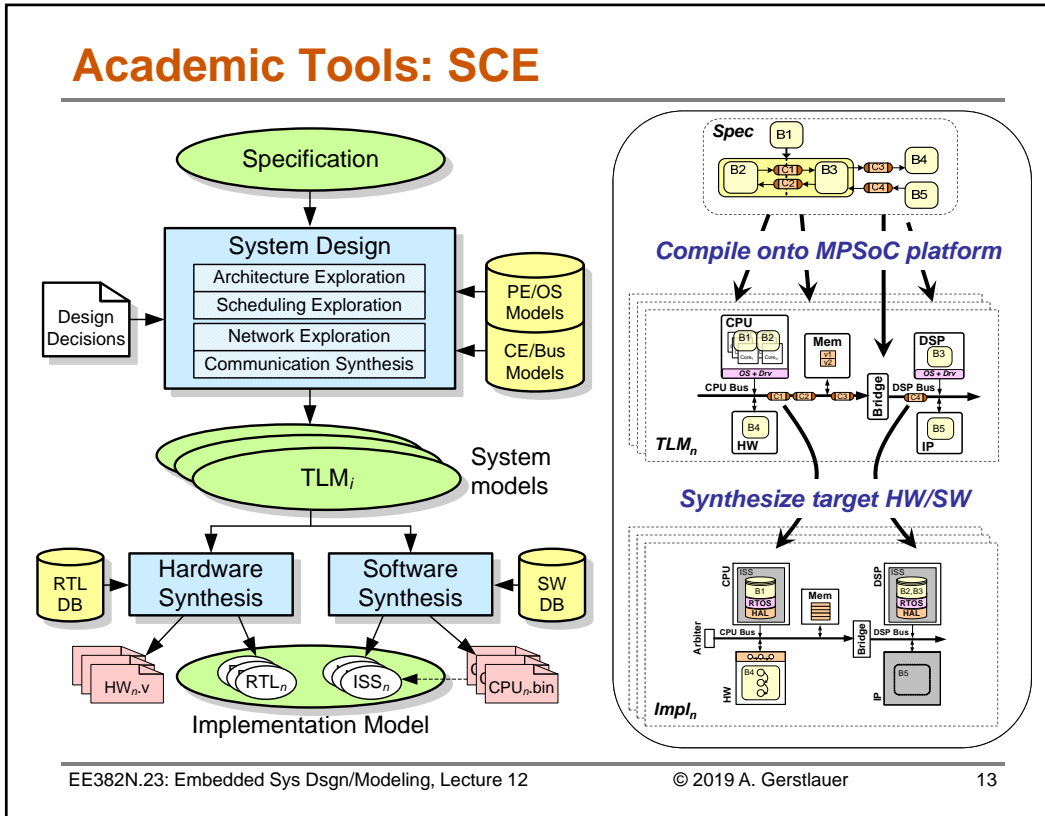
- XML-based open infrastructure

## Academic Tools: PeaCE

- Ptolemy-based
  - Heterogeneous SDF+FSM application MoC
- Stepwise flow
  - Application partitioning
  - Communication architecture exploration
  - Code and interface generation
- Software extensions: HOPES
  - Parallel programming API
  - Multi-processor code generation



### Academic Tools: SCE



EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

13

### Academic MPSoC Design Tools

Approach	DSE	Comp. decision	Comm. decision	Comp. refine	Comm. refine
Daedalus	•	•	○	•	○
Koski	•	•	○	•	○
Metropolis		○		○	
PeaCE/HoPES	○	○		•	○
SCE				•	•
SystemCoDesigner	•	•	•	•	

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

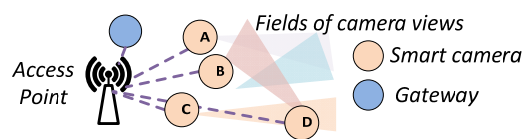
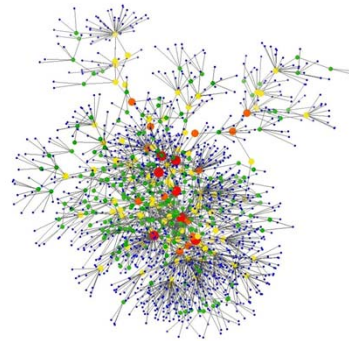
14

