Lake Verification

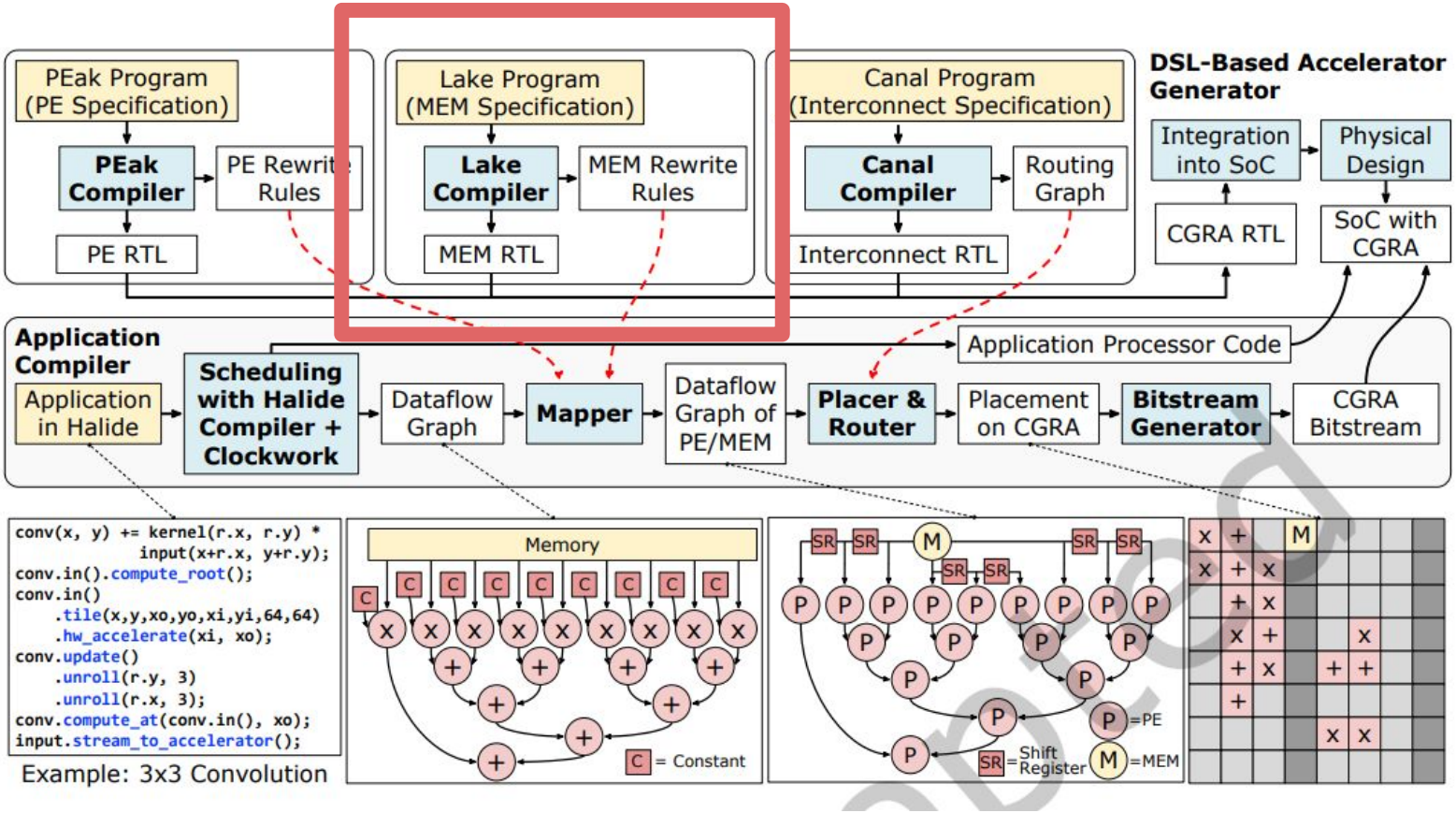
@peggylin

2022/10/7

Agenda

- Verification test
  - Introduction
  - Integration test, unit test
  - Current task
- Lake feature
  - Affine controller
  - Config file
  - Chaining
  - Mode

Hardware **domain-specific** language



Verification

- For a complete set of tests, we need to know:
  - All **features** that system has
  - Different types of **input**
  - Expected **output**

Verification

- Automated testing
  - Execute test plan by script
  - We use ***pytest*** as our test runner

Integration test vs Unit test

- **Integration test**
  - Testing of whole system
  - **Right** result: Everything work perfectly! :)
  - **Wrong** result: Hard to diagnose which part of the system failed… :(
  - checks that components in your application operate with each other.
  - **Slower** since it have to generate RTL
- **unit test**
  - check **single small component** in system
  - isolate what is broken from the system
  - **Faster** debug process

