

EE382V: Embedded System Design and Modeling

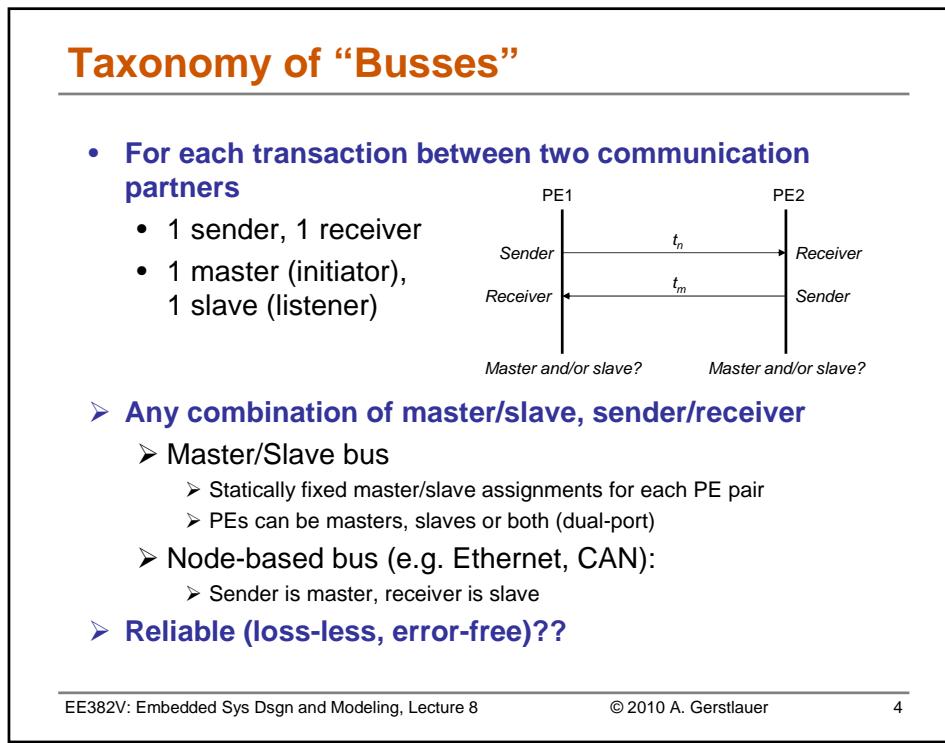
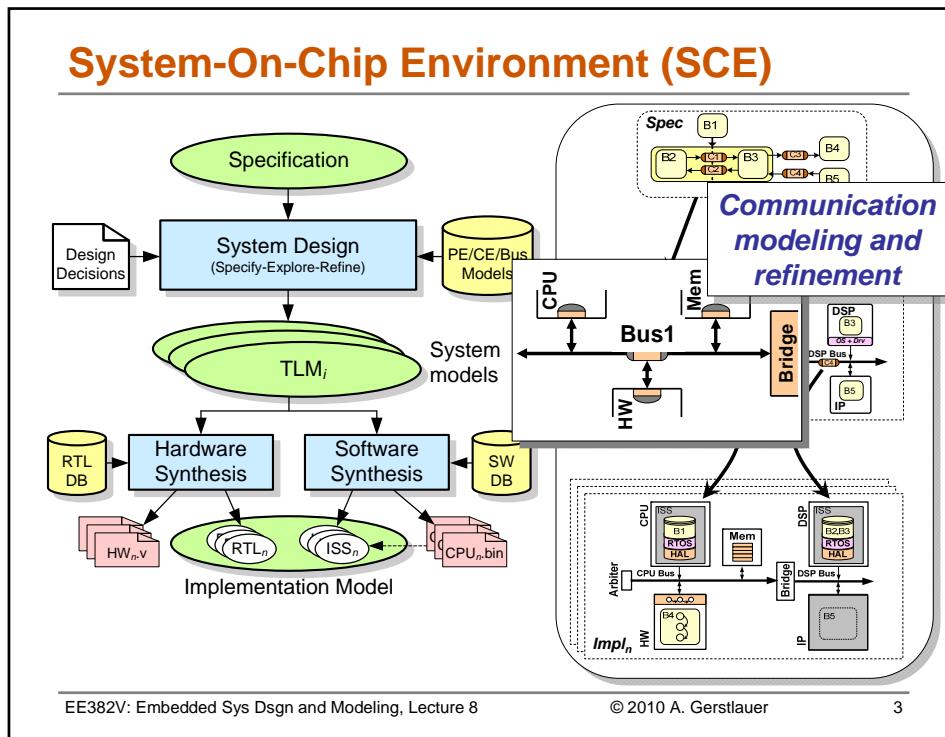
Lecture 8 – Communication Modeling & Refinement

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



Lecture 8: Outline

- **Communication layers**
 - Application
 - Network: presentation, session, transport
 - Communication: link, stream, media access
 - Protocol, physical
- **Communication synthesis**
 - Protocol stack generation
 - Interface backend synthesis
- **System models**
 - From specification to TLM to Cycle-Accurate



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 !

