Schedule (EST)

Time slot	Topic	
9:10 to 9:30	Introduction to DNNs and Accelerator Design	Tushar
9:30 to 9:55	Overview of SCALE-Sim	Anand
10:00 to 10:30	Tutorial 1: Design Space Exploration using SCALE-Sim	Anand
10:30 to 11:00	Tutorial 2: Modifying SCALE-Sim to add custom features	Moritz
11:05 to 11:45	Tutorial 3: Krittika: Simulation of distributed GEMM operations using SCALE-sim API	Anand

Brief Q/A at the end of each talk.

Attention: Tutorial is being recorded!

Slides + Videos will be available on the SCALE-sim tutorial website

https://scalesim-project.github.io/tutorials-2021-isca.html

http://synergy.ece.gatech.edu

Tutorial 2: Modifying SCALE-Sim to add custom features

ISCA-2021

June 19, 2021

Objective

To demonstrate how you can make use of SCALE-Sim's modular structure to add features

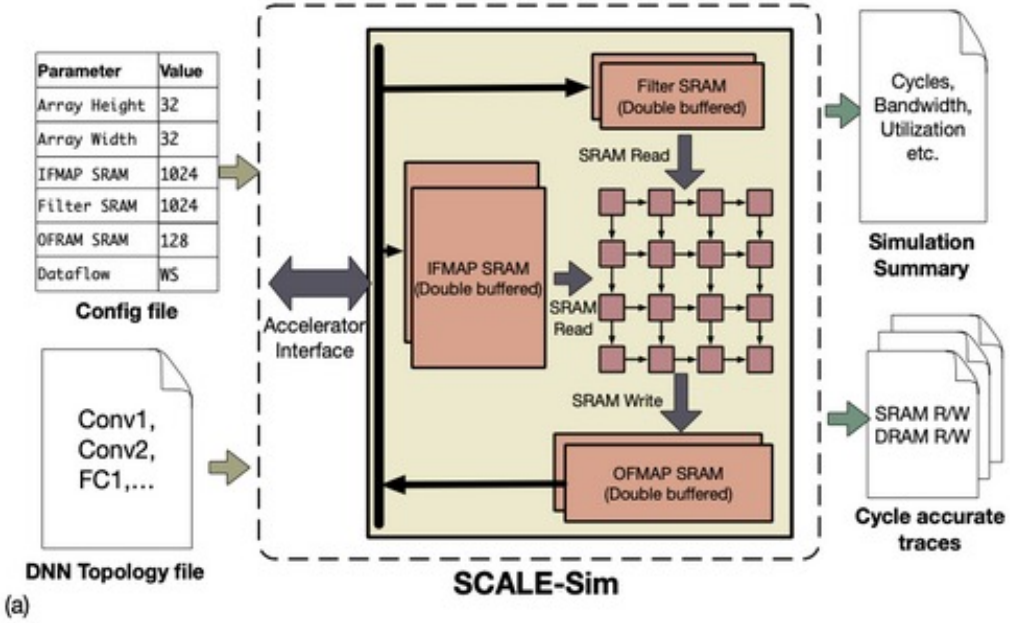
Overview

1. **Introduction to the architectural modification (case study).**
2. SCALE-Sim's modular software architecture.
3. Modifying SCALE-Sim
4. Case study results

What you have learned so far

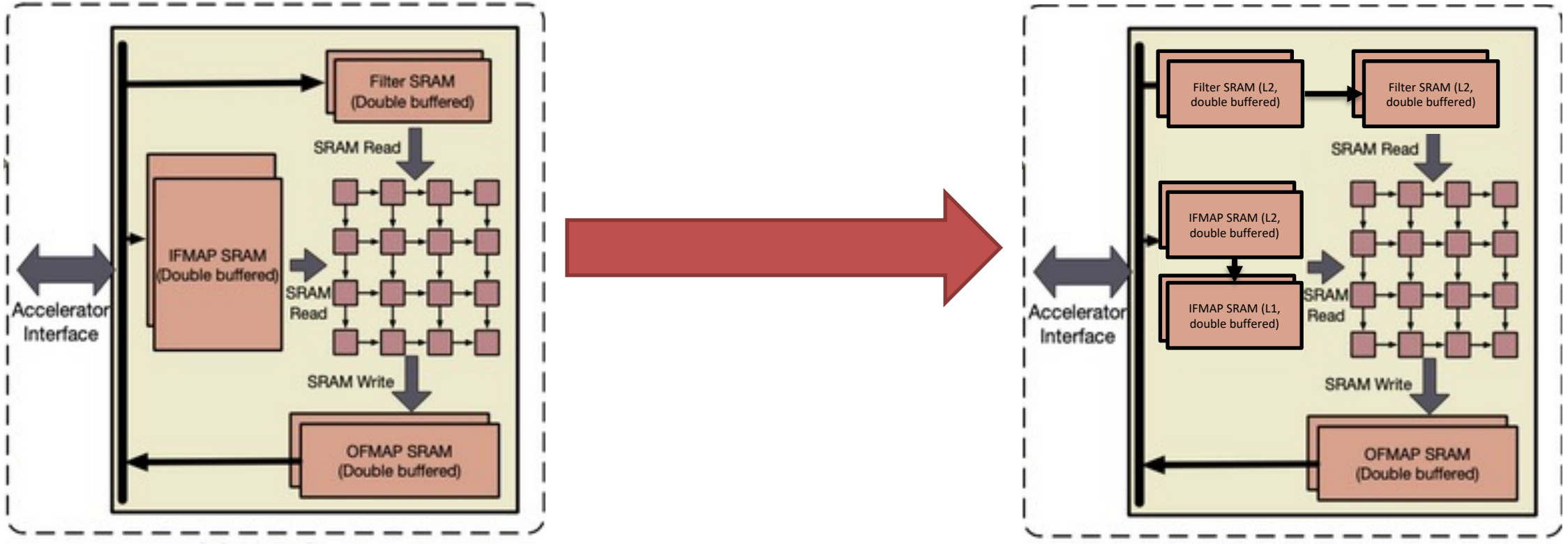
- Input to SCALE-Sim:
  - Workload parameters (DNN topology file) and architecture parameters (config file)
  - Architecture parameters allow to set array size, buffer sizes, dataflow, …
- Results:
  - Cycle accurate traces of the memory
  - Summarizes results, e.g., cycles, bandwidth, utilization

- Your architecture is fixed in this case.
- What if you want to simulate other architectures, e.g., another memory hierarchy?
- We go you covered!



