MGSim

Multi-Core Research & Education

R. Poss – University of Amsterdam

SAMOS XIII – July 16th 2013

Take-away

- Abstraction level **between RTL and TLM**
  (similar to SystemC)
- Focus on research on **new architectures**
  (provided examples are research models)
- **Free and open source**
- **Small** and **comprehensible by students**
  (bigger than SimpleScalar, smaller than Gem5)
- Focus on **full-system emulation**
  (more like Gem5 than SimpleScalar)

Motivation: enable disruptive research

- Disruptive research: **ability to touch every aspect** (micro-architecture, NoC, memory, etc.)
- Friction to disruptive research:
  - Many tools with **different styles and licenses** = **technical barrier**
  - **Fragmentation of expertise** between sub-areas = **human barrier**
- **Friction prevents no-holds-barred experiments** and mandates small incremental changes
- *High-level functional simulation is easy and cheap but limited to investigate realizability of new hardware designs*

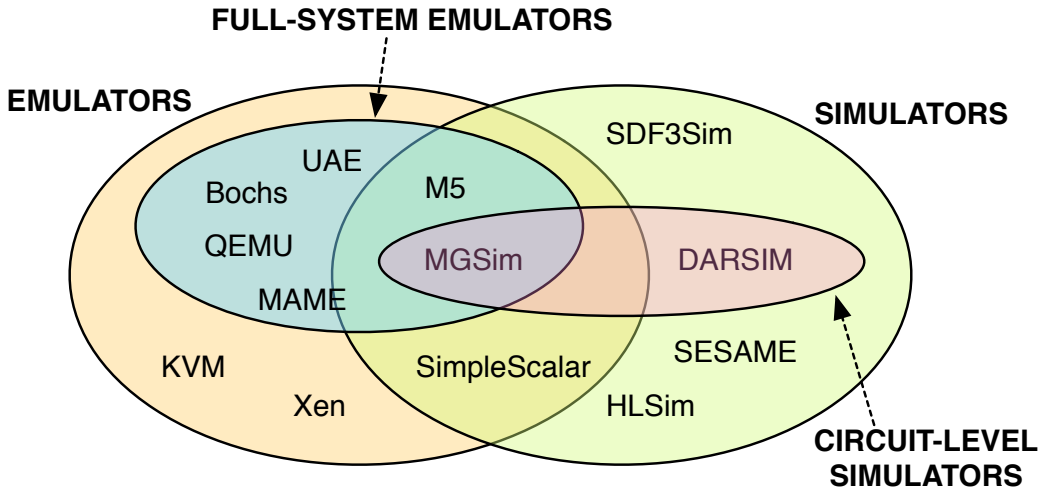
Requirement: Small

- How small is “small enough”?
  - BSc – know / understand all abstractions involved
  - MSc - know / understand all interactions in framework
  - PhD - know / understand the impact of any change, anywhere in the code

Requirement: Accurate and fast enough

- How small is “too small”?
  - Want: many-core, multi-clock, cycle-by-cycle
  - Challenge:
    sim: ≧1G cycles, ≧100 cores
    real: ≦1day, ≦100MB RAM
- Higher level than FUs too inaccurate
- RTL / Graphical / small-scale too slow
- Current cheap FPGAs too small for ≧100 cores