## Lecture 12: Outline

- **System-level design tools**
  - Tool landscape
  - Commercial & academic tools
- **Outlook**
  - Beyond system-level design
  - Network-level design

## Outlook

- **Embedded systems increasingly networked**
  - Application-specific
  - Resource-constrained
  - Heterogeneous
  - Distributed
- **Cyber-physical systems (CPS)**
  - Real-time sensing & acting
  - Interact with physical world
- **Internet-of-things (IoT)**
  - Edge computing at/near sink/source
  - Open public networks

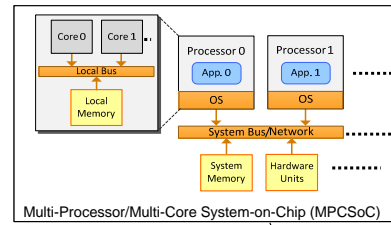




## Beyond System-Level Design

- **Networks-of-Systems (NoS)**

- Distributed computation & communication
- Network & system interactions

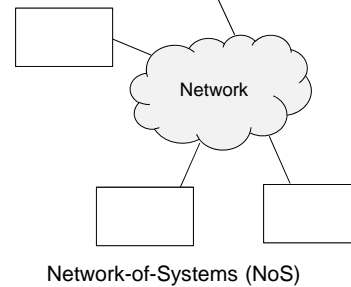


- **Network/system co-design**

- Mapping & offloading
- Middleware & runtime systems
- Network uncertainties

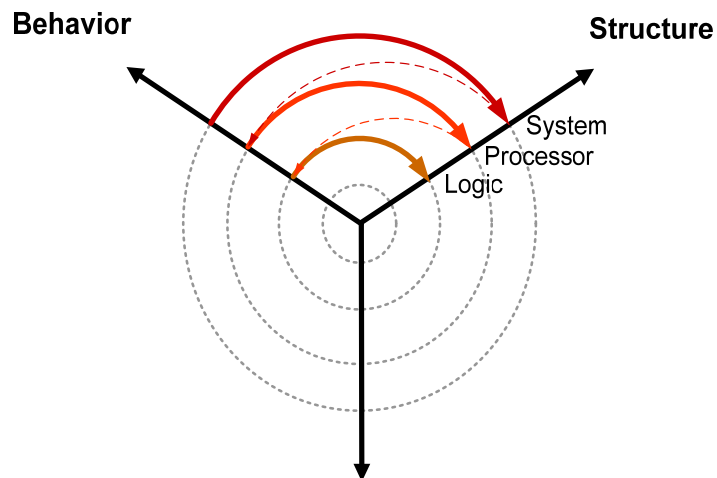
- **Network-level design automation**

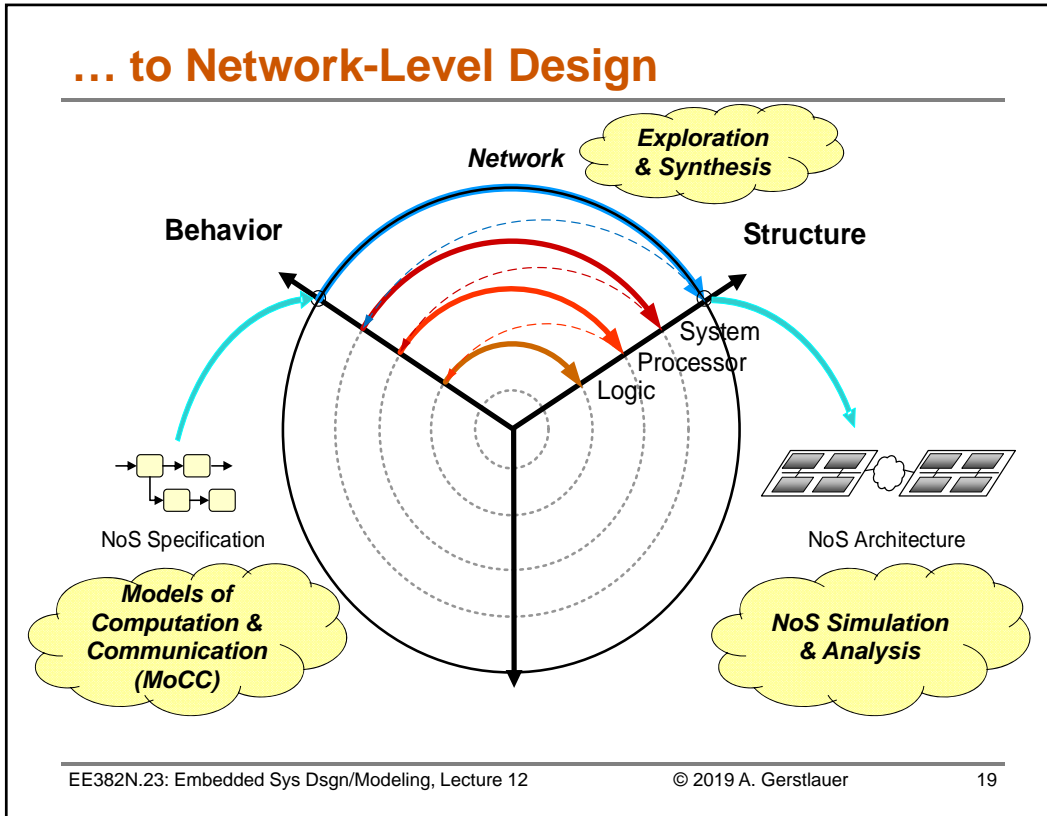
- Specification, analysis
- Modeling, simulation
- Synthesis, verification



