Computer Architecture

Architectural Simulation

Prof. Onur Mutlu
ETH Zürich

Fall 2024

6 December 2024

Potential New Ideas Evaluation Methods

- How do we assess how an idea will affect a target metric X?
- A variety of evaluation methods are available:
  - Theoretical proof
  - Analytical modeling/estimation
  - Simulation (at varying degrees of abstraction and accuracy)
  - Prototyping with a real system (e.g., FPGAs)
  - Real implementation

The Difficulty in Architectural Evaluation

- The answer is usually workload dependent
  - E.g., think caching
  - E.g., think pipelining
- Workloads change
- System has many design choices and parameters
  - Architect needs to decide many ideas and many parameters for a design
  - Not easy to evaluate all possible combinations!
- System parameters may change

Dreaming and Reality

- An architect is in part a dreamer, a creator
- Simulation is a key tool of the architect
  - Allows the evaluation & understanding of non-existent systems
- Simulation enables
  - The exploration of many dreams
  - A reality check of the dreams
  - Deciding which dream is better
- Simulation also enables
  - The ability to explore false dreams

Why High-Level Simulation?

- Problem: RTL simulation is intractable for design space exploration → too time consuming to design and evaluate
  - Especially over a large number of workloads
  - Especially if you want to predict the performance of a good chunk of a workload on a particular design
  - Especially if you want to consider many design choices
    - Cache size, associativity, block size, algorithms
    - Memory control and scheduling algorithms
    - In-order vs. out-of-order execution
    - Reservation station sizes, load/sttore queue size, register file size, …

Different Goals in Simulation

- Explore the design space quickly and see what you want to
  - potentially implement in a next-generation platform
  - propose as the next big idea to advance the state of the art
  - the goal is mainly to see relative effects of design decisions
- Match the behavior of an existing system so that you can
  - debug and verify it at cycle-level accuracy
  - propose small tweaks to the design that can make a difference in performance or energy
  - the goal is very high accuracy
- Other goals in-between:
  - Refine the explored design space without going into a full detailed, cycle-accurate design
  - Gain confidence in your design decisions made by higher-level design space exploration

Tradeoffs in Simulation

- Three metrics to evaluate a simulator
  - Speed
  - Flexibility
  - Accuracy

- Speed: How fast the simulator runs (xIPS, xCPS, slowdown)
- Flexibility: How quickly one can modify the simulator to evaluate different algorithms and design choices?
- Accuracy: How accurate the performance (energy) numbers the simulator generates are vs. a real design (Simulation error)

- The relative importance of these metrics varies depending on where you are in the design process (what your goal is)

Trading Off Speed, Flexibility, Accuracy

- Speed & flexibility affect:
  - How quickly you can make design tradeoffs/decisions
- Accuracy affects:
  - How good your design tradeoffs/decisions might end up being
  - How fast you can build your simulator (simulator design time)
- Flexibility also affects:
  - How much human effort you need to spend modifying the simulator
- You can trade off between the three to achieve design exploration and decision goals

High-Level Simulation

- Key Idea: Raise the abstraction level of modeling to give up some accuracy to enable speed & flexibility (and quick simulator design)

- Advantage

  + Can still make the right tradeoffs, and can do it quickly

    + All you need is modeling the key high-level factors, you can omit corner case conditions

    + All you need is to get the "relative trends" accurately, not exact performance numbers

- Disadvantage

  -- Opens up the possibility of potentially wrong decisions

    -- How do you ensure you get the "relative trends" accurately?

Simulation as Progressive Refinement

- High-level models (Abstract, C)
- …
- Medium-level models (Less abstract)
- …
- Low-level models (RTL with "everything" modeled)
- …
- Real design

- As you refine (go down the above list)
  - Abstraction level reduces
  - Accuracy (hopefully) increases (not necessarily, if not careful)
  - Flexibility reduces; Speed likely reduces except for real design
  - You can loop back and fix higher-level models

Making The Best of Architecture

- A good architect is comfortable at all levels of refinement
  - Including the extremes
- A good architect knows when to use what type of simulation
  - And, more generally, what type of evaluation method

Goal of Simulation Dictates Many Things

- Drives many design choices and what simulator to build/use
- Speed, flexibility, accuracy
- Many possible goals (as discussed earlier)
  - Entire system performance estimation
  - Component performance estimation (cache, SSD, memory, …)
  - Profiling for statistics
  - Understanding bottlenecks of an existing system
  - ...

