ECE382N.23: Embedded System Design and Modeling

Lecture 7 – System Architecture Modeling

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

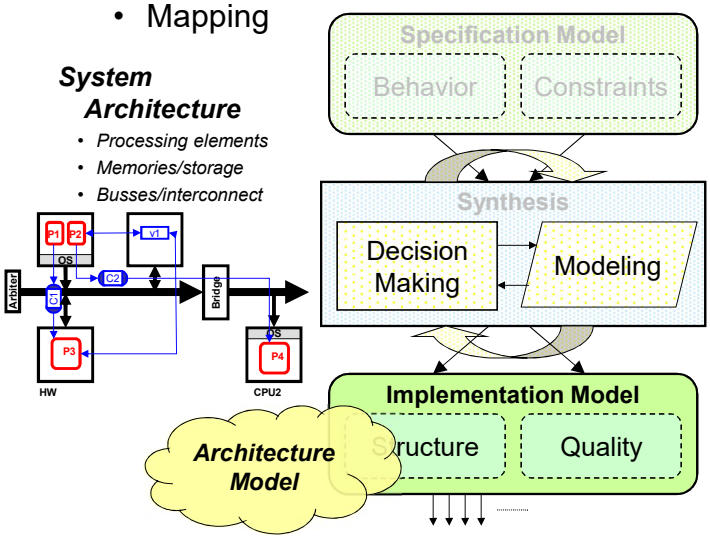
The University of Texas at Austin
Chandra Department of Electrical and Computer Engineering
Cockrell School of Engineering

Lecture 7: Outline

- **Host-compiled modeling of computation**
  - Source-level application models
  - Operating system and processor models
- **Transaction-level modeling of communication**
  - Communication layers and protocol stacks
  - Transaction-level modeling (TLM)
- **Virtual platform prototyping**
  - System-on-chip modeling

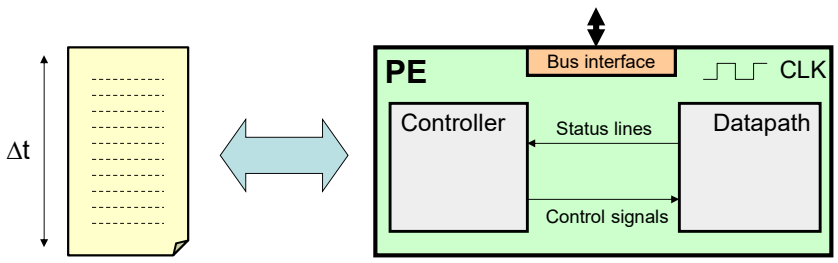
System Architecture Modeling

- **System definition**
  - Platform netlist
  - Mapping
- **System quality**
  - Performance, power, …



Computation Models

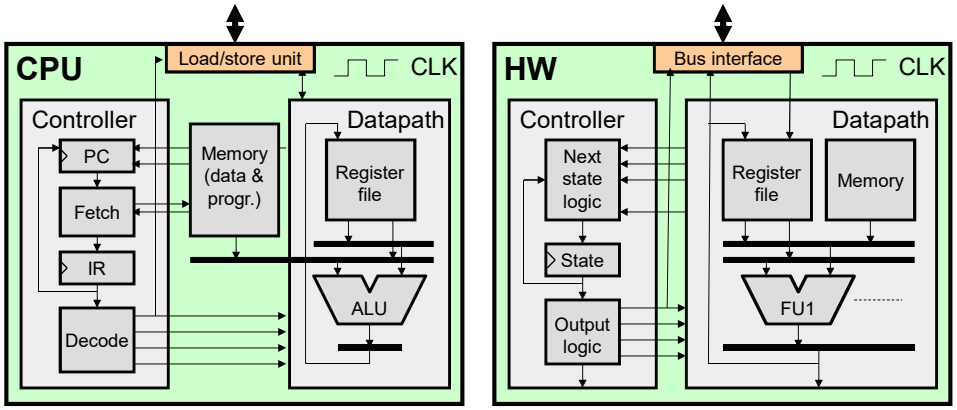
- **Basic component is a *processing element (PE)***
  - Programmable, general-purpose software processor (CPU)
  - Programmable special-purpose processor (e.g. DSPs)
  - Application-specific instruction set processor (ASIP)
  - Custom hardware processor



➢ **Functionality *and* timing (and power and …)**

Computation Modeling (1)

