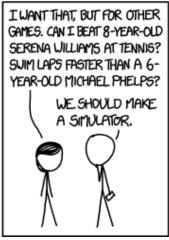
CS/ECE 752: Advanced Computer Architecture I

Architectural Simulation (mainly gem5)

Professor Matthew D. Sinclair









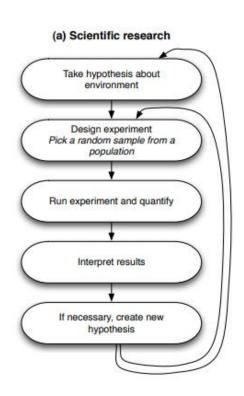
Source: XKCD

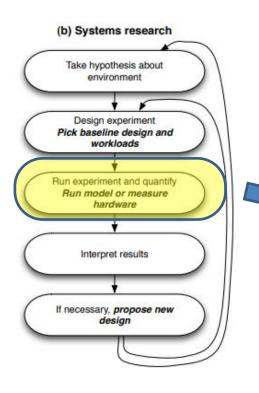
Based on slides originally developed by Sangyuen Cho (Pittsburgh), Swapnil Haria (UW-Madison), Jason Lowe-Power (UC, Davis), Onur Mutlu (CMU/ETH), and Matt Sinclair (UW-Madison)

Announcements

- Advanced Topics Voting Poll due Tuesday (tomorrow, on Piazza)
- No lecture Wednesday
- HW1 Due Saturday

Computer Systems Research/Engineering





From <u>Computer Architecture</u>
<u>Performance Evaluation</u>
<u>Methods</u> by Lieven Eeckhout

Computer architecture simulation!

Why Simulation?

- Challenge: Need a tool to evaluate systems that don't exist (yet)
 - Performance, power, energy, reliability, etc.
- Goal: I have some cool (HW) idea ... how do I determine if it any good?
- (Broadly) Three choices
 - 1. Tape out a chip! (RTL simula tim)
 - Model the behavior of the (future) hardware analytically
 - Simulate the behavior of the (future) hardware
- Tradeoffs to each all three can be valid
 - Pick the right tool for your goal [Nowatzki IEEE Micro '15]
- Today's Focus: Simulation

Option 1: Taping Out a Chip (RTL Simulation)

- RTL: Register transfer level/logic
 - Model is the hardware design
 - You specify ever wire and every register
 - Close to actual ASIC (or is actual ASIC)

Pros:

- "Cycle accurate" should be same in model and ASIC
- Very high fidelity
- Get the ultimate "right" answer
- Run programs on HW with your idea integrated, get performance/power/etc. directly

Cons:

- Takes 12-24 months (perhaps for many people) need entire design
- Limited configurability
- More difficult to combine functional and timing (later slides)
- Very costly process to make the actual HW

Option 1: Taping Out a Chip (RTL Simulation)

- Cousin of tapeout: FPGAs (e.g., FireSim)
 - Get your design running on 1+ FPGAs
 - Tradeoff tapeout costs for somewhat reduced accuracy
 - Usually (?) can get answers relatively more quickly

Option 2: Analytical Models

- Idea: use mathematical model(s) to study HW behavior
 - Estimate how much your change impacts --> plug into equation
- Examples:

Amdahl's Law:
$$S_{ ext{latency}}(s) = rac{1}{(1-p) + rac{p}{s}}$$

Queueing models: λ Waiting Service area spade

- Pros
 - Get answers extremely **quickly** (e.g., from spreadsheet)
- Cons
 - How to make models accurate?
 - Can manipulate models to show almost anything is a good idea

Option 3: Simulation

- Computer systems very complex, many interdependent parts
 - Not easy to be accurate without modeling full system
- Idea: tool that reproduces behavior of a computing device



- Simulators often written in C, C++, Python, DSLs
 - Model behavior of HW components
- System input (usually): program you want to run on the HW
- System output: (hopefully) same as real HW
- System metrics: stats about app behavior on HW model

Option 3: Simulation (Cont.)

Why use a simulator?

