[AI System Design – Term Project]

Optimizations of AI Accelerator Networks

Chester Sungchung Park (박성정)
SoC Design Lab, Konkuk University
Webpage: http://soclab.konkuk.ac.kr

Teaching Assistants

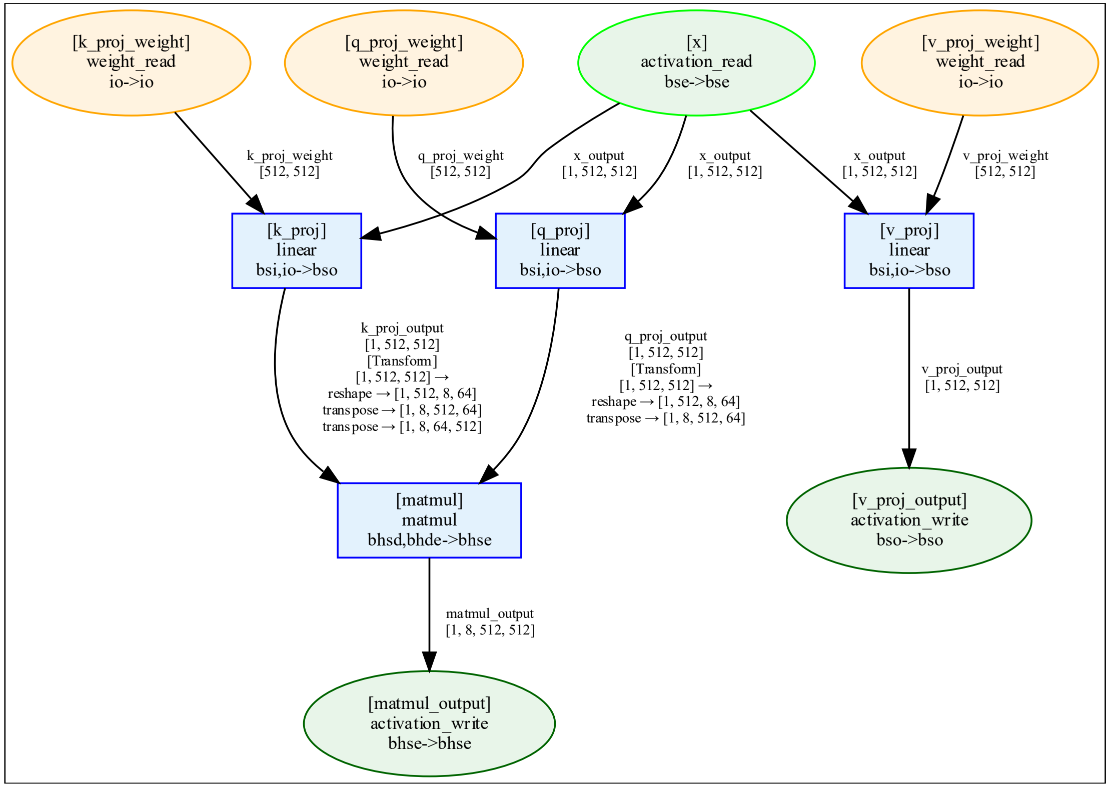
- ❑ Shinyoung Kim (shinyoungkim@konkuk.ac.kr)
- ❑ Jaesuk Lee (jaesuki98@konkuk.ac.kr)

Outline

- ❑ Introduction
  - Target workload
  - NetTLMSim
- ❑ Problem to solve
- ❑ Reference codes
- ❑ Evaluation
- ❑ Submission

Target Workload

- ❑ BERT-medium ***Layer group A***
  - Layer group 0 + matmul (QK^T)