Requirement: Full-system emulation + Circuit-level simulation



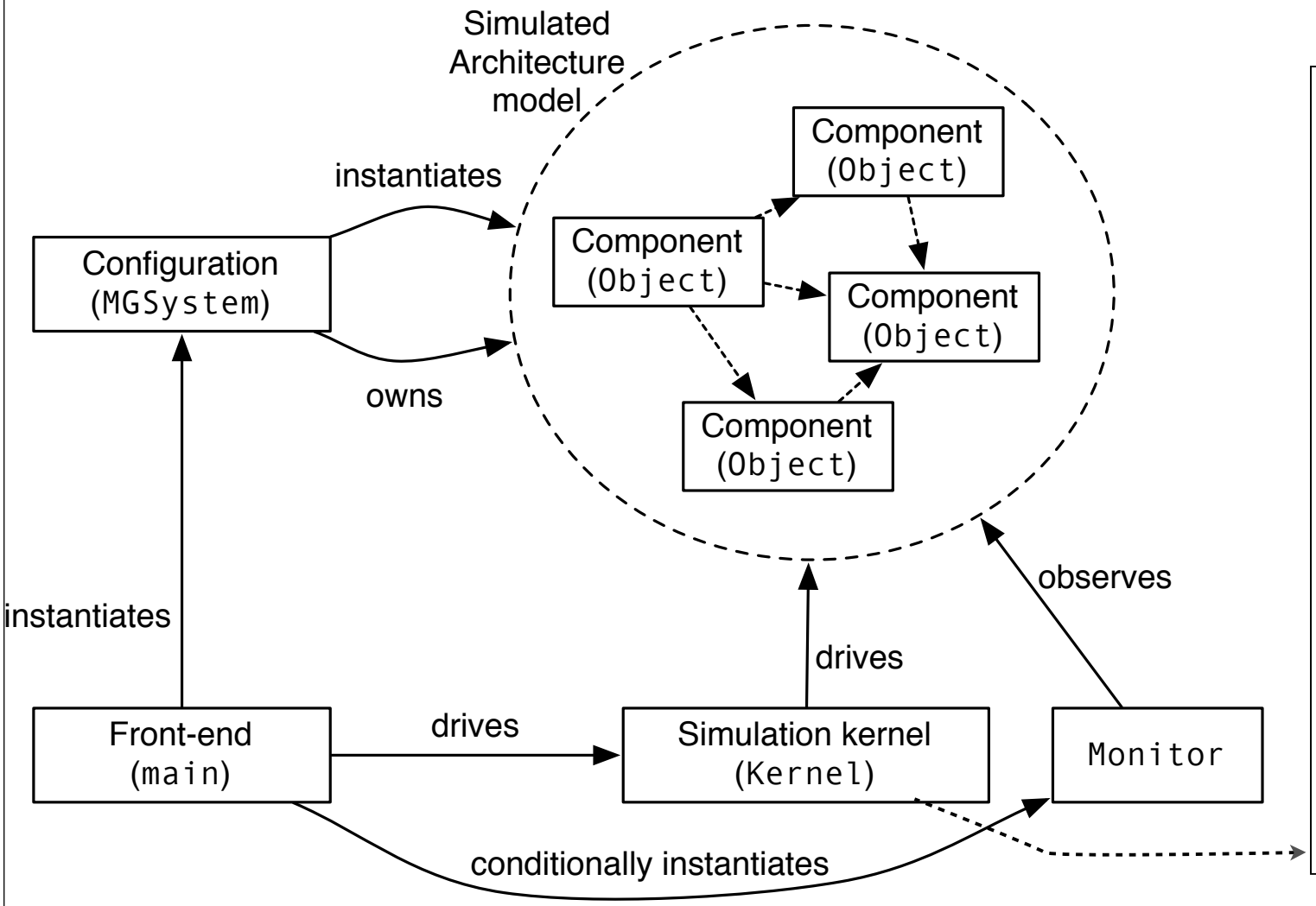
- **Emulators**: implement a **software execution environment** = fully programmable platform
- used to **develop, validate, analyze and debug software**
- **Full-system needed when disrupting the ISA and system topology**

- **Simulators**: implement a **behavior model** for components and systems
- used to **validate system designs** and **predict system behavior**
- **Circuit-level needed when disrupting the micro-architecture or memory protocol**

MGSim: What's in the box?

- Simulation **framework (C++)**
- Library of **example component models**
- Parameterizable **example full system model**
  = "MGSim simulated platform"
- **Trace processing utilities (Python)**
- **Documentation** and (some) support
- As an add-on (sister project):
  C language tool chain + guest OS
  for the MGSim platform

Simulation Framework



3-phase controller:

acquire

shared resources

(arbitrate)