- Leverage faster, more flexible SW development cycle
- Simulation can be parametrized → easy to study sensitivity of different components
- Permits more (early-stage) design space exploration
- Levels of abstraction can be throttled to design task
- Possible to increase/improve system instrumentation

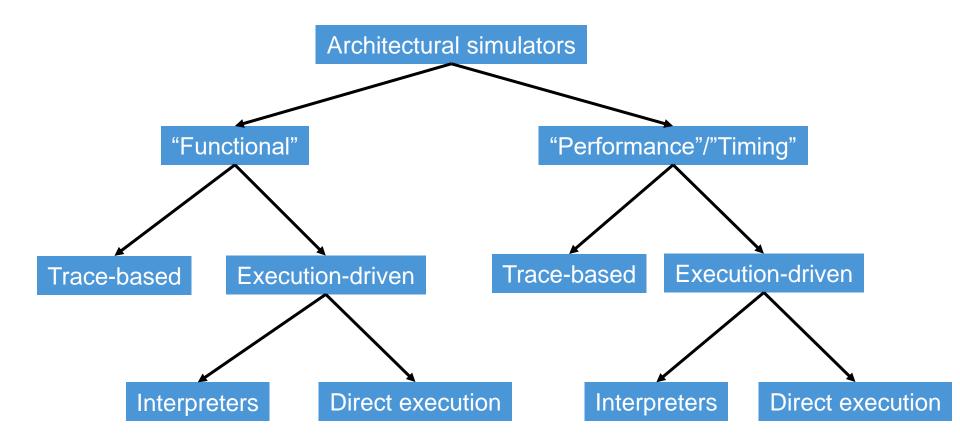
Pros

- Faster results than tapeout/FPGA (hopefully w/o big accuracy loss)
- Facilitates validation before HW becomes available
- Can do co-design with other layers (e.g., OS, compiler)

Cons

- Slower than analytical modeling (but hopefully more accurate)
- Make sure you implement HW prototypes in realistic manner

Taxonomy of Simulation Tools



Many different kinds of architecture simuators

Types of Architecture Simulation

- Functional simulation
- Instrumentation-based/Interpreters
- Trace-based
- Execution-driven
- Full system (not shown on previous slide)
- Each of these also has tradeoffs
- This Course's Focus: Execution-driven/Full System

Functional vs. Timing Simulation

- Functional simulators "just" implement the architecture
 - Executes program correctly (same as real HW)
 - Updates processor states (e.g., registers, memory)
 - Usually provides no timing information
 - Used to validate correctness of things like compilers
 - **Examples**: RISC-V Spike, QEMU, gem5 "atomic" mode
 - Can collect basic program execution information
 - How many instructions are in different classes? (instruction mix)

Functional vs. Timing Simulation (Cont.)

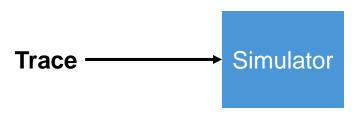
- Timing simulators implement the microarchitecture
 - (Typically) performs functional implementation
 - Model system internals (e.g., microarchitecture)
 - Pipeline, caches, branch predictor, interconnect, ...
 - Captures timing of events to obtain program execution time
 - Can collect many timing-related statistics
- Functional simulation takes less time

Types of Architecture Simulation (Cont.)

- Instrumentation
 - Often uses binary translation
 - Runs on actual HW with callbacks
 - Similar to trace-based
 - Not flexible to new ISAs (usually only works for a fixed ISA)
 - Often has some opaque things
 - Examples: Pin (CPUs), NVBit/SASSI (GPUs)

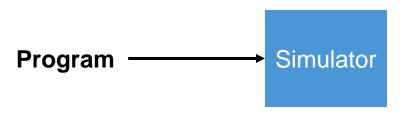
Types of Architecture Simulation (Cont.)

- Trace-based simulation
 - Simulator reads a "trace" of instructions captured during previous execution → generate addresses/events and re-execute
 - Reuses traces
 - Can be fast (doesn't need to do functional simulation)
 - If execution of program depends on timing, this will not work (e.g., synchronization, microarchitecture timing)
 - Works well for "specialized" simulators for single aspect (e.g., just cache hits/misses)



Types of Architecture Simulation (Cont.)