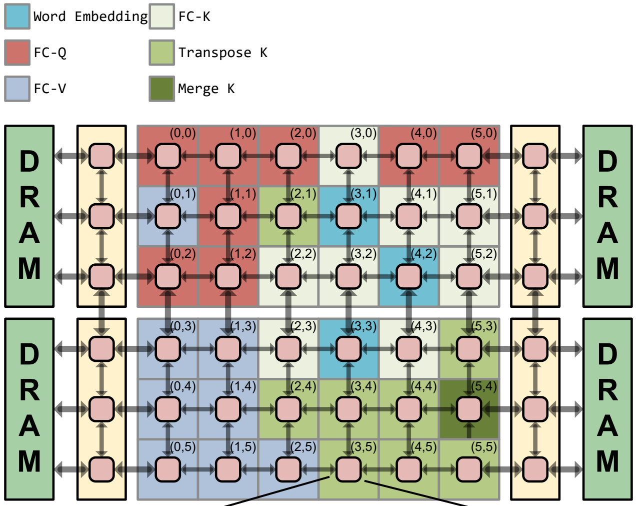
NetTLMSim

❑ Virtual prototype

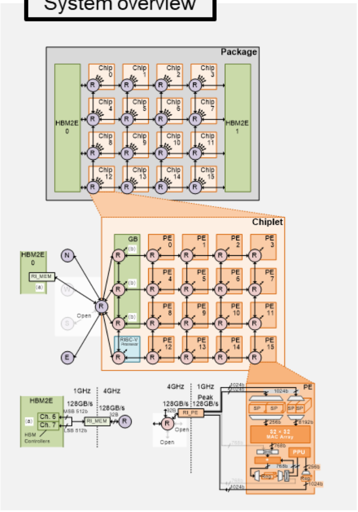
Assumed design parameters

Space-Time Mapping	Target Hardware	16 chiplets per package [1], 16 PEs per chiplet [1, 2], 1024 MACs per PE [1, 3] Total 512 TOPS at 1 GHz freq.
	PE Style	NVDLA-like vector MAC array
	Loop Order	OS-LWS [4]
Network Architecture	Topology	Mesh [2]
	Routing	DOR YX [2]
	Flow Ctrl	Cut-through [2]
	Packet Len	1 head flit, 16 body flits [2]
Memory Architecture	DRAM Type	HBM2E, 2 devices Total 1024 GB/s
	SRAM Type	Dual port [3]
	Global Buf. Size, Bandwidth	640 KB activation storage [4], 128 GB/s per NoC router
	Local Buf. Size, Bandwidth	36 KB x2 (double buffered) [4], 128 GB/s at 1 GHz freq.
	Precision	8 bits (24 bits for partial sums) [4]

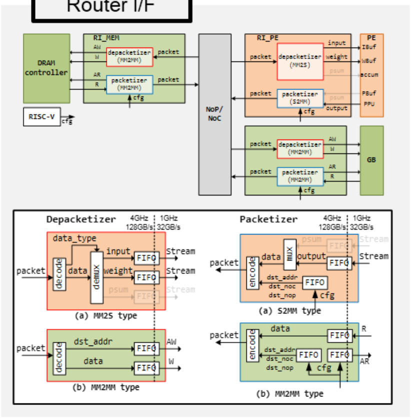
[1] Cai at el. 2024
[2] Zimmer at el. 2020
[3] Keller at el. 2023
[4] Venkatesan at el. 2019