The modification to SCALE-Sim's base architecture

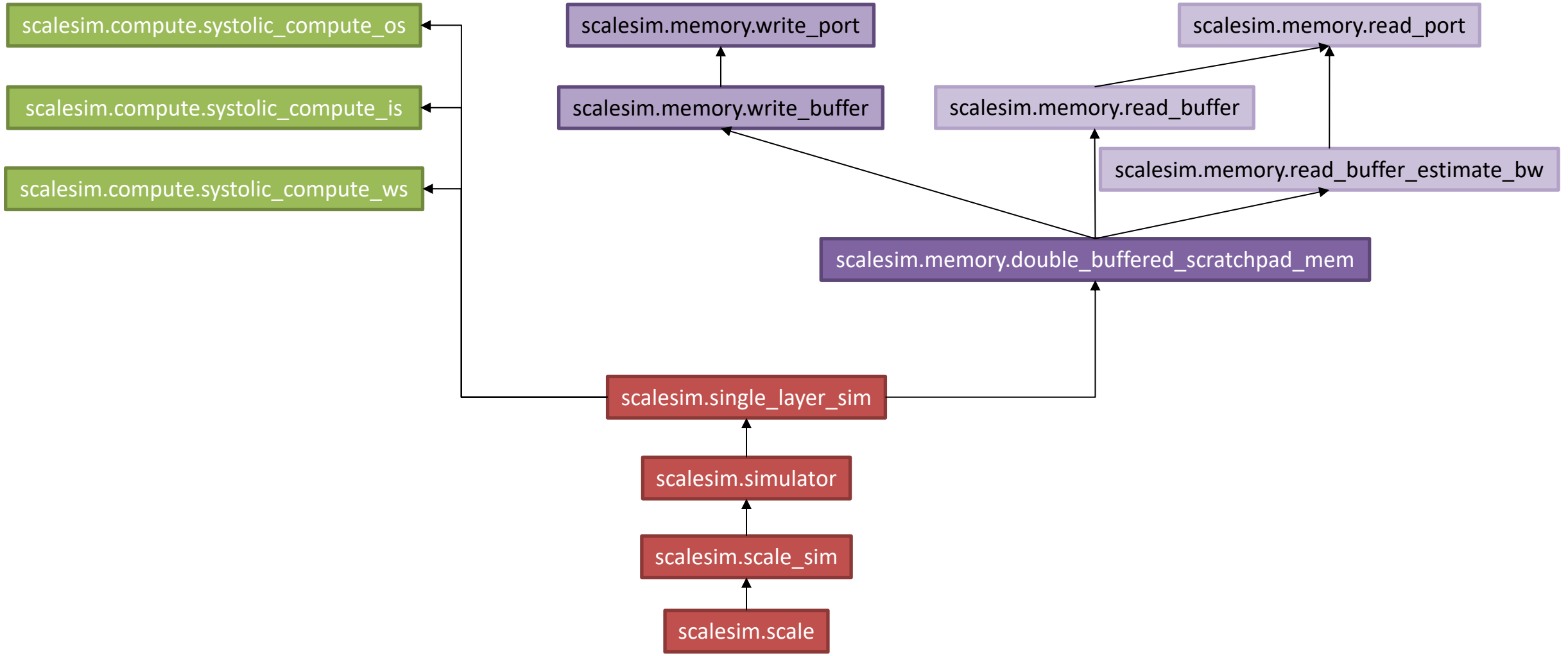
- Case Study: Use a new, custom memory hierarchy with L1 and L2 scratchpad memories.



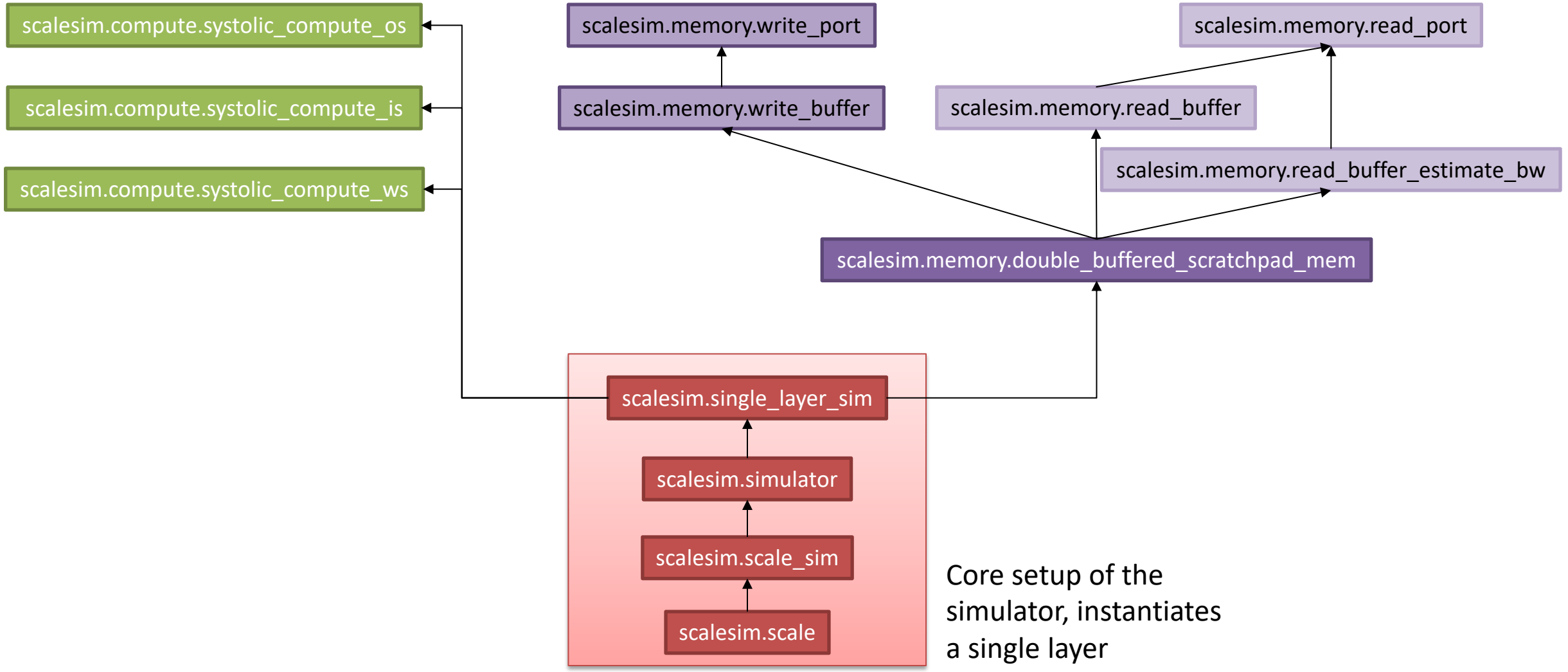
Overview

1. Introduction to the architectural modification (case study).
2. **SCALE-Sim's modular software architecture.**
3. Modifying SCALE-Sim
4. Case study results

