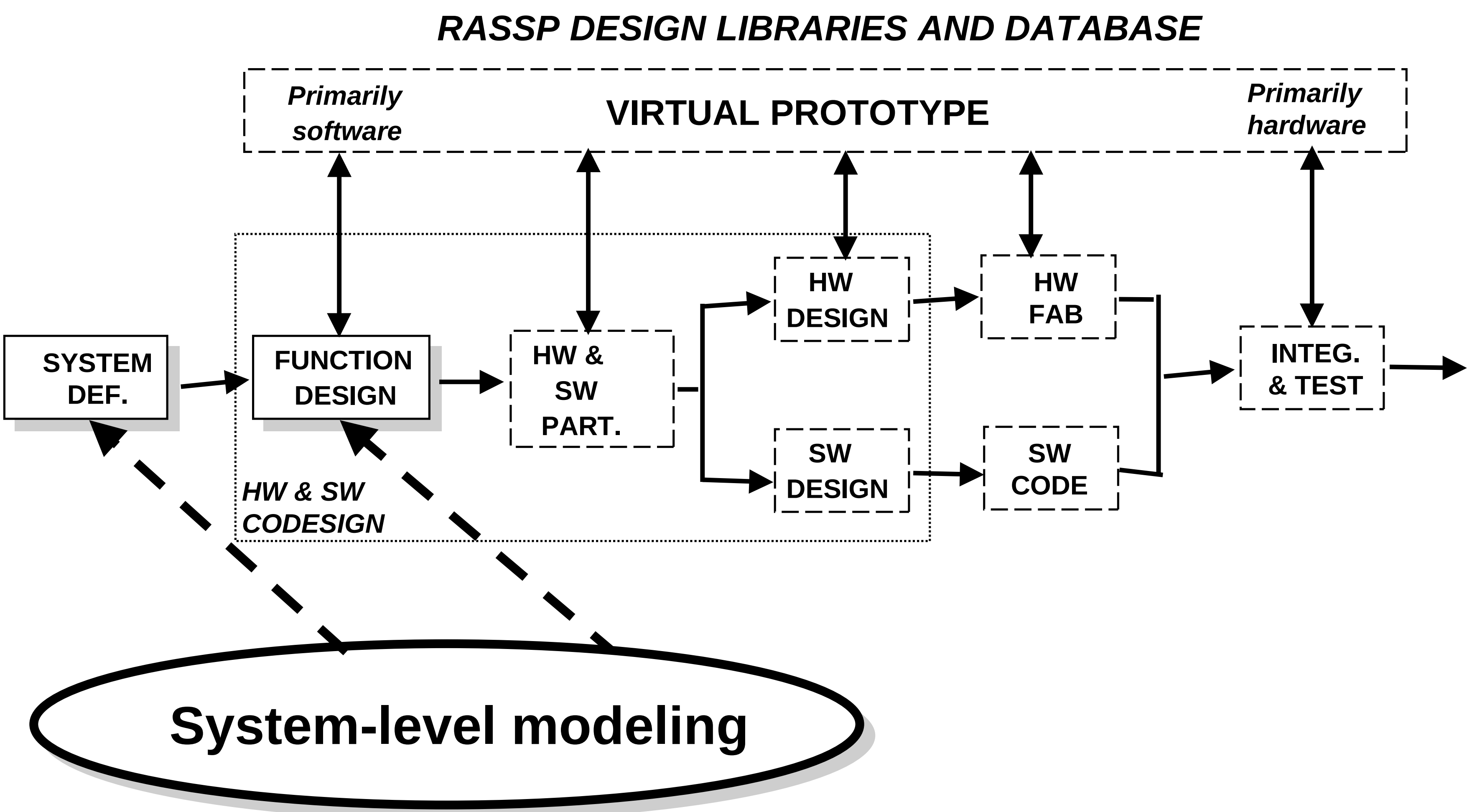
System-Level Modeling
(KJH, with slides removed from RASSP Module 9, vhdl_M.ppt)
Fall 2000

RASSP Education & Facilitation

Version D 0.2

RASSP Roadmap



[Richards94]

Module Goals

- To provide motivation for the use of system-level modeling
- To show how the use of system-level modeling can improve design methodology
- To detail the types of system-level modeling and what types of analysis can be done with each
- To show how to incorporate system-level modeling into a design environment

Introduction

- **Motivation**
  - **Digital systems have become large and complex**
    - **Breadboard and prototypes are too costly for demonstrating complex system performance**
    - **Need analysis and simulation of hardware and software**
  - **There is a shift from structural to behavioral design**
  - **Different models of the same system are used at different stages and by different designers, resulting in**
    - **Possibility of loss of information**
    - **Difficulties or misunderstandings caused by inconsistencies between different models**
    - **Need to use different tools for different models**
  - **Redesign of digital systems costs $ 5-10 billions annually in US alone**

System-level Modeling

SYSTEM DESIGN

System Requirements
((4) Executable requirements)

(5) bus functional

System-level modeling
(System Designers)

(1) Performance Modeling
a. ADEPT
b. Honeywell PML

(3) Functional Modeling
a. MAT2DSP
b. Ptolemy

(2) Dependability Modeling
a. REST
b. ADEPT

Specifications

[Rao94]

Requirements for Effective System-level Modeling

- **Need a unified environment**
  - **Need capability for transition from system-level model to the final implementation in a step-wise manner**
- **Need an integrated system-level analysis and design**
- **Need to incorporate performance, dependability, and functional modeling capability at all hierarchies of the design**
- **Need to have power and flexibility to model digital systems at many different levels of description**
  - **Support "mixed" simulation at different levels of abstraction, representation, and interpretation with an ability for step-wise refinement**

System-level Modeling Definitions

- **Model: Representation of an entity in some form other than the form in which the entity exists**
  - **Necessarily lacks some detail of the real system**
  - **Examples include textual specification, requirements documents, analytical models, simulation models, physical models**
  - **Are useful in the design phases when the actual device is not available or the necessary experimentation is destructive, etc.**
- **Simulation: The act of animating a model with respect to some of the parameters of the model**
  - **An example is movement of tokens representing information flow according to the simulation rules of the model**

System-level Modeling Definitions (Cont.)

- **Behavioral model: Describes the function and timing of hardware independent of any specific implementation**
  - **Can exist at multiple levels of abstraction, depending on the granularity of the timing and the data types that are used in the functional description**
  - **Data flow, procedural and structural constructs may be used to express behavior**
- **Structural model: Represents a system in terms of the interconnections of a set of components**
  - **Components are described structurally or behaviorally, with interfaces between structural and behavioral-level models**
- **Physical model: Specifies the relationship between the component model and the physical packaging of the component.**

Abstraction

- A model is classified as being at a certain level of abstraction depending on the features of its behavior, structure, and timing measures
- Different levels of abstraction imply that
  - There exists an algorithm for the conversion of a model at one level of abstraction to another level of abstraction without loss or gain of information
  - Information describing the system is merely transferred between the external algorithm and the system description

Levels of Abstraction

- **Network level**
  - **Encompasses performance and interface models**
  - **Basic structural model components are processors, memories, and interconnection elements**
  - **Behavior is described through the transmission and receipt of messages**
  - **Granularity of time is given by response times to messages**
  - **Evaluation of response times to stimuli and throughput of the hardware is possible at this level**
- **Algorithmic level**
  - **Models the functions of a hardware system kernel without the functionality or timing of its interface**
  - **Can be called *functional modeling***

