Computer Architecture Project
4x4 Multi-Core Simulation with Gem5

Prakhar Gupta, Harshil Mital, Harsh Popat
2021550, 2021050, 2021048

Explanation of code snippets implemented:

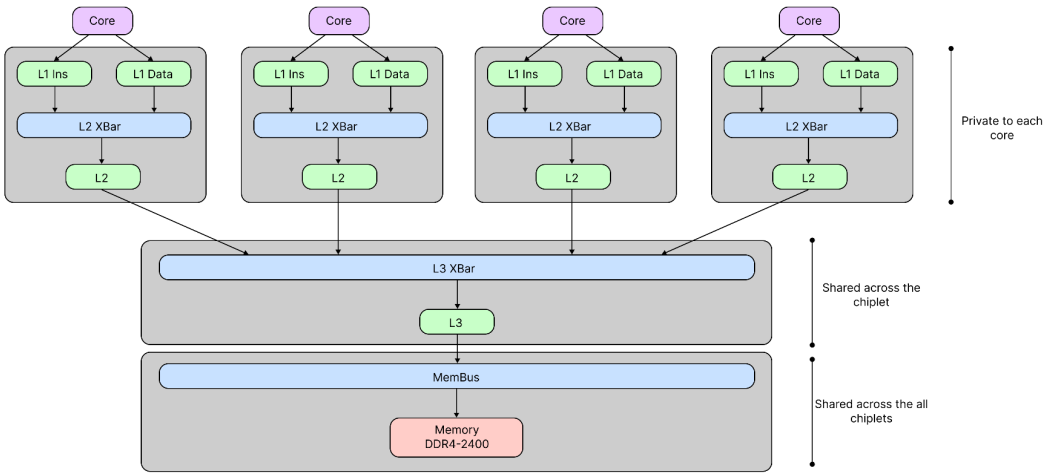
1. **Simple_ruby.py:** This script configures and runs a simple simulation using the gem5 simulator with a RISC-V architecture implementing a 4-core out-of-order processor, a three-level MESI cache hierarchy, and a DDR4 memory subsystem. The script begins by importing necessary gem5 modules and utility functions. It includes components for building the system, such as processors, memory, and cache hierarchies, all part of the gem5 library.
   Cache Hierarchy configures a MESI three-level cache hierarchy, specifying the sizes and associativity of L1 instruction/data caches, L2 cache, and a shared L3 cache.
   The L3 cache is shared among the cores, indicated by num_l3_banks=1. Creates a 4-core, out-of-order RISC-V processor using the SimpleProcessor class from gem5. The ISA is RISC-V, and the processor model is set to be out-of-order (O3), simulating a more realistic CPU behaviour. A 4GiB DDR4 memory system with a clock frequency of 2400 MHz is added, providing the memory interface for the system. The SimpleBoard class combines the processor, cache hierarchy, and memory into a single board/system object. The hello program (from the gem5 test programs) is used as the workload. This binary will run on the simulated RISC-V CPU. The script initializes the simulation environment (m5.instantiate()), prints a message, and starts the simulation (m5.simulate()).

2. **riscv_parsec_benchmarks.py**

   This script sets up a full-system simulation using gem5 to run PARSEC benchmarks on a RISC-V architecture. It uses a dual-core processor with a switchable core type, starting with KVM cores for fast OS boot and switching to Timing cores for detailed benchmarking. The simulation features a two-level MESI cache hierarchy and 3GB of DDR4 memory. The user can specify the benchmark program and input size via command-line arguments.

The script also includes handlers for key simulation events, such as the start and end of the region of interest (ROI) in the benchmark, where relevant statistics are collected. The RISC-V Linux kernel and PARSEC disk image are automatically obtained if not already present. After completing the simulation, key performance metrics such as simulated time and wall clock time are printed.

Architecture Diagram:



Steps to run and compile:

I. To run a demo program with MESI Three Level protocol -

1. Setup gem5 as per the steps available here.
2. cd into gem5's directory.
3. Run the below commands to build gem5 with MESI three-level protocol -

```
scons defconfig build/RISCV_MESI build_opts/RISCV
scons setconfig build/RISCV_MESI
RUBY_PROTOCOL_MESI_THREE_LEVEL=y
scons build/RISCV_MESI/gem5.opt -j 7
```

4. Place the `simple_ruby.py` file in `gem5/configs/learning_gem5/Project/` and run the following command -

```
build/RISCV_MESI/gem5.opt
configs/learning_gem5/Project/simple_ruby.py
```

II. To run the PARSEC benchmark on a RISCV configuration file:

1. Place the `riscv-parsec-benchmark.py` file inside `configs/example/gem5_library` and run the `ferret` suite with the following command-

```
build/RISCV_MESI/gem5.opt \
```

```
configs/example/gem5_library/riscv-parsec-benchmark.py
--benchmark “ferret” \
--size “simsmall”
```