- **Structural RTL models**

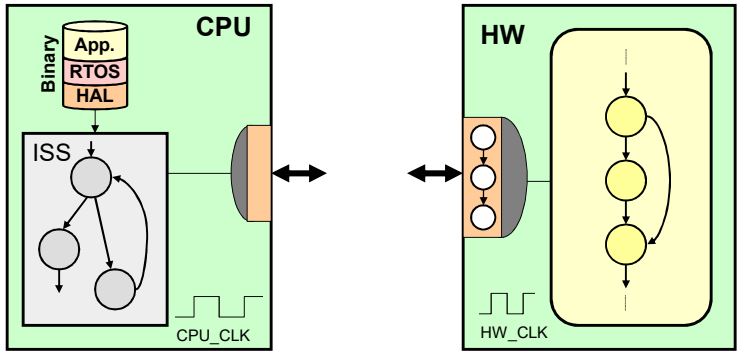


Software processor Hardware processor

➢ **Sub-cycle accurate**

Computation Modeling (2)

- **Behavioral RTL models (FSMD)**
- **Instruction-set simulation (ISS) models**
  - Purely functional (binary translation) [QEMU,…]
  - Micro-architectural (RTL in C) [GEM5,…]

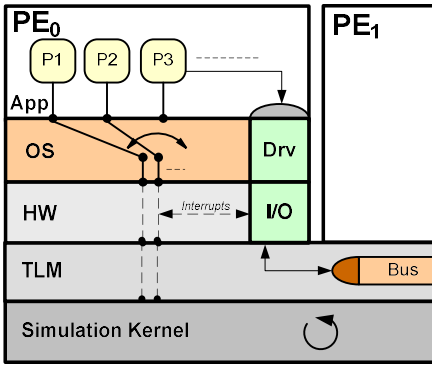


Instruction set simulation (ISS) FSMD

➢ **Cycle or timing accurate**

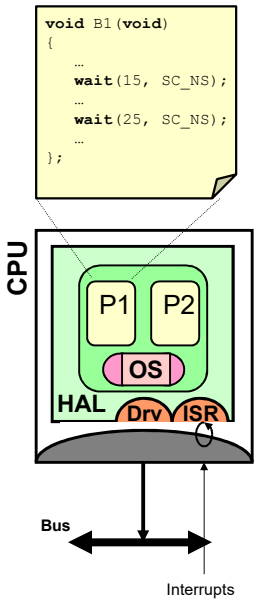
Computation Modeling (3)

- **Host-compiled models**
  - Source-level application model
    - Compile & execute natively
    - Fast functional simulation
  - Back-annotate timing and other metrics
  - Abstract OS and processor models
  - Transaction-level model (TLM) backplane
  - C-based discrete-event simulation kernel [SpecC,SystemC]



➢ **Fast and accurate full-system simulation**

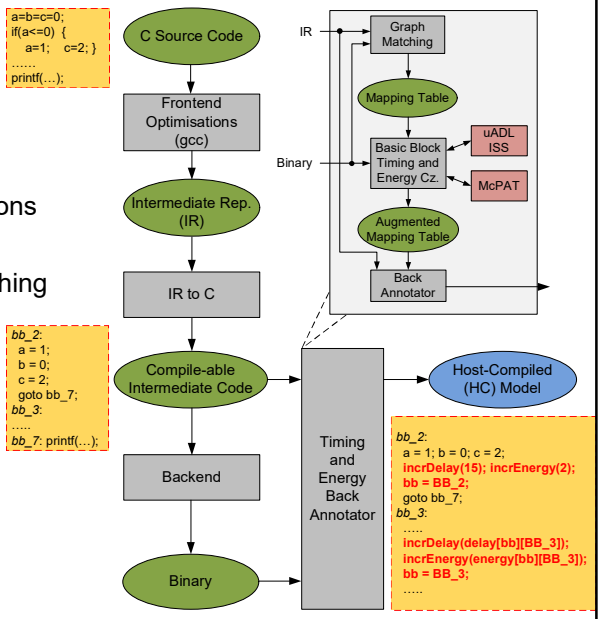
Host-Compiled Modeling Layers



- **Application**
  - Process execution (C code)
  - Execution timing
- **OS & processor**
  - Operating system
    - Real-time multi-tasking (RTOS model)
    - Bus drivers (C code)
  - Hardware abstraction layer (HAL)
    - Interrupt handlers
    - Media accesses
  - Processor hardware
    - Bus interfaces (I/O state machines)
    - Interrupt suspension and timing