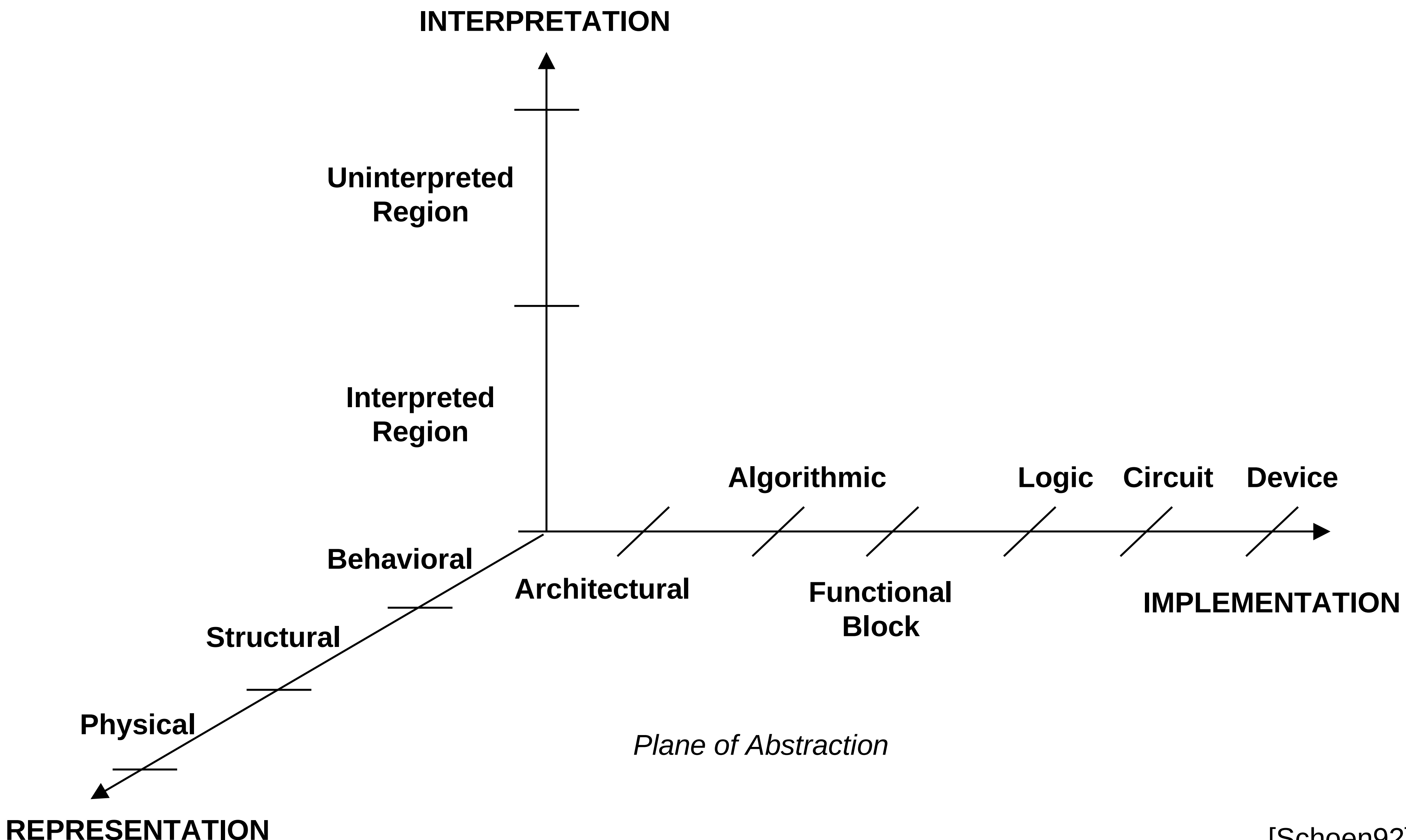
Levels of Abstraction (Cont.)

- **Instruction set architecture (ISA) level**
  - **Functions of an ISA model of a processor are the instruction set of the processor**
  - **Supports simulated execution of software; if the compilers are available, can be used to debug the software written for the processor**
  - **Timing of an ISA model is the time required to perform each instruction of the instruction set of the processor**
- **Fully functional level**
  - **Models all the documented characteristics of the processor**
  - **Pin behavior of the component is modeled accurately, both in function and timing**

Levels of Abstraction (Cont.)

- Register-transfer level (RTL)
  - Is similar to FFM, but with subtle differences
  - Models undocumented characteristics of the device
  - Models more of physical characteristics in terms of internal and interface timing and function than the FFM
- Gate level
  - Constructed structurally with primitive cells that represent Boolean logic functions

Levels of Abstraction (Cont.)



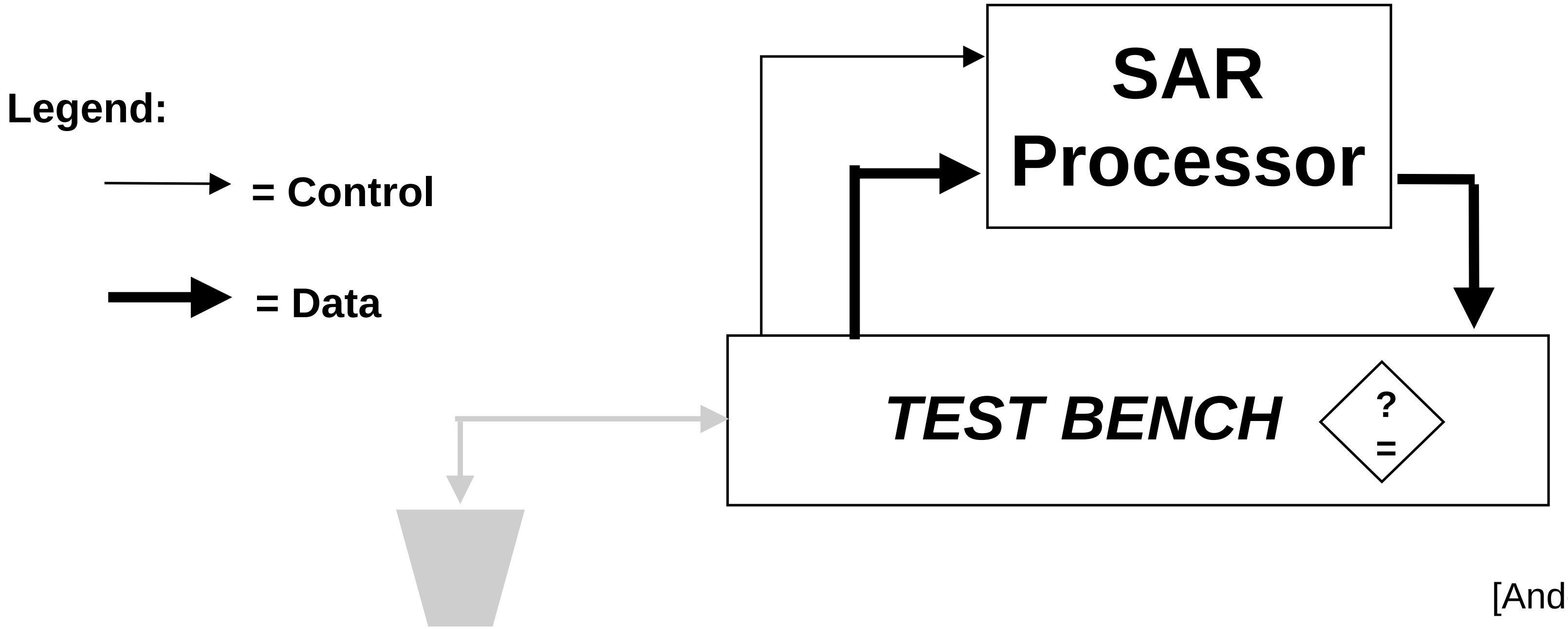
[Schoen92]

(4) Executable Specifications

- Executable Specification - specification capable of simulating the required external behavior of a system
  - Can be treated as a very early prototype of the system
  - Removes the ambiguity associated with written specifications
  - Bridges the gap between the specifications and design
  - Enhances communication among and within customer and designer groups
  - Enhances high-level of conformance between a specification model and the performance model
- Ensures conformance between a specification and the performance model being developed based on the specifications

Executable Specifications MIT Lincoln Laboratories

- RASSP benchmarking of an executable requirement includes
  - A simulatable model of the system
  - A testbench which sources commands and data, sinks output data and may perform some checking



[Anderson94]

Express VHDL/i-Logix

- **Can graphically create specification models**
- **Generates the equivalent VHDL code**
- **Methodology**
  - **Designer captures the statecharts of the specification with the *Statecharts Editor***
  - ***Model Execution Tool* operates on the statecharts of the model and animates the behavior of the specification**
  - **Results in both a screen animation of the specification's behavior and a textual trace report of the scenario tested**

(1) Performance Modeling Overview

- Performance models provide information on system timing and do not simulate the functions of the system being modeled
- Performance models are typically simulated, not analyzed
  - Analytical models can rapidly become too complex to fully represent important system features; e.g., resource contention
  - Simulation models can accommodate mixed levels of design and various levels of fidelity and accuracy
  - Simulation models suffer from significant startup costs, complexity, and significant execution (CPU) times

Performance Modeling Overview (Cont.)

- **Performance models support performance and architectural tradeoffs (what-if analysis)**
  - **Facilitate early integration of hardware and software, and documentation of design decisions**
  - **Aid in identification of bottlenecks**
  - **Serve as a guideline for the model developers, system architects, and review teams**

Performance Modeling Overview (Cont.)

- In the early part of the design
  - The exact functionality of the components is not known
  - Develop structure, architecture and basic design goals of the system
- In the later phases
  - As the functions of the individual components are developed or components are selected from existing libraries
  - System description can be systematically converted into a fully interpreted description for final verification

Performance Evaluation

- **Typical metrics**
  - **Utilization: Percentage of time the system resources are busy**
  - **Throughput: The rate at which system can process data**
  - **Latency (Response Time): Time to process data values**
  - **Fault Tolerance: System reliability, safety & availability**
- **To allow measurement of these metrics, performance models must have as little detail as possible**

Uninterpreted Models (Cont.)

- Used by system designers prior to creating specifications for the hardware designers
- Represents flow of information without regard to information itself; deals with presence or absence of data or control signals
  - Tokens represent presence of information - not particular values
- Represents the performance of a system by aggregating the delays associated with tokens flowing through the system

Uninterpreted Models (Cont.)

