ECE 587 – Hardware/Software Co-Design
Lecture 11 Communication Modeling

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

February 19, 2025

Reading Assignment

- This lecture: 3.5
- Next lecture: 7.1–7.3

Outline

Communication Modeling

- Adopt well-established ISO/OSI 7-layer model
  - Layers are stacked on top of each other.
  - Each layer provides services to layers above by using services of the layer below.
- Layers are tailored to specific system design requirements.
  - E.g. to reflect the HW/SW partitioning of the communication functionality
- Use of layers facilities reasoning about communication stacks
  - However, it should not prevent implementations to merging functionalities across layers for various optimizations.
  - The whole communication stack should be treated as a single specification.

The Upper 5 ISO/OSI Layers

TABLE 3.2 Communication layers

Layer	Semantics	Functionality	Implementation	OSI
Application	Channels, variables	Computation	Application	7
Presentation	End-to-end typed messages	Data formatting	OS	6
Session	End-to-end untyped messages	Synchronization, multiplexing	OS	5
Transport	End-to-end data streams	Packeting, flow control	OS	4
Network	End-to-end data packets	Subnet bridging, routing	OS	3

(Gajski et al.)

The Lower 2 ISO/OSI Layers

TABLE 3.2 Communication layers

Layer	Semantics	Functionality	Implementation	OSI
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

(Gajski et al.)

Outline

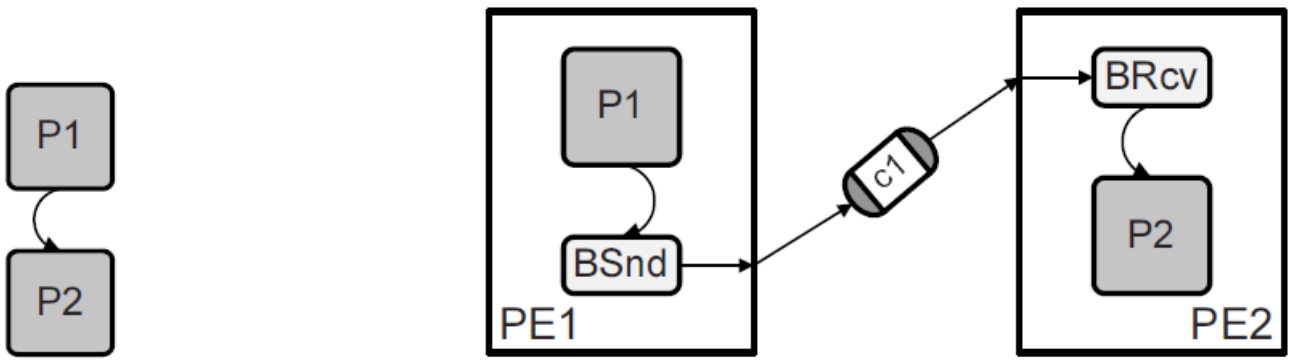
Application-Level Communication Primitives

- Application-level communication modeling should support a rich set of primitives.
  - For processes to communicate no matter they are on the same processor or not.
  - The primitives are expected to provide guaranteed delivery.
- Events for one-way synchronization, e.g. control flow transitions
- Shared variables
- Message-passing channels
  - Synchronous: both parties block
  - Asynchronous: receiver blocks
- Queues as a special case of asynchronous message-passing with well defined, fixed buffer sizes.
- Other complex and user-defined channels with extended semantics.
  - E.g. semaphores or mutexes.

Application Layer Communication Modeling

- Once we map processes to processors, communications among them should also be mapped.
- We are interested in communications between processes running on different processors here.
  - OS will utilize similar technique to model communications between processes running on a single processor.
- The application layer is responsible to translate the previous mentioned primitives to a restricted set of primitives supported by the following design process.

Making Synchronizations Explicit



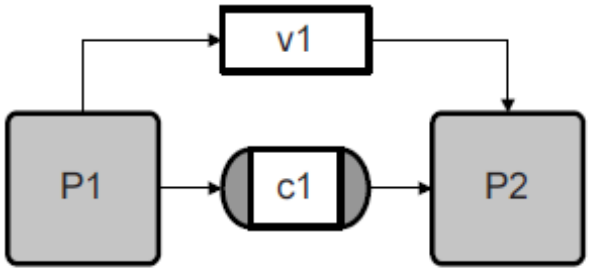
(a) Behavioral model (b) Synchronization

FIGURE 3.16 Application layer synchronization

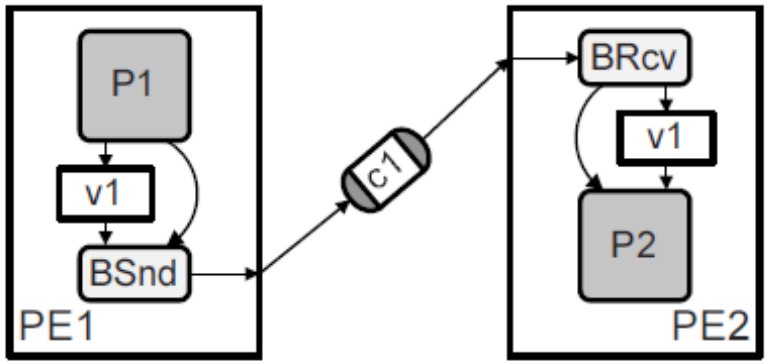
(Gajski et al.)

