# 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
  - Computational models
  - System-level description languages
  - Computation, communication, IP

## Course Administration

- Course web pages at **`http://eee.uci.edu/07f/18430/`**
  - Instructor information
  - Course description and policies
  - Objectives and outcomes
  - Contents and schedule
  - Resources and communication
  - Assignments



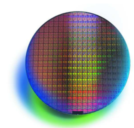
## Introduction to SoC Design

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

## System-on-Chip Design

- Embedded systems are everywhere…

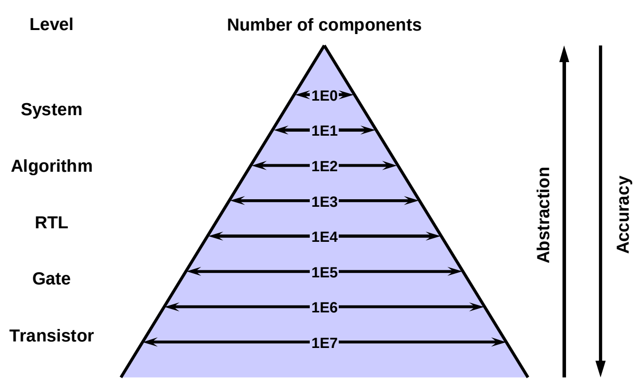
- Deep sub-micron design enables System-on-Chip (SoC)



EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 5

## Abstraction Levels

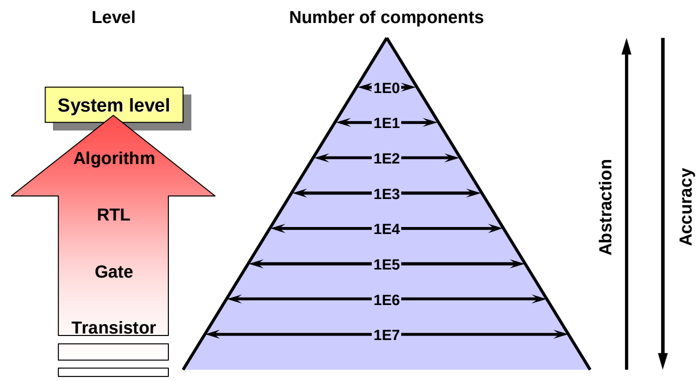
- System-on-Chip (SoC) design faces tremendous increase of design complexity




EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 6

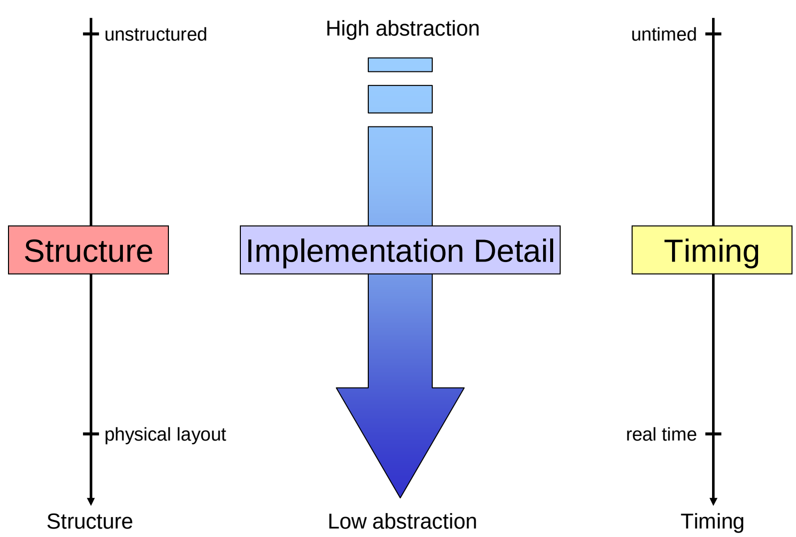
# Abstraction Levels

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

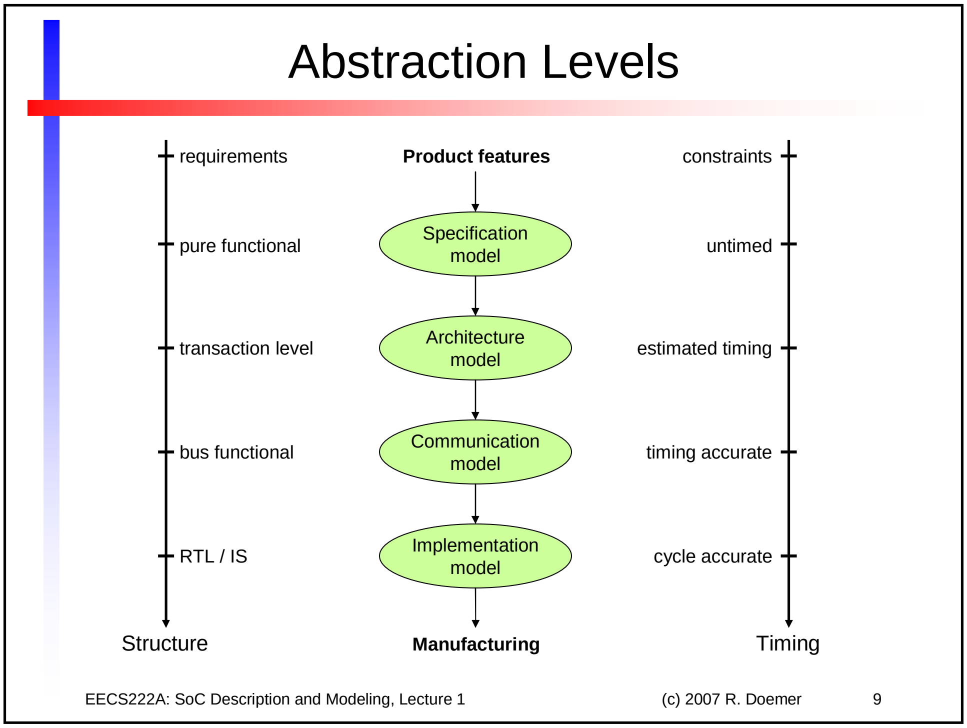


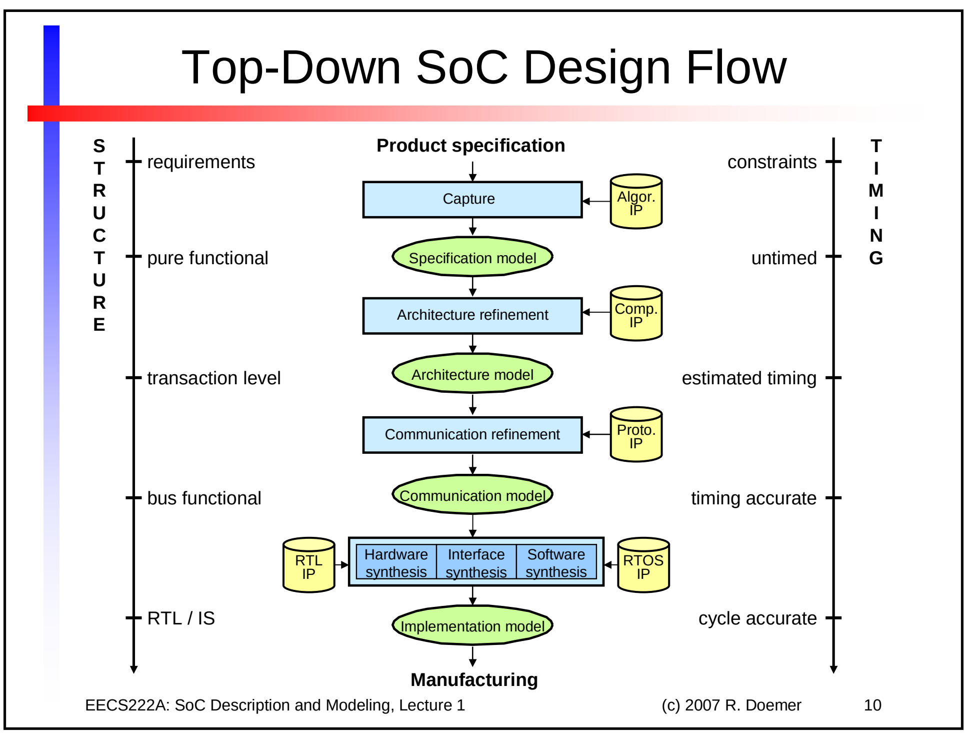

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 7

# Abstraction Levels




EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 8


Abstraction Levels
requirements
Product features
constraints
pure functional
Specification model
untimed
transaction level
Architecture model
estimated timing
bus functional
Communication model
timing accurate
RTL / IS
Implementation model
cycle accurate
Structure
Manufacturing
Timing
EECS222A: SoC Description and Modeling, Lecture 1
(c) 2007 R. Doemer
9