Uninterpreted Models

Queuing Models

Statistical Data

Petri Nets

Deterministic/statistical Data (Event-based)

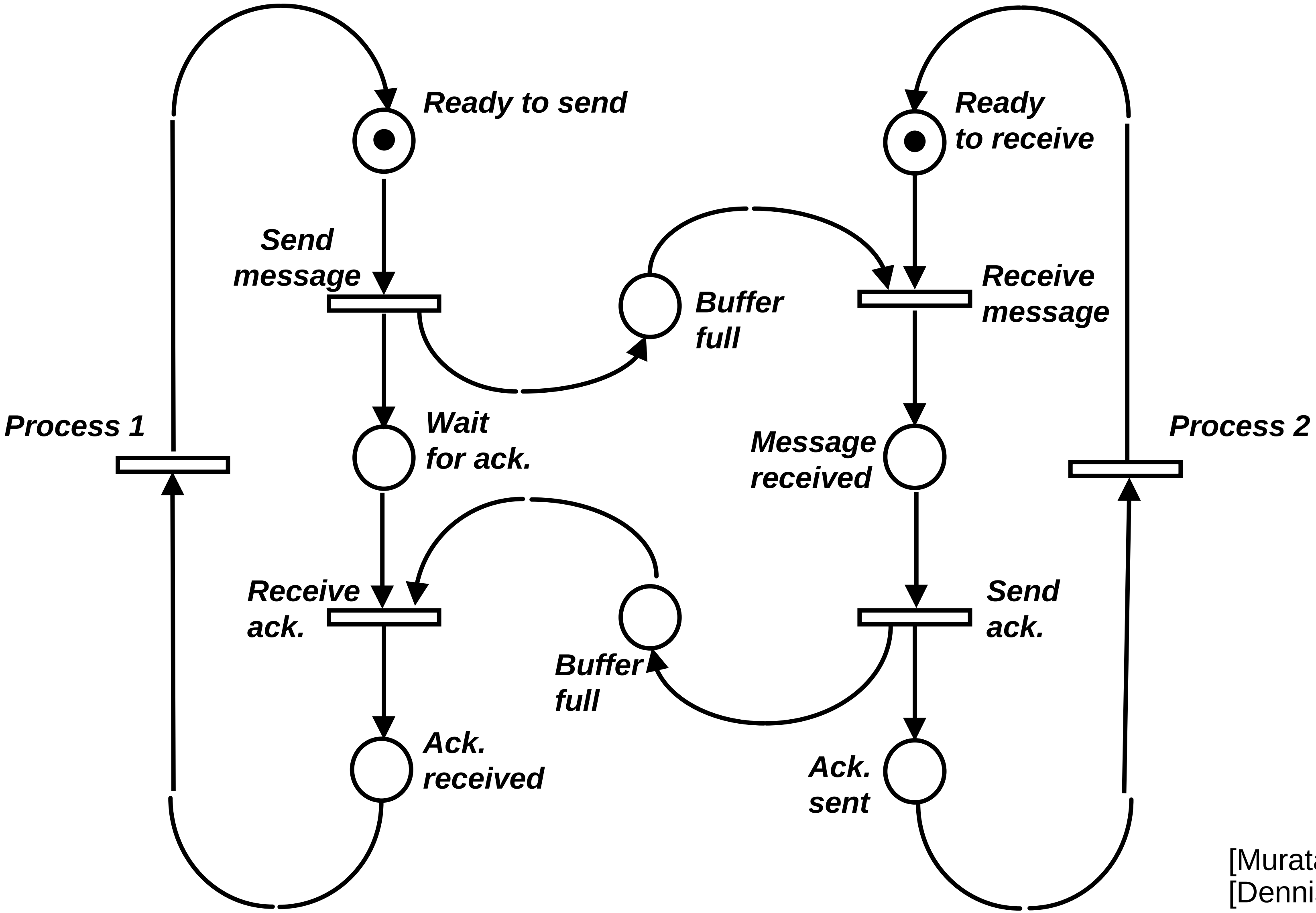
Token-based Simulation Models

Statistical Data

Petri Nets

- Describe the flow of information between "places," with flow control being defined by "transitions" and "firing rules (mapping)"
- Simulation rules for the Petri Net describe the conditions for movement of tokens
- Approach is useful for modeling the actual hardware and software systems
- Marked Petri Net has a mapping which can assign multiple tokens to each place in the net
- Colored Petri Net has color fields (mostly integer and Boolean) associated with tokens
  - An uncolored token represents presence of information only
  - Useful for adding functionality to the model

Petri Nets Examples

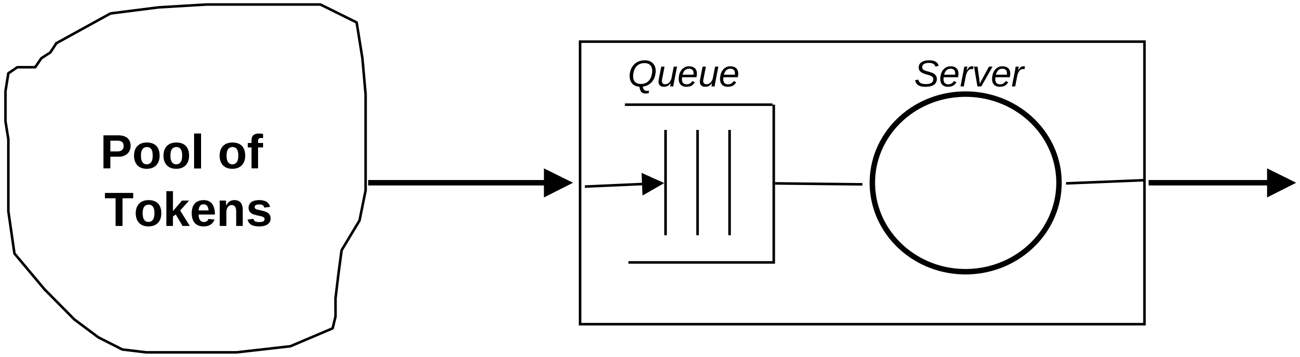


[Murata89], [Dennis70]

Queuing Models

- **Queuing models represent the system in which tokens are not serviced immediately, but are made to wait in a queue for the server to become free to service the token**
- **Queuing models work well for gaining statistical data on very high-level uninterpreted models of digital systems**
- **At a lower level, the queuing model has difficulty expressing the deterministic nature of the system**

Queuing Models (Cont.)



Single-server Queuing System

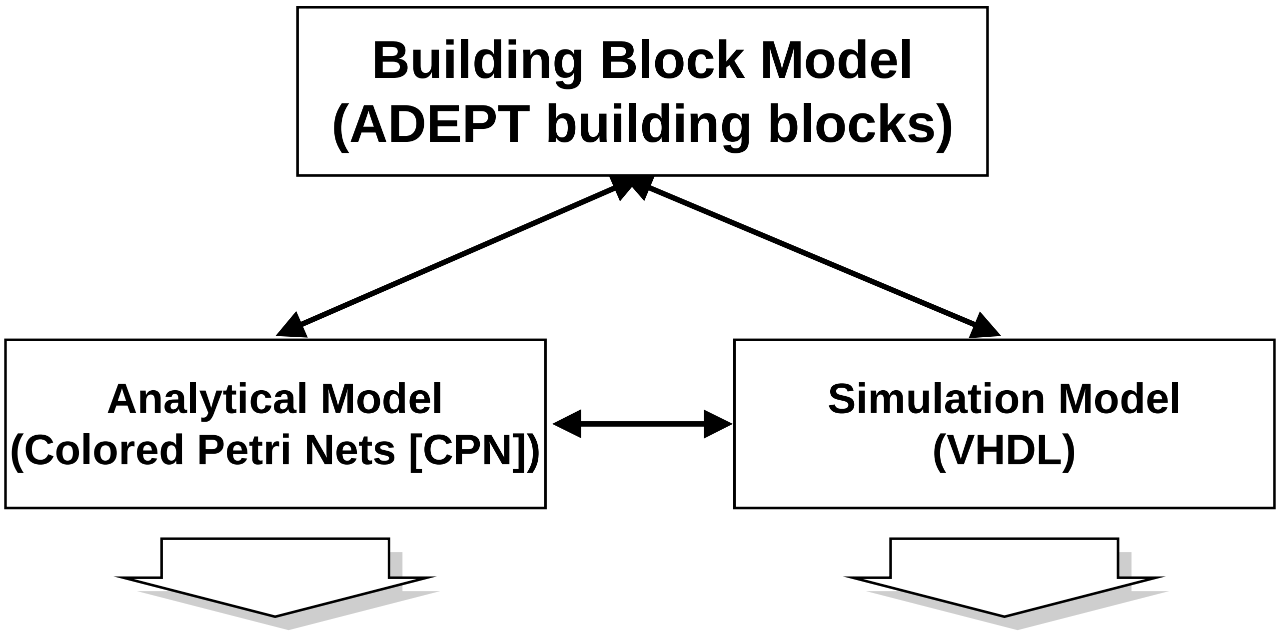
[MacDougall87]

Token-based Simulation Models

- Are driven by simulation semantics, and the dynamic behavior of the system is studied
- Allow arbitrary precision for a given simulation time
- Are useful for analyzing huge systems where analytical methods are computationally expensive (exponential complexity)
- Examples
  - ADEPT
  - RESQ
  - Honeywell PML
  - ADAS

ADEPT

ADEPT VIEW



Analytical Studies
- **Behavior**
- **Performance**
- **Dependability**

Performance Analysis
- **Behavior**
- **Performance**
- **Dependability**

[Rao94]

ADEPT (Cont.)