- There exists a dependency among P1 and P2 running on different processors.
- Explicit synchronization between the processes are necessary.
- Two processes and one channel are introduced.
  - No modification of the processes P1 and P2.

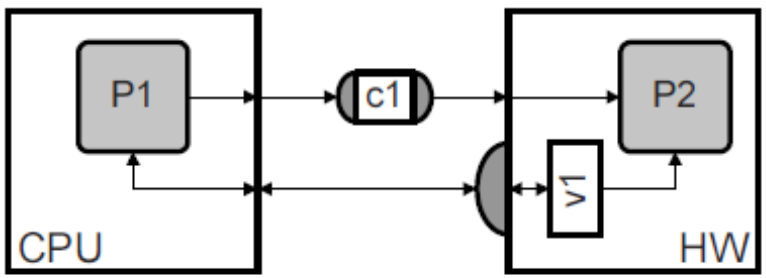
Resolving Memory Visits



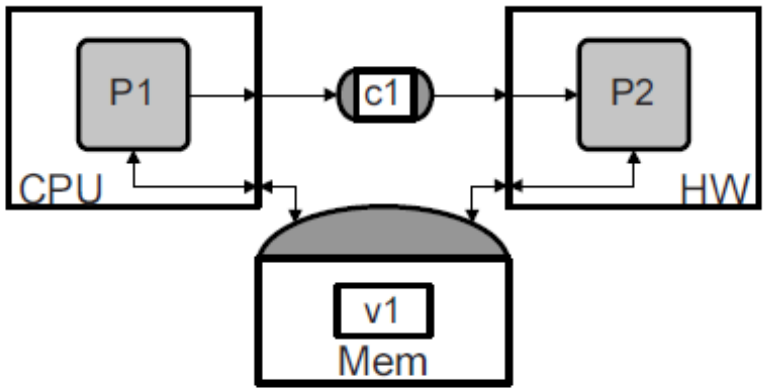
(a) Behavioral model



(b) Distributed message-passing



(c) Memory-mapped I/O

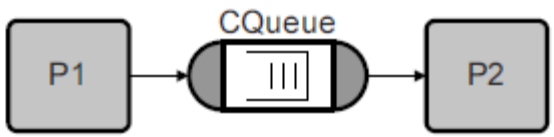


(d) Shared memory

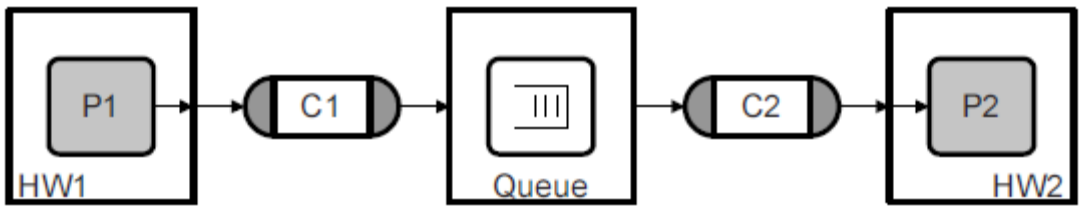
FIGURE 3.17 Application layer storage

(Gajski et al.)

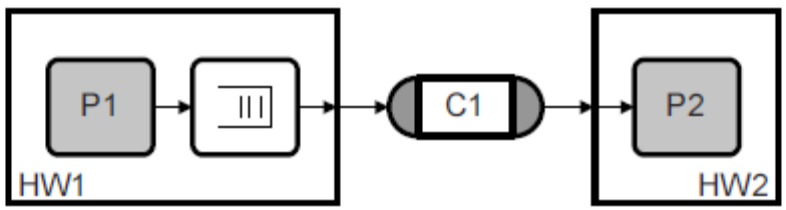
Incorporating Computation for Complex Channels



(a) Behavioral model



(b) Dedicated PE



(c) Local processes

FIGURE 3.18 Application layer channels

(Gajski et al.)

