

EECS 222A: System-on-Chip Description and Modeling Lecture 1

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 1: Overview

- Course administration
 - Overview
 - Contents
 - Schedule
 - Assignments
- Introduction to System-on-Chip design
 - Levels of abstraction
 - System design flow
 - Models of computation
 - System-level description languages
 - Computation, communication, IP

Course Administration

- Course web pages at <http://eee.uci.edu/09f/18415/>
 - Instructor information
 - Course description and policies
 - Objectives and outcomes
 - Contents and schedule
 - Resources and communication
 - Assignments

Introduction to SoC Design

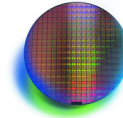
- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC Design Flow
- Models of Computation
- System-Level Description Languages
- Computation vs. Communication
- Intellectual Property

System-on-Chip Design

- Embedded systems are everywhere...

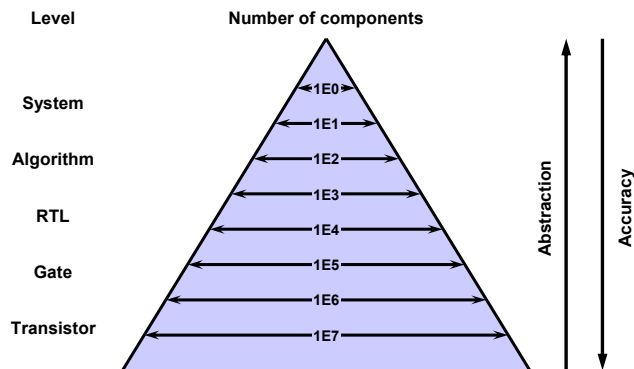


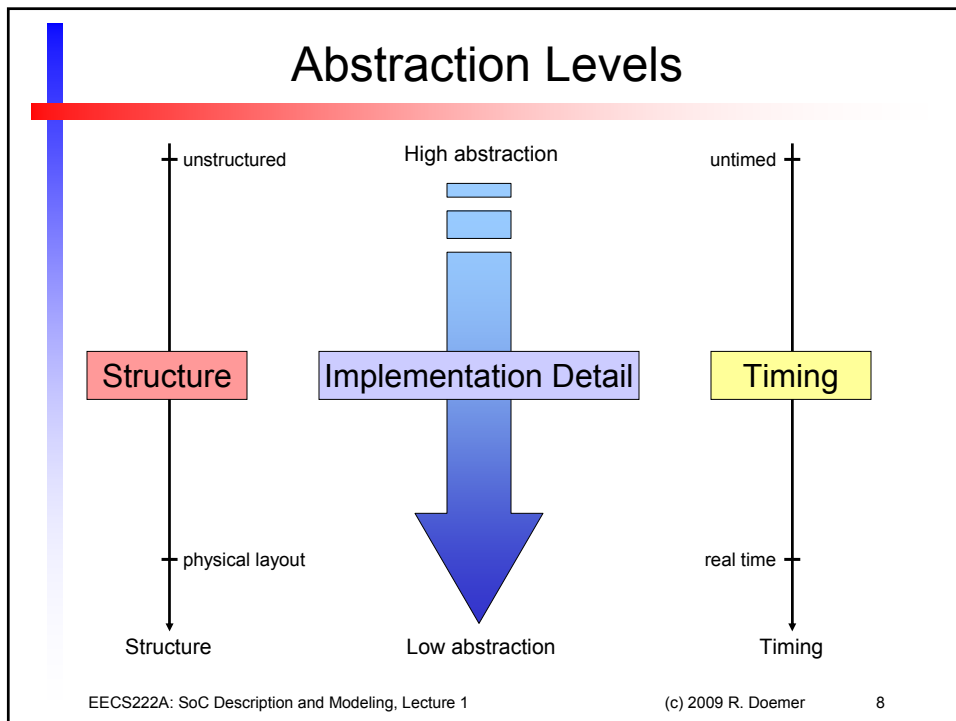
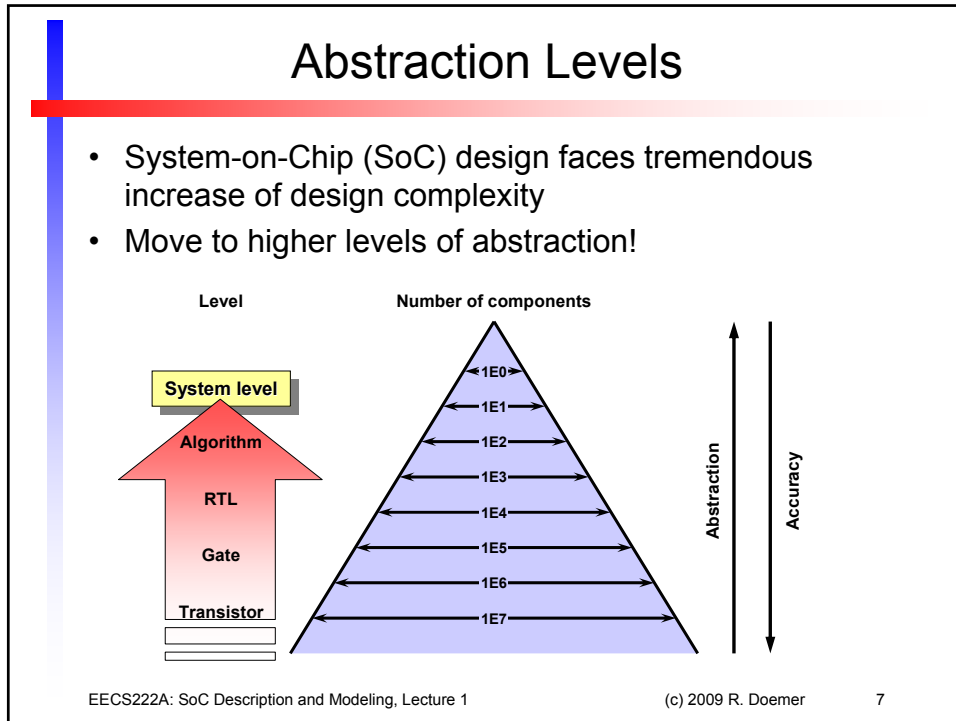
- Deep sub-micron technology enables System-on-Chip (SoC)

