Introduction to gem5

Mohammad Tahghighi

Outline

- Introduction
  - What is Gem5 useful for ? and what NOT!
- Overview of the system simulator
  - Simulation modes
  - Behind the scene of a simulation
  - What’s under the hood ?

Introduction

- Gem5 is the fusion of two projects
  - GEMS
    - detailed and flexible memory system model
    - Includes support for multiple cache coherence protocols and interconnect models
    - developed @ The University of Wisconsin Madison
  - M5
    - Highly configurable simulation framework to support multiple ISAs, and diverse CPU models
    - developed @ The University of Michigan
- System simulator
  - Good support of complex components interactions (OS / CPU / Caches / Devices / …)
    - Accuracy depends on the model completeness
  - Lot of components available out-of-the-box (CPUs, memories, I/Os, …)

What is Gem5 useful for ?

- Architectural exploration
  - Gem5 provides a fast and easy framework to interconnect hardware components and evaluate them !

- Hardware/software performance evaluation ?
  - Gem5 has a good support of various ISA and allows for realistic HW/SW performance evaluation

What is Gem5 NOT useful for ?

- Hardware/software verification ?
  - RTL functional verification is much more accurate !

- Software development and verification ?
  - Faster technologies are available through binary-translation (e.g. QEMU, OVP)

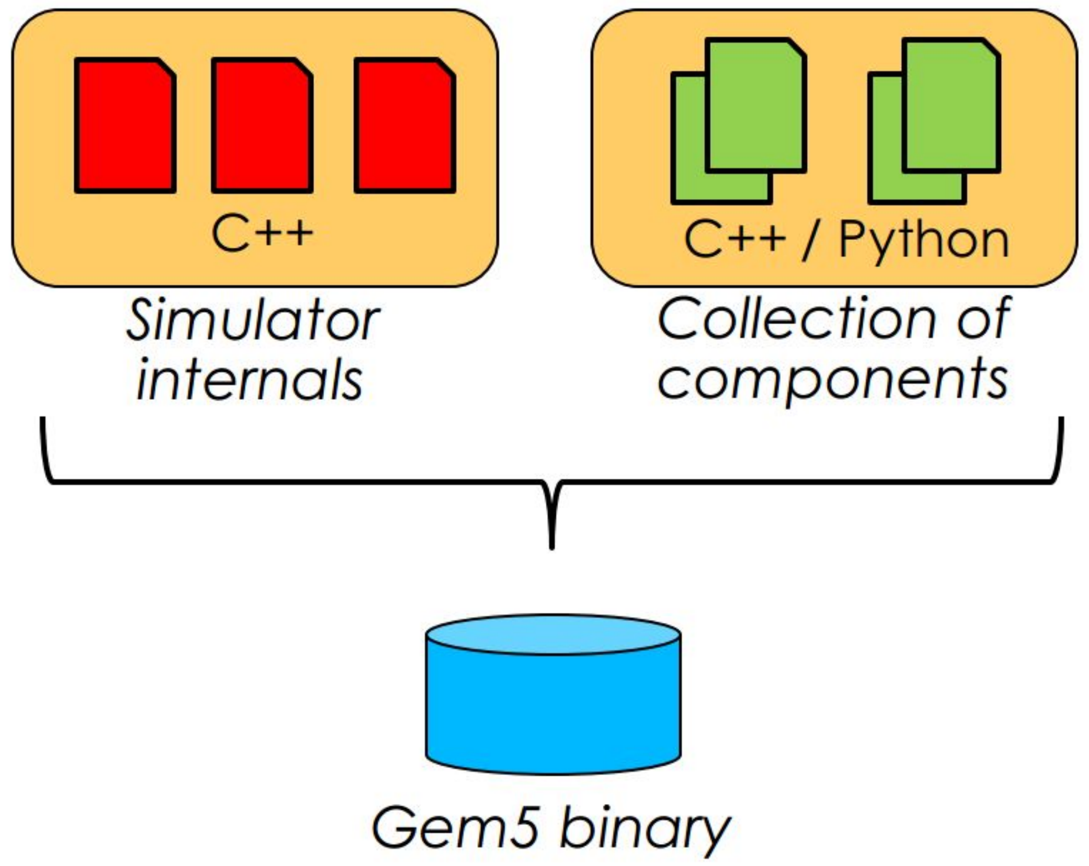
Simulation modes

- Full-system (FS)
  - Models bare-metal hardware
    - Includes the various specified devices, caches, …
  - Boots an entire OS from scratch
    - Gem5 can boot Linux (several variants) or Android out of-the-box

