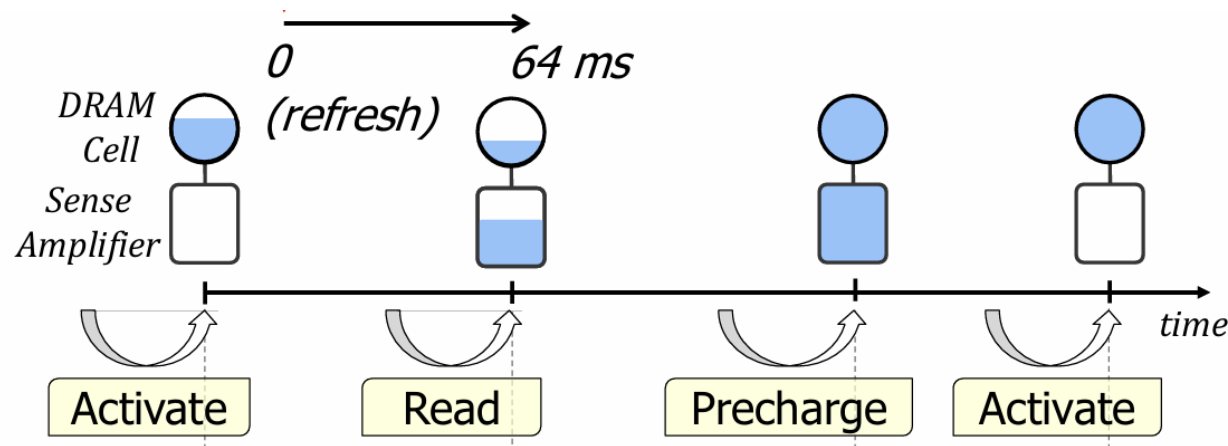

Ramulator 2.0 Summary

***Intelligent System
Laboratory***

DRAM Operations & States

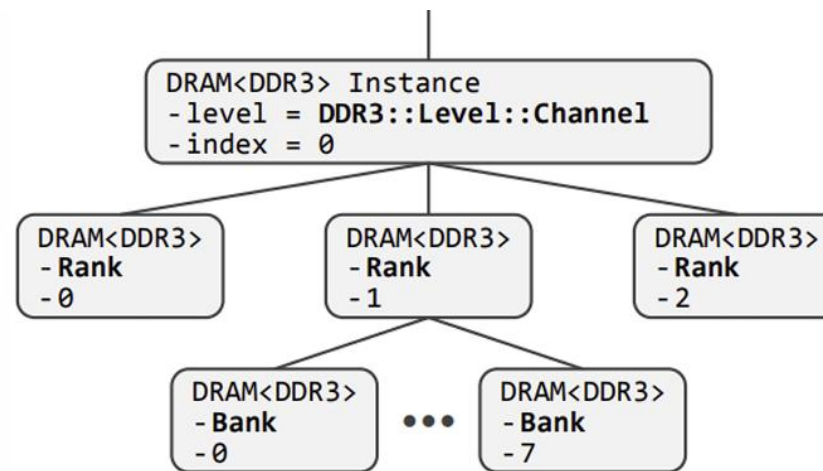
□ DRAM Operations & States



```
1 // DRAM.h
2 template <typename T>
3 class DRAM {
4     DRAM<T>* parent;
5     vector<DRAM<T>*>
6         children;
7     T::Level level;
8     int index;
9     // more code...
10 };
```

```
1 // DDR3.h/cpp
2 class DDR3 {
3     enum class Level {
4         Channel, Rank, Bank,
5         Row, Column, MAX
6     };
7
8     // more code...
9
10 };
```