- Execution-driven
 - Combines functional and timing simulation
 - Simulator "runs" the program, effectively generating trace on-the-fly
 - More difficult to implement, but has many advantages
 - Direct-execution: instrumented program runs on host
 - gem5 is "execute in execute" or "timing directed"
 - **Examples**: gem5, Accel-Sim, and many others

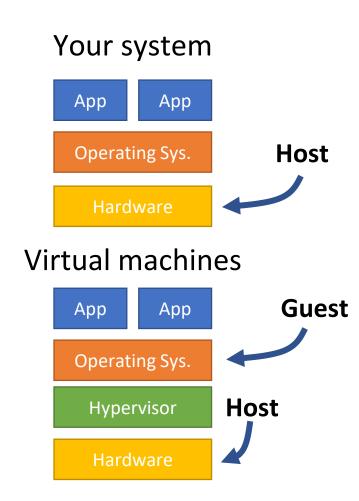


Full System Simulation

- Components modeled with enough fidelity to run mostly unmodified apps
- Often "bare metal" simulation
- Entire program is functionally emulated by simulator
- Often means running the OS in the simulator, not faking it
- Often combine functional and execution-based
- Enables studying interactions across computing stack
 - **Example**: see effects of interaction with OS
- But also often increases simulation time

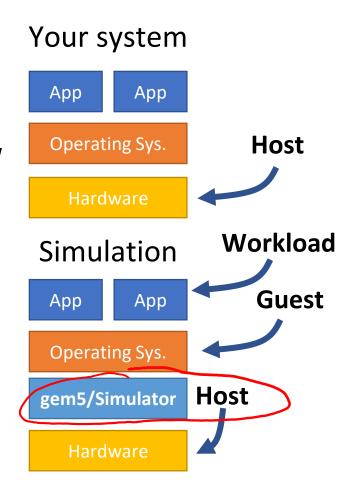
Nomenclature (Traditional)

- Host: the actual hardware you're using
- Running things directly on the HW:
 Native execution
- Guest: Code running on top of "fake" HW
 - OS in virtual machine is guest OS
 - Running "on top of" hypervisor
 - Hypervisor is emulating HW



Nomenclature (Simulation)

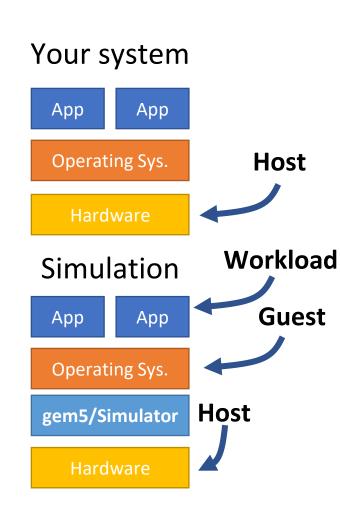
- Host: the actual HW you're using
- **Simulator**: Runs on the host
 - Exposes HW to the guest
- **Guest**: Code running on *simulated* HW
 - OS running on gem5 is guest OS
 - gem5 is simulating HW
- **Simulator's Code**: Runs natively
 - Executes/emulates the guest code
- Guest's Code: (or benchmark, workload, etc.) Runs on gem5, not on the host



Nomenclature (Simulation, Cont.)

- Host: the actual HW you're using
- **Simulator**: Runs on the host
 - Exposes HW to the guest
- Simulator's performance:
 - Time to run simulation on host
 - Wallclock time as you perceive it
- Simulated performance:
 - Time predicted by the simulator
 - Time for guest code to run on simulator

never the same



Tradeoffs in Simulation

- Development Time: time to develop simulator/models
- Evaluation Time: wallclock time to run simulator
- Fidelity: How close is the simulator to real HW
 - Many academic simulators: first-order trends (not cycle-accurate)
 - Industry simulators often focus on cycle-accurate simulation
- Flexibility: how quickly one can modify the simulator to evaluate different algorithms and design choices?
- Coverage: How broadly can the simulator be used?