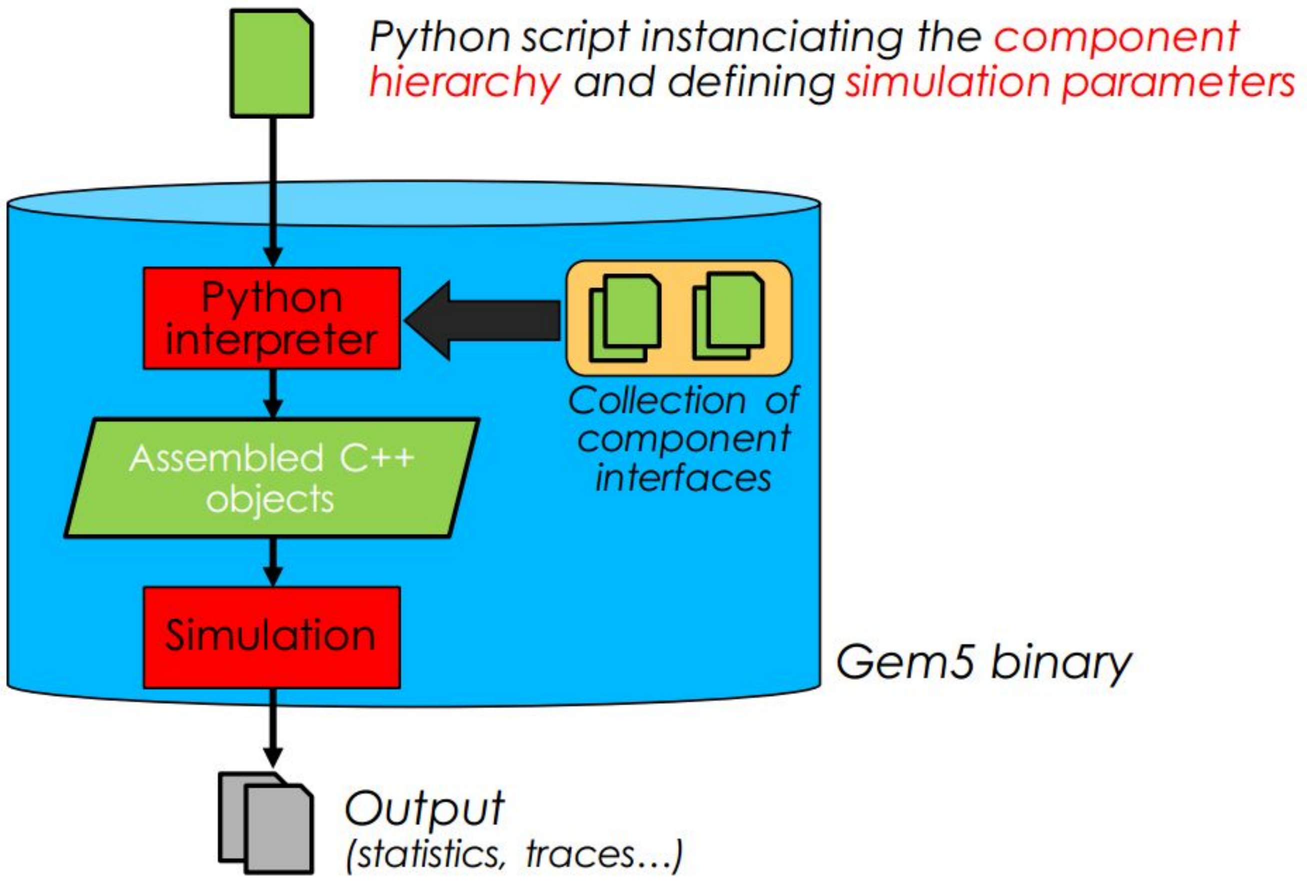
- Syscall Emulation (SE)
  - Runs a single static application
  - System calls are emulated or forwarded to the host OS
  - Lot of simplifications (address translation, scheduling, no pthread …)

