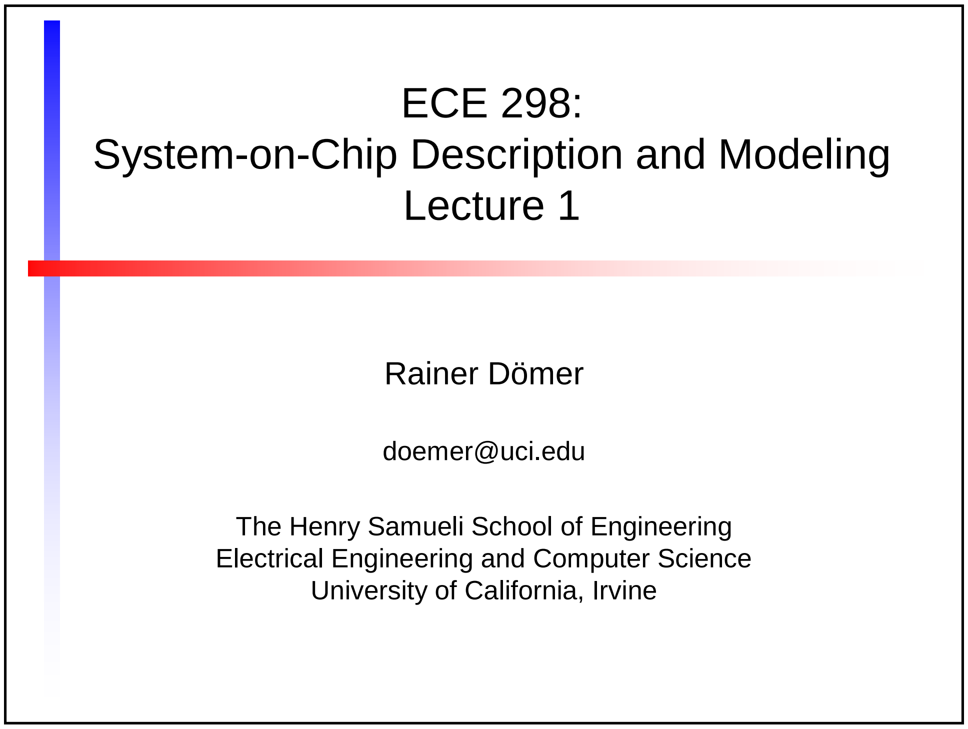
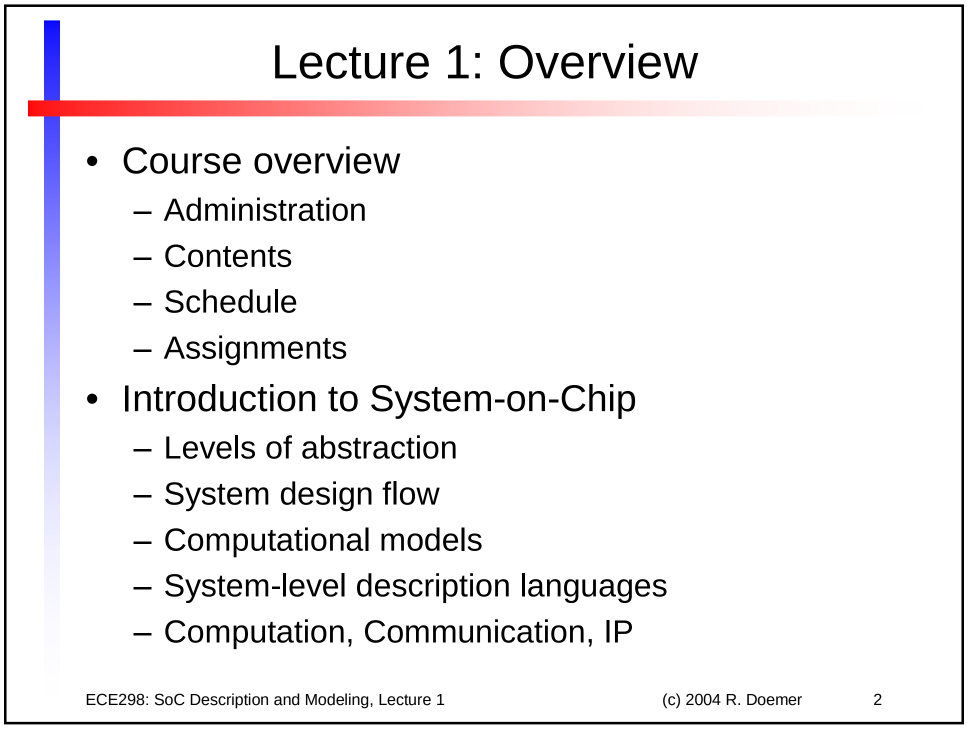


ECE 298: System-on-Chip Description and Modeling Lecture 1

Rainer Dömer

doemer@uci.edu

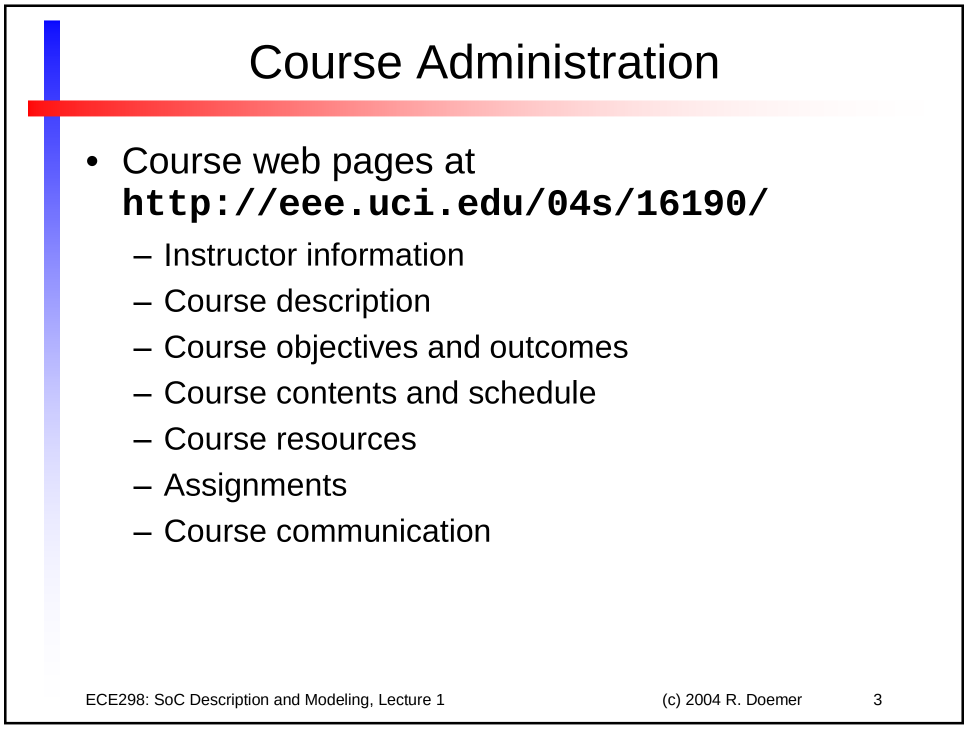
The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine



Lecture 1: Overview

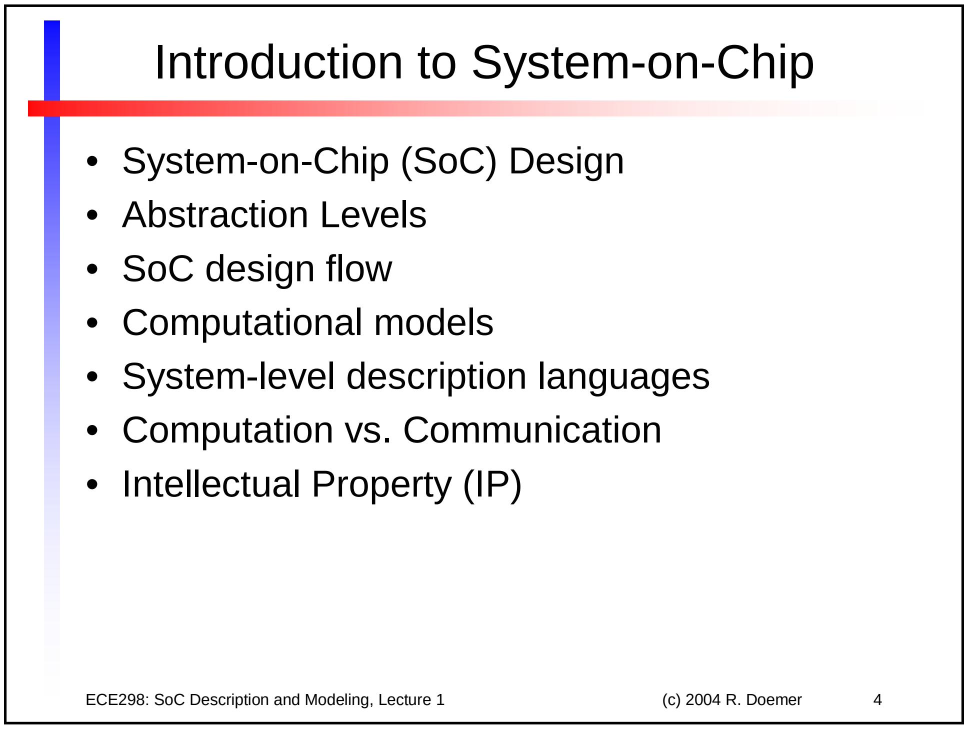
- Course overview
  - Administration
  - Contents
  - Schedule
  - Assignments
- Introduction to System-on-Chip
  - Levels of abstraction
  - System design flow
  - Computational models
  - System-level description languages
  - Computation, Communication, IP