SCALE-Sim's modular software architecture (UML extract)

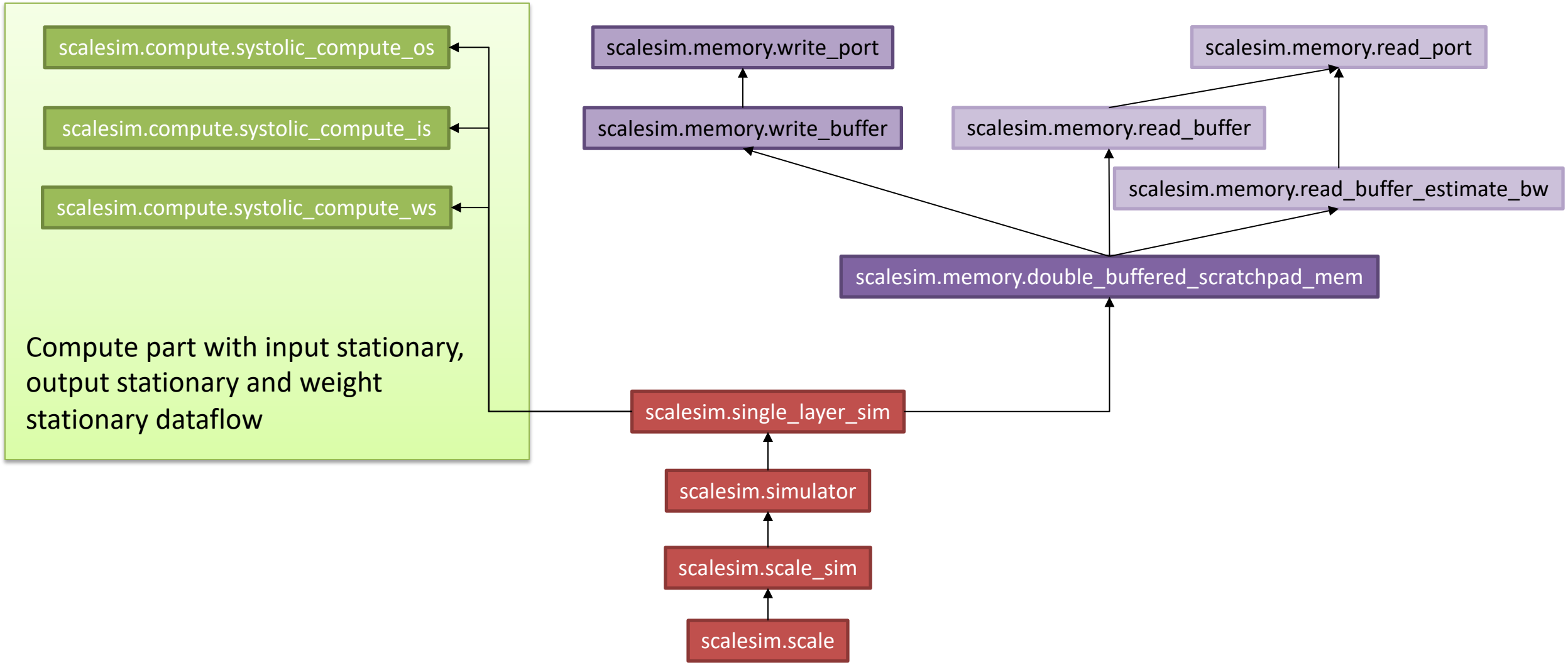


SCALE-Sim's modular software architecture (UML extract)

