

Ramulator 2.1 Deeper Dive

Haocong Luo

ETH Zurich

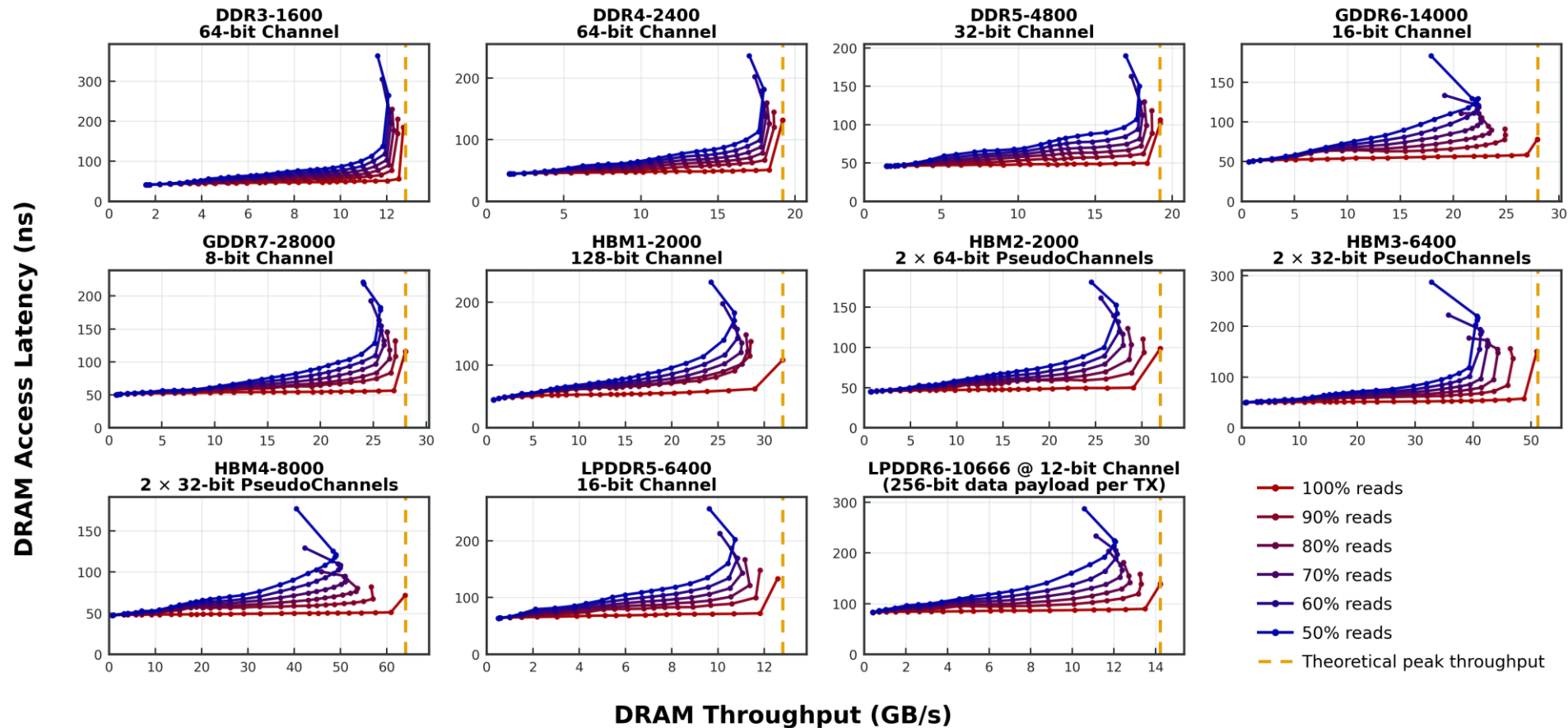
3rd R&DB Workshop and Tutorial @ ICS'26

06.07.2026

Introduction

- **Ramulator 2.1 is a major overhaul of Ramulator 2.0**
 - ❑ **Fixes bugs** identified by Github issues and internal testing
 - ❑ More comprehensive support for **new DRAM & controller features**
 - ❑ More comprehensive set of **tests and validations**
 - ❑ Significantly improved **ease of use, configuration, and extension**
 - ❑ Overall code quality and documentation improvements
- Github main branch updated to v2.1.0
 - ❑ <https://github.com/CMU-SAFARI/ramulator2/>
 - ❑ codespace and devcontainer ready!
 - ❑ V2.0a is archived as legacy/v2.0a
- **Welcomes your use, contribution, and feedback!**
 - ❑ Issues & Discussions on Github
 - ❑ Open PRs for bug fixes, improvements, and new features