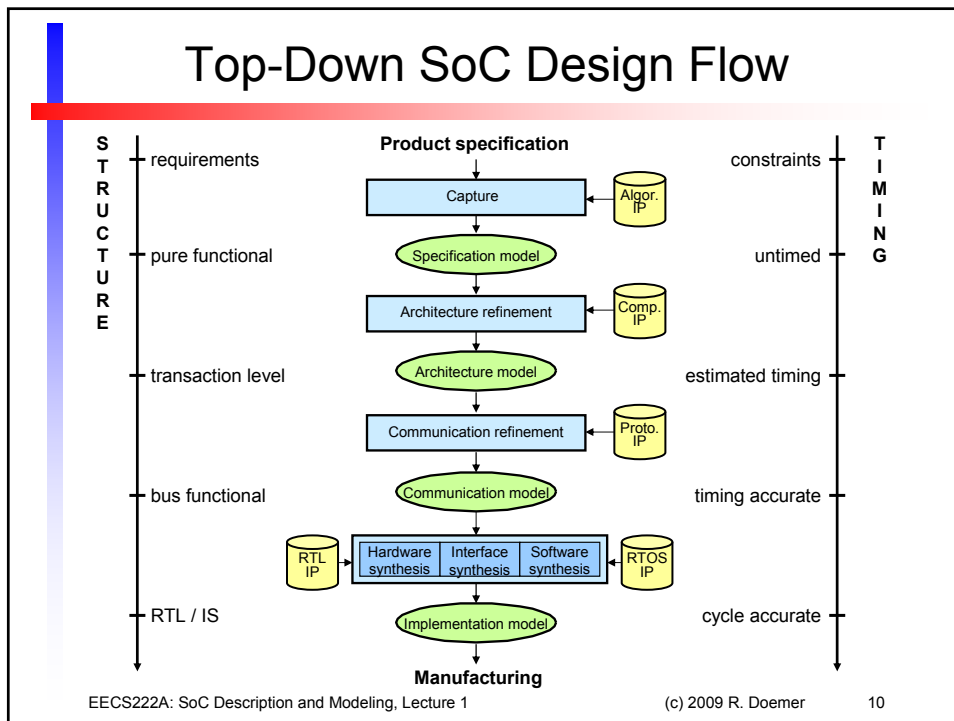
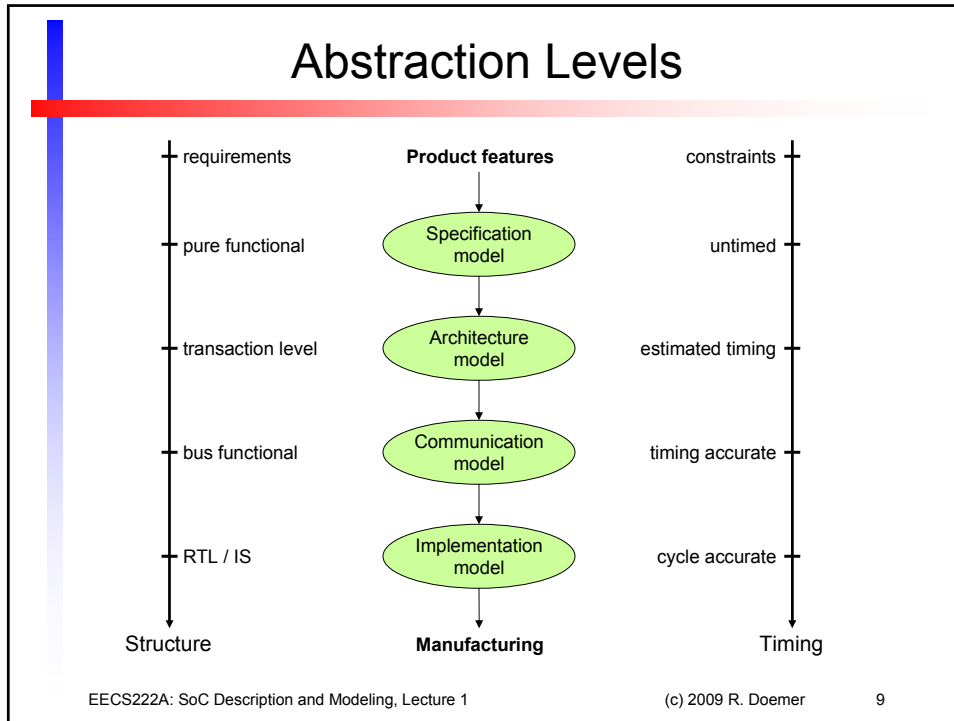