- Is an integrated design environment that permits linking of the design phases from initial concept to the final physical implementation
- Has an inherent top-down hierarchical design
  - A single model from which different representations can be obtained
  - Building block approach:
    - Has a library of primitive modules; enables the user to define modules and to include them in the library
    - Has VHDL description as well as the underlying CPN representation associated with them
  - Modules may be interconnected to mimic hardware, software, and the interaction between the two
  - Uses CPN theory for development of model reduction techniques - decreases simulation time

ADEPT (Cont.)

- **Permits the designers to transcend several levels of abstraction and interpretation within the same environment using the same language**
  - **Uninterpreted modeling is supported by a set of primitive modeling modules and an underlying communication mechanism**
  - **Supports simultaneous performance, reliability, and functional modeling in a single environment**
- **User interface for**
  - **Specifying model; allows user to visually interconnect modules that constitute the model**
  - **Allows extraction of performance metrics, graphical display in the form of bar graphs and waveforms**
- **Hardware/software codesign techniques can be developed within ADEPT**

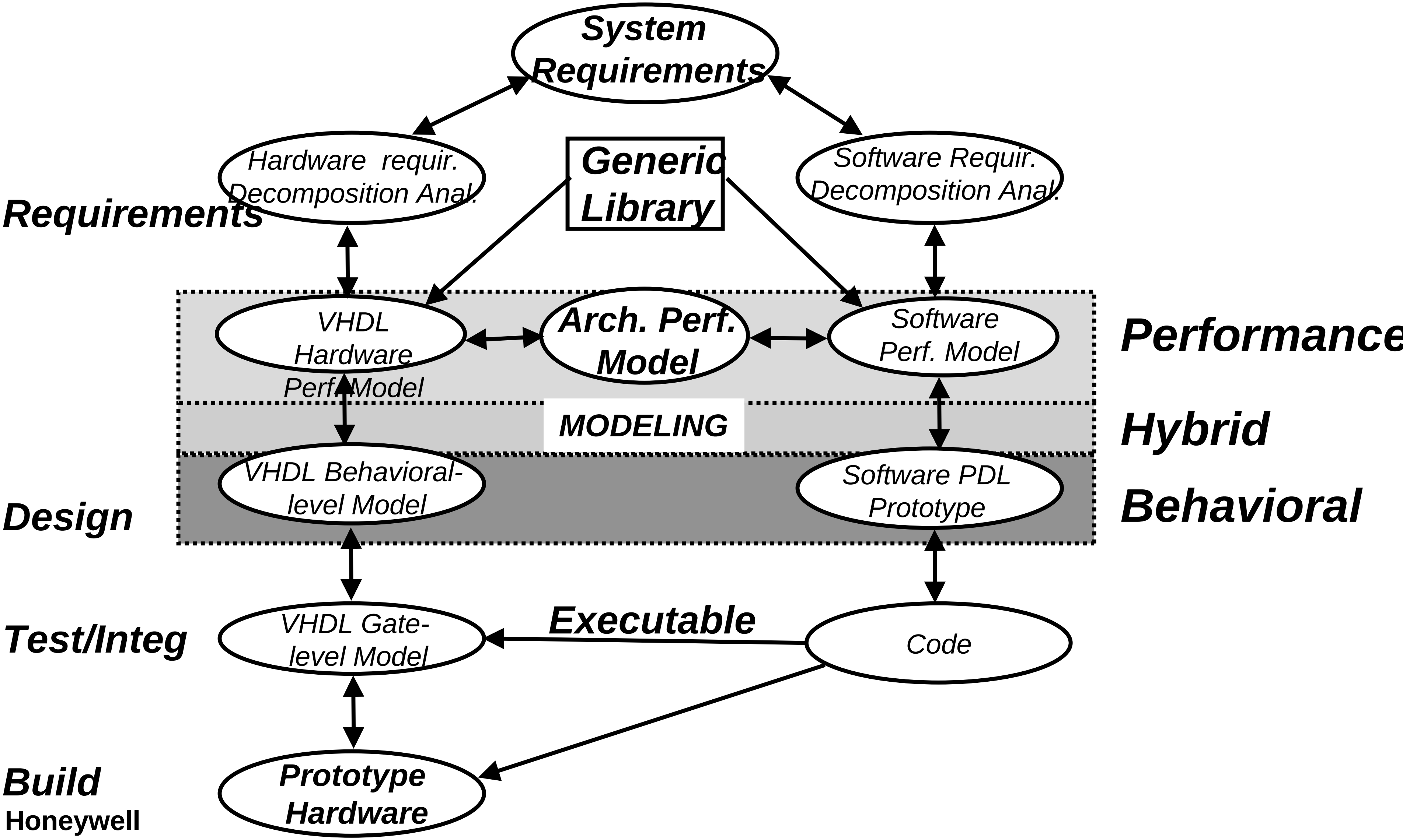
Honeywell Performance Modeling Library (PML)

- Targeted towards high-level description, specification, and performance analysis of computing systems at a system level
- Serves as a simulatable specification, aids the identification of bottlenecks, and supports performance validation
- Can be used for capturing and documenting architectural-level designs, and can be used as a testbed for architectural performance analysis studies
- Provides VHDL performance model within the RASSP design environment

Methodology
RASSP
Reinventing Electronic Design
Architecture Infrastructure
ARPA • Tri-Service

Honeywell PML

RASSP E&F
SCRA GT UVA Raytheon UCinc EIT MMG ADL



Honeywell PML Features

- **Generic building block**
  - **Can be assembled and configured rapidly to many degrees of fidelity with minimal effort**
  - **Modules are interconnected with structural VHDL**
  - **Types available**
    - **Configurable input/output devices**
    - **Memories**
    - **Communication elements**
    - **Processor element**
- **Appropriate to apply at architectural level**
  - **Actual device under study (such as a signal processor) and its environment (such as sensors and actuators)**

Honeywell PML

- Standard output routines tabulate and graph performance statistics such as latency, utilization, and throughput
- Interoperability guidelines ensure that models from multiple sources will integrate smoothly
- Hybrid models support smooth integration between performance and functional models
- Capable of representing systems consisting of ASICs, boards, subsystem cabinets, and sensor networks
- Effect of software on the architecture can be characterized and modeled

Interpreted Models

- Includes behavioral models and functional models
- Contain functions and data values to be transformed according to these functions
- More diverse and difficult to classify
  - Behavioral or language-based models: Programming design language (PDL) constructs allow the development of simulation models
    - VHDL (IEEE and DoD standard)
    - Do not specifically address hardware timing considerations
  - Structural or primitive (macro-) based models: (Gate-level models): Allow the system to be specified in terms of predefined primitive elements
  - Physical models: (SPICE): Describe the system in terms of the fundamental differential equations that govern the circuit and device operation