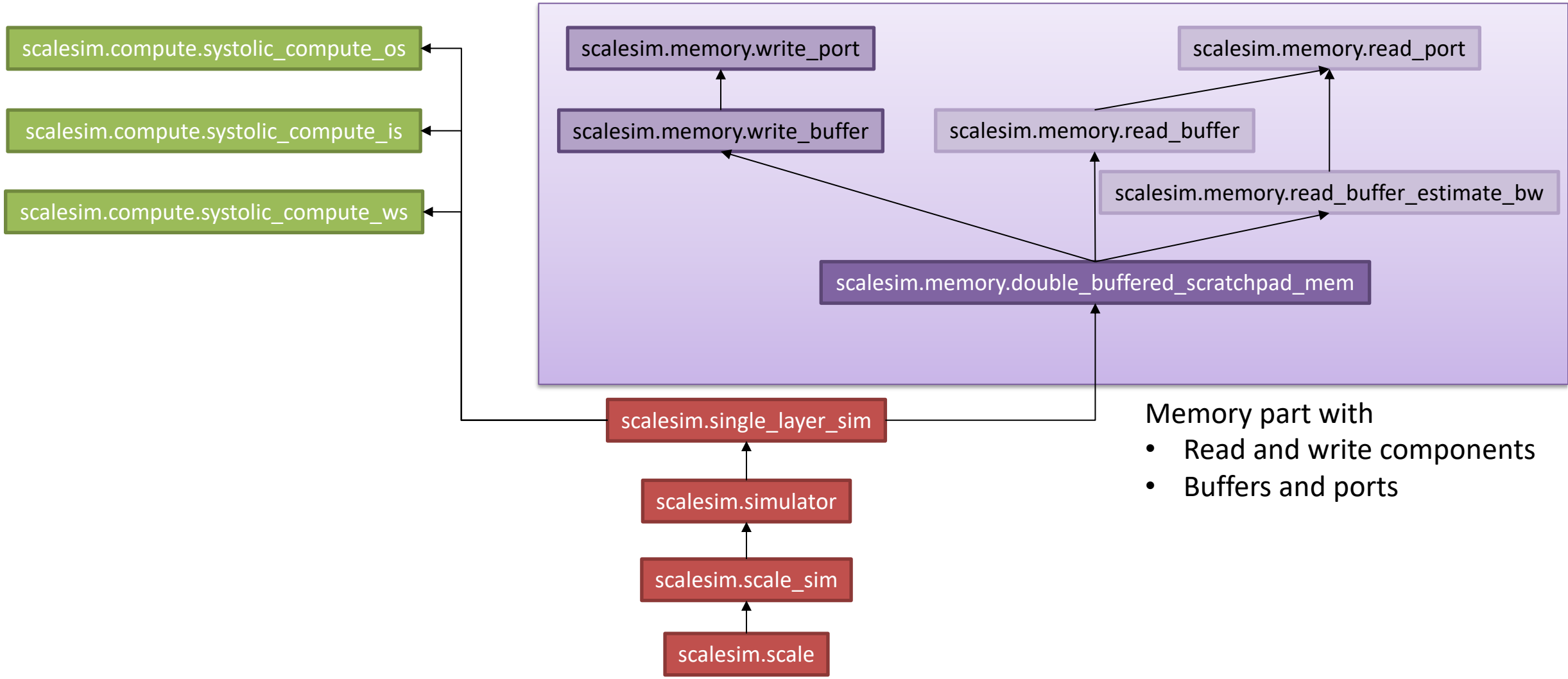


Core setup of the simulator, instantiates a single layer

SCALE-Sim's modular software architecture (UML extract)



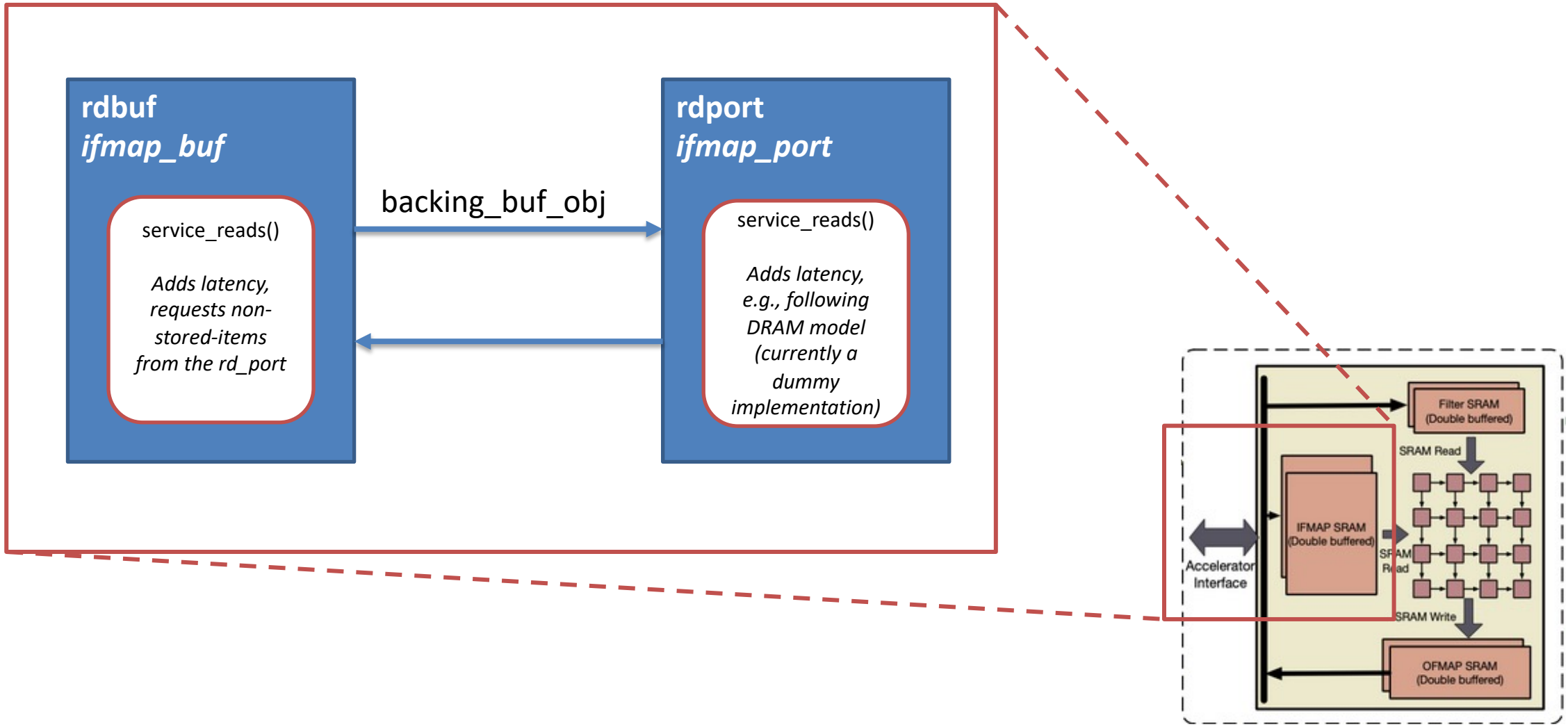
SCALE-Sim's modular software architecture (UML extract)



Memory part with

- Read and write components
- Buffers and ports

A closer look at the double-buffered memory



The implementation

- See `scalesim/memory/double_buffered_scratchpad_mem.py`

Instantiating the memory

```
5  from scalesim.memory.read_buffer import read_buffer as rdbuf
6  from scalesim.memory.read_buffer_estimate_bw import ReadBufferEstimateBw as rdbuf_est
7  from scalesim.memory.read_port import read_port as rdport
8  from scalesim.memory.write_buffer import write_buffer as wrbuf
9  from scalesim.memory.write_port import write_port as wrport
10
11
12 class double_buffered_scratchpad:
13     def __init__(self):
14         self.ifmap_buf = rdbuf()
15         self.filter_buf = rdbuf()
```

Connecting the memory and defining the size

```
85             self.ifmap_buf = rdbuf()
86             self.filter_buf = rdbuf()
87
88             self.ifmap_buf.set_params(backing_buf_obj=self.ifmap_port,
89                                       total_size_bytes=ifmap_buf_size_bytes,
90                                       word_size=word_size,
91                                       active_buf_frac=rd_buf_active_frac,
92                                       backing_buf_bw=ifmap_backing_buf_bw)
93
94             self.filter_buf.set_params(backing_buf_obj=self.filter_port,
95                                        total_size_bytes=filter_buf_size_bytes,
96                                        word_size=word_size,
97                                        active_buf_frac=rd_buf_active_frac,
98                                        backing_buf_bw=filter_backing_buf_bw)
```

Overview

1. Introduction to the architectural modification (case study).
2. SCALE-Sim's modular software architecture.
3. **Modifying SCALE-Sim**
4. Case study results