check

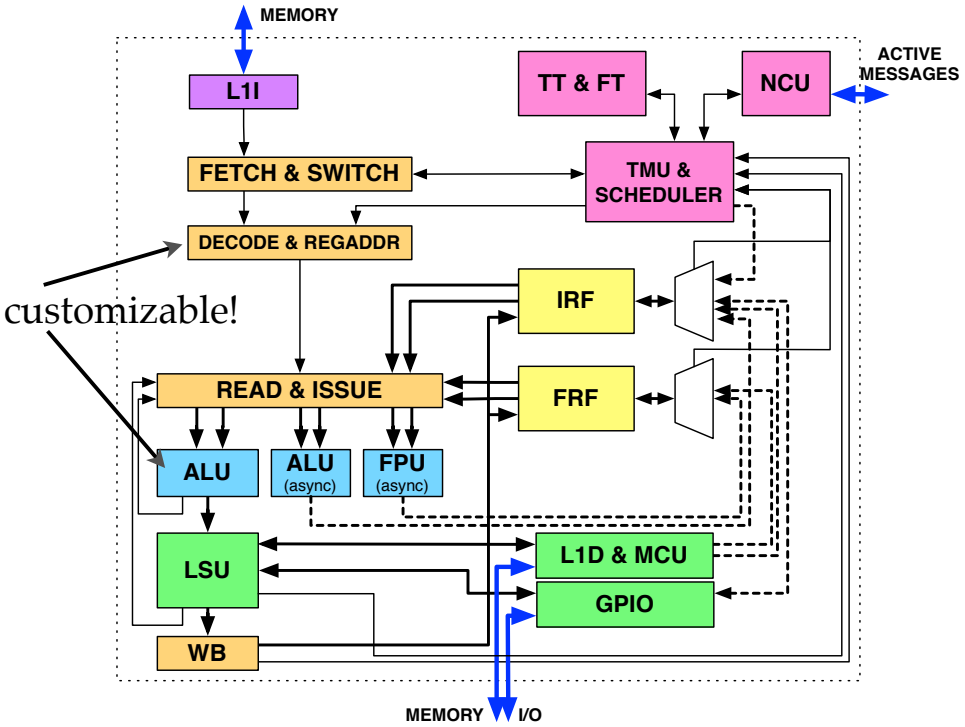
run control logic

commit

state changes

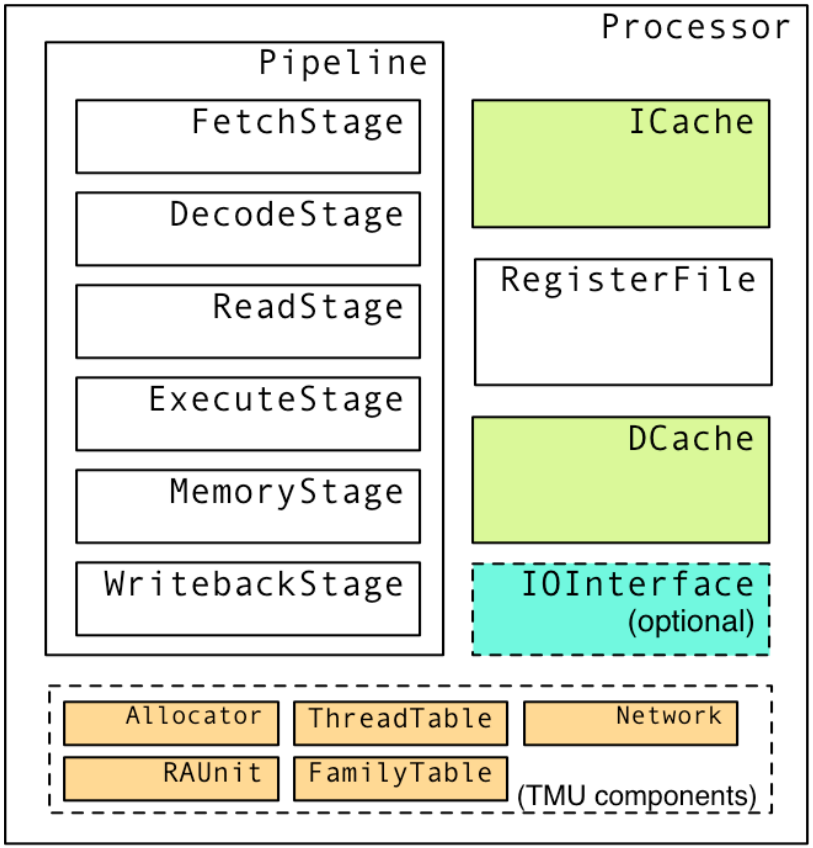
Example models: Apple-core D-RISC

Micro-architecture:

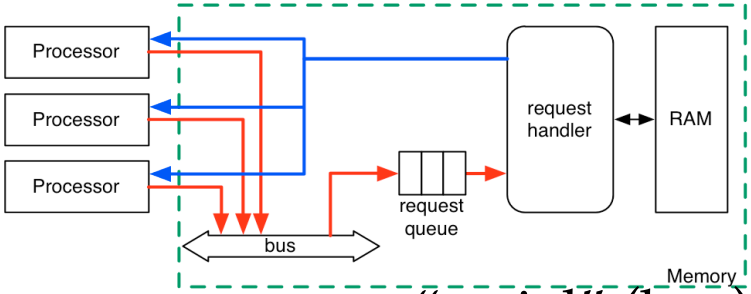


= 6-stage in-order single-issue with **hardware multithreading** & **configurable ISA**

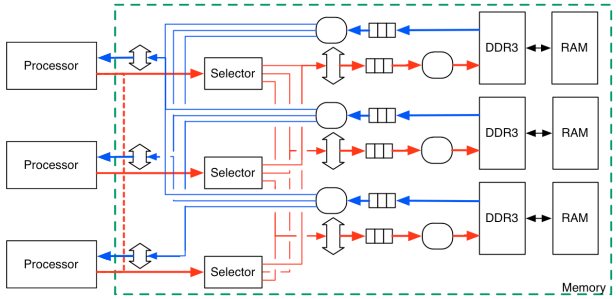
Components in model:



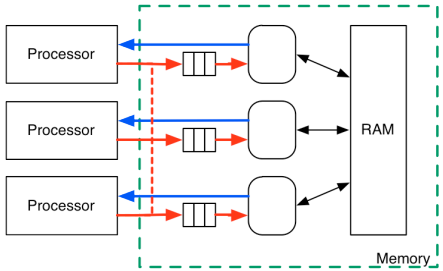
Example models: Apple-CORE memory networks



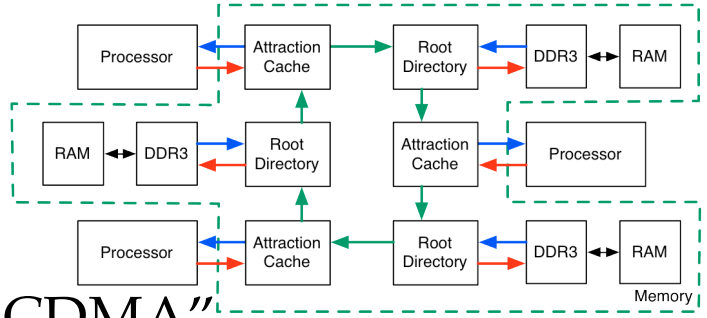
"serial" (bus)



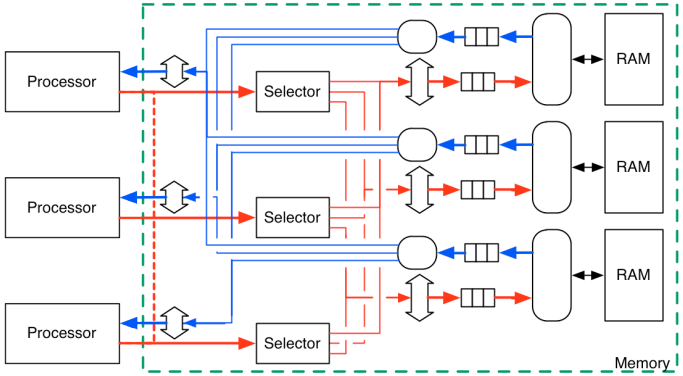
"DDR"



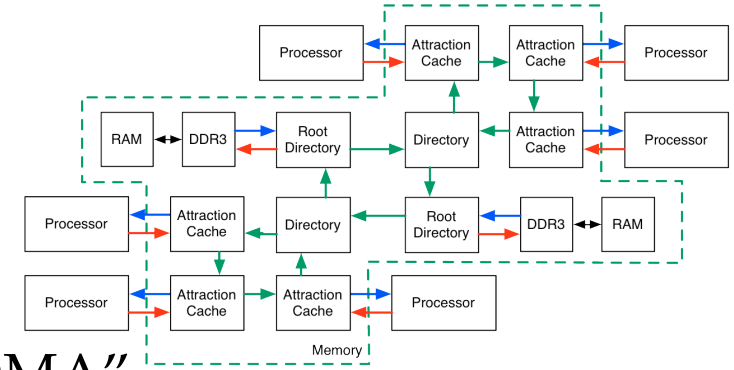
"parallel" (P-RAM)



