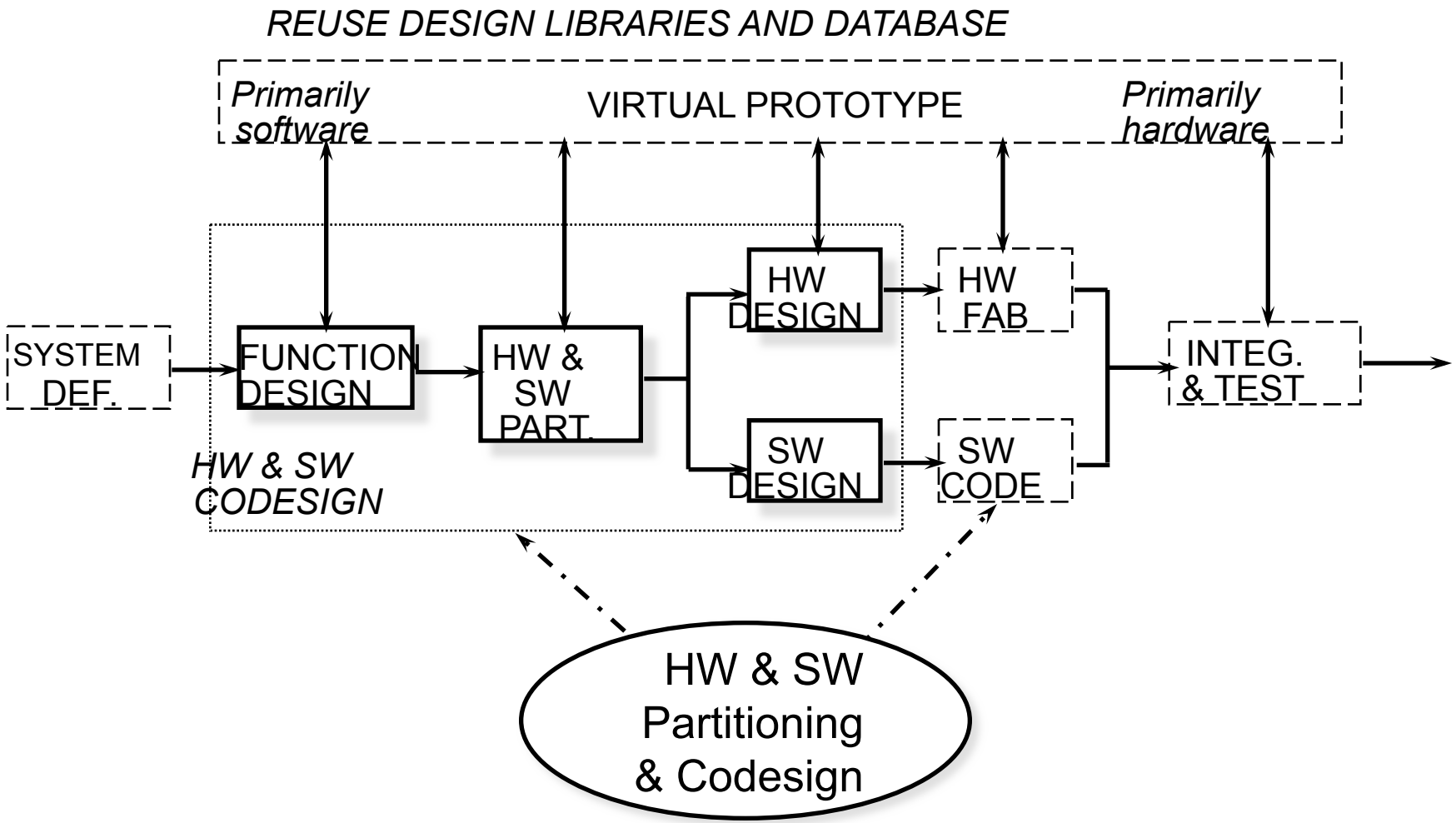
EEL 5722C
Field-Programmable Gate Array Design

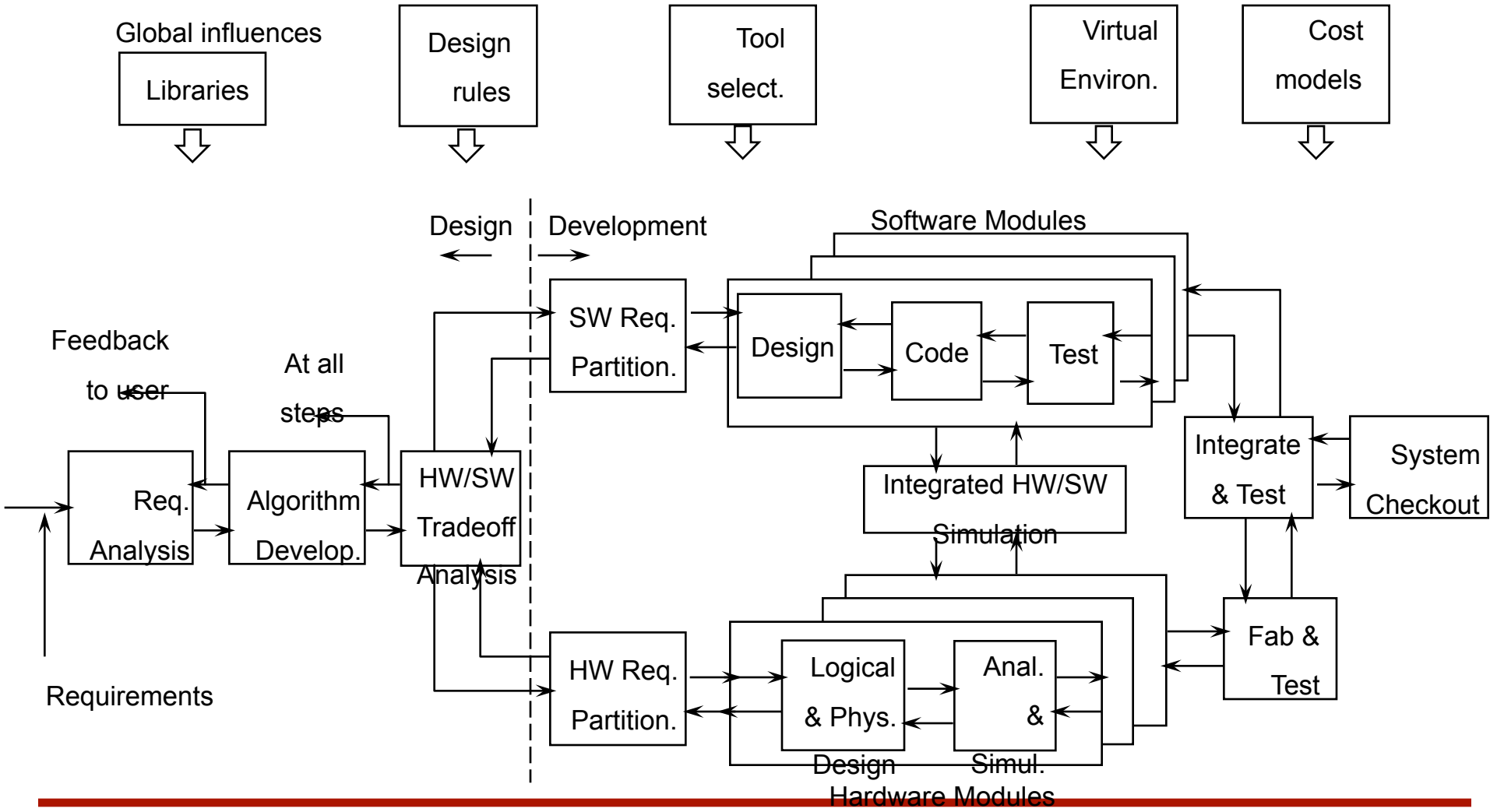
Lecture 22: HW/SW Codesign:
Industry Practice and Academic Research*