Outline

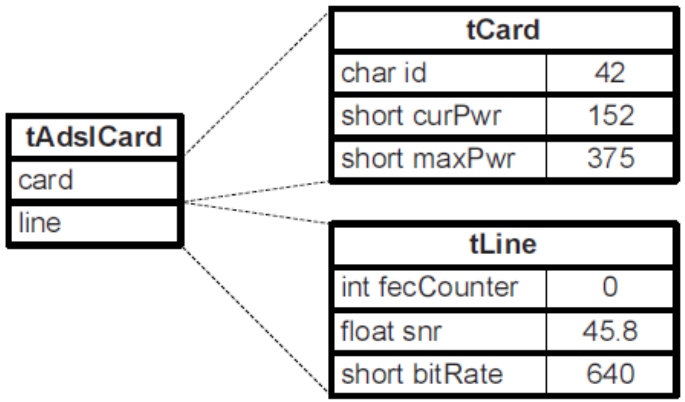
Presentation Layer

- Layout of data is processor dependent.
- Presentation layer is responsible for explaining the meaning of bytes.
  - Formatting of data
  - Conversion between data in abstract types to untyped blocks of bytes
- Layout parameters
  - Bitwidth of machine character, i.e. the smallest addressable unit
  - Size of primitive data types
  - Alignment of primitive data type
  - Endianess, i.e. ordering of characters in primitive data types
- Presentation layer adds the details to model the storage/performance overhead associated data formatting and conversion.

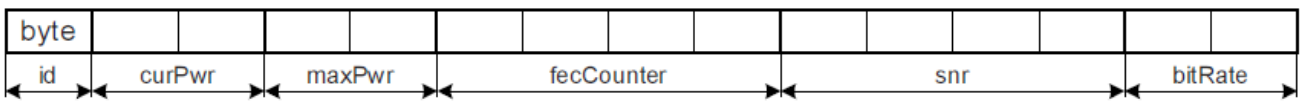
Presentation Layer Communication Modeling

- For each set of communication partners, a common data layout is chosen to exchange data.
  - The communication mechanism is based on message passing or shared memory.
- The chosen layout can be the same as in both, one or none of the involved processors.
  - Any difference in the layout parameters of network, memory, and processor would require conversions.
  - You may have an empty presentations layer if all parameters are the same.
- The layouts may either be globally defined or chosen differently.
  - Globally, one can save data traffic by choosing a layout without alignment requirement.
  - One can alternatively choose different layouts to match parameters of participating processors to reduce conversion overhead.

Presentation Layer Modeling Example



(a) Application data structure



(b) Network byte stream

FIGURE 3.19 Presentation layer

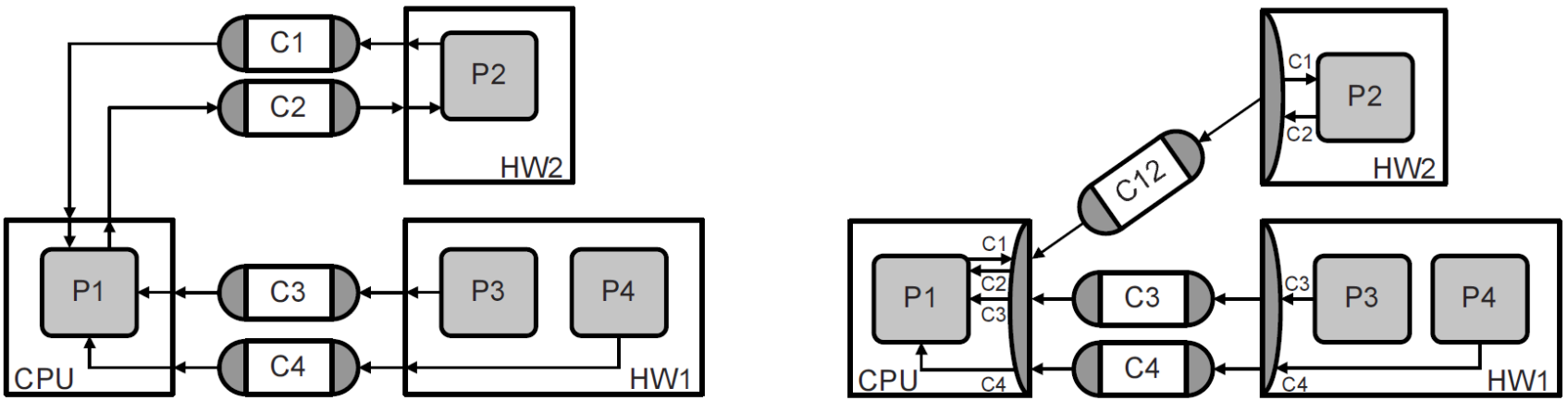
(Gajski et al.)