Behind the scene of a simulation

Compilation of the simulator

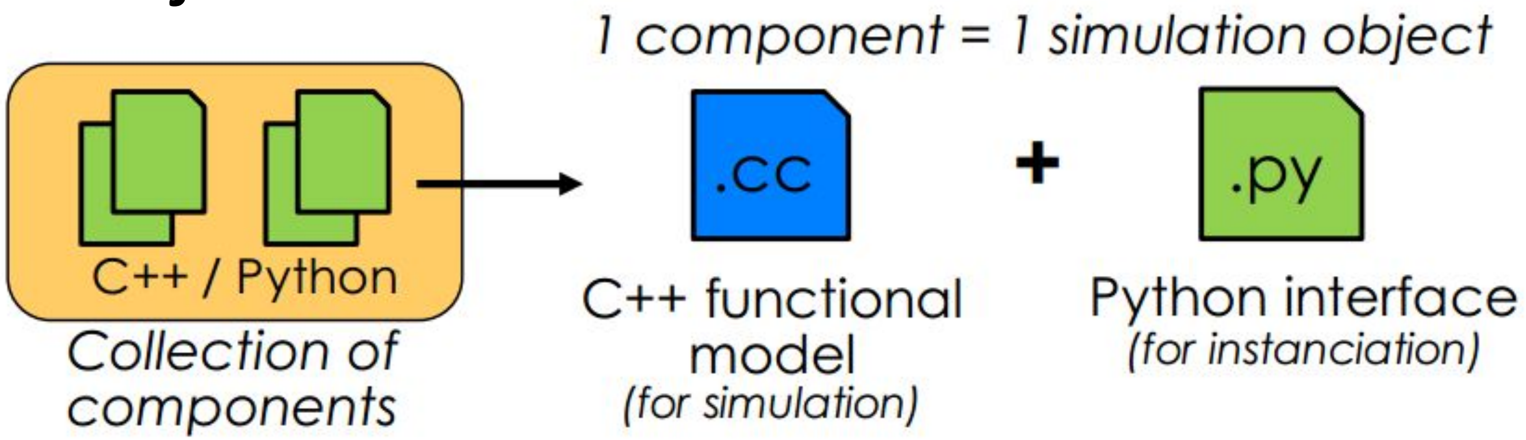


Behind the scene of a simulation
Compilation of the simulator

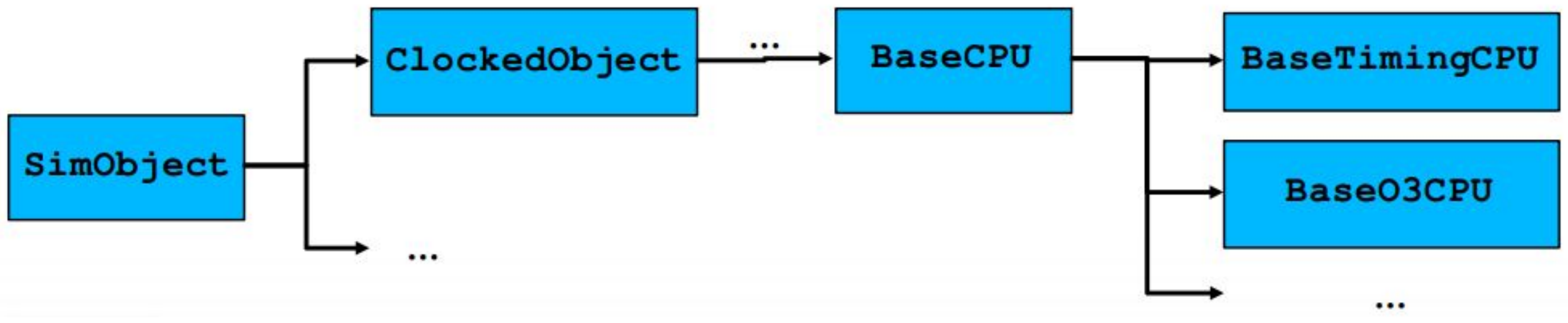

Python script instanciating the component hierarchy and defining simulation parameters
Python interpreter
Collection of component interfaces
Assembled C++ objects
Simulation
Gem5 binary
Output
(statistics, traces…)

What's under the hood ?