What, How, Where, When

- What do you simulate?
  - Component(s) or full system
  - Program(s) or trace(s)
- How do you simulate it?
  - Many choices: Functional vs. timing, event-driven vs. cycle-by-cycle, state maintenance & recovery, …
- Where do you simulate (each thing you want to simulate)?
  - Software vs. hardware-accelerated
- When do you do things in simulation?
  - Oracle information vs. execute-like-a-real-machine

Simulator Inputs and Outputs: Generalized

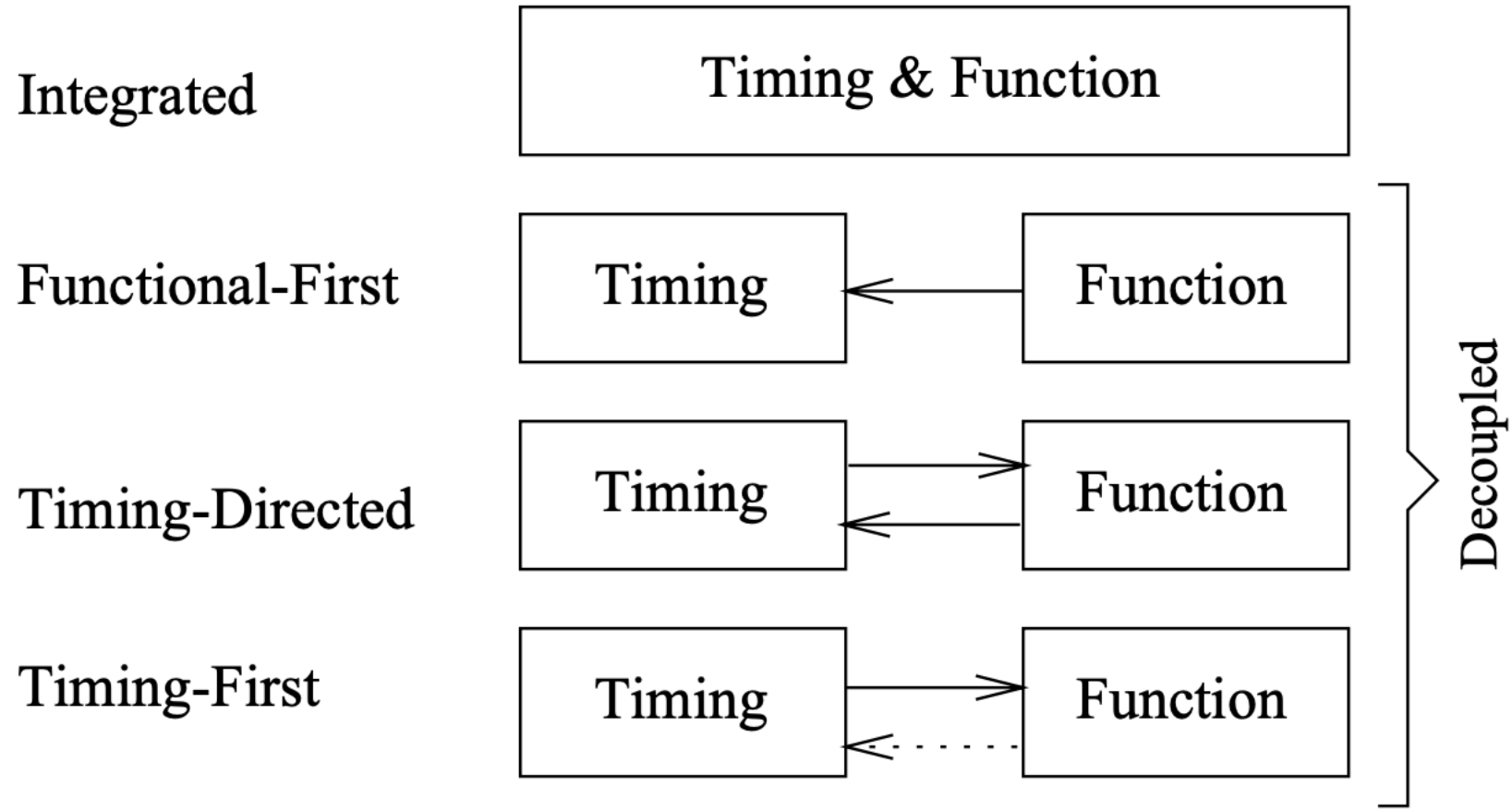
- Inputs
  - Program binary (can be multiple program binaries)
  - System state/checkpoint (e.g., including OS, memory, devices, storage, network, …)

- Outputs
  - Functional
    - Program results
    - Statistics about functional execution
  - Timing related
    - Execution time of each program
    - System throughput of all programs
    - Statistics about timing and performance events

Some General Issues in Architecture Simulation

- Functional vs. Timing Simulation
  - Purpose: When/why functional vs. timing
  - Integration of functional and timing simulation
- Full-System vs. Component Simulation
  - Purpose: When/why full system vs. component(s)
  - What should be modeled? OS, VMs, memory allocator?
- Complete Workload vs. Sampling Based Simulation
  - How do you form workloads to simulate; what parts to simulate?
- Warm-Up of Simulated Structures
  - Steady-state vs. cold-start (e.g., caches, branch predictor tables)

Functional vs. Timing: Many Choices



Arrows indicate inter-simulator interactions per committed instruction.

Figure 1. Simulator Organizations

Mauer et al., "Full-system timing-first simulation," SIGMETRICS 2002.