Prof. Mingjie Lin

Rapid Prototyping Design Process

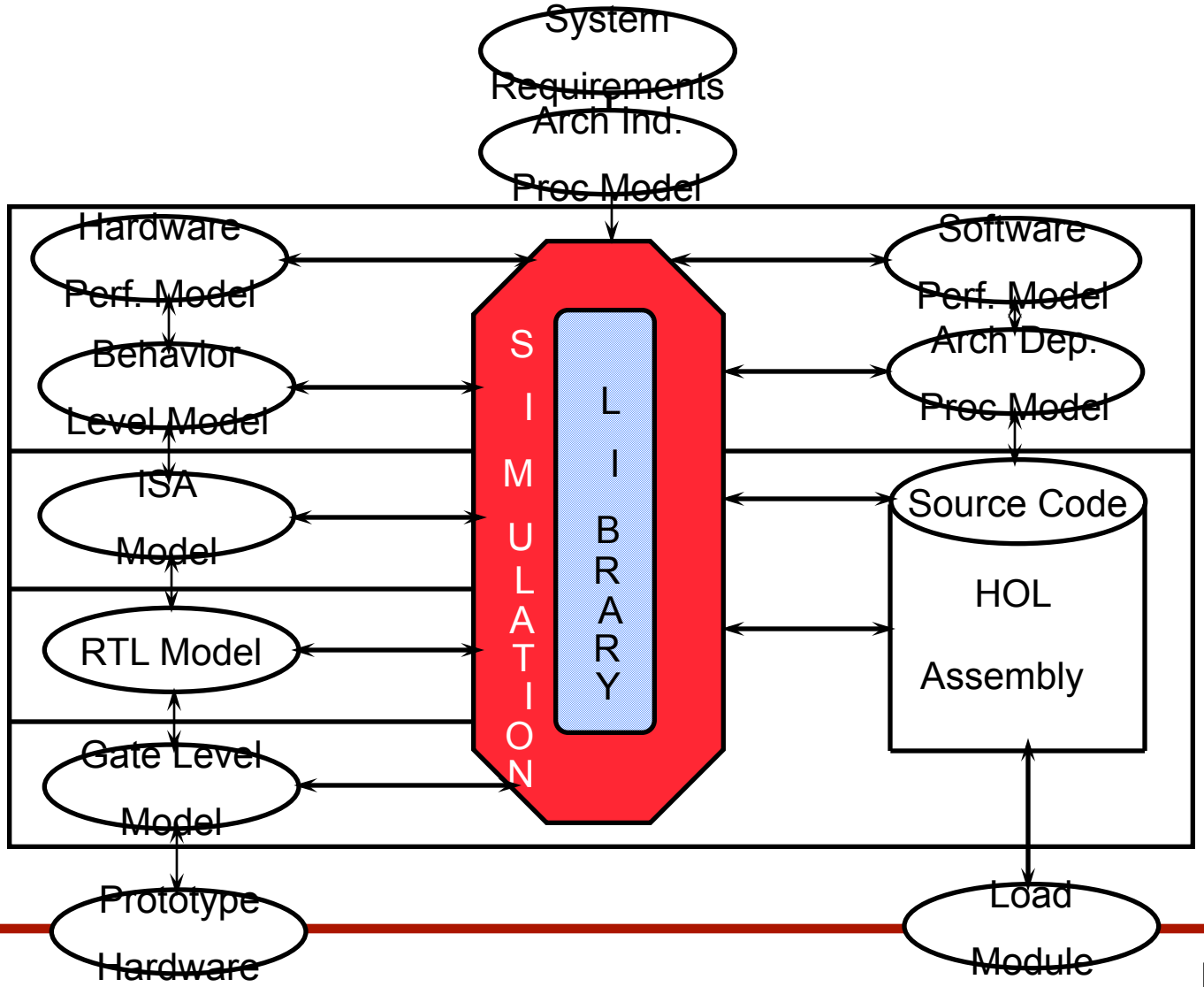


Sanders Codesign Methodology



[HOOD94]

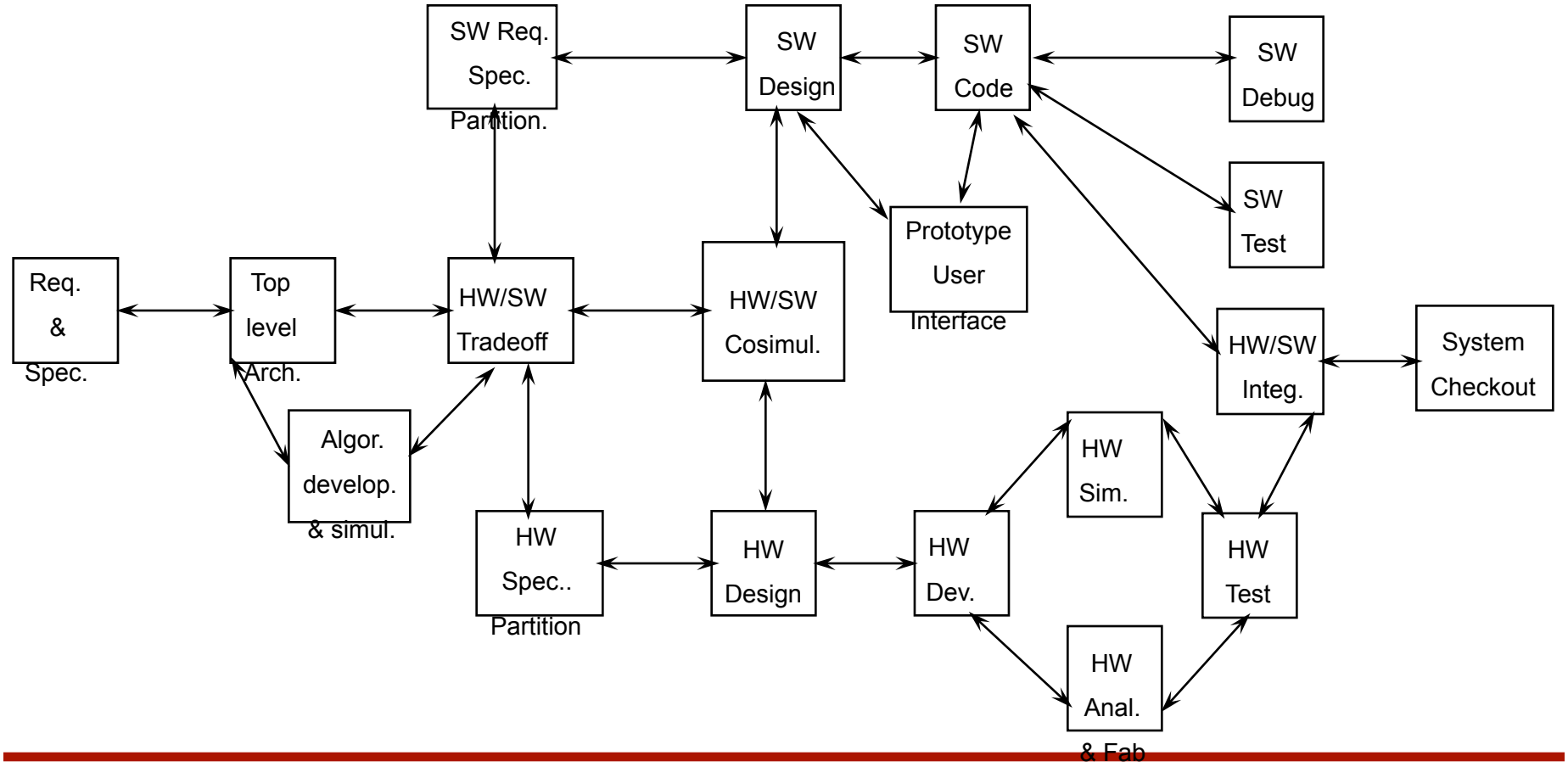
Sanders Codesign Methodology
Integrated Modeling Substrate