Outline

Session Layer

- For communications at application layer, as many channels as desired are introduced.
- Such approach make specification at application layer easier but is not realistic.
  - There are only a limited number of lower layer communication channels, i.e. end-to-end transports.
- Session layer is responsible for mapping application layer channels into those end-to-end transports.
  - Note that we don't need to worry about different data types passing through those channels as the presentation layer will convert them into untyped byte streams.
- Session layer is also reponsible for carrying information across application layer channels, e.g. for authentication and authorization in web applications.

Merging Channels Accessed Sequentially



(a) Application channels

(b) Network transports

FIGURE 3.20 Session layer

(Gajski et al.)

- If the channels (C1 and C2) lifetimes are not overlapped they can be merged directly.

Merging Channels Accessed Concurrently

- One need to define a new type of messages that are transfered over the end-to-end transport, consisting of
  - Channel ID: identify which application layer channel this message belongs to
  - Payload: the original message
- Computation/communication overheads
  - To send a message, session layer need to form the the message by adding the Channel ID.
  - When a message is received, session layer need to deliver the payload to its destination according to the Channel ID.
- It may be too costly to resolve the issue at this layer.
  - Addressing schemes introduced at lower layers will help and are usually a better solution.

Outline

Packeting

- So far we assume that the messages are of arbitrary size.
- In a networked communication system, efficient utilization of resources usually require to transfer messages of a fixed size, i.e. packets.
  - Simplify buffer/queue designs
  - Avoid long messages to block short ones
  - Reduce overhead associated with sending many short messages.
- Transport layer leverage packeting to solve certain existing issues and need to solve issues brought by packeting.

Transport Layer Modeling

- Synchronization: when necessary, acknowledgements for packet arrivals are provided.
- Guaranteed delivery: lost packets can be identified via missing acknowledgements and then re-transmitted.
- Packets are reordered if lower layers may deliver them out of order.
- Flow control: possible congestions are identified and outgoing traffic is reduced if necessary.
- Most above tasks require packets to carry more information than chunks of original messages.
  - packet = header+payload
  - Additional multiplexing information can be added to header to allow multiple end-to-end transports to share the same lower layer network route.

Outline

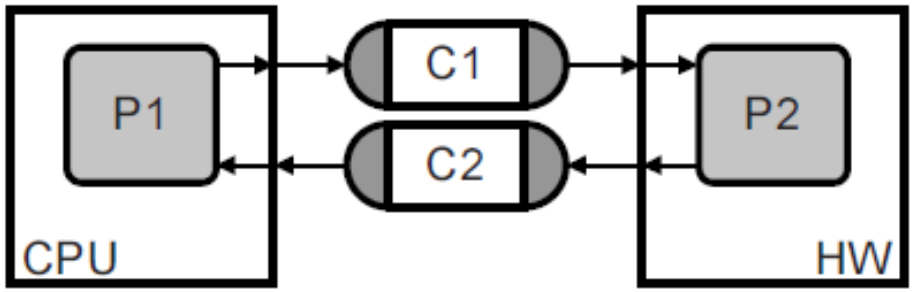
Addressing and Routing

- We do assume end-to-end transports exist when necessary.
  - However, when there are many end points in a system, it is not feasible to have a physical link between each communicating pair.
- Possible solutions
  - Share a physical link among multiple end points
  - Utilize additional communication elements (CEs), e.g. routers, to relay communications
- The network layer focuses on the second solution.
  - Addressing: provide means to identify end points
  - Routing: choose immediate destinations when relaying packets, i.e. to choose from the end points that have a physical link to the current one.

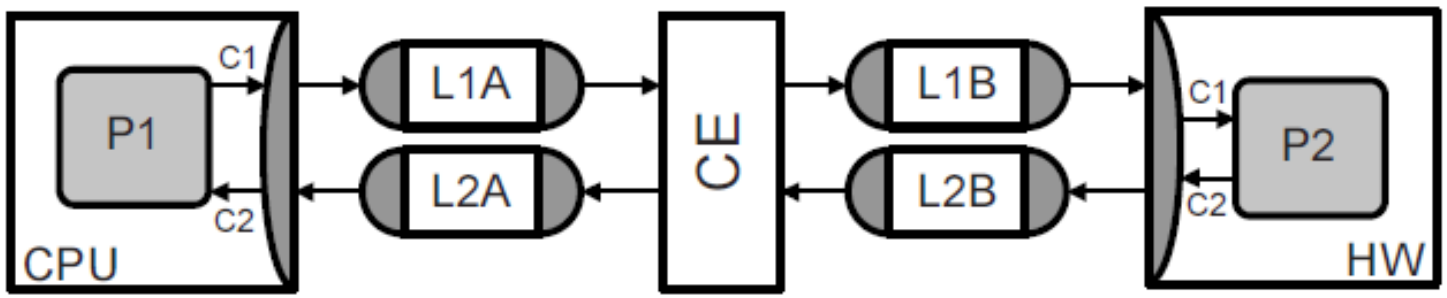
Network Layer Modeling

- Form networks by choosing addressing and routing schemes
  - Communication elements are introduced when necessary.
- Interconnect different networks
  - They may be based on different addressing and routing schemes due to different trade-offs.
- Allow to model the impacts of network structure.

