EE382V:
Embedded System Design and Modeling

Lecture 11 – Communication Modeling & Refinement

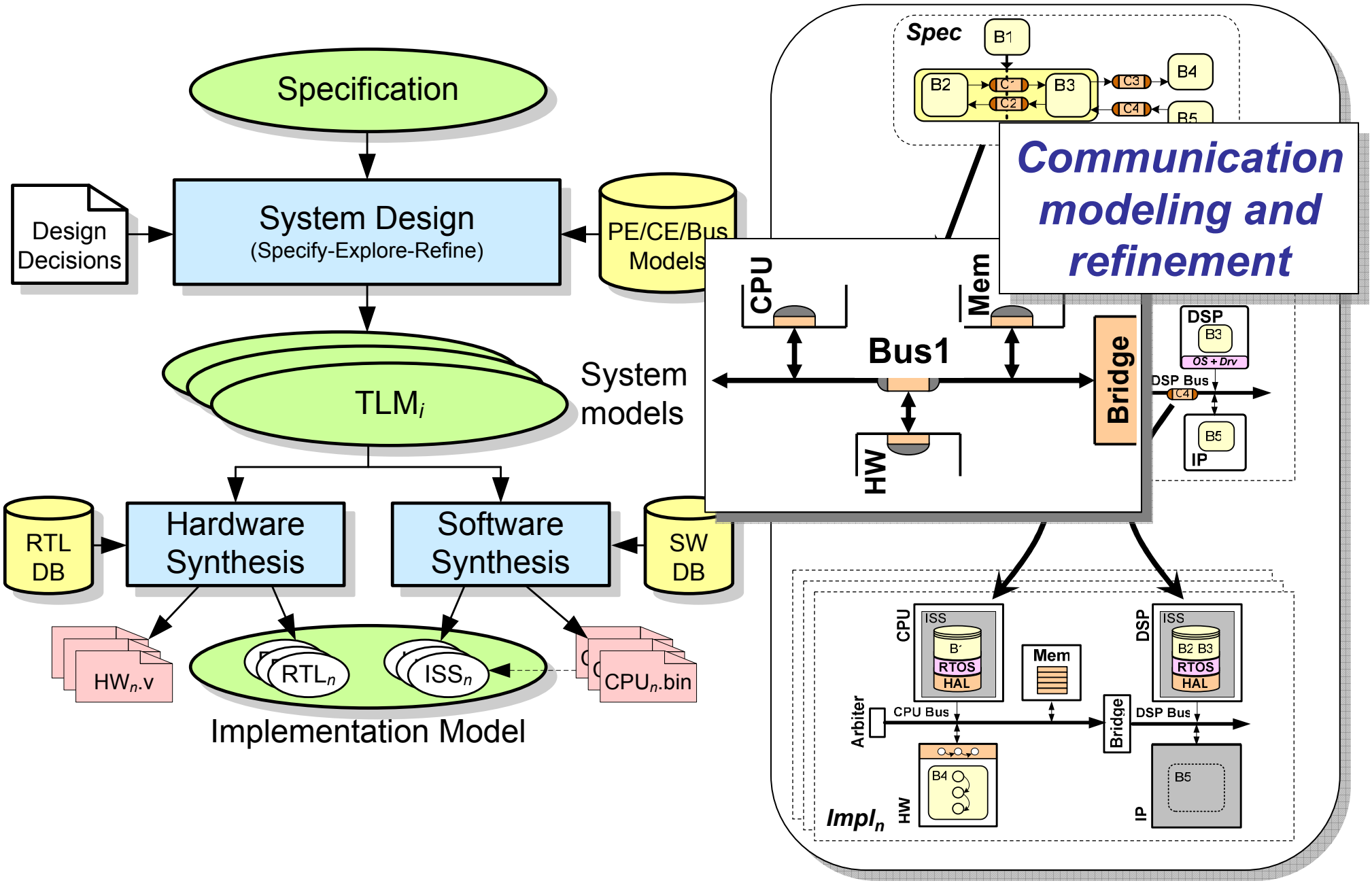
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Lecture 11: Outline

- **Communication layers**
  - Application
  - Network: presentation, session, transport
  - Communication: link, stream, media access
  - Protocol, physical

- **Communication synthesis**
  - Automatic layer-based generation
  - Protocol stack optimizations
  - Interface backend synthesis

System-On-Chip Environment (SCE)



© 2008 A. Gerstlauer

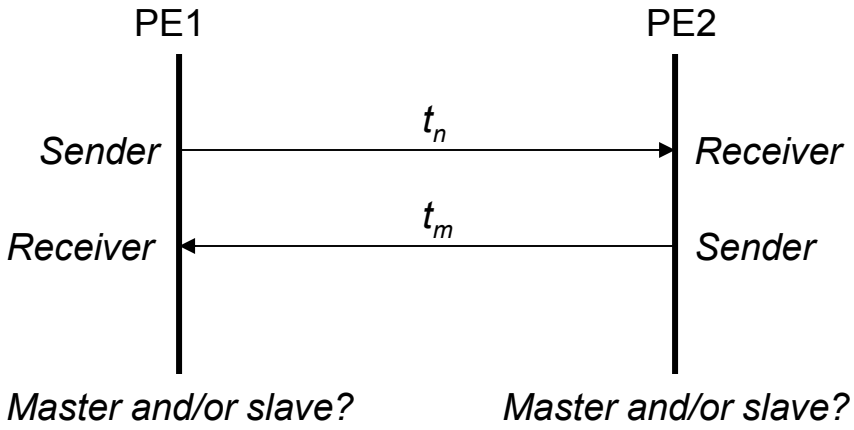
Communication Primitives

- **Events, transitions**
  - Pure control flow, no data
- **Shared variables**
  - No control flow, no synchronization
- **Synchronous message passing**
  - No buffering, two-way control flow
- **Asynchronous message passing**
  - Only control flow from sender to receiver guaranteed
  - May or may not use buffers (implementation dependent)
- **Queues**
  - Fixed, defined queue length (buffering)
- **Complex channels**
  - Semaphores, mutexes

➢ **Reliable communication primitives (lossless, error-free)**

Taxonomy of "Busses"

- **For each transaction between two communication partners**
  - 1 sender, 1 receiver
  - 1 master (initiator), 1 slave (listener)



- **Any combination of master/slave, sender/receiver**
  - Master/Slave bus
    - Statically fixed master/slave assignments for each PE pair
    - PEs can be masters, slaves or both (two ports for comm. with different PEs)
  - Node-based bus (e.g. Ethernet, CAN):
    - Sender is master, receiver is slave
- **Reliable (loss-less, error-free)??**

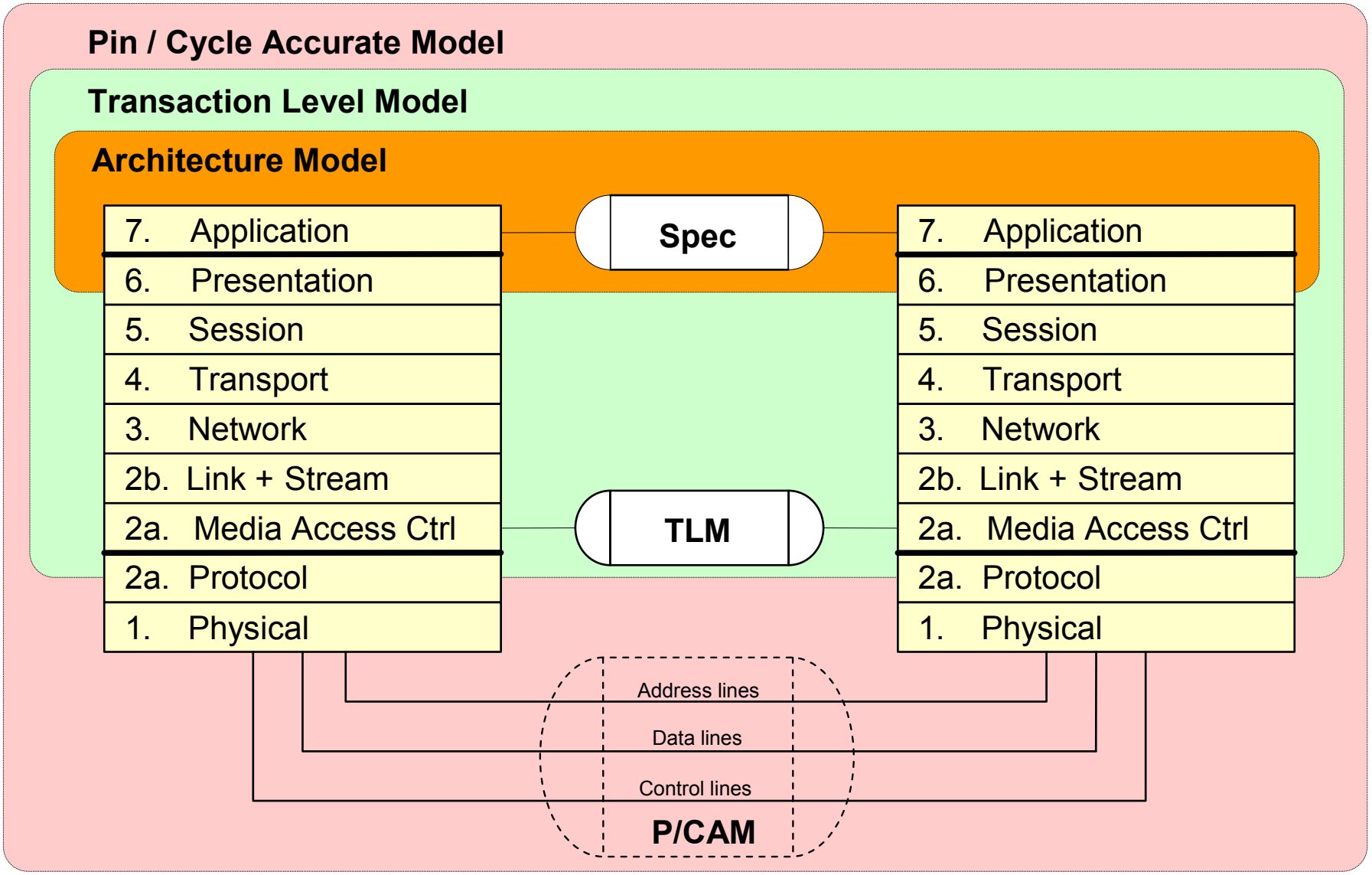
Communication Modeling

- **ISO/OSI 7-layer network model**

Layer	Semantics	Functionality	Implementation	OSI
Application	Channels, variables	Computation	Application	7
Presentation	End-to-end typed messages	Data formatting	Application	6
Session	End-to-end untyped messages	Synchronization, Multiplexing	OS kernel	5
Transport	End-to-end data streams	Packeting, Flow control, Error correction	OS kernel	4
Network	End-to-end packets	Routing	OS kernel	3
Link	Point-to-point logical links	Station typing, Synchronization	Driver	2b
Stream	Point-to-point control/data streams	Multiplexing, Addressing	Driver	2b
Media Access	Shared medium byte streams	Data slicing, Arbitration	HAL	2a
Protocol	Media (word/frame) transactions	Protocol timing	Hardware	2a
Physical	Pins, wires	Driving, sampling	Interconnect	1

