Communication-Based Design

Motivation

- System-level verification of large component-oriented designs will be very costly.
- We cannot afford to debug interface mismatches between internal components
- . . . especially considering that there will be many, many interfaces between so many components.
- Current situation is unacceptable
  - Interfaces are not specified precisely
  - No clear specification formalism exist.
  - Tools to create, debug, and make maximal use of the specifications don't exist.

Basic Goals

- Identify precisely and formally the concept of *communication, its level of abstraction* and of the corresponding models of computation.
- New theories to combine different models of computation
- Determine a set of properties that characterizes each level of abstraction.
- Provide methods and tools to extract formal properties and specifications from informal ones and existing, ambiguously specified standards.

Basic Goals

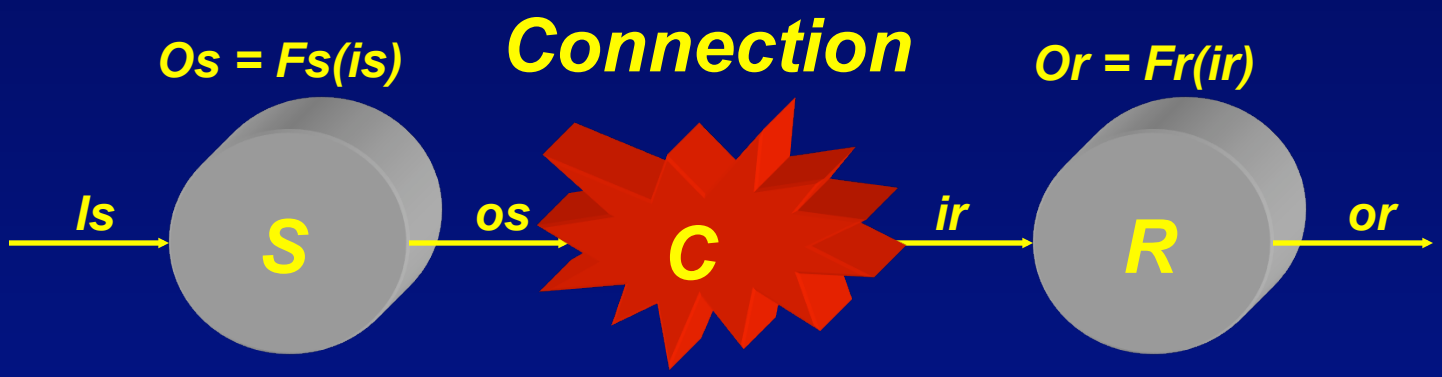
- Formal synthesis of protocols
- Bus architectures analysis and verification
  - formal specs
  - monitors
  - semi-formal analysis
- Abstract the notion of communication architectures and include estimation processes to quickly evaluate different architecture

Why separating computation and communication?

- **Verification (debugging). If not:**
  - Communication hard-wired with computation
  - Often hard to tell who is at fault
  - Bugs may be distributed, difficult to track down
  - Changes in the system may require rewriting of entire blocks, often leading to new bugs
- **Reuse**
  - Component may be plugged in different environments
  - Functions and interface behavior are difficult to separate
- **Architecture exploration**
  - Design components with abstract communication primitives
  - Explore different implementations without touching the component

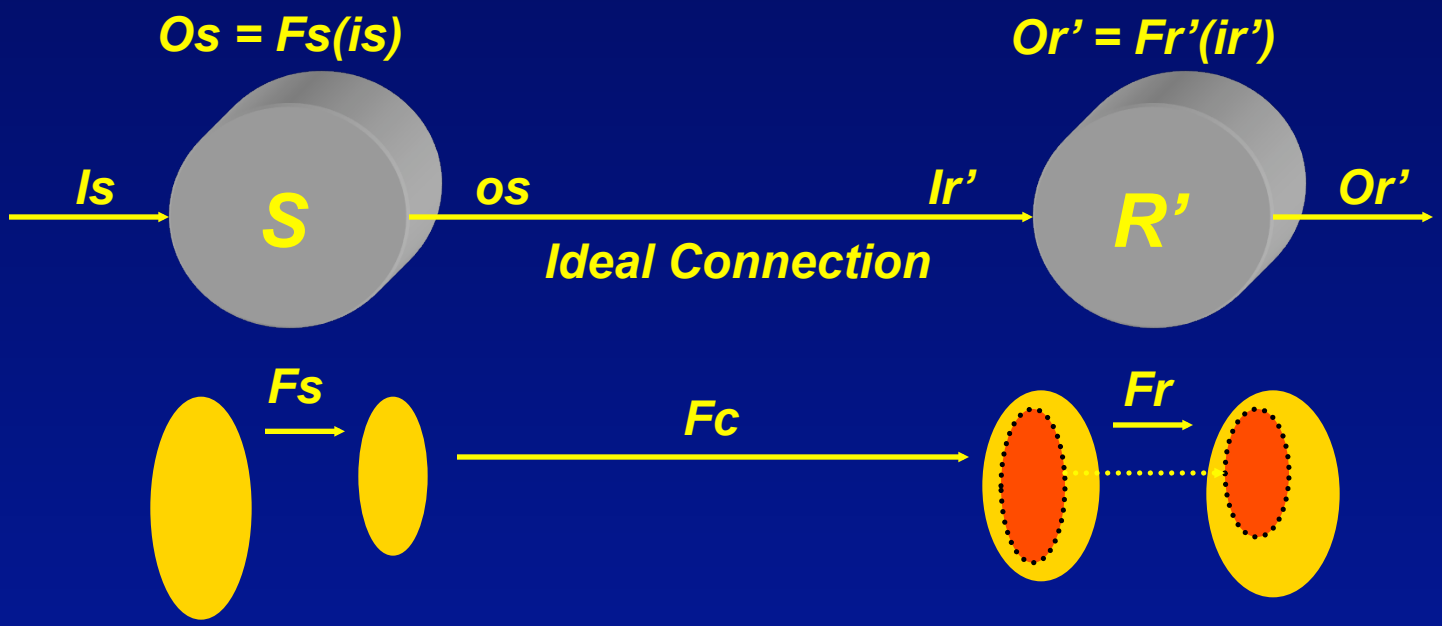
What is Communication?

What is Communication?