Relaying Communication by CEs



(a) End-to-end transports



(b) Point-to-point links

FIGURE 3.21 Network layer

(Gajski et al.)

Outline

Links and Stations

- Logical links for packet transfers
  - Point to point between two stations.
  - On top of a direct physical communication media, i.e. the stations are directly connected
- Roles of stations
  - Sender/receiver
  - Master/slave
  - Senders are not necessary masters.

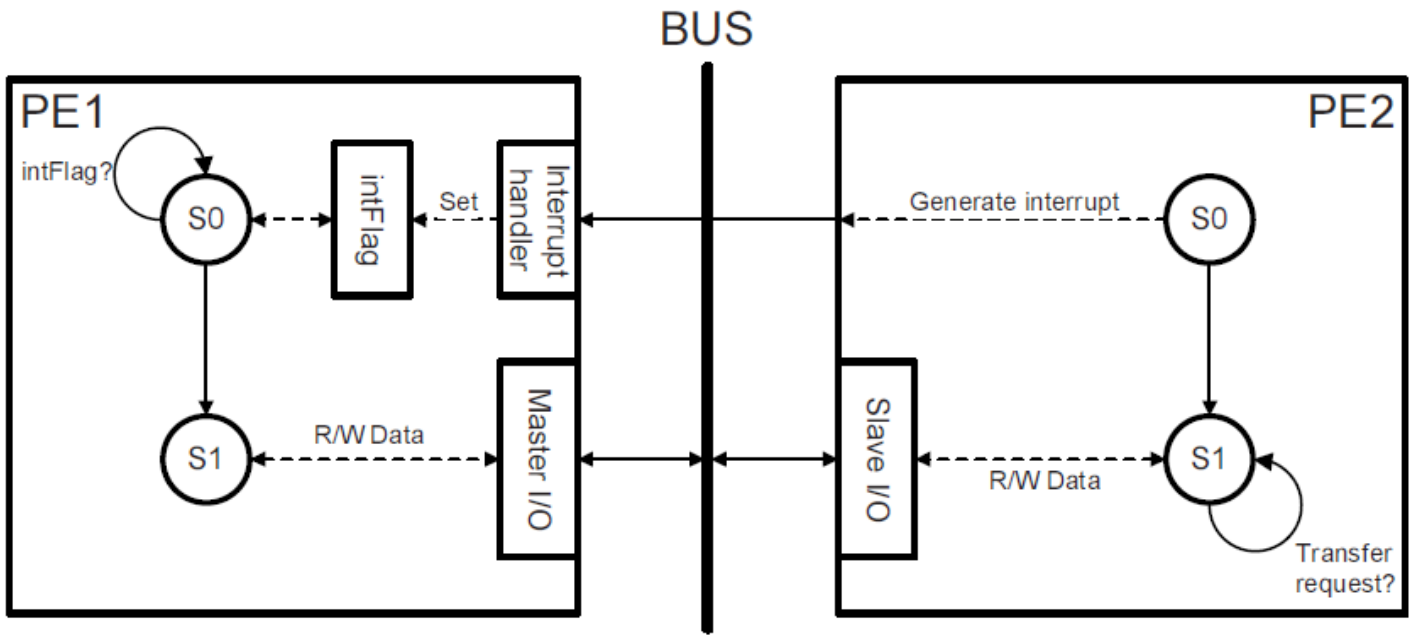
Master/Slave Behavior

FIGURE 3.23 Link layer

(Gajski et al.)

- Master must wait for slave to be ready before initiating a transfer, except:
  - The slave is always ready, e.g. a memory-mapped I/O.
  - With lower-level protocols supporting full two-way synchronizations
- Allow to model the impacts of congestion in great detail.