Top-Down SoC Design Flow
STRUCTURE
requirements
Product specification
constraints
TIMING
Capture
Algor. IP
pure functional
Specification model
untimed
Architecture refinement
Comp. IP
transaction level
Architecture model
estimated timing
Communication refinement
Proto. IP
bus functional
Communication model
timing accurate
RTL IP
Hardware synthesis
Interface synthesis
Software synthesis
RTOS IP
RTL / IS
Implementation model
cycle accurate
Manufacturing
EECS222A: SoC Description and Modeling, Lecture 1
(c) 2007 R. Doemer
10

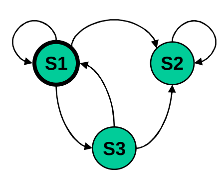
# Computational Models

- Models of Computation
  - 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) 2007 R. Doemer 11

# 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)

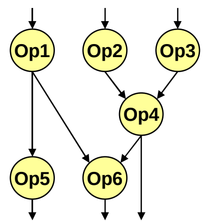



**FSM model**

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 12

# Computational Models

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

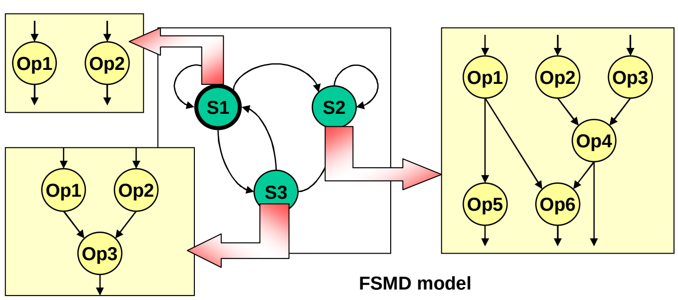



DFG model



# Computational Models

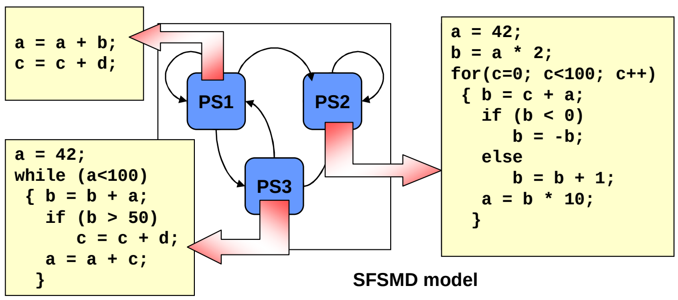
- Finite State Machine (FSM)
- Data Flow Graph (DFG)
- Finite State Machine with Data (FSMD)
  - Combined model for control and computation
    - FSMD = FSM + DFG
  - Implementation: controller plus datapath