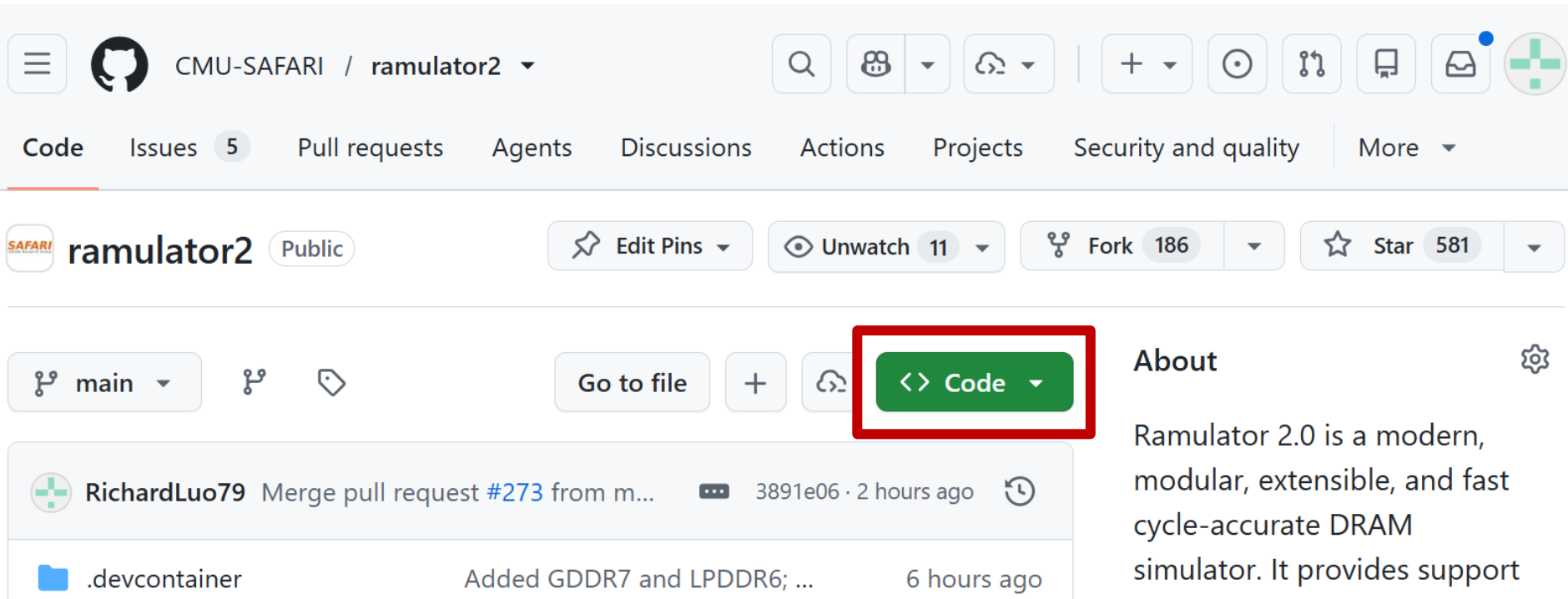
Currently Supported DRAM Standards



Getting Started (I)

■ Github codespace

□ <https://github.com/CMU-SAFARI/ramulator2>



The screenshot shows the GitHub repository page for CMU-SAFARI/ramulator2. The repository is public and has 581 stars, 186 forks, and 11 watchers. The 'Code' button is highlighted with a red box. Below the repository name, there are buttons for 'main', 'Go to file', '+', and 'Code'. The 'Code' button is a green button with a code icon and a dropdown arrow. Below the repository name, there is a list of recent pull requests and commits. The first pull request is by RichardLuo79, titled 'Merge pull request #273 from m...', and the commit is 'Added GDDR7 and LPDDR6; ...'.