Integration test vs Unit test

- **Integration test**
  - Testing of whole system
  - **Right** result: Everything work perfectly! :)
  - **Wrong** result: Hard to diagnose which part of the system failed… :(
  - checks that components in your application operate with each other.
  - **Slower** since it have to generate RTL
- **unit test**
  - check **single small component** in system
  - isolate what is broken from the system
  - **Faster** debug process

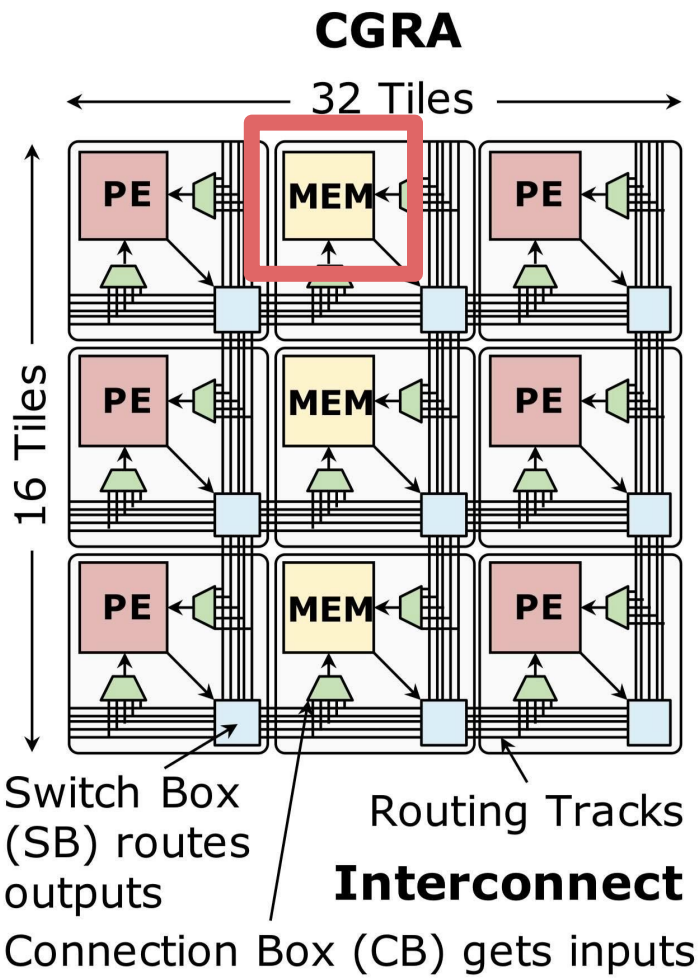
We need **BOTH**!

Current Tests

- **Unit test** for small modules exist
- But current **integration test** for **only one MEM tile** is outdated
  - New infrastructure change the interface
  - We use **clockwork test** previously, which would generate several MEM tiles and mock PE tiles to test the interaction between each other
  - Longer test time
  - Too complicate to debug

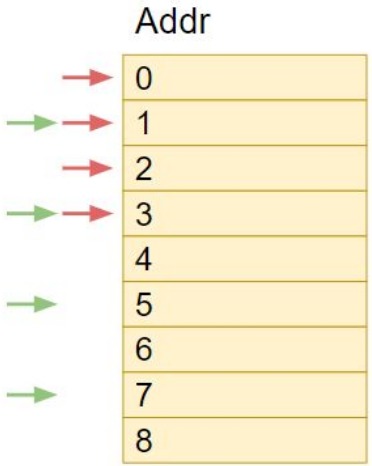
Current Tests

- Other version support **sparsity** application
  - Current **unit tests** don’t support sparsity
  - **interface between unit tests** need to change

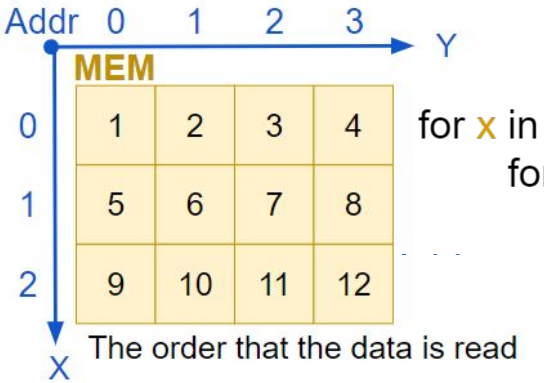

CGRA
32 Tiles
16 Tiles
PE
MEM
PE
PE
MEM
PE
PE
MEM
PE
Switch Box
(SB) routes
outputs
Routing Tracks
Interconnect
Connection Box (CB) gets inputs