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 2



Course Administration

- Course web pages at **`http://eee.uci.edu/04s/16190/`**
  - Instructor information
  - Course description
  - Course objectives and outcomes
  - Course contents and schedule
  - Course resources
  - Assignments
  - Course communication



Introduction to System-on-Chip

- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- Computational models
- System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

Introduction to System-on-Chip

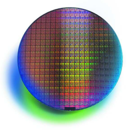
- ➢ System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- Computational models
- System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 5

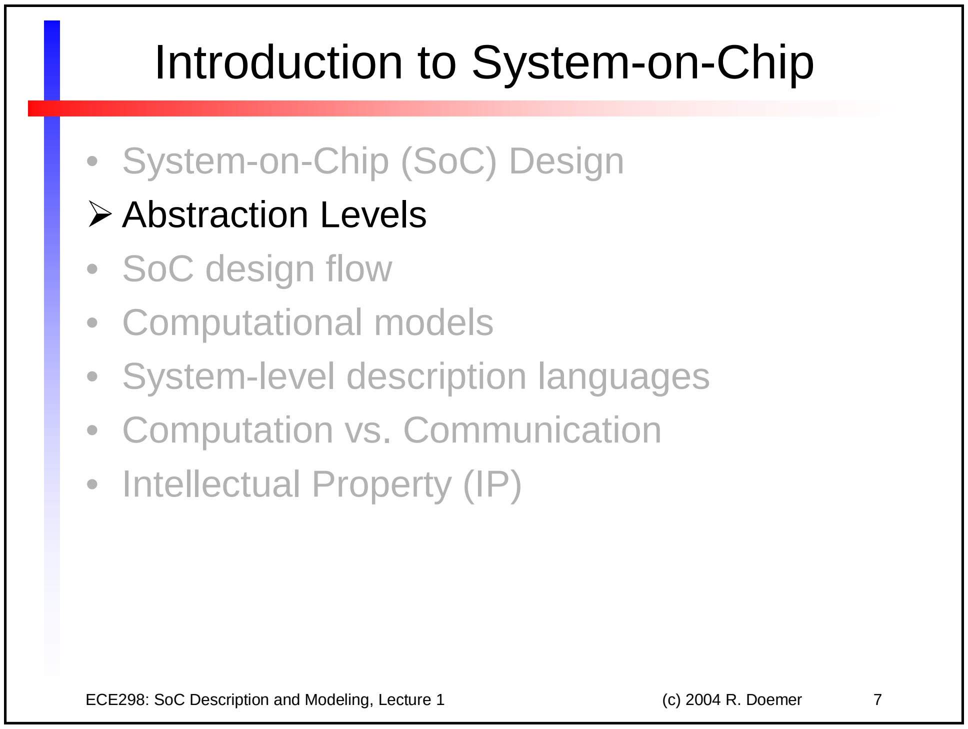
System-on-Chip Design

- Embedded systems are everywhere…

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



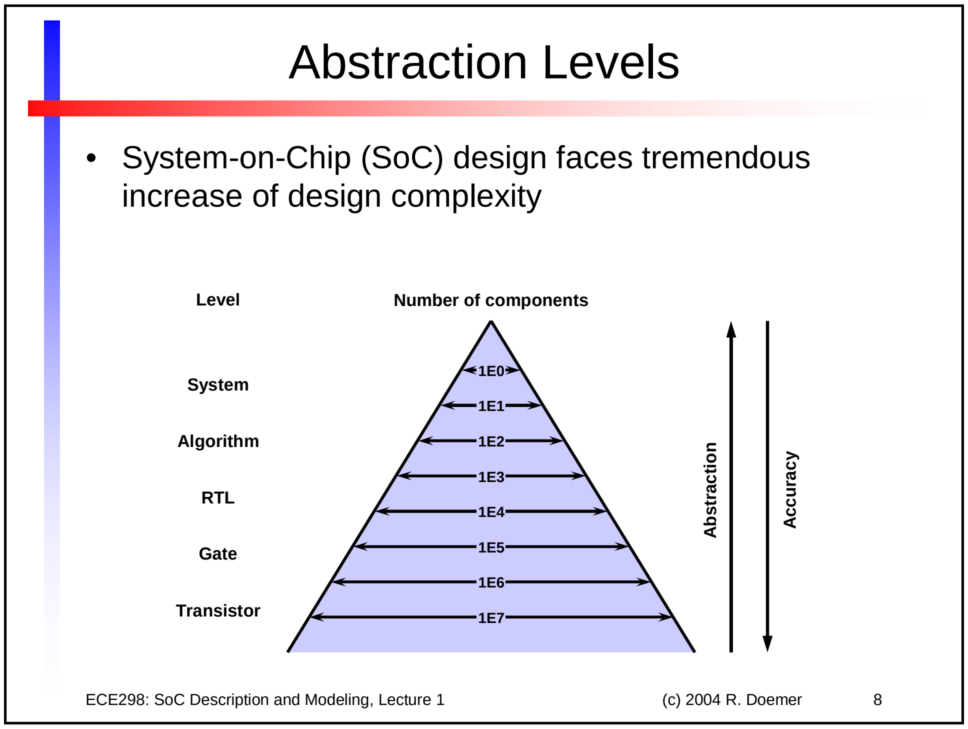
ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 6



Introduction to System-on-Chip

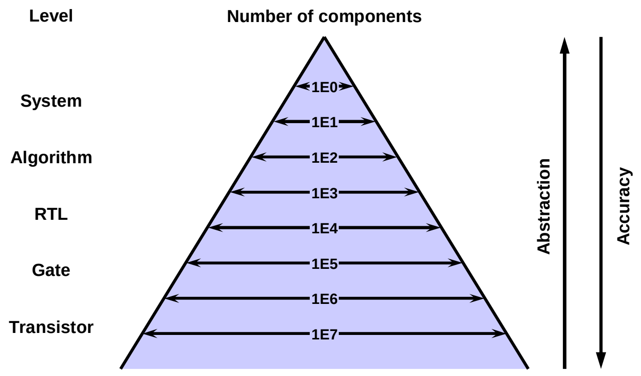
- System-on-Chip (SoC) Design
- ➢ Abstraction Levels
- SoC design flow
- Computational models
- System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 7