- **A *model*, not an implementation !**

From OSI Model to System Models…



Source: G. Schirner

Lecture 11: Outline

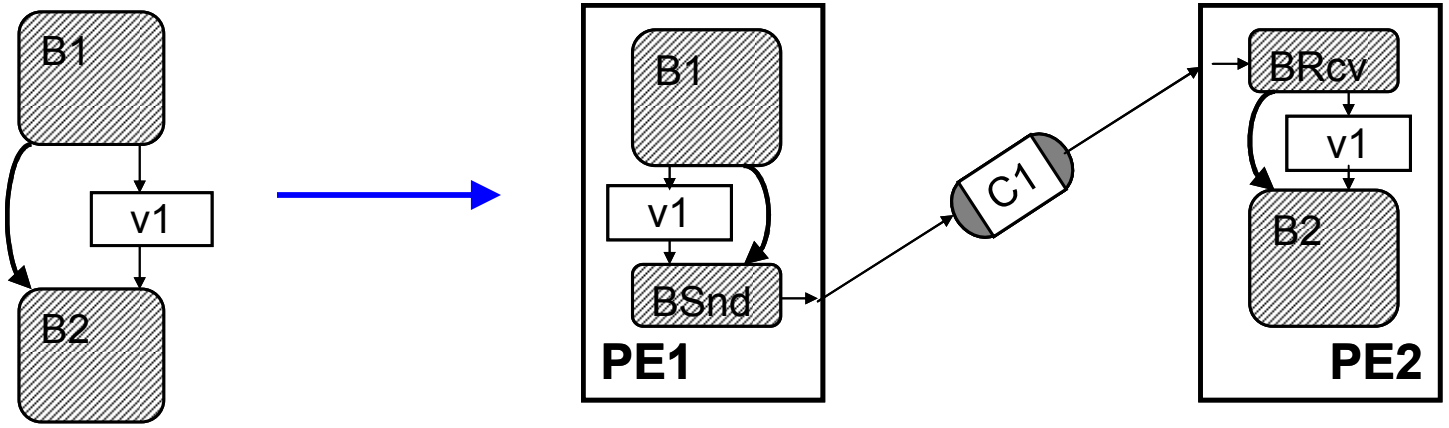
- **Communication layers**
  - Application
  - Network: presentation, session, transport
  - Communication: link, stream, media access
  - Protocol, physical

- **Communication synthesis**
  - Automatic layer-based generation
  - Protocol stack optimizations
  - Interface backend synthesis

Source: A. Gerstlauer, G. Schirner, D. Shin, J. Peng, "Necessary and Sufficient Functionality and Parameters for SoC Communication," CECS-TR-06-01

Architecture Model: Application Layer (1)

- **Synchronization**
  - Synthesize control flow

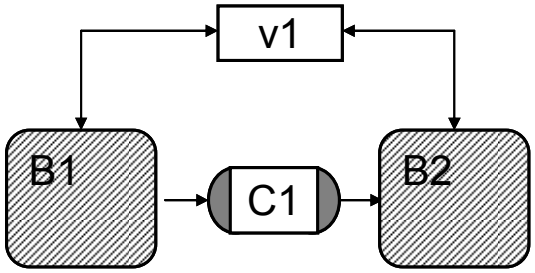


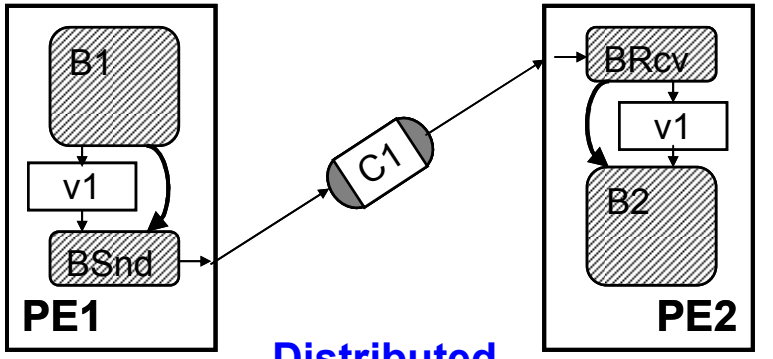
Parallel processes plus synchronization events

  - ➢ Implement sequential transitions across parallel components

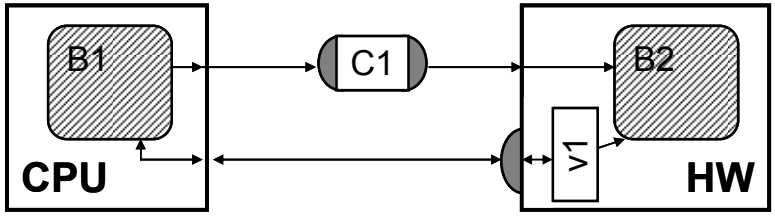
Architecture Model: Application Layer (2)

- **Storage**
  - Shared variable mapping to memories

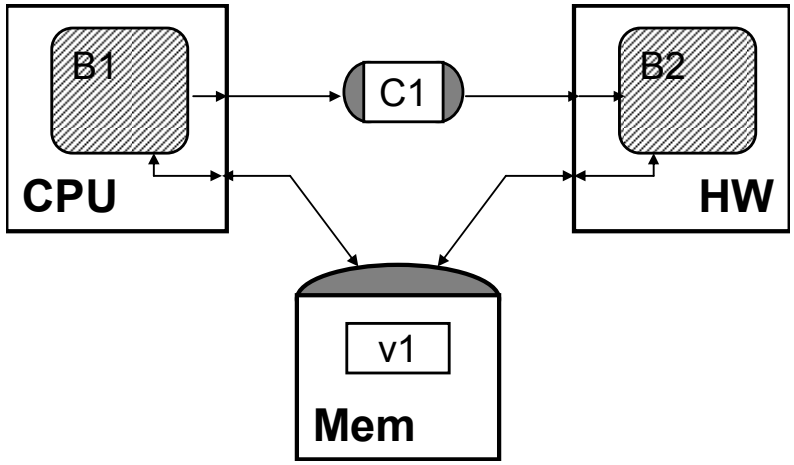




Distributed



Memory-mapped I/O

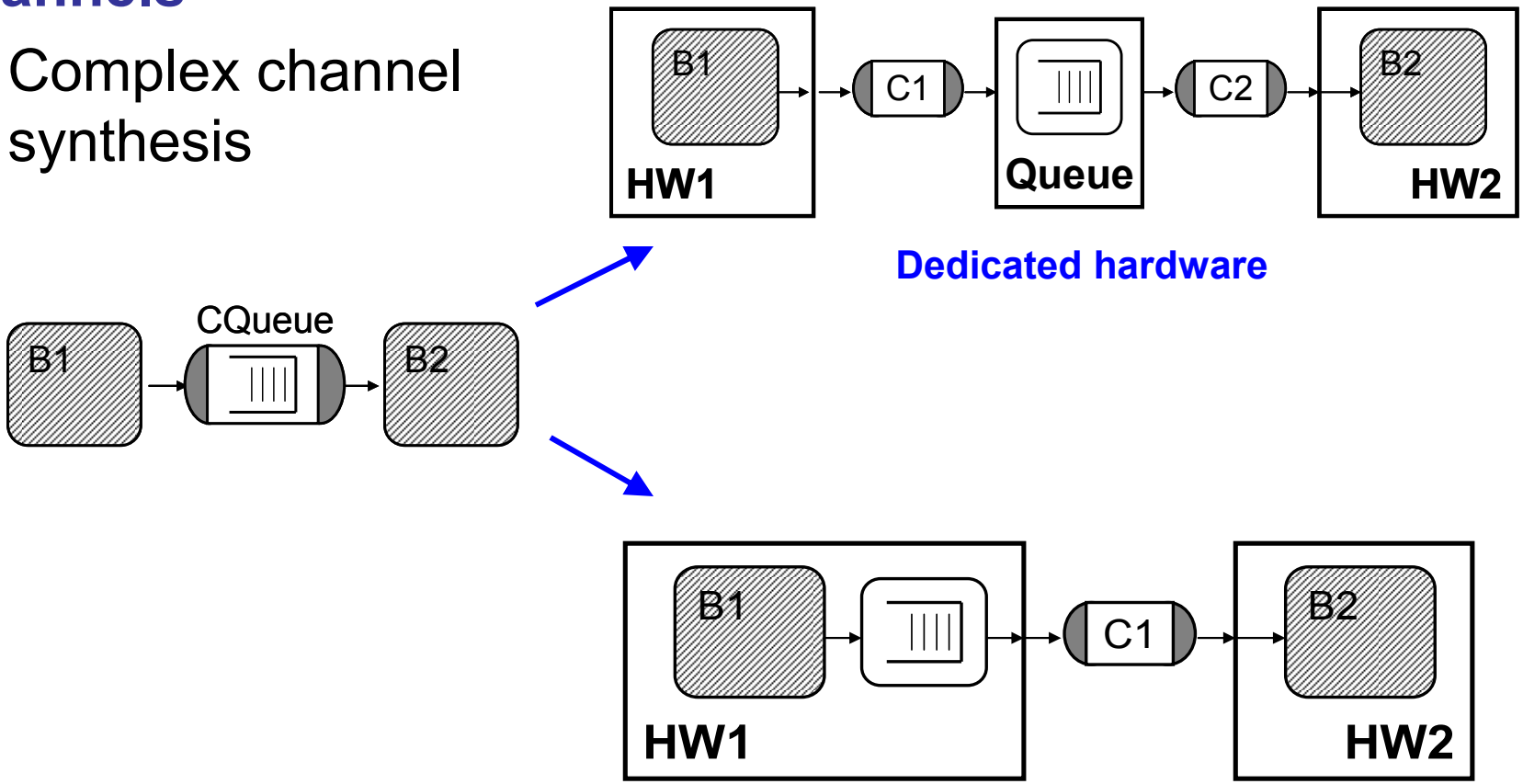


Shared memory

  - ➢ Map global storage to local component memories

Architecture Model: Application Layer (3)

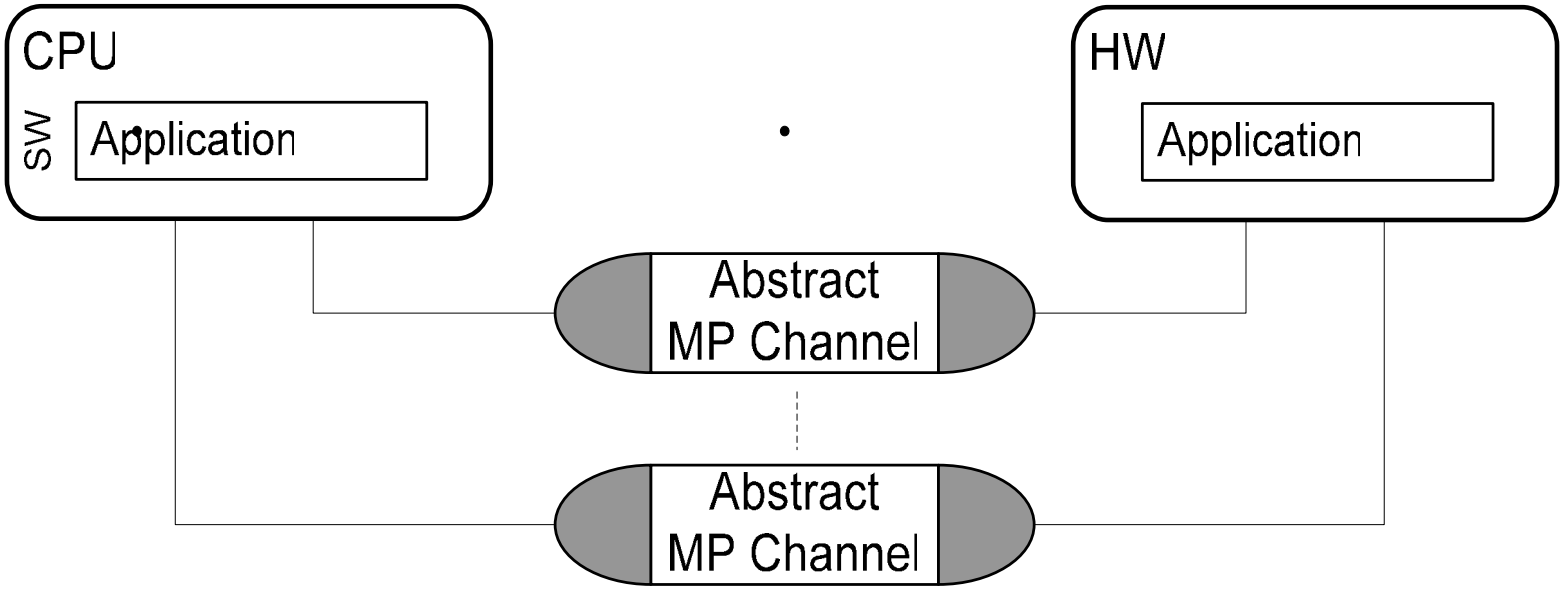
- **Channels**
  - Complex channel synthesis