Target Application

- Accelerator of **Dense Linear Algebra Applications**
  - Frequent **reuse** of the data
  - **Affine access pattern** (can be written as **for-loop**)



1-D Affine Access Pattern

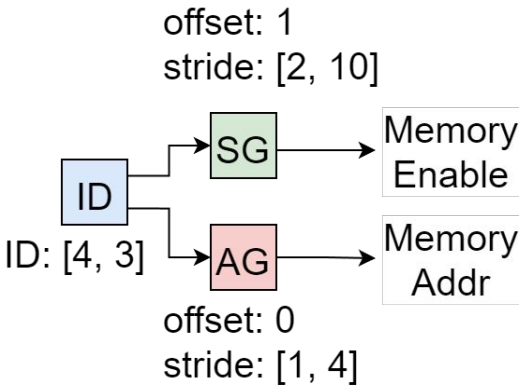



```
for x in [0,3):
    for y in [0,4):
        read MEM[x][y]
```

2-D Affine Access Pattern

Affine Controller

- **Memory controllers** to **match these patterns**
  - **Iteration Domain (ID)**: for-loop (range of the memory operations)
  - **Address Generator (AG)**: memory address
  - **Schedule Generator (SG)**: read/write enable



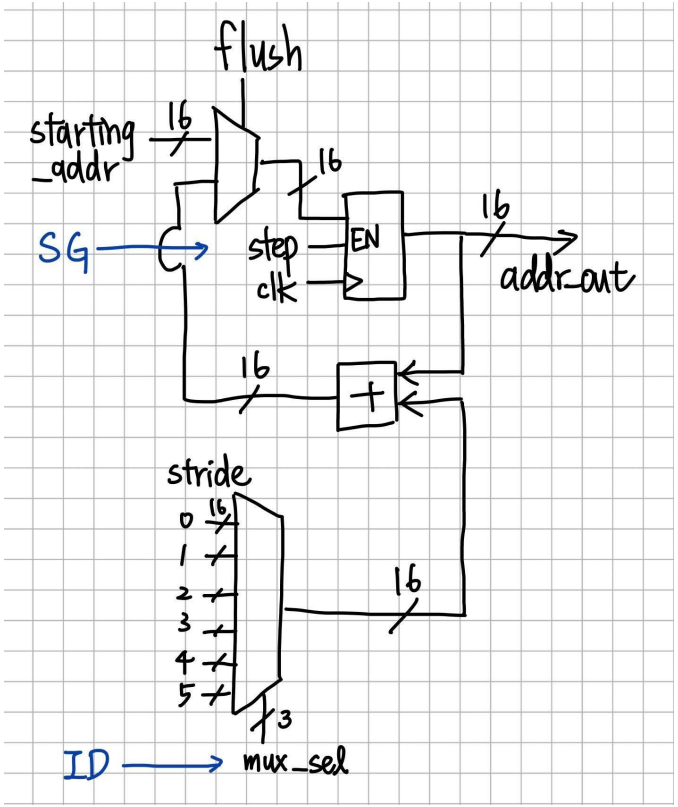
Address Generator (AG)

- **Recurrence** relation

addr_out =
stride_x***x** + stride_y*y + starting_addr

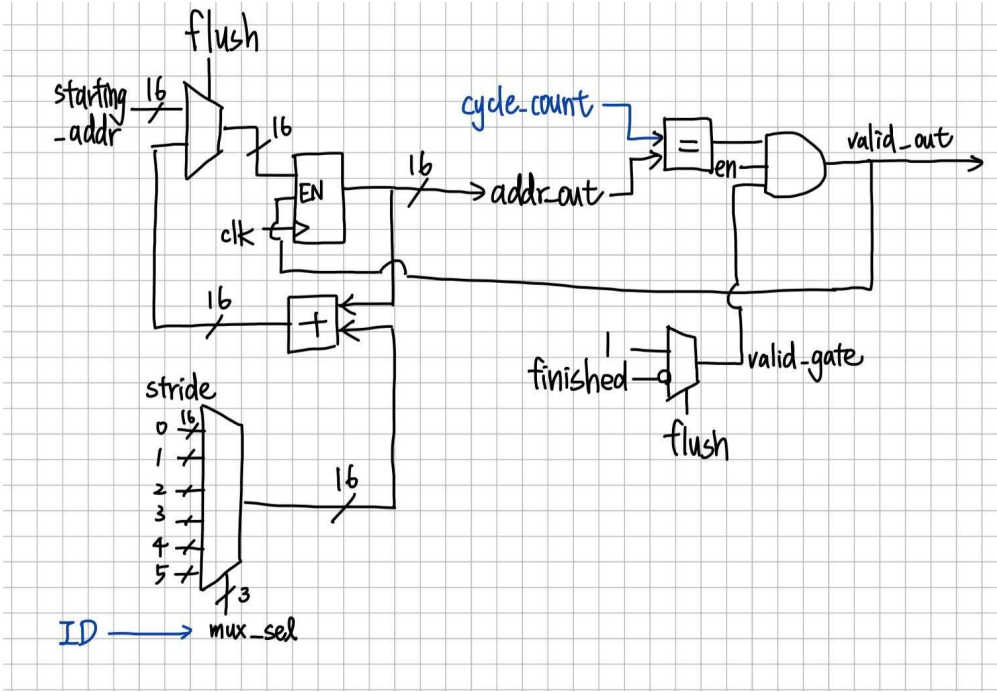
addr_out =
current_addr + strides[**mux_sel**]