Step 0: Check your environment

- **Python:** Make sure that python >3.6 is installed:

```
[➜  ~ python --version
Python 2.7.16
➜  ~
```

```
[➜  ~ python3 --version
Python 3.9.2
➜  ~
```

If Python is not installed, use a terminal:

Ubuntu: `>sudo apt install python3 python3-venv`

MacOS: use homebrew, see https://docs.python-guide.org/starting/install3/osx/

```
>/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh) "
>brew install python
```

- Git: Make sure that git is installed

```
[➜  ~ git --version
git version 2.24.3 (Apple Git-128)
```

If Git is not installed, use a terminal:

Ubuntu: `>sudo apt install git`

MacOS: `>xcode-select --install`

Step 1: Get a copy of SCALE-Sim v2 for this tutorial

Short version:

- Download the tutorial2.sh file https://github.com/scalesim-project/tutorial-isca-2021/raw/master/tutorial2.sh
- Open a terminal
- `>chmod +x tutorial2.sh`
- `>sh tutorial2.sh`
- `>cd scale-sim-v2`
- `>source venv/bin/activate` Activates the virtual environment
- If successful, you should have a (venv)-mark before your terminal (see below)

```
1  #!/bin/bash
2
3  git clone https://github.com/scalesim-project/scale-sim-v2.git
4  cd scale-sim-v2
5  python3 -m venv venv
6  source venv/bin/activate
7  pip install -r requirements.txt
8  python setup.py install
```

`tutorial2.sh`, so that you know, what is does…

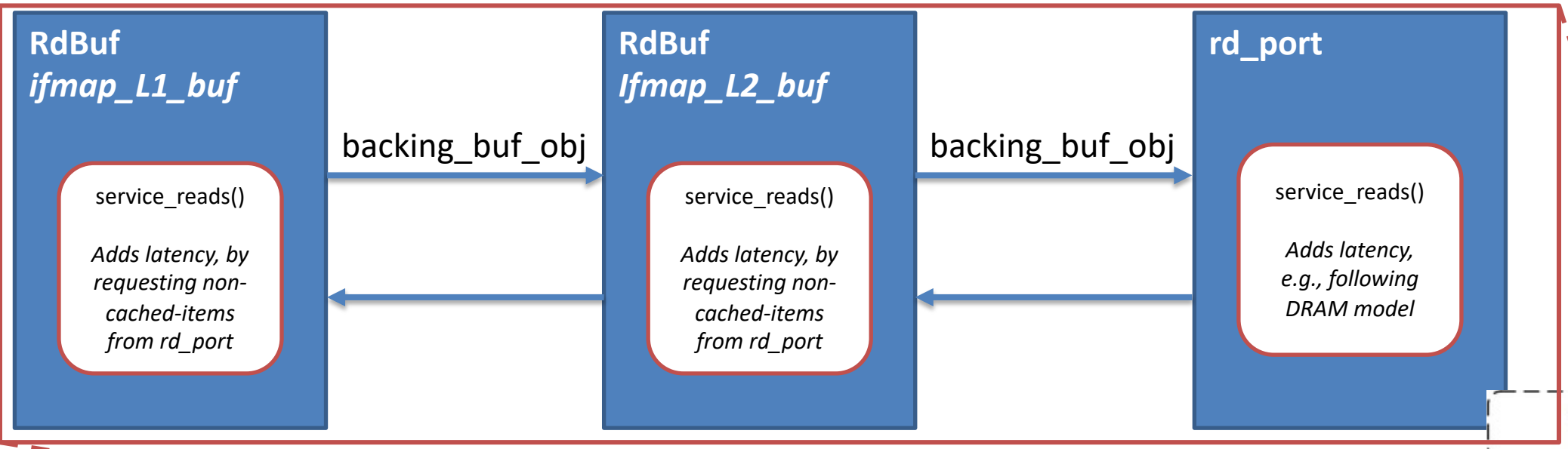
Long version, you can also do the steps from the file manually:

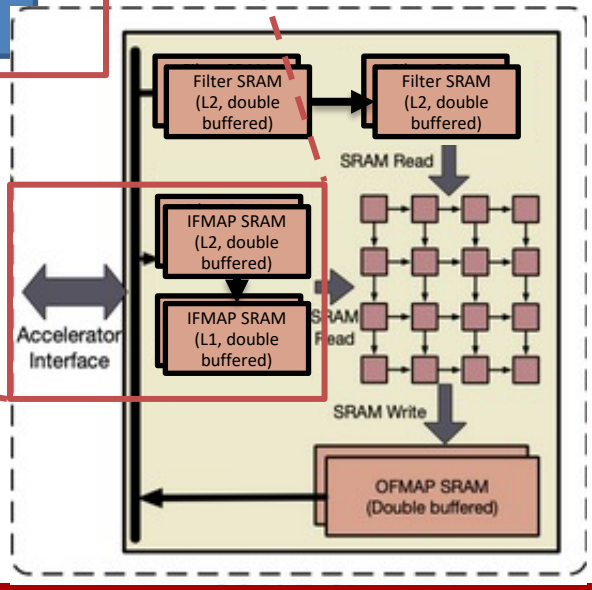
- Open a terminal
- `>git clone https://github.com/scalesim-project/scale-sim-v2` Clone SCALE-Sim
- `>cd scale-sim-v2` Enter SCALE-Sim folder
- `>python3 -m venv venv` Create a python virtual environment
- `>source venv/bin/activate` Activate the virtual environment
- Check that you run python 3.X using `python --version`, otherwise use `python3 --version`
- Terminal should now look like this:

```
→  scale-sim-v2-main git:(main) ✗ source venv/bin/activate
(venv) →  scale-sim-v2-main git:(main) ✗
```

- `>pip install -r requirements.txt` Install dependencies
- `>python setup.py install` Install SCALE-Sim into the virtual environment

Modifications to the memory system





Step 2: Instantiating two memory objects

Remark:
If you do not want to change the code, you can find a working version by:
`>git checkout tutorial/isca2021/tutorial2`

Here, you can compare the files `double_buffered_scratchpad_mem.py` and `double_buffered_tutorial2_scratchpad_mem.py`