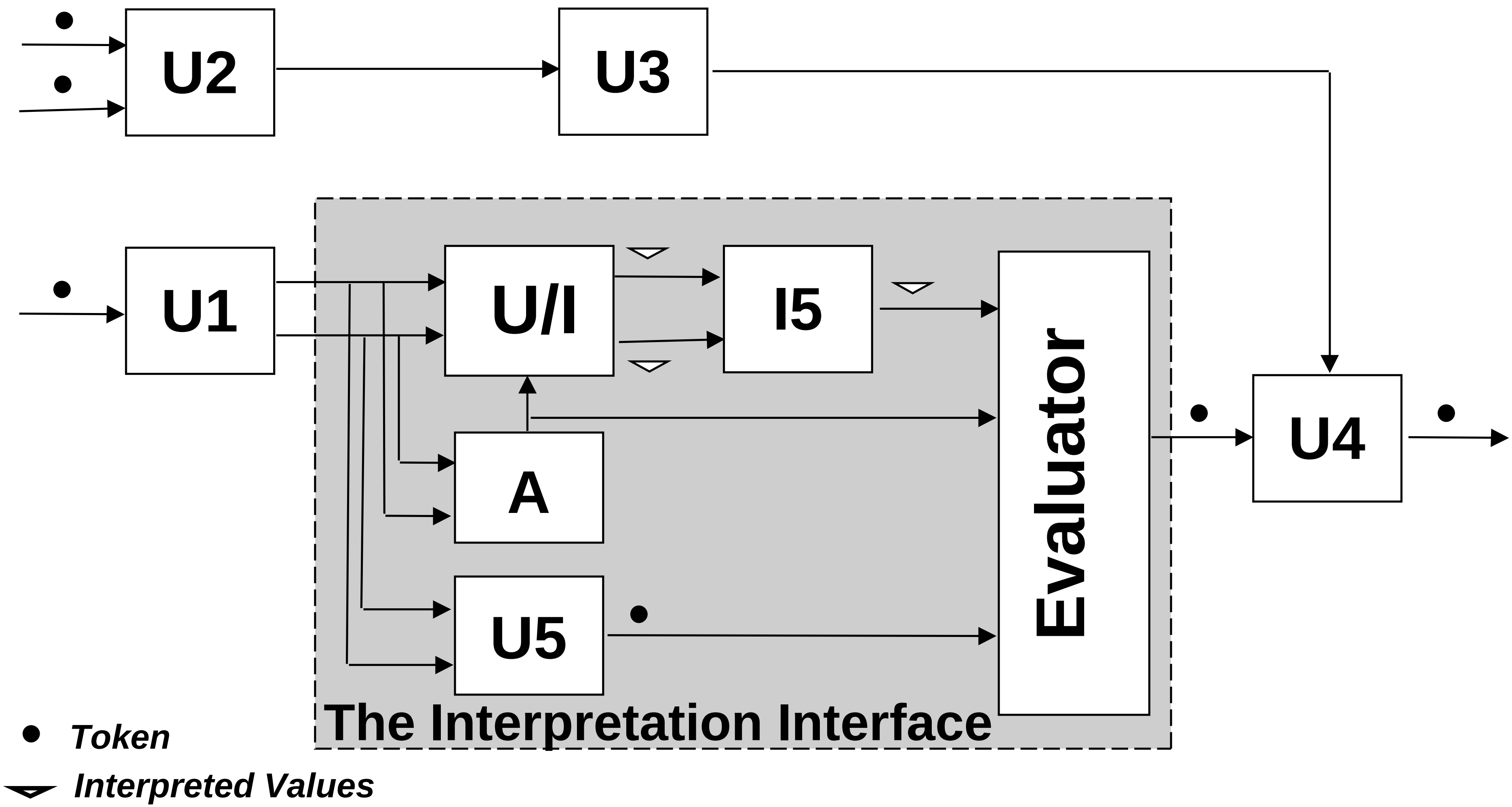
Hybrid Models

- **Contain uninterpreted and interpreted elements attributes**
  - **Use HDLs and their simulators**
  - **Adding uninterpreted modeling to HDLs provides a single design environment**
  - **Communicating between different regions takes place through interfaces which convert tokens to values and values to tokens**
- **Delay statistics of the interpreted models can be back-annotated with the statistics obtained from the hybrid model, giving an "improved" uninterpreted model**

Hybrid Models

- Uninterpreted-to-interpreted conversion
  - Requires that the input values be supplied to the interpreted model
  - Tokens can be tagged with necessary values - values are read from tokens and applied to interpreted model
  - Values can be generated from
    - List of known values
    - Values based on probability distributions
- Interpreted-to-uninterpreted conversion
  - Requires quantization of output data into tokens
  - Effective loss of information
  - Quantization of information into tokens can occur on an event with
    - No change in value, on a particular value, on a change in value

ADEPT Hybrid Modeling



[ADEPT UM94]

Object-oriented Analysis Shlaer-Mellor Model

- Highest-level partitioning construct - domain
  - Is a set of conceptual entities, or objects, that can exist independently of the objects in other domains
  - May be partitioned into one or more subsystems, each consisting of a set of related objects
  - Hardware/software partitioning is done using domains
- Within a subsystem, objects are represented in an object information model
  - Attributes of an object are used to define its characteristics
  - A connection between two objects in the information model represents a relationship that holds between them
  - Relationship includes one-to-one, one-to-many, and many-to-many
  - Object-information model also supports inheritance relationships

Object-oriented Analysis

- **A state model describes the behavior of object instances throughout their lifecycle**
  - **A state model consists of a set of states and events**
  - **An object instance can only be in one state at any given point in time; an event causes a transition from one state to another**
  - **Different object instances execute concurrently and can be in different states simultaneously**

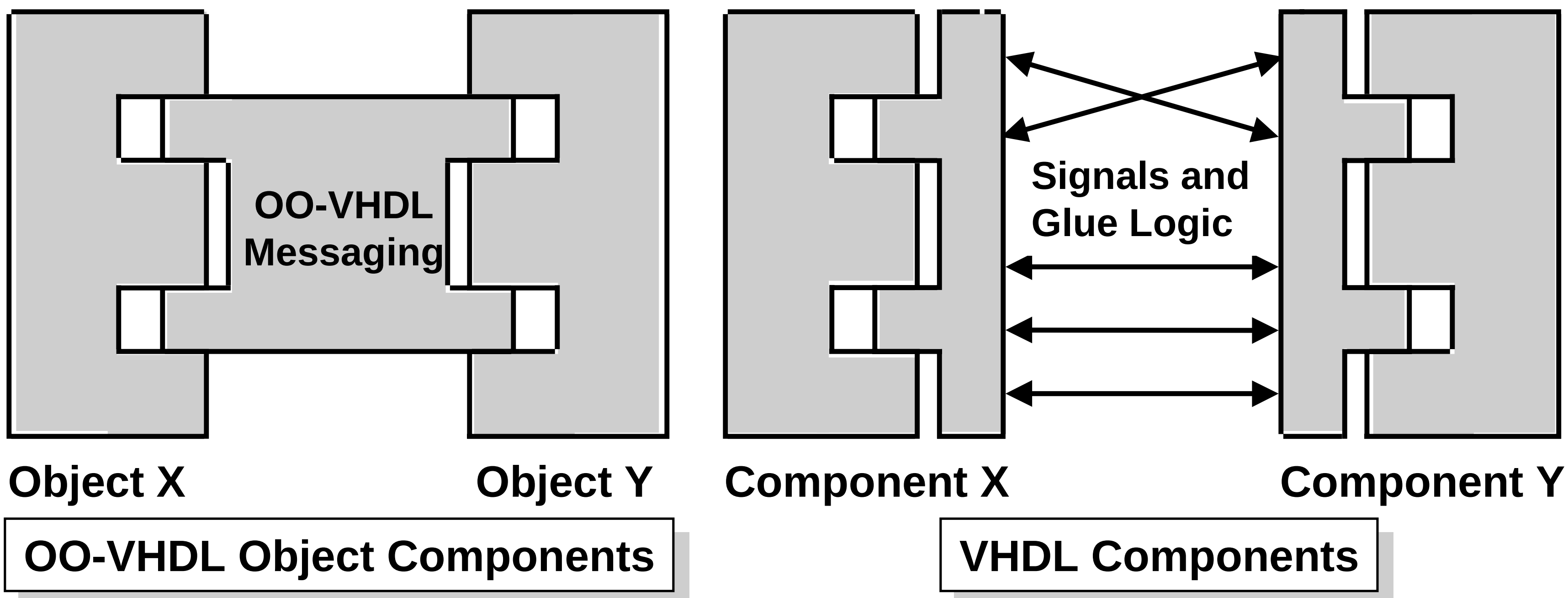
Object-oriented Analysis (Cont.)

- **An activity, or action, is executed when an object arrives at a state**
  - **State models synchronize and communicate with each other using events**
  - **During execution of an action, an object instance may generate an event destined for itself or another object instance**
  - **A definition of the processing that takes place as part of an action is termed *process model***

Advantages of OO-VHDL (Cont.)

- Benefit: No Compatibility Problems between Models
- Enables: Interoperability among object components

Any object can send/receive from any other object.
Object communication protocol implicitly defined.



(2) Dependability Outline

- Dependability Modeling
  - Errors and faults
  - Definitions
  - Need
  - Additional metrics
  - Evaluation metrics
  - Analytical techniques
  - Simulation-based techniques

Errors and Faults

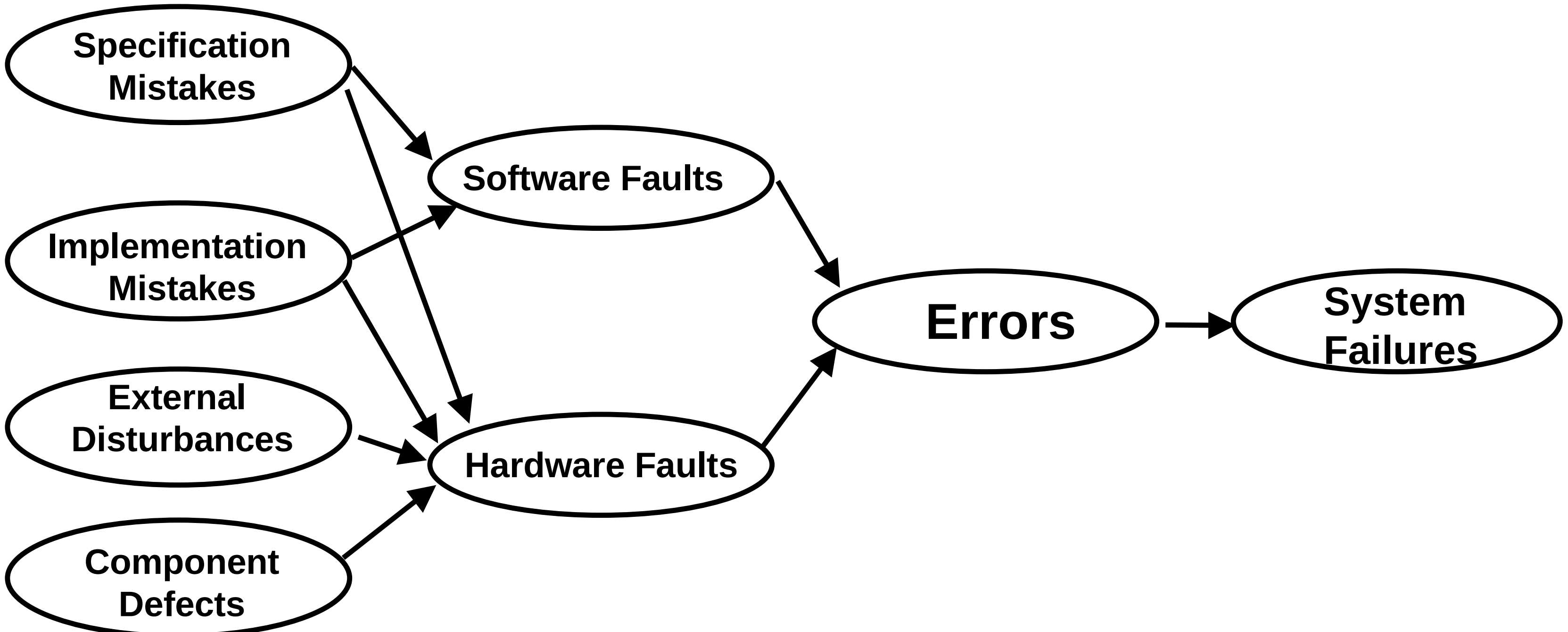
Physical Universe	Informational Universe	External Universe
FAULT →	ERROR →	FAILURE