- **enable** signal (step) from SG
- **mux_sel** from ID

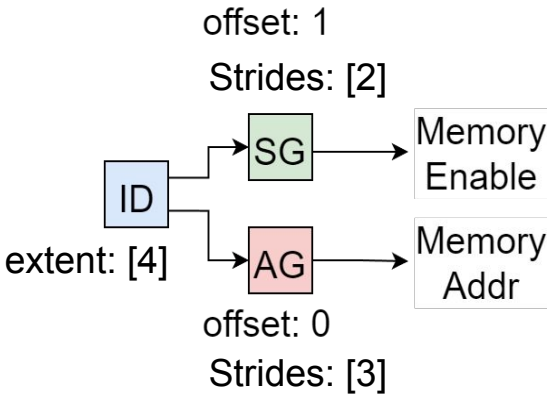


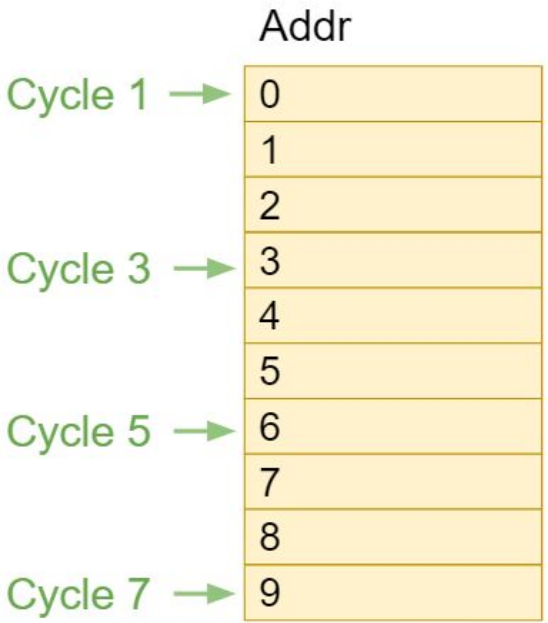
Schedule Generator (SG)

- **Static** scheduling
  - no VALID or READY signal
  - **Cycle accurate schedule**
  - **cycle counter** for each MEM tile