Synchronization for Fixed Master/Slave Assignments I



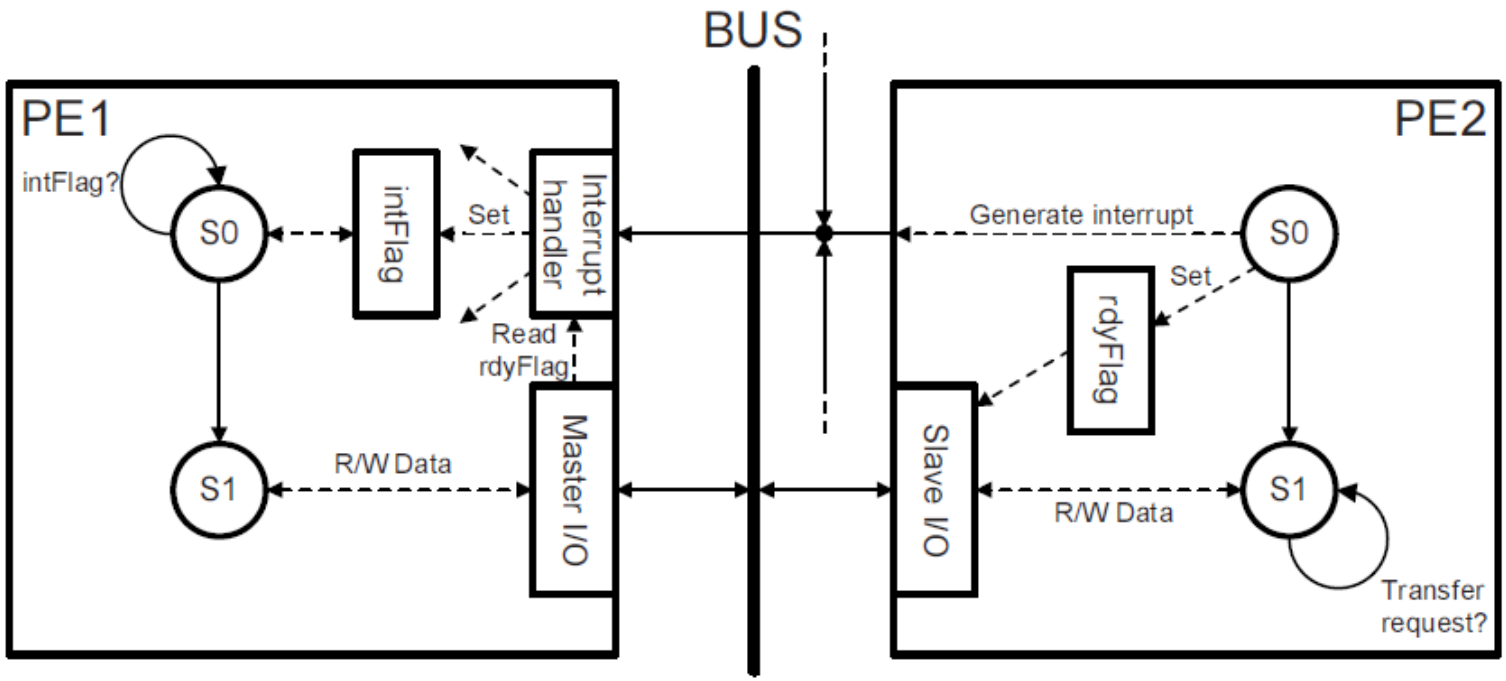
(a) Dedicated interrupts

FIGURE 3.24 Link layer synchronization

(Gajski et al.)