Abstraction Levels

- System-on-Chip (SoC) design faces tremendous increase of design complexity







Models of Computation

- Computational Model
 - Formal, abstract description of a system
 - Various degrees of
 - supported features
 - complexity
 - expressive power
- Examples
 - Evolution process from FSM to PSM
 - Finite State Machine (FSM)
 - FSM with Data (FSMD)
 - Super-state FSMD
 - ...
 - Program State Machine (PSM)

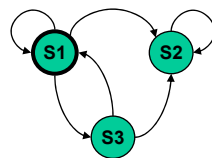
EECS222A: SoC Description and Modeling, Lecture 1

(c) 2009 R. Doemer

11

Models of Computation

- Finite State Machine (FSM)
 - Basic model for describing control
 - States and state transitions
 - $FSM = \langle S, I, O, f, h \rangle$
 - Two types:
 - Mealy-type FSM (input-based)
 - Moore-type FSM (state-based)



FSM model

EECS222A: SoC Description and Modeling, Lecture 1

(c) 2009 R. Doemer

12

Models of Computation

- Finite State Machine (FSM)
- Data Flow Graph (DFG)
 - Basic model for describing computation
 - Directed graph
 - Nodes: operations
 - Arcs: dependency of operations

The diagram shows a Data Flow Graph (DFG) model. It consists of six yellow circular nodes labeled Op1 through Op6. Op1, Op2, and Op3 are at the top level, each with a downward arrow indicating an input. Op1 has arrows pointing to Op5 and Op6. Op2 has an arrow pointing to Op4. Op3 has an arrow pointing to Op4. Op4 has arrows pointing to Op5 and Op6. Op5 and Op6 have downward arrows indicating outputs.