Memory controller example





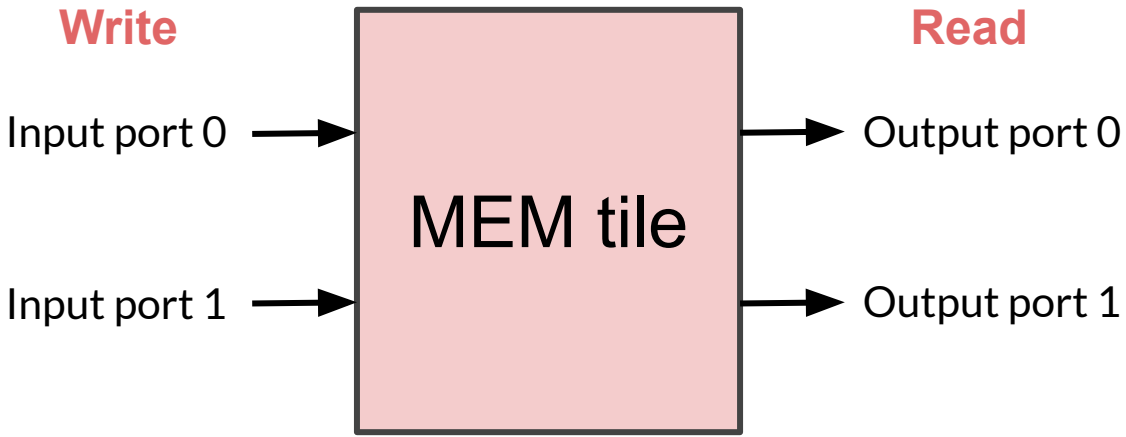
ConfigRegAttr

- Input with additional attribute **ConfigRegAttr**
  - could be **reconfigured directly by the compiler**
  - *starting_addr*, *stride*, *dimensionality*, *extent*….
- Where **reconfigurable** array comes from

```
self._starting_addr = self.input("starting_addr", self.config_width)
self._starting_addr.add_attribute(ConfigRegAttr("Starting address of address generator"))
self._starting_addr.add_attribute(FormalAttr(f"{self._starting_addr.name}", FormalSignalConstraint.SOLVE))
```