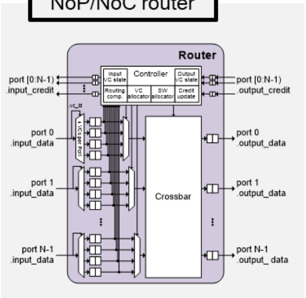
System overview

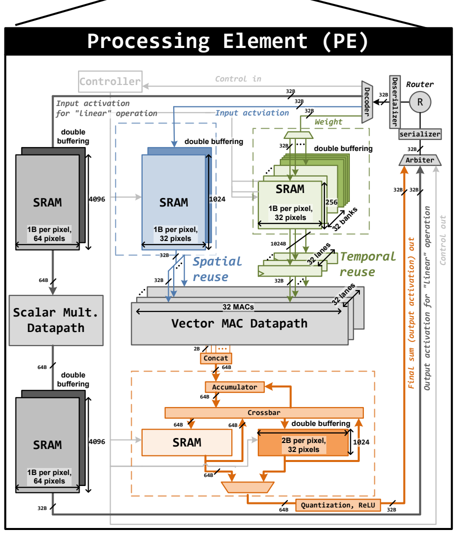


Router I/F



NoP/NoC router

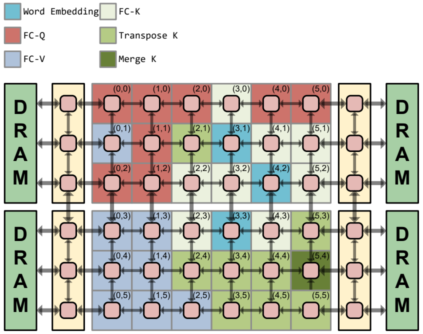


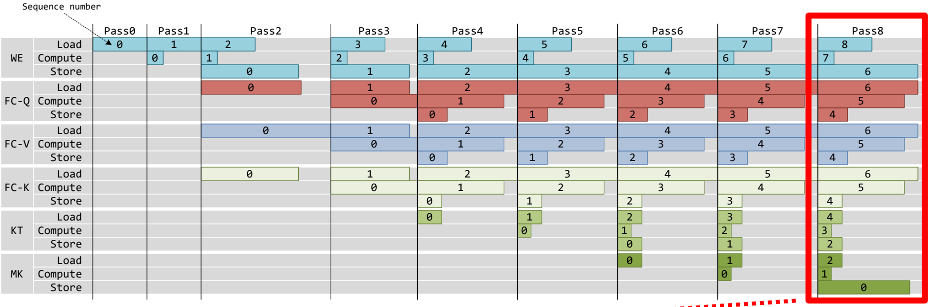


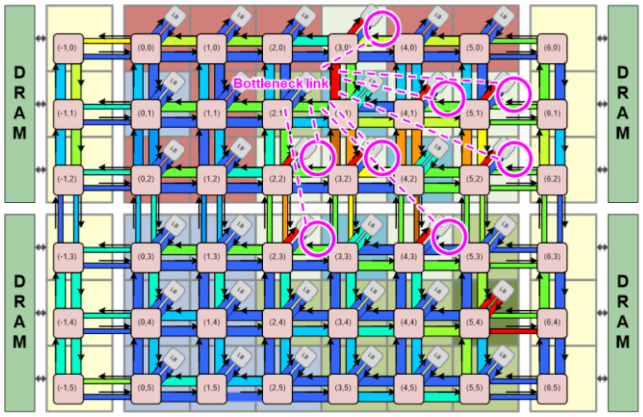
For more details, check the following presentation:
F. S. Park and C. S. Park, *Pre-RTL simulation based design space exploration for multi-chiplet dataflows*, HiPChips (MICRO 2024).

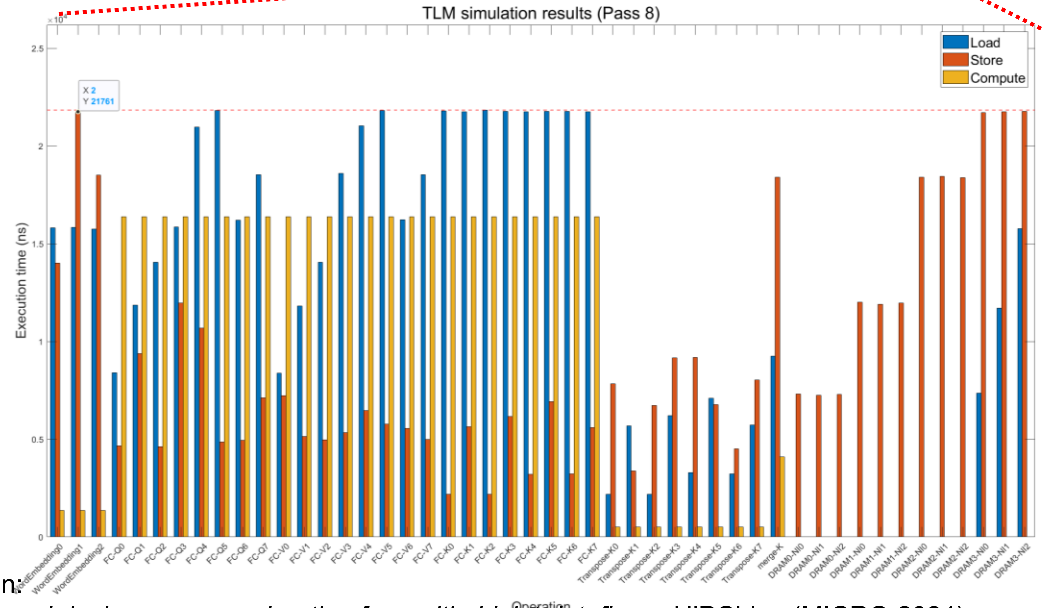
NetTLMSim

❑ Virtual prototype (cont'd)