Sanders Codesign Methodology

- Subsystems process
  - Processing requirements are modeled in an architecture-independent manner
  - Codesign not an issue
- Architecture process
  - HW/SW allocation analyzed via modeling of SW performance on candidate architectures
  - Hierarchical verification is performed using finer grain modeling (ISA and below)
- Detailed design
  - Downloadable executable application and test code is verified to maximum extent possible
- Library support
  - SW models validated on test data
  - HW models validated using existing SW models
  - HW & SW models jointed iterated throughout designs

Lockheed Martin ATL Codesign Methodology



[RASSP94]

Major Codesign Research Efforts

- Chinook - University of Washington - Chou, Ortega, Borriello
- Cosmos - Grenoble University - Ismail, Jerraya
- Cosyma - University of Braunschweig - Ernst, Henkel, Benner
- Polis - U. C. Berkeley - Chiodo, Giusto, Jurecska, Hsieh, Lavagno, Sangiovanni-Vincentelli
- Ptolemy - U. C. Berkeley - Kalavade, Lee
- Siera- U. C. Berkeley - Srivastava, Broderson

Chinook

- Unified representation: Event Graph (CDFG)
- Partitioning: constraint driven by scheduling requirements
- Scheduling: timing driven
- Modeling substrate: based on Verilog HDL
- Validation: simulation based (Verilog)
- Main emphasis on synthesis of hardware/software *interfaces*

Cosmos

- Unified representation: Initial description is done in SDL (specification description language) which is translated into SOLAR, an intermediate form that allows several description levels (CSPs, FSMs, etc.)
- Partitioning: user driven using a tool that allows processes to be grouped together or split into sub-processes
- Scheduling: based on the partitioning
- Modeling substrate: VHDL simulation after architecture mapping
- Validation: simulation based
- Main emphasis on synthesis of communications mechanisms between processes - reuse of existing communication models

Cosyma

- Unified representation: ES graph (CDFG)
- Partitioning: combined method based on course partitioning by user with cost guidance and finer scheduling done by simulated annealing
- Scheduling: no specific method
- Modeling substrate: based on C++
- Validation: simulation based (C++)
- Main emphasis on partitioning for hardware accelerators

Polis

- Unified representation: Codesign Finite State Machine (CFSM) based
- Partitioning: user driven with cost estimated provided by co-simulation
- Scheduling: classical real-time algorithms
- Modeling substrate: Ptolemy based (C++)
- Validation: co-simulation and formal FSM verification
- Main emphasis on verifiable specification not biased to either hardware or software implementation

Ptolemy

- Unified representation: Data Flow Graph
- Partitioning: greedy algorithm based on scheduling constraints
- Scheduling: linear based on sorting blocks by “criticality”
- Modeling substrate: heterogeneous modeling and simulation framework based on C++
- Validation: based on simulation
- Main emphasis on heterogeneous modeling framework (mixing different models of computation)

Siera

- Unified representation: static, hierarchical network of concurrent sequential processes communicating via message queues (similar to DFG)
- Partitioning: manual user driven
- Scheduling: static process to processor mapping, priority based preemptive schedulers available within real-time OS on processors
- Modeling substrate: based on VHDL - includes support for modeling continuous time systems such as sensors and actuators
- Validation: based on simulation
- Main emphasis on the design of embedded systems targeted towards a predefined architectural template

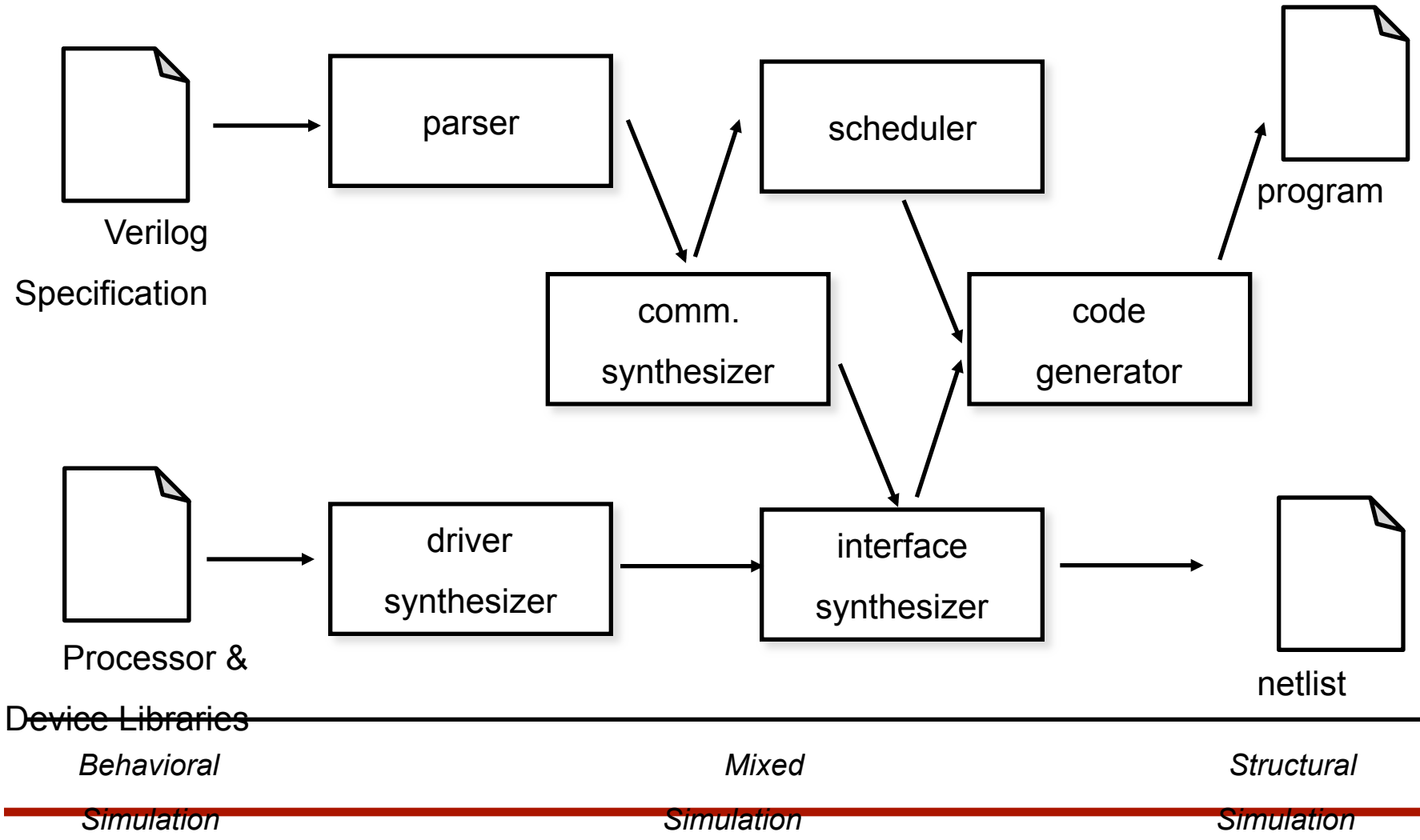
Chinook

- Hardware/Software Co-synthesis system developed at the University of Washington
- Targeted at real-time reactive embedded systems
- Control dominated designs constructed from off-the-shelf components

Chinook's Principal Innovations

- Single Specification - one specification, with explicit timing/ performance constraints is used for the system's hardware and software
- One Simulation Environment - the high level specification, the final result, and any intermediate steps can be simulated to verify and debug the design
- Software Scheduling - the appropriate software architecture is synthesized to meet the timing requirements
- Interface Synthesis - the hardware and software necessary to interface between system components (glue logic and device drivers) is automatically synthesized
- Complete Information for Physical Prototyping - a complete netlist is generated for the hardware, and C source code is generated for the software

The Chinook System



System Specification in Chinook
(Unified Representation)

- The system specification is written in a dialect of Verilog and includes the system's behavior and the structure of the system architecture
- The behavior is specified as a set of tasks in a style similar to communicating finite state machines - control states of the system are organized as *modes* which are behavioral regimes similar to hierarchical states
- In a given mode, the system's responses are defined by a set of *handlers* which are essentially event-triggered routines
- The designer must *tag* tasks or modules with the processor that is preferred for their implementation - untagged tasks are implemented in software
- The designer can specify response times and rate constraints for tasks in the input description