Lecture 8: Outline

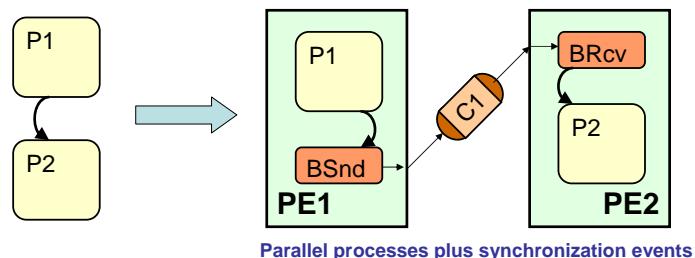
- Communication layers
 - Application
 - Network: presentation, session, transport
 - Communication: link, stream, media access
 - Protocol, physical
- Communication synthesis
 - Protocol stack generation
 - Interface backend synthesis
- System models
 - From specification to TLM to Cycle-Accurate

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)

Application Layer (1)

- **Synchronization**
 - Synthesize control flow



- Implement sequential transitions across parallel components

Application Layer (2)

- Storage
 - Shared variable mapping to memories

The diagram shows three different ways to map shared variables to local memories:

- Distributed:** Two processes (P1 and P2) running on separate Processing Elements (PE1 and PE2) both have local copies of variable v1. They communicate via channel C1. PE1 has a local copy of v1 and a block labeled "BSnd". PE2 has a local copy of v1 and a block labeled "BRcv".
- Memory-mapped I/O:** A process P1 running on a CPU has a local copy of v1. It communicates via channel C1 with a hardware component (HW) which also contains a copy of v1. The HW has a local copy of v1 and a block labeled "V".
- Shared memory:** A process P1 running on a CPU has a local copy of v1. It communicates via channel C1 with a shared memory block (Mem) which contains a copy of v1. The Mem has a local copy of v1 and a block labeled "v1".

➤ Map global storage to local memories

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 9

Application Layer (3)

- Channels
 - Complex channel synthesis

The diagram shows three different channel synthesis mechanisms:

- Dedicated hardware:** Two hardware components (HW1 and HW2) are connected by a queue (Queue). HW1 contains process P1 and HW2 contains process P2. They communicate via channels C1 and C2.
- Client-server implementation:** A process P1 and a process P2 are connected by a queue (CQueue). They communicate via channel C1.
- Additional process:** Two hardware components (HW1 and HW2) are connected by a queue (Queue). HW1 contains process P1 and HW2 contains process P2. They communicate via channel C1.

➤ Client-server implementation

➤ Server process

➤ Remote procedure call (RPC) channels

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 10

Presentation Layer

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

The diagram shows the mapping of abstract message structures to memory byte layout. On the left, two message structures are shown: **tCard** and **tLine**. **tCard** contains fields: char id (42), short curPwr (152), and short maxPwr (375). **tLine** contains fields: int fecCounter (0), float snr (45.8), and short bitRate (640). An arrow points from these structures to a memory byte layout table below. The byte layout table has columns: id, curPwr, maxPwr, fecCounter, snr, and bitRate. A double-headed arrow between 'bytes' and the table indicates the mapping.

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

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 11

Session Layer

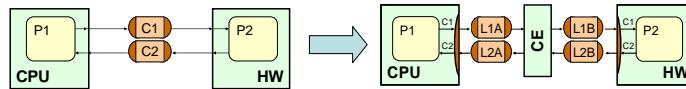
- Channel merging**
 - Merge application channels into a set of untyped end-to-end message streams
 - Unconditionally merge sequential channels
 - Merge concurrent channels with additional session ID (message header)
 - Channel selection over end-to-end transports

The diagram illustrates channel merging and session selection. It shows two hosts, HW1 and HW2, connected to a CPU. HW1 contains processes P1, P2, P3, and P4, and has four channels C1, C2, C3, and C4. HW2 contains process P2 and has channels C1 and C2. In the merged state, the channels are reorganized. C1 and C2 from HW2 are merged into a single channel C12. C1, C2, C3, and C4 from HW1 remain separate. The merged channel C12 is then selected by the CPU to communicate with the merged HW2 host.

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 12

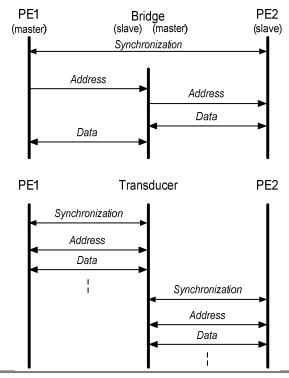
Network Layer

- **Split network into subnets**
 - Routing of end-to-end paths over point-to-point links
 - Insert communication elements (CEs) to connect busses