	Development time	Evaluation time	Accuracy	Coverage
functional simulation	excellent	good	poor	poor
instrumentation	excellent	very good	poor	poor
specialized cache and predictor simulation	good	good	good	limited
full trace-driven simulation	poor	poor	very good	excellent
full execution-driven simulation	very poor	very poor	excellent	excellent

[Source: Computer Architecture Performance Evaluation Methods]

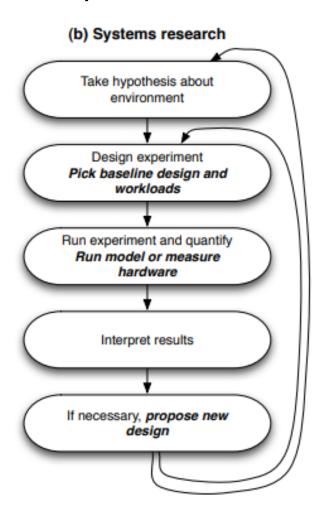
Relative importance of these metrics vary depending on where you are in design process

Trading Off Metrics (Inter-Related)

- Evaluation time & flexibility affect:
 - How quickly you can make design tradeoffs
 - (Sometimes) How accurate the model(s) is/are
- Fidelity affects:
 - How good your design tradeoffs may end up be
 - How fast you can build your simulator (simulator design time)
- Flexibility also affects
 - How much human effort you need to spend modifying the simulator
- Coverage affects what you can model
- Tradeoff between these to achieve design exploration and decision goals

What "Level" Should we Simulate At?

- Question: What fidelity is required for this problem?
 - Example: New register file design
- Often, the answer is a mix
- gem5 is well suited for this mix
 - Models with different fidelity
 - Drop-in replacements for each other
- "Cycle level" vs. "Cycle accurate"



Cycle-level Simulation (g/m5)

- Models system cycle-by-cycle
- Often "event-driven" (subsequent slides)
- Can be very accurate
 - Not exact cycle-by-cycle behavior as ASIC
 - But similar timing (gets first order trends right)
- Easily parameterizable
 - No need for full HW design
- Faster than cycle-accurate
 - Can "cheat" and functionally emulate some things

What is gem5? (History)





Created at Michigan by students of Steve Reinhardt, principally Nate Binkert.

"A tool for simulating systems"

Two Views of M5

- A framework for event-driven simulation
 - Events, objects, statistics, configuration
- 2. A collection of predefined object models
 - CPUs, caches, busses, devices, etc.

- This tutorial focuses on #2
- You may find #1 useful even if #2 is not



What is gem5? (History, Cont.)





Created at Michigan by students of Steve Reinhardt, principally Nate Binkert.

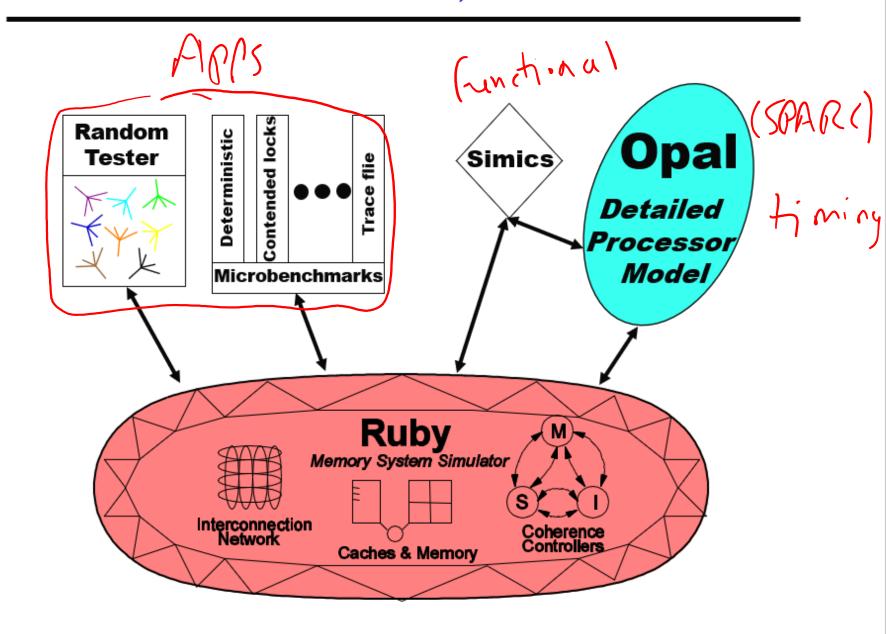
"A tool for simulating systems"



Created at Wisconsin by students of Mark Hill and David Wood.