- **Fault: Is a physical defect, imperfection, or flaw that occurs within some hardware or software component**
- **Error: Is a manifestation of a fault - deviation from accuracy or correctness**
- **Failure: Is the non-performance of some action that is due or expected - Also the performance of some function in a subnormal quantity or quality**
- **Latent fault: Is one that is present in a system but has not yet produced an error**
  - **Fault Latency: Time between the occurrence of a fault and appearance of an error due to that fault**

Errors and Faults (Cont.)

- **Error Latency: Is the length of time between the occurrence of an error and the appearance of the resulting failure**

Cause-Effect Relationships



[Johnson89]

Dependability Modeling Definitions (Cont.)

Safety

- The probability that a system will either perform its functions correctly (reliability) or will discontinue its functions in a manner that does not disrupt the operation of other systems or compromise the safety of any components associated with the system
- Provides a measure of fail-safe capability of a system

Performability

- The probability that the system performance will be at, or above, some level L at the instant of time t
- Used as a measure of performance of a system when there are failures in the system
- Graceful degradation: Is the ability of a system to automatically decrease its level of performance to compensate for hardware failures and software errors

Dependability Modeling Definitions (Cont.)

Maintainability

- The probability that a failed system will be restored to an operational state within a specified period of time t
- A measure of the ease with which a system can be repaired once it has failed

Testability

- The ability to test for certain attributes in a system
- Describes the ease with which certain tests can be performed

Dependability Modeling Definitions (Cont.)

Dependability

Is the quality of service that a particular system provides

Includes reliability, availability, safety, maintainability, performability, and testability

Currently, reliability is the main concern, as it is the most tractable and important concern in dependability modeling

Need for Dependability Modeling

- Because of increasing complexity of digital systems, systems have become less reliable
  - Long-life applications: Maintainability, safety, and performability become important
  - Critical-computation applications: Need high reliability
  - Maintenance postponement applications: Maintenance is costly, need performability
  - High-availability applications: Banking applications
- Recently, digital systems have become cheap; so can afford to add redundant components to improve reliability and, to some extent, other aspects of dependability

Additional Metrics

- **Failure rate**
  - **The expected number of failures of a type of device or system in a given period of time**

$$\mathbf{z(t) = -\frac{dR(t)}{dt}\frac{1}{R(t)}}$$

- **Mean time to failure *(MTTF)***
  - **Is the expected time that a system will operate before the first failure occurs**

$$MTTF = \frac{1}{N}\sum_{i=1}^{N} t_i = \int_0^{\infty} R(t)dt$$

- **Mission time *MT[r]*: Is the time at which the reliability of a system falls below a level r**
  - **Systems can be compared by the ratio of mission times**

Additional Metrics (Cont.)

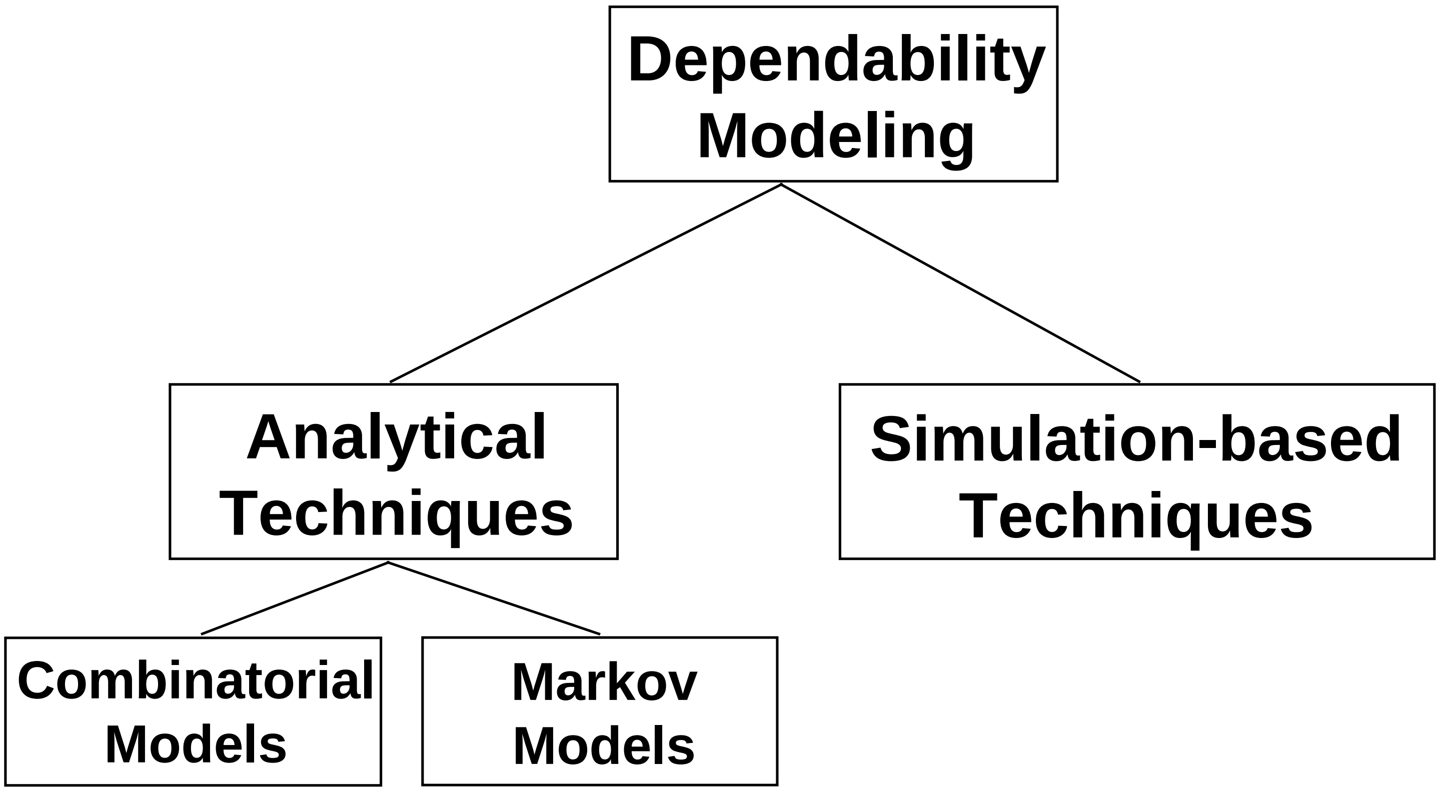
- **Mean time to repair *(MTTR)***
  - **Average time required to repair a system**

$$MTTR = \frac{1}{N}\sum_{i=1}^{N} t_i$$

  - **MTTR is given by repair rate μ, which is the average number of repairs that occur per time period, MTTR=1/μ**
- **Mean time between failures, *MTBF = MTTF+MTTR***
- **Fault coverage**
  - **Measure of a system's ability to perform fault detection, fault containment, and/or fault recovery**
  - **Ex: Fault detection coverage factor**

$$= \frac{\text{number of faults that can be detected}}{\text{total number of faults}}$$

Analytical Techniques



Analytical Techniques (Cont.)

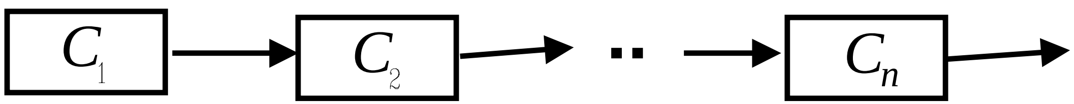
- **Advantages**
  - **For accurate modeling assumptions, the solution is accurate and deterministic**
  - **Can be solved in continuous time with high accuracy**
- **Disadvantages**
  - **Based on models of the components, so details of the system behavior might be lost in the modeling assumptions**
  - **State-based models are difficult to solve**
    - **Exponential time and space complexity**

Analytical Techniques (Cont.)