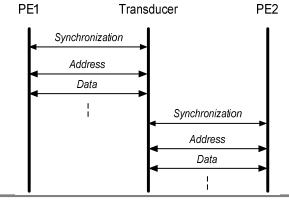
- **Bridges**

- 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



EE382V: Embedded Sys Dsgn and Modeling, Lecture 8

© 2010 A. Gerstlauer

13

Transport Layer

- **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
 - If 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/headers)
 2. Dynamic (runtime) routing

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8

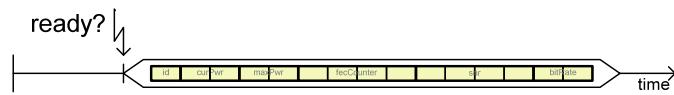
© 2010 A. Gerstlauer

14

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

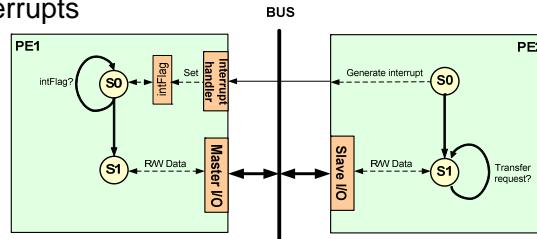


- Events from slave to master for master/slave busses
- Synchronization packets for node-based busses

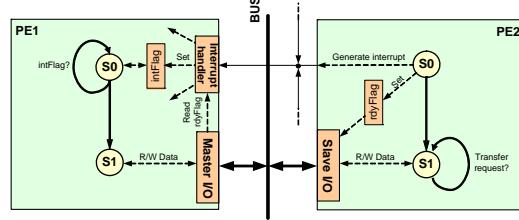
Link Layer (2)

- **Synchronization (2)**

- Dedicated interrupts



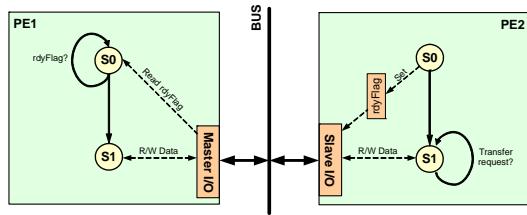
- Shared interrupts



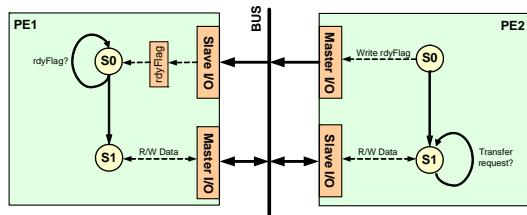
Link Layer (3)

- **Synchronization (3)**

➤ Slave polling



➤ Flag in master



Stream Layer

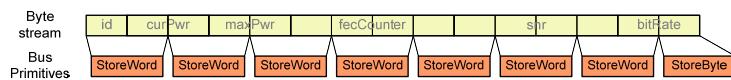
- **Addressing**

- 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

Media Access (MAC) Layer

- **Data slicing**

- Split data packets into multiple bus word/frame transactions



- Optimized data slicing utilizing supported bus modes (e.g. burst)

- **Arbitration**

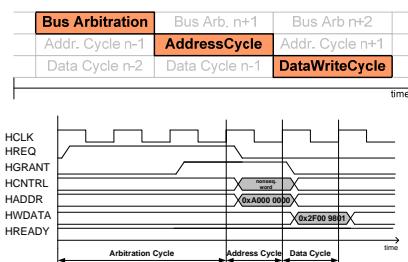
- Separate individual bus transactions in time
 1. Centralized using arbiters
 2. Distributed
- Insert arbiter components

Protocol, Physical Layers

- **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

Lecture 8: Outline

- ✓ **Communication layers**
 - ✓ Application
 - ✓ Network: presentation, session, transport
 - ✓ Communication: link, stream, media access
 - ✓ Protocol, physical
- **Communication synthesis**
 - Protocol stack generation
 - Interface backend synthesis
- **System models**
 - From specification to TLM to Cycle-Accurate

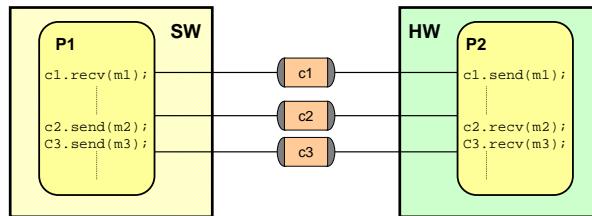
Source: A. Gerstlauer, D. Shin, J. Peng, R. Dömer, D. Gajski, "Automatic, Layer-based Generation of System-On-Chip Bus Communication Models," TCAD07

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8

© 2010 A. Gerstlauer

21

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

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8

© 2010 A. Gerstlauer

22

Network Model

The diagram illustrates a network model with two hosts, P1 and P2, connected through a network layer (T) and hardware (HW).