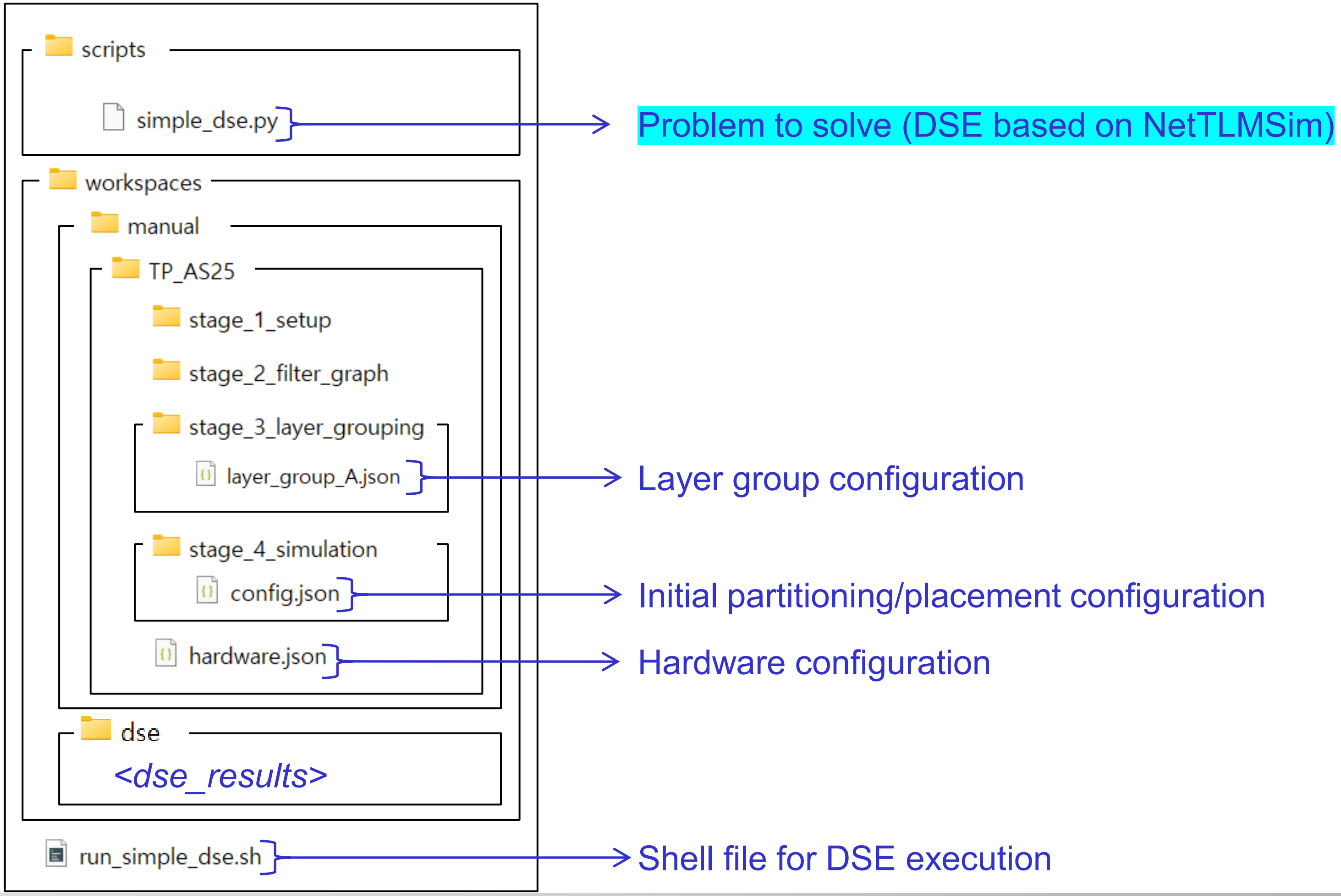




For more details, check the following presentation:
F. S. Park and C. S. Park, *Pre-RTL simulation based design space exploration for multi-chiplet dataflows*, HiPChips (MICRO 2024).

Reference Project

❑ Build project and run simulation (See ***Appendix A***)



```
scripts
    simple_dse.py
workspaces
    manual
        TP_AS25
            stage_1_setup
            stage_2_filter_graph
            stage_3_layer_grouping
                layer_group_A.json
            stage_4_simulation
                config.json
            hardware.json
    dse
        <dse_results>
run_simple_dse.sh
```

simple_dse.py → Problem to solve (DSE based on NetTLMSim)