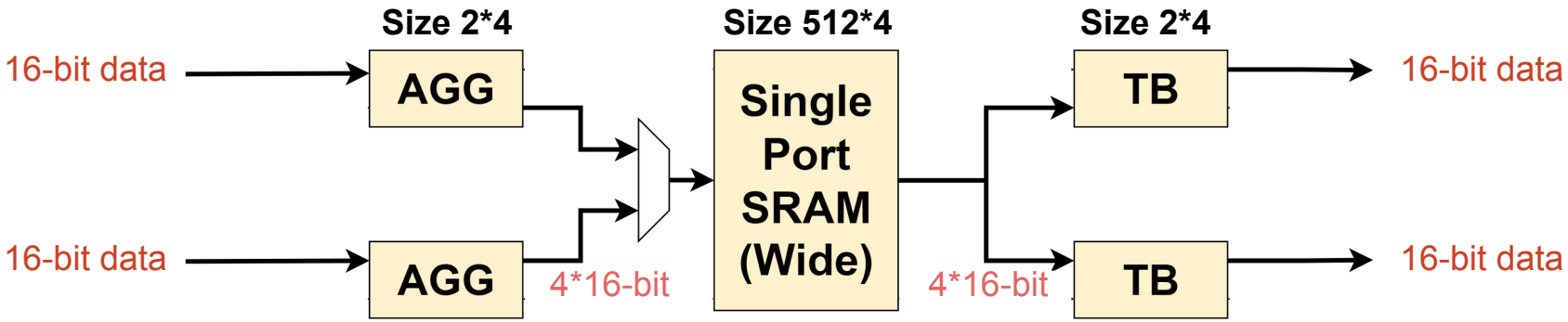
MEM tile

- Two read ports + two write ports

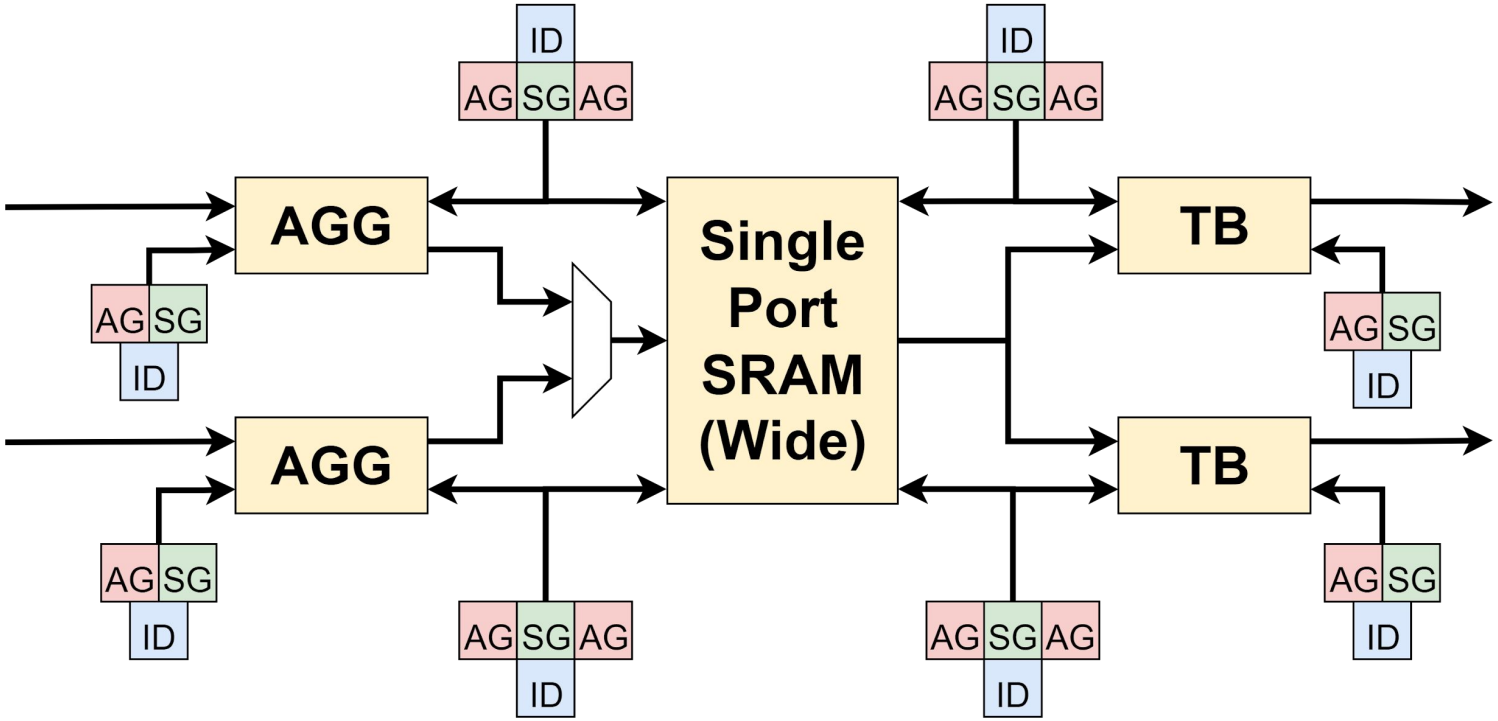


Implement Multi-ported memory