DFG model

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 13

Models of Computation

- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
 - Combined model for control and computation
 - FSMD = FSM + DFG
 - Implementation: controller plus datapath

The diagram shows a Finite State Machine with Data (FSMD) model. It features a central state machine with three green circular states: S1, S2, and S3. S1 and S2 are at the top, and S3 is below them. S1 has a self-loop and an arrow pointing to S2. S2 has a self-loop and an arrow pointing to S3. S3 has a self-loop and an arrow pointing back to S1. Three yellow boxes represent datapaths. The top-left box contains Op1 and Op2. The bottom-left box contains Op1, Op2, and Op3. The right box contains Op1, Op2, Op3, Op4, Op5, and Op6, which is identical to the DFG model shown in the previous slide. Red arrows indicate control signals from the states to the datapaths: S1 controls the top-left box, S2 controls the bottom-left box, and S3 controls the right box.

FSMD model

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 14

Models of Computation

- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
- Super-State FSM with Data (SFSMD)
 - FSMD with complex, multi-cycle states
 - States described by procedures in a programming language

```
a = a + b;
c = c + d;
```

```
a = 42;
while (a<100)
{ b = b + a;
  if (b > 50)
    c = c + d;
  a = a + c;
}
```

SFSMD model

```
a = 42;
b = a * 2;
for(c=0; c<100; c++)
{ b = c + a;
  if (b < 0)
    b = -b;
  else
    b = b + 1;
  a = b * 10;
}
```

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 15

Models of Computation

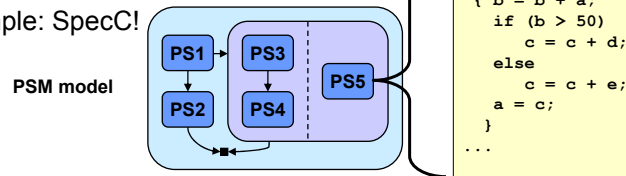
- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
- Super-State FSM with Data (SFSMD)
- Hierarchical Concurrent FSM (HCFSM)
 - FSM extended with hierarchy and concurrency
 - Multiple FSMs composed hierarchically and in parallel
 - Example: Statecharts

HCFSM model

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 16

Models of Computation

- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
- Super-State FSM with Data (SFSMD)
- Hierarchical Concurrent FSM (HCFSM)
- Program State Machine (PSM)
 - HCFSMD plus programming language
 - States described by procedures in a programming language
 - Example: SpecC!



EECS222A: SoC Description and Modeling, Lecture 1

(c) 2009 R. Doemer

17

System-Level Description Languages

- Goals and Requirements
 - Formality
 - Formal syntax and semantics
 - Executability
 - Validation through simulation
 - Synthesizability
 - Implementation in HW and/or SW
 - Support for IP reuse
 - Modularity
 - Hierarchical composition
 - Separation of concepts
 - Completeness
 - Support for all concepts found in embedded systems
 - Orthogonality
 - Orthogonal constructs for orthogonal concepts
 - Simplicity
 - Minimality

EECS222A: SoC Description and Modeling, Lecture 1

(c) 2009 R. Doemer

18

System-Level Description Languages

- Requirements supported by existing languages

	C	C++	Java	VHDL	Verilog	HardwareC	Statecharts	SpecCharts	SpecC
Behavioral hierarchy	○	○	○	○	○	○	○	●	●
Structural hierarchy	○	○	○	●	●	●	○	○	●
Concurrency	○	○	◐	●	●	●	●	●	●
Synchronization	○	○	◐	●	●	●	●	●	●
Exception handling	◐	●	●	○	●	○	◐	●	●
Timing	○	○	○	●	●	◐	◐	◐	●
State transitions	○	○	○	○	○	○	●	●	●
Composite data types	●	●	●	●	◐	○	○	●	●

○ not supported ◐ partially supported ● supported

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 19

The Future of Design Languages

...can be predicted best from the past!