  - ➢ Implement channel as separate process + RPC channels

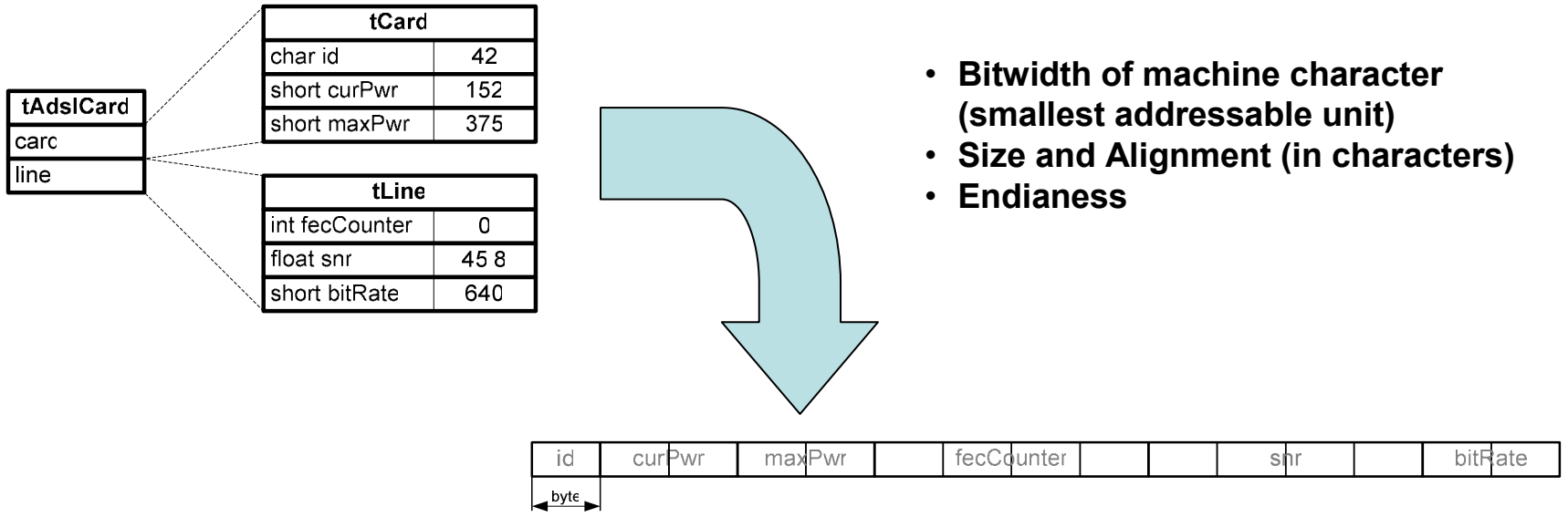
Architecture Model

- **Virtual system architecture**
  - Computation
    - PEs (functionality)
    - Memories (storage)
  - Abstract end-to-end communication
    - Sync./async. message-passing
    - Memory interfaces
    - Events



Network Model: Presentation Layer

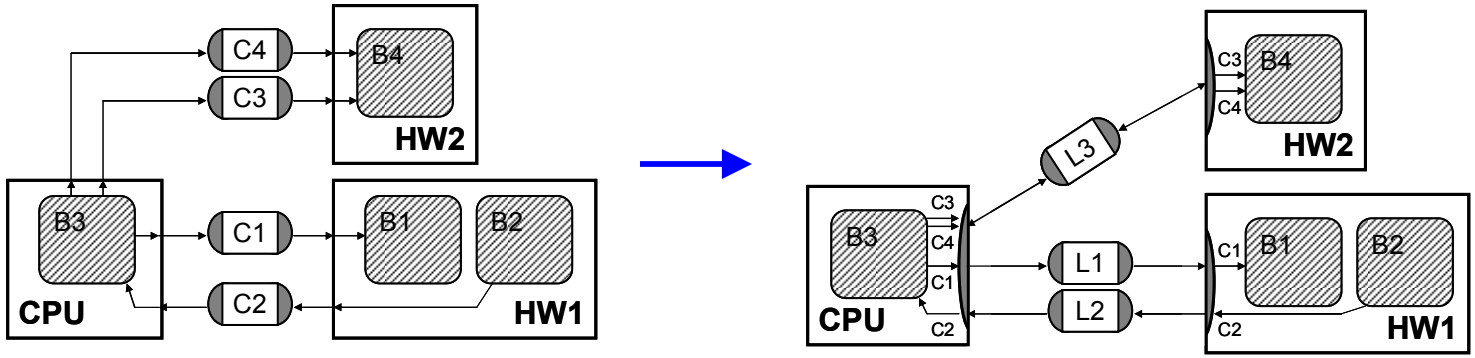
- **Data formatting**
  - Format abstract data types into canonical network byte layout
    1. Global network data layout
    2. Shared, optimized layout for each pair of communicating PEs
  - ➢ Convert typed messages into untyped, ordered byte streams
  - ➢ Convert variables into memory byte layout



- **Bitwidth of machine character (smallest addressable unit)**
- **Size and Alignment (in characters)**
- **Endianess**

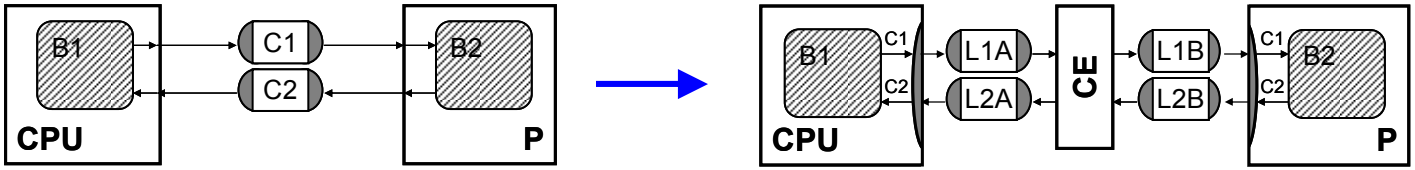
Network Model: Session Layer

- **Channel merging**
  - Merge application channels into a set of untyped end-to-end message streams
    1. Unconditionally merge sequential channels
    2. Merge (concurrent) channels with additional session ID (message header)
  - ➢ Channel selection over end-to-end transports

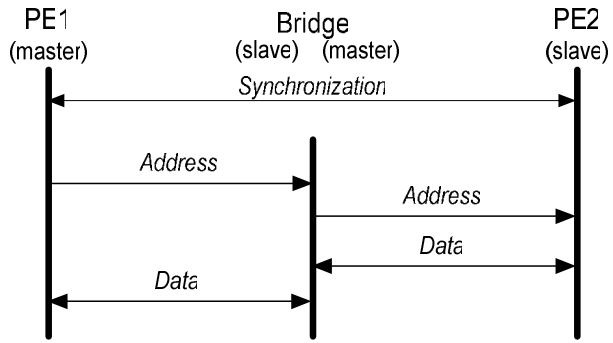


Network Model: Network Layer

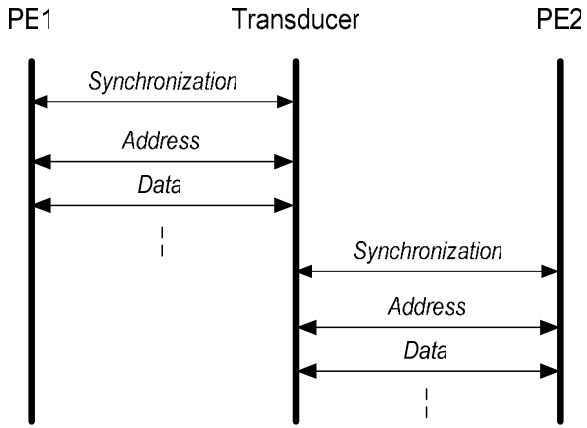
- **CE synthesis**
  - Routing of end-to-end paths over point-to-point links
  - Insert communication elements (CEs) to connect busses



  - Bus bridging
    - Transparently connect slave & master sides at protocol level
    - Bridges maintain synchronicity, no buffering



  - Transducers
    - Store-and-forwarding of data packets between incompatible busses
    - Intermediate buffering, results in asynchronous communication

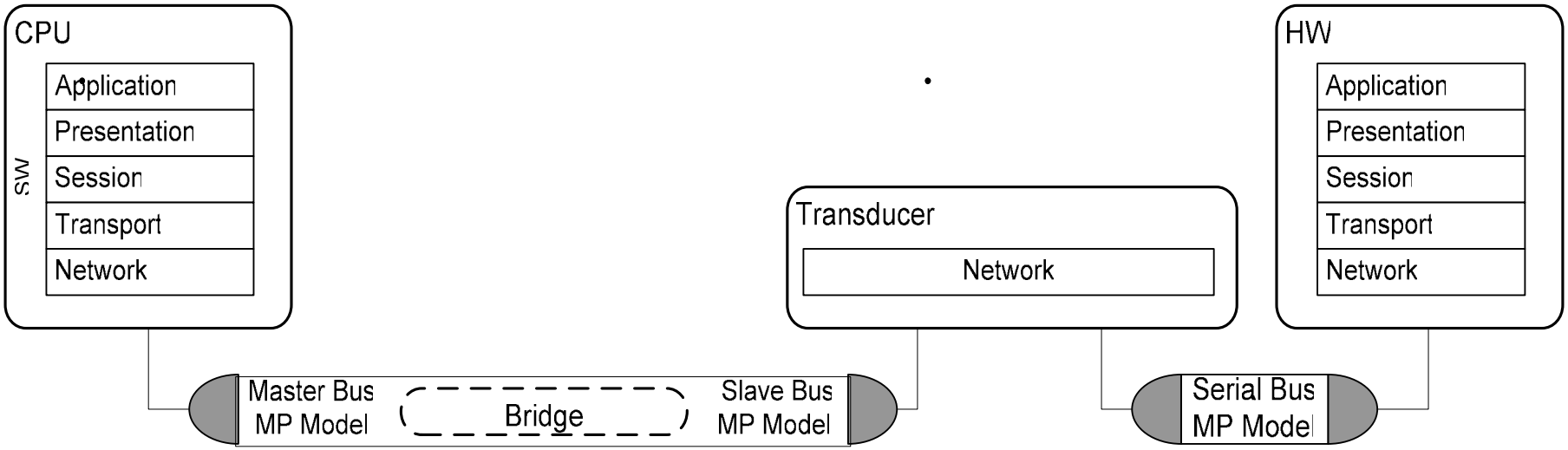


Network Model: Transport Layers

- **Packeting and routing**
  - Packetization to reduce buffer sizes
    1. Fixed packet sizes (plus padding)
    2. Variable packet size (plus length header)
  - Protocol exchanges (ack) to restore synchronicity
    - Iff synchronous message passing and transducer in the path
  - Packet switching and identification (logical routing)
    1. Dedicated logical links (defer identification to lower layers)
    2. Network endpoint addressing (plus packet address headers)
  - Physical routing in case of multiple paths between PEs
    1. Static (predetermined) routing based on connectivity or packet ID headers
    2. Dynamic (runtime) routing