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



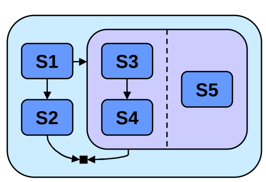

SFSMD model

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 15

# 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

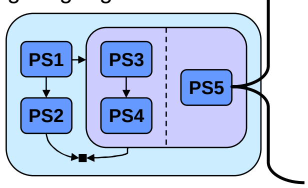



EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 16

# 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;
 }
...
```



# 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

# 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**

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 19

# 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) 2007 R. Doemer 20

## 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) 2007 R. Doemer 21

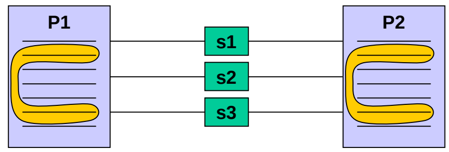
## Separation of Concerns

- 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

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 22

## Computation vs. Communication

- Traditional model

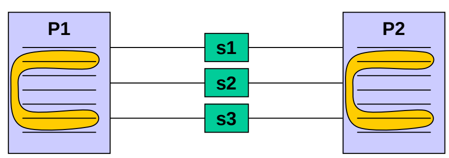



  - Processes and signals
  - Mixture of computation and communication
  - Automatic replacement impossible

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 23

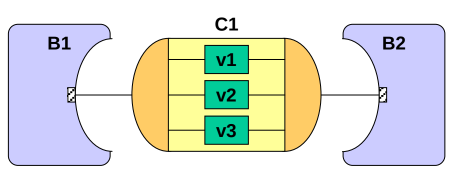
## Computation vs. Communication

- Traditional model




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

- SpecC model

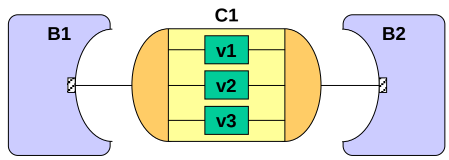



  - Behaviors and channels
  - Separation of computation and communication
  - Plug-and-play

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 24

## Computation vs. Communication

- Protocol Inlining
  - Specification model
  - Exploration model

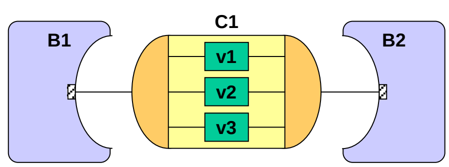



- Computation in behaviors
- Communication in channels

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 25