Open `scalesim/memory/double_buffered_scratchpad_mem.py` in an editor and follow my changes.

```
5  from scalesim.memory.read_buffer import read_buffer as rdbuf
6  from scalesim.memory.read_buffer_estimate_bw import ReadBufferEstimateBw as rdbuf_est
7  from scalesim.memory.read_port import read_port as rdport
8  from scalesim.memory.write_buffer import write_buffer as wrbuf
9  from scalesim.memory.write_port import write_port as wrport
10
11
12 class double_buffered_scratchpad:
13     def __init__(self):
14         self.ifmap_buf = rdbuf()
15         self.filter_buf = rdbuf()
```

```
5  from scalesim.memory.read_buffer import read_buffer as rdbuf
6  from scalesim.memory.read_buffer_estimate_bw import ReadBufferEstimateBw as rdbuf_est
7  from scalesim.memory.read_port import read_port as rdport
8  from scalesim.memory.write_buffer import write_buffer as wrbuf
9  from scalesim.memory.write_port import write_port as wrport
10
11
12 class double_buffered_scratchpad:
13     def __init__(self):
14         self.ifmap_L1_buf = rdbuf()
15         self.filter_L1_buf = rdbuf()
16         self.ifmap_L2_buf = rdbuf()
17         self.filter_L2_buf = rdbuf()
```

Step 3: Memory connections and sizes

```
85              self.ifmap_buf = rdbuf()
86              self.filter_buf = rdbuf()
87
88              self.ifmap_buf.set_params(backing_buf_obj=self.ifmap_port,
89                                        total_size_bytes=ifmap_buf_size_bytes,
90                                        word_size=word_size,
91                                        active_buf_frac=rd_buf_active_frac,
92                                        backing_buf_bw=ifmap_backing_buf_bw)
93
94              self.filter_buf.set_params(backing_buf_obj=self.filter_port,
95                                         total_size_bytes=filter_buf_size_bytes,
96                                         word_size=word_size,
97                                         active_buf_frac=rd_buf_active_frac,
98                                         backing_buf_bw=filter_backing_buf_bw)
```

```
71          if self.estimate_bandwidth_mode:
72              self.ifmap_L1_buf = rdbuf_est()
73              self.ifmap_L2_buf = rdbuf_est()
74              self.filter_L1_buf = rdbuf_est()
75              self.filter_L2_buf = rdbuf_est()
76
77              self.ifmap_L1_buf.set_params(backing_buf_obj=self.ifmap_L2_buf,
78                                           total_size_bytes=ifmap_buf_size_bytes,
79                                           word_size=word_size,
80                                           active_buf_frac=rd_buf_active_frac,
81                                           backing_buf_default_bw=ifmap_backing_buf_bw)
82
83              self.ifmap_L2_buf.set_params(backing_buf_obj=self.ifmap_port,
84                                           total_size_bytes=ifmap_buf_size_bytes * 2,
85                                           word_size=word_size,
86                                           active_buf_frac=rd_buf_active_frac,
87                                           backing_buf_default_bw=ifmap_backing_buf_bw)
88
89              self.filter_L1_buf.set_params(backing_buf_obj=self.filter_L2_buf,
90                                            total_size_bytes=filter_buf_size_bytes,
91                                            word_size=word_size,
92                                            active_buf_frac=rd_buf_active_frac,
93                                            backing_buf_default_bw=filter_backing_buf_bw)
94
95              self.filter_L2_buf.set_params(backing_buf_obj=self.filter_port,
96                                            total_size_bytes=filter_buf_size_bytes * 2,
97                                            word_size=word_size,
98                                            active_buf_frac=rd_buf_active_frac,
99                                            backing_buf_default_bw=filter_backing_buf_bw)
```

Step 4: Connecting set and reset

```
111     def set_read_buf_prefetch_matrices(self,
112                                        ifmap_prefetch_mat=np.zeros((1,1)),
113                                        filter_prefetch_mat=np.zeros((1,1))
114                                        ):
115
116         self.ifmap_buf.set_fetch_matrix(ifmap_prefetch_mat)
117         self.filter_buf.set_fetch_matrix(filter_prefetch_mat)
118
119     #
120     def reset_buffer_states(self):
121
122         self.ifmap_buf.reset()
123         self.filter_buf.reset()
124         self.ofmap_buf.reset()
```

```
141     def set_read_buf_prefetch_matrices(self,
142                                        ifmap_prefetch_mat=np.zeros((1,1)),
143                                        filter_prefetch_mat=np.zeros((1,1))
144                                        ):
145
146         self.ifmap_L1_buf.set_fetch_matrix(ifmap_prefetch_mat)
147         self.ifmap_L2_buf.set_fetch_matrix(ifmap_prefetch_mat)
148         self.filter_L1_buf.set_fetch_matrix(filter_prefetch_mat)
149         self.filter_L2_buf.set_fetch_matrix(filter_prefetch_mat)
150
151     #
152     def reset_buffer_states(self):
153
154         self.ifmap_L1_buf.reset()
155         self.ifmap_L2_buf.reset()
156         self.filter_L1_buf.reset()
157         self.filter_L2_buf.reset()
158         self.ofmap_buf.reset()
```

Step 5: Connecting the `service_read` function

```
126     # The following are just shell methods for users to control each mem individually
127     def service_ifmap_reads(self,
128                             incoming_requests_arr_np,   # 2D array with the requests
129                             incoming_cycles_arr):
130         out_cycles_arr_np = self.ifmap_buf.service_reads(incoming_requests_arr_np, incoming_cycles_arr)
131
132         return out_cycles_arr_np
133
134     #
135     def service_filter_reads(self,
136                             incoming_requests_arr_np,   # 2D array with the requests
137                             incoming_cycles_arr):
138         out_cycles_arr_np = self.filter_buf.service_reads(incoming_requests_arr_np, incoming_cycles_arr)
139
140         return out_cycles_arr_np
```

```
160     # The following are just shell methods for users to control each mem individually
161     def service_ifmap_reads(self,
162                             incoming_requests_arr_np,   # 2D array with the requests
163                             incoming_cycles_arr):
164         out_cycles_arr_np = self.ifmap_L1_buf.service_reads(incoming_requests_arr_np, incoming_cycles_arr)
165
166         return out_cycles_arr_np
167
168     #
169     def service_filter_reads(self,
170                             incoming_requests_arr_np,   # 2D array with the requests
171                             incoming_cycles_arr):
172         out_cycles_arr_np = self.filter_L1_buf.service_reads(incoming_requests_arr_np, incoming_cycles_arr)
173
174         return out_cycles_arr_np
```