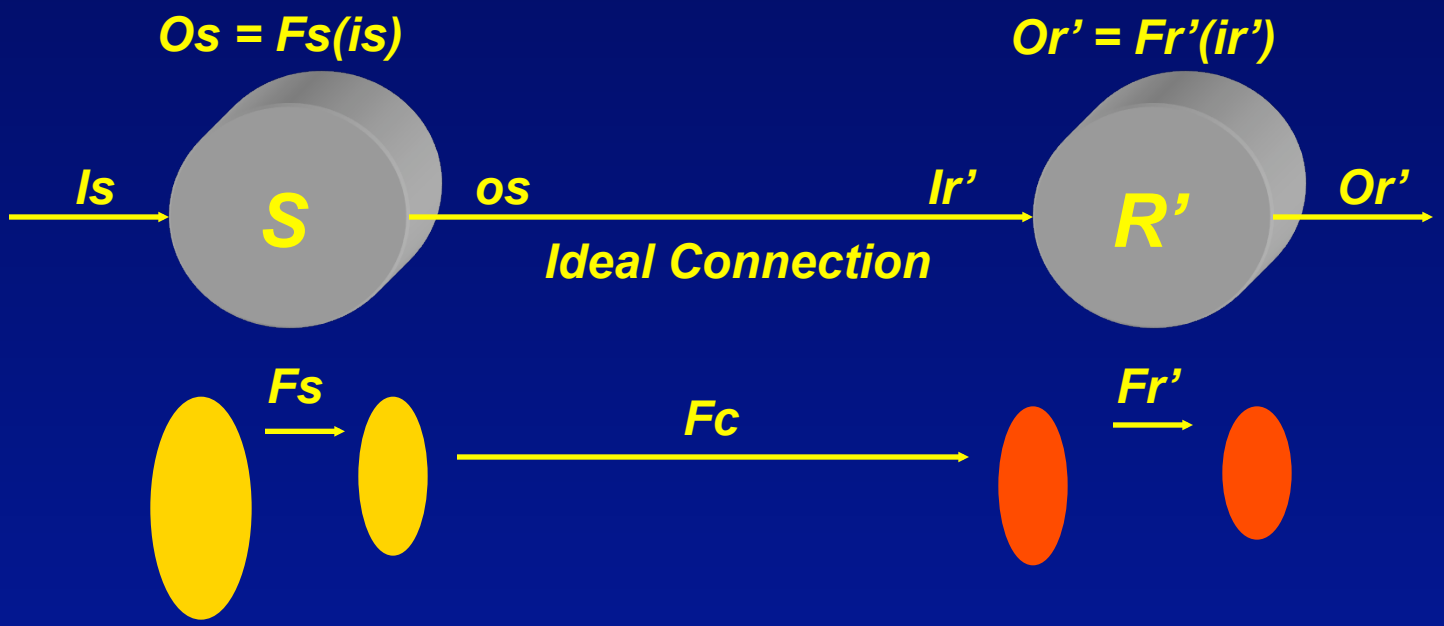


- Connection C enables the interaction between the behaviors S and R

What is Communication?

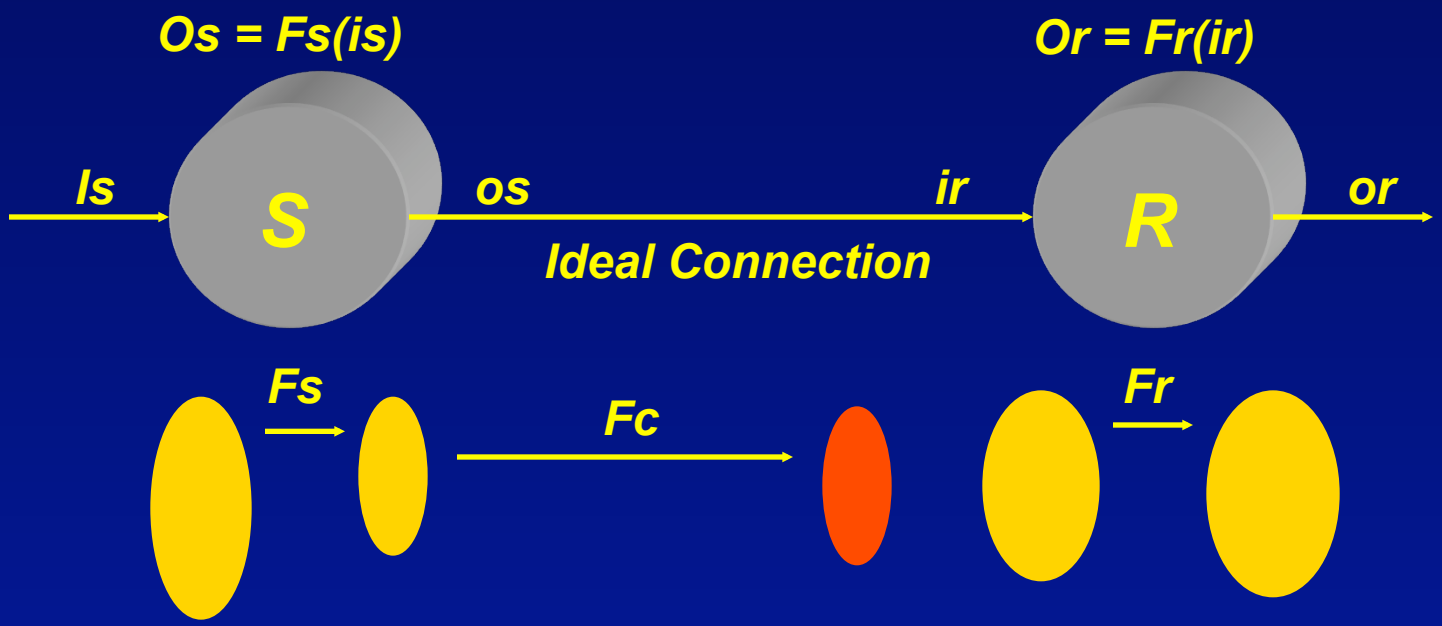


What is Communication?



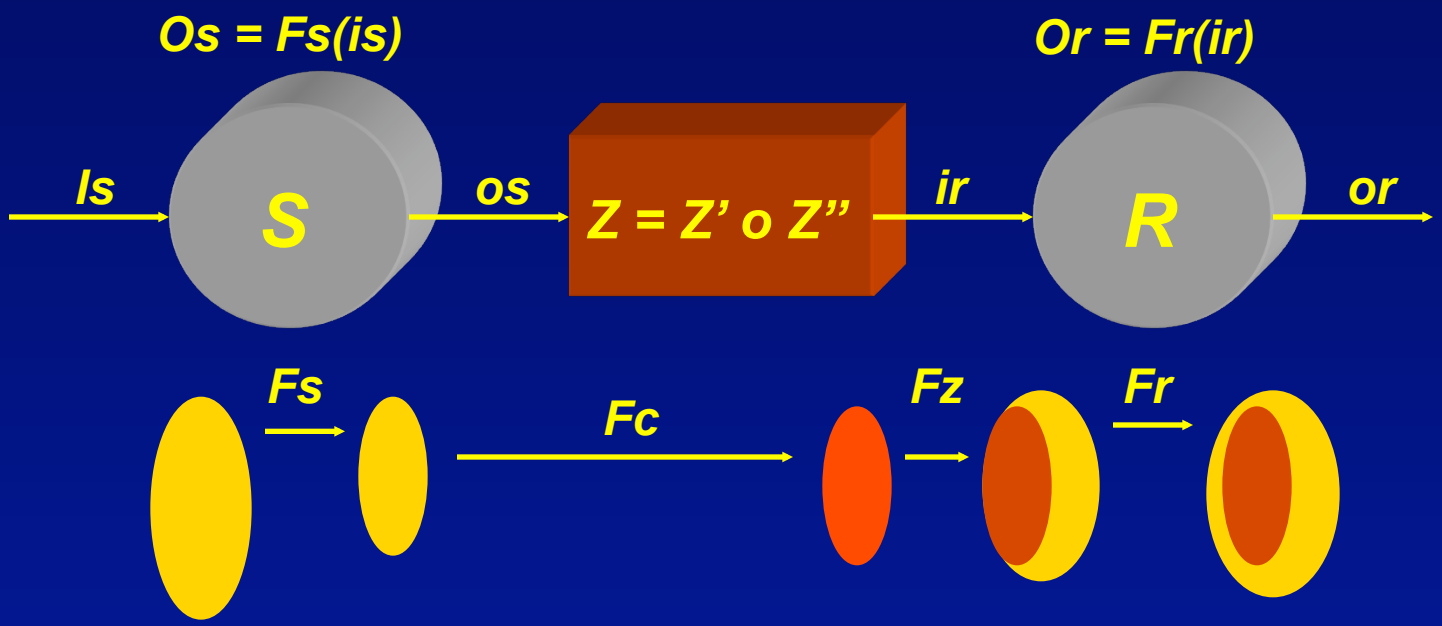
- S restricts the behavior of R to R'