CMU-SAFARI / ramulator2

Code Issues 5 Pull requests Agents Discussions Actions Projects Security and quality More

ramulator2 Public

Edit Pins Unwatch 11 Fork 186 Star 581

main Go to file + <> Code

RichardLuo79 Merge pull request #273 from m... 3891e06 · 2 hours ago

.devcontainer Added GDDR7 and LPDDR6; ... 6 hours ago

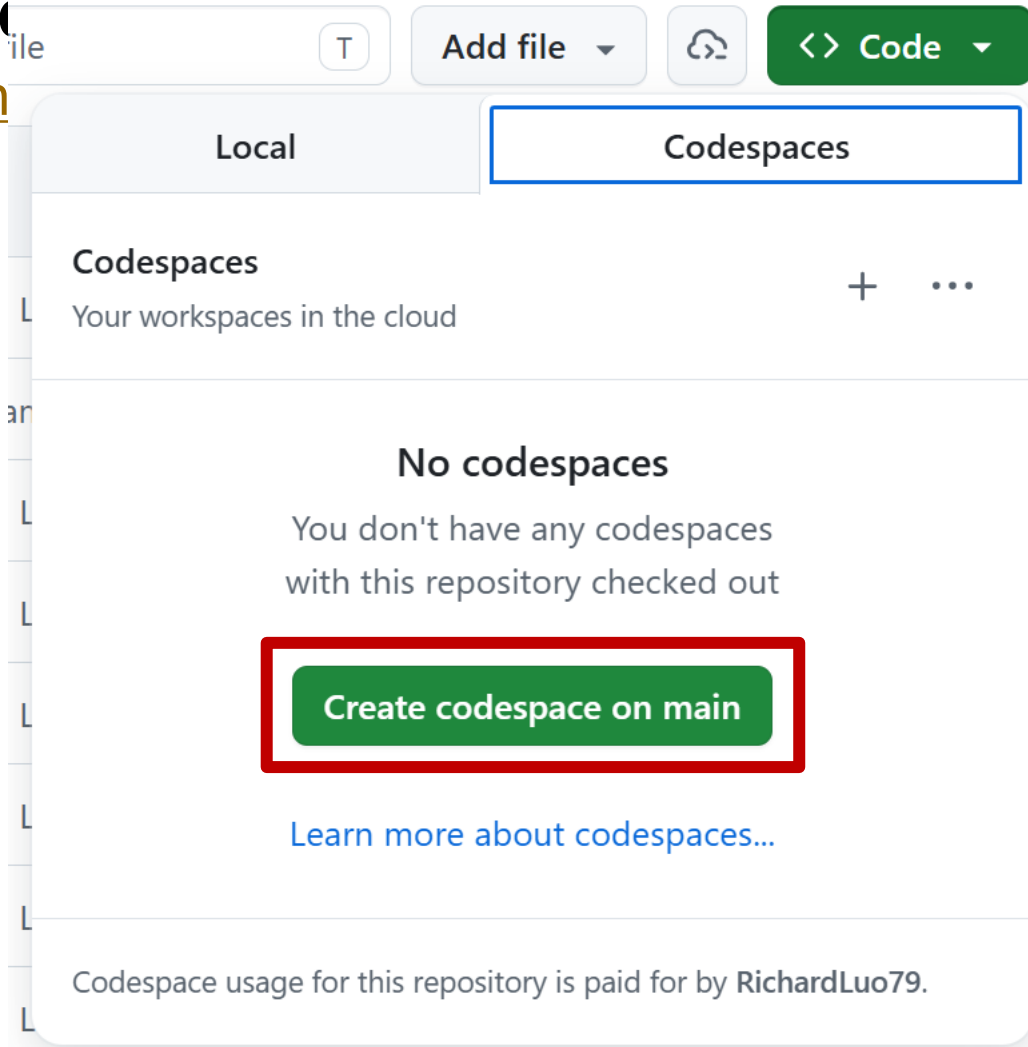
About

Ramulator 2.0 is a modern, modular, extensible, and fast cycle-accurate DRAM simulator. It provides support

Getting Started (I)