"flat CDMA"

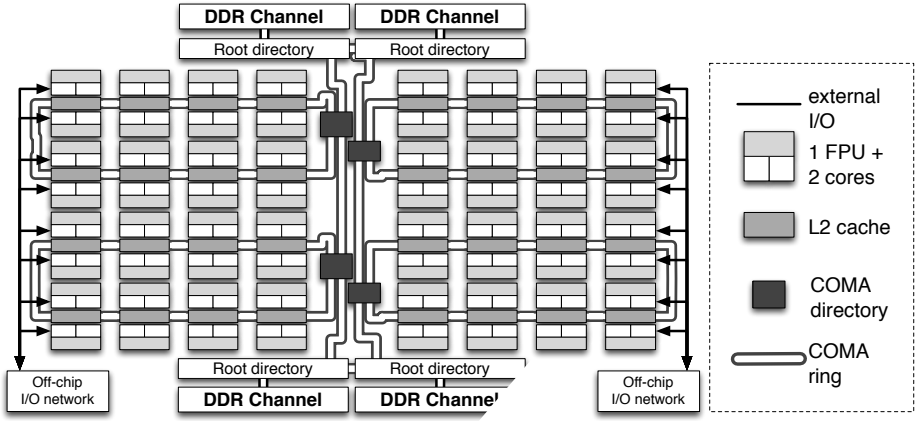


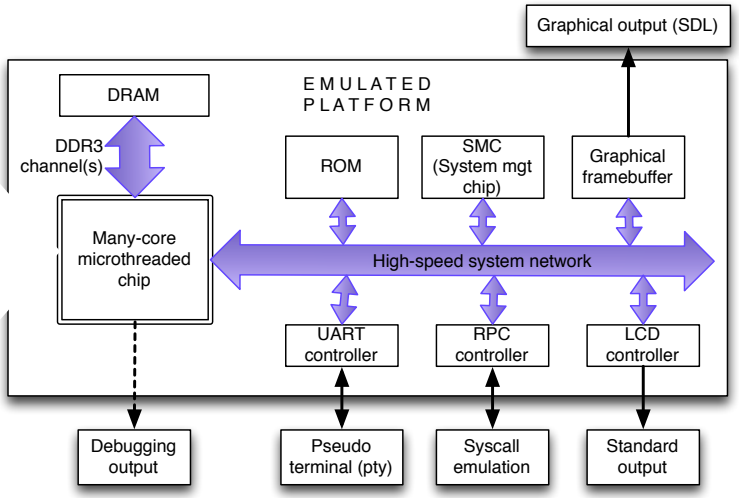
"banked"

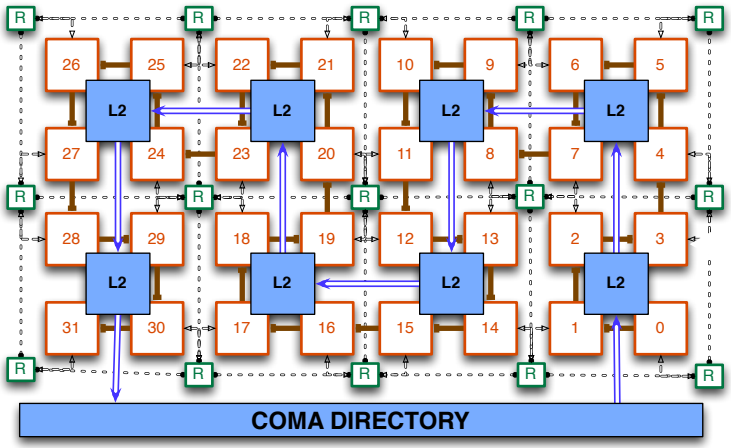


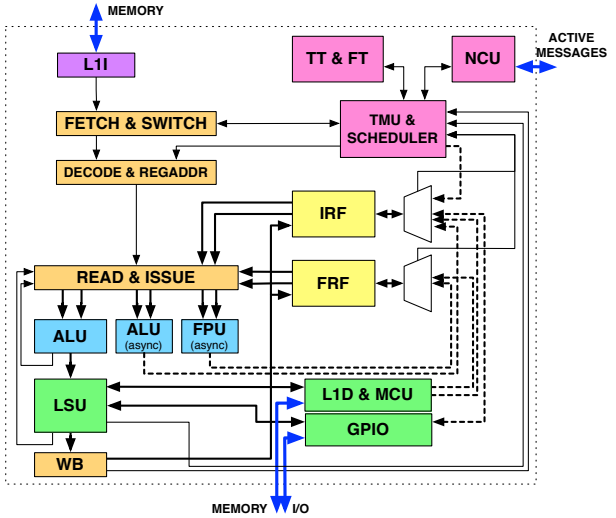
"CDMA"