- P1:** Contains components c1, c2, and c3. It receives messages (c1.recv(m1);) and sends messages (c2.send(m2);, C3.send(m3);).
- HW:** Contains components c1, c2, and c3. It receives messages (c1.send(m1);) and sends messages (c2.recv(m2);, C3.recv(m3);).
- Network (T):** Represented by a green box containing nodes s0 and so, connected by a double-headed arrow.
- Link1 and Link2:** Represented by orange boxes connecting P1 to T and T to HW respectively.
- Data conversion:** A code snippet shows how data is converted from memory buffers to network packets:


```
send(type msg) {
    char buf[M];
    1: msg->buf;
    2: net.send(buf);
}
```
- Packeting, acknowledgement:** A code snippet shows the packeting process and acknowledgement handling:


```
send(void* msg, len) {
    for(pkt in msg):
        (S1) link.send(pkt);
        (S2) link.recv(ack);
    }
}
```
- 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**

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 23

Transaction-Level Model

The diagram illustrates a Transaction-Level Model with two hosts, CPU and HW, connected through a transaction layer (T) and TLM busses.

- CPU Host:** Contains components P1, SW, Net, MAC, and Int. It receives messages (c1.recv(m1);) and sends messages (c2.send(m2);, C3.send(m3);).
- HW Host:** Contains components P2, MAC, Link, Net, and C3. It receives messages (c1.send(m1);) and sends messages (c2.recv(m2);, C3.recv(m3);).
- TLM Bus1 and TLM Bus2:** Represented by orange boxes connecting the hosts to the transaction layer (T).
- Transaction Layer (T):** Represented by a green box containing nodes s0 and so, connected by a double-headed arrow.
- Link, stream:** A code snippet shows the process of sending data over a link and performing synchronization:


```
send(void* p, len) {
    (S1) wait(intr);
    (S2) mac.write(p, addr);
}
```
- Media access:** A code snippet shows the process of writing data to a bus:


```
intHandler() {
    notify(intr);
    write(void* d, len, a) {
        (S0) bus.writeWord(a, w);
    }
}
```
- Link layers:**
 - PEs + Memories + CEs + Busses
 - Bus bridges
 - Communication via bus transactions (bus TLM)
 - Address, data, arbitration
 - Synchronization (interrupts, polling)

➤ **System communication architecture**

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 24

Pin-Accurate Model (PAM)

Protocol, physical:

```

writeWord(addr, word) {
    HCLK
    HREQ
    HGRANT
    HCTRL
    HADDR
    HWDATA
    HREADY
}
    }
    
```

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

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 25

Protocol Stack Optimizations (1)

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

Presentation:

```

send(type msg) {
    char buf[M];
    1: msg->buf;
    2: net.send(buf);
}
    
```

Network:

```

send(d) {
    1: for (p in d) {
        link.send(d[p]);
    }
    2: link.recv(ack);
}
    
```

Link:

```

send(p) {
    1: wait(intr);
    2: mac.write(p);
}
    
```

MAC:

```

write(b) {
    1: for(w in b) {
        protocol.write(w);
    }
}
    
```

Data conversion

Packeting, acknowledgement

Synchronization

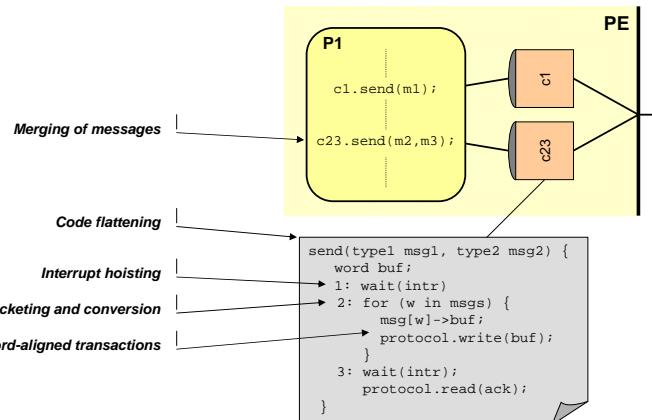
Data slicing

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 26

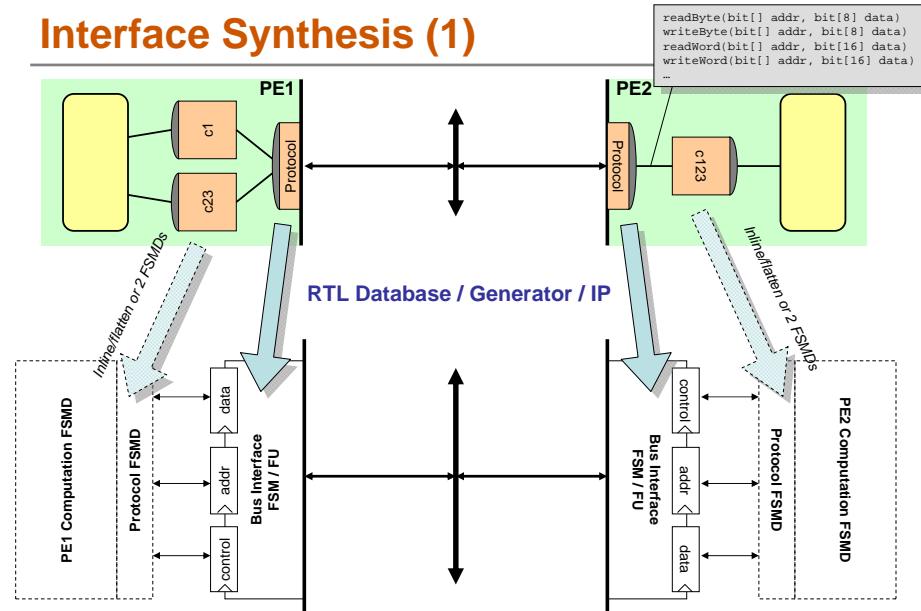
Protocol Stack Optimizations (2)