■ Github codespaces

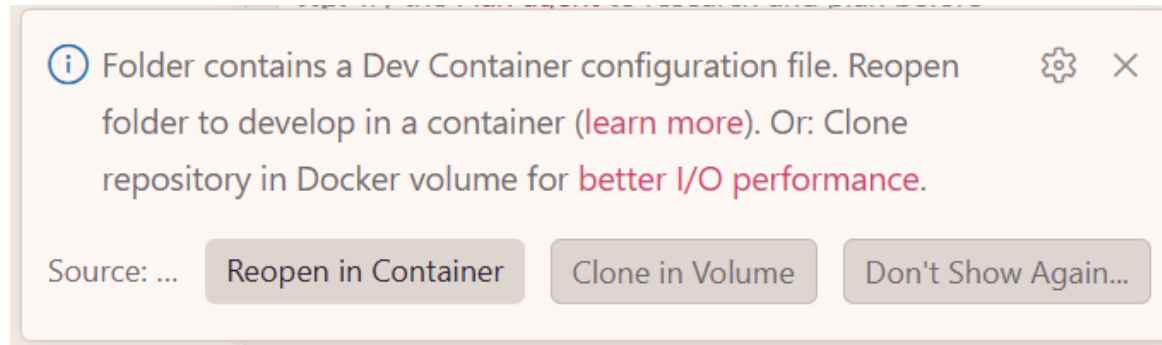
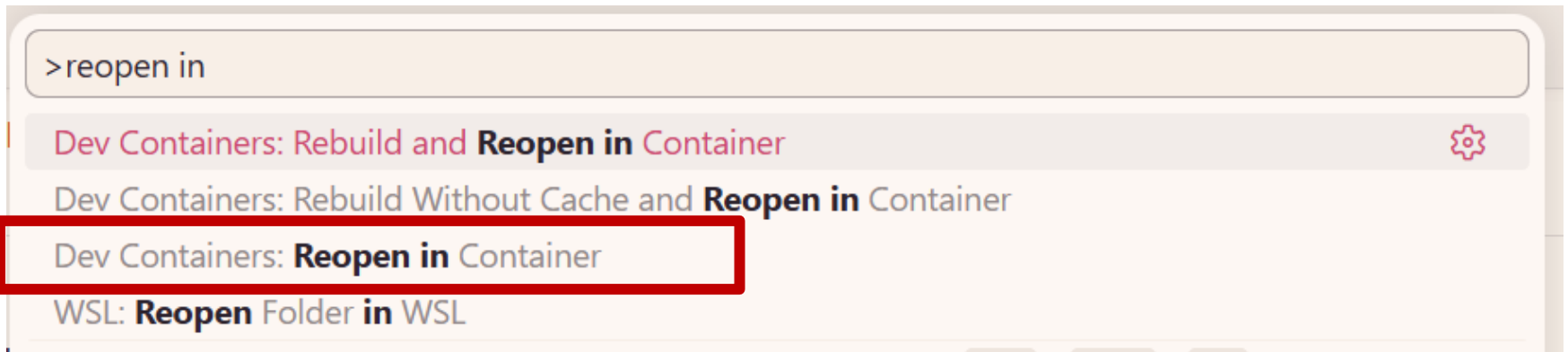
□ [https://github](https://github.com)



Getting Started (II)

■ Local (devcontainer)

- ❑ Clone the repository
- ❑ Open with VSCode



Getting Started (III)

■ Local

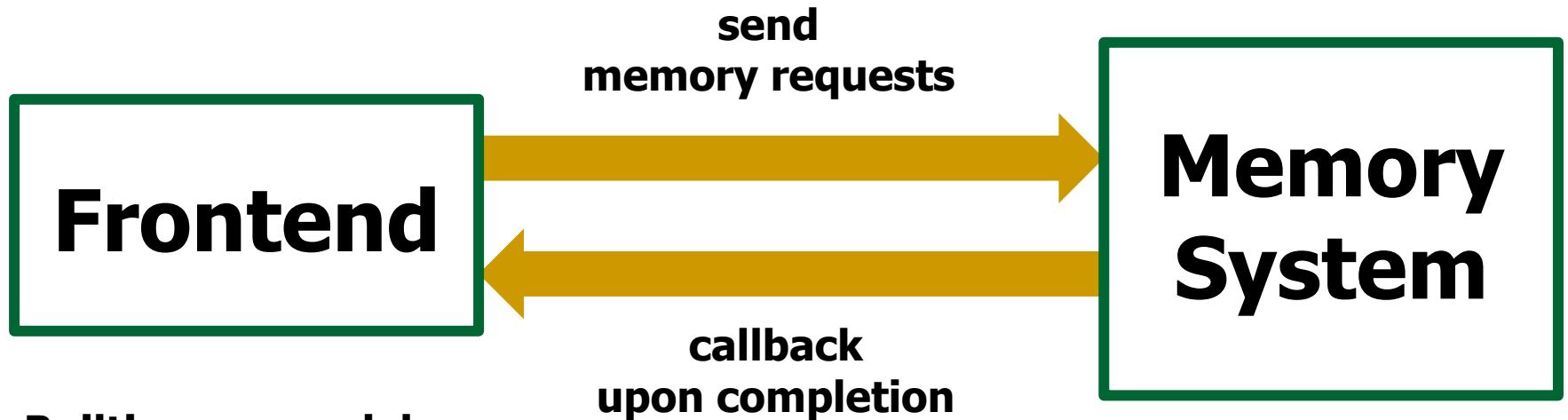
- ❑ .devcontainer/Dockerfile
- ❑ .devcontainer/post-create.sh