- **Combinatorial models**
  - **Are difficult to construct and the reliability expressions are often very complex**
  - **Difficult to incorporate fault coverage**
  - **Process of repair is difficult to incorporate**
- **Markov models**
  - **Rely on two mechanisms to describe system:**
    - **System state: Represents all that must be known to describe the system at any given instant of time. It represents a distinct combination of faulty and fault-free modules**
    - **State transitions: Described as probabilities that transitions will occur between adjacents states**
  - **Set of simultaneous differential equations provides accurate solutions for stated transition probabilities**

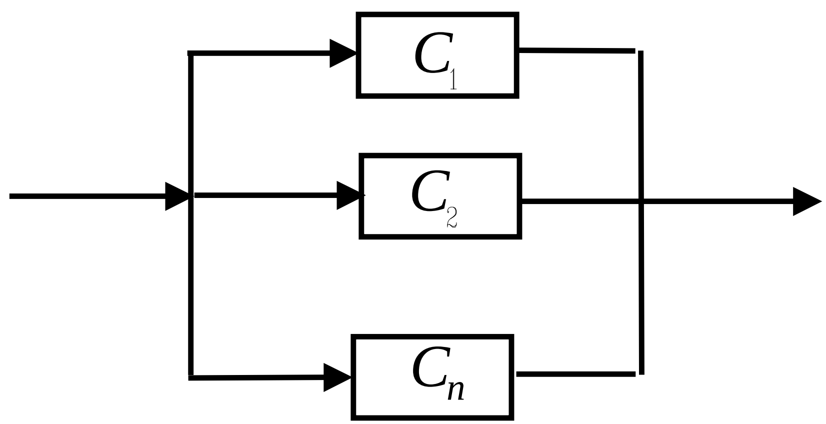
Combinatorial Models

- Two types of connections
  - Series: the system contains no redundancy



$$R_{series}(t) = \prod_{i=1}^{n} R_i(t)$$

  - Parallel: only one of the elements is required for the system to function



$$1 - R_{parallel}(t) = [1 - R_1(t)][1 - R_2(t)]\ldots[1 - R_n(t)]$$

Semi-Markov Unreliability Range Evaluator (SURE)

- Calculates the upper and lower bounds on the probability of failed state of a Markov model
  - Requires solution of a set of coupled differential equations
- Computes probabilities using algebraic formulas, large state spaces can be accommodated
- Based on
  - White's method: The means and variances of the recovery times are sufficient to obtain tight bounds on the probability of system failures
    - Useful in design studies in which properties of fast distributions are assumed
  - Lee's theorem:
    - Useful in analysis where experimental data is available

Simulation-based Techniques

- **Advantages**
  - **Flexible, no restrictions caused by computational complexity**
  - **Detailed, no modeling assumptions made**
  - **Arbitrary precision for a given simulation time**
- **Disadvantages**
  - **Low failure rates and high repair rates require large number of simulations**
  - **Reducing the size of the model, by taking into account only relevant components (importance sampling)**

Reliability Estimation System Testbed (REST)

- Software system designed by NASA to support
  - The hardware reliability analysis of complex fault-tolerant computer systems
  - Simulates failure modes and effect analysis (SFMEA) and automatically generates and analyzes a semi-Markov model of the system of interest
  - Calculates upper and lower bounds on the probability of encountering a failure state and a summary of conditions under which those failures occur
- Main components
  - REST modeling language, RML
  - Translators
  - An X Window front window

REST

- **REST modeling language (RML)**
  - **Uses “modules” to describe simple components and complex systems of such components**
  - **Types of module variables**
    - **State variables**
    - **Rate variables**
    - **Relation variables**
    - **Event declarations**
- **System definition starts by creating a list of all the module types to be found in the system**
- **Model analysis portion**
  - **Takes the system description and repeatedly transforms the system state in accordance with rules given with module type definition**

REST (Cont.)

- **REST translator**
  - **Maps all the state variables implicit in the declaration of module variables onto a global state vector**
  - **Requires local variables to be handled by the user explicitly**
- **REST run-time system**
  - **Responsible for all analysis, and sequencing of routines declared in the RML modules**

(3) Functional Modeling

- **Describes the function of the hardware/software system kernel, but not the functionality or timing of the interface.**
  - **Uses the information about the structure of the component to be modeled**
- **Examples**
  - **MAT2DSP**
  - **Ptolemy**

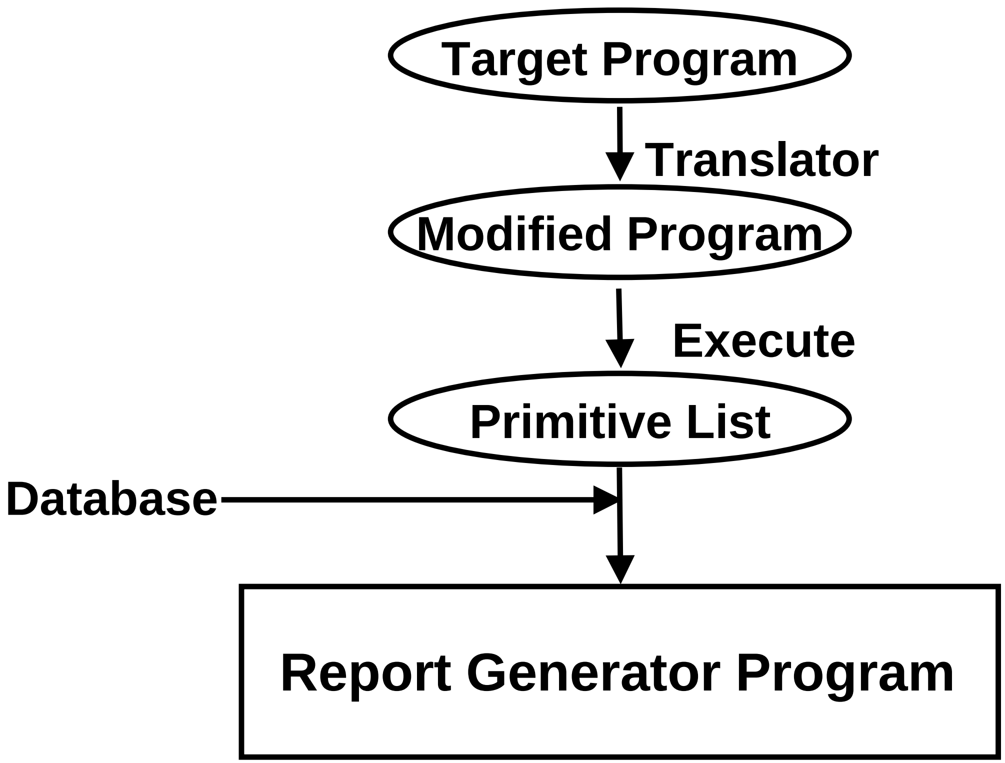
Functional Modeling with MATLAB

- **MATLAB is a high-level signal processing software package built from a set of primitive functions**
  - **Vector-vector and vector-matrix multiplies, FFT, convolution, filtering**
  - **Operating on data vectors or arrays**
- **The algorithm chosen to solve a particular problem has a tremendous impact on the complexity and cost of the final implementation**
- **MATLAB attempts to bridge the gap between algorithm development and its hardware/software implementation**
- **Multiple algorithmic solutions for any problem have different cost/performance trade-offs**

MAT2DSP

- Developed by University of California, Davis
- Estimates the implementation requirements of algorithms specified in the form of a MATLAB program
- Future versions will take into account more detailed information related to dataflow, program overhead, and data transfer times
- Target program
  - MATLAB program that implements a given signal/image processing algorithm
  - MAT2DSP program operates on this program and produces one of several user-selected reports which contains information about the computational requirements of the algorithm and an estimate of its runtime on a user-specified processor or a mix of processors

MAT2DSP



MATLAB Program

MATLAB Program

Number and types of computations performed by the target program

Run-time of the target program on a user-specified processing hardware

- Different types of reports of varying levels of complexity can be generated based on the data contained in the primitive list and the database

Ptolemy

- System-level design framework
  - Covers higher levels of system specifications as well as lower level of system description
    - Implements heterogeneous embedded systems
    - Allows mixing models of computation and implementation languages
  - Provides graphical specification of system parameters and mathematical models of systems
  - Supports hierarchy using object-oriented principles of polymorphism and information hiding in C++
  - Provides capability for interaction between different domains

Ptolemy (Cont.)