Scheduling in Chinook

- Chinook provides an automated scheduling algorithm
- Low-level I/O routines and high level routines grouped in modes are scheduled statically
- A static, nonpreemptive scheduling algorithm is used to meet min/max timing constraints on low-level operations
  - Determines serial ordering for operations
  - Inserts delays as necessary to meet minimum constraints
  - Includes heuristics in the scheduling algorithm to help exact algorithm generate valid solution to NP-hard scheduling problem
- A customized dynamic scheduler may be generated for the top-level modes

Interface Synthesis in Chinook

- Realization of communication between system components is an area of emphasis in the Chinook system
- Chinook synthesizes device drivers from timing diagrams
- Custom code for the processor being used is generated
  - For processors with I/O ports, an efficient heuristic is used to connect devices with minimal interface hardware
  - For processors w/o I/O ports, a memory mapped I/O interface is generated including allocating address spaces, and generating the required bus logic and instructions
- Portions of the interface that cannot be implemented in software are synthesized into external hardware

Communications Synthesis and System Simulation in Chinook

- Chinook provides methods for synthesizing communications systems between multiple processors if a multicomputer implementation is chosen
  - Bus-based, point-to-point, and hybrid communications schemes are supported
  - Communications library that includes FIFOs, arbiters, and interconnect templates is provided

- Simulation of the design at different levels of detail is supported
  - Verilog-XL Programming Language is used
  - Verilog PLI is used to interface to device models written in C
  - Each device supports the same API for simulation and synthesis - API calls can be used by the designer to animate the model interactively
  - RTL level models of the processors are used to simulate the final implementation of the system (software)

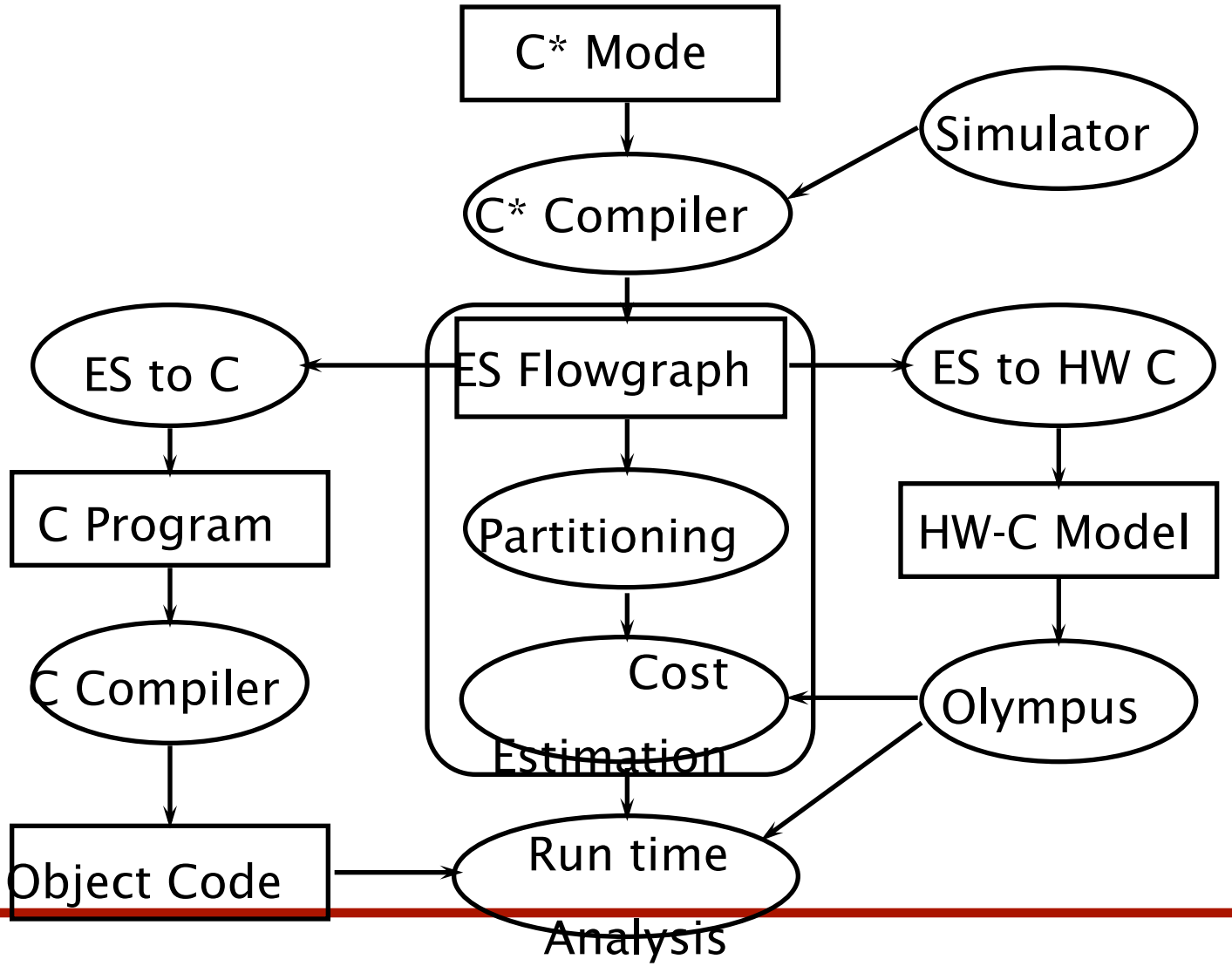
Cosynthesis of Embedded Applications (COSYMA)

- Developed at the Technical University of Braunschweig, Germany
- An experimental system for HW/SW codesign of small embedded real time systems
  - Implements as many operations as possible in software running on a processor core
  - Generates external hardware only when timing constraints are violated
- Target architecture:
  - Standard RISC processor core
  - Application-specific processor
- Communication between HW and SW through shared memory with a communicating sequential processes (CSP) type protocol

COSYMA (Cont.)

- Input description of system in C* is translated into an internal graph representation supporting
  - Partitioning
  - Generating hardware descriptions for parts moved to hardware
- Internal graph representation combines
  - Control and dataflow graph
  - Extended syntax (ES) graph
    - Syntax graph
    - Symbol table
    - Local data/control dependencies

Design Flow in a COSYMA System



COSYMA - Aims and Strategies

- Major aim is automating HW/SW partitioning process, for which very few tools currently exist
- COSYMA partitions at the basic block and function level (including hierarchical function calls)
  - Simulated annealing algorithm is used because of its flexibility in the cost function and the possibility to trade-off computation time vs result quality
  - Starts with an unfeasible all-software solution

COSYMA - Cost Function and Metrics

- The cost function is defined to force the annealing to reach a feasible solution before other optimization goals (e.g., area)
- The metrics used in cost computation are:
  - Expected hardware execution times
  - Software execution times
  - Communication
  - Hardware costs
- The cost function is updated in each step of the simulated annealing algorithm

COSYMA - Cost Function and Metrics (Cont.)