- Use a separate, out-of-band connection
  - E.g. a dedicated interrupt

Synchronization for Fixed Master/Slave Assignments II



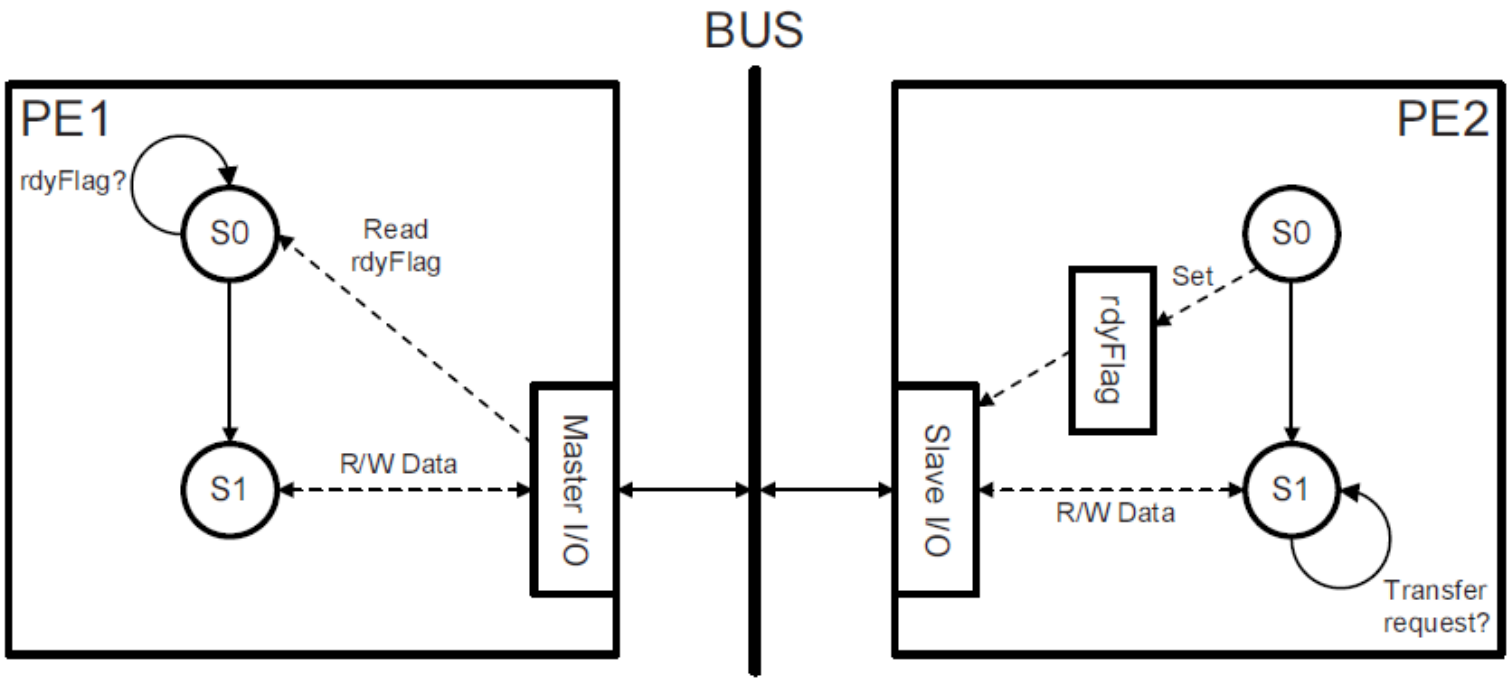
(b) Shared interrupts

FIGURE 3.24 Link layer synchronization

(Gajski et al.)

- Shared interrupts can also be used for synchronization.
  - Upon receiving an interrupt, the ISR will query slaves through Master I/O for their rdyFlags.