layer_group_A.json → Layer group configuration

config.json → Initial partitioning/placement configuration

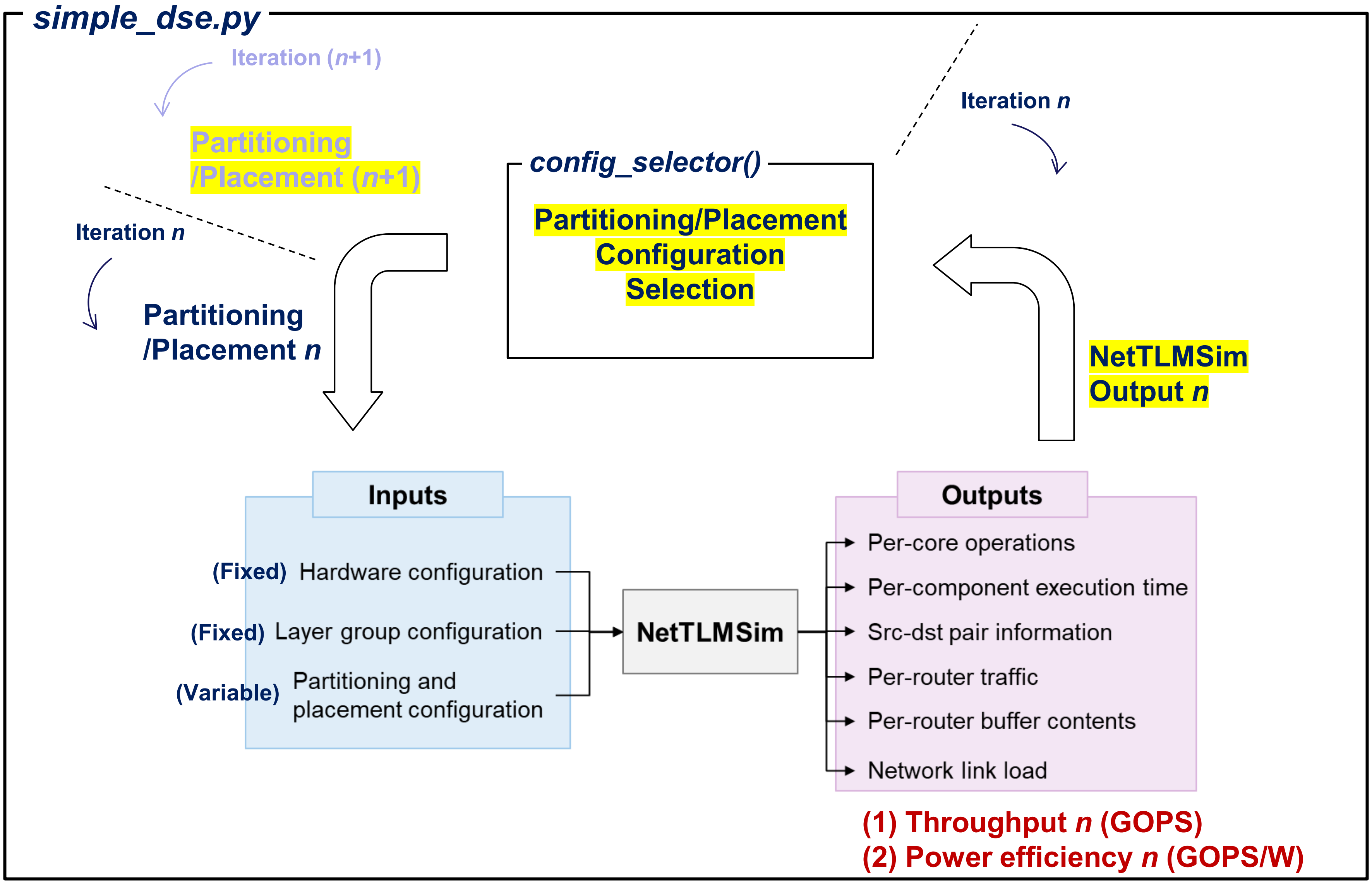
hardware.json → Hardware configuration

run_simple_dse.sh → Shell file for DSE execution

Problem to Solve

- Partitioning/placement configuration selection
  - Input: **NetTLMSim output** for the *n*-th iteration
  - Output: **NetTLMSim input** for the (*n*+1)-th iteration
    - ✓ Only partitioning/placement configuration *varied*
    - ✓ Hardware/layer group configuration *kept fixed*

Problem to Solve



simple_dse.py

Iteration (*n*+1)

Partitioning /Placement (*n*+1)

Iteration *n*

Partitioning /Placement *n*

config_selector()

Partitioning/Placement Configuration Selection

Iteration *n*

NetTLMSim Output *n*

Inputs

- (Fixed) Hardware configuration
- (Fixed) Layer group configuration
- (Variable) Partitioning and placement configuration

NetTLMSim

Outputs

- Per-core operations
- Per-component execution time
- Src-dst pair information
- Per-router traffic
- Per-router buffer contents
- Network link load

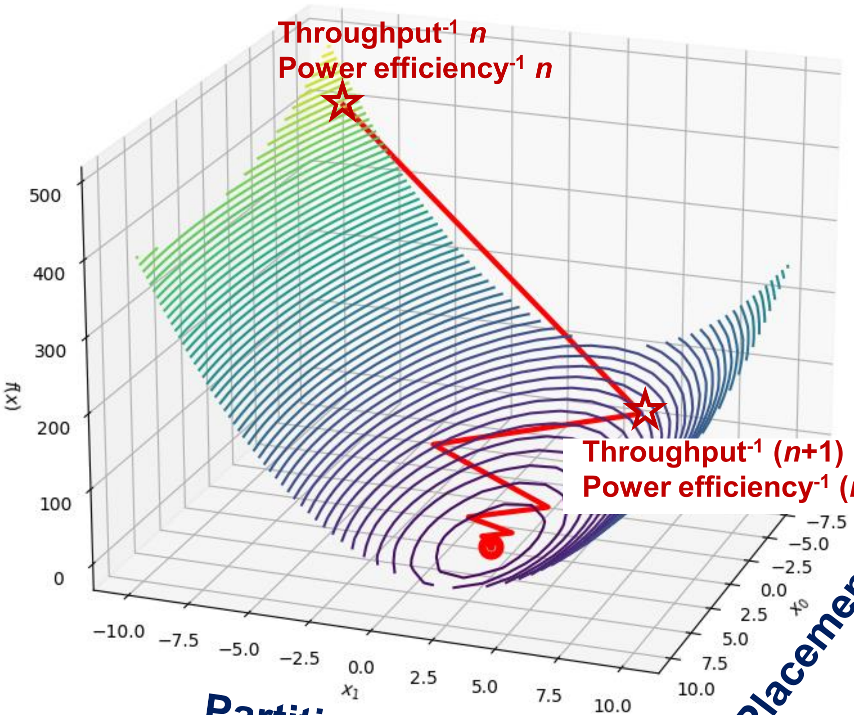
(1) Throughput *n* (GOPS)

(2) Power efficiency *n* (GOPS/W)

