

Raising the Level of Abstraction: A Signal Processing System Design Course

Brian L. Evans and Guner Arslan



**Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712-1084**

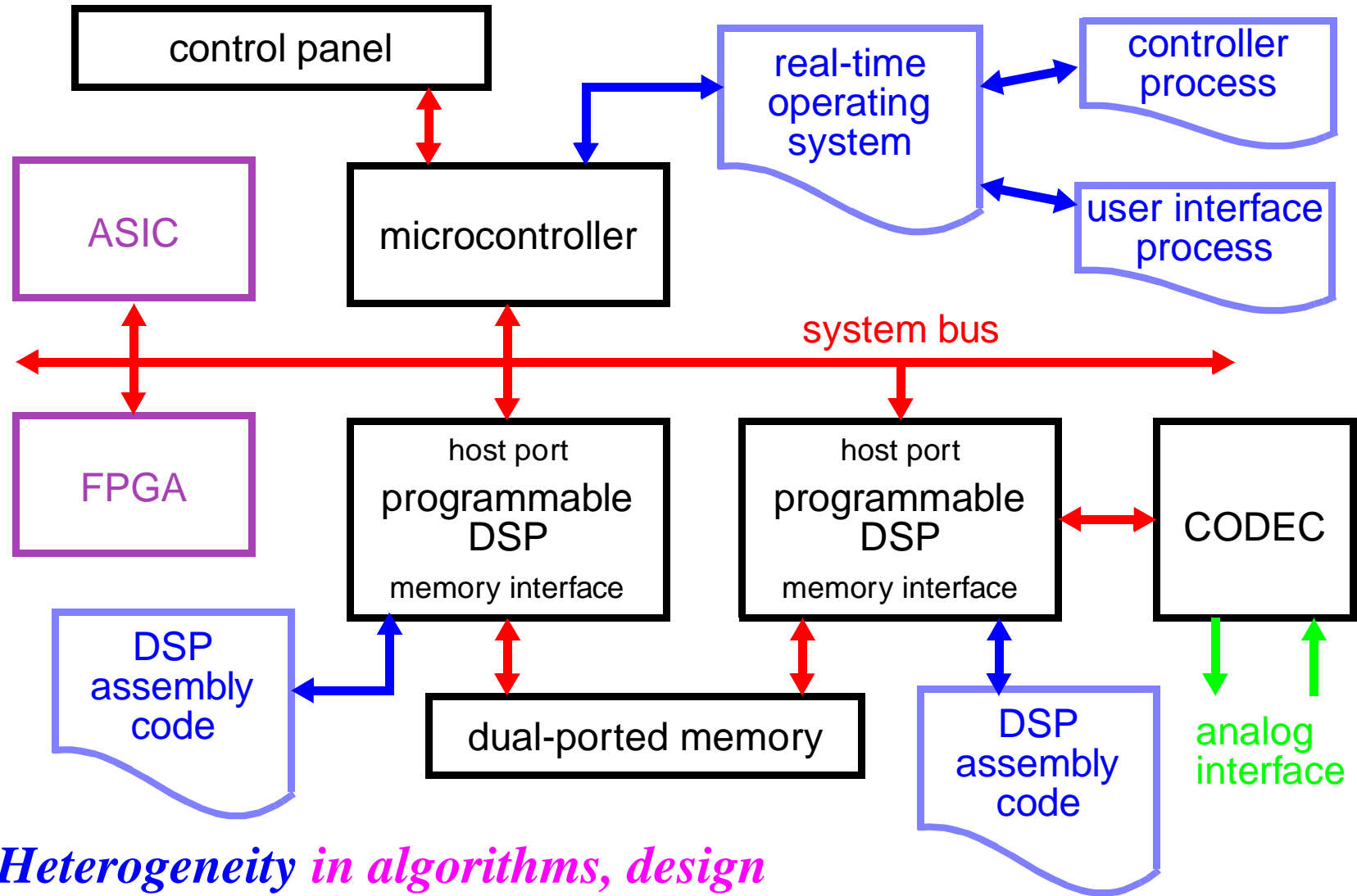


<http://www.ece.utexas.edu/~bevans/courses/ee382c/>

Introduction

- **New standard for speech/audio compression, image/video compression and mobile communication each year**
- **Growth and diversity of implementation technologies**
 - **Dedicated and configurable hardware**
 - **Dozens of high-volume programmable processors introduced each year (general-purpose processors, digital signal processors, microcontrollers)**
 - **Revolution in high-level languages every 10 years (Fortran, C, C++, Java)**
- **Increasing complexity in implementation technologies**
 - **Moore's Law: Number of transistors on a chip doubles every 18 months**
 - **Networked, distributed, and multiprocessor systems**
- **Signal processing system design course raises abstraction by decoupling system specification from its implementation**
 - **Enables mapping of the same subsystem onto a variety of implementation technologies**
 - **Reuse of existing designs**