Synchronization for Fixed Master/Slave Assignments III



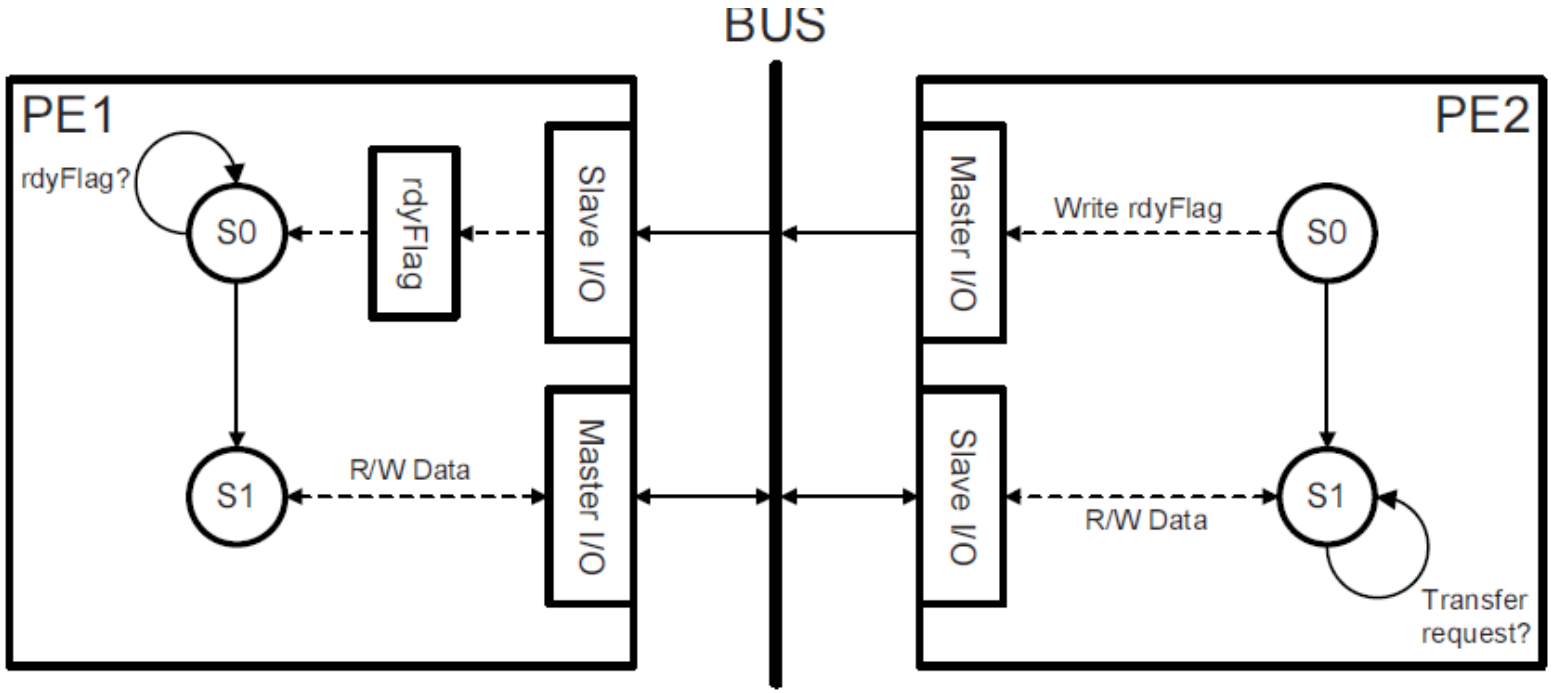
(c) Slave polling

FIGURE 3.24 Link layer synchronization

(Gajski et al.)

- If there is no out-of-band connection, the master has to poll the rdyFlag in the slave through Master I/O.
  - Incur computation overhead for the master
  - Incur communication overhead for the bus

Synchronization with Both Master/Slave Interfaces



(d) Flag in master

FIGURE 3.24 Link layer synchronization

(Gajski et al.)

- Each station owns both interfaces.
- To initialize a communication, the master (PE1) will utilize its variable rdyFlag that can be changed by the slave (PE2).

Stream (Sub-)Layer

- Multiplex data streams on different logical links over an underlying shared medium for different protocols.
- Addresses are used to distinguish and distribute different data streams.
- One can assign a unique physical bus address to each packet stream going over a shared medium/bus.
- Multiple addresses should be used if there are multiple concurrent and overlapping streams going in and out.
- Additional IDs are introduced when available addresses are not sufficient.

Media Access (Sub-)Layer (MAC)

- Provide an immediate abstraction of the shared underlying medium
  - Underlying bus is modeled as a single, shared medium where blocks of bytes are transferred.
  - The MAC layer slices data into transactions supported by the underlying bus protocol and assembles those transactions back into data.
- Use available protocol transactions effectively and optimally
- Provide access control and locking to manage/resolve conflicting accesses to the shared medium.
  - Centralized arbitration: via arbiter components, modeled as part of the protocol
  - Distributed arbitration: via bus protocols
- The MAC layer separates and translates between target-dependent aspects in lower layers and application-specific code in higher layers.

Outline

Protocol Layer

- Provide services to transfer groups of bits over the actual bus
  - Support all transaction types of the bus
  - Different transactions can vary in the size of words or frames being transferred.
- Implementations of the protocol layer includes the state machines to perform access control, synchronization and data transfers.

Protocol Layer Example

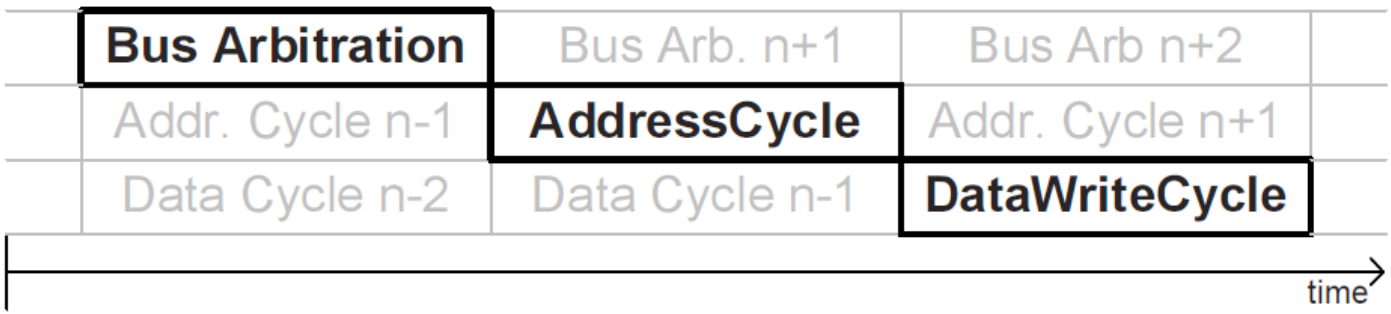


FIGURE 3.26 Protocol layer

(Gajski et al.)

Physical Layer

- Provide transmission of raw bits over a physical communication channel
- Introduce the pins and wires of a bus
- Define the corresponding voltages and bit timing