Problem to Solve

(1) Throughput^{-1}

(2) Power efficiency^{-1}



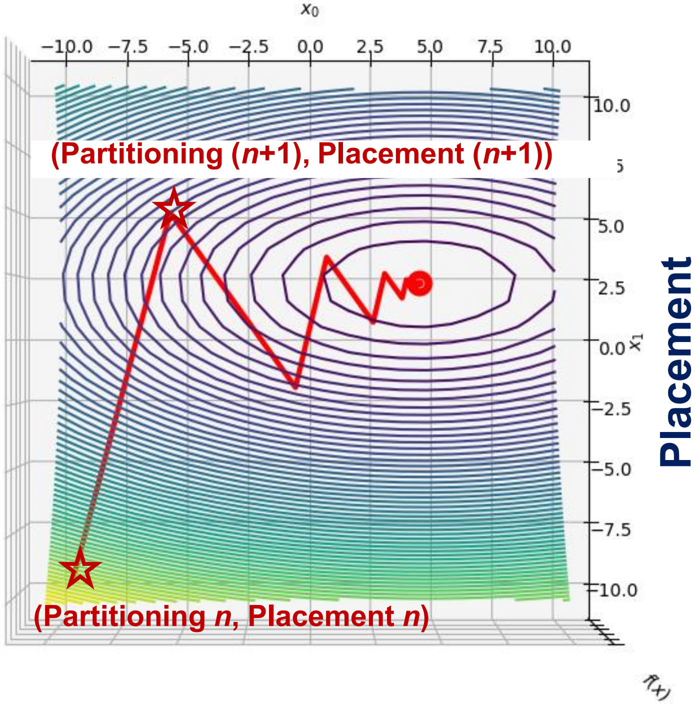
Throughput^{-1} *n*
Power efficiency^{-1} *n*

Throughput^{-1} (*n*+1)
Power efficiency^{-1} (*n*+1)

Partitioning

Placement

Partitioning



(Partitioning (*n*+1), Placement (*n*+1))

(Partitioning *n*, Placement *n*)

Placement

Design Constraints

❑ Optimize with respect to partition and placement only

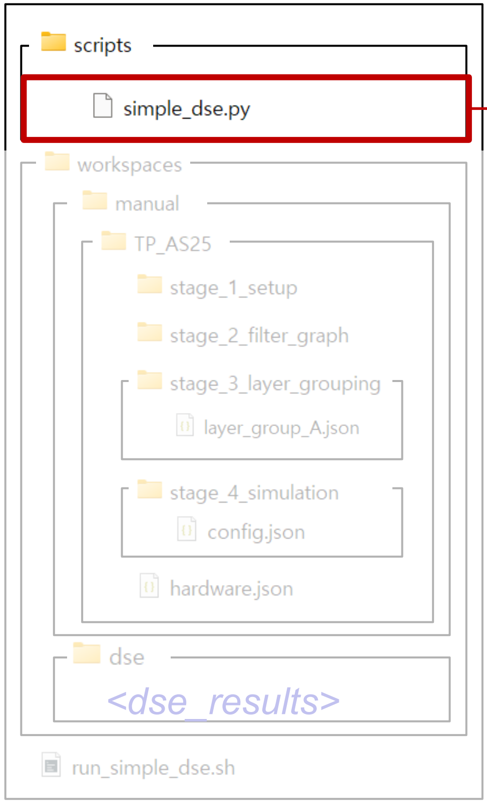
- You are ***not*** allowed to modify any other settings, e.g., hardware, layer group configurations etc.
  - ✓ Otherwise, it won’t be considered for evaluation

❑ Run *no* additional simulations (e.g., those using NetTLMSim) *inside* the partitioning/placement configuration selection

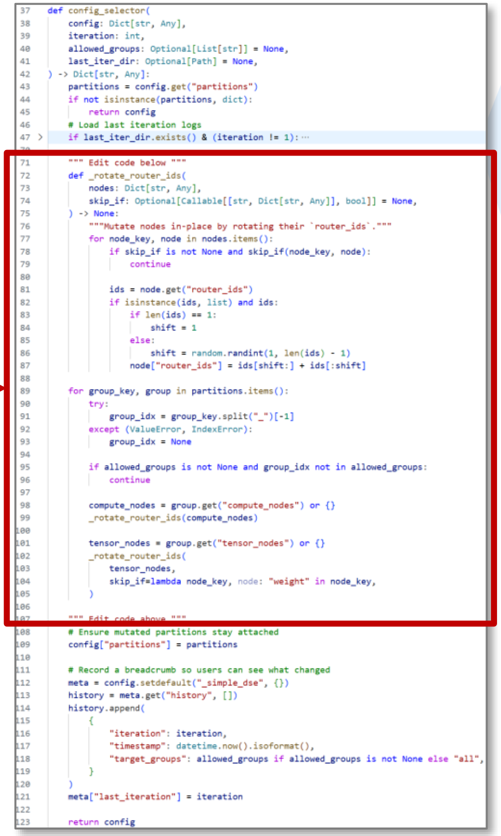
- Only a single simulation run is allowed per iteration.