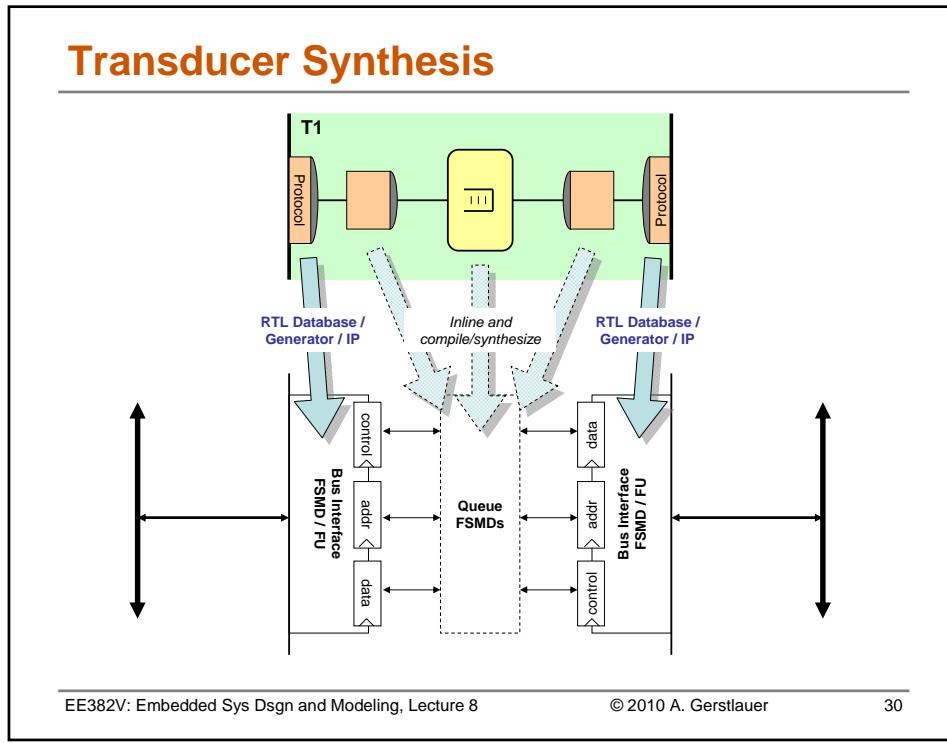
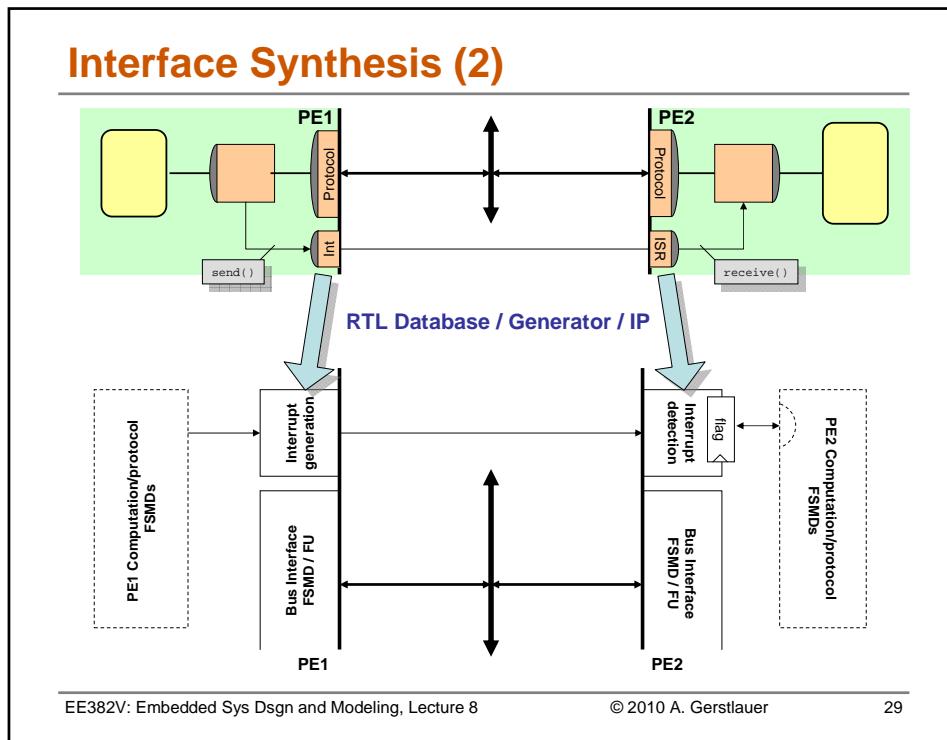
- Apply automatic code optimizations during refinement