Detailed memory system

GEMS From 50,000 Feet



m5 + GEMS = gem5





Created at Michigan by students of Steve Reinhardt, principally Nate Binkert.

"A tool for simulating systems"



Created at Wisconsin by students of Mark Hill and David Wood.

Detailed memory system

What is gem5?

Michigan m5 + Wisconsin GEMS = gem5

"The gem5 simulator is a modular platform for computersystem architecture research, encompassing system-level architecture as well as processor microarchitecture."

Lowe-Power et al. **The gem5 Simulator: Version 20.0+**. ArXiv Preprint ArXiv:2007.03152, 2021. https://doi.org/10.48550/arXiv.2007.03152

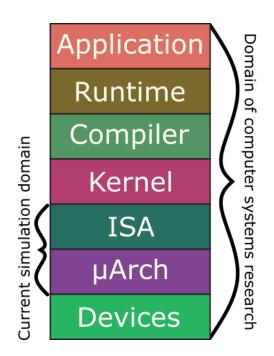
Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. **The gem5 simulator**. *SIGARCH Comput. Archit. News* 39, 2 (August 2011), 1-7. DOI=http://dx.doi.org/10.1145/2024716.2024718

gem5-20+: A new era in CA simulation

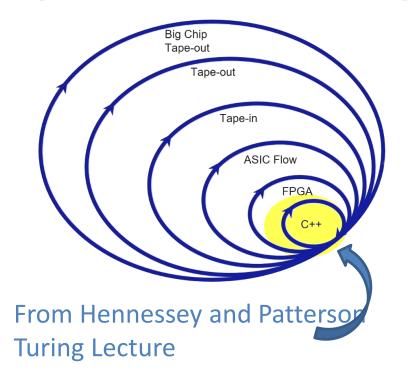


Abdul Mutaal **Bagus** Curtis Dunham Gabe Black Jakub Jermar Koan-Sin Tan Martinasso Nicolas Zea Riken Gohil Srikant Uri Wiener Gabe Loh Maximilian Stein Nikos Nikoleris Ahmad Hanindhito Dam Sunwoo James Clarkson Korey Sewell Rizwana Begum Bharadwaj Victor Garcia Beniamin Nash Dan Gibson Gabor Dozsa Jan-Peter Krishnendra Maximilien Robert Kovacsics Stan Czerniawski Vilas Sridharan Adrian Herrera Nils Asmussen Nuwan Jayasena Robert Scheffel Stanislaw Adrien Pesle Bertrand Daniel Carvalho Gedare Bloom Larsson Nathella Breughe Vince Weaver Marquis Daniel Johnson Gene WU Lena Olson Rohit Kurup Czerniawski Vincentius Adrià Armejach lason Lowe-Michael Adler Ola Jeppsson Binh Pham Power Lisa Hsu Akash Bagdia **Daniel Sanchez** Gene Wu Michael LeBeane Omar Naji Ron Dreslinski Stephan Robby Alec Roelke Bjoern A. Zeeb David Guillen-Geoffrey Blake Javier Bueno Lluc Alvarez Michael Pablo Prieto Ruben Diestelhorst Wade Walker Blake Hechtman Fandos Alexandru Dutu Georg Kotheimer Hedo Lluís Vilanova Levenhagen Palle Lyckegaard Ayrapetyan Stephen Hines Weiping Liao Rune Holm Ali Jafri Bobby R. Bruce David Hashe Giacomo Javier Cano-Cano Mahyar Samani Michiel Van Tol Pau Cabre Steve Raasch Wendy Elsasser Ali Saidi ingarov David Oehmke Gabrielli Javier Setoain Malek Musleh zuel Serrano Paul Rosenfeld William Wang lan Amin Farmahini Willy Wolff uja Anders Handler gyu Dong Andrea Mondelli B i Zhang Swapnil Haria Andrea Pellegrini Brandon Potter Diordie Hamid Reza Marco Balboni Polina