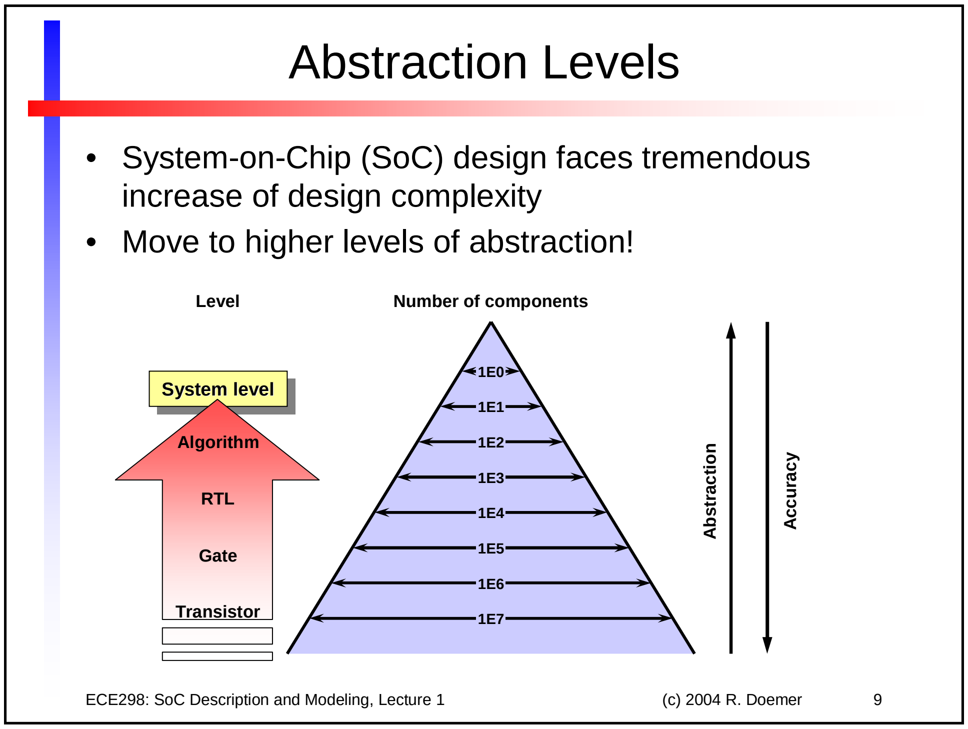


Abstraction Levels

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

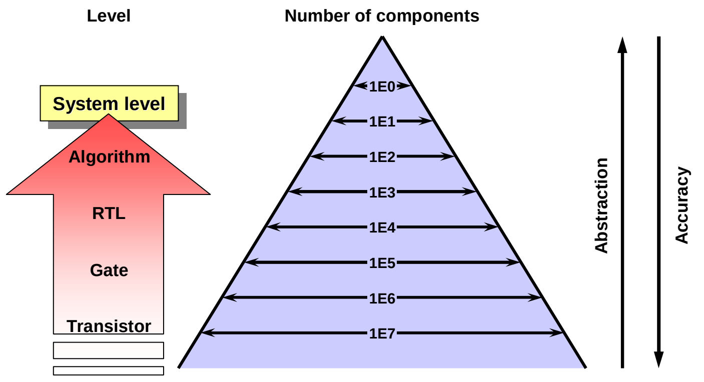


ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 8

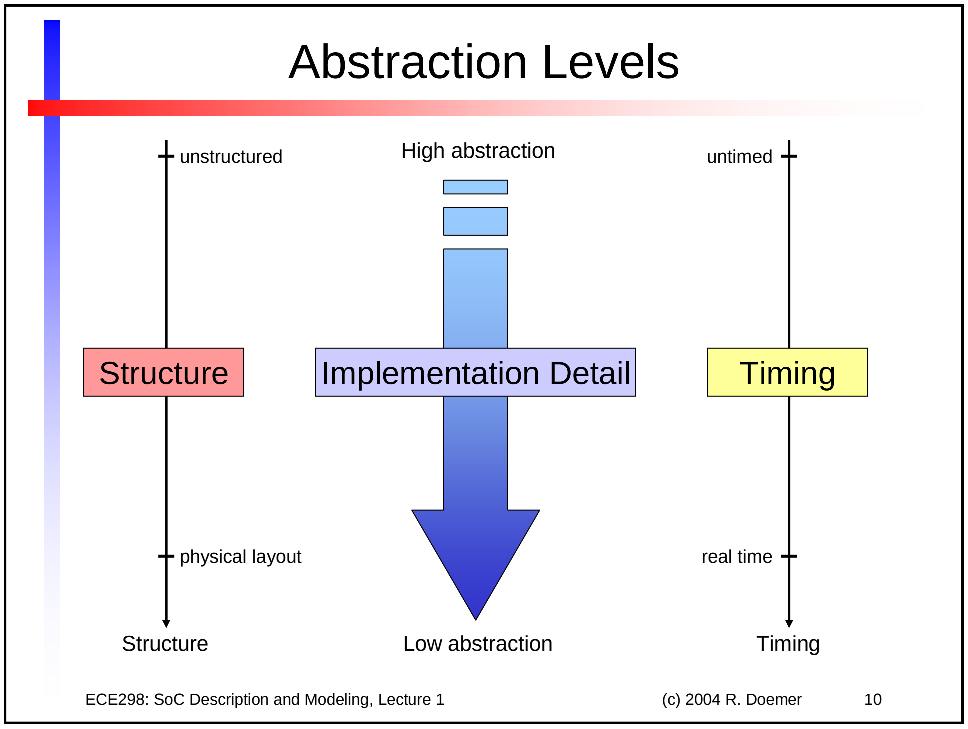


Abstraction Levels

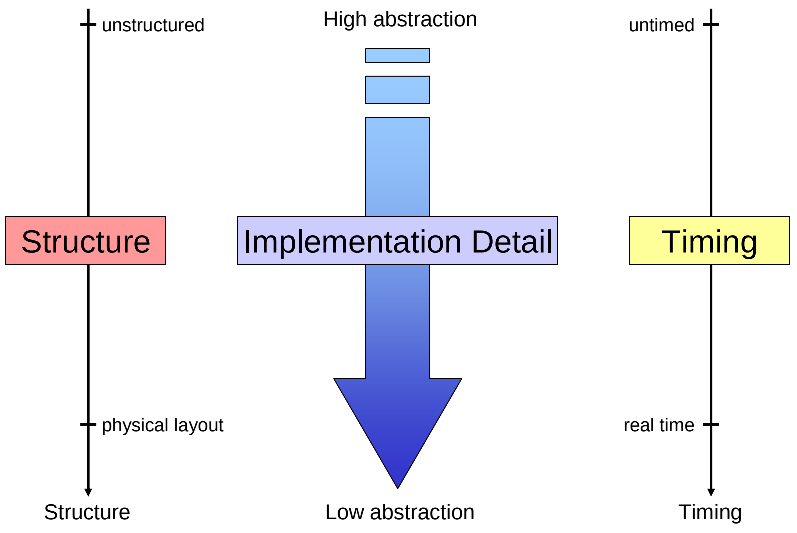
- System-on-Chip (SoC) design faces tremendous increase of design complexity
- Move to higher levels of abstraction!



ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 9

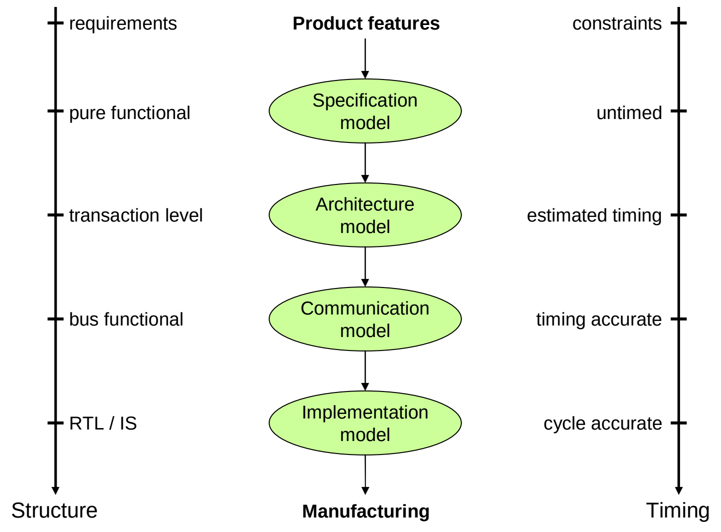


Abstraction Levels



ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 10

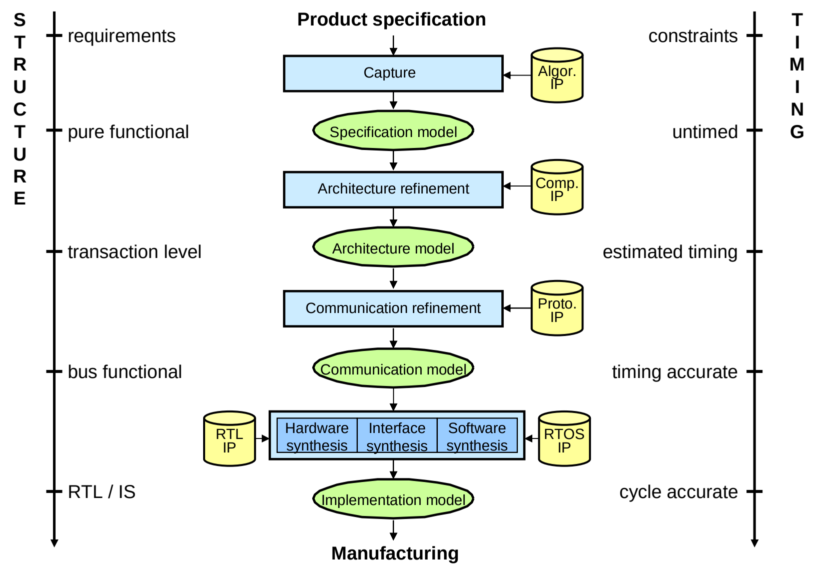
Abstraction Levels



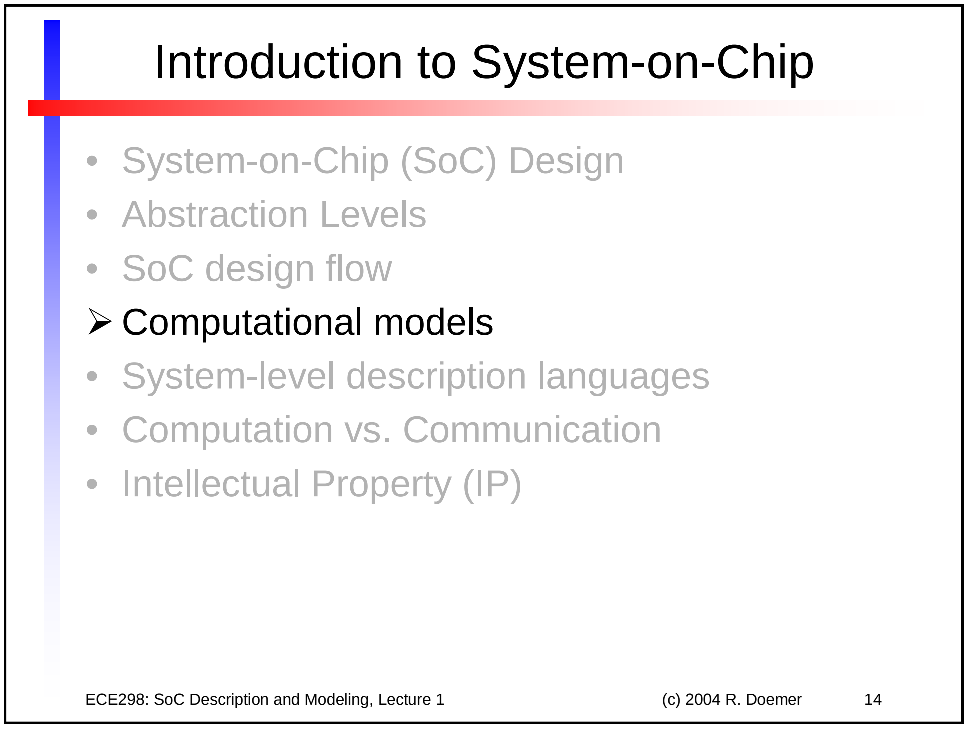
Introduction to System-on-Chip

- System-on-Chip (SoC) Design
- Abstraction Levels
- ➢ SoC design flow
- Computational models
- System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

SoC Design Flow



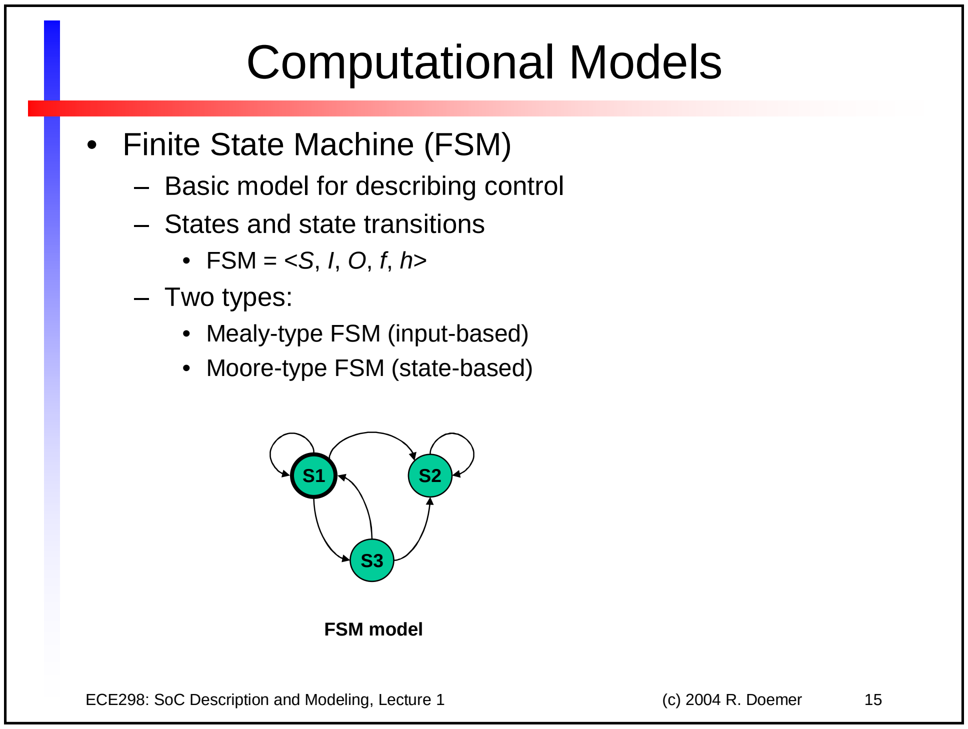
ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 13



Introduction to System-on-Chip

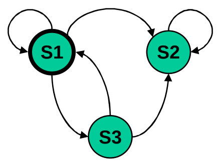
- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- ➢ Computational models
- System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 14



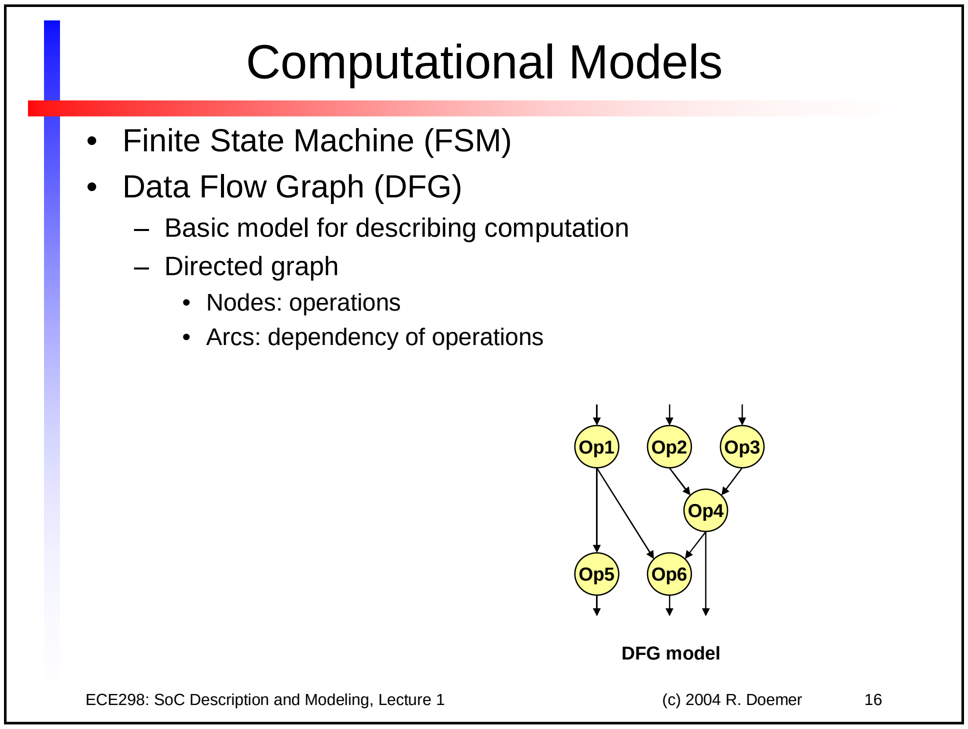
Computational Models

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



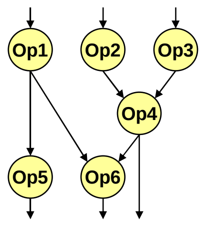
FSM model

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 15



Computational Models

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

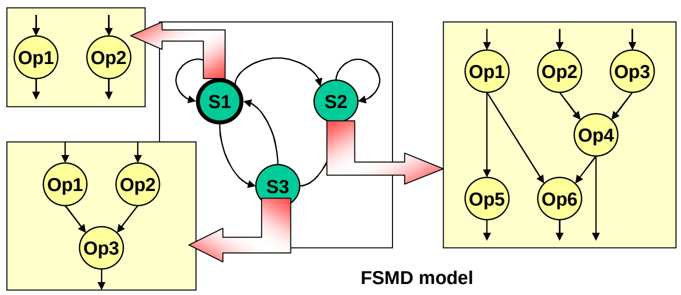


DFG model

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 16

Computational Models

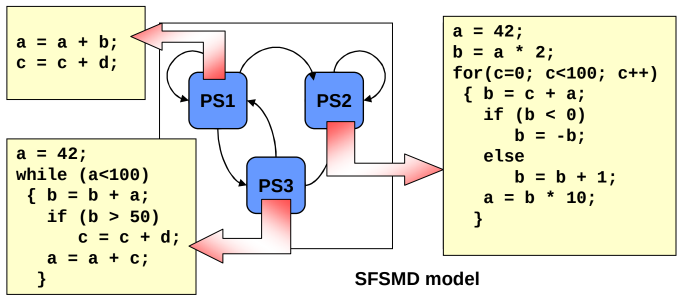
- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
  - Combined model for control and computation
    - FSMD = FSM + DFG
  - Implementation: controller plus datapath