- Code optimized for HW/SW synthesis
- Layer merging and cross-optimizations (inline/interleave)



Interface Synthesis (1)





Lecture 8: Outline

- ✓ **Communication layers**
 - ✓ Application
 - ✓ Network: presentation, session, transport
 - ✓ Communication: link, stream, media access
 - ✓ Protocol, physical
- ✓ **Communication synthesis**
 - ✓ Protocol stack generation
 - ✓ Interface backend synthesis
- **System models**
 - From specification to TLM to Cycle-Accurate

System Design Flow

- **Abstraction based on level of detail & granularity**
 - Computation and communication
 - **System design flow**
 - Path from model A to model F
-
- A. System specification model
 B. Timed functional model
 C. Transaction-level model (TLM)
 D. Bus cycle-accurate model (BCAM)
 E. Computation cycle-accurate model (CCAM)
 F. Cycle-accurate model (CAM)

Source: L. Cai, D. Gajski. "Transaction level modeling: An overview", ISSS 2003

- **Design methodology and modeling flow**
 - Set of models and transformations between models

System Models

- From layers to system models...

The diagram illustrates a Cycle Accurate Model (CAM) architecture. It is organized into several layers:

- Specification Model:** Contains the **Transaction Level Models** (TLM).
- OS:** Contains layers 7 (Application), 6 (Presentation), 5 (Session), 4 (Transport), 3 (Network), 2b (Link + Stream), and 2a (Media Access).
- HAL:** Contains layers 2a (Media Access) and 2a (Protocol).
- HW:** Contains layer 1 (Physical).

Communication between layers is managed by two components:

- MP (Message Passing):** Handles communication between adjacent layers.
- TLM (Transaction Level Model):** Handles communication between the Specification Model and the OS/HAL/HW layers.

Below the layers, a CAM (Cycle Accurate Model) interface is shown, consisting of Address, Data, Control, and CAM fields.

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 33

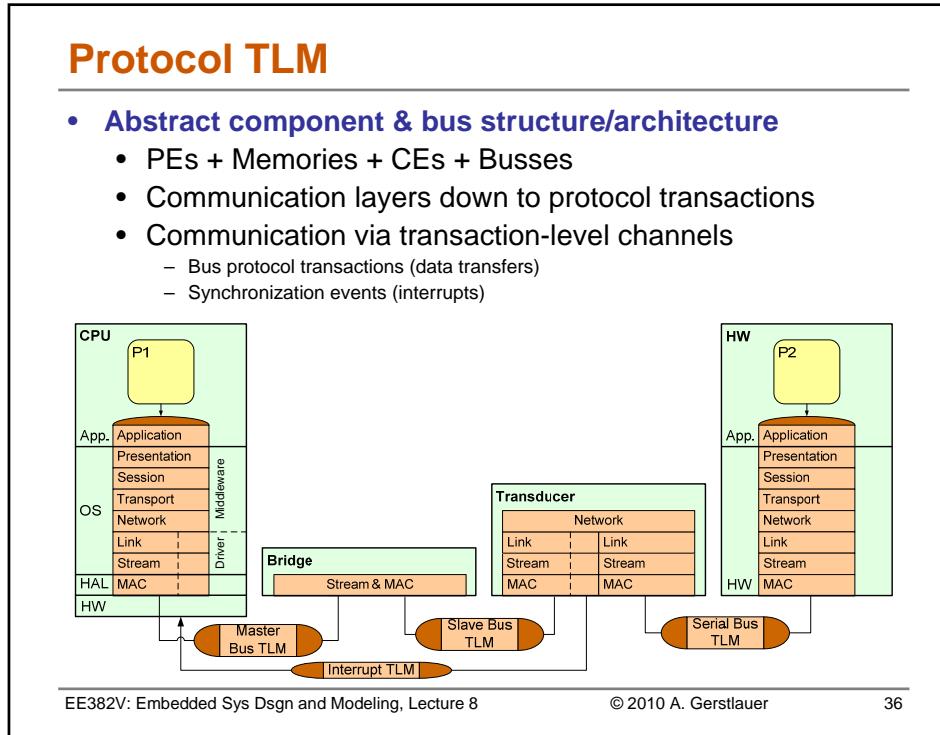
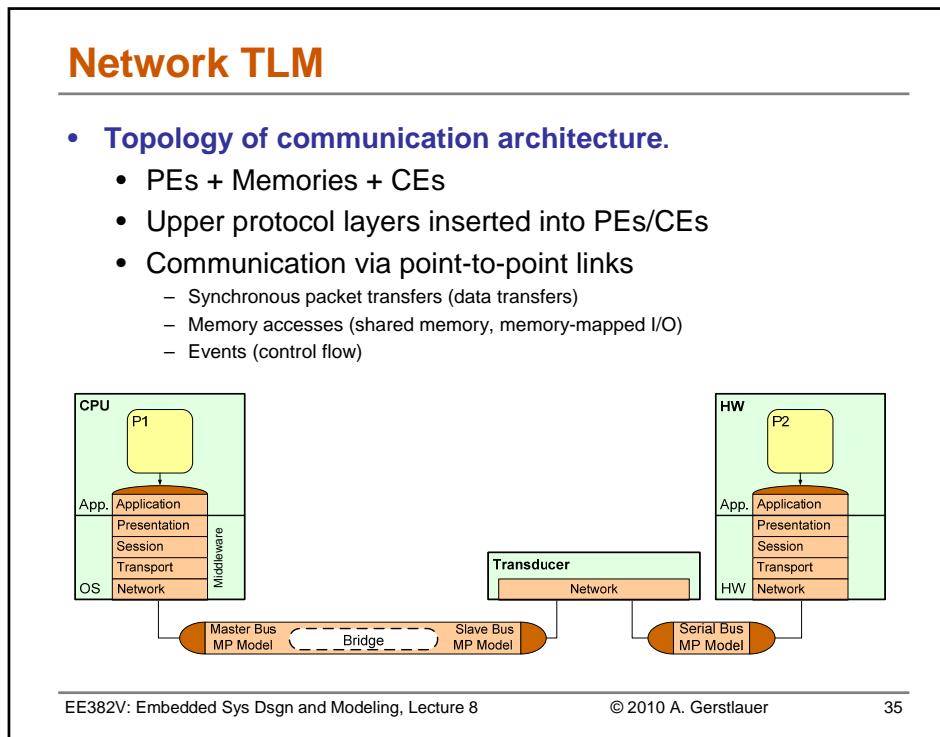
Specification Model