Source-Level Modeling

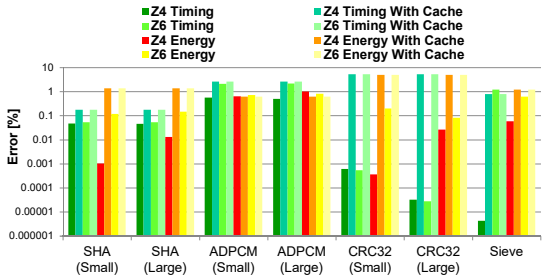
- **Automatic back-annotation**
  - Basic block level
    - Intermediate compiler representation (IR)
    - After frontend optimizations
  - Target binary matching
    - Control-flow graph matching
    - Memory access re-construction
  - Basic block characterization
    - Cycle-accurate or RTL
    - Energy model [McPAT]
  - Back-annotation
    - IR basic block level
    - Optional cache model

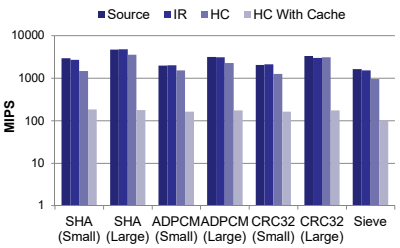


Source: Z. Zhao, L. John, A. Gerstlauer. "Source-Level Performance, Energy, Reliability, Power and Thermal (PERPT) Simulation," IEEE TCAD, 2016.

Source-Level Simulation

- **One-time back-annotation overhead**
  - 3min. to 3s runtime (function of code size)

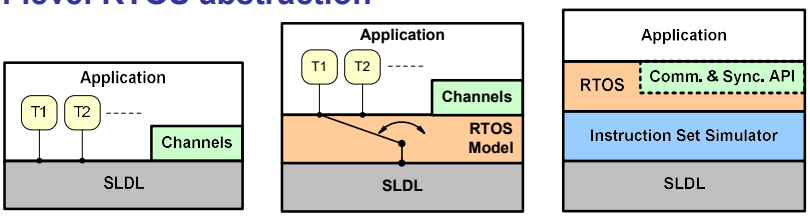




- **Close to cycle-accurate at source-level speeds**
  - >98% timing and energy accuracy @ 2000 MIPS
  - >95% accuracy @ 160 MIPS including cache
- **Extensions to power, thermal & reliability modeling**

OS Modeling

- **High-level RTOS abstraction**



Specification *Host-Compiled* ISS/cycle-accurate

  - Specification is fast but inaccurate
    - Native execution, truly concurrent model
  - Traditional ISS-based validation infeasible
    - Accurate but slow (esp. in multi-processor context), requires full binary

  - ➢ Model of operating system (task interleaving in time)
    - ➢ High accuracy but small overhead at early stages
    - ➢ Focus on key effects, abstract unnecessary implementation details
    - ➢ Model all concepts: multi-tasking, scheduling, preemption, interrupts, IPC

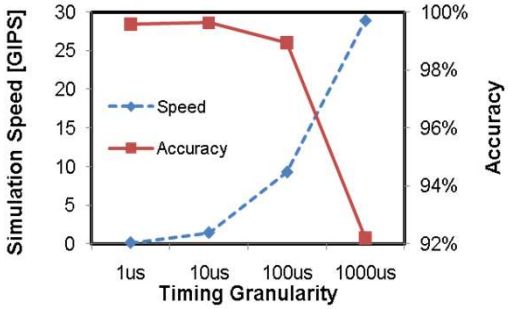
Source: A. Gerstlauer, H. Yu, D. Gajski. "RTOS Modeling for System-Level Design," DATE03.

OS Modeling Results

- **Configurable, generic and flexible OS model**
  - Configurable scheduling strategies and parameters
    - Round-robin or priority-based scheduling
  - ➢ Scheduling exploration

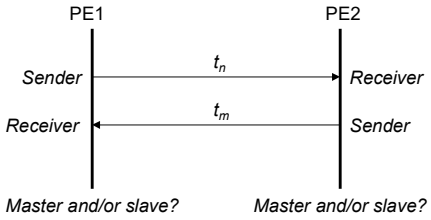
- ➢ **Accuracy & speed**
  - Artificial task set example

Granularity	Avg. speed per core	Avg. err.
1 µs	140 MIPS	0.4 %
10 µs	1500 MIPS	0.4 %
100 µs	9000 MIPS	1.0 %
1000 µs	29000 MIPS	8.0%



