What is gem5 and where do I get it?

Andreas Sandberg & Nikos Nikoleris
ARM Research
Why gem5?

Runs real workloads
  • Runs complex workloads like Android & ChromeOS

System-level insights
  • Device interactions (storage, NICs, ...)
  • OS interactions like PA fragmentation

Can be wired to custom models
  • Add detail where it matters, when it matters!

Rapid early prototyping
  • Parameterized models enable rapid design space exploration

Large user base in industry & academia

But not a microarchitectural model out of the box!
Configurable level of detail

Detail vs. Speed

- Cycle Accurate
  - 1–50 KIPS
  - RTL simulation
  - Architecture exploration
  - 0.2–3 MIPS

- Approximately Timed
  - 50–200 MIPS
  - gem5

- Loosely Timed
  - GIPS
  - gem5 + kvm

- HW Virt.
  - SW Dev

Can be changed at runtime in gem5!
When not to use gem5

Performance validation

• gem5 is not a cycle-accurate microarchitecture model!
• This typically requires more accurate models such as RTL simulation.

Core microarchitecture exploration*

• gem5’s core models were not designed to replace more accurate microarchitectural models.

To validate functional correctness

• New (e.g., Armv8.0+) or optional instructions are sometimes not implemented.
• gem5 is not as rigorously tested as commercial products.
How to get involved

Tutorials: [http://gem5.org/Tutorials](http://gem5.org/Tutorials)

- The ASPLOS 2017 slides provide an up-to-date general overview

Mailinglists: [http://gem5.org/Mailing_Lists](http://gem5.org/Mailing_Lists)

- gem5-dev: Development discussions
- gem5-users: Using gem5 and running experiments

Contribute some code: [https://gem5-review.googlesource.com/](https://gem5-review.googlesource.com/) (see CONTRIBUTING.md)

gem5 is a community effort!
A brief technical overview
Example System
Configuring and running gem5

Python
- Create & config Python objects
  - Instantiate objects
    - Run simulation
      - Run simulation

C++
- Instantiate C++ objects
  - Simulate in C++
    - Simulate in C++

Simulated system
- Running guest code
  - Running guest code
**Simulating Time**

Discrete: Handles time in discrete steps (ticks)
- Usually 1THz in gem5

Simulator skips to the next event on the timeline
- More efficient than traditional clocked simulators
How are models implemented

- **Describes parameters and exported methods**
  - Python description
  - Generates Python wrappers
  - Includes Parameter structs
  - C++ model

- **Implements your model**
Examples

Configuration & running:

- **Syscall emulation**: configs/learning_gem5/part1
- **Full-system**: configs/example/arm/{fs_bigLittle.py, devices.py}

Simple memory-mapped IO devices: IsaFake

- **See**: src/dev/isa_fake.{cc,hh} and src/dev/Device.py
- **Simple PCI devices with interrupts**: PciVirtIO
- **See**: src/dev/virtio/pci.{cc,hh} and src/dev/VirtIO.py

More complex PCI device with DMA: CopyEngine

- **See**: src/dev/pci/copy_engine.{cc,hh} and src/dev/pci/CopyEngine.py
CPU models
CPU models overview

- BaseCPU
  - BaseKvmCPU
    - ArmV8KvmCPU
    - X86KvmCPU
    - Very Fast & No Timing: • No caches • No BPs
  - BaseSimpleCPU
    - TimingSimpleCPU
    - AtomicSimpleCPU
    - Fast & Timing: • Caches • Limited BPs
  - TraceCPU
    - Fast & Timing: • Caches • No BPs
  - DerivO3CPU
  - MinorCPU
    - Slow & Full timing: • Caches • Branch predictors

14:15 - Trace-driven simulation of multithreaded applications in gem5
14:45 - Generating Synthetic Traffic for Heterogeneous Architectures
Memory systems
On-chip memory system

Classic Memory
- MOESI
- Snooping
- Crossbar Interconnect

Ruby
- Flexible Coherence
- Snooping or Directory Based
- Flexible Network Topology

09:45 - Learning gem5: Modeling Cache Coherence with gem5

11:15 - A Detailed On-Chip Network Model inside a Full-System Simulator
Off-chip memory system

- **AbstractMemory**

- **SimpleMemory**
  - Fixed latency (w/ variance)
  - Fixed bandwidth

- **DRAMCtrl**
  - Detailed DRAM controller: DDRx, LPDDRx, WideIO, HBM etc
Top-down DRAM controller model

Don’t model the actual DRAM, only the timing constraints

- DDR3/4, LPDDR2/3/4, WIO1/2, GDDR5, HBM, HMC, even PCM
- See src/mem/DRAMCtrl.py and src/mem/dram_ctrl.{hh, cc}

Hansson et al, *Simulating DRAM controllers for future system architecture exploration*, ISPASS’14
Ports: Connecting memory objects

MemObjects are connected through master and slave ports

A master module has at least one master port, a slave module at least one slave port, and an interconnect module at least one of each

- A master port always connects to a slave port
- Similar to TLM-2 notation

15:15 - System Simulation with gem5, SystemC and other Tools
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद
# Workshop schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>09.30</td>
<td>Interacting with gem5 using workload-automation &amp; devlib</td>
</tr>
<tr>
<td>09.45</td>
<td>ARM Research Starter Kit: System Modeling using gem5</td>
</tr>
<tr>
<td>10.00</td>
<td>Break</td>
</tr>
<tr>
<td>10.15</td>
<td>Debugging a target-agnostic JIT compiler with gem5</td>
</tr>
<tr>
<td>10.30</td>
<td>Learning gem5: Modeling Cache Coherence with gem5</td>
</tr>
<tr>
<td>11.00</td>
<td>Break</td>
</tr>
<tr>
<td>11.15</td>
<td>A Detailed On-Chip Network Model inside a Full-System Simulator</td>
</tr>
<tr>
<td>11.45</td>
<td>Integrating and quantifying the impact of low power modes in the DRAM controller in gem5</td>
</tr>
<tr>
<td>12.00</td>
<td>Break</td>
</tr>
<tr>
<td>12.15</td>
<td>CPU power estimation using PMCs and its application in gem5</td>
</tr>
<tr>
<td>12.45</td>
<td>gem5: empowering the masses</td>
</tr>
<tr>
<td>13.00</td>
<td>Lunch</td>
</tr>
<tr>
<td>14.15</td>
<td>Trace-driven simulation of multithreaded applications in gem5</td>
</tr>
<tr>
<td>14.45</td>
<td>Generating Synthetic Traffic for Heterogeneous Architectures</td>
</tr>
<tr>
<td>15.00</td>
<td>Break</td>
</tr>
<tr>
<td>15.15</td>
<td>System Simulation with gem5, SystemC and other Tools</td>
</tr>
<tr>
<td>15.30</td>
<td>COSSIM: An Integrated Solution to Address the Simulator Gap for Parallel Heterogeneous Systems</td>
</tr>
<tr>
<td>15.45</td>
<td>Simulation of Complex Systems Incorporating Hardware Accelerators</td>
</tr>
<tr>
<td>16.15</td>
<td>Break</td>
</tr>
<tr>
<td>16.30</td>
<td>Introduction to ARM Research</td>
</tr>
<tr>
<td>18.20</td>
<td>Poster session</td>
</tr>
<tr>
<td>20.00</td>
<td>Dinner</td>
</tr>
</tbody>
</table>