- After partitioning, the parts selected to be realized in software are translated to a C program, thereby inserting code for communicating with the coprocessor
- The rest of the system is translated to the input description of the high-level synthesis system, and an application-specific coprocessor is synthesized
- Lastly, a fast-timing analysis of the whole HW/SW system is performed to test whether all constraints are satisfied

Ptolemy

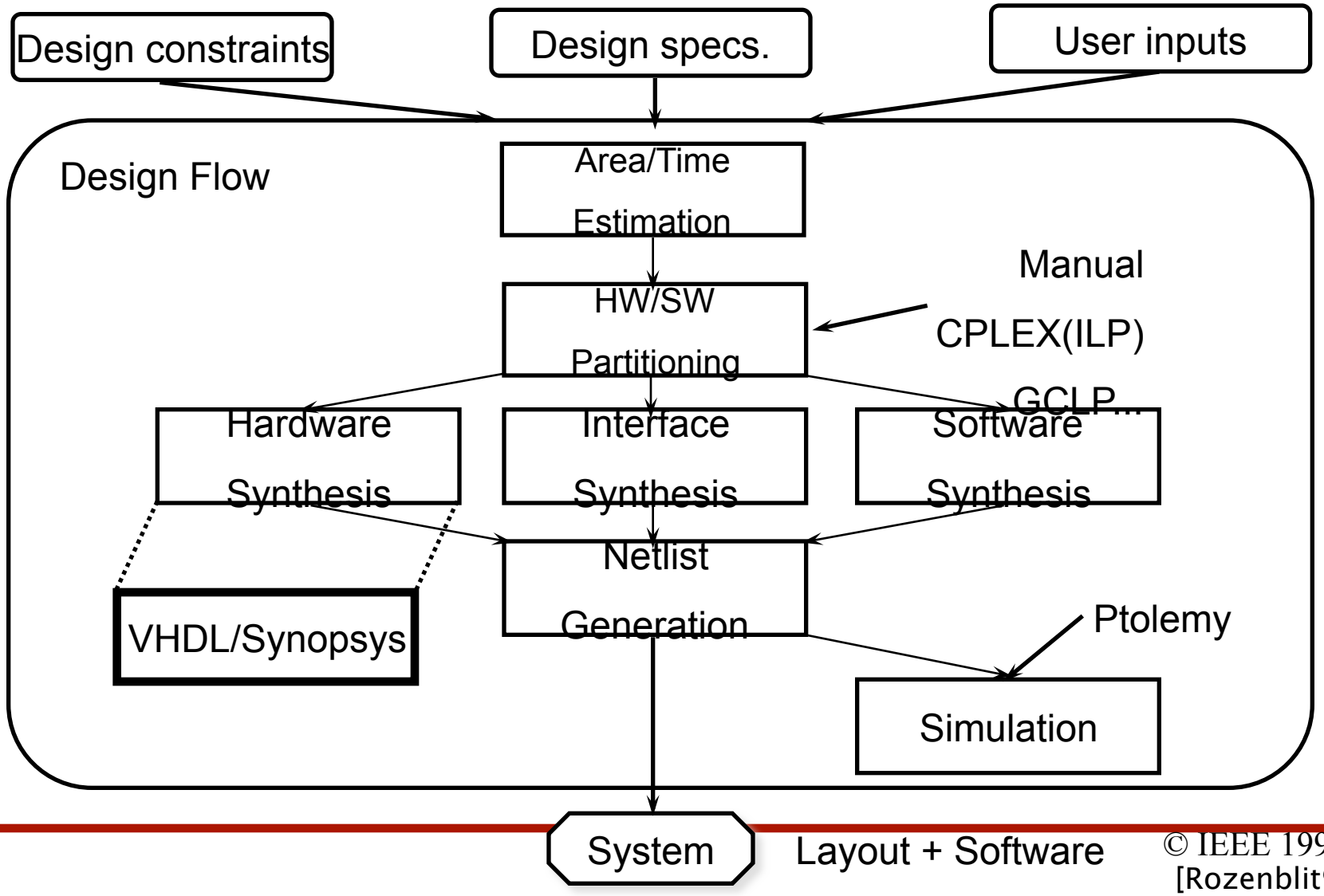
- A software environment for simulation and prototyping of heterogeneous systems

- Attributes
  - Facilitates mixed-mode system simulation, specification, and design
  - Supports generation of DSP assembly code from a block diagram description of algorithm
  - Uses object-oriented representations to model subsystems efficiently
  - Supports different design styles called *domains*

Codesign Methodology Using Ptolemy

- Ptolemy supports a framework for hardware/software codesign, called the *Design Assistant*
- The Design Assistant consists of two components
  - Specific point tools for estimation, partitioning, synthesis, and simulation
  - An underlying design methodology management infrastructure for design space exploration

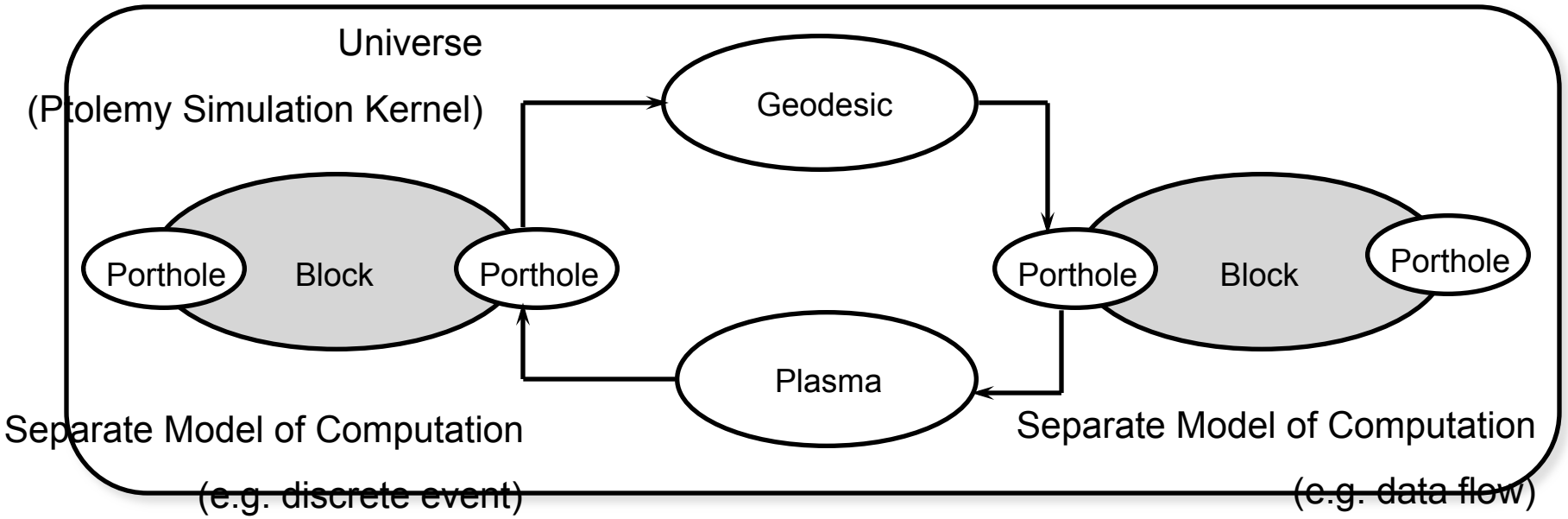
Codesign Methodology Using Ptolemy (Cont.)



[Rozenblit94]

Ptolemy Heterogeneous Simulation Environment

Structural Components



- Data encapsulated in “particles”
- “Block” objects send and receive messages
- Particles travel to/from external world through “portholes”