Navigating the Source Tree

- examples/ # example configs, traces
- python/ramulator/
 - <auto-generated python wrappers>
 - dram/ # source code for DRAM standards
- resources/
 - gem5_wrappers/
- src/ramulator/
 - base/ # sw framework of Ramulator 2.1
 - controller/ # source code of the mem. ctrl. Interface
 - i_controller.h # interface source
 - impl/ # implementation source
 - hbm34_controller.cpp
 - ...
 - dram/ # auto-generated C++ source of DRAM device
 - ...
- tests/

Simulator Architecture (I)

■ Two top-level components

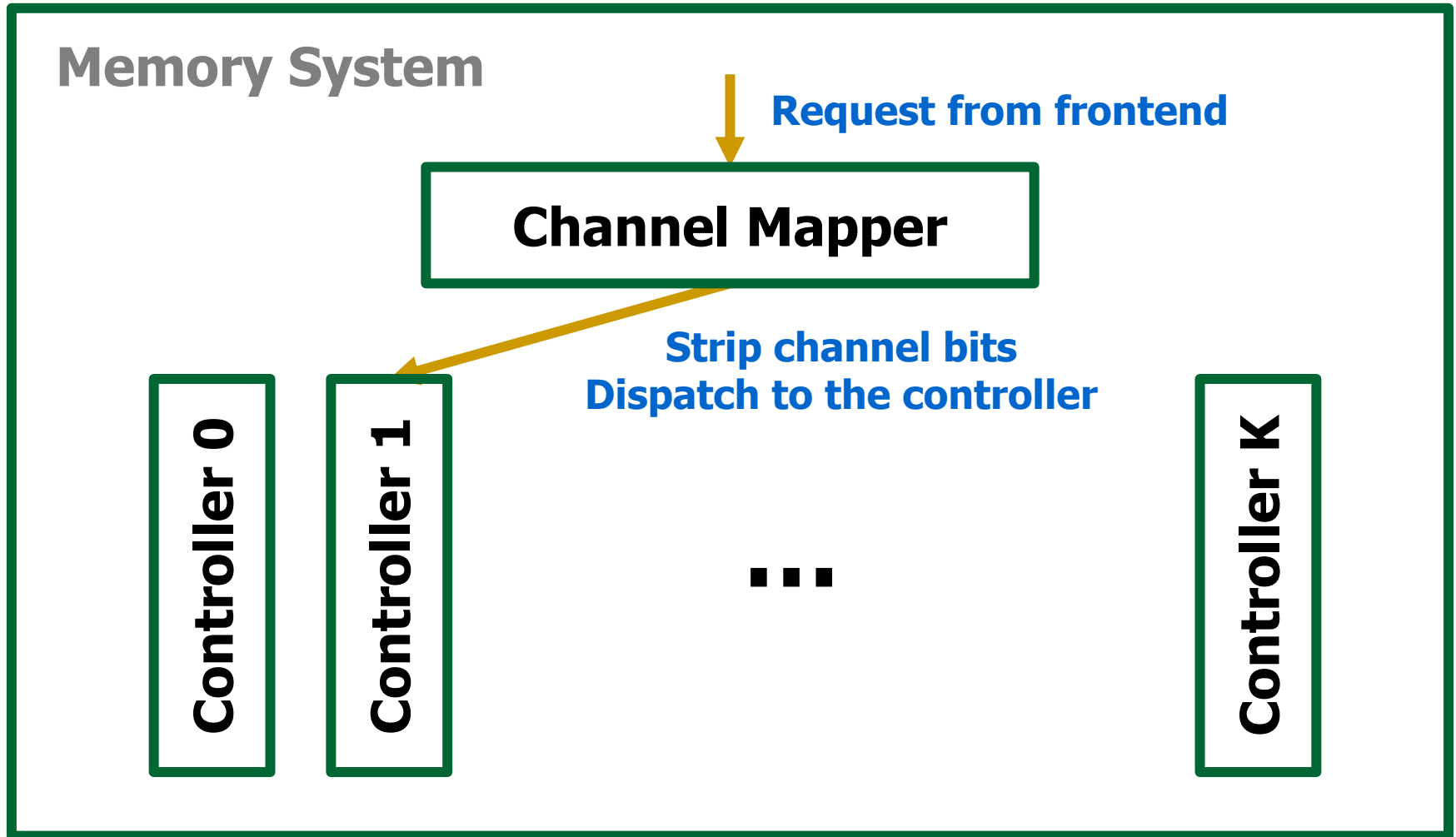


- **Builtin core model**
- **Memory trace parser**
- **Wrappers for other simulators**