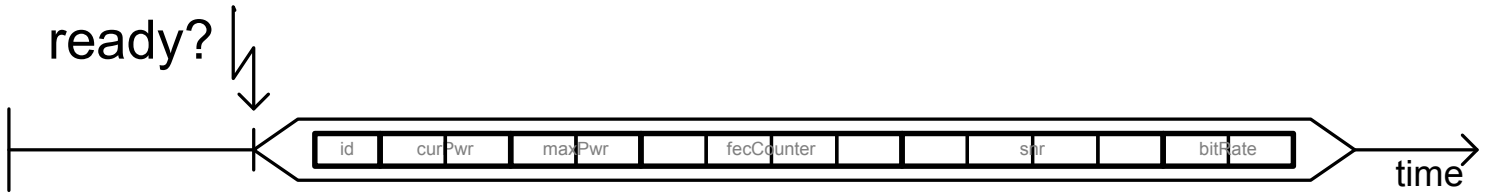
Network Model

- **Topology of communication architecture.**
  - PEs + Memories + CEs
  - Upper protocol layers inserted into PEs/CEs
  - Communication via point-to-point links
    - Synchronous packet transfers (data transfers)
    - Memory accesses (shared memory, memory-mapped I/O)
    - Events (control flow)



Communication Model: Link Layer (1)

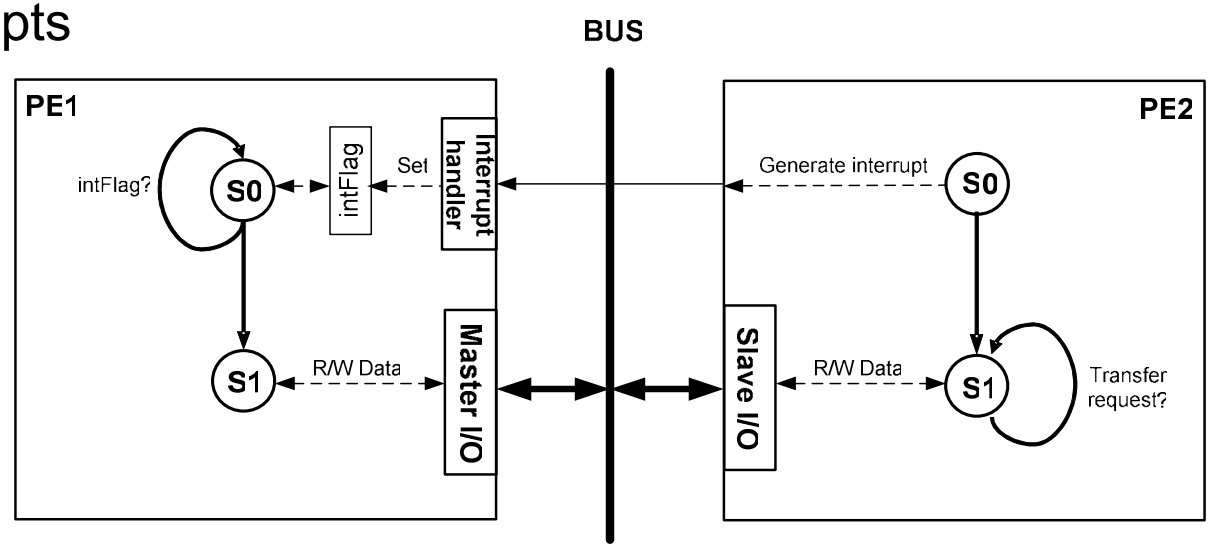
- **Synchronization (1)**
  - Ensure slave is ready before master initiates transaction
    1. Always ready slaves (memories and memory-mapped I/O)
    2. Defer to fully synchronized bus protocol (e.g. RS232)
    3. Separate synchronization mechanism



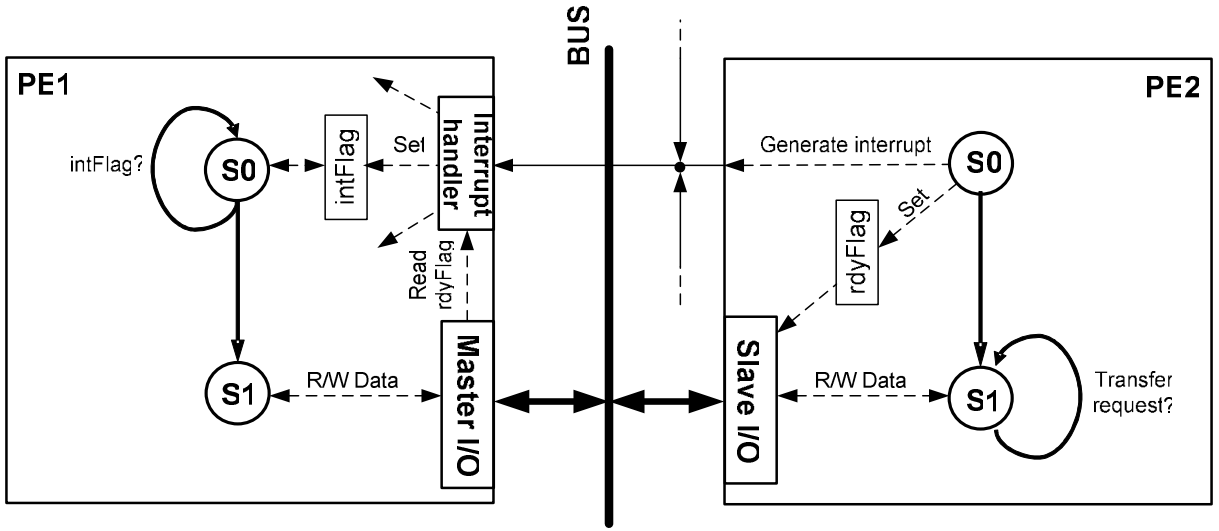
  - ➢ Events from slave to master for master/slave busses
  - ➢ Synchronization packets for node-based busses

Communication Model: Link Layer (2)

- **Synchronization (2)**
  - Dedicated interrupts

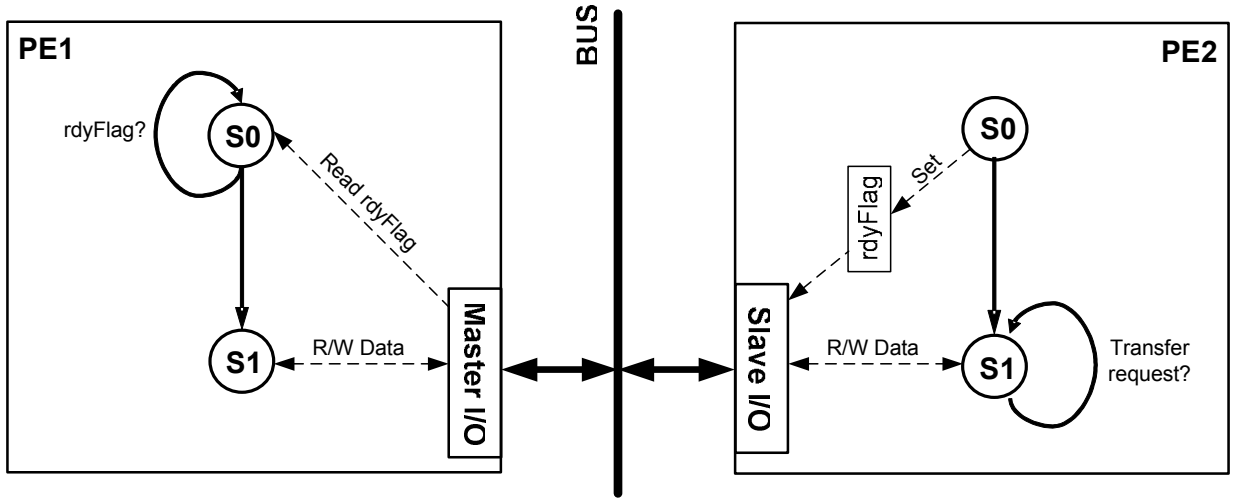


  - Shared interrupts

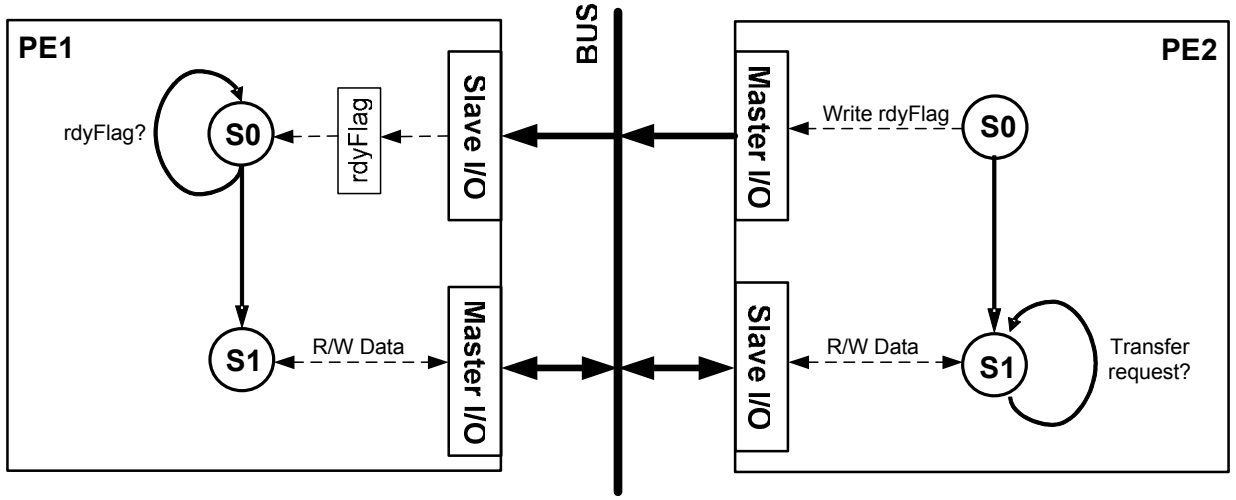


Communication Model: Link Layer (3)

- **Synchronization (3)**
  - Slave polling



  - Flag in master

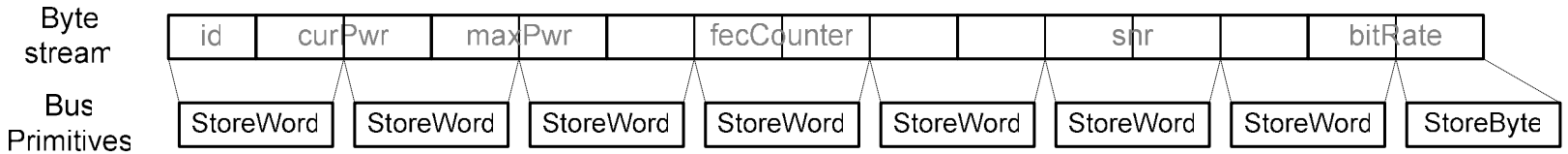


Communication Model: Stream Layer

- **Addressing**
  - Multiplexing of links over shared medium
  - Separation in space through addressing
  - ➢ Assign physical bus addresses to links
    1. Dedicated physical addresses per link
    2. Shared physical addresses plus packet ID/address in packet header

Communication Model: MAC Layer

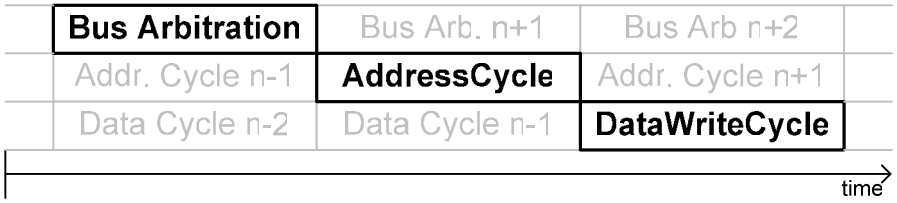
- **Data slicing and arbitration**
  - Split data packets into multiple bus word/frame transactions
  - Separate individual bus transactions in time through arbitration