POLIS

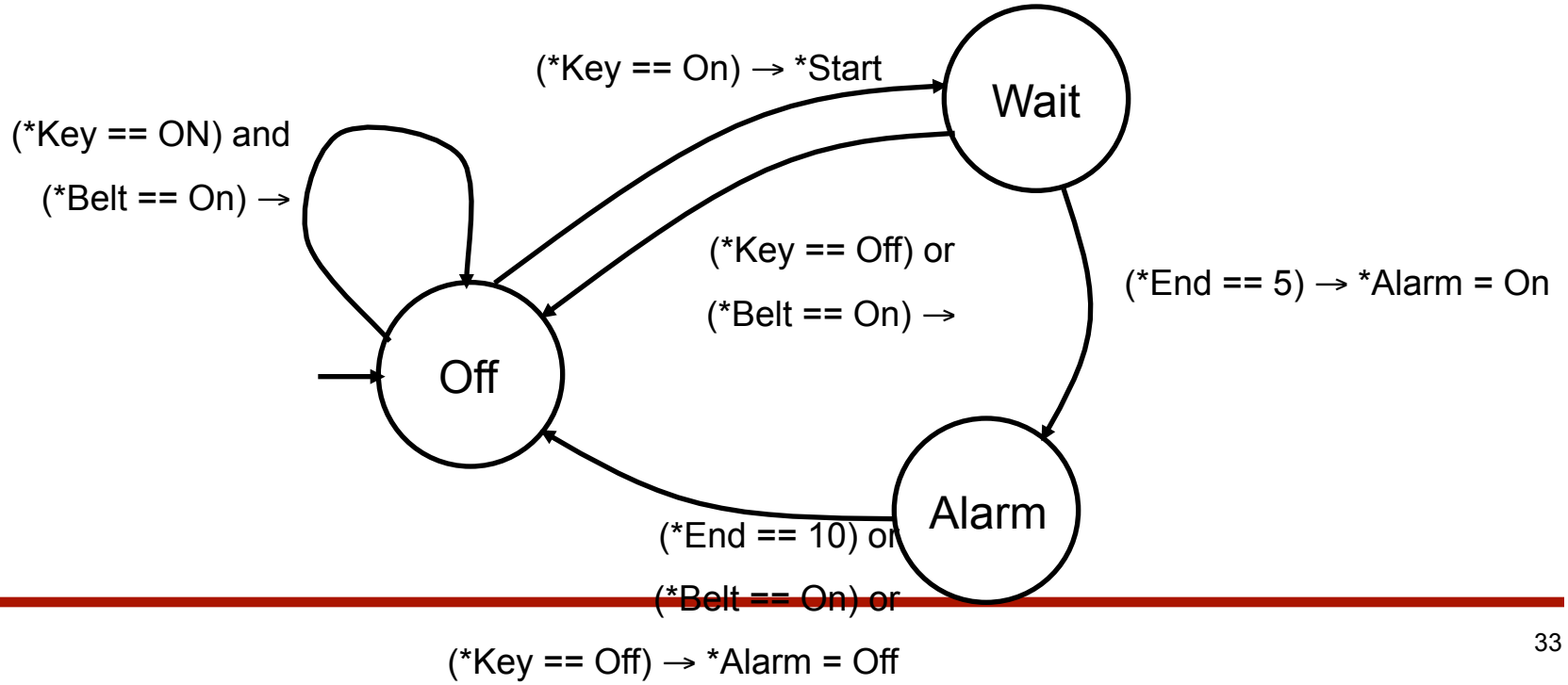
- Hardware/Software Codesign and synthesis system developed at the University of California, Berkeley
- Targeted towards small, scale, reactive, control dominated embedded systems
- Includes an “unbiased” mechanism for specifying the system’s function that allows for maximum flexibility in mapping to hardware or software and also allows for formal verification

POLIS
Unified Representation

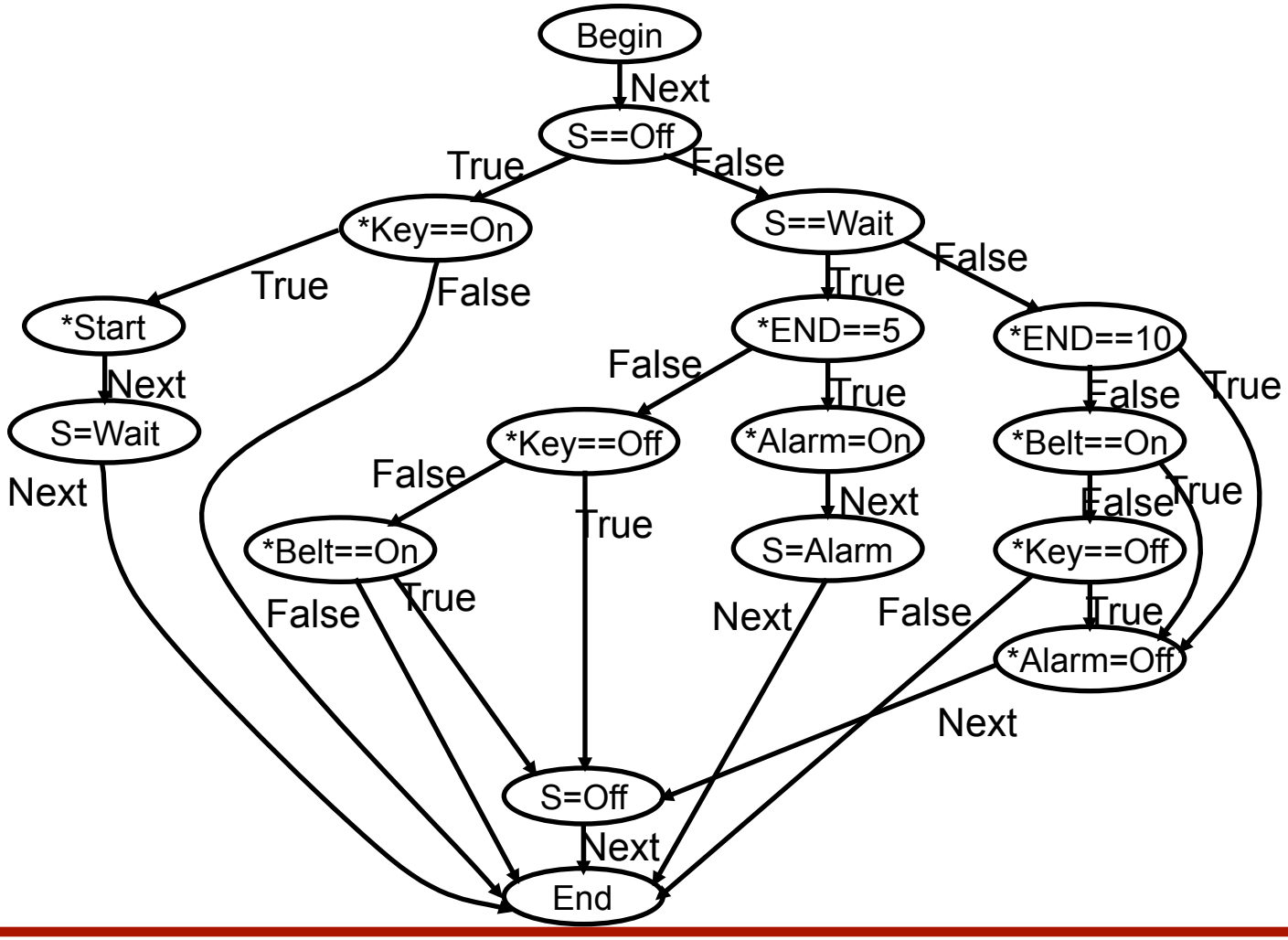
- System behavior is specified in a formal manner using Codesign Finite State Machines (CFSMs)
  - CFSMs translate a set of inputs to a set of outputs with only a finite amount of internal state
  - Unlike traditional FSMs, CFSMs do not all change state exactly at the same time (globally asynchronous)
- CFSMs are designed to be unbiased towards hardware or software
- Translators exist to convert other specification languages (e.g. ESTEREL) into CFSMs
- CFSMs can be translated into traditional FSMs to allow formal verification
- CFSMs can communicate with each other using events
  - Events are unidirectional and happen in non-zero, unbounded time
  - Events can be used to communicate across all domains (hardware or software)
  - Events are unbuffered and can be overwritten - however, they can be used to implement fully interlocked handshaking
- CFSMs are translated into behavioral FSMs for hardware synthesis and into S-graphs for software synthesis

Codesign Finite State Machines

- Specification: *"Five seconds after the key is turned on, if the belt has not been fastened, an alarm will beep for ten seconds or until the key is turned off"*