FSMD model

Computational Models

- 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;
 }
```

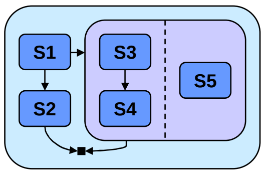
```
a = 42;
b = a * 2;
for(c=0; c<100; c++)
 { b = c + a;
   if (b < 0)
      b = -b;
   else
      b = b + 1;
   a = b * 10;
 }
```

SFSMD model

Computational Models

- 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

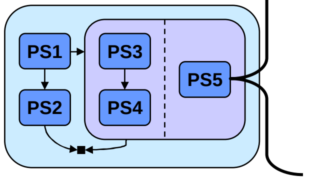
HCFSM model



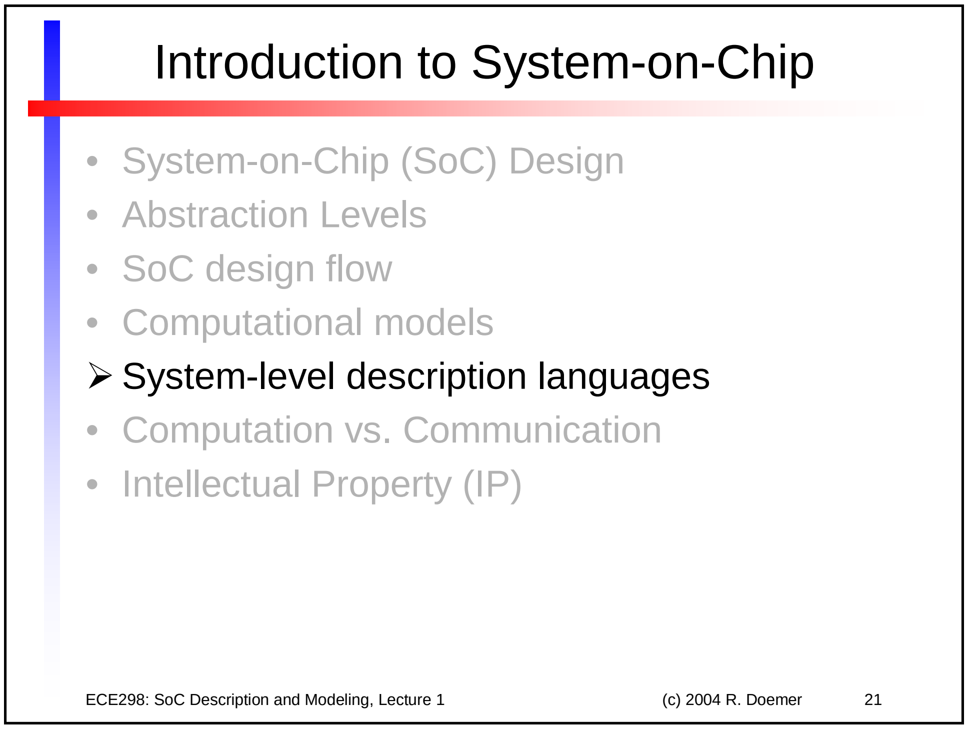
Computational Models

- 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!

PSM model



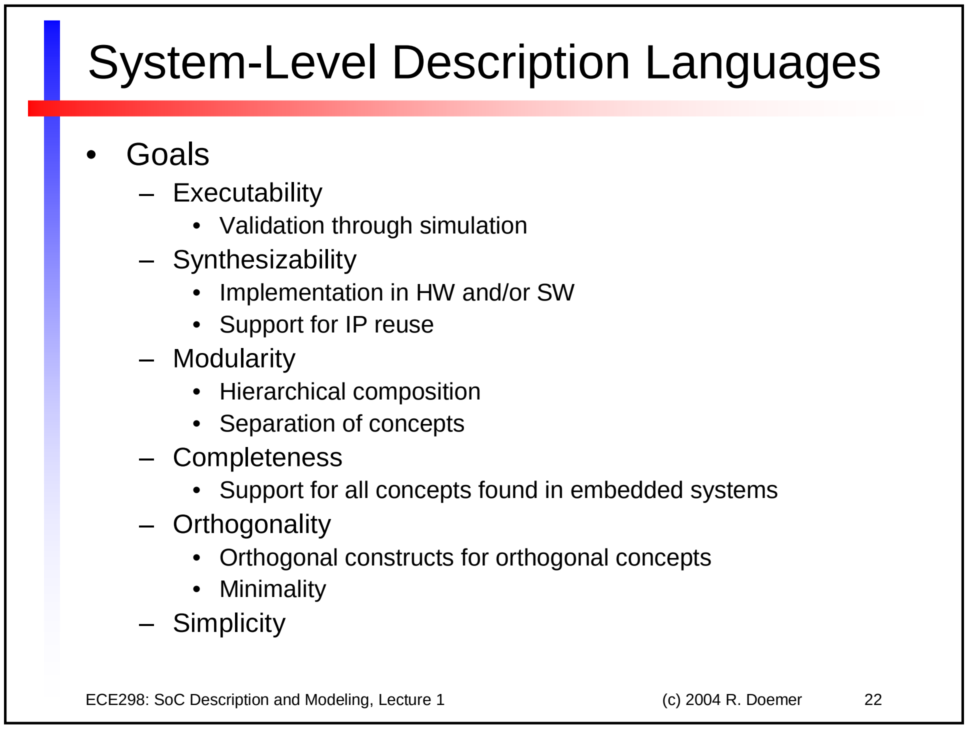
```
...
a = 42;
while (a<100)
 { b = b + a;
   if (b > 50)
      c = c + d;
   else
      c = c + e;
   a = c;
  }
...
```



Introduction to System-on-Chip

- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- Computational models
- ➢ System-level description languages
- Computation vs. Communication
- Intellectual Property (IP)

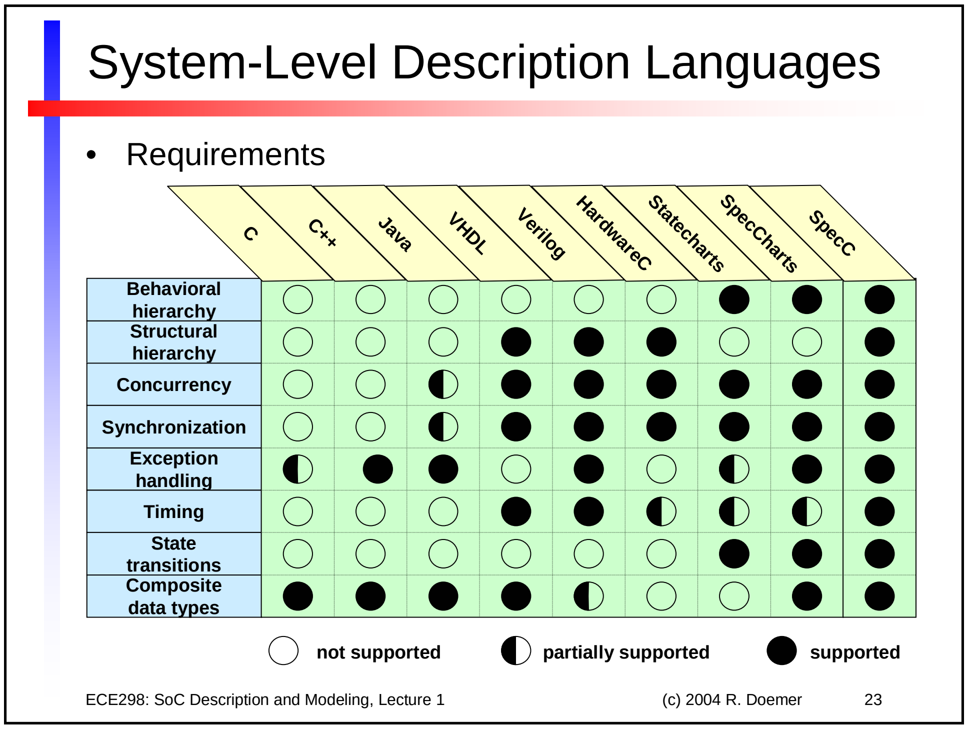
ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 21



System-Level Description Languages

- Goals
  - 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
    - Minimality
  - Simplicity

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 22



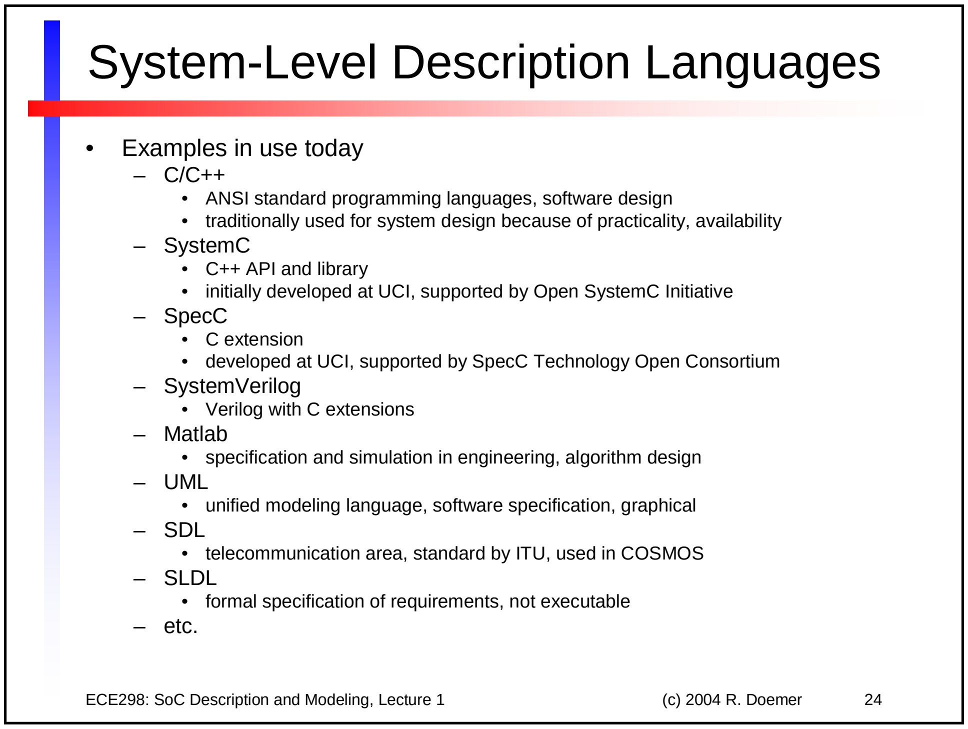