Behavior Adaptation



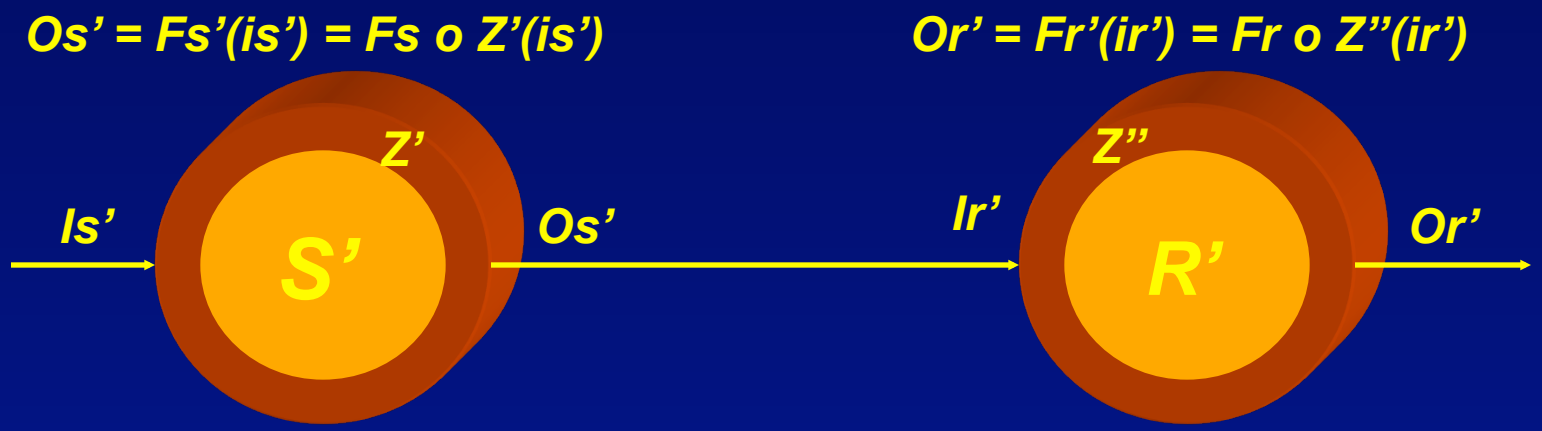
- R not defined for some output of S: behavior mismatch

Behavior Adaptation



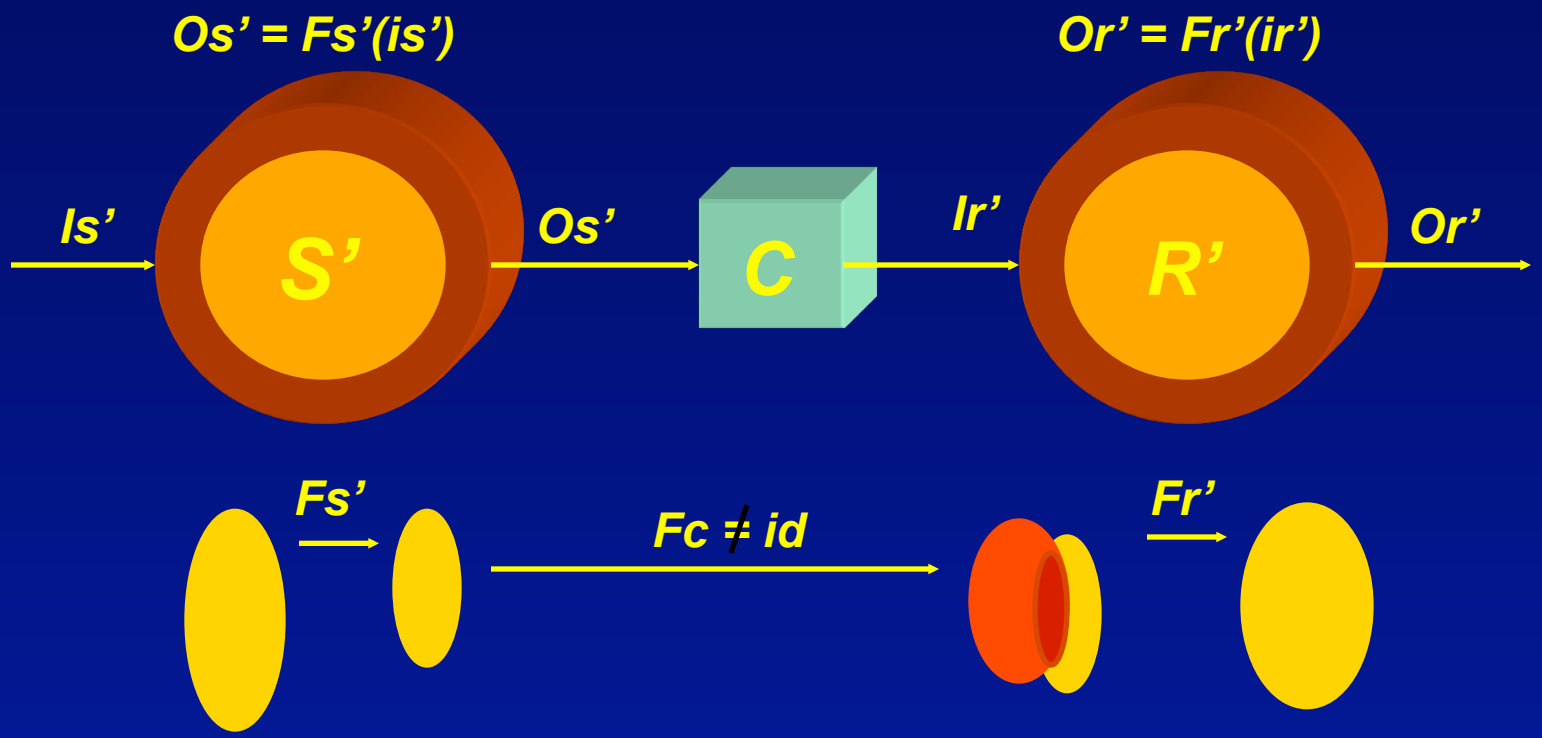
- Behavior Adapter Z maps outputs of S into the domain of R

Behavior Adaptation



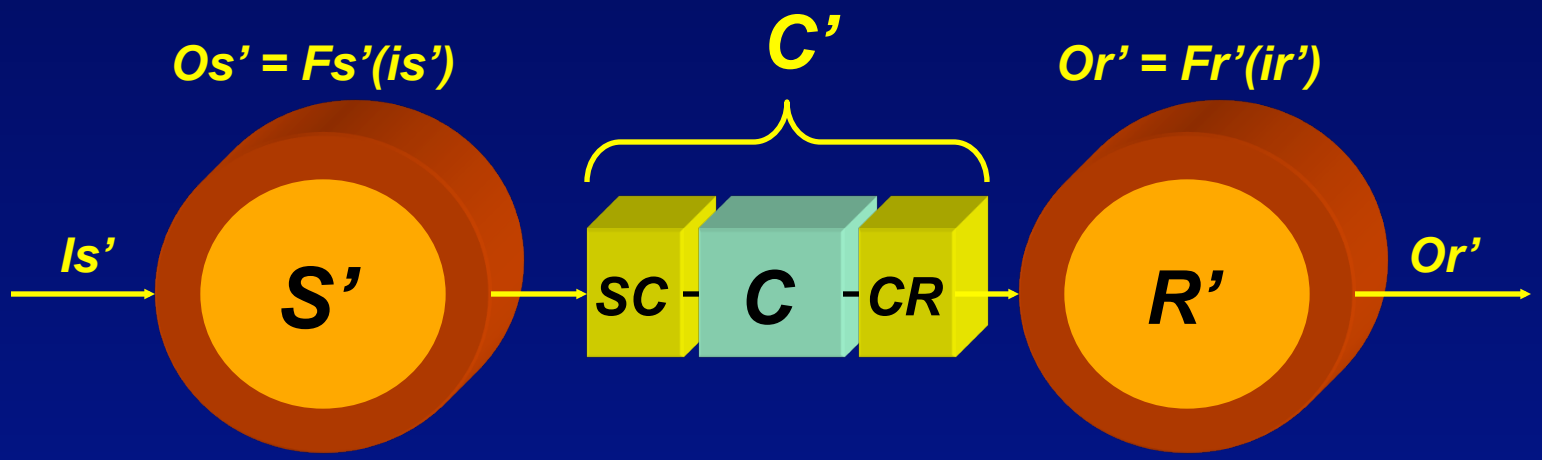
- Behavior Adapter encapsulates S and R
- S' and R' communicate successfully over an ideal connection