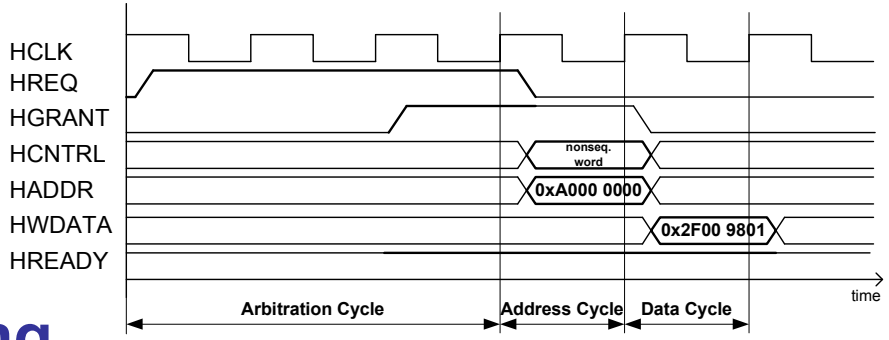
  - ➢ Optimized data slicing utilizing supported bus modes (e.g. burst)
  - ➢ Insertion of arbiters in case of centralized arbitration schemes

Communication Model: Protocol, Physical Layer

- **Bus interface**
  - Generate state machines implementing bus protocols
  - Timing-accurate based on timing diagrams and timing constraints



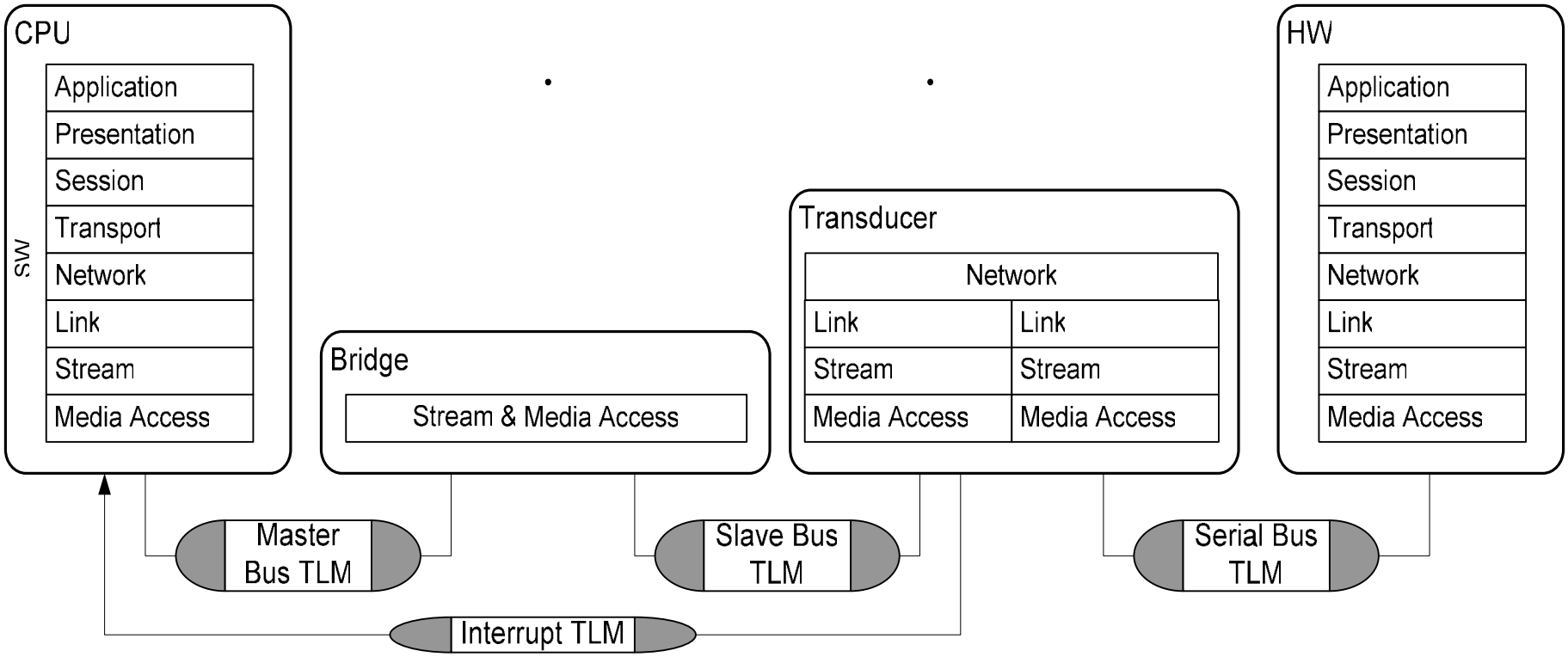
  - ➢ Bus protocol database



- **Port mapping and bus wiring**
  - Connectivity of component ports to bus, interrupt wires/lines
  - ➢ Generate top-level system netlist

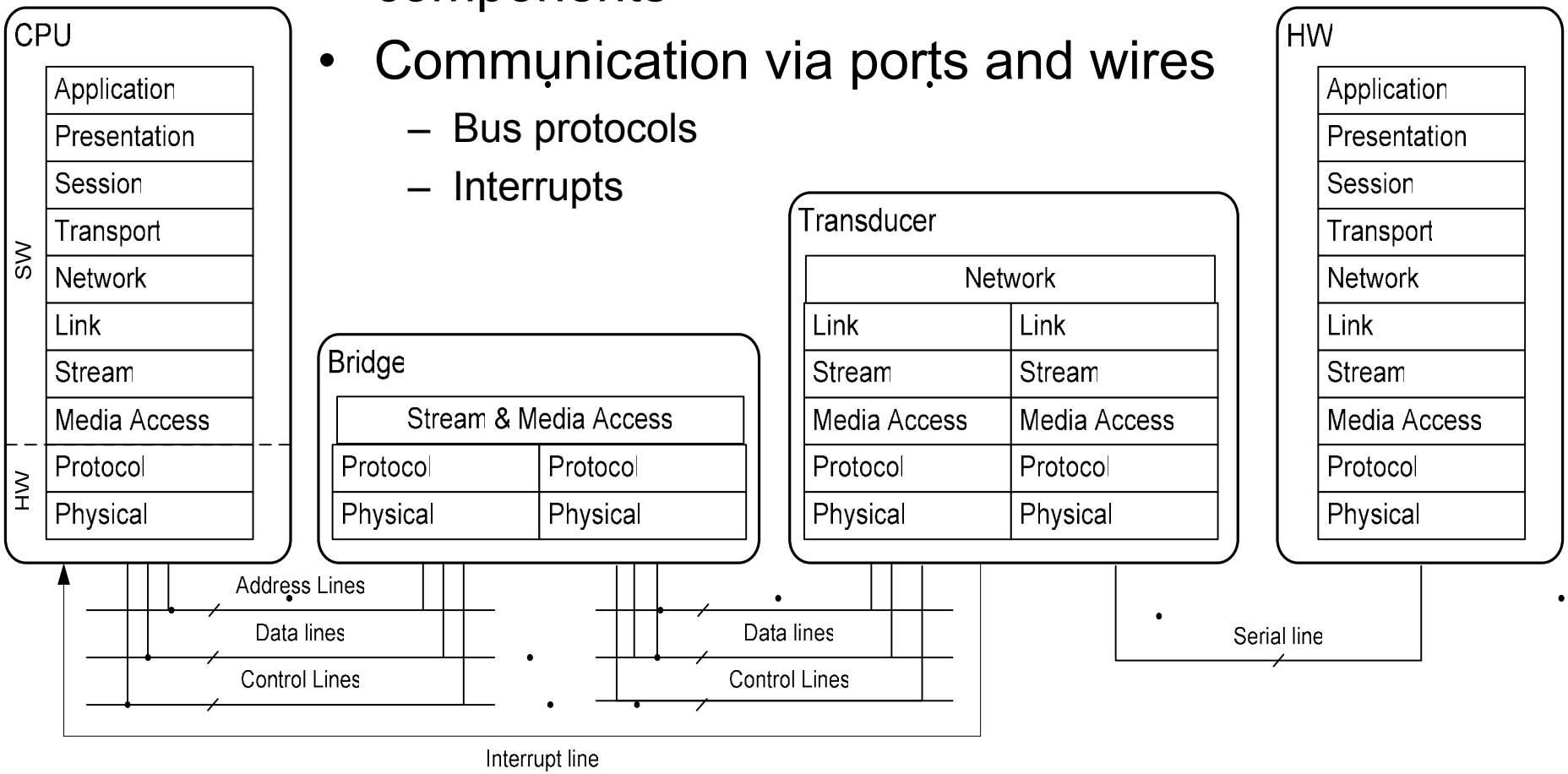
Transaction-Level Model (TLM)

- **Abstract component & bus structure/architecture**
  - PEs + Memories + CEs + Busses
  - Communication layers down to protocol transactions
  - Communication via transaction-level channels
    - Bus protocol transactions (data transfers)
    - Synchronization events (interrupts)



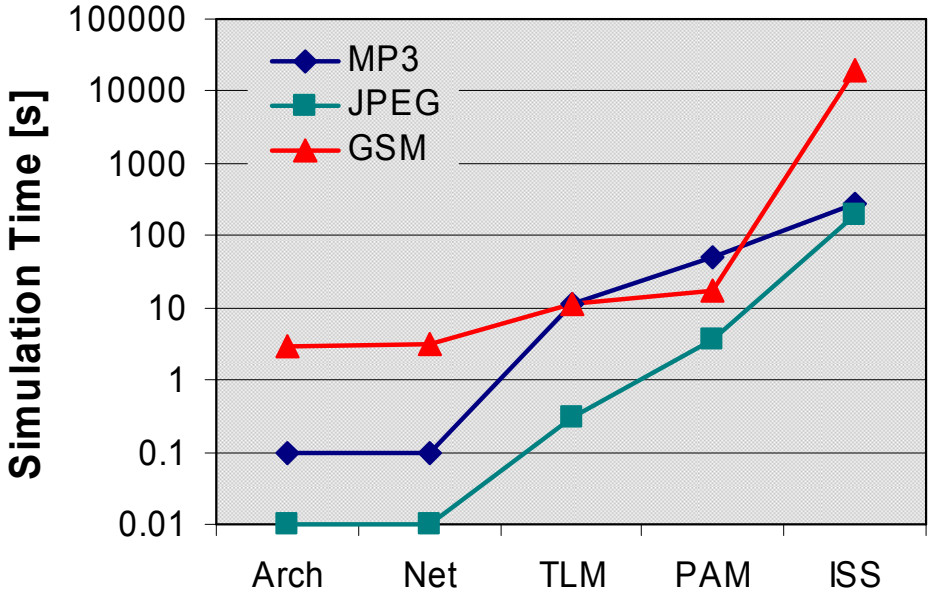
Pin-Accurate Model (PAM)

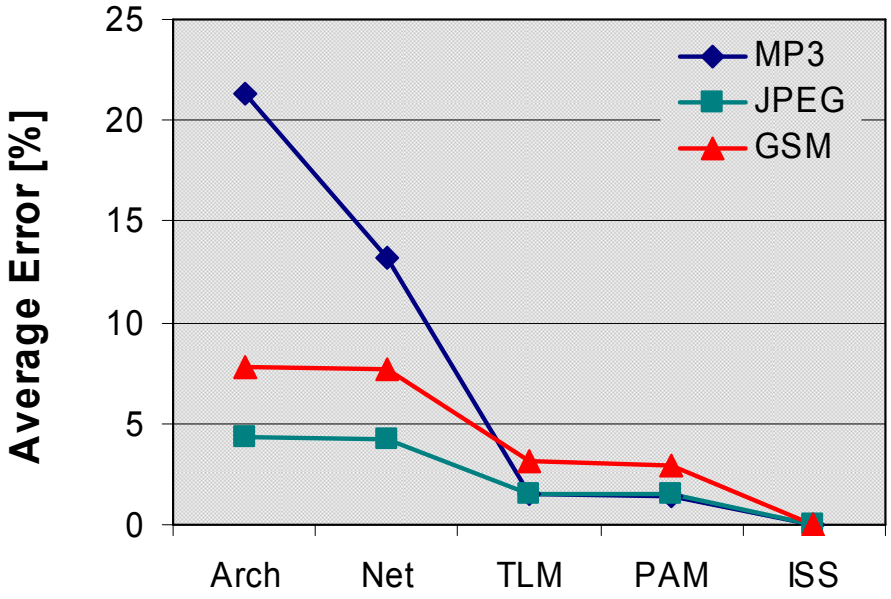
- **Component & bus structure/architecture**
  - PEs + Memories + CEs + Busses
  - Pin-, timing- and bit-accurate bus-functional components
  - Communication via ports and wires
    - Bus protocols
    - Interrupts



