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

February 24, 2021
Reading Assignment

- This lecture: 3.3, 3.4
- Next lecture: 3.4
Outline

**System Modeling**

Software Processor Modeling

Application Layer
System Design Challenges

▶ Input: a high-level system specification
  ▶ Could be functionalities only described in MoCs
▶ Output: a low-level system implementation
  ▶ Software: programs for the targeted instruction sets
  ▶ Hardware: what level?
▶ Methodologies and tools are mature for abstraction levels at and below RTL for hardware.
  ▶ For system design, hardware implementations stop at RTL.
▶ No single step solution for system design
  ▶ Semantic gap: there are multiple ways to implement a single MoC
  ▶ Huge semantic gap exists between specification and implementation.
System Design Process

▶ Decompose the whole system design process into a series of smaller steps.
  ▶ Ensure the semantic gap is small enough for a single step
▶ Each step is defined by a pair of system models.
  ▶ The one at higher abstraction level serves as specification.
  ▶ The one at lower abstraction level serves as implementation.
▶ Refinement: generate the implementation from the specification for each step
  ▶ Introduce additional details limited to certain scope of the specification
  ▶ Incorporate design decisions to choose one implementation from multiple possible ones
▶ While more tools are available for refinement, it is critical for designers to provide proper design decisions.
  ▶ Especially when an initial system implementation fails to meet design constraints and multiple design iterations are necessary.
Typical System Design Tool and Design Process

![Diagram of system design and modeling flow](Gajski et al.)
Roles of Models

- The implementation of the previous design step will serve as the specification of the next one.
- For implementation, models allow designers to reason about design decisions by simulating and analyzing certain aspects of the system.
- For specification, models document system features that need to be implemented and decided.
- As design progresses,
  - More details are included into the models so simulation and analysis takes more time to finish.
  - Simulation and analysis will become more accurate due to the available details.
  - Designers will be able to afford the increased simulation and analysis time by focusing on the most important parts of the system.
Abstraction Levels for System Design

- At the highest abstraction level, we would assume the system is specified using process-based models while each process is specified using state-based models.
- What intermediate abstraction levels should we introduce for HW/SW implementations?
- Separate communication from computation
  - Compilers will help to implement a single process as HW or SW.
  - Communications become limiting factors for system performance.
- Accurate system analysis demands accurate communication modeling.
  - The ratio of communication latency to computation latency generally increases as more transistors are packed into a chip.
  - Complex system requires more data to be transferred among subsystems, resulting in latency with limited bandwidth and excessive power consumption.
Processor and Communication Modelings

- Processor modeling provides necessary details to evaluate mappings from processes to processors.
  - Processors: a.k.a. processing elements
    - General-purpose processors
    - DSPs and ASIPs
    - ASICs
  - The computation part of system modeling
- The communication part of system modeling is based on communication modeling.
Processor Modeling

- Processors specific to certain applications
  - Processes are usually specified with MoCs that can be directly mapped to the processors.
  - Based on hardware metrics, estimating system design metrics could be straightforward.
  - Otherwise, we can model such processors as special cases of general-purpose processors.
- General-purpose processors, or software processor
  - Processes are usually specified as sequential programs.
  - To estimate system design metrics, one has to consider not only the programs but the supporting software (e.g. OS).
  - Much more complicated than the above case. Will be our focus.
- Challenges for software processors modeling
  - Most estimations of system design metrics, e.g. latency, throughput, and power consumption, depend on simulation.
  - Models enable fast simulations while provide accurate (relatively) estimations are desired.
  - At what level should we model software processors?
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.
System Design Models

- **Cycle-Accurate Model (CAM)**
  - Both computation and communication are specified cycle-by-cycle.
  - Model functionality in implementation

- **Specification Model (SM)**
  - Both computation and communication are not timed.

- **Timed Functional Model**

- **Transaction-Level Model (TLM)**
  - Both computation and communication are approximated timed

- **Bus Cycle-Accurate Model (BCAM)**

- **Computation Cycle-Accurate Model (CCAM)**
A design methodology is illustrated as a path from A to F.
We will be interested in the one passing C.
Outline

System Modeling

Software Processor Modeling

Application Layer
Modeling Considerations

- At instruction level, processors can be modeled as FSMs, and processes can be modeled as assembly programs. However,
  - Simulations are slow as the supporting software are treated the same way.
  - With all the details mixed together, it is difficult for the designers to make design decisions for improvements, especially when multiple processes are mapped to a single processor.

- Separate the computation into layers
  - E.g. processor, OS/library, application processes
  - Allow designers to reason about various parts of the system.
  - Certain functionality, e.g. those from OS/library, can be simulated faster at levels higher than instructions.

- Software processor modeling is not limited only to the processors themselves but should include all pieces of the supporting software.
Typical Modeling of Software Processor

**FIGURE 3.8** Processor modeling layers

(Gajski et al.)
Typical Layers of Computation

- **Processor Hardware (HW)**
  - Actual hardware that communicates with the other parts of the system.

- **Hardware Abstraction Layer (HAL)**
  - Provide abstractions of actual hardware as canonical interfaces as most modern OS are hardware independent.

- **Operating System (OS)**
  - Provide necessary services like multi-tasking, scheduling, inter-process communication

- **Application**
  - With the help of OS, we can still model multiple processes mapped to the same processor as they are.

- **Notion of time can be introduced as execution delay.**
  - Via back-annotation
Outline

System Modeling

Software Processor Modeling

Application Layer
Applications are modeled as communicating processes.

- All these processes are running on the SAME processor so somewhat you may ignore the costs associated with their communications.

Processes may be composed hierarchically, e.g. through the fork-join model.

- Process scheduling is handled at the OS layer and is irrelevant at this layer, though it may affect the performance of the system.

Communication mechanisms: shared variable and message passing

- Message passing could be in its special and high-level forms like events, semaphores, queues, and abstract channels.
- Message passing is most likely implemented at the OS layer, though we need to include the specification for applications.
Specifying Applications

- It is quite common that the initial application specifications are migrated from some other systems.
  - Computations are specified in a programming language like C/C++
  - Except shared variables, communications are specified as function calls to certain library/OS.
  - Process management are specified in similar ways like communication.

- Methods for migrating code
  - Computations are included directly.
  - Function calls for communications/process management can be replaced by corresponding primitives.
Considerations for Simulation

- Fast simulation
  - Computation could be emulated in host instructions, usually much faster than simulating target instructions.
  - Simulation kernel may be optimized to reduce process management/scheduling overhead and communication costs, and to utilize the multi-processing power of the host.

- Accurate estimations
  - Back-annotation is necessary if timing behavior is of concern as simulation is not performed on targeted platform.
  - System design languages support back-annotations via execution timings, which are independent of their running times on host.
Application Layer Modeling Example

**FIGURE 3.9  Application layer**

(Gajski et al.)
More about Back-Annotation

- Back-annotations may be included at various levels of computations.
  - E.g. the whole process or basic blocks
- As a process may react differently for different inputs, assigning it a simple delay number could be inaccurate.
- Associating delay numbers with basic blocks will be able to capture such dynamic behavior and thus improve accuracy.
  - However, it will slow down simulation because of the excessive interactions with the simulation kernel required by back-annotation.
- There is always a trade-off between simulation speed and accuracy.
- The delay numbers may be obtained by estimation or measurement.
Summary

- Processor modeling should include all pieces of supporting software, e.g. HAL and OS.
- Application layer includes functional models that can be efficiently simulated.
- Back-annotations enable accurate estimations.