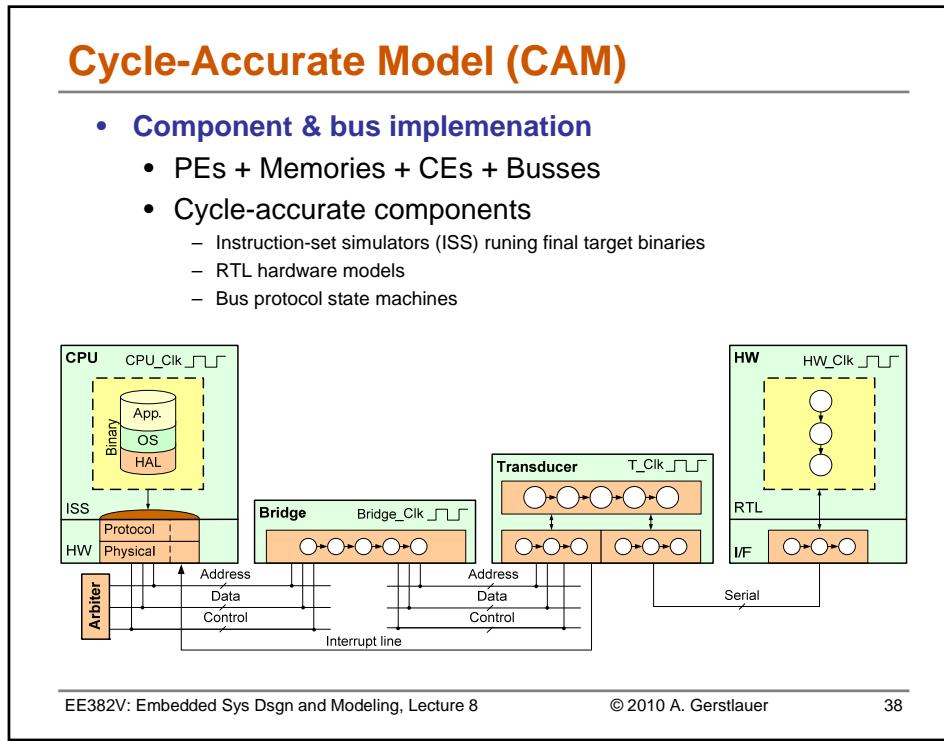
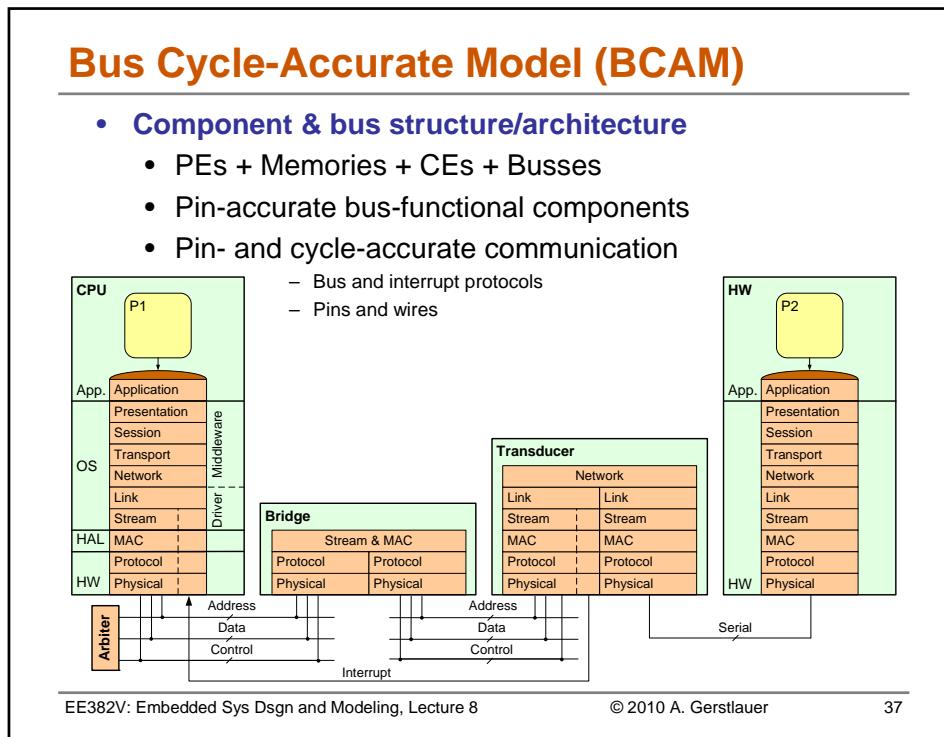
- Abstract, high-level system functionality