Dudnik Sandipan Das Иα Jieming Yin Mingyuan Andreas HanssonBrian Grayson Polydoros Taeho Kgil Kovacevic Khaleghzadeh Jing Qu Marco Elver Mitch Hayenga Santi Galan ng Andreas Cagdas Dirik Dongxue Zhang Hanhwi Jang Jiuyue Ma Marjan Fariborz Mohammad Petrakis Sascha Bischoff Tao Zhang ckert Sandberg Chander Doğukan Hoa Nguyen Joe Gross Matt DeVuyst Alian Pouva Fotouhi Sean McGoogan Thomas Grass Andrew Bardsley Sudanthi Korkmaztürk Hongil Yoon Joel Hestness Matt Evans Monir Prakash Sean Wilson Tiago Mück Matt Horsnell Andrew Lukefahr Chen Zou Dylan Johnson Hsuan Hsu John Alsop Mozumder Ramrakhyani Sergei Trofimov Tim Harris Vang Andrew Schultz Chris Adeniyi-Earl Ou Hussein Matt Poremba Pritha Ghoshal Severin Iohn Moyang Wang Timothy Hayes Wischmann Andriani Jones **Edmund Grimley Elnawawy** Kalamatianos Matt Sinclair Mrinmoy Ghosh Radhika Jagtap Timothy M. odama Mappoura Chris Emmons Evans Ian Jiang Jordi Vaguero Matteo Nathan Binkert Rahul Thakur Shawn Rosti Jones eng Christian MenardEmilio Castillo **IanJiangICT** Jose Marinho Andreozzi Nathanael Sherif Elhabbal Tom Jablin Ani Udipi Reiley Jeapaul Anis Peysieux Christoph Pfister Erfan Azarkhish Ilias Vougioukas Jui-min Lee Matteo M. Fusi Premillieu Rekai Gonzalez- Siddhesh **Tommaso** Anouk Van Laer Christopher Eric Van Isaac Richter Kanishk Sugand Matthew Nayan Alberquilla Poyarekar Marinelli Isaac Sánchez **Arthur Perais** Torng Hensbergen Karthik Sangaiah Poremba Deshmukh Rene de Jong Somayeh Tony Gutierrez seazzw Ashkan Tousi Chuan Zhu Erik Hallnor Barrera Ke Meng Matthias Hille Neha Agarwal Ricardo Alves Sardashti Trivikram Reddy Éder F. Zulian **Austin Harris** Chun-Chen Hsu Erik Tomusk Ivan Pizarro Nicholas Lindsay Richard D. StrongSooraj Puthoor Tuan Ta Kevin Brodsky Matthias Jung Maurice Becker Nicolas Avishai Tvila Ciro Santilli Faissal Sleiman Jack Whitham Kevin Lim Richard Strong Sophiane Senni Tushar Krishna 32 Fernando Endo Ayaz Akram Clint Smullen Jairo Balart Khalique Maxime Derumigny Rico Amslinger Soumyaroop Roy Umesh Bhaskar

gem5's Goals



Agile Hardware Dev. Methodology



gem5's Goals (Cont.)

- Anyone (including non-architects) can download and use
- Widely used in academia, industry, and national labs
- Used for cross-stack research
 - Change kernel, runtime, HW, etc. all in concert
 - Run full ML stacks, as well as other emerging apps
- Living SW constantly evolving.
- Lots of rough edges, but you can help!