- **1 or more memory controllers**

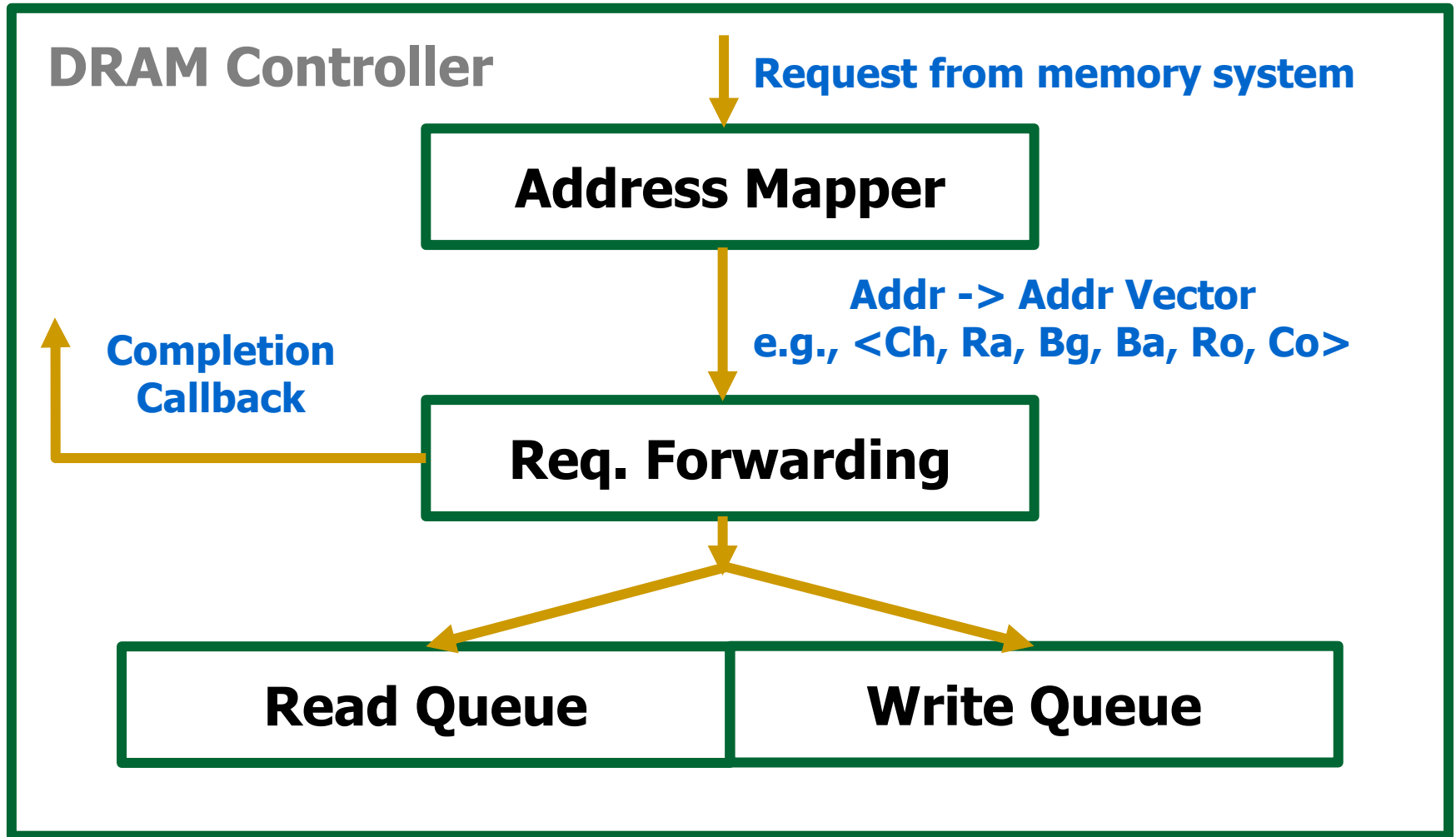
Simulator Architecture (II)

