ECE 587 – Hardware/Software Co-Design
Lecture 15 Verification I

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

March 2, 2026

Reading Assignment

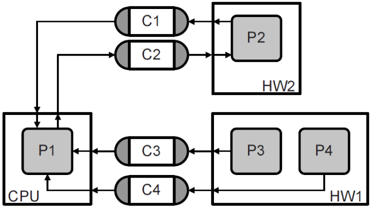
- This lecture: 3.5, 7.1
- Next lecture: 7.2, 7.3

Outline

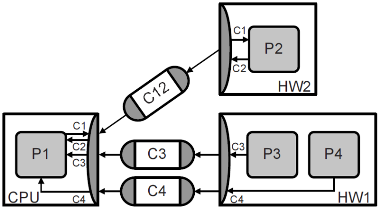
Session Layer

- 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.

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.

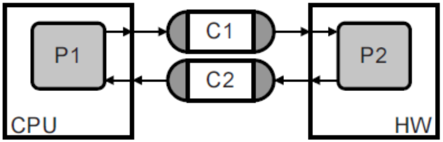
Addressing and Routing

- 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.

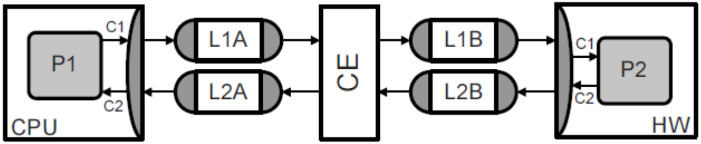
Network Layer Modeling

- 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

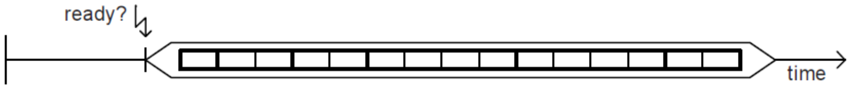
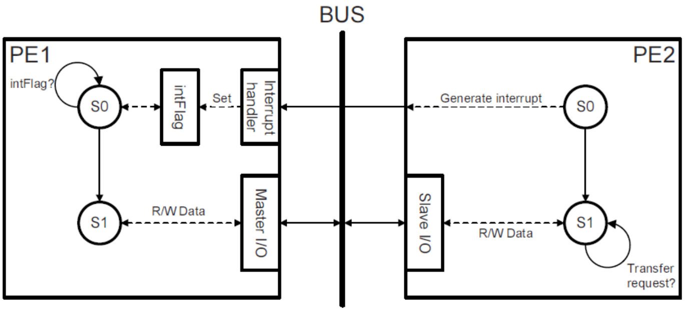