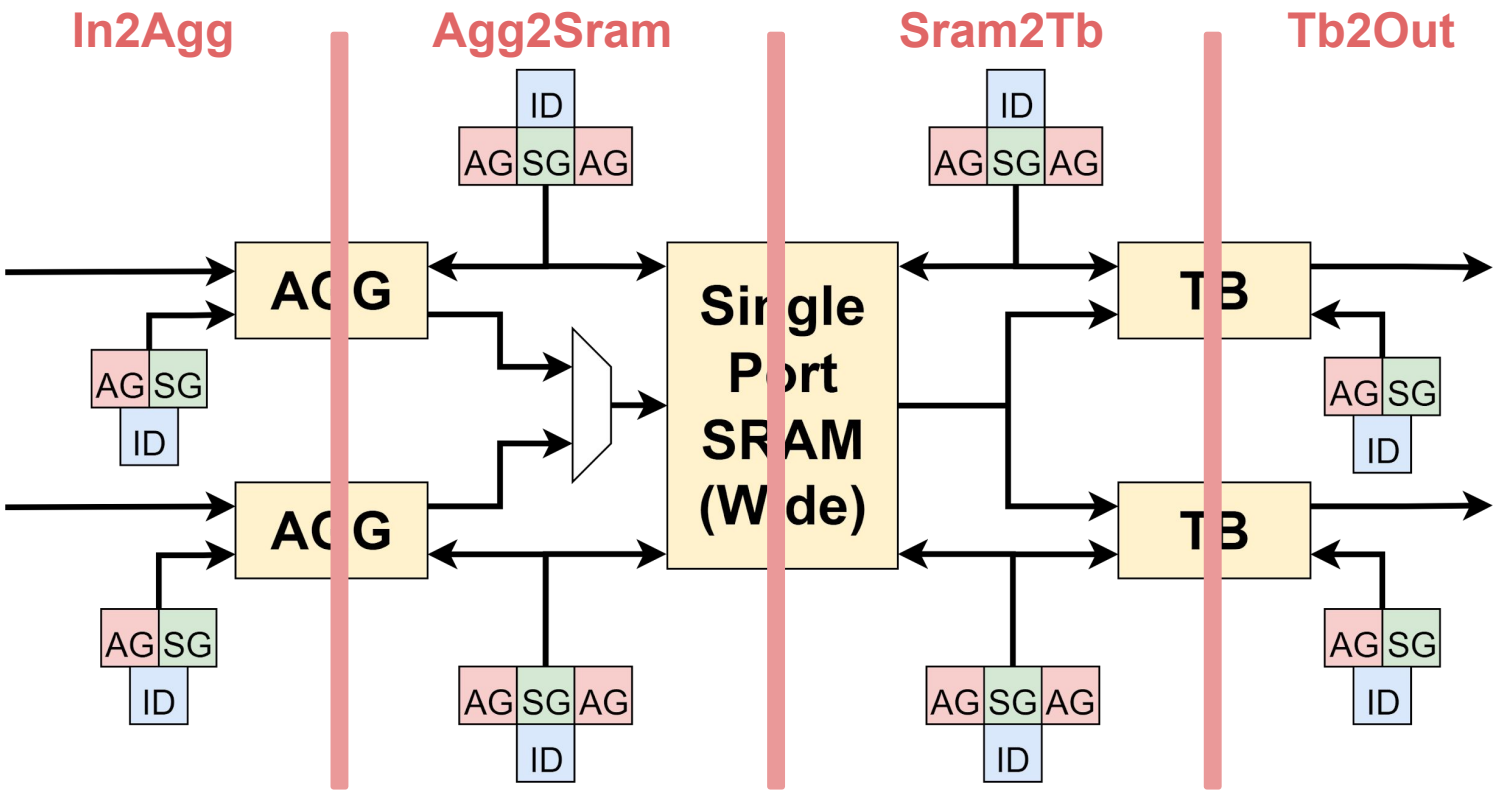
- Implement a **multi-ported** memory with a **single-ported** SRAM memory
- **AGG** (SIPO) → **single port SRAM** → **TB** (PISO)