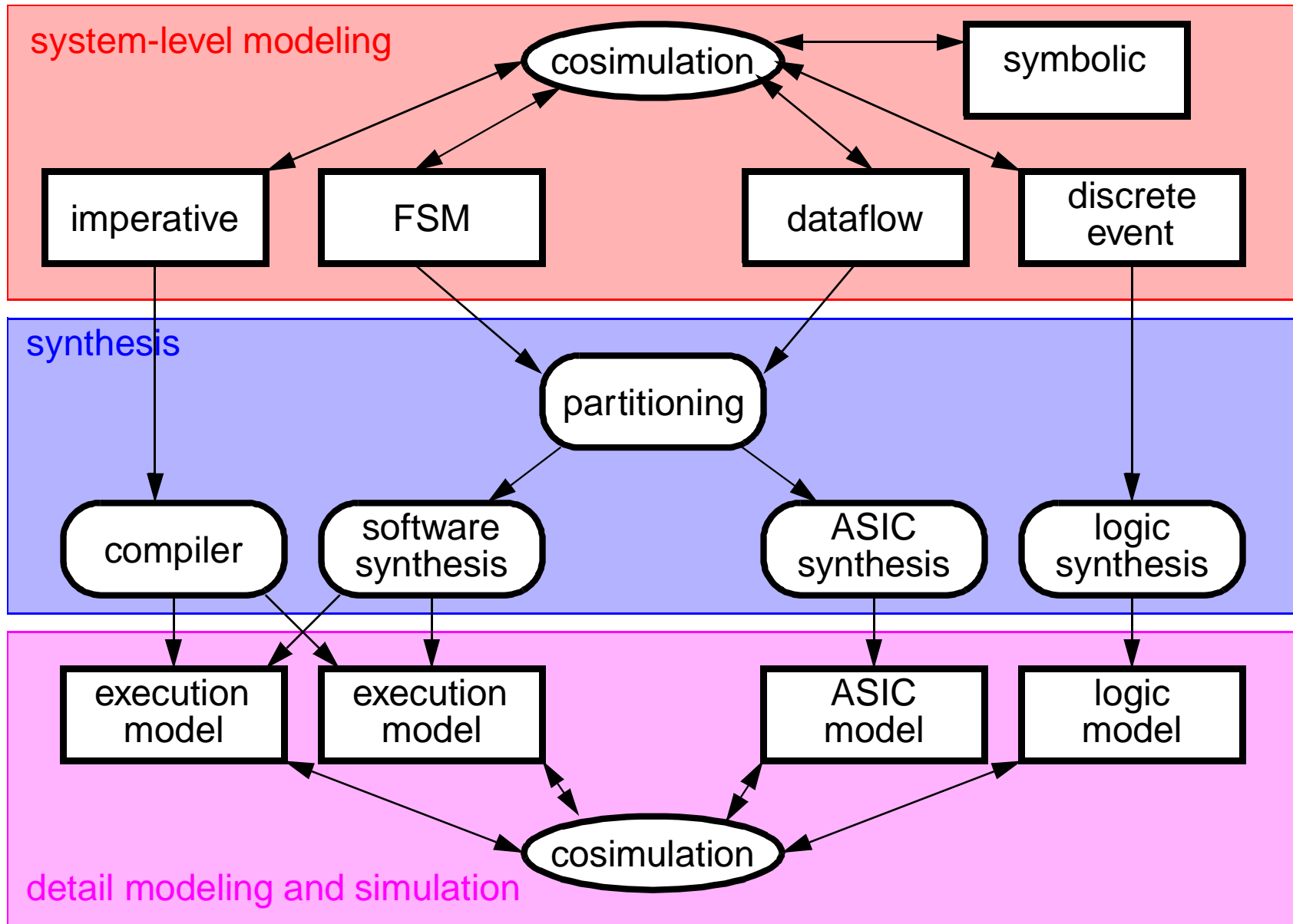
Embedded Signal Processing Systems



Heterogeneity in algorithms, design methods, and implementation technologies

Slide by Prof. Edward A. Lee, UC Berkeley. Used by permission

Heterogeneity in a System-Level Design Flow



Slide by Prof. Edward A. Lee, UC Berkeley. Used by permission

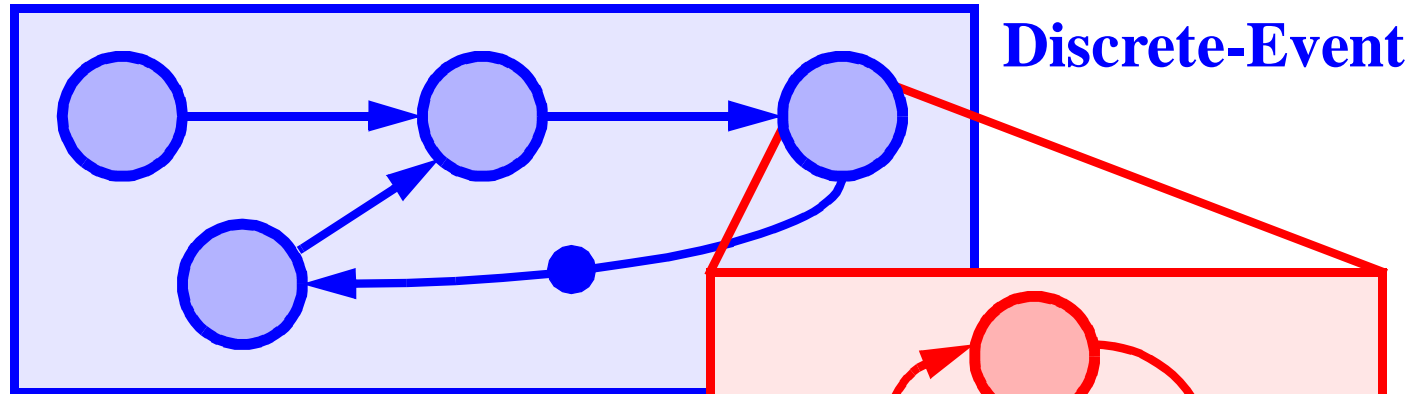
System Modeling

- **Models of computation**
 - Coordinate computation of and communication between subsystems
 - Ideally unbiased towards implementation in software and hardware