System-Level Description Languages

- Requirements

	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**

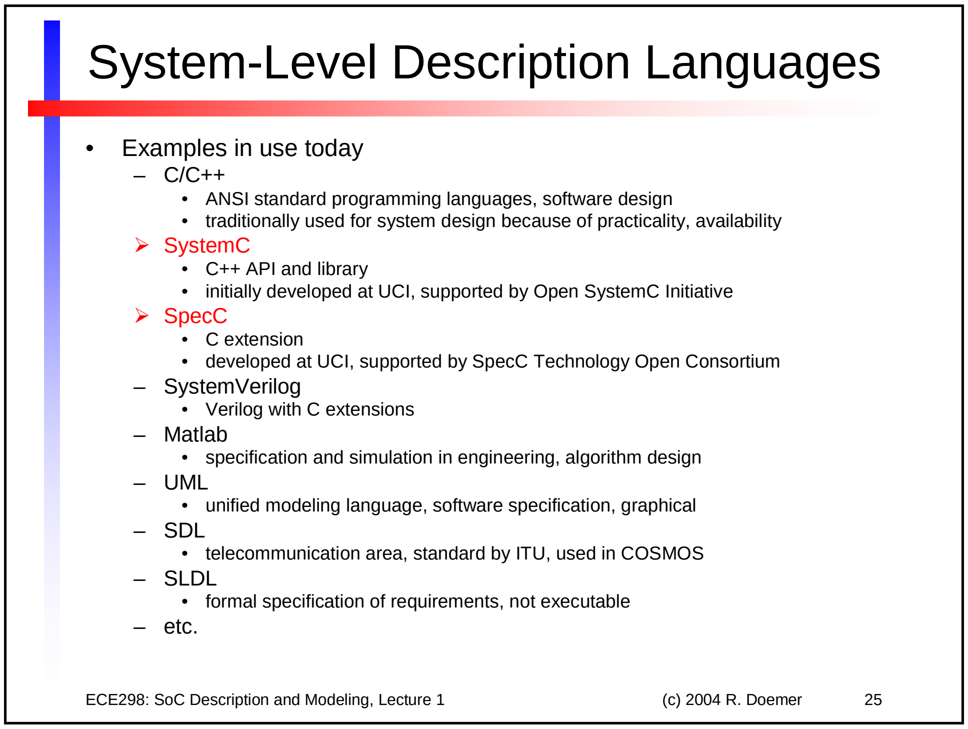
ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 23



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.

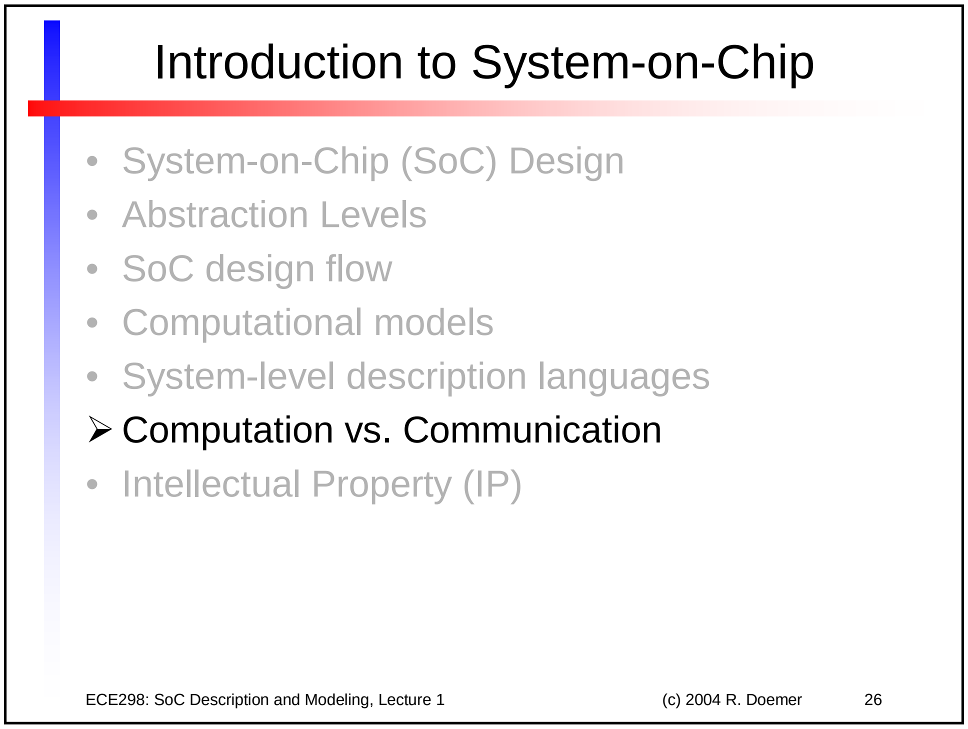
ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 24



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.

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 25



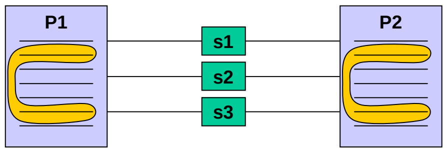
Introduction to System-on-Chip

- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- Computational models
- System-level description languages
- ➢ Computation vs. Communication
- Intellectual Property (IP)

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 26

Computation vs. Communication

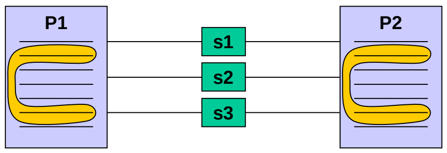
- Traditional model



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

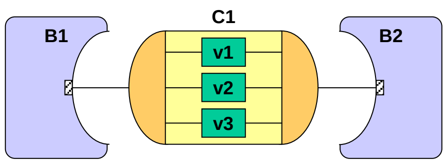
Computation vs. Communication

- Traditional model

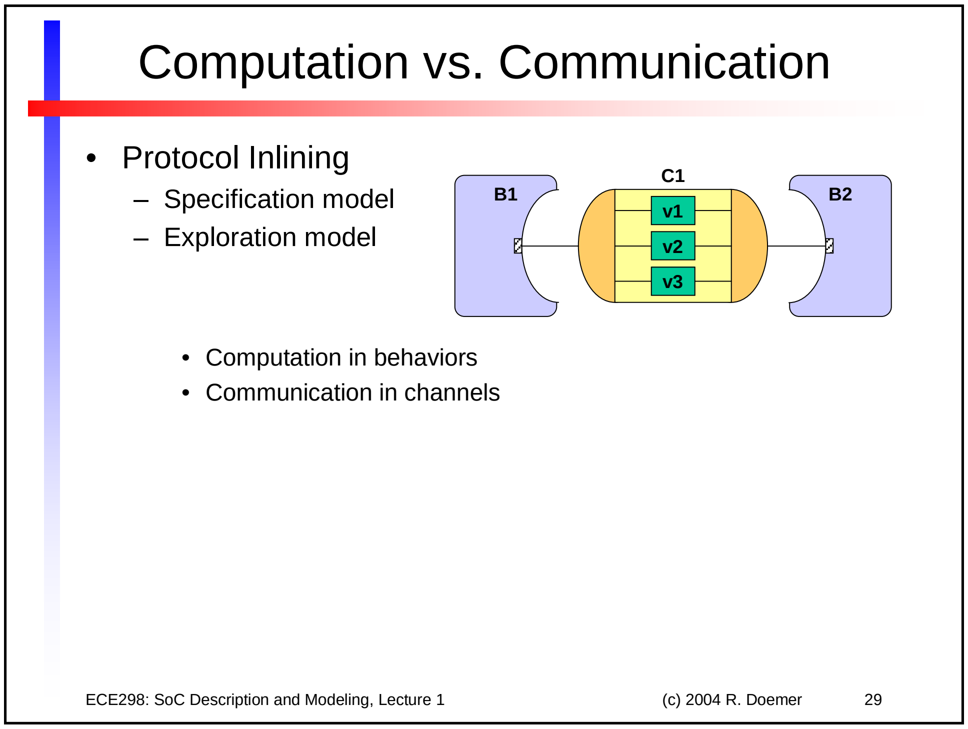


  - Processes and signals
  - Mixture of computation and communication
  - Automatic replacement impossible

- SpecC model

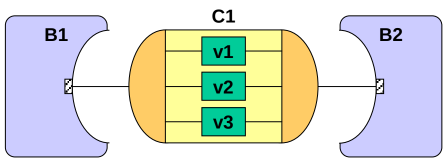