gem5's Architecture: SimObjects

Model

C++ code in src/

Parameters

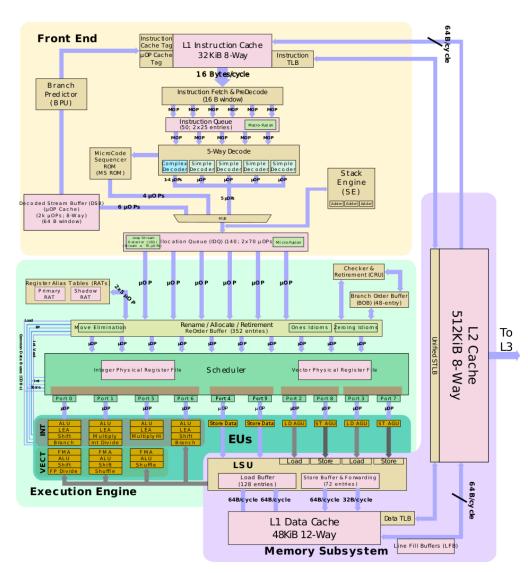
- Python code in src/
- In SimObject declaration file all C++ SimObjects exposed to Python
- This is the interface with users scripts define system to model

Instance or configuration

- A particular choice for the parameters
- In standard library, your extensions, or Python runscript
 - HW1: Python runscript

Model vs. Parameters

- Generic model and timing in C++
 - Not vendor or ISA-specific
- Expose parameters to Python
- Set parameters and connections in Python



[Source: Intel SunnyCove, Wikichip]

Some gem5 Nomenclature

- You can extend a model, to model new things
 - Want to *inherit* from the object in C++

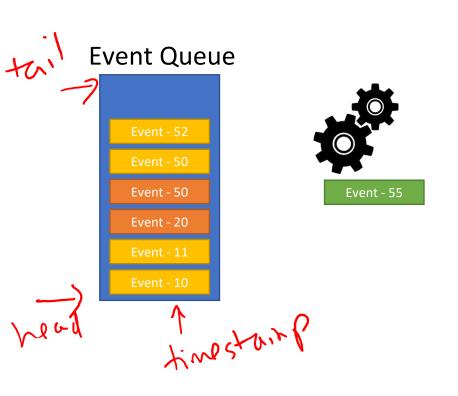
```
class 03CPU : public BaseCPU
{
```

- You can specialize a model with specific parameters
 - Want to *inherit* from the object in Python

```
class i7CPU(03CPU):
   issue_width = 10 ______
```

gem5 Architecture: Simulating

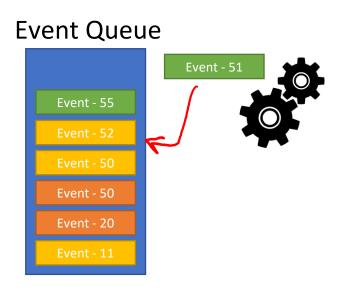
• gem5 is a discrete-even simulator



- 1) Event at head dequeued
- 2) Event executed
- 3) More events queued

gem5 Architecture: Simulating (Cont.)

gem5 is a discrete-even simulator

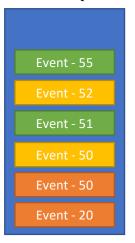


- 1) Event at head dequeued
- 2) Event executed
- 3) More events queued

gem5 Architecture: Simulating (Cont.)