Communication Modeling Results

- **Standalone, single-processor systems**
  - ARM platform
    - MP3 player with HW accelerators
    - JPEG encoder
  - DSP platform
    - GSM voice coder/decoder with HW co-processor
  - Simulated on Sun Fire V240 (1.5 GHz)
    - 1.5 second MP3
    - 640x480 picture
    - 1.5 speech GSM





Source: G. Schirner, A. Gerstlauer, R. Doemer

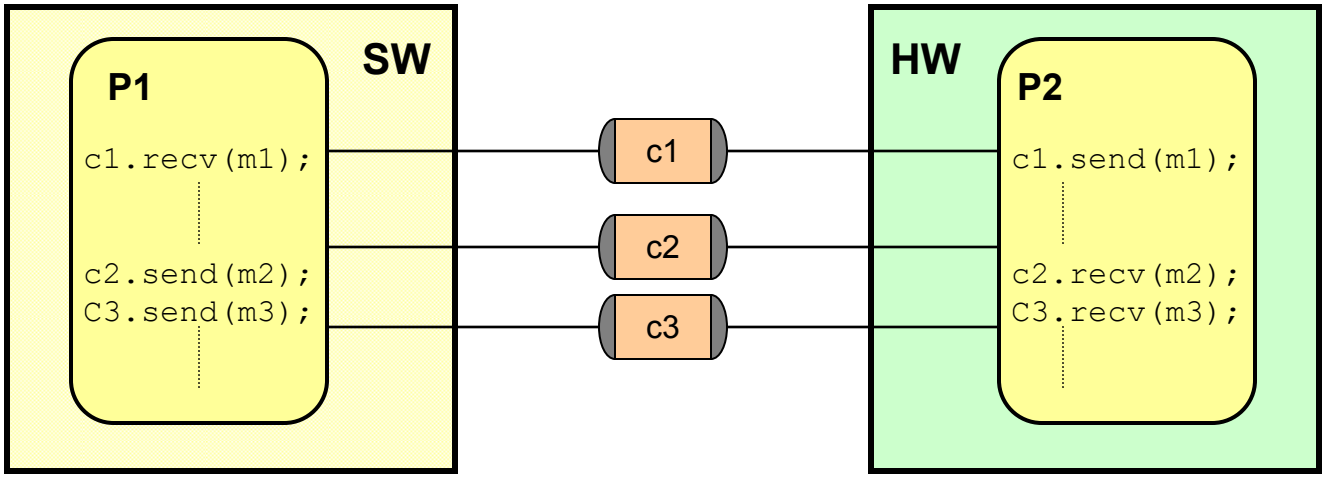
Lecture 11: Outline

- ✓ Communication layers
  - ✓ Application
  - ✓ Network: presentation, session, transport
  - ✓ Communication: link, stream, media access
  - ✓ Protocol, physical

- **Communication synthesis**
  - Automatic layer-based generation
  - Protocol stack optimizations
  - Interface backend synthesis

Source: A. Gerstlauer, D. Shin, J. Peng, R. Dömer, D. Gajski, "Automatic, Layer-based Generation of System-On-Chip Bus Communication Models," TCAD07

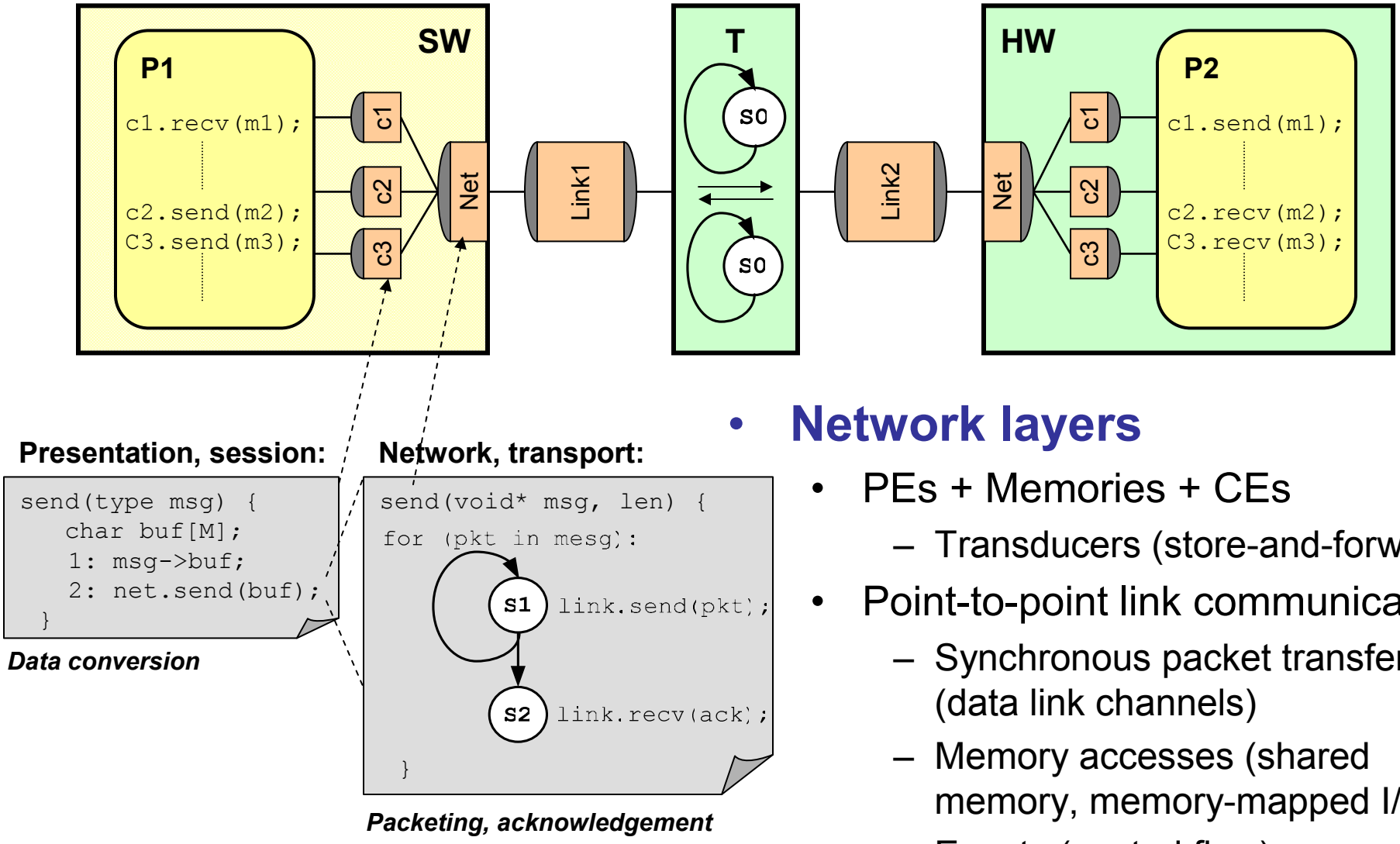
Architecture Model



- **Application layer (virtual system architecture)**
  - Computation
    - PEs (functionality)
    - Memories (storage)
  - Abstract end-to-end communication
    - Queues, semaphores
    - Sync./async. message-passing
    - Shared variables/memories
    - Events, transitions

➢ **Reliable, loss-less application communication**

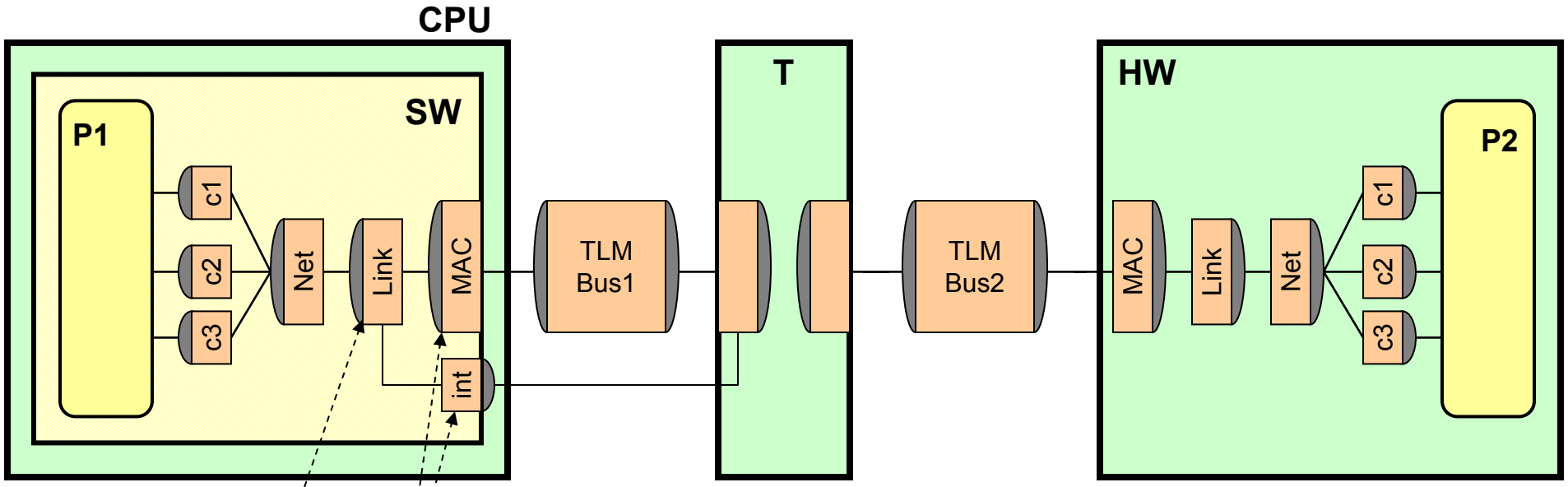
Network Model



- **Network layers**
  - PEs + Memories + CEs
    - Transducers (store-and-forward)
  - Point-to-point link communication
    - Synchronous packet transfers (data link channels)
    - Memory accesses (shared memory, memory-mapped I/O)
    - Events (control flow)

➢ **Communication topology**

Transaction-Level Model



Link, stream:

```
send(void* p, len) {
  S1  wait(intr);
  S2  mac.write(p,
                addr);
}
```

Synchronization, addressing

Media access:

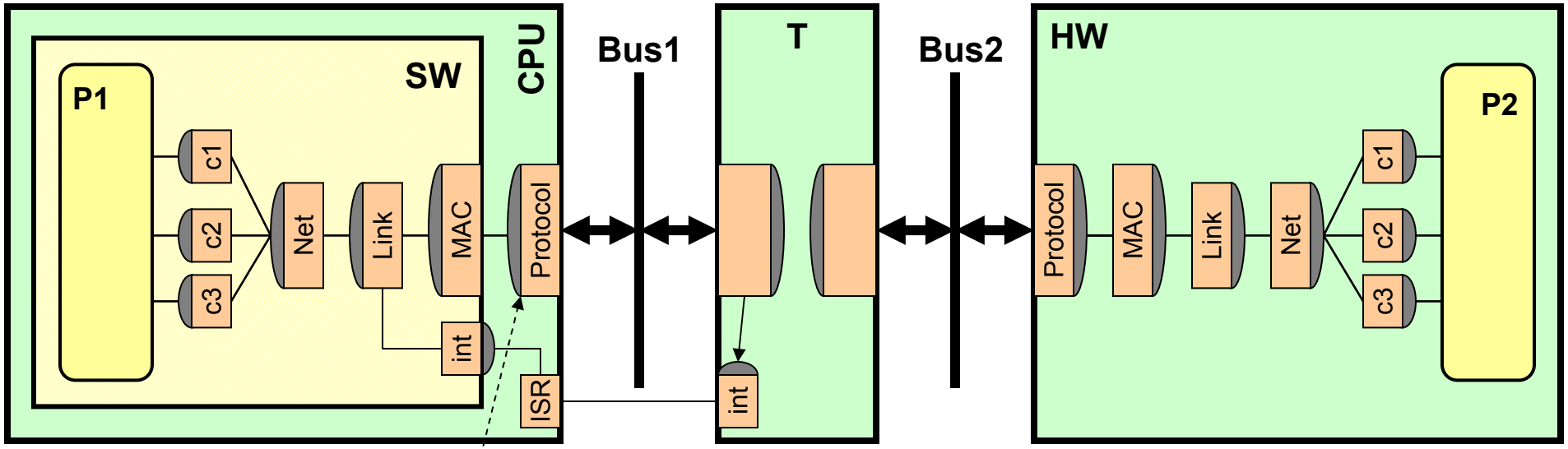
```
intHandler() {
   notify(intr);
}

write(void* d, len, a) {
  S0  bus.writeWord(a,w);
}
```

Data slicing

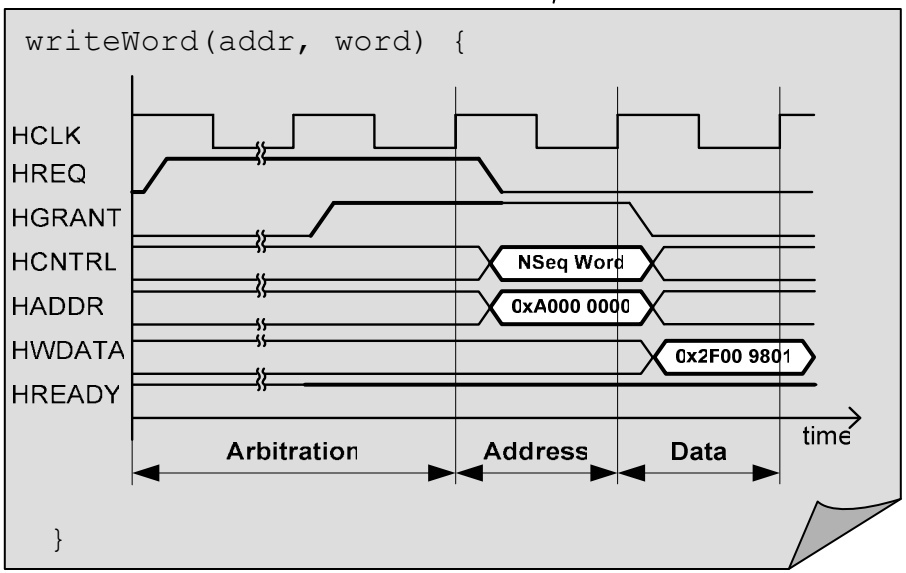
- **Link layers**
  - PEs + Memories + CEs + Busses
    - Bus bridges
  - Communication via bus transactions (bus TLM)
    - Address, data, arbitration
    - Synchronization (interrupts, polling)

➢ **System communication architecture**

Pin-Accurate Model (PAM)



Protocol, physical:

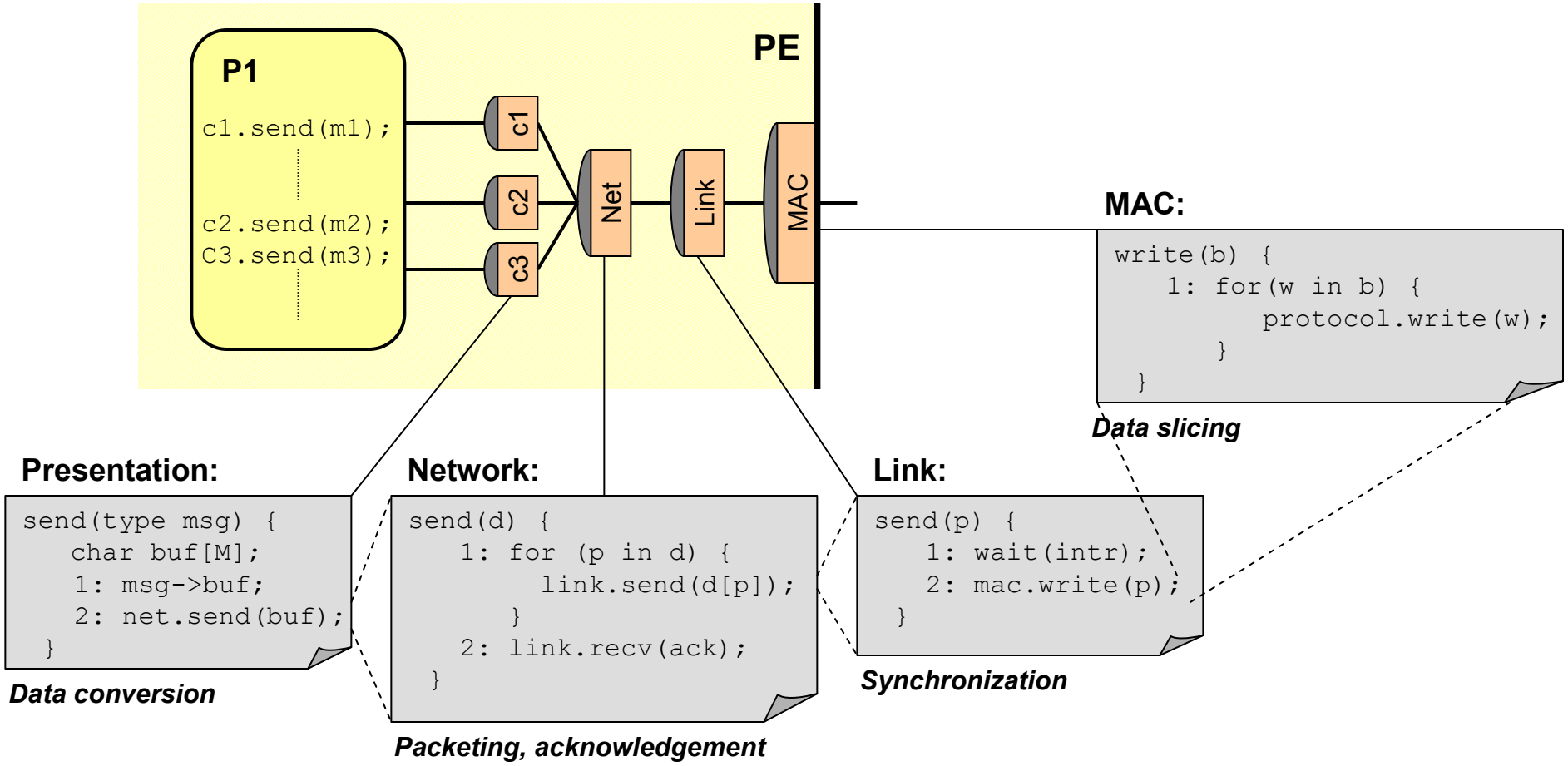


Protocol timing, wire driving & sampling

- **Bus-functional layers**
  - PEs + Memories + CEs + Busses
    - Pin-, cycle- and bit-accurate bus-functional components
  - Communication via ports and wires
    - Address, data, control busses
    - Interrupts

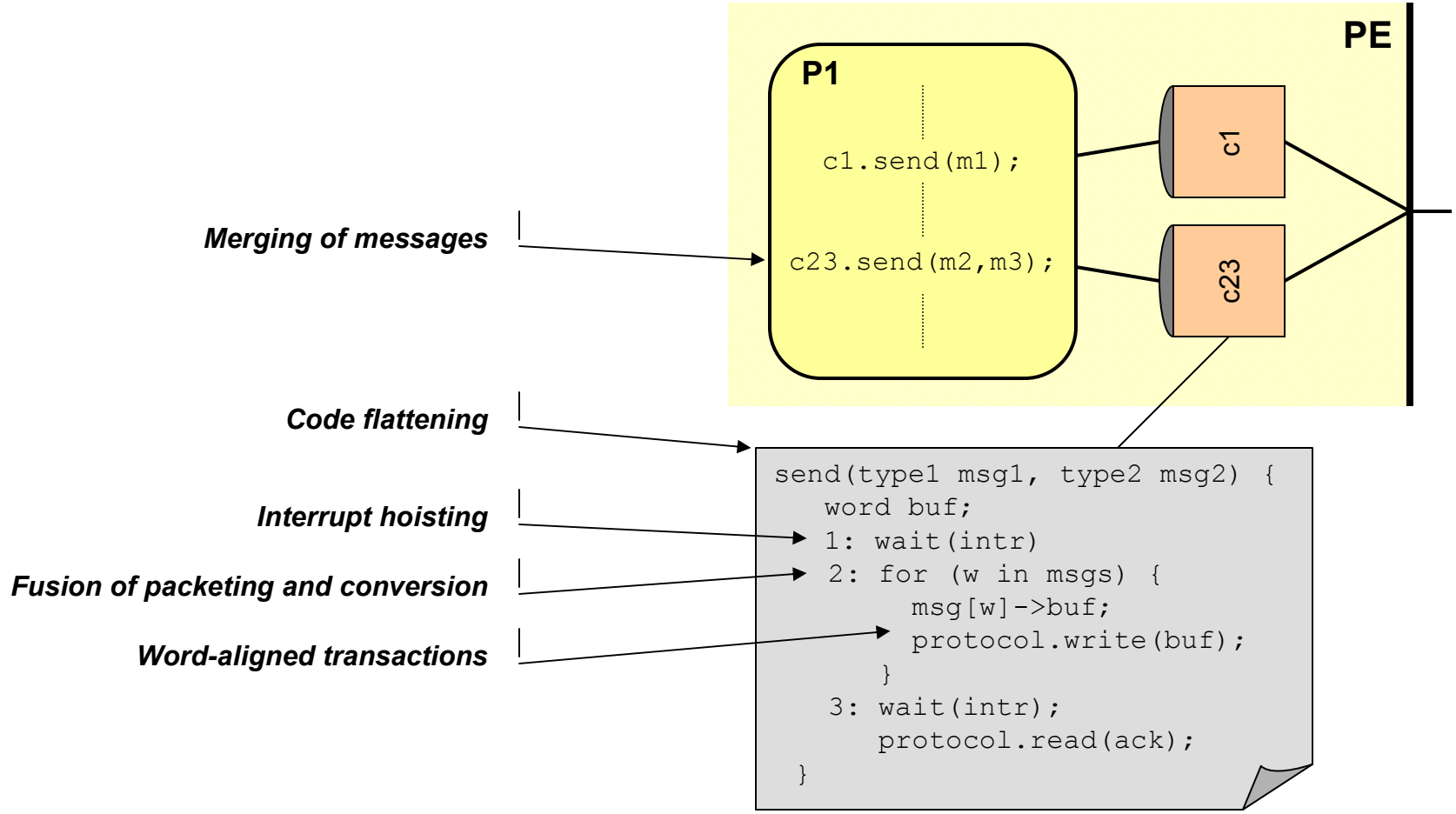
Protocol Stack Optimizations (1)

- **Automatically generated code during refinement**
  - Layer-based code organization and separation