An Example Functional-First Model

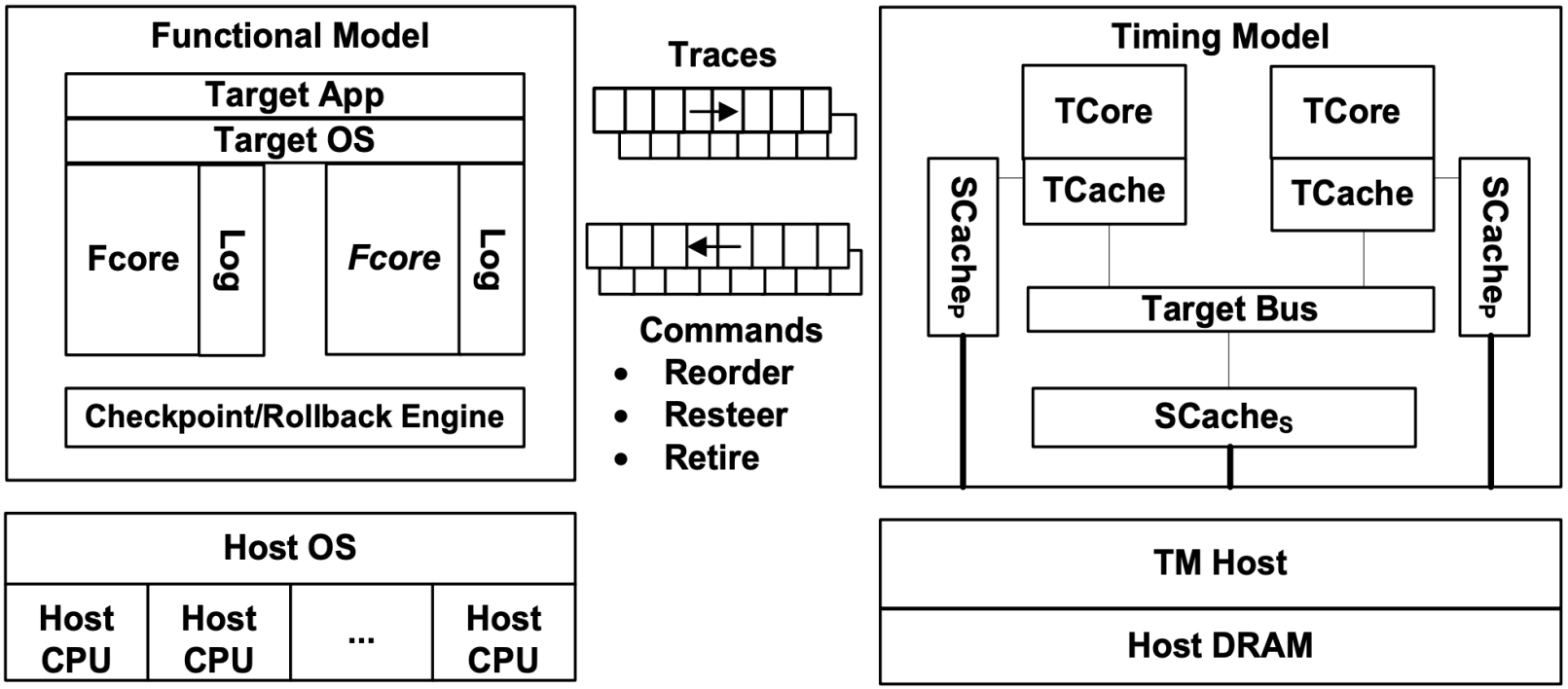


Fig. 1. FAST Simulator of a Parallel Computer System

Some General Issues in Architecture Simulation

- Validation of Accuracy
  - Functional accuracy vs. timing accuracy
  - Online vs. offline validation
- Simulation Acceleration
  - via Hardware Support (e.g., FPGAs)
  - via Software Methods (e.g., Memorization)
  - via Better Software Engineering
- Trace-driven vs. Execution-driven Simulation
  - Captured trace drives what is modeled in simulation (and timing)
  - Simulator itself emulates program execution and determines what is executed (and timing)
  - Affects what can be modeled (easily): e.g., wrong path

Some General Issues in Architecture Simulation

- Execute-at-Frontend vs. Execute-at-Execute
  - Is execution done only in functional simulator or in the timing simulator as well?
  - Timing-dependent execution becomes harder if execution is done only in the frontend, in the timing simulator
  - Examples: value prediction of L2 misses
- State Maintenance and Recovery
  - Modeling of mispredicted execution (wrong path, wrong values, …)
- Event-driven vs. Cycle-by-cycle Polling
  - One of many simulator design choices

Simulator Examples

- Ramulator 2.0 for memory simulation
- gem5 full system multi-core simulation
- MQSim for SSD simulation
- DiskSim for Hard Disk simulation
- DAMOV-Sim for Processing-near-Memory simulation
- Virtuoso for Virtual Memory (HW/SW codesign) simulation
- Sniper for fast Processor Simulation
- Scarab for detailed Microarchitectural Simulation
- Simics, Bochs, QEMU for full-system functional simulation
- …

- Or, develop your own simulator for your purpose…