Physical Channels



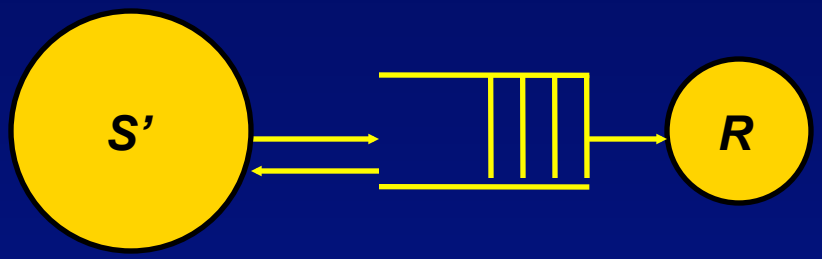
- Invalid Channels may introduce mismatch due to their physical properties (noise, interference…)
- Valid Channels satisfy QoS requirements
  - QoS-equivalent to the ideal connection (Fs' o Fc o Fr' ~ Fs' o Fr')

Channel Adapter



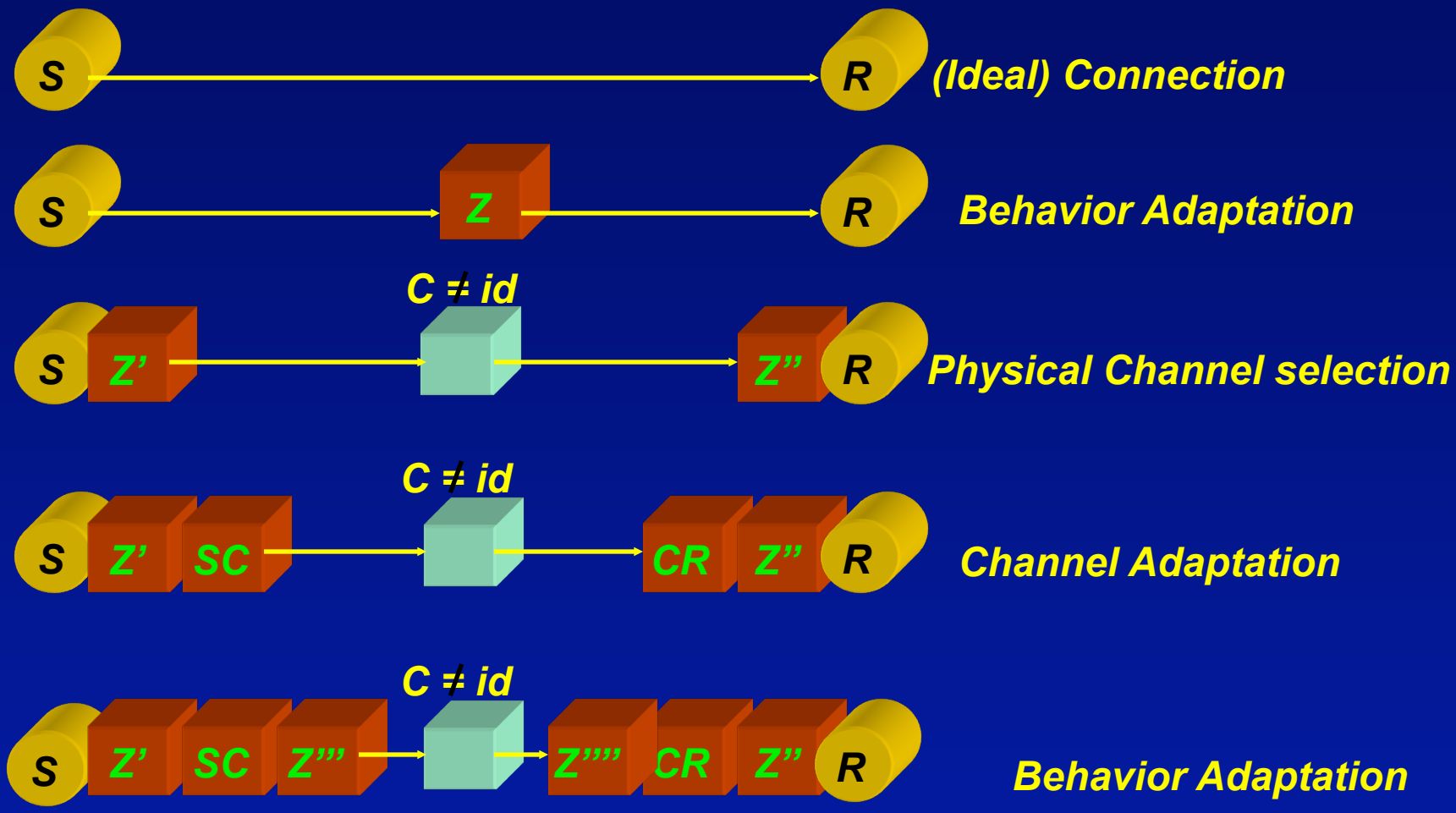
- Choose SC and CR such that C' is valid
  - *Fs' o Fsc o Fc o Fcr o Fr' ~ Fs' o Fr'*
- Channel Adapter may introduce behavior mismatch
  - need a Behavior Adapter