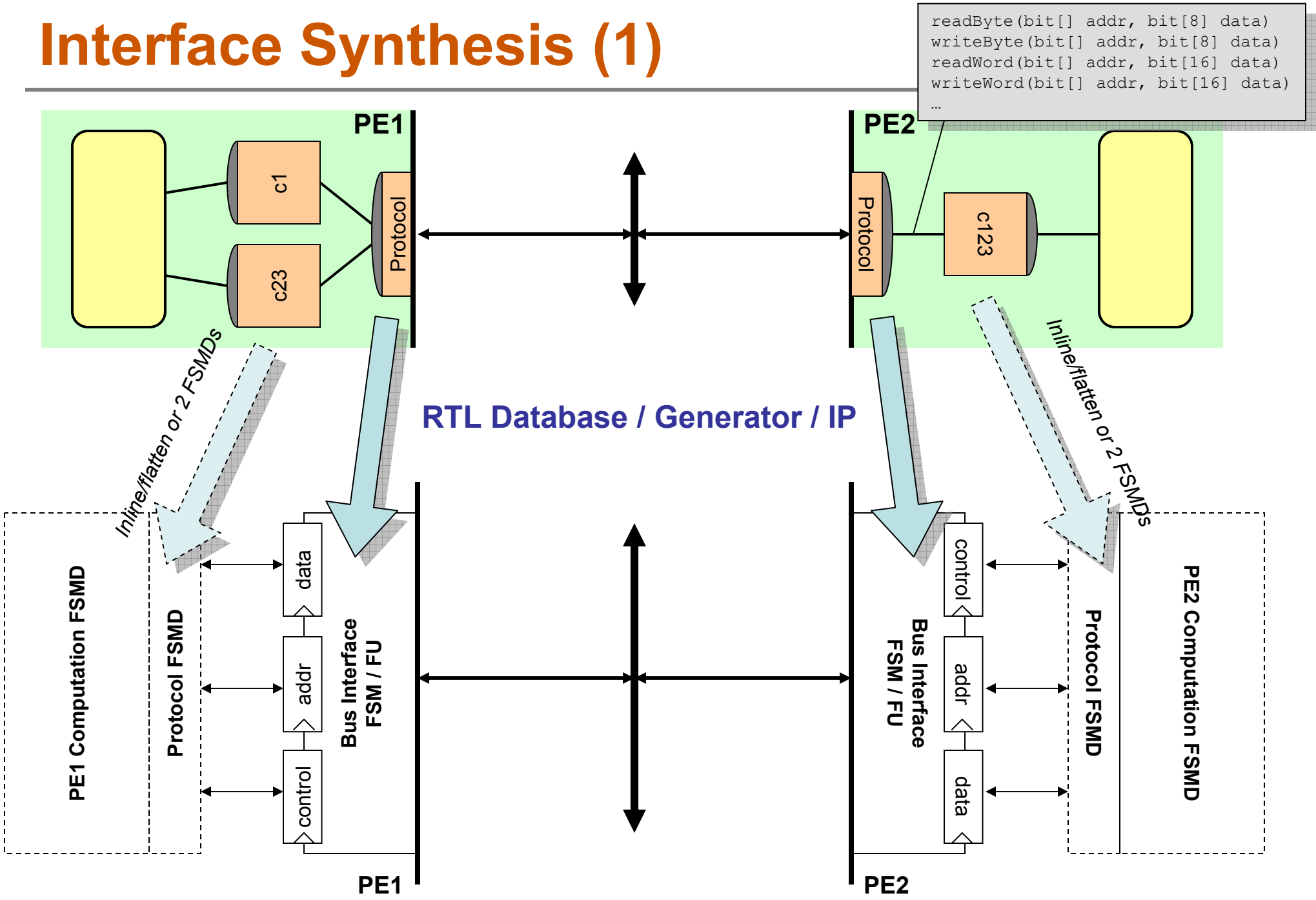


Protocol Stack Optimizations (2)

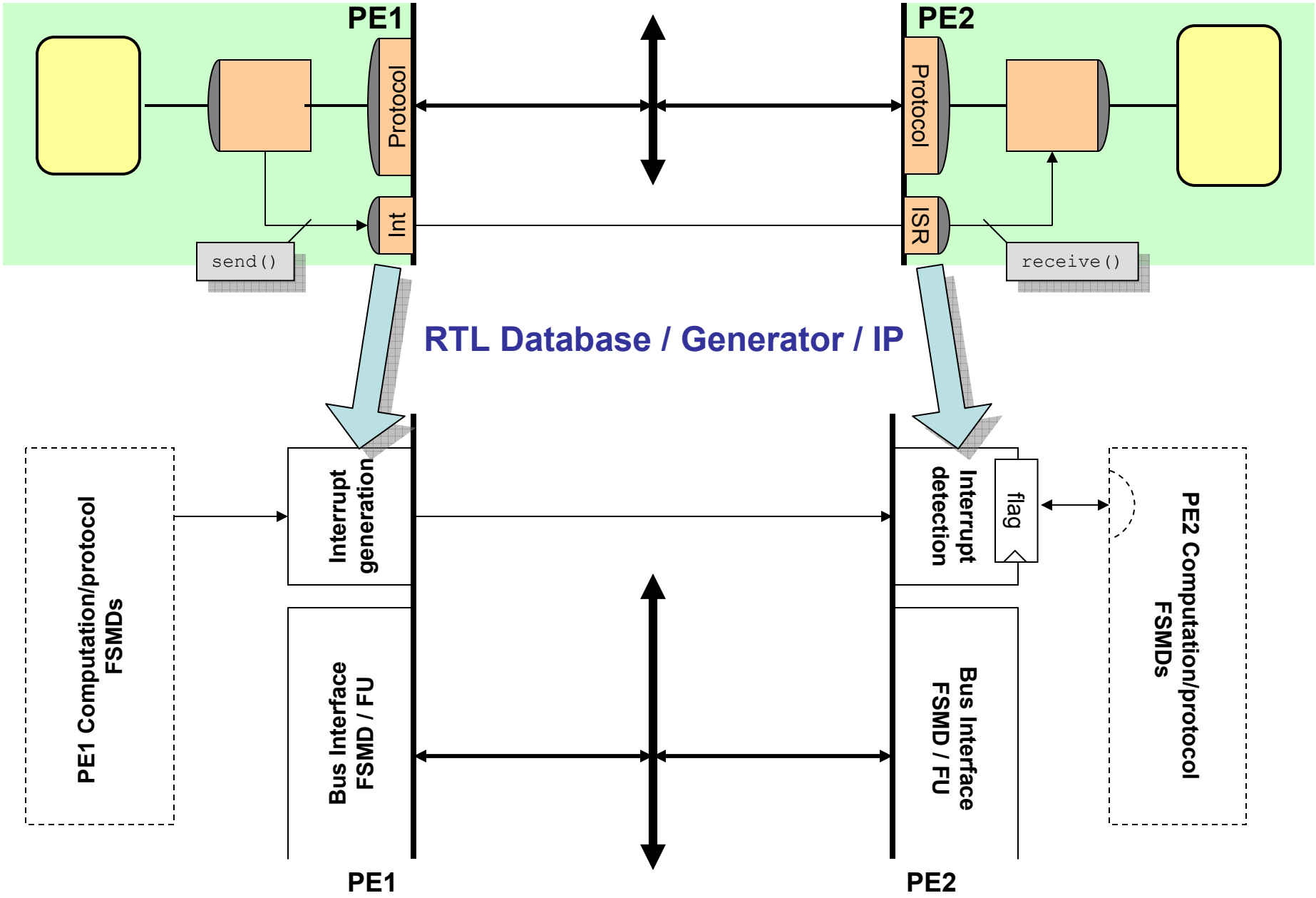
- **Apply automatic code optimizations during refinement**
  - Code optimized for HW/SW synthesis
  - Layer merging and cross-optimizations (inline/interleave)



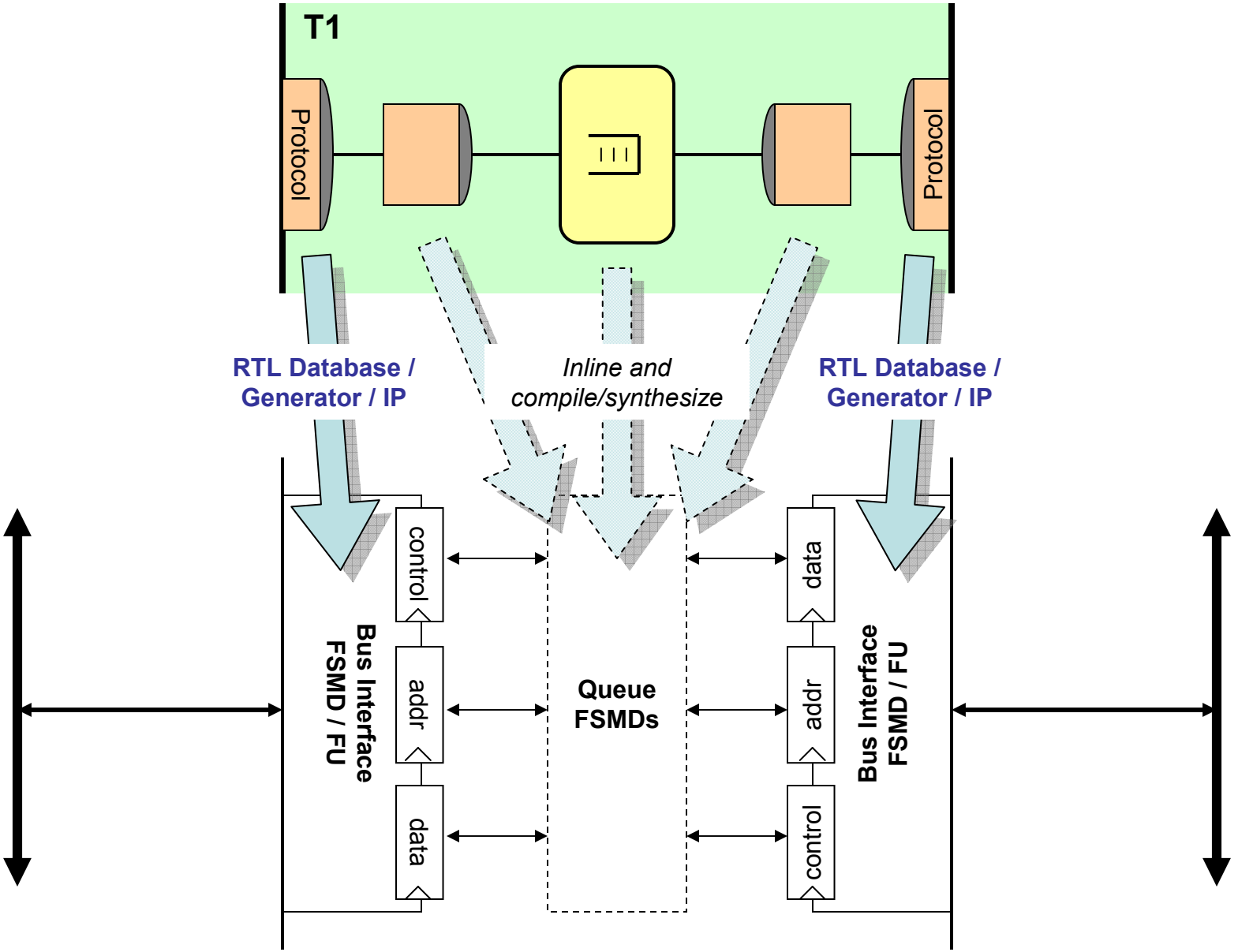
Interface Synthesis (1)



Interface Synthesis (2)



Transducer Synthesis



Lecture 11: Summary

- **Communication modeling & refinement**
  - Systematic, structured communication design flow
    - Layer-based modeling and refinement
    - Well-defined levels, models and design steps
    - Support for rich applications and wide variety of target architectures
  - Intermediate abstractions & models
    - Rapid, early feedback, validation and exploration
    - Accuracy vs. speed tradeoffs
- **Communication synthesis**
  - Generation of communication layer implementations
    - Application and target-architecture specific, customized and optimized
  - Protocol stack optimizations
    - Merging and cross-optimizations of layers
  - Interface backend synthesis