FIGURE 3.23 Link layer

(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



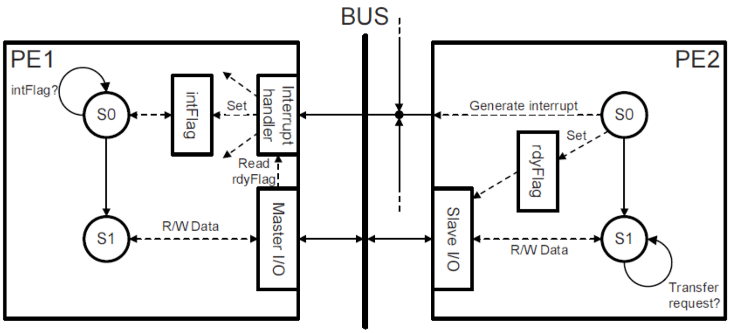
(a) Dedicated interrupts

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

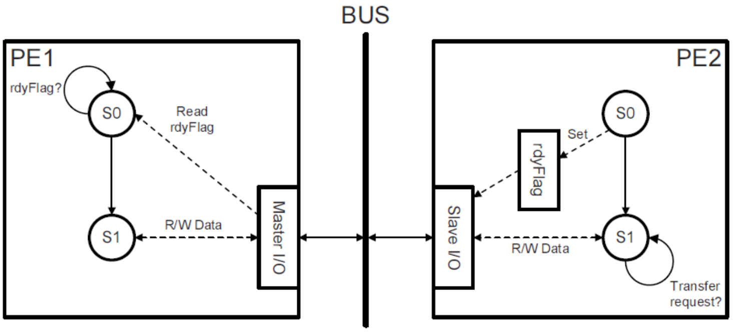


(b) Shared interrupts
FIGURE 3.24 Link layer synchronization

(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



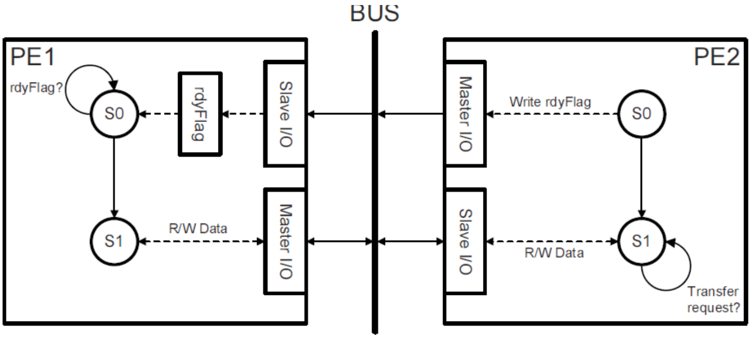
(c) Slave polling

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.

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

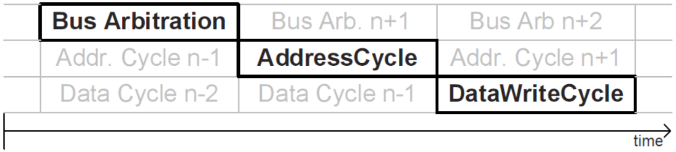


FIGURE 3.26 Protocol layer

(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

Outline

Functional Verification

- Analysis and reasoning on a computer model of the system
  - Before manufacturing
- Establish confidence in functional correctness before the product is shipped
  - Critical for systems where safety is the first concern
  - Prevent costly recall for non-critical systems as well
- Designers need to make sure the model at each step of design does reflect the original intent.
  - Catch bugs as early as they are introduced so one can locate them effectively

Simulation Based Verification vs. Formal Verification

- Both verification methods need a golden reference as part of the specification.
  - Simulation based methods: stimuli (inputs) and monitors (expected outputs)
  - Formal methods: mathematical model of desired properties
- Simulation-based methods are effective to catch bugs at early design stages, though only as good as the stimuli.
- Formal methods can provide a proof of correctness, but require more effort at design time.
- Simulation is still predominant while formal methods are catching up.
  - More formal verification tools become available.
  - More designers are trained to use these tools.
  - e.g. most gate-level verifications are based on formal methods nowadays

Outline

Simulation Based Methods

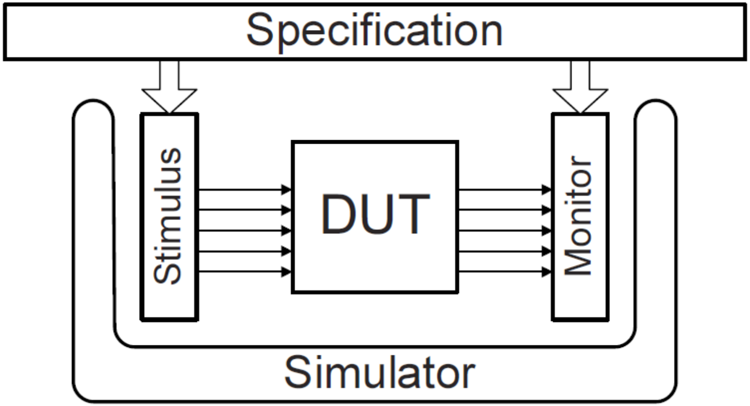


FIGURE 7.1 A typical simulation environment

(Gajski et al.)

Test-Bench

- A test-case consists of a stimulus and the corresponding monitor.
  - Monitors are usually generated from stimuli using a higher level model (golden reference) that is more likely to be correct.
- A collection of test-cases forms the test-bench.
  - To achieve maximum productivity, each test-case should uncover some bug that has not been uncovered by a previous one – waste of time to test part of DUT (device under test) that have already been tested.
- How to measure the part being tested under a test-case?