Step 6: Changing the `service_memory_request` function

```python
def service_memory_requests(self, ifmap_demand_mat, filter_demand_mat, ofmap_demand_mat):
    assert self.params_valid_flag, 'Memories not initialized yet'

    ofmap_lines = ofmap_demand_mat.shape[0]

    self.total_cycles = 0
    self.stall_cycles = 0

    ifmap_hit_latency = self.ifmap_buf.get_hit_latency()
    filter_hit_latency = self.ifmap_buf.get_hit_latency()

    ifmap_serviced_cycles = []
    filter_serviced_cycles = []
    ofmap_serviced_cycles = []

    pbar_disable = not self.verbose
    for i in tqdm(range(ofmap_lines), disable=pbar_disable):

        cycle_arr = np.zeros((1,1)) + i + self.stall_cycles

        ifmap_demand_line = ifmap_demand_mat[i, :].reshape((1,ifmap_demand_mat.shape[1]))
        ifmap_cycle_out = self.ifmap_buf.service_reads(incoming_requests_arr_np=ifmap_demand_line,
                                                       incoming_cycles_arr=cycle_arr)
        ifmap_serviced_cycles += [ifmap_cycle_out[0]]
        ifmap_stalls = ifmap_cycle_out[0] - cycle_arr[0] - ifmap_hit_latency

        filter_demand_line = filter_demand_mat[i, :].reshape((1, filter_demand_mat.shape[1]))
        filter_cycle_out = self.filter_buf.service_reads(incoming_requests_arr_np=filter_demand_line,
                                                         incoming_cycles_arr=cycle_arr)
        filter_serviced_cycles += [filter_cycle_out[0]]
        filter_stalls = filter_cycle_out[0] - cycle_arr[0] - filter_hit_latency

        ofmap_demand_line = ofmap_demand_mat[i, :].reshape((1, ofmap_demand_mat.shape[1]))
        ofmap_cycle_out = self.ofmap_buf.service_writes(incoming_requests_arr_np=ofmap_demand_line,
                                                        incoming_cycles_arr_np=cycle_arr)
        ofmap_serviced_cycles += [ofmap_cycle_out[0]]
        ofmap_stalls = ofmap_cycle_out[0] - cycle_arr[0] - 1

        self.stall_cycles += int(max(ifmap_stalls[0], filter_stalls[0], ofmap_stalls[0]))

    if self.estimate_bandwidth_mode:
        # IDE shows warning as complete_all_prefetches is not implemented in read_buffer class
        # It is harmless since, in estimate bandwidth mode, read_buffer_estimate_bw is instantiated
        self.ifmap_buf.complete_all_prefetches()
        self.filter_buf.complete_all_prefetches()
```

```python
def service_memory_requests(self, ifmap_demand_mat, filter_demand_mat, ofmap_demand_mat):
    assert self.params_valid_flag, 'Memories not initialized yet'

    ofmap_lines = ofmap_demand_mat.shape[0]

    self.total_cycles = 0
    self.stall_cycles = 0

    ifmap_hit_latency = self.ifmap_L1_buf.get_hit_latency()
    filter_hit_latency = self.ifmap_L2_buf.get_hit_latency()

    ifmap_serviced_cycles = []
    filter_serviced_cycles = []
    ofmap_serviced_cycles = []

    pbar_disable = not self.verbose
    for i in tqdm(range(ofmap_lines), disable=pbar_disable):

        cycle_arr = np.zeros((1,1)) + i + self.stall_cycles

        ifmap_demand_line = ifmap_demand_mat[i, :].reshape((1,ifmap_demand_mat.shape[1]))
        ifmap_cycle_out = self.ifmap_L1_buf.service_reads(incoming_requests_arr_np=ifmap_demand_line,
                                                          incoming_cycles_arr=cycle_arr)
        ifmap_serviced_cycles += [ifmap_cycle_out[0]]
        ifmap_stalls = ifmap_cycle_out[0] - cycle_arr[0] - ifmap_hit_latency

        filter_demand_line = filter_demand_mat[i, :].reshape((1, filter_demand_mat.shape[1]))
        filter_cycle_out = self.filter_L1_buf.service_reads(incoming_requests_arr_np=filter_demand_line,
                                                            incoming_cycles_arr=cycle_arr)
        filter_serviced_cycles += [filter_cycle_out[0]]
        filter_stalls = filter_cycle_out[0] - cycle_arr[0] - filter_hit_latency

        ofmap_demand_line = ofmap_demand_mat[i, :].reshape((1, ofmap_demand_mat.shape[1]))
        ofmap_cycle_out = self.ofmap_buf.service_writes(incoming_requests_arr_np=ofmap_demand_line,
                                                        incoming_cycles_arr_np=cycle_arr)
        ofmap_serviced_cycles += [ofmap_cycle_out[0]]
        ofmap_stalls = ofmap_cycle_out[0] - cycle_arr[0] - 1

        self.stall_cycles += int(max(ifmap_stalls[0], filter_stalls[0], ofmap_stalls[0]))

    if self.estimate_bandwidth_mode:
        # IDE shows warning as complete_all_prefetches is not implemented in read_buffer class
        # It is harmless since, in estimate bandwidth mode, read_buffer_estimate_bw is instantiated
        self.ifmap_L1_buf.complete_all_prefetches()
        self.filter_L1_buf.complete_all_prefetches()
        self.ifmap_L2_buf.complete_all_prefetches()
        self.filter_L2_buf.complete_all_prefetches()
```

Step 7: Generating correct DRAM traces

```
451    def get_ifmap_dram_details(self):
452        assert self.traces_valid, 'Traces not generated yet'
453
454        self.ifmap_dram_reads = self.ifmap_buf.get_num_accesses()
455        self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle \
456            = self.ifmap_buf.get_external_access_start_stop_cycles()
457
458        return self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle, self.ifmap_dram_reads
459
460    #
461    def get_filter_dram_details(self):
462        assert self.traces_valid, 'Traces not generated yet'
463
464        self.filter_dram_reads = self.filter_buf.get_num_accesses()
465        self.filter_dram_start_cycle, self.filter_dram_stop_cycle \
466            = self.filter_buf.get_external_access_start_stop_cycles()
467
468        return self.filter_dram_start_cycle, self.filter_dram_stop_cycle, self.filter_dram_reads
```

```
487    def get_ifmap_dram_details(self):
488        assert self.traces_valid, 'Traces not generated yet'
489
490        self.ifmap_dram_reads = self.ifmap_L2_buf.get_num_accesses()
491        self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle \
492            = self.ifmap_L2_buf.get_external_access_start_stop_cycles()
493
494        return self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle, self.ifmap_dram_reads
495
496    #
497    def get_filter_dram_details(self):
498        assert self.traces_valid, 'Traces not generated yet'
499
500        self.filter_dram_reads = self.filter_L2_buf.get_num_accesses()
501        self.filter_dram_start_cycle, self.filter_dram_stop_cycle \
502            = self.filter_L2_buf.get_external_access_start_stop_cycles()
503
504        return self.filter_dram_start_cycle, self.filter_dram_stop_cycle, self.filter_dram_reads
```