Simulation objects



- SimObjects follow a strict C++ class hierarchy for easier extension with code reuse



What’s under the hood ?
Events

- Gem5 is event-driven
    - Discrete event timing model

- Simulation objects schedule events for the next cycle after a specific time elapsed

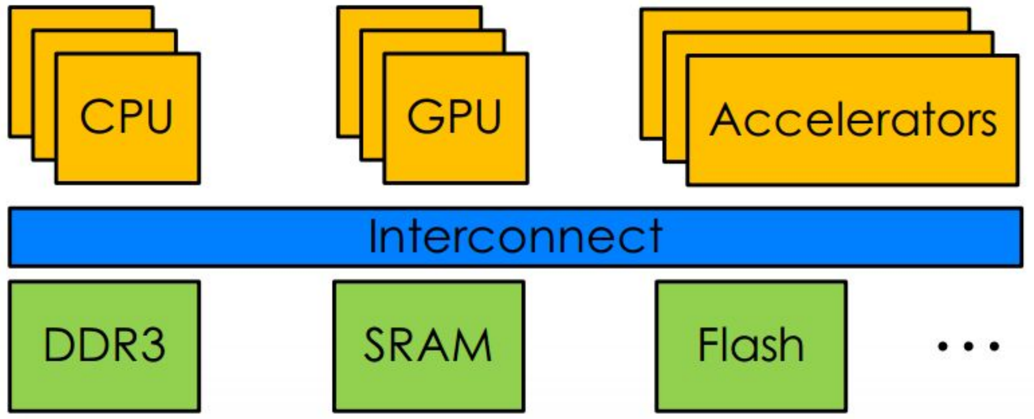
What’s under the hood ?
CPU Models

- Supports: Alpha, ARM, MIPS, Power, SPARC, and x86
- Configurable CPU models : Supports 3 CPU models
  - Simple Atomic/Timing
    - Fast CPU model
  - InOrder
    - Detailed pipelined in-order CPU model
  - O3
    - Detailed pipelined out-of-order CPU model
- Supports a domain specific language to represent ISA details

What’s under the hood ?

Memory System

- Models a system running heterogeneous applications
  - running on heterogeneous processing tiles
  - using heterogeneous memories and interconnect



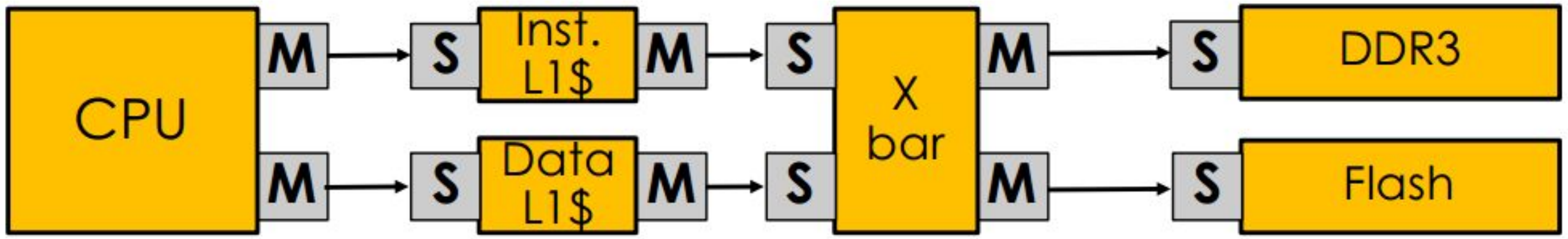
What’s under the hood ?
Memory System

- Two memory systems in Gem5
  - Classic
    - All components instantiated in a hierarchy along with CPUs, etc.
    - Only MOESI coherence protocol
    - Fast, but less detailed than Ruby
  - Ruby
    - Detailed simulation model of various cache hierarchies
    - Various cache coherence protocols (MESI, MOESI, …)
    - Interconnection networks

What's under the hood ?
Memory ports

- Memory ports are present on every MemObject
  - They model physical memory connections
  - You interconnect them during the hierarchy instantiation
    - E.g. a CPU data bus to a L1 cache
    - 1 master port always connects to 1 slave port

- Data is exchanged atomically as packets



What's under the hood ?
Memory ports

- 3 types of transport interfaces for packets
  - Functional
    - Instantaneous in a single function call
    - Caches and memories are updated at once
  - Atomic
    - Instantaneous
    - Approximate latency, but no contention nor delay
  - Timing
    - Transaction split into multiple phases
    - Models all timing in the memory system

Nice feature

- Checkpointing
  - Snapshot the relevant system state and restore it later
- Fast-forward
  - Idea is to start the simulation in atomic mode and switch over to detailed mode for important simulation period

Thanks!