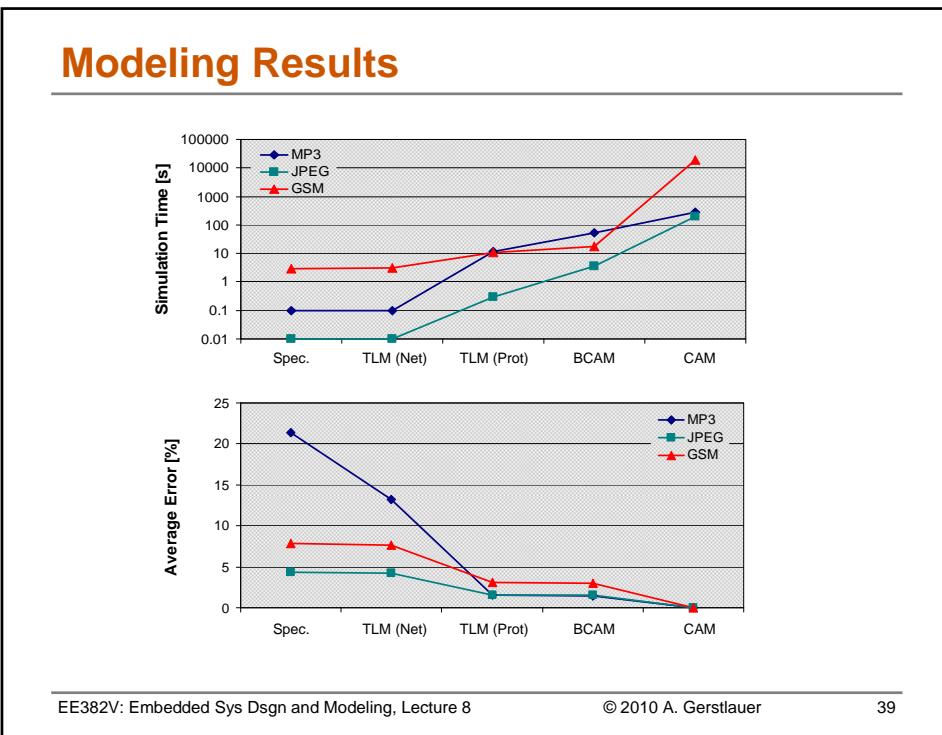
- Computation
 - Processes
 - Variables
- Communication
 - Sync./async. message-passing
 - Memory interfaces
 - Events

The Specification Model is represented as a network of processes and communication channels. Two processes, P1 and P2, are connected to an **Abstract MP Channel**. This channel is depicted as a horizontal oval with an orange center and brown end caps, representing a message-passing interface.

EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 34







- ## Lecture 12: Summary
- **Modeling of system computation and communication**
 - From specification
 - System behavior, Models of Computation (MoCs)
 - To implementation
 - Layers of implementation detail
 - Flow of well-defined models as basis for automated design process
 - **Various level of abstraction, accuracy and speed**
 - Functional specification
 - Native speeds but inaccurate
 - Traditional cycle-accurate model (CAM)
 - 100% accurate but slow
 - Transaction-level models (TLMs)
 - Fast and accurate virtual prototyping
- EE382V: Embedded Sys Dsgn and Modeling, Lecture 8 © 2010 A. Gerstlauer 40

Lecture 8: 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