S-graph Software Specification



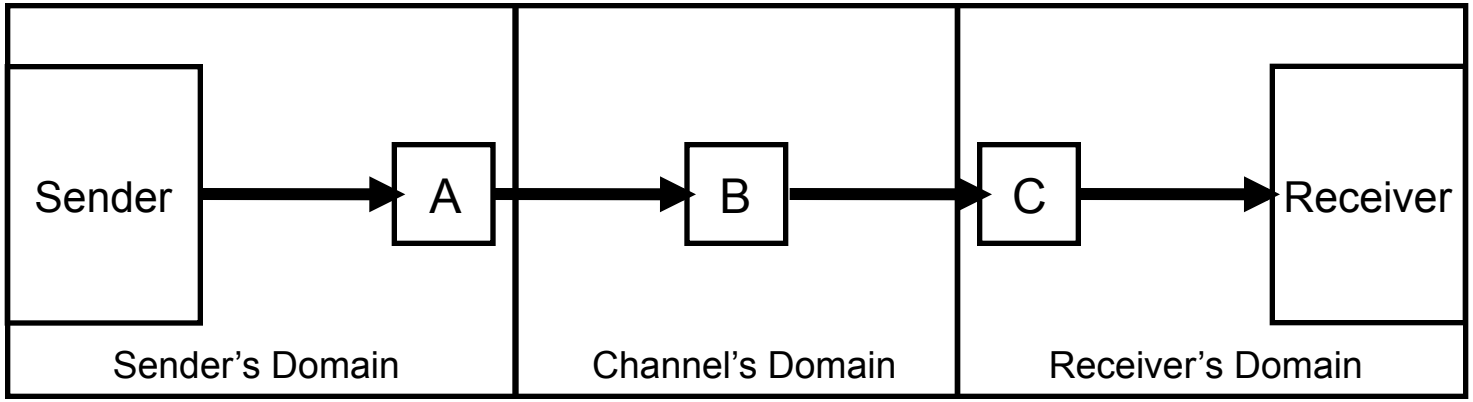
[Chiodo94]

Partitioning and Scheduling in POLIS

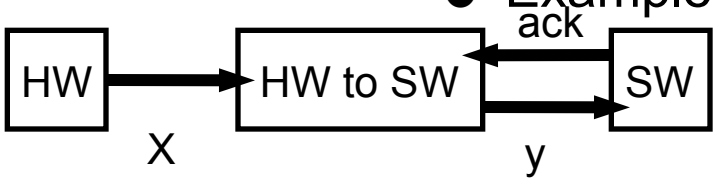
- Partitioning based on mapping CFSMs to either hardware or software
- This mapping is left to the user - performance feedback is provided by simulation
- Interfaces between partitions are automatically generated
- Scheduling based on executing CFSMs
- Selection of scheduling algorithm left to user - built into RTOS
  - Round-robin cyclic executive
  - Off-line I/O rate-based cyclic executive
  - Static pre-emptive: rate monotonic scheduling
  - Dynamic pre-emptive: Earliest Deadline First

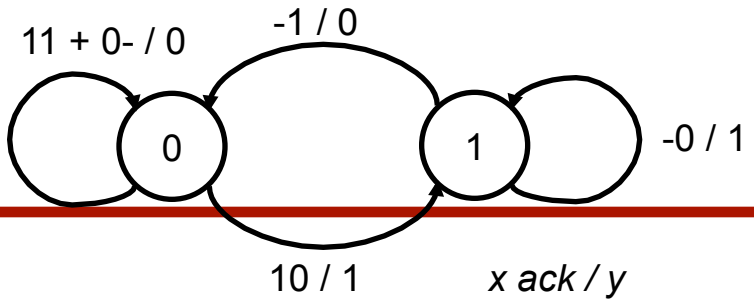
Interfaces Among Partitions

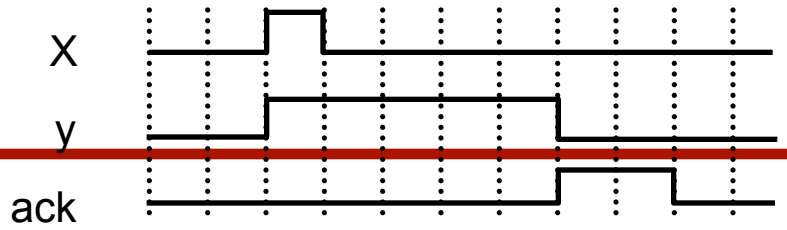
- Interfaces use strobe/data protocol (corresponding to the event/ value primitive)



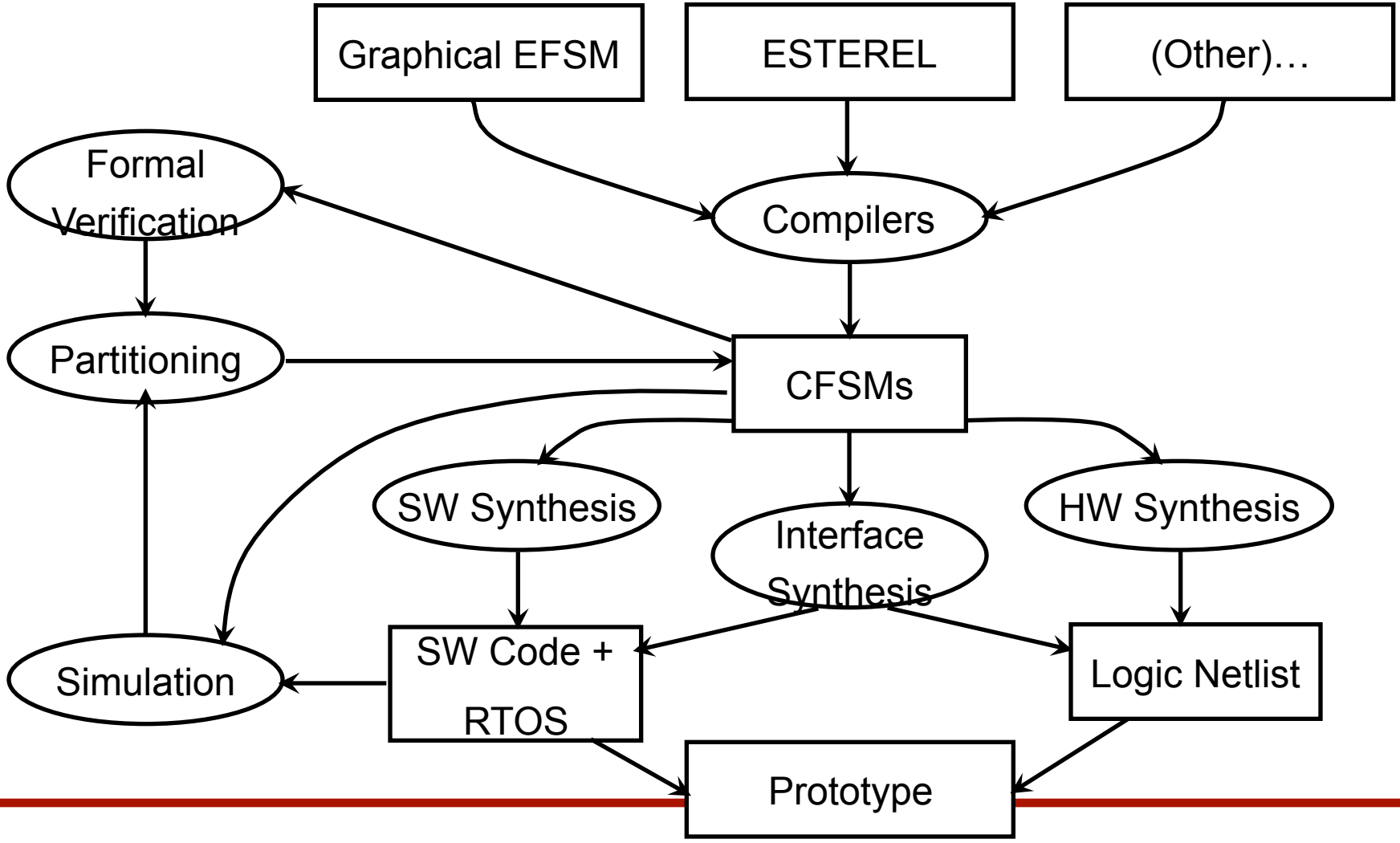
- Example HW to SW interface







The POLIS Co-design Environment



References

[Boehm73] Boehm, B.W. “Software and its Impact: A Quantitative Assessment,” *Datamation,* May 1973, p. 48-59.