  - Behaviors and channels
  - Separation of computation and communication
  - Plug-and-play

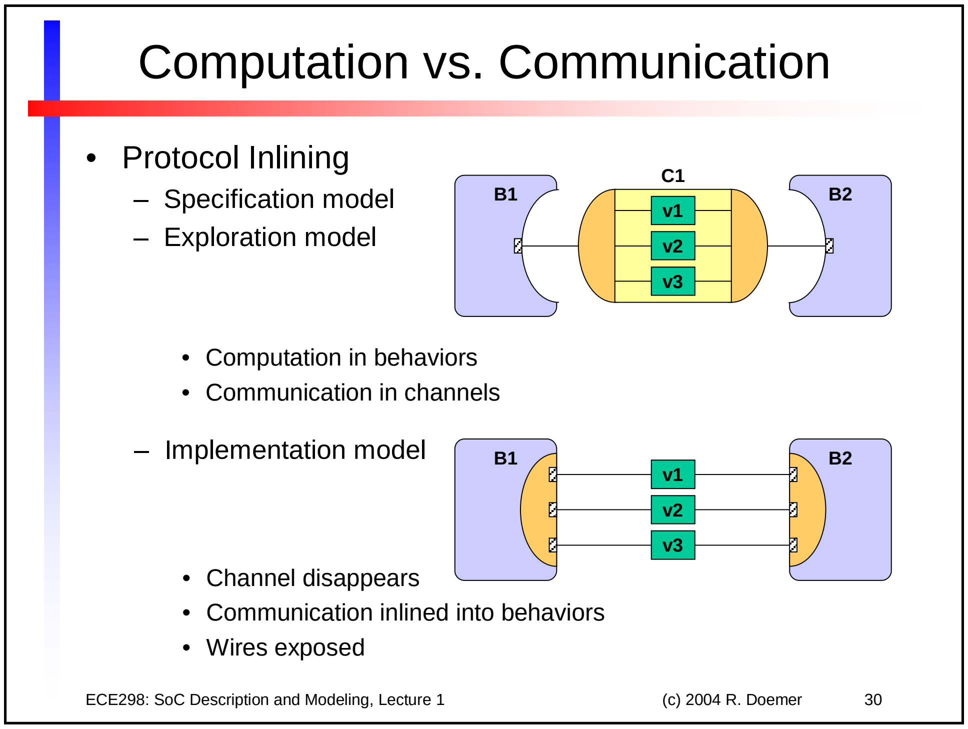


Computation vs. Communication

- Protocol Inlining
  - Specification model
  - Exploration model

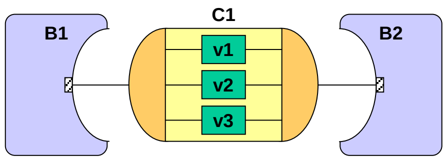


- Computation in behaviors
- Communication in channels



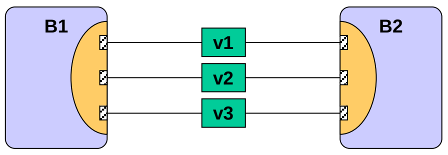
Computation vs. Communication

- Protocol Inlining
  - Specification model
  - Exploration model

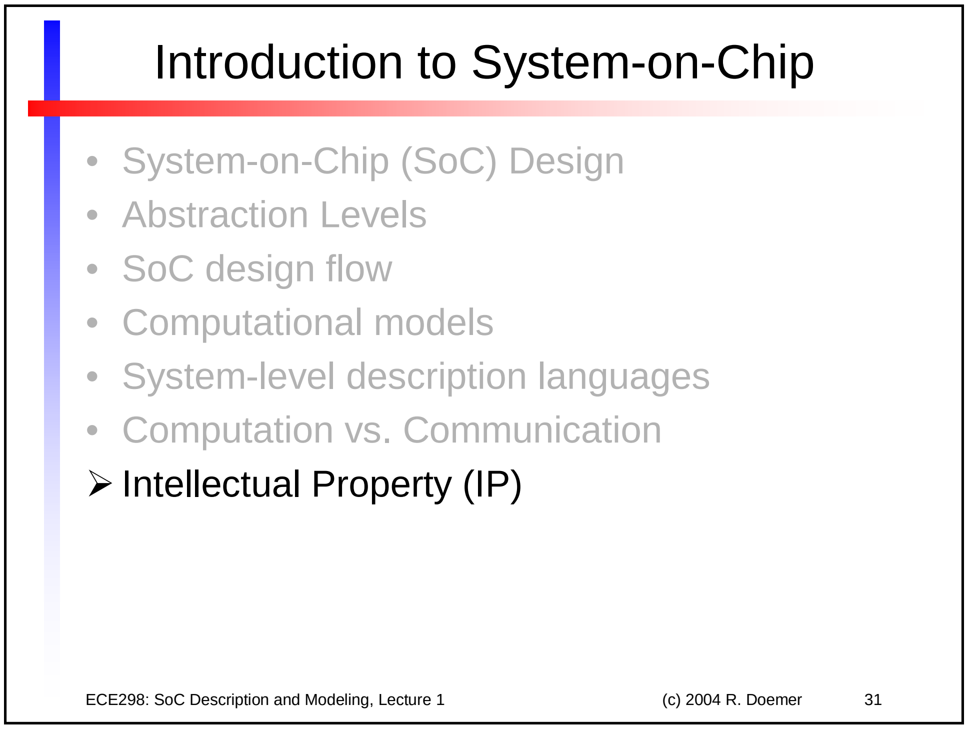


- Computation in behaviors
- Communication in channels

  - Implementation model



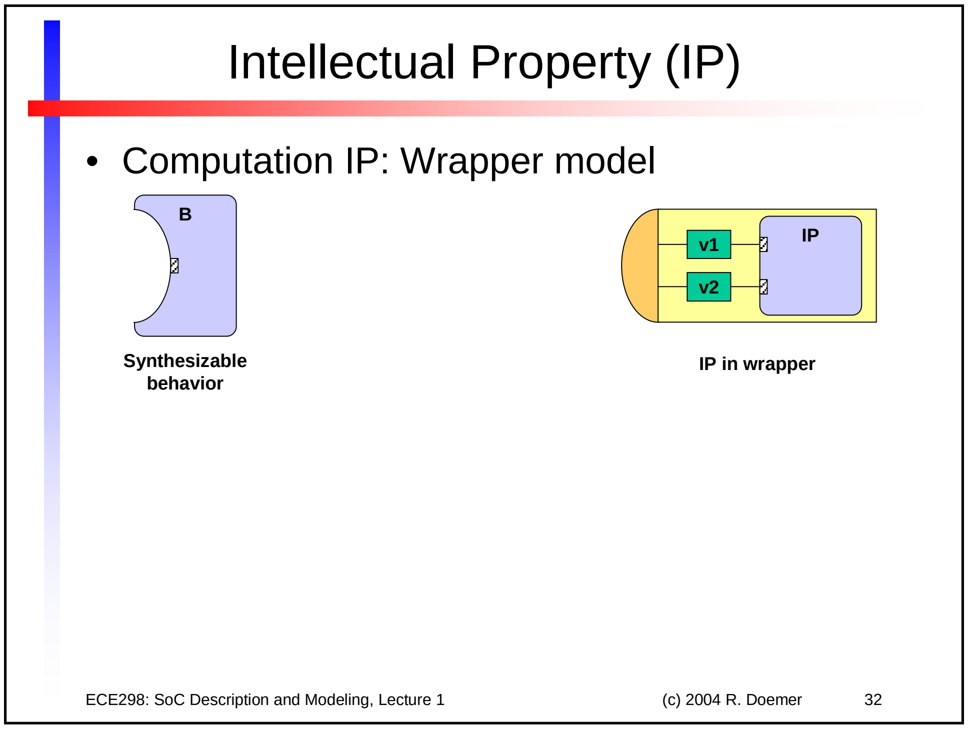
- Channel disappears
- Communication inlined into behaviors
- Wires exposed



Introduction to System-on-Chip

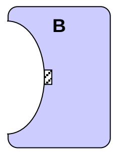
- System-on-Chip (SoC) Design
- Abstraction Levels
- SoC design flow
- Computational models
- System-level description languages
- Computation vs. Communication
- ➢ Intellectual Property (IP)

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 31

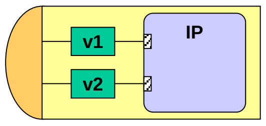


Intellectual Property (IP)

- Computation IP: Wrapper model



Synthesizable behavior

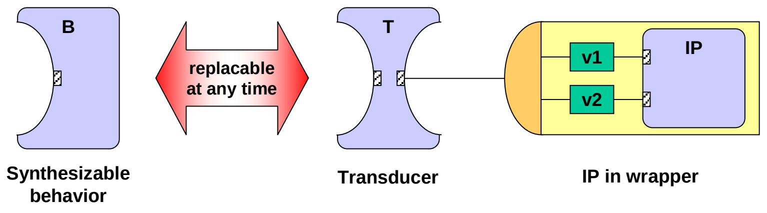


IP in wrapper