- Main DRAM states
 - Activate
 - Read
 - Precharge



src files \Leftrightarrow DRAM Operation

□ Simulation Configuration

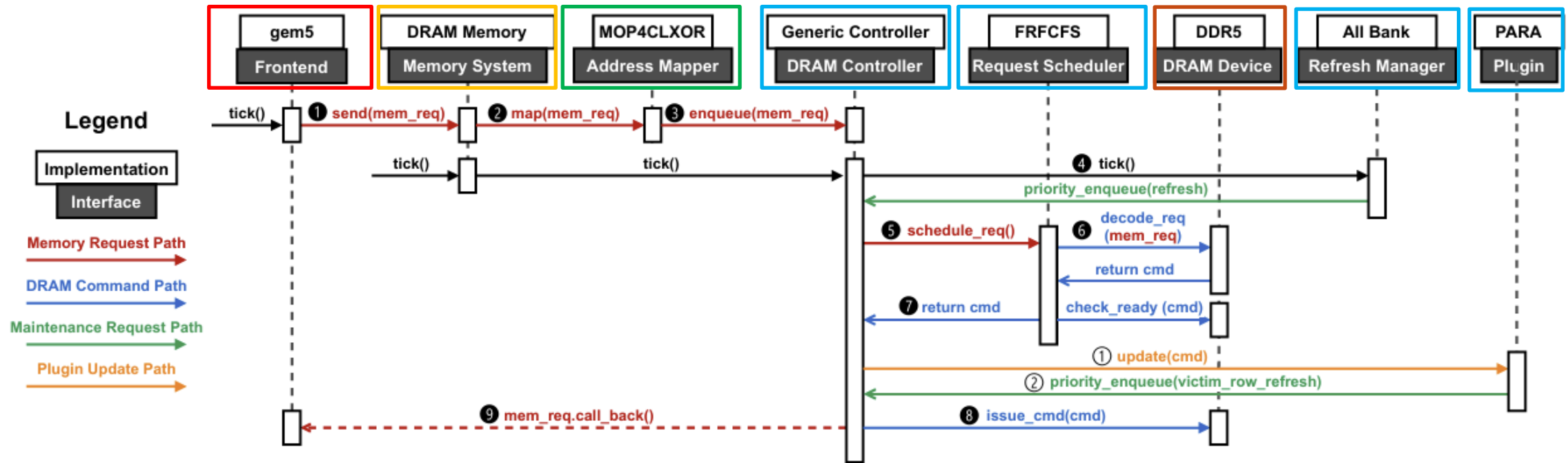
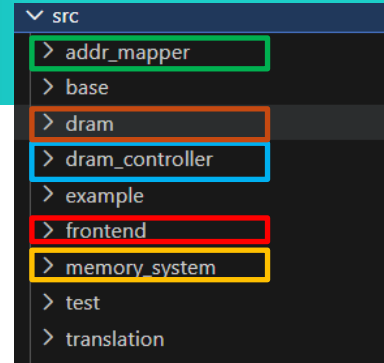


Fig. 1: High-level software architecture of Ramulator 2.0 using an example DDR5 system configuration

src files <=> DRAM Operation

□ Simulation Configuration

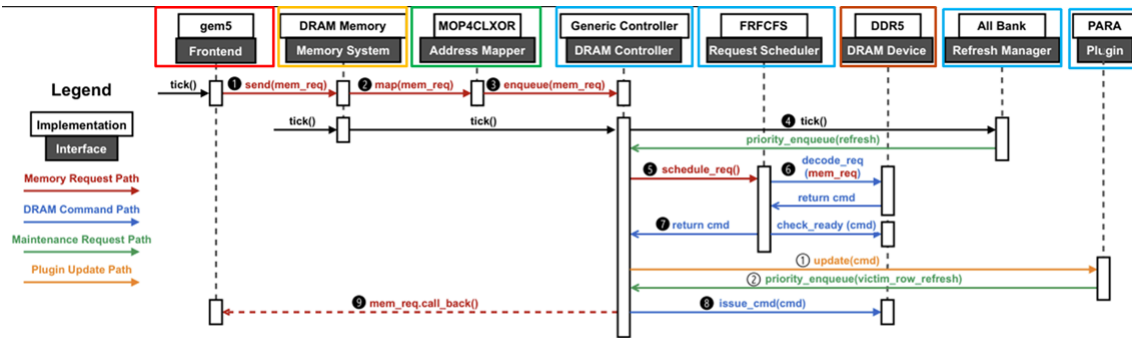
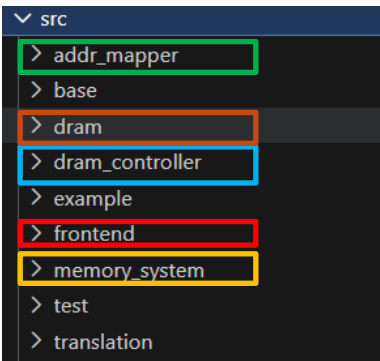