[Buchenrieder93] Buchenrieder, K., “Codesign and Concurrent Engineering”, Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 85-86

[Buck94] Buck, J., et al., “Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems,” *International Journal of Computer Simulation*, Vol. 4, April 1994, pp. 155-182.

[Chiodo92] Chiod0, M., A. Sangiovanni-Vincentelli, “Design Methods for Reactive Real-time Systems Codesign,” *International Workshop on Hardware/Software Codesign*, Estes Park, Colorado, September 1992.

[Chiodo94] Chiodo, M., P. Giusto, A. Jurecska, M. Marelli, H. C. Hsieh, A. Sangiovanni-Vincentelli, L. Lavagno, “Hardware-Software Codesign of Embedded Systems,” *IEEE Micro*, August, 1994, pp. 26-36; © IEEE 1994.

[Chou95] P. Chou, R. Ortega, G. Borriello, “The Chinook hardware/software Co-design System,” *Proceedings ISSS*, Cannes, France, 1995, pp. 22-27.

[DeMicheli93] De Micheli, G., “*Extending CAD Tools and Techniques*”, Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 84

[DeMicheli94] De Micheli, G., “Computer-Aided Hardware-Software Codesign”*, IEEE Micro*, August, 1994, pp. 10-16

[DeMichelli97] De Micheli, G., R. K. Gupta, “Hardware/Software Co-Design,” *Proceedings of the IEEE*, Vol. 85, No. 3, March 1997, pp. 349-365.

[Ernst93] Ernst, R., J. Henkel, T. Benner, “*Hardware-Software Cosynthesis for Micro-controllers*”, *IEEE Design and Test*, December, 1993, pp. 64-75

[Franke91] Franke, D.W., M.K. Purvis. “Hardware/Software Codesign: A Perspective,” *Proceedings of the 13th International Conference on Software Engineering,* May 13-16, 1991, p. 344-352; © IEEE 1991

References (Cont.)

[Gajski94] Gajski, D. D., F. Vahid, S. Narayan, J. Gong, *Specification and Design of Embedded Systems*, Prentice Hall, Englewood Cliffs, N J, 07632, 1994

[Gupta92] Gupta, R.K., C.N. Coelho, Jr., G.D. Micheli. "Synthesis and Simulation of Digital Systems Containing Interactive Hardware and Software Components," *29th Design Automation Conference,* June 1992, p. 225-230.

[Gupta93] Gupta, R.K., G. DeMicheli, "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design and Test*, September 1993, p.29-40; © IEEE 1993.

[Hermann94] Hermann, D., J. Henkel, R. Ernst, "An approach to the estimation of adapted Cost Parameters in the COSYMA System*", 3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 100-107

[Hood94] Hood, W., C. Myers, "RASSP: Viewpoint from a Prime Developer," *Proceedings 1st Annual RASSP Conference*, Aug. 1994.

[IEEE] All referenced IEEE material is used with permission.

[Ismail95] T. Ismail, A. Jerraya, "Synthesis Steps and Design Models for Codesign," *IEEE Computer*, no. 2, pp. 44-52, Feb 1995.

[Kalavade93] A. Kalavade, E. Lee, "A Hardware-Software Co-design Methodology for DSP Applications," *IEEE Design and Test*, vol. 10, no. 3, pp. 16-28, Sept. 1993.

[Klenke96] Klenke, R. H., J. H. Aylor, R. Hillson, D. J. Kaplan, "VHDL-Based Performance Modeling for the Processing Graph Method Tool (PGMT) Environment," *Proceedings of the VHDL International Users Forum*, Spring 1996, pp. 69-73.

[Kumar95] Kumar, S., *"*A Unified Representation for Hardware/Software Codesign*"*, Doctoral Dissertation, Department of Electrical Engineering, University of Virginia, May, 1995

[Jalote91] Jalote, P., *An Integrated Approach to Software Engineering,* Springer-Verlag, New York, 1991.

[McFarland90] McFarland, M.C., A.C. Parker, R. Camposano. "The High-Level Synthesis of Digital Systems," *Proceedings of the IEEE*, Vol. 78, No. 2, February 1990, p.301-318, © IEEE 1990.

References (Cont.)

[Parker84] Parker, A.C., "Automated Synthesis of Digital Systems," *IEEE Design and Test*,, November 1984, p. 75-81.

[RASSP94] *Proceedings of the 1st RASSP Conference*, Aug. 15-18, 1994.

[Rozenblit94] Rozenblit, J. and K. Buchenrieder (editors). Codesign Computer -Aided Software/Hardware Engineering, IEEE Press, Piscataway, NJ, 1994; © IEEE 1994.

[Smith86] Smith, C.U., R.R. Gross. "Technology Transfer between VLSI Design and Software Engineering: CAD Tools and Design Methodologies," *Proceedings of the IEEE*, Vol. 74, No. 6, June 1986, p.875-885.

[Srivastava91] M. B. Srivastava, R. W. Broderson, "Rapid prototyping of Hardware and Software in a Unified Framework," *Proceedings ICCAD*, 1991, pp. 152-155.

[Subrahmanyam93] Subrahmanyam, P. A., "Hardware-Software Codesign -- Cautious optimism for the future", Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 84

[Tanenbaum87] Tanenbaum, A.S., *Operating Systems: Design and Implementation,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1987.

[Terry90] Terry, C. "Concurrent Hardware and Software Design Benefits Embedded Systems," *EDN*, July 1990, p. 148-154.

[Thimbleby88] Thimbleby, H. "Delaying Commitment," *IEEE Software,* Vol. 5, No. 3, May 1988, p. 78-86.

[Thomas93] Thomas, D.E., J.K. Adams, H. Schmitt, "A Model and Methodology for Hardware-Software Codesign," *IEEE Design and Test,* September 1993, p.6-15; © IEEE 1993.

[Turn78] Turn, R., "Hardware-Software Tradeoffs in Reliable Software Development," *11th Annual Asilomar Conference on Circuits, Systems, and Computers,* 1978, p.282-288.

[Vahid94] Vahid, F., J. Gong, D. D. Gajski, "*A Binary Constraint Search Algorithm for Minimizing Hardware During Hardware/Software Partitioning",* 3rd International Conference on Hardware/Software Codesign, Grenoble, France, Sepetember22-24, 1994, pp. 214-219

[Wolf94] Wolf, W.H. "Hardware-Software Codesign of Embedded Systems," *Proceedings of the IEEE*, Vol. 82, No.7, July 1994, p.965-989.

References (Cont.)

Additional Reading:

Aylor, J.H. et al., "The Integration of Performance and Functional Modeling in VHDL" in *Performance and Fault Modeling with VHDL,* J. Schoen, ed., Prentice-Hall, Englewood Cliffs, N.J., 1992.

D'Ambrosio, J. G., X. Hu, "Configuration-level Hardware-Software Partitioning for Real-time Embedded Systems", *3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 34-41

Eles, P., Z. Peng, A. Doboli, "VHDL System-Level Specification and Partitioning in a Hardware-Software Cosynthesis Environment", *3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 49-55

Gupta, R.K., G. DeMicheli, "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design and Test*, September 1993, p.29-40.

Richards, M., Gadient, A., Frank, G., eds. *Rapid Prototyping of Application Specific Signal Processors*, Kluwer Academic Publishers, Norwell, MA, 1997

Schultz, S.E., "An Overview of System Design," *ASIC and EDA*, January 1993, p.12-21.

Thomas, D. E, J. K. Adams, H. Schmit, "A Model and Methodology for Hardware-Software Codesign", *IEEE Design and Test*, September, 1993, pp. 6-15

Zurcher, F.W., B. Randell, "Iterative Multi-level Modeling - A Methodology for Computer System Design," *Proceedings IFIP Congress '68,* Edinburgh, Scotland, August 1968, p.867-871.

Final issues

- Come by my office hours (right after class)
- Any questions or concerns?