gem5 is a discrete-event simulator

Event Queue

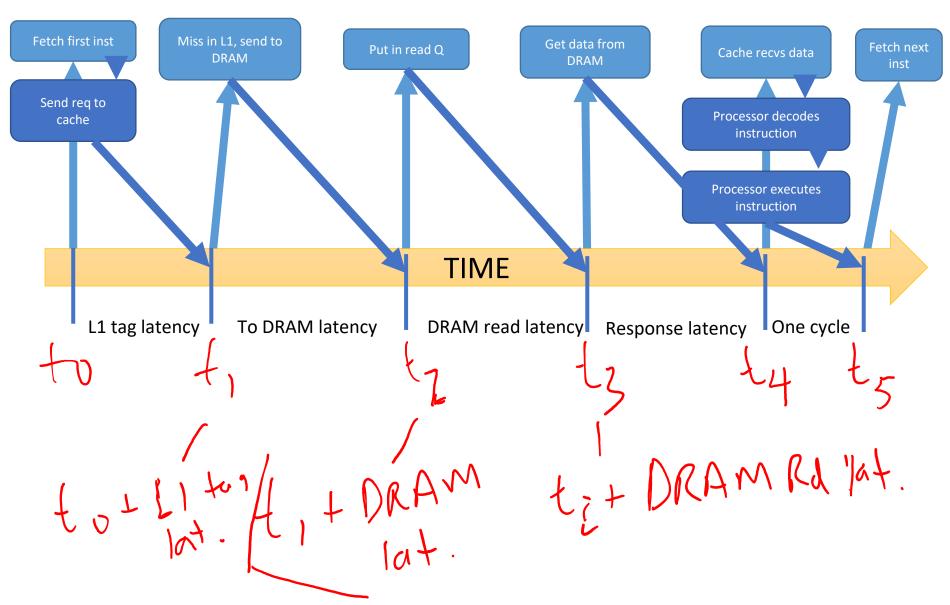




- 1) Event at head dequeued
- 2) Event executed
- 3) More events queued

All SimObjects can enqueue events to the event queue

Discrete Event Simulation Example

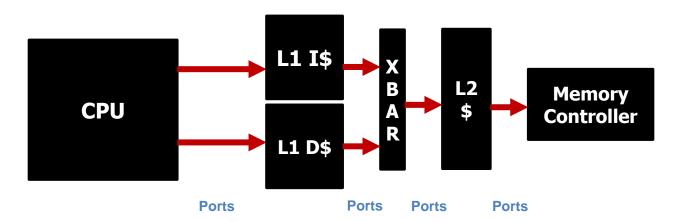


Discrete Event Simulation

- "Time" needs a unit
 - In gem5 we use a unit called "Tick"
- Need to convert a simulation "tick" to user-understandable time (e.g., seconds)
- In gem5 this is the global simulation tick rate
 - Usually this is 1 ps per tick or 10¹² ticks per second

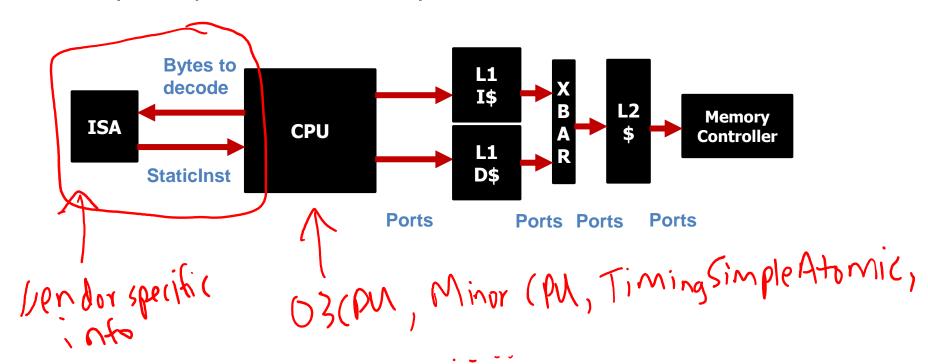
gem5's Main Abstractions

- ISA vs. CPU model & Memory request abstractions
- Ports allow you to send requests and receive responses
 - Ports are unidirectional (two types: Request, Response)
 - Anything* with a Request port can be connected to any Response port



gem5's Main Abstractions

- ISA vs. CPU model & Memory request abstractions
- CPU model abstracted from ISA (x &6, ARM, RTSC-V)...)
 - Any* CPU model can be used with any* ISA
 - **ISA** provides StaticInst with execute(), initiateAccess(), etc.
 - If you implement interface, you can add a new CPU model or ISA



Summary

- Often using simulation and modeling to evaluate our ideas
- Many different types of tools to use pick what is best for your goals
- This Course: focus on gem5
 - Widely used throughout the community (e.g., 15-25% of top tier research papers each year)
 - Event-driven, cycle-level simulator
 - Other features gem5 has (not discussed today)
 - Detailed GPU, interconnect, main memory, etc. models
 - Supports running full stack ML workloads
 - (Ongoing) Accelerator integration, SST integration
- HW1: your first attempt at using gem5!
 - Subsequent assignments/maybe course project will build on this