ECE298: SoC Description and Modeling, Lecture 1 (c) 2004 R. Doemer 32

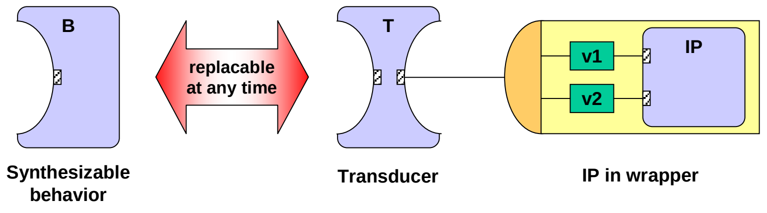
Intellectual Property (IP)

- Computation IP: Wrapper model

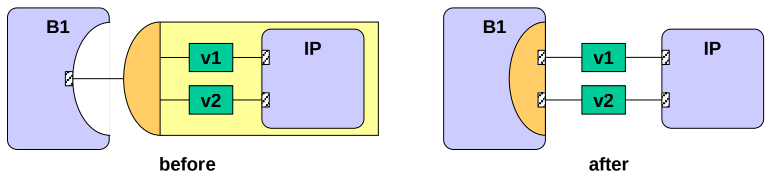


Intellectual Property (IP)

- Computation IP: Wrapper model

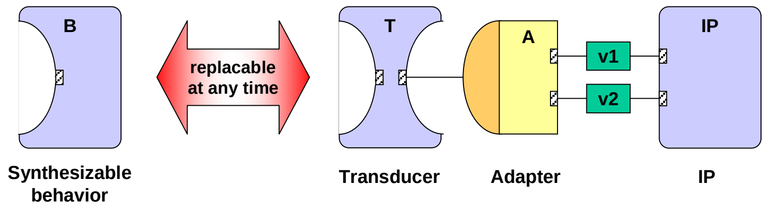


- Protocol inlining with wrapper



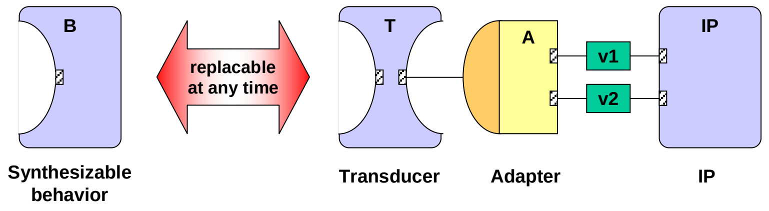
Intellectual Property (IP)

- Computation IP: Adapter model

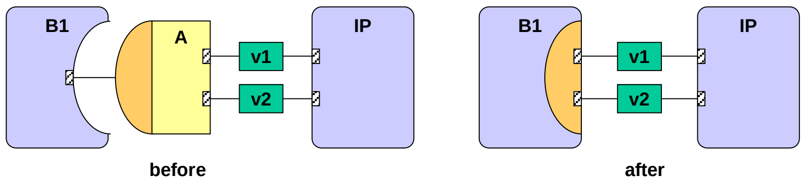


Intellectual Property (IP)

- Computation IP: Adapter model

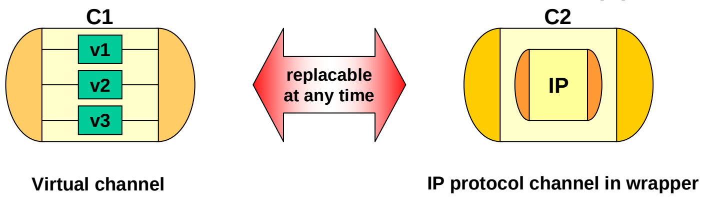


- Protocol inlining with adapter



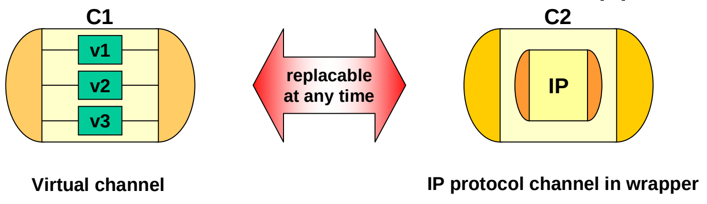
Intellectual Property (IP)

- Communication IP: Channel with wrapper

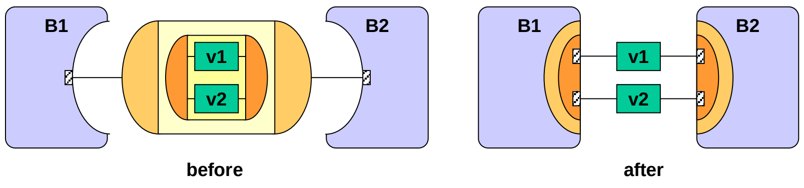


Intellectual Property (IP)

- Communication IP: Channel with wrapper

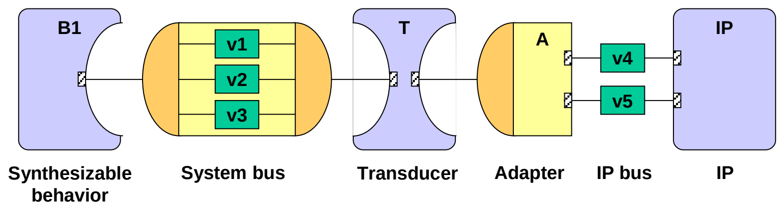


- Protocol inlining with hierarchical channel



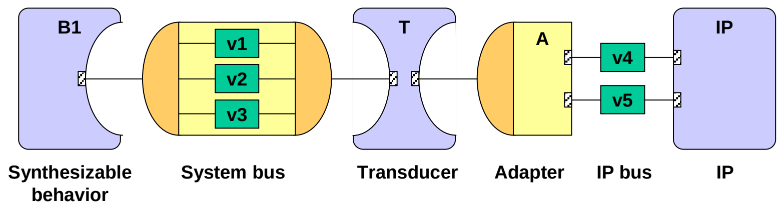
Intellectual Property (IP)

- Incompatible busses: Transducer insertion

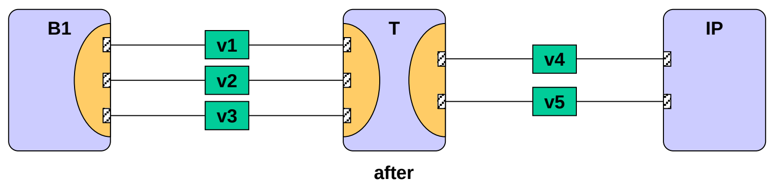


Intellectual Property (IP)

- Incompatible busses: Transducer insertion



- Protocol inlining with transducer


B1
v1
v2
v3
T
v4
v5
IP
after