Step 8: Generating correct buffer traces

```
501     def get_ifmap_dram_trace_matrix(self):
502         return self.ifmap_buf.get_trace_matrix()
503
504     #
505     def get_filter_dram_trace_matrix(self):
506         return self.filter_buf.get_trace_matrix()
507
508     #
509     def get_ofmap_dram_trace_matrix(self):
510         return self.ofmap_buf.get_trace_matrix()
511
512     #
513     def get_dram_trace_matrices(self):
514         dram_ifmap_trace = self.ifmap_buf.get_trace_matrix()
515         dram_filter_trace = self.filter_buf.get_trace_matrix()
516         dram_ofmap_trace = self.ofmap_buf.get_trace_matrix()
517
518         return dram_ifmap_trace, dram_filter_trace, dram_ofmap_trace
```

```
537     def get_ifmap_dram_trace_matrix(self):
538         return self.ifmap_L2_buf.get_trace_matrix()
539
540     #
541     def get_filter_dram_trace_matrix(self):
542         return self.filter_L2_buf.get_trace_matrix()
543
544     #
545     def get_ofmap_dram_trace_matrix(self):
546         return self.ofmap_buf.get_trace_matrix()
547
548     #
549     def get_dram_trace_matrices(self):
550         dram_ifmap_trace = self.ifmap_L2_buf.get_trace_matrix()
551         dram_filter_trace = self.filter_L2_buf.get_trace_matrix()
552         dram_ofmap_trace = self.ofmap_buf.get_trace_matrix()
553
554         return dram_ifmap_trace, dram_filter_trace, dram_ofmap_trace
```

Step 9: Print correct traces

```
536     def print_ifmap_dram_trace(self, filename):
537         self.ifmap_buf.print_trace(filename)
538
539     #
540     def print_filter_dram_trace(self, filename):
541         self.filter_buf.print_trace(filename)
```

```
572     def print_ifmap_dram_trace(self, filename):
573         self.ifmap_L2_buf.print_trace(filename)
574
575     #
576     def print_filter_dram_trace(self, filename):
577         self.filter_L2_buf.print_trace(filename)
```

Step 10: Integration into the project

1. Download the `double_buffered_tutorial2_scratchpad_mem.py` from https://github.com/scalesim-project/tutorial/isca2021/2021/raw/main/double_buffered_tutorial2_scratchpad_mem.py and paste it to `scalesim/memory`
2. Add the new memory to SCALE-Sim:
   1. Replace `scalesim/single_layer_sim.py` with the new version from https://github.com/scalesim-project/scale-sim-v2/raw/tutorial/isca2021/tutorial2/scalesim/single_layer_sim.py
      Modifications:

```
10 from scalesim.memory.double_buffered_tutorial2_scratchpad_mem import double_buffered_scratchpad as tut2mem
11
12
13 class single_layer_sim:
14     def __init__(self):
15         self.layer_id = 0
16         self.topo = topo()
17         self.config = cfg()
18
19         self.op_mat_obj = opmat()
20         self.compute_system = systolic_compute_os()
21         #self.memory_system = mem_dbsp()
22         self.memory_system = tut2mem()
```

Import new memory

Replace old memory

Overview

1. Introduction to the architectural modification (case study).
2. SCALE-Sim's modular software architecture.
3. Modifying SCALE-Sim
4. **Case study results**

Step 11: Run and get results

Running the project

- `>cd scalesim`
- `>python scale.py`

Old memory hierachy w/
one double-buffered memory

```
(venv) ➜  scale-sim-v2-main git:(main) ✗ cd scalesim
(venv) ➜  scalesim git:(main) ✗ python scale.py
====================================================
****************** SCALE SIM **********************
====================================================
Array Size:     32x32
SRAM IFMAP (kB):        64
SRAM Filter (kB):       64
SRAM OFMAP (kB):        64
Dataflow:       Output Stationary
CSV file path:  ../topologies/conv_nets/test.csv
Number of Remote Memory Banks:  1
Working in ESTIMATE BANDWIDTH mode.
====================================================

Running Layer 0
100%|████████████████████████████████████| 622920/622920 [04:26<00:00, 2335.93it/s]
Compute cycles: 622919
Stall cycles: 0
Overall utilization: 93.62%
Mapping efficiency: 96.98%
Average IFMAP DRAM BW: 4.742 words/cycle
Average Filter DRAM BW: 30.761 words/cycle
Average OFMAP DRAM BW: 0.612 words/cycle
************ SCALE SIM Run Complete ****************
```

New memory hierachy w/
two double-buffered memories

```
(venv) ➜  scale-sim-v2-main git:(main) ✗ cd scalesim
(venv) ➜  scalesim git:(main) ✗ python scale.py
====================================================
****************** SCALE SIM **********************
====================================================
Array Size:     32x32
SRAM IFMAP (kB):        64
SRAM Filter (kB):       64
SRAM OFMAP (kB):        64
Dataflow:       Output Stationary
CSV file path:  ../topologies/conv_nets/test.csv
Number of Remote Memory Banks:  1
Working in ESTIMATE BANDWIDTH mode.
====================================================

Running Layer 0
100%|████████████████████████████████████| 622920/622920 [05:16<00:00, 1968.27it/s]
Compute cycles: 622919
Stall cycles: 0
Overall utilization: 93.62%
Mapping efficiency: 96.98%
Average IFMAP DRAM BW: 3.898 words/cycle
Average Filter DRAM BW: 1.337 words/cycle
Average OFMAP DRAM BW: 0.612 words/cycle
************ SCALE SIM Run Complete ****************
```

Schedule (EST)

Time slot	Topic	
9:10 to 9:30	Introduction to DNNs and Accelerator Design	Tushar
9:30 to 9:55	Overview of SCALE-Sim	Anand
10:00 to 10:30	Tutorial 1: Design Space Exploration using SCALE-Sim	Anand
10:30 to 11:00	Tutorial 2: Modifying SCALE-Sim to add custom features	Moritz
11:05 to 11:45	Tutorial 3: Krittika: Simulation of distributed GEMM operations using SCALE-sim API	Anand

Brief Q/A at the end of each talk.

Attention: Tutorial is being recorded!

Slides + Videos will be available on the SCALE-sim tutorial website

https://scalesim-project.github.io/tutorials-2021-isca.html