■ Inside the Memory System



Simulator Architecture (III)

■ Inside the Memory Controller



Simulator Architecture (IV)

■ Inside the Scheduler

```
void GenericDDRController::tick() {  
    // Common bookkeeping: clk advance, req queue stats update, completed reads draining  
    tick_prologue();  
  
    // We give refresh requests high priority in the same tick  
    m_refresh->tick();  
  
    // Pre-schedule hooks  
    m_rowpolicy->pre_schedule(); // e.g., CloseRow policy may inject PREpb here  
    for (auto* p : m_plugins) {  
        p->pre_schedule();  
    }  
}
```

Simulator Architecture (IV)

■ Inside the Scheduler

```
// Try to find a candidate request to schedule  
// Priority: active > priority > read/write  
// 1. Try to schedule from active  
Candidate cand = pick_best_ready_from(m_active_buffer, {});  
  
// 2. If no candidate found, try to schedule from priority  
if (!cand.valid) {  
    cand = pick_priority_if();  
}  
  
// 3. If no candidate found, try to schedule from read/write (with write mode check)  
if (!cand.valid && m_priority_buffer.size() == 0) {  
    cand = pick_rw_if();  
}
```

Simulator Architecture (IV)

■ Inside the Scheduler

```
// We have a valid request to serve this cycle
if (cand.valid) {
    // Rowpolicy *may* upgrade the command to AutoPrecharge version
    m_rowpolicy->try_upgrade_command(*cand.it);

    if (!cand.it->is_stat_updated) {
        | update_request_stats(cand.it);
    }

    // Issue command to DRAM device
    m_device.issue_command(cand.it->command, cand.it->addr_vec, m_clk);

    // Notify row policy and plugins of the issued command
    m_rowpolicy->on_issue(*cand.it);
    for (auto* p : m_plugins) {
        | p->on_issue(*cand.it);
    }

    // Advance request
    if (cand.it->command == cand.it->final_command) {
        | retire_request(cand.it, *cand.buffer);
    } else if (m_device.m_spec->command_meta[cand.it->command].is_opening) {
        | promote_to_active(cand.it, *cand.buffer);
    }
}
```

Simulator Architecture (IV)

■ Inside the Scheduler

```
// Post-schedule hooks  
m_rowpolicy->post_schedule();  
for (auto* p : m_plugins) {  
    p->post_schedule();  
}  
}
```

Simulator Architecture (IV)

■ Inside the Scheduler

```
ControllerBase::Candidate ControllerBase::pick_best_ready_from(
    ReqBuffer& buffer,
    RequestFilterRef filter) {
    Candidate c;
    // Scheduler contract: get_best_request() derives req.command before applying
    // filter, so predicates can inspect the current command directly.
    auto it = m_scheduler->get_best_request(buffer, filter);
    if (it == buffer.end()) {
        return c;
    }
    if (!check_timing(it->command, it->addr_vec)) {
        return c;
    }
    c.valid = true;
    c.it = it;
    c.buffer = &buffer;
    return c;
}
```

Simulator Architecture (IV)