FIFOs as Behavior Adapters

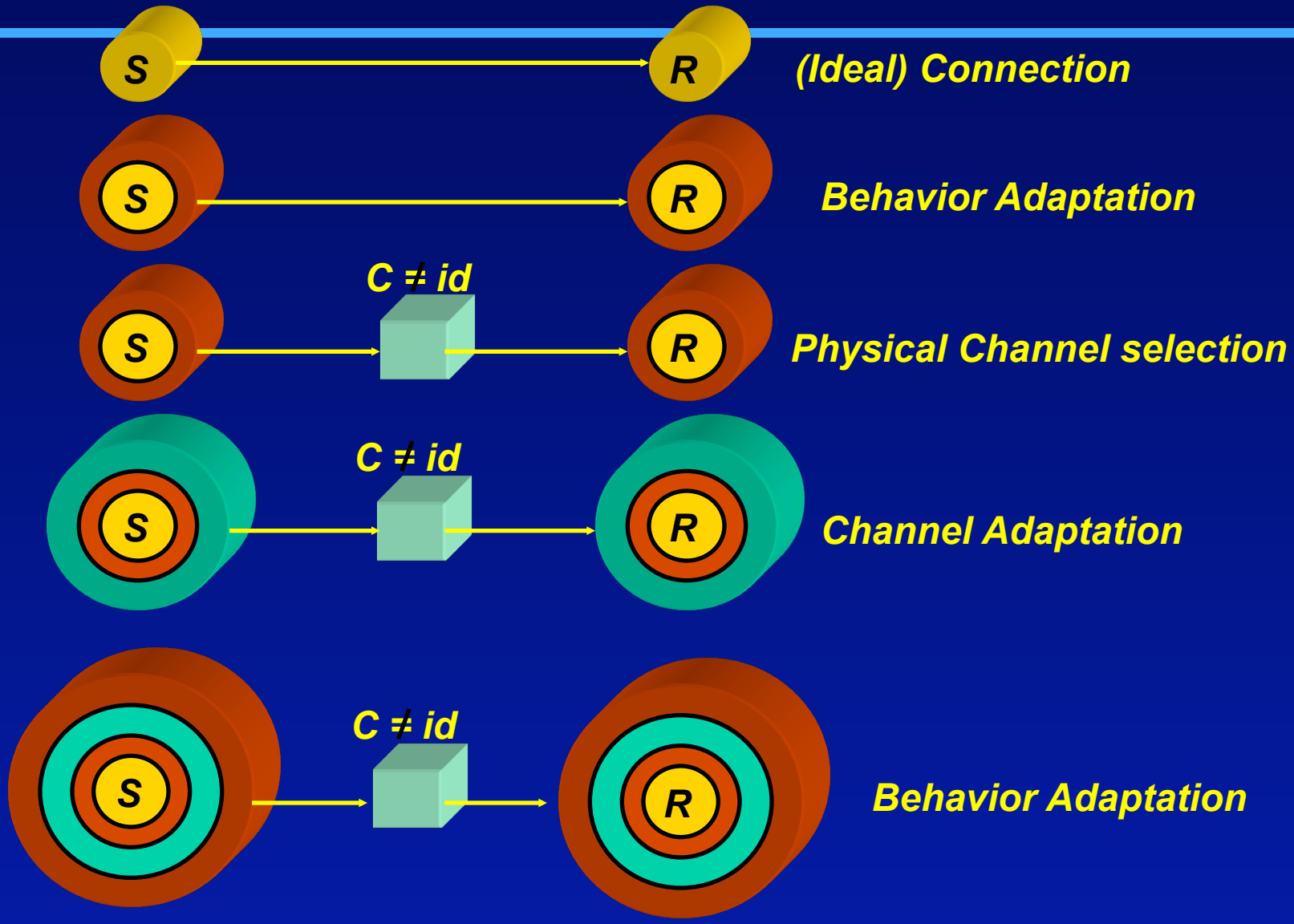


- FIFOs adapt the rates of S and R
- Unbounded FIFOs
  - ideal adapter
- Bounded FIFOs
  - to prevent overflow, restrict S using blocking write (Req/Ack)

Protocol Design



Protocol Design



Design Methodology

Functional Decomposition

Behavior Adaptation

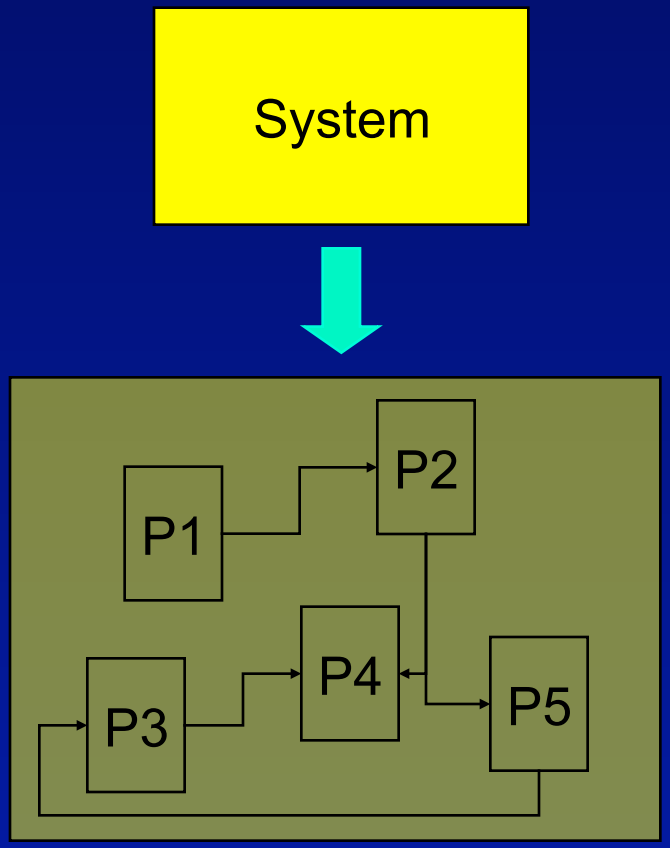
Communication Media Insertion
MoC Wrapping

Communication Refinement
Channel Adaptation

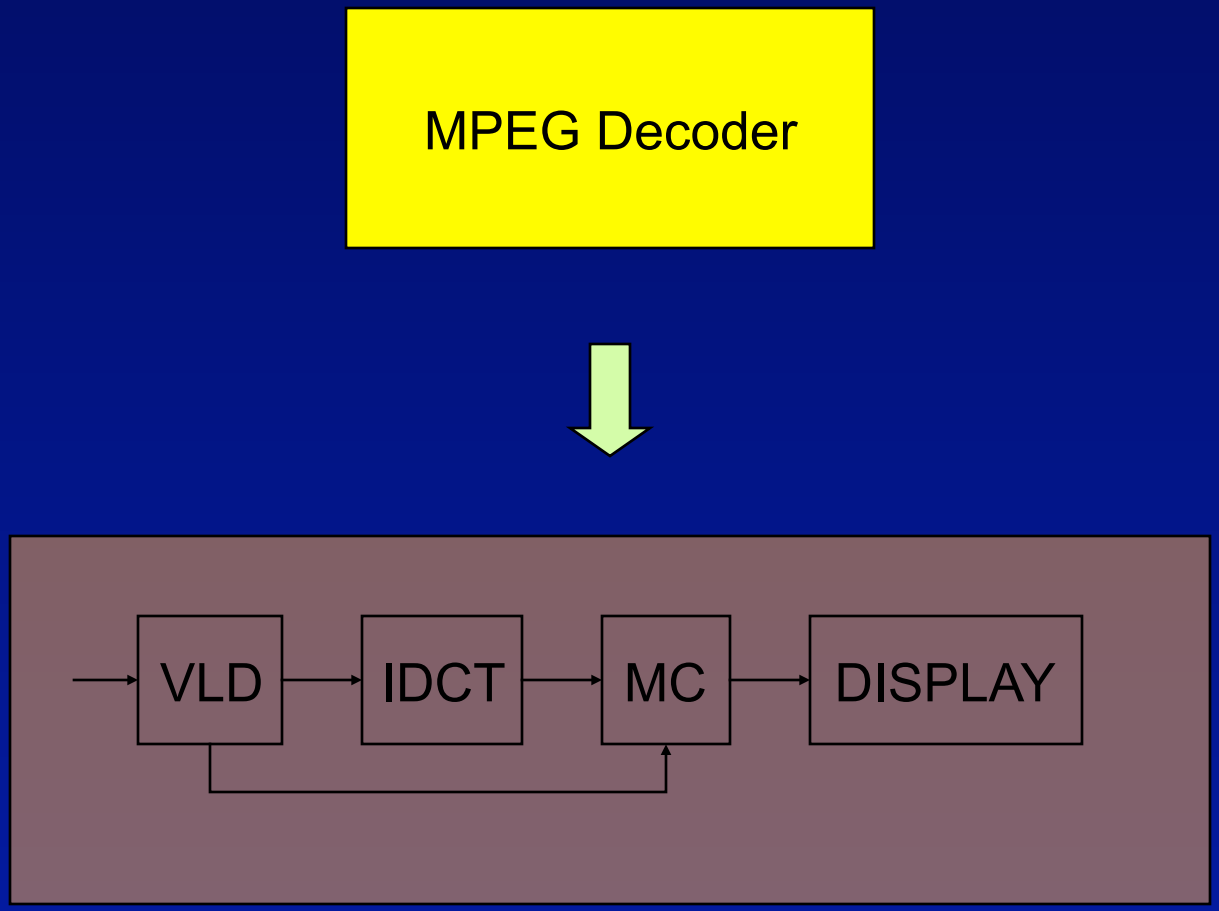
Mapping and Optimizations

Functional Decomposition

- **Functional Decomposition**
  - at the highest abstraction level, a system is a single process
  - it is refined into a set of concurrent processes
- **Process:**
  - relation between an input domain and an output co-domain
  - only behavior, no communication
  - denotational specification