Past

Present

Future

Dates indicate year of publication of first reference book and/or year of invention. Please correct me if I'm wrong!

Panel at MEMOCODE 2009, Cambridge, MA (c) 2009 R. Doemer 20

The Future of Design Languages

- ConcurrnC
 - Formal Model of Computation (MoC)
 - Representable in C-based SLDLS
 - SystemC
 - SpecC

Panel at MEMOCODE 2009, Cambridge, MA (c) 2009 R. Doemer 21

System-Level Description Languages

- Examples in use today
 - C/C++
 - ANSI standard programming languages, software design
 - traditionally used for system design because of practicality, availability
 - SystemC
 - C++ API and library
 - initially developed at UCI, supported by Open SystemC Initiative
 - SpecC
 - C extension
 - developed at UCI, supported by SpecC Technology Open Consortium
 - SystemVerilog
 - Verilog with C extensions
 - Matlab
 - specification and simulation in engineering, algorithm design
 - UML
 - unified modeling language, software specification, graphical
 - SDL
 - telecommunication area, standard by ITU, used in COSMOS
 - SLDL
 - formal specification of requirements, not executable
 - etc.

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 22

System-Level Description Languages

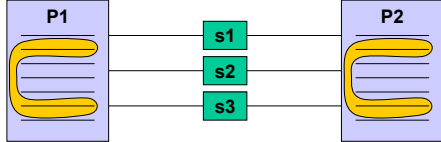
- Examples in use today and **coverage in this course**
 - C/C++
 - ANSI standard programming languages, software design
 - traditionally used for system design because of practicality, availability
 - **SystemC**
 - C++ API and library
 - initially developed at UCI, supported by Open SystemC Initiative
 - **SpecC**
 - C extension
 - developed at UCI, supported by SpecC Technology Open Consortium
 - SystemVerilog
 - Verilog with C extensions
 - Matlab
 - specification and simulation in engineering, algorithm design
 - **UML**
 - unified modeling language, software specification, graphical
 - SDL
 - telecommunication area, standard by ITU, used in COSMOS
 - SLDL
 - formal specification of requirements, not executable
 - etc.

Separation of Concerns

- Fundamental Principle in Modeling of Systems
- Clear *separation of concerns*
 - address separate issues independently
- System-Level Description Language (SLDL)
 - orthogonal concepts
 - orthogonal constructs
- System-level Modeling
 - Computation
 - encapsulated in modules / behaviors
 - Communication
 - encapsulated in channels

Computation vs. Communication

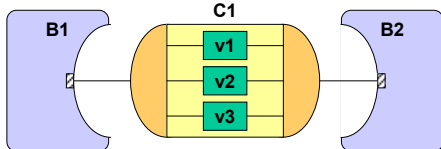
- Traditional model
 - Processes and signals
 - Mixture of computation and communication
 - Automatic replacement impossible



EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 25

Computation vs. Communication

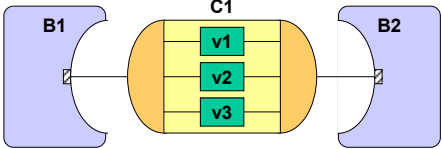
- Traditional model
 - Processes and signals
 - Mixture of computation and communication
 - Automatic replacement impossible
- SpecC model
 - Behaviors and channels
 - Separation of computation and communication
 - Plug-and-play



EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 26

Computation vs. Communication

- Protocol Inlining
 - Specification model
 - Exploration model

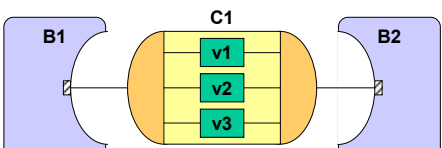


- Computation in behaviors
- Communication in channels

EECS222A: SoC Description and Modeling, Lecture 1
(c) 2009 R. Doemer
27

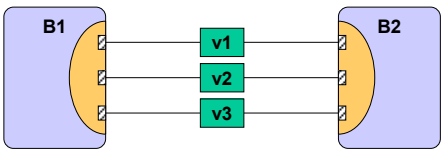
Computation vs. Communication

- Protocol Inlining
 - Specification model
 - Exploration model



- Computation in behaviors
- Communication in channels

- Implementation model




- Channel disappears
- Communication inlined into behaviors
- Wires exposed

EECS222A: SoC Description and Modeling, Lecture 1
(c) 2009 R. Doemer
28

Intellectual Property (IP)

- Computation IP: Wrapper model

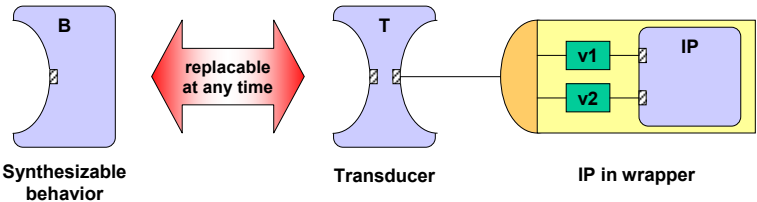


The diagram illustrates the wrapper model for computation IP. On the left, a blue block labeled 'B' with a notch on its left side is labeled 'Synthesizable behavior'. On the right, a yellow block labeled 'IP in wrapper' contains a purple block labeled 'IP' connected to two green blocks labeled 'v1' and 'v2'. The wrapper has a semi-circular end on the left.

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 29

Intellectual Property (IP)

- Computation IP: Wrapper model



The diagram illustrates the wrapper model for computation IP, highlighting the replaceability of the behavior. On the left, a blue block labeled 'B' with a notch on its left side is labeled 'Synthesizable behavior'. In the middle, a blue block labeled 'T' with a notch on its left side and a protrusion on its right side is labeled 'Transducer'. A red double-headed arrow between 'B' and 'T' is labeled 'replacable at any time'. On the right, a yellow block labeled 'IP in wrapper' contains a purple block labeled 'IP' connected to two green blocks labeled 'v1' and 'v2'. The wrapper has a semi-circular end on the left.

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 30

Intellectual Property (IP)

- Computation IP: Wrapper model

Synthesizable behavior Transducer IP in wrapper

- Protocol inlining with wrapper

before after

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 31

Intellectual Property (IP)

- Computation IP: Adapter model

Synthesizable behavior Transducer Adapter IP

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 32

Intellectual Property (IP)

- Computation IP: Adapter model

- Protocol inlining with adapter

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 33

Intellectual Property (IP)

- Communication IP: Channel with wrapper

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 34

Intellectual Property (IP)

- Communication IP: Channel with wrapper**

Virtual channel IP protocol channel in wrapper
- Protocol inlining with hierarchical channel**

before after

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 35

Intellectual Property (IP)

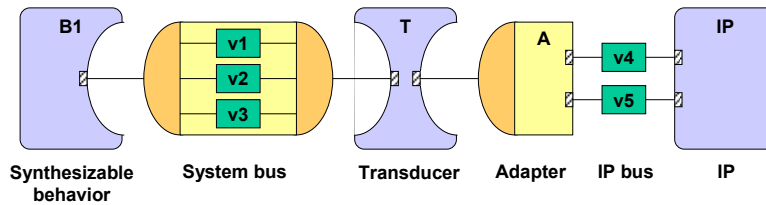
- Incompatible busses: Transducer insertion**

Synthesizable behavior System bus Transducer Adapter IP bus IP

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2009 R. Doemer 36

Intellectual Property (IP)

- Incompatible busses: Transducer insertion



- Protocol inlining with transducer