 - Map be mapped onto a variety of implementation technologies
- **Modeling signal processing subsystems**

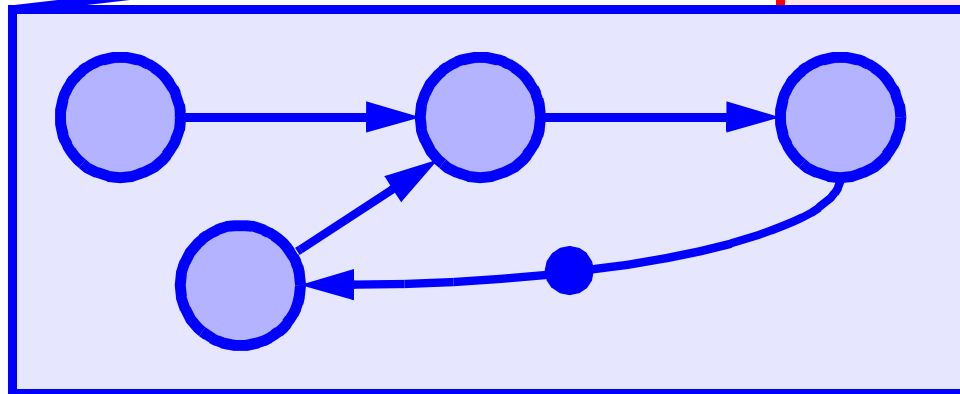
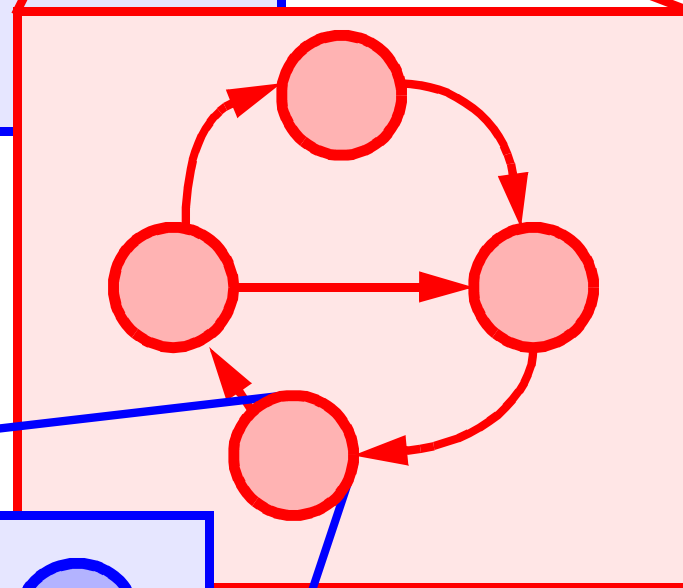
<i>Subsystem</i>	<i>Model of Computation</i>
speech/audio processing	1-D dataflow
image processing	1-D/2-D dataflow
image/video resampling	m-D multirate dataflow
user interface	synchronous/reactive
communication protocols	finite state machine
digital control	dataflow
scalable descriptions	process networks