Fig. 1: High-level software architecture of Ramulator 2.0 using an example DDR5 system configuration



1. **Requests are sent:** Front-end(trace file)에서 Memory Request를 보냄
2. **Memory Addresses are Mapped:** Address Mapper가 Request Address를 DRAM 구조에 맞게 변환
3. **Enqueue:** DRAM Ctrlr의 Buffer에 Request를 넣음
4. **DRAM Ctrlr - Ticking Refresh Manager:** Ctrlr가 Refresh Manager를 호출해 high-priority maintenance request(ex. Refresh)을 추가
5. **DRAM Ctrlr - Request Scheduling:** Request Scheduler에게 최적의 Request를 선택하라고 요청
6. **DRAM Device가 Request 확인:** Scheduler가 DRAM Device Model을 참조해 적합한 Command를 Decode
7. **Issue Command:** DRAM Ctrlr가 DRAM Command를 보냄
8. **Updates the behavior and timing information:** DRAM Command Issue시 State & Timing이 Update
9. **Notify the frontend:** Memory Request가 끝나면 callback으로 frontend에 알림

main function

□ main.cpp

```
13 ~ int main(int argc, char* argv[]) {
14     // Parse command line arguments
15     argparse::ArgumentParser program("Ramulator", "2.0");
16     program.add_argument("-c", "--config").metavar("\ndumped YAML configuration")
17     .help("String dump of the yaml configuration.");
18     program.add_argument("-f", "--config_file").metavar("path-to-configuration-file")
19     .help("Path to a YAML configuration file.");
20     program.add_argument("-p", "--param").metavar("KEY=VALUE")
21     .append()
22     .help("Specify parameter to override in the configuration file. Repeat this option to change multiple parameters.");
```

⋮

```
88     // Connect the frontend and the memory system together,
89     // this recursively calls the "setup" function in all instantiated components
90     // so that they can get each other's parameters (if needed) after their initialization
91     frontend->connect_memory_system(memory_system);
92     memory_system->connect_frontend(frontend);
93
94     // Get the relative clock ratio between the frontend and memory system
95     int frontend_tick = frontend->get_clock_ratio();
96     int mem_tick = memory_system->get_clock_ratio();
97
98     int tick_mult = frontend_tick * mem_tick;
99
100     for (uint64_t i = 0; i < 1000000; i++) {
101         if ((i % tick_mult) % mem_tick == 0) {
102             frontend->tick();
103         }
104
105         if (frontend->is_finished()) {
106             break;
107         }
108
109         if ((i % tick_mult) % frontend_tick == 0) {
110             memory_system->tick();
111         }
112     }
113
114     // Finalize the simulation. Recursively print all statistics from all components
115     frontend->finalize();
116     memory_system->finalize();
117
118     return 0;
119 }
```

main.cpp

1. Argument 받는 부분

- Options

1. -c: commandline dump
2. -f: YAML document
3. -p: overriding parameters in a YAML document

2. Long for loop를 통한 tick() 기반 simul

1. frontend(core)가 발행한 예상 instructions들을 모두 처리시 is_finished()가 true가 됨

yaml file

□ example_config.yaml

```
1  Frontend:
2    impl: SimpleO3
3    clock_ratio: 8
4    num_expected_insts: 500000
5  traces:
6    - example_inst.trace
7
8  Translation:
9    impl: RandomTranslation
10    max_addr: 2147483648
11
12
13 MemorySystem:
14   impl: GenericDRAM
15   clock_ratio: 3
16
17   DRAM:
18     impl: DDR4
19     org:
20       preset: DDR4_8Gb_x8
21       channel: 1
22     rank: 2
23     timing:
24       preset: DDR4_2400R
25
26   Controller:
27     impl: Generic
28     Scheduler:
29       impl: FRFCFS
30     RefreshManager:
31       impl: AllBank
32     RowPolicy:
33       impl: ClosedRowPolicy
34       cap: 4
35     plugins:
36
37   AddrMapper:
38     impl: RoBaRaCoCh
```