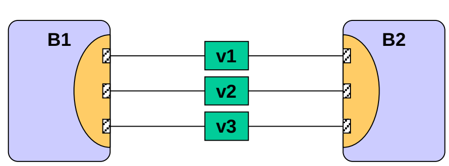
## Computation vs. Communication

- 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) 2007 R. Doemer 26

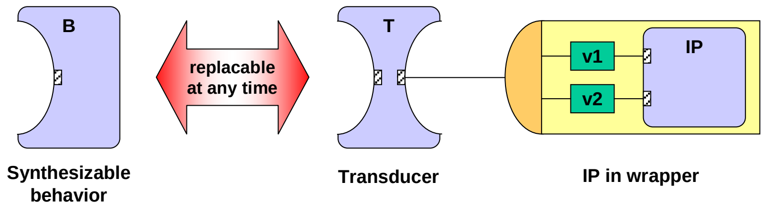
## Intellectual Property (IP)

- Computation IP: Wrapper model


EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 27

## Intellectual Property (IP)

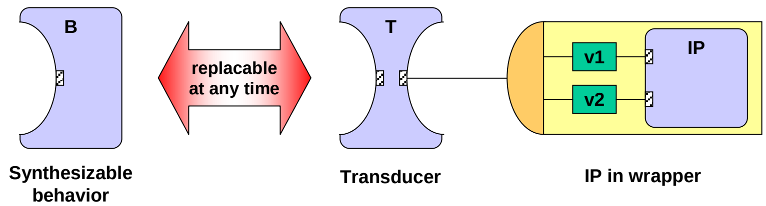
- Computation IP: Wrapper model



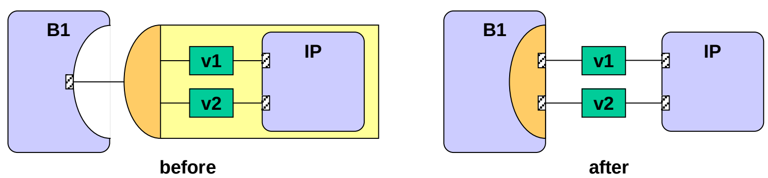

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 28

# Intellectual Property (IP)

- Computation IP: Wrapper model



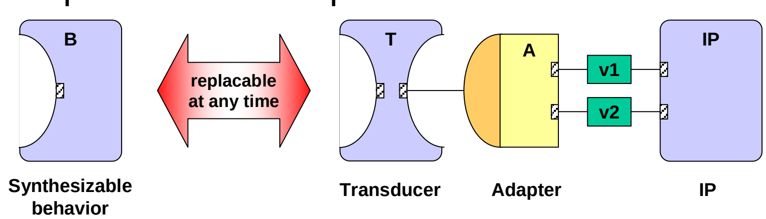

- Protocol inlining with wrapper




EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 29

# Intellectual Property (IP)

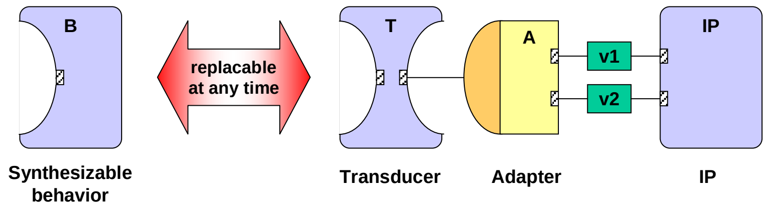
- Computation IP: Adapter model



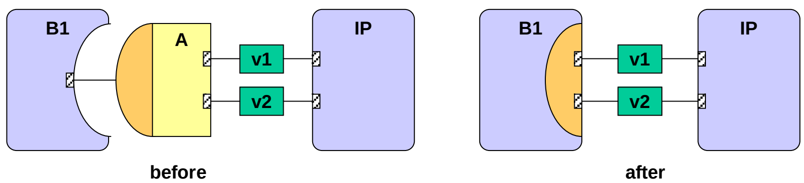

EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 30

# Intellectual Property (IP)

- Computation IP: Adapter model



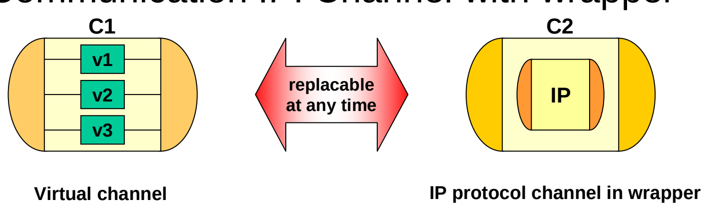

- Protocol inlining with adapter





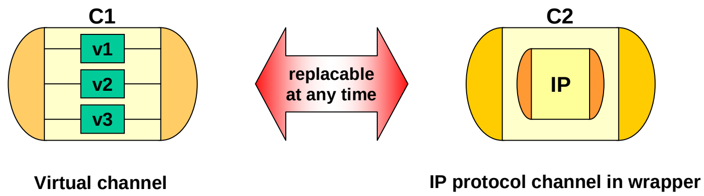

# Intellectual Property (IP)

- Communication IP: Channel with wrapper

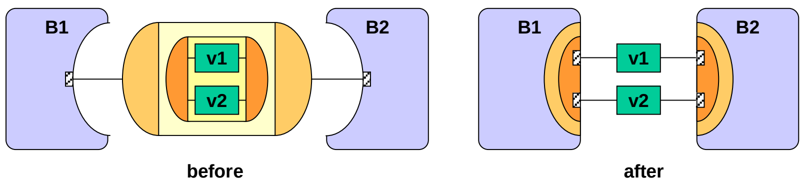


# Intellectual Property (IP)

- Communication IP: Channel with wrapper



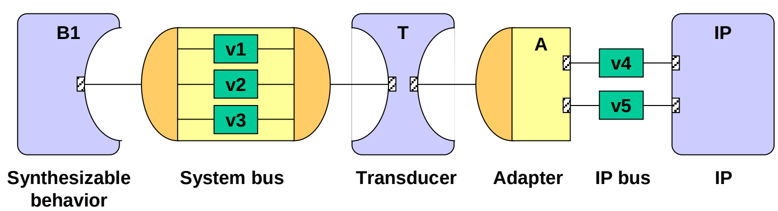

- Protocol inlining with hierarchical channel




EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 33

# Intellectual Property (IP)

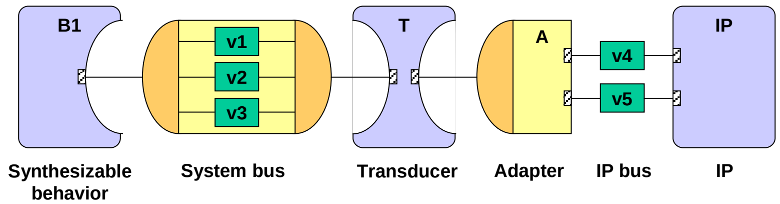
- Incompatible busses: Transducer insertion




EECS222A: SoC Description and Modeling, Lecture 1 (c) 2007 R. Doemer 34

# Intellectual Property (IP)

- Incompatible busses: Transducer insertion




- Protocol inlining with transducer