Design Constraints

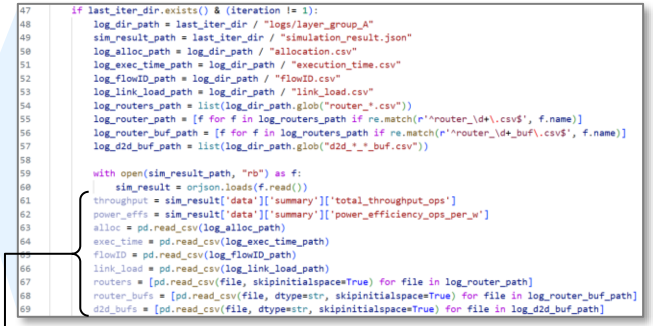
- ❑ Modify **only** (a part of) the body of ***simple_dse.py***
  - You are allowed to change ***only*** the paragraphs between ***"Edit code below"*** and ***"Edit code above"***



```
scripts
    simple_dse.py
workspaces
    manual
        TP_AS25
            stage_1_setup
            stage_2_filter_graph
            stage_3_layer_grouping
                layer_group_A.json
            stage_4_simulation
                config.json
            hardware.json
    dse
        <dse_results>
run_simple_dse.sh
```



```python
def config_selector(
    config: Dict[str, Any],
    iteration: int,
    allowed_groups: Optional[List[str]] = None,
    last_iter_dir: Optional[Path] = None,
) -> Dict[str, Any]:
    partitions = config.get("partitions")
    if not isinstance(partitions, dict):
        return config
    # Load last iteration logs
    if last_iter_dir.exists() & (iteration != 1):···

    """ Edit code below """
    def _rotate_router_ids(
        nodes: Dict[str, Any],
        skip_if: Optional[Callable[[str, Dict[str, Any]], bool]] = None,
    ) -> None:
        """Mutate nodes in-place by rotating their `router_ids`."""
        for node_key, node in nodes.items():
            if skip_if is not None and skip_if(node_key, node):
                continue

            ids = node.get("router_ids")
            if isinstance(ids, list) and ids:
                if len(ids) == 1:
                    shift = 1
                else:
                    shift = random.randint(1, len(ids) - 1)
                node["router_ids"] = ids[shift:] + ids[:shift]

    for group_key, group in partitions.items():
        try:
            group_idx = group_key.split("_")[-1]
        except (ValueError, IndexError):
            group_idx = None

        if allowed_groups is not None and group_idx not in allowed_groups:
            continue

        compute_nodes = group.get("compute_nodes") or {}
        _rotate_router_ids(compute_nodes)

        tensor_nodes = group.get("tensor_nodes") or {}
        _rotate_router_ids(
            tensor_nodes,
            skip_if=lambda node_key, node: "weight" in node_key,
        )

    """ Edit code above """
    # Ensure mutated partitions stay attached
    config["partitions"] = partitions

    # Record a breadcrumb so users can see what changed
    meta = config.setdefault("_simple_dse", {})
    history = meta.get("history", [])
    history.append(
        {
            "iteration": iteration,
            "timestamp": datetime.now().isoformat(),
            "target_groups": allowed_groups if allowed_groups is not None else "all",
        }
    )
    meta["last_iteration"] = iteration

    return config
```



```python
    if last_iter_dir.exists() & (iteration != 1):
        log_dir_path = last_iter_dir / "logs/layer_group_A"
        sim_result_path = last_iter_dir / "simulation_result.json"
        log_alloc_path = log_dir_path / "allocation.csv"
        log_exec_time_path = log_dir_path / "execution_time.csv"
        log_flowID_path = log_dir_path / "flowID.csv"
        log_link_load_path = log_dir_path / "link_load.csv"
        log_routers_path = list(log_dir_path.glob("router_*.csv"))
        log_router_path = [f for f in log_routers_path if re.match(r'^router_\d+\.csv$', f.name)]
        log_router_buf_path = [f for f in log_routers_path if re.match(r'^router_\d+_buf\.csv$', f.name)]
        log_d2d_buf_path = list(log_dir_path.glob("d2d_*_*_buf.csv"))

        with open(sim_result_path, "rb") as f:
            sim_result = orjson.loads(f.read())
        throughput = sim_result['data']['summary']['total_throughput_ops']
        power_effs = sim_result['data']['summary']['power_efficiency_ops_per_w']
        alloc = pd.read_csv(log_alloc_path)
        exec_time = pd.read_csv(log_exec_time_path)
        flowID = pd.read_csv(log_flowID_path)
        link_load = pd.read_csv(log_link_load_path)
        routers = [pd.read_csv(file, skipinitialspace=True) for file in log_router_path]
        router_bufs = [pd.read_csv(file, dtype=str, skipinitialspace=True) for file in log_router_buf_path]
        d2d_bufs = [pd.read_csv(file, dtype=str, skipinitialspace=True) for file in log_d2d_buf_path]
```