- Special Features
  - Graphical interface (pigi, Ptolemy interactive graphical interface), based on vem, a graphical editor
    - Animation and visualization
  - Multidimensional signal processing dataflow models
  - higher order functions
  - Silage and VHDL code generation
  - Interfaces to other design tools; e.g., MATLAB and Hyper
- Applications
  - Signal processing, telecommunications, wireless communications, network design
  - Parallel processing, real-time systems, hardware/software codesign

Ptolemy Capabilities

- **Design of signal processing and communication systems**
  - **Specifying, designing, and simulating algorithms to synthesize hardware and software**
  - **Providing techniques for dataflow modeling of algorithms**
  - **Managing regularity in dataflow graphs using higher order functions**
  - **Synthesizing embedded software from dataflow models**
  - **Scheduling dataflow graphs on uniprocessors and multiprocessors efficiently**
  - **Supporting hardware/software partitioning of dataflow graphs**
- **Parallelizing algorithms**
- **Prototyping real-time systems**

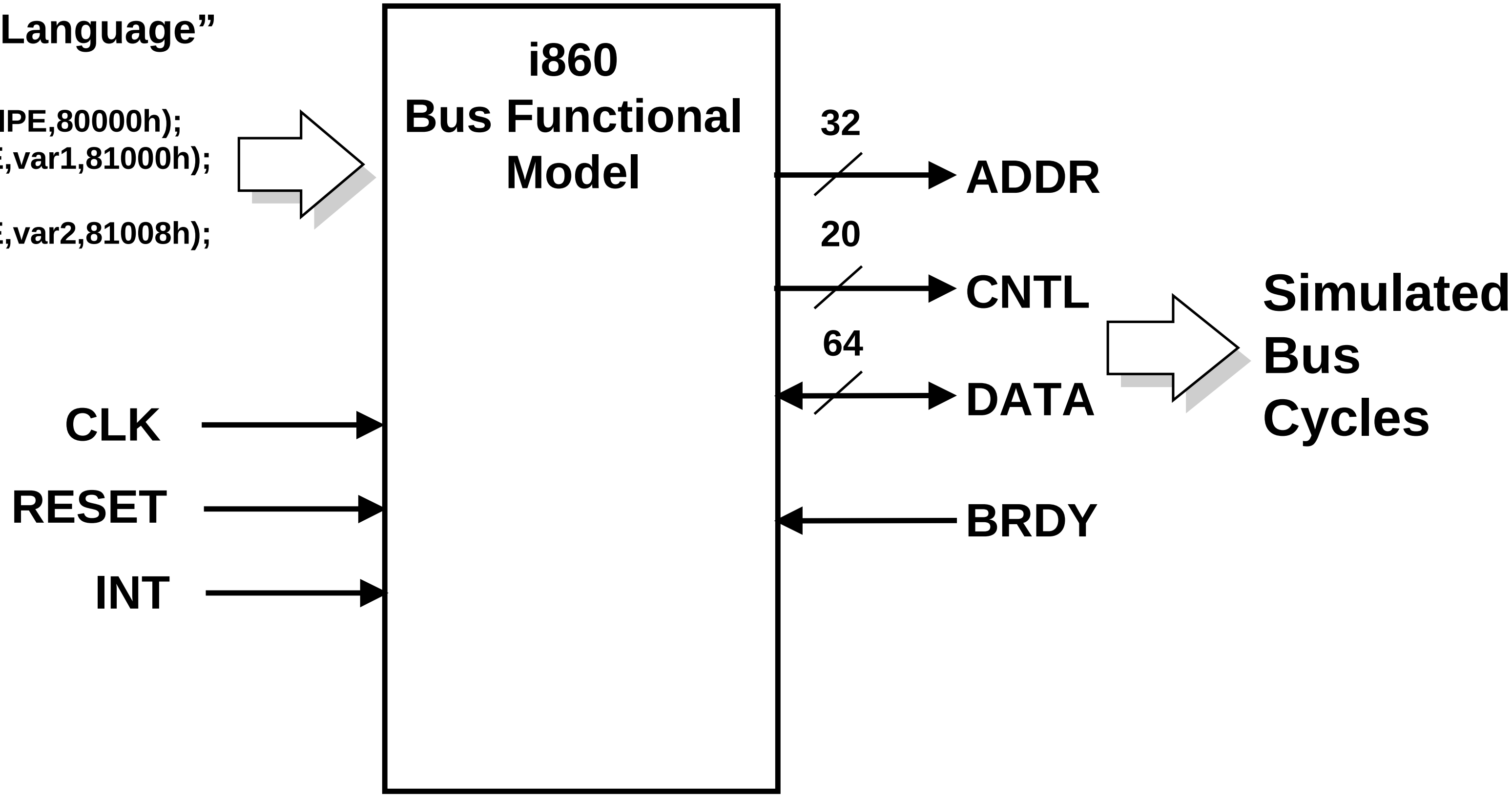
(5) Bus Functional Models

- Interface models that describe the functionality and timing of the interface (e.g. of processors or memories) to a bus
- Descriptions are at network level of abstraction
- Only enough detail is included to model external behavior of the device - internal state is not modeled
  - E.g., processor model will perform bus cycles for memory read/write, but will not execute actual code
- Internal control language is sometimes used to allow user to program different bus cycles
- Obtained by
  - Commercial suppliers (Logic Modeling Group, Synopsys, Inc.)
  - Bus functional model generators (OmniView)

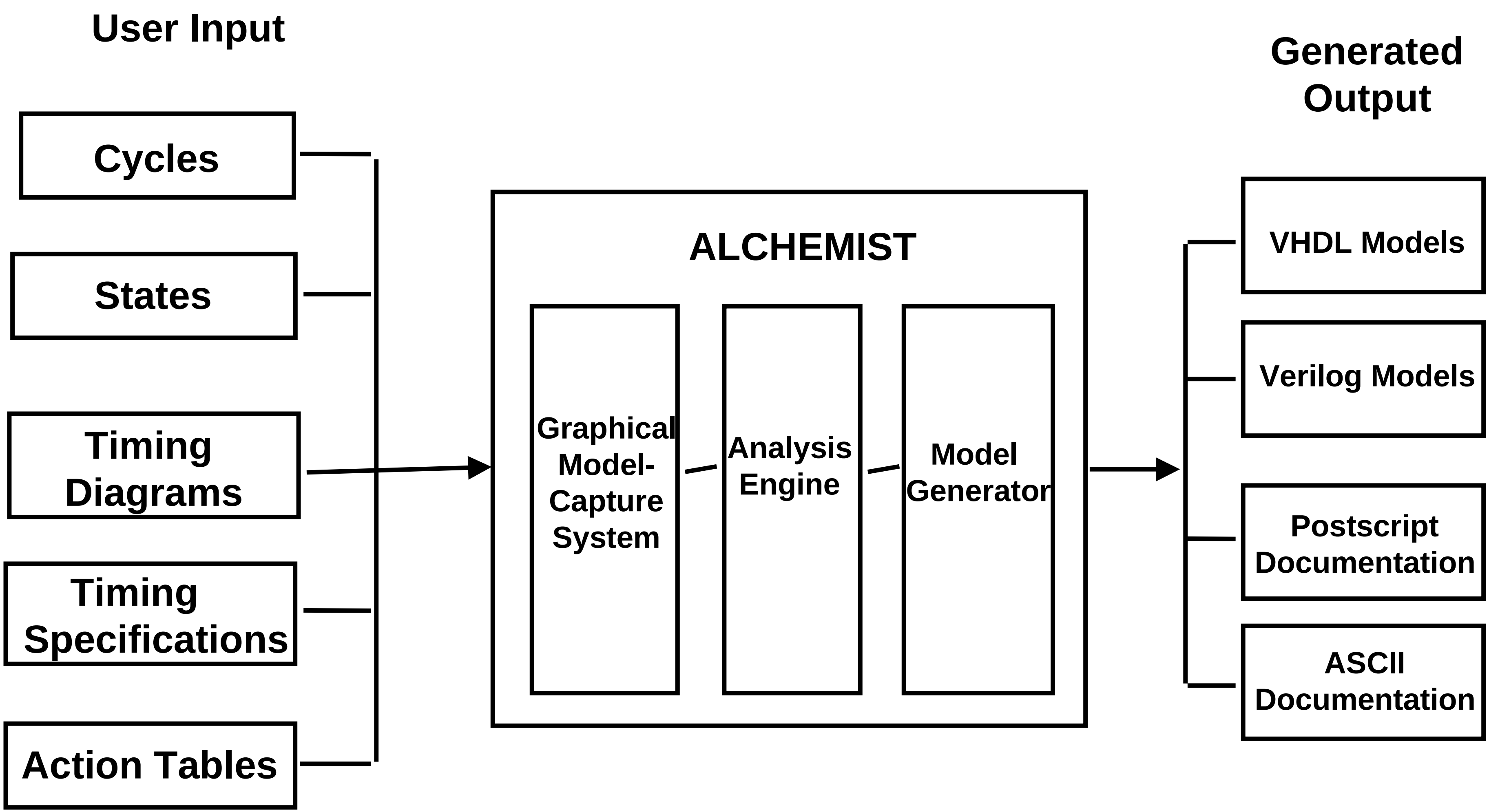
Bus Functional Models Example

Model Specific
"Programming Language"

cache_read(inst,8,PIPE,80000h);
nc_read(data,8,PIPE,var1,81000h);
nop
nc_read(data,8,PIPE,var2,81008h);



Bus Functional Model Generation - ALCHEMIST



Summary

- The general forms of system level modeling were introduced
  - Performance Modeling
  - Dependability Modeling
  - Functional Modeling
  - Executable Requirements
  - Bus Functional Models
- The goals of each type of modeling and where it fits into the design process was discussed
- Examples tools for each type of modeling were presented