## From System-Level Design...

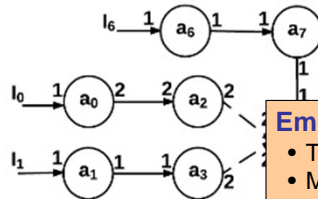
- **Gajski's Y-Chart**





## Reactive and Adaptive Data Flow (RADF)

- **Dataflow basis**
  - Streaming applications, e.g. based on SDF
- **Extended by two channel types**
  - Lossless (solid) and lossy (dashed) channels



**Empty tokens**

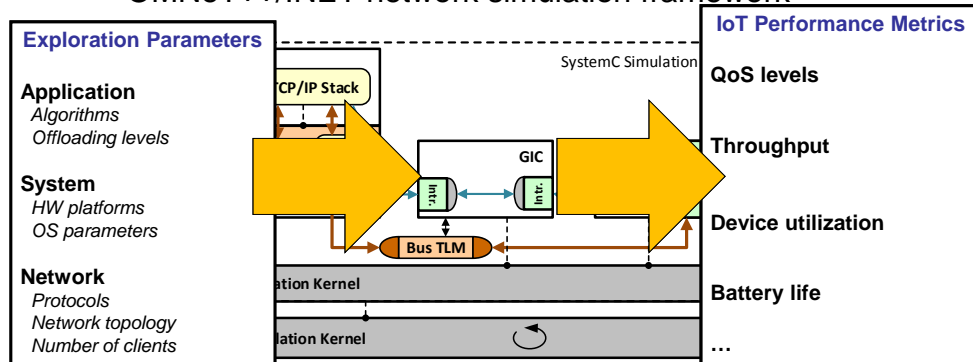
- Tokens that do not carry useful data
- Model network losses
- Preserve token order determinism

- **Lossy channels**
  - May replace tokens with empty ones:  $[* \dots *] \rightarrow [* \dots \emptyset \dots *]$
- **Actor variants**
  - Based on firing rules of empty/non-empty tokens

Source: S. Francis, A. Gerstlauer, "A Reactive and Adaptive Data Flow Model For Network-of-System Specification," IEEE ESL, 9(4), 2017.

## NoSSim

- **System simulation model**
  - SystemC-based host-compiled device model
  - Capture system-wide interactions between application, OS and underlying hardware components
- **Network simulation backplane**
  - OMNeT++/INET network simulation framework



Source: Z.Zhao, V. Tsoutsouras, D. Soudris, A. Gerstlauer, "Network/System Co-Simulation for Design Space Exploration of IoT Applications," SAMOS'17.

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

21

## Lecture 12: Summary

- **System-level design tools**
  - Commercial focus still only on modeling and simulation
  - Academic approaches towards true system-level design
  - Emerging commercial backend HW/SW synthesis
  - Complete, automated system design flow
    - From specification to implementation
- **Network-level design**
  - Beyond system-level design
  - Distributed and networked embedded systems
  - Network uncertainties

EE382N.23: Embedded Sys Dsgn/Modeling, Lecture 12

© 2019 A. Gerstlauer

22