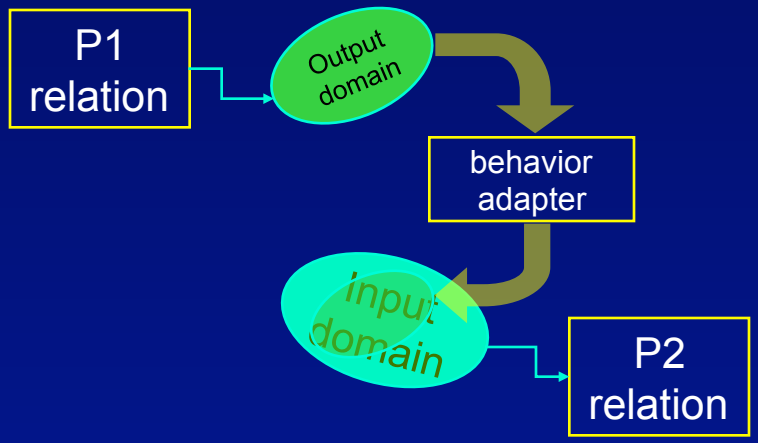


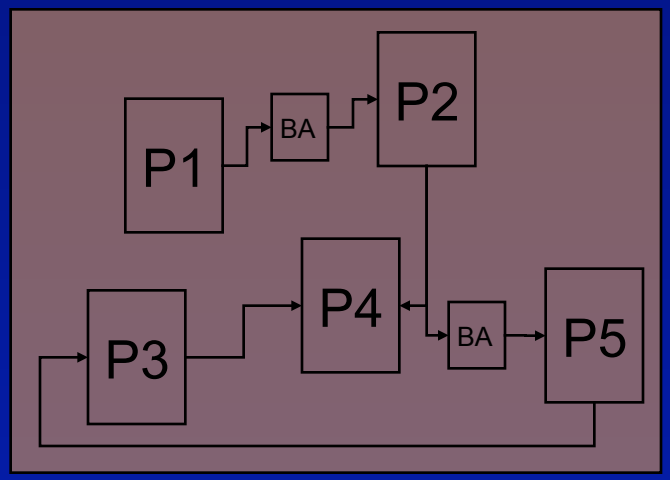
Functional Decomposition (ex.)



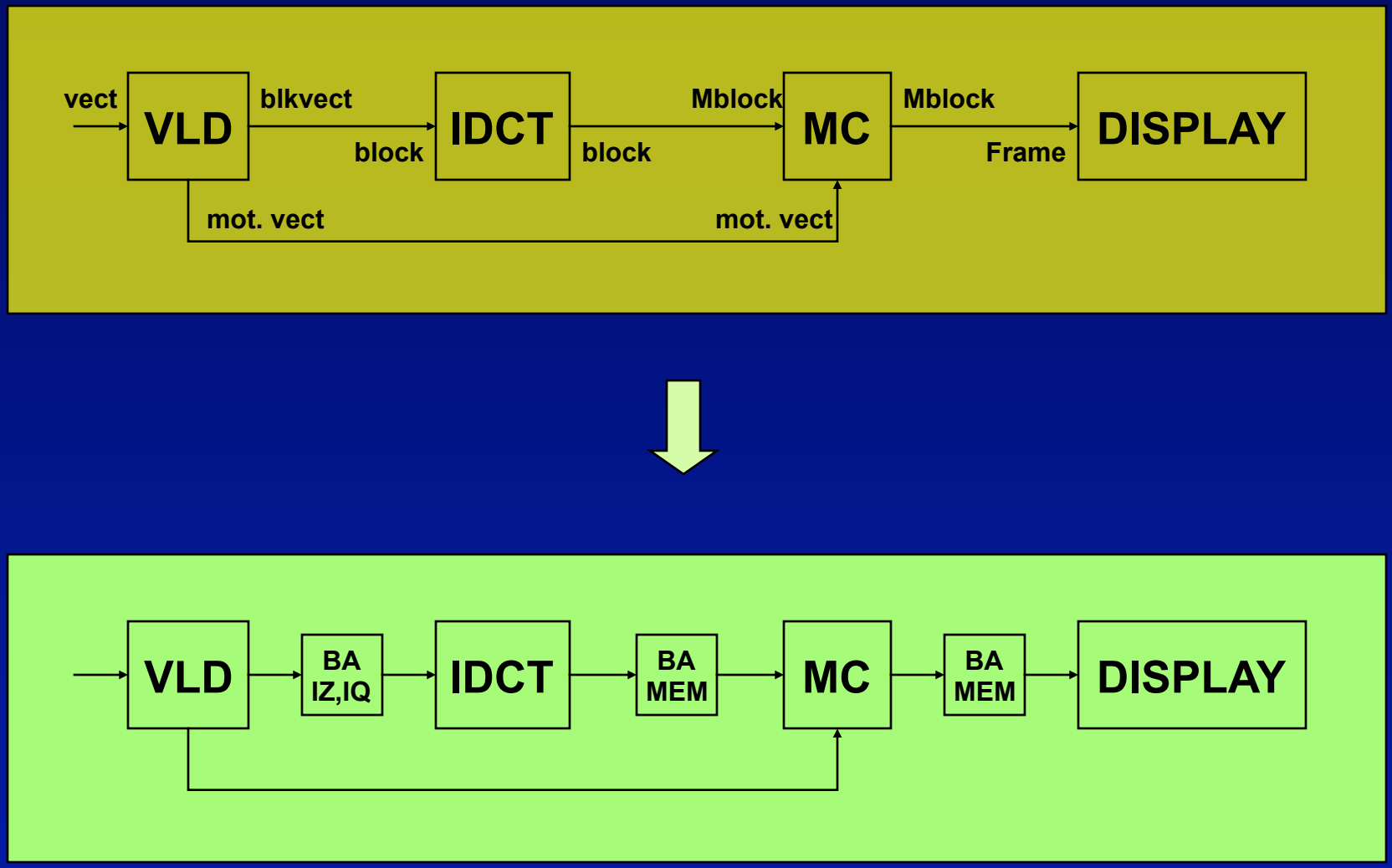
Behavior Adaptation

- **Behavior adapters**
  - match different domains, so that processes can understand each other
  - relation between two domains
  - not part of original system specification: needed because of the particular decomposition
  - needed independently of how the communication is performed



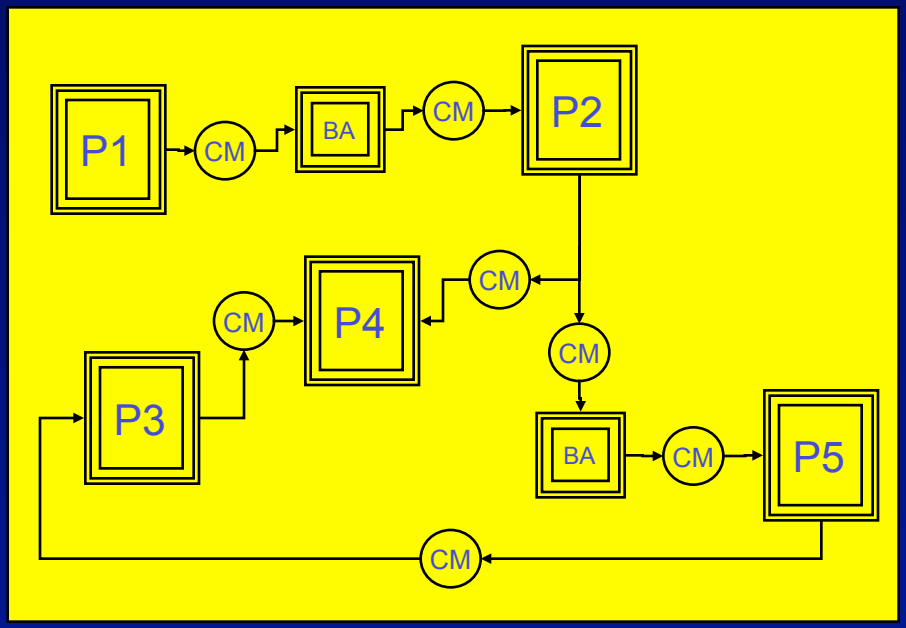


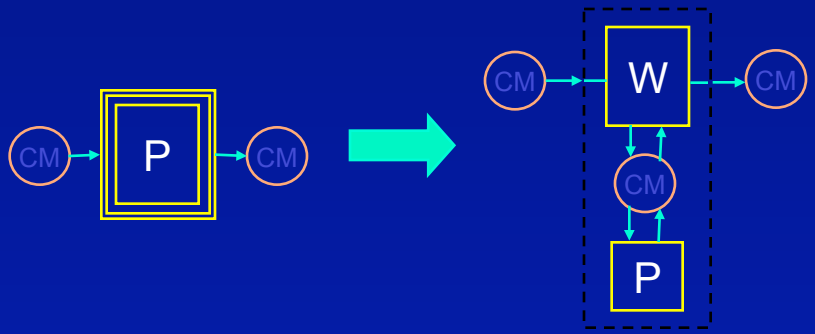
Behavior Adaptation (ex.)