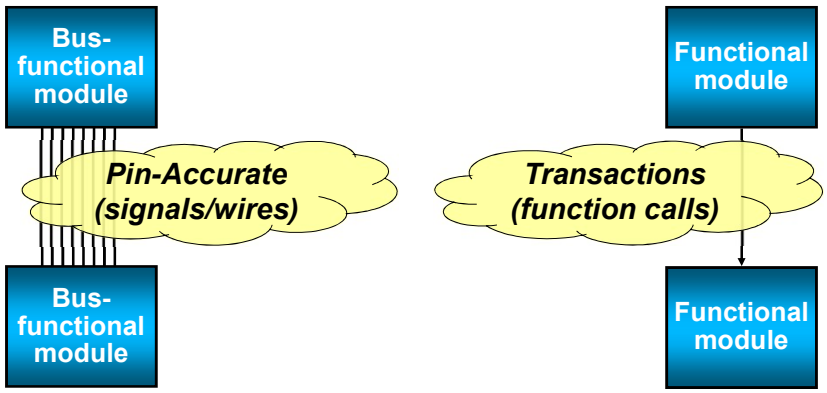
Communication Models

- **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 (dual-port)
  - Node-based bus (e.g. Ethernet, CAN):
    - Sender is master, receiver is slave
- **Reliable (loss-less, error-free)??**

Communication Modeling

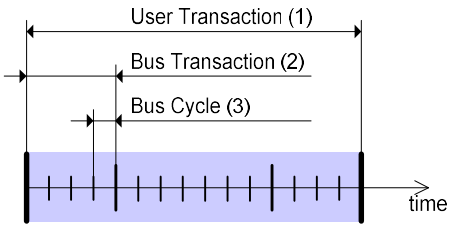


- **Pin-accurate model (PAM)**
  - Simulate every event (protocols)
- **Transaction-level model (TLM)**
  - Communications by transactions (abstract channels)
  - Granularity of transactions? Dynamic effects?

Source: OSCI TLM-2.0

TLM Abstraction Levels

- **ISO/OSI reference layer-based architecture**
  - Granularity of data and arbitration handling
- **Abstraction levels**
  1) Media Access Control (MAC)
     - User Transaction
       - » Contiguous block of bytes
       - » Arbitrary length, base address
  2) Protocol
     - Bus Transaction
       - » Bus primitives (e.g. store word)
       - » Observes bus address restrictions
  3) Physical
     - Bus Cycle
       - » Drive or sample bus wires on bus cycle



- **Abstraction levels define granularity**
  - Higher level yields a more coarse-grain model

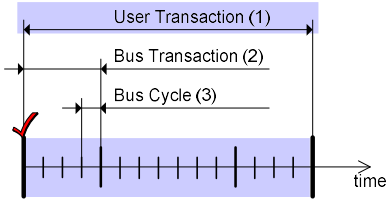
Source: G. Schirner, R. Dömer, "Quantitative Analysis of the Speed/Accuracy Tradeoff in Transaction-Level Models," ACM TECS, 2008.

Media Access Model (MAC)

Implemented Layers:

MAC

Granularity:



- **User transaction (message)**
  - Arbitrary length, contiguous block of bytes
- **No arbitration: contention avoidance by semaphore**
  - Resolution depends on simulator

➢ **Expected to be the fastest model**

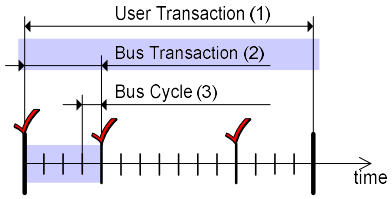
- Single `memcpy`, Single `time wait`

Transaction Level Model (TLM)

Implemented Layers:

- MAC
- Protocol

Granularity:



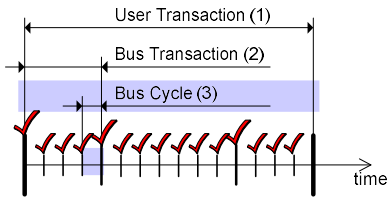
- **Bus primitives**
  - StoreWord, StoreBurst4
- **Abstract model**
  - Not pin accurate, not bus cycle accurate in all cases
- **Priority arbitration per bus transaction**
  - ➢ May lead to wrong arbitration decision, depending on execution order

Bus Functional Model (BFM)

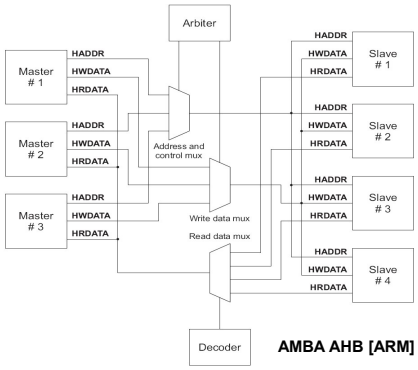
Implemented Layers:

- MAC
- Protocol
- Physical

Granularity:



- **May or may not be pin accurate**
- **Bus cycle accurate**
  - Arbitration check ✓ on each cycle
- **Includes additional active components**
  - Multiplexers (tri-state-free bus)
  - Arbiter
  - Address decoder
  - Clock generator