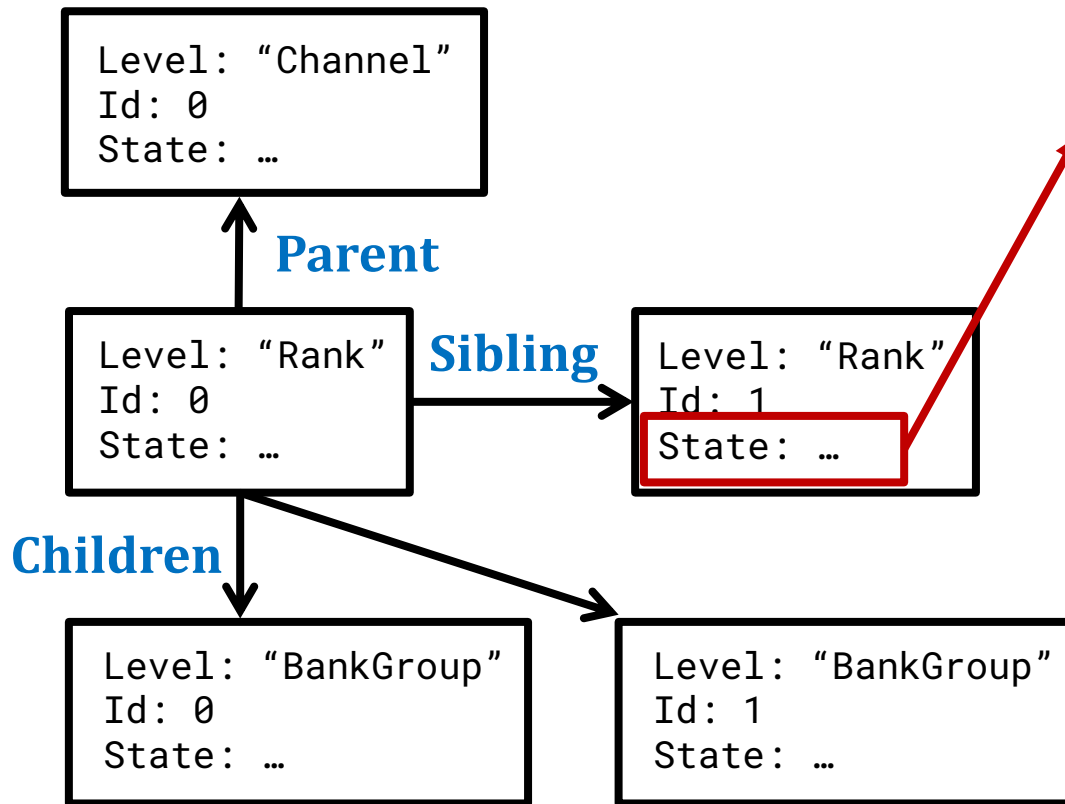
■ Inside `issue_command()`

```
void DRAMDevice::issue_command(int command, const AddrVec_t& addr_vec, Clk_t clk) {  
    m_root->update_timing(command, addr_vec, clk);  
    apply_action(command, addr_vec, clk);  
}
```

- ❑ Recursively walks the tree hierarchy of DRAM nodes
- ❑ Update timings along the way
 - Update the earliest ready cycle of following commands based on timing constraints
- ❑ Apply the action that the issued command has on involved nodes
 - Node state changes

Simulator Architecture (V)

■ Hierarchical state-machine based modeling of DRAM



State of a DRAM node:

- ❑ **Current state:** open, close, activating, etc.
- ❑ **Timing constraints:** Earliest time in the future that each DRAM command is allowed to be issued
- ❑ **Can be extended to include more:**
 - **Energy & power:** Time spent in each state and the number of DRAM commands served (DRAMPower model)

Simulator Architecture (VI)

- DRAM commands implemented as lambda functions that **hierarchically traverses and updates the states** of the nodes
 1. Checks the current states of the nodes to **decode which DRAM command to issue**
 2. **Programmatically** apply state changes
 3. Updates the **timing constraints, power metrics**, etc.
- **Templated** library of generalized DRAM command lambda functions across different architectures

```
template <class DRAM_t>
int RequireAllBanksClosed(typename DRAM_t::Node* node, actions
    int cmd, int target_id, Clk_t clk) {
    // for all banks {
    // ...
    if (bank->m_state == DRAM_t::m_states["Closed"]) {
        continue;
    } else {
        return T::m_commands["PREab"];
    }
    // }
    return cmd;
};
```

Example Function

Contribute to Ramulator 2.1 (II)

■ Example directions of contributions

□ Better timing presets of DRAM standards

- Some of the DRAM timings are our best effort “guesstimates”
- We welcome your more accurate set of timings w/o breaking NDA

□ More DRAM request scheduling techniques

□ More DRAM read disturbance mitigation techniques

- Make use of controller plugins and scheduler filters

□ More intelligent scheduling of refresh

- Refresh postponing & pull-in
- Dynamic choice of REFab/sb/pb

□ PUD/PNM modeling

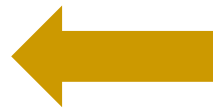
□ More integration with other simulators

■ Please open an Issue/Discussion first to coordinate with potential internal development!

Acknowledgements

■ Contributors (growing)

- ❑ Haocong Luo
- ❑ Nisa Bostanci
- ❑ Ataberk Olgun
- ❑ Maria Makeenkova
- ❑ Ziad Malik
- ❑ Ipek Akdeniz
- ❑ Kislay Arya
- ❑ Roman Hoffmann
- ❑ Wonhyuk Yang
- ❑ @jiegec
- ❑ Julien Eudine
- ❑ Sangjae Park
- ❑ ...



Join the list!