Coverage

- Could be based on the percentage of DUT that has been checked
  - e.g. in terms of # lines of source code
  - or # states in a state diagram modeling the system
- However, that's different from the entire behavior of the design.
  - A single statement may involve many variables and a test-case covers it may miss some important corners.
- All corner cases may be modeled explicitly in the state diagram for coverage.
  - But then the size of the diagram will become a big concern.

Performance Considerations

- Coverage may increase as you simulate with more test-cases.
  - However, simulation takes time.
  - Need to trade-off verification performance with quality
- Stimulus optimization: simulate less cases
  - Use coverage feedback mechanism to improve test-cases that are otherwise generated randomly
  - Not all test-cases are valid – only simulate with valid ones
- Monitor optimization: discover bugs faster
  - White box testing – also monitor internal variables
  - Use assertions instead of golden for internal variables
  - Communicate more effectively with designers via visualization
- Speed-up techniques: faster simulation
  - Faster algorithms
  - Hardware assistance, e.g. on FPGA
  - Simulate at higher abstraction levels, if certain details can be omitted, e.g. to use an instruction set simulator instead of a cycle-accurate simulator.

Coverage Feedback I

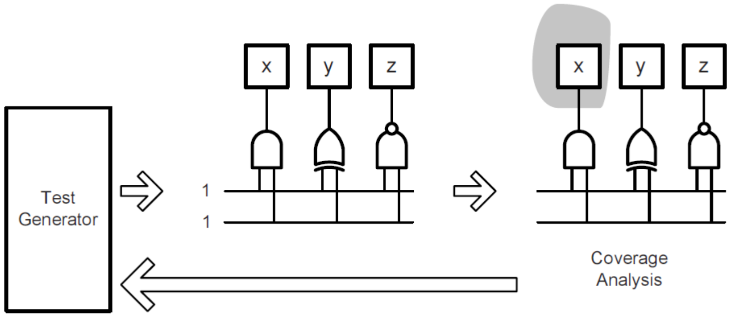


FIGURE 7.2 A test case that covers only part of the design.

(Gajski et al.)

- The logic gates could be a simplification of branch conditions.
- It could be impossible to generate a single test-case to cover all branches.

Coverage Feedback II

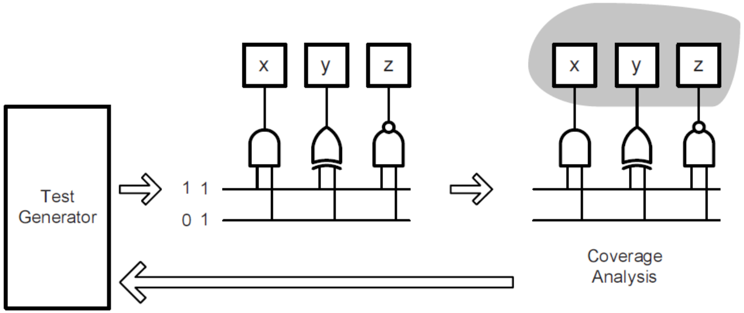


FIGURE 7.3 Coverage analysis results in a more useful test case.

(Gajski et al.)

- The additional test-cases can be found manually or automatically (by formal methods).

Hardware Emulation

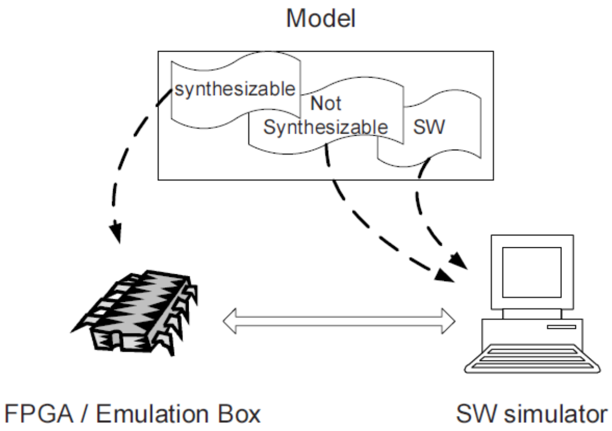


FIGURE 7.5 A typical emulation setup.

(Gajski et al.)

- Speed-up simulations that cannot be speeded-up otherwise.
  - Usually when cycle-accurate behavior has to be simulated