Example system model: MGSim platform



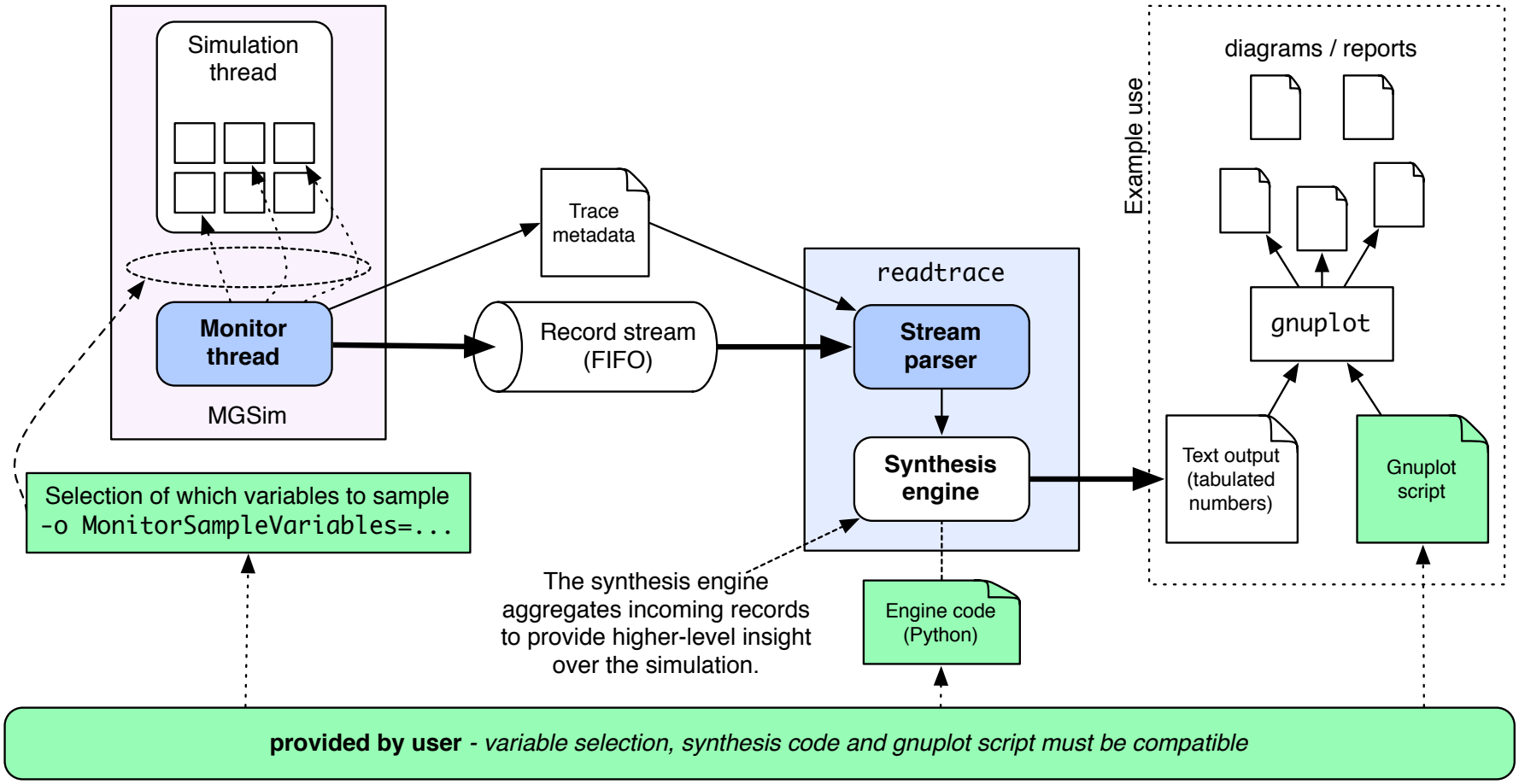




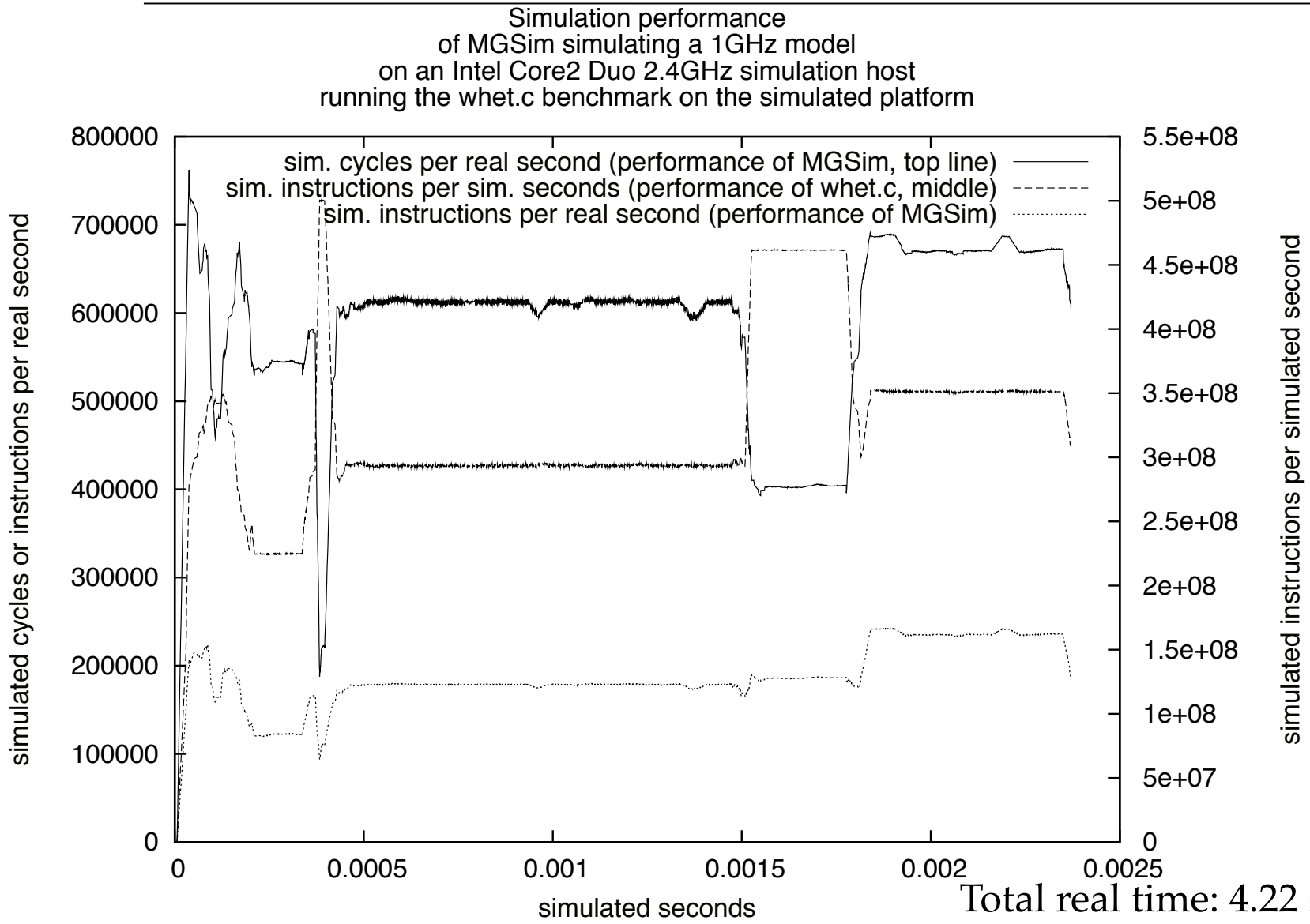


Simulator size: 7111 components, 2669 processes, 62MB on host

Example application: asynchronous tracing

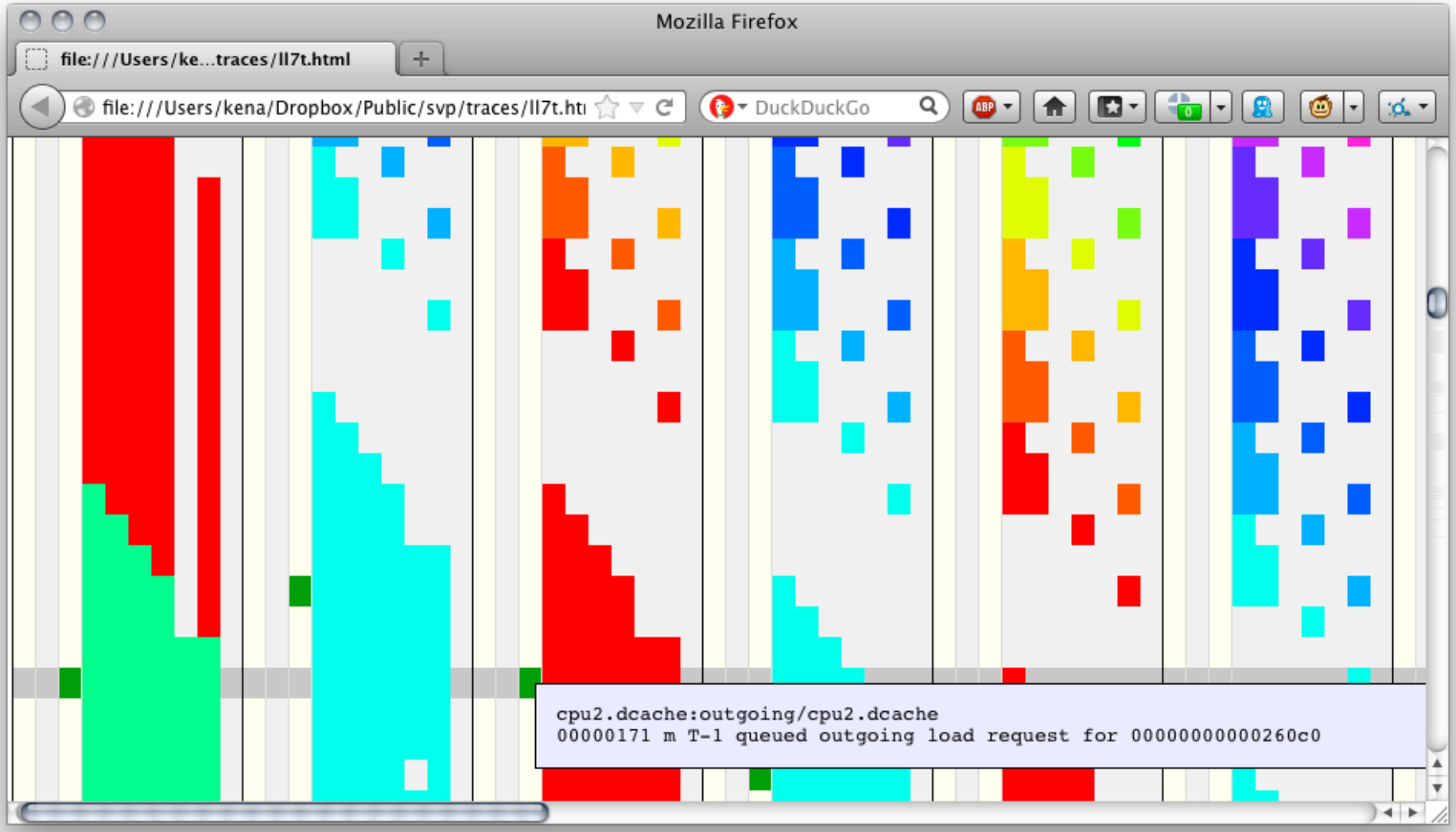


Example application: asynchronous tracing



Total real time: 4.22 seconds

Example application: Trace visualization



MGSim: What's not in the box?

- MGSim should have, but does not have **yet**:
  - Checkpointing
  - Generic mesh NoC model
  - Co-design of MGSim model and RTL spec
- MGSim's project goals do not require:
  - x86 ISA emulation
  - Validated cycle-accurate models of existing hardware → expert knowledge needed to validate MGSim before publications of results

Take away

- At a glance:
  C++, small-ish, cycle-accurate (~RTL),
  full-system via example library, free
- **Teach it!** (designed for education)
- **Use it!** (designed for prototyping)
- **Fork it!** (it's on Github)
- Break it! (we love bug reports)

Thank you.