Using these variables to utilize the results from the previous iteration

Evaluation

- ❑ Submission completeness (10pt)
  - Reproducibility
- ❑ Optimality (40pt)
  - Final throughput and power efficiency
- ❑ DSE efficacy (30pt)
  - Sampling efficiency
  - General applicability
- ❑ Documentation (20pt)
  - Covering all those mentioned in the above

Submission Completeness

❑ Make sure that your submission is **complete**

- In other words, it should be possible to reproduce your design together with the claimed throughput and power efficiency using **only** the files that you submitted by the submission deadline

Optimality

- ❑ Evaluated both absolutely and relatively
  - In other word, the quantity as well as the ranking matters
- ❑ Evaluate separately for throughput and power efficiency

DSE Efficacy

❑ Sampling efficiency

- Defined as the final quantity divided by the total number of iterations
  - ✓ Evaluates *how many simulation runs are required to reach the final throughput and power efficiency*
- Considered for **both** *training* (fine-turning) and *inference* in the case of AI/ML (and separately, if necessary)
- The fewer iterations, the better sampling efficiency

❑ General applicability

- Evaluates whether the proposed idea is generally applicable
  - ✓ For example, different layer group configurations (including layer groups 0~2)
- May be further verified by extra hardware/layer group configurations

Submission

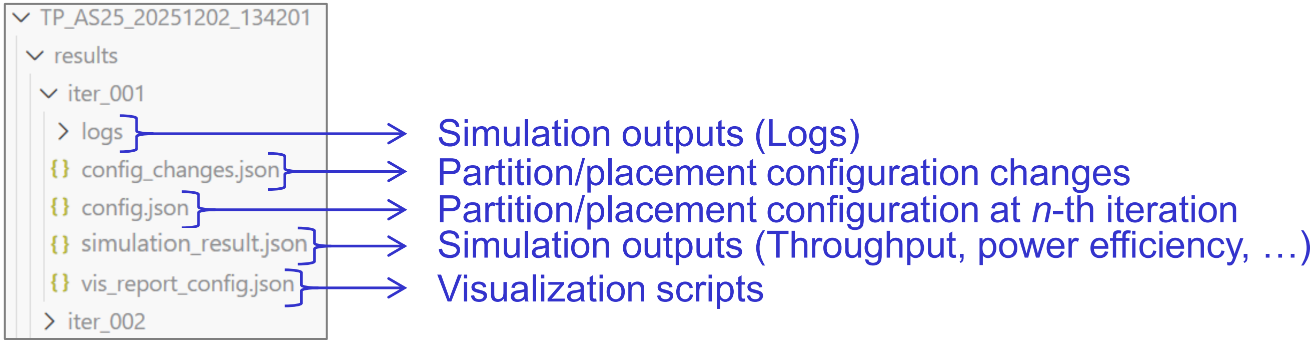
- ❑ Deadline
  - **Dec. 12 (Fri), 10:00** GMT+9
- ❑ **Only one zip file** submission **per team** including the following files:
  - Source code (all those needed to reproduce your results)
    - ✓ Except all those provided by TA (e.g., reference project)
  - Documentation (PPT)
    - ✓ Including the explanation of the above source code
- ❑ Upload the zip file to the Ecampus
  - Plus, send to chesterku2013@gmail.com as a *backup*
- ❑ You can post questions in the Ecampus (Q&A)
- ❑ Delayed submission will result in penalty!

Appendix A:
Running a DSE

Runing a DSE

❑ Start the DSE

- Open the terminal with ***'Ctrl + `'***
- Type the following command and press ***'Enter'***
  - ✓ `./run_simple_dse.sh TP_AS25 --layer-groups A --iterations N`
  - ✓ Set iteration count to ***N***



```
TP_AS25_20251202_134201
  results
    iter_001
      logs
      {} config_changes.json
      {} config.json
      {} simulation_result.json
      {} vis_report_config.json
    iter_002
```

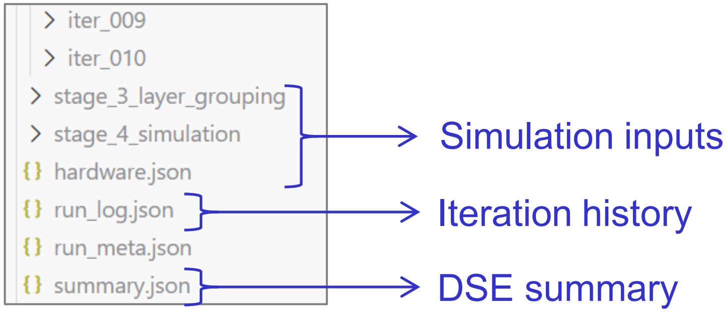
Simulation outputs (Logs)

Partition/placement configuration changes

Partition/placement configuration at *n*-th iteration

Simulation outputs (Throughput, power efficiency, …)

Visualization scripts



```
    iter_009
    iter_010
  stage_3_layer_grouping
  stage_4_simulation
  {} hardware.json
  {} run_log.json
  {} run_meta.json
  {} summary.json
```

Simulation inputs

Iteration history

DSE summary