1. Frontend Interface(IFrontEnd) 부분

- trace file에서 Instruction읽고, Memory Request 생성
- impl: SimpleO3
⇒ Simple Out-of-Order (O3) CPU
- clock_ratio: 8
⇒ global CLK 대비 Frontend CLK 속도
- num_expected_insts: 500000
⇒ Simulation이 해당 instruction 수에 도달 시 종료
- traces
⇒ Instruction trace file(include memory access Inst)
- impl: RandomTranslation
⇒ Physical Memory ↔ Virtual Memory 변환
⇒ System의 Page Table 등을 간단히 Modeling
- max_addr: 2147483648
⇒ Translation 시 address overflow 방지

yaml file

□ example_config.yaml

2. MemorySystem 부분

```
1  √ Frontend:
2    impl: SimpleO3
3    clock_ratio: 8
4    num_expected_insts: 500000
5  √ traces:
6    - example_inst.trace
7
8  √ Translation:
9    impl: RandomTranslation
10   max_addr: 2147483648
11
12
13 MemorySystem:
14   impl: GenericDRAM
15   clock_ratio: 3
16
17   DRAM:
18     impl: DDR4
19     org:
20       preset: DDR4_8Gb_x8
21       channel: 1
22       rank: 2
23     timing:
24       preset: DDR4_2400R
25
26   Controller:
27     impl: Generic
28   Scheduler:
29     impl: FRFCFS
30   RefreshManager:
31     impl: AllBank
32   RowPolicy:
33     impl: ClosedRowPolicy
34     cap: 4
35   plugins:
36
37   AddrMapper:
38     impl: RoBaRaCoCh
```

trace file

□ example_inst.trace

```
1 ✓ Frontend: 13
2   impl: SimpleO3 14
3   clock_ratio: 8 15
4   num_expected_insts: 500000 16
5 ✓ traces: 17
6   - example_inst.trace 18
7 19
8 ✓ Translation: 20
9   impl: RandomTranslation 21
10   max_addr: 2147483648 22
11 23
24
25
26 MemorySystem:
27   impl: GenericDRAM
28   clock_ratio: 3
29
30 DRAM:
31   impl: DDR4
32   org:
33     preset: DDR4_8Gb_x8
34     channel: 1
35     rank: 2
36   timing:
37     preset: DDR4_2400R
38
39 Controller:
40   impl: Generic
41 Scheduler:
42   impl: FRFCFS
43 RefreshManager:
44   impl: AllBank
45 RowPolicy:
46   impl: ClosedRowPolicy
47   cap: 4
48 plugins:
49
50 AddrMapper:
51   impl: RoBaRaCoCh
```

trace file

1. F

src/base folder

□ **base**

- **abcd**

src/base folder

□ **base**

- **abcd**

src/base folder

□ **base**

- **abcd**

src/frontend folder

□ frontend

- abcd

src/frontend folder

□ frontend

- abcd

src/frontend folder

□ frontend

- abcd

src/memory_system folder

□ **memory_system**

- **abcd**

src/memory_system folder

□ **memory_system**

- **abcd**

src/addr_mapper folder

□ **addr_mapper**

- **abcd**

src/addr_mapper folder

❑ **addr_mapper**

- **abcd**

src/translation folder

□ translation

- abcd

src/translation folder

□ translation

- abcd

src/translation folder

□ translation

- abcd

src/dram folder

□ dram

- abcd

src/dram folder

□ dram

- abcd

src/dram folder

□ dram

- abcd

src/dram_controller folder

❑ **dram_controller**

- **abcd**

src/dram_controller folder

□ **dram_controller**

- **abcd**

src/dram_controller folder

❑ **dram_controller**

- **abcd**

src/dram_controller folder

□ **dram_controller**

- **abcd**