- **Hierarchical combination forms heterogeneous systems: models must compose with each other**

Specification Using Hierarchical Block Diagrams



- Attach meaning to graphs
- Example: digital cellular phone design



**Finite State
Machine**

Dataflow

System Simulation and Synthesis

- **Two sides of the same coin**
 - *Simulation*: scheduling then execution on desktop computer(s)
 - *Synthesis*: scheduling then code generation in C++, C, assembly, VHDL, etc.
- **Validation by simulation important throughout design flow**
- **Models of computation enable**
 - Global optimization of computation and communication
 - Scheduling and communication that is *correct by construction*

<i>Model of Computation</i>	<i>Global State</i>	<i>Type of Comm.</i>	<i>Type of Scheduling</i>	<i>Optimal Scheduling</i>	<i>Simulation Speed</i>
synchronous dataflow	finite	asynch	static	n^3	fast
Boolean dataflow	infinite	asynch	quasi-static	infinite	fast
process networks	infinite	asynch	dynamic	infinite	fast
finite state machine	finite	either	static	not poly.	fast
synchronous/reactiv	finite	synch	static	n^2	fast
discrete event	infinite	synch	dynamic	infinite	slow

Educational Objectives

- **Design space (global state)**
 - *Finite*, e.g. finite state machine, synchronous dataflow, synchronous/reactive
 - *Infinite*, e.g. imperative programming, Boolean dataflow, process networks
- **Worst-case optimal scheduling time**
 - Finite time if design space is finite, and infinite time if design space is infinite
 - Infinite-time off-line scheduling is impractical: use heuristics (e.g. compilers)
 - Process networks on-line scheduling takes infinite time but bounded memory
- **Use models of computation with finite state when possible**
 - Enables formal analysis (consistency, deadlock, boundedness, verification)
 - Suitable for fixed topologies (VLSI and embedded software implementations)
- **Example: model signal processing in a sonar beamformer**
 - Statically schedule a synchronous dataflow model of the processing onto a fixed number of processors, e.g. on a workstation or embedded processors
 - Dynamically schedule a process networks model for scalable software
 - Both approaches lead to real-time solutions

Conclusion

- **Traditional signal processing course**
 - One style of algorithm (e.g. speech/audio or image/video)
 - One implementation technology (e.g. Matlab, C, or digital signal processor)
- **Signal processing system design course raises abstraction by decoupling system specification from its implementation**
 - Implementation-unbiased models of computation
 - Compose models to specify complex heterogeneous systems
 - Simulate and synthesize heterogeneous systems
- **Students use and extend system-level CAD tools**
 - *UC Berkeley Ptolemy*: dataflow, FSM, discrete event, synchronous/reactive models plus synthesis in C, assembly, and VHDL
 - *HP EEsof Advanced Design System*: mixed analog, RF, and digital design for wireless communication systems using Spice, harmonic balance, and dataflow modeling
- **All course notes, handouts, and lectures are on the Web**

<http://www.ece.utexas.edu/~bevans/courses/ee382c/>