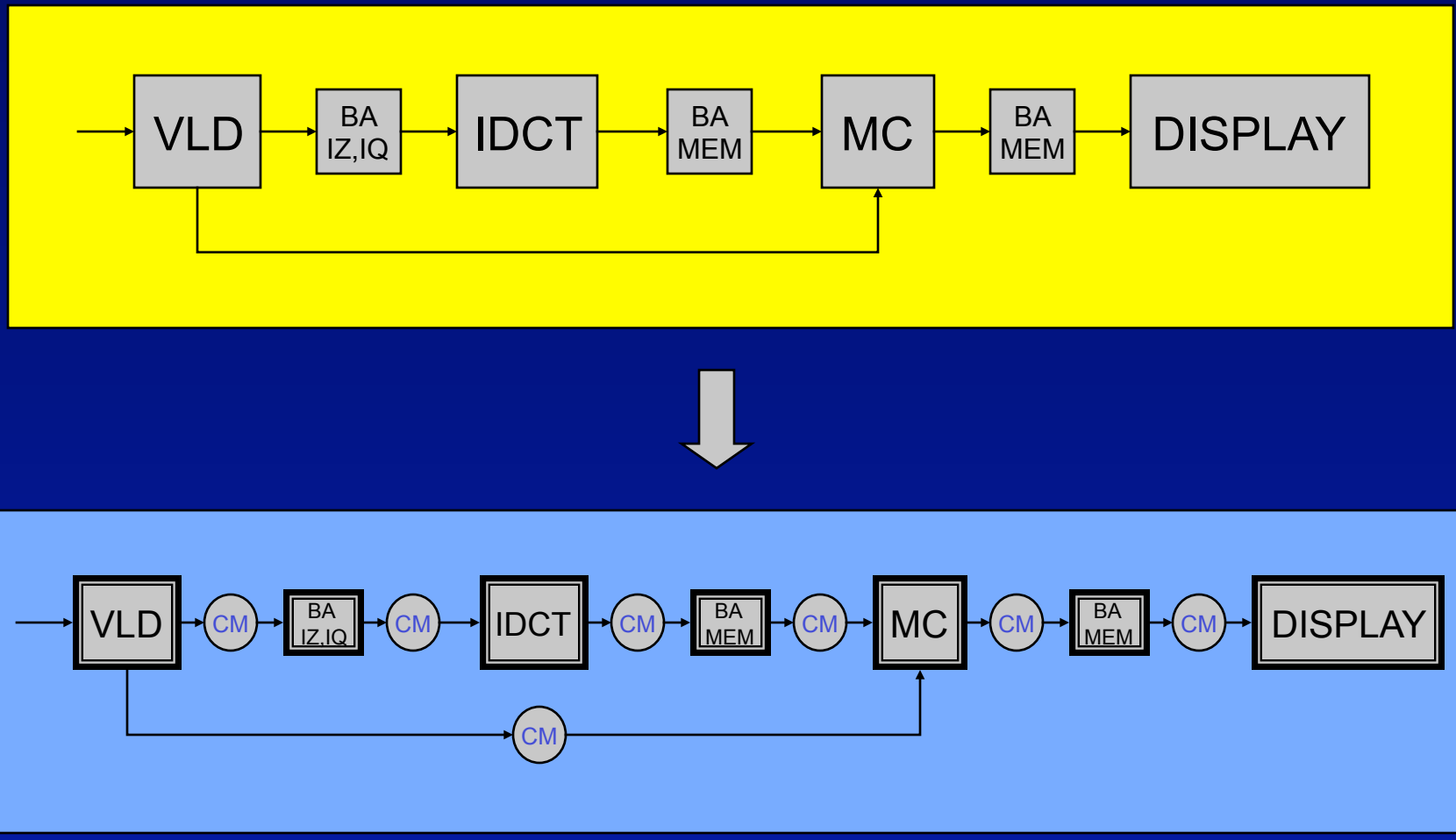
Communication and MoC

- **Communication medium**
  - each link needs a communication medium
  - does not affect or change the relation inside processes
- **MoC wrapper**
  - used to establish a firing rule and a communication semantics for each process
  - only the Moc wrapper is modified if a medium is changed





Communication and Moc (ex.)