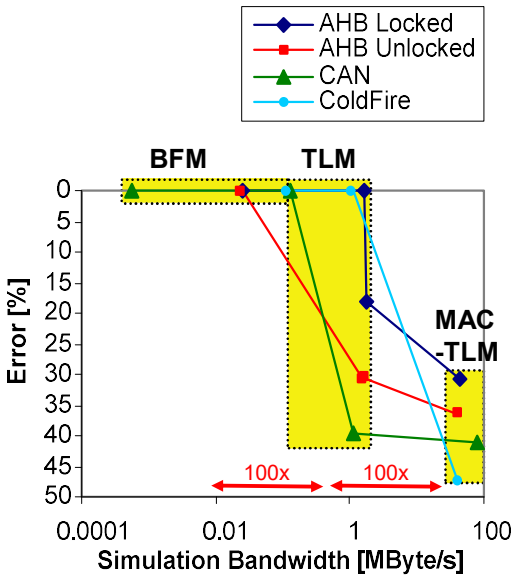


AMBA AHB [ARM]

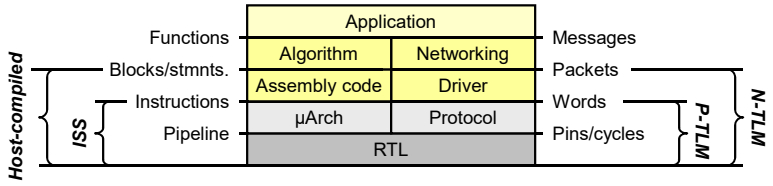
TLM Trade-Off

- **Bus models**
  - AMBA AHB, CAN, ColdFire
  - BFM, TLM, MAC-TLM
- **Performance analysis**
  - 2 masters, 2 slaves
  - Randomly distributed traffic
  - 100 byte transactions
  - 40% bus contention
  - ➢ Transfer duration

Model	Speed	Error
M-TLM	<100 MByte/s	32% - 47%
TLM	~ 1 MByte/s	18% - 39%
BFM	<0.2 MByte/s	0%

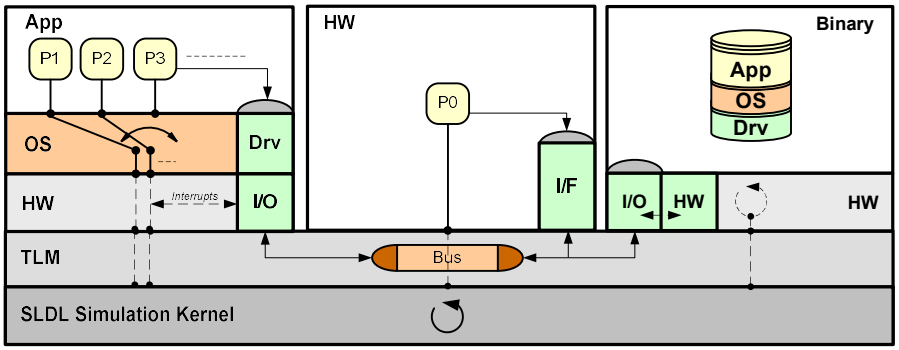


Modeling Levels



- **Host-compiled modeling of computation**
  - Abstract execution above instructions
  - ➢ Native execution of functionality
  - ➢ Back-annotation of timing, energy, …
  - ➢ Models of execution environment (OS & processor)
- **Transaction-level modeling (TLM) of communication**
  - Abstract transactions above pins and wires
  - ➢ Function calls for data transfer functionality
  - ➢ Back-annotation of timing, energy, …
  - ➢ Models of topology and glue logic

Virtual Platform Models



- **CPU model**
  - Source-level timing, energy, .. back-annotation
  - OS & processor models
- **Hardware/IP model**
  - Functional model
  - Timing, energy, … back-annotation
- **ISS model**
  - Cycle-accurate [GEM5]
  - Functional [QEMU] + timing, energy, … back-annotation
- **System-level design language (SLDL) & TLM backplane [SystemC]**

Lecture 7: Summary

- **Host-compiled computation modeling**
  - Model of software running in execution environment
    - Timed application, OS, processor and interface models
    - ➢ Embedded software development and validation
    - ➢ Viable complement to ISS-based validation
- **Transaction-level communication modeling**
  - Systematic, structured communication design flow
    - Protocol stacks and communication layers
    - ➢ Rapid, early feedback, validation and exploration
    - ➢ Various levels of abstraction, accuracy vs. speed tradeoffs
- ➢ **SoC simulation models**
  - ➢ Virtual platform prototypes