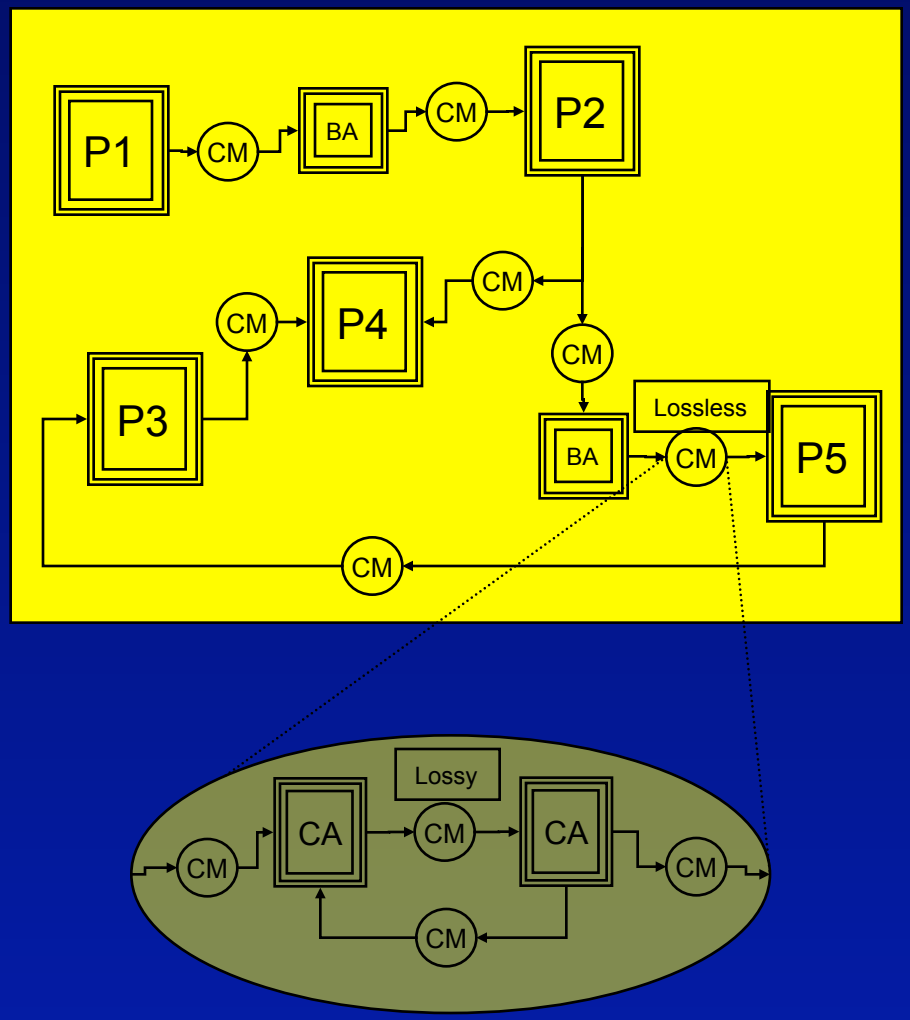


Refinement

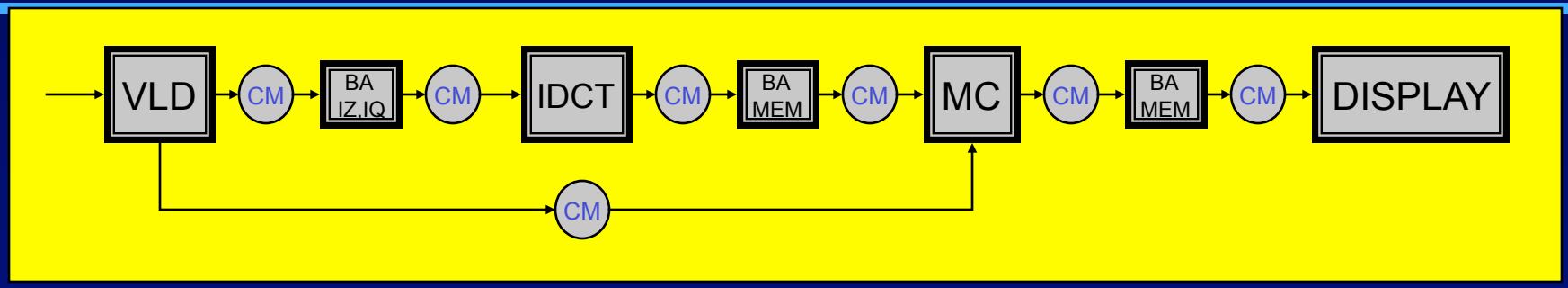
- **Refinement**
  - any communication medium can be refined into an arbitrary netlist, as long as the interface is not changed
- **Channel adapters**
  - used to preserve properties of a given interface
  - example:

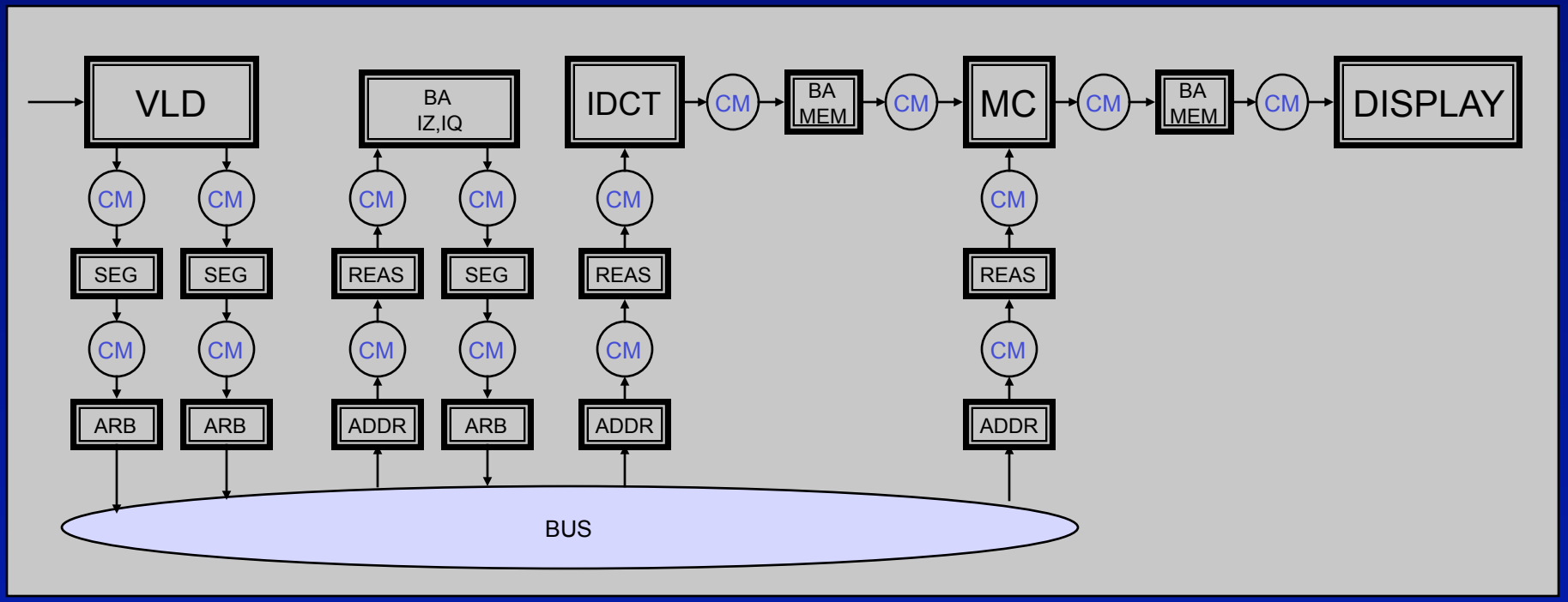
    lossless communication realized with a lossy medium

    (retransmission + acknowledge)



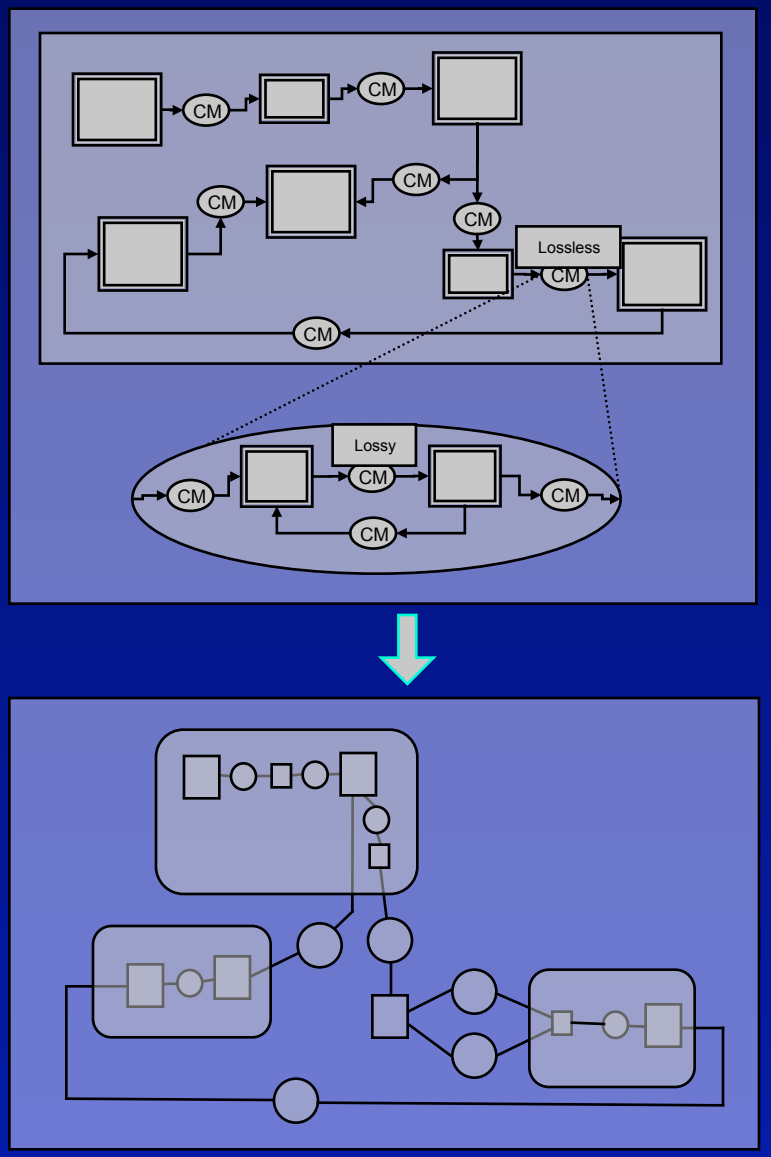
Refinement (ex.)





Mapping and Optimization

- **Optimization**
  - map each element (processes, adapters, media) onto architecture
  - merge processes, adapters and media into a single process, when applicable
  - provide an imperative description for each process



Mapping and Optimization (ex.)