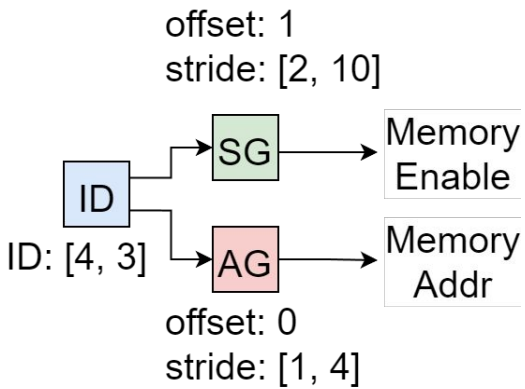
Affine controller at each interface

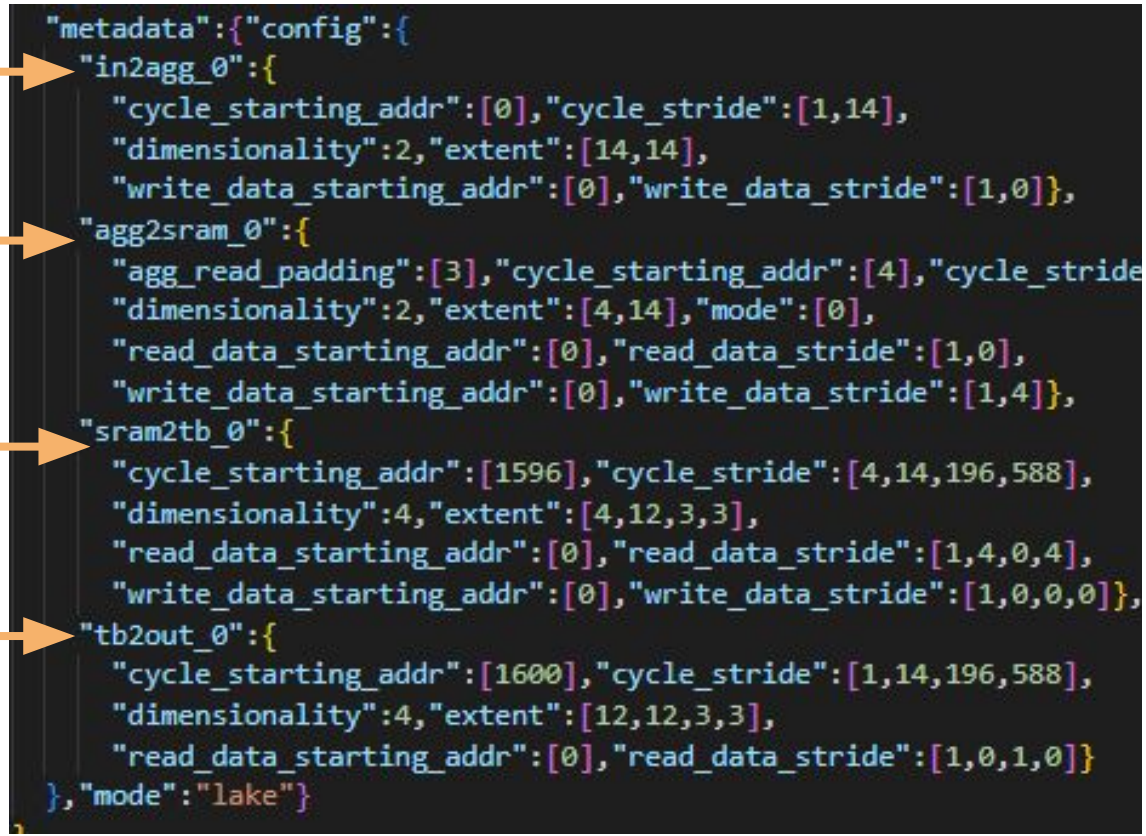


Affine controller at each interface

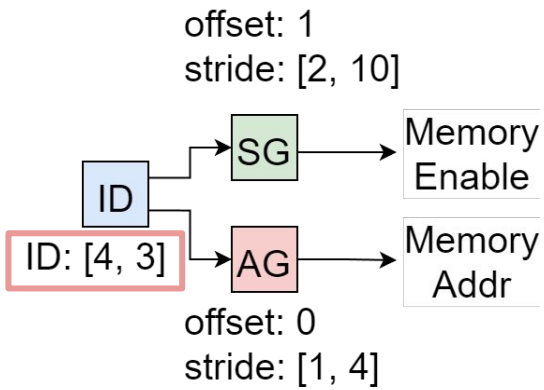


Config file


offset: 1
stride: [2, 10]
SG
Memory Enable
ID
ID: [4, 3]
AG
Memory Addr
offset: 0
stride: [1, 4]

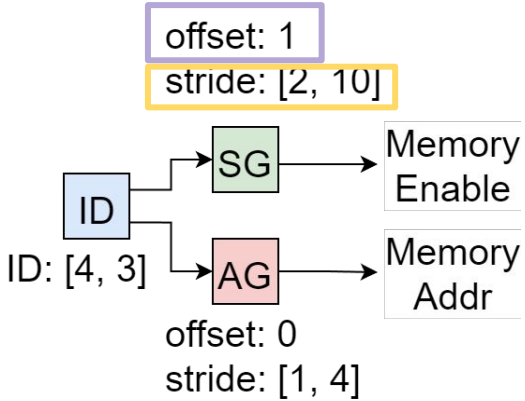

"metadata":{"config":{
"in2agg_0":{
"cycle_starting_addr":[0],"cycle_stride":[1,14],
"dimensionality":2,"extent":[14,14],
"write_data_starting_addr":[0],"write_data_stride":[1,0]},
"agg2sram_0":{
"agg_read_padding":[3],"cycle_starting_addr":[4],"cycle_stride
"dimensionality":2,"extent":[4,14],"mode":[0],
"read_data_starting_addr":[0],"read_data_stride":[1,0],
"write_data_starting_addr":[0],"write_data_stride":[1,4]},
"sram2tb_0":{
"cycle_starting_addr":[1596],"cycle_stride":[4,14,196,588],
"dimensionality":4,"extent":[4,12,3,3],
"read_data_starting_addr":[0],"read_data_stride":[1,4,0,4],
"write_data_starting_addr":[0],"write_data_stride":[1,0,0,0]},
"tb2out_0":{
"cycle_starting_addr":[1600],"cycle_stride":[1,14,196,588],
"dimensionality":4,"extent":[12,12,3,3],
"read_data_starting_addr":[0],"read_data_stride":[1,0,1,0]}
},"mode":"lake"}

Config file



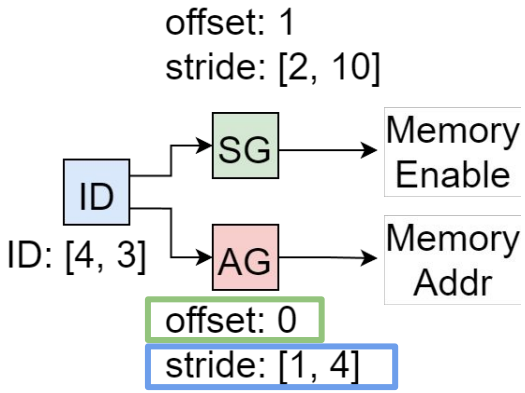

```
"metadata":{"config":{
  "in2agg_0":{
    "cycle_starting_addr":[0],"cycle_stride":[1,14],
    "dimensionality":2,"extent":[14,14],
    "write_data_starting_addr":[0],"write_data_stride":[1,0]},
  "agg2sram_0":{
    "agg_read_padding":[3],"cycle_starting_addr":[4],"cycle_stride
    "dimensionality":2,"extent":[4,14],"mode":[0],
    "read_data_starting_addr":[0],"read_data_stride":[1,0],
    "write_data_starting_addr":[0],"write_data_stride":[1,4]},
  "sram2tb_0":{
    "cycle_starting_addr":[1596],"cycle_stride":[4,14,196,588],
    "dimensionality":4,"extent":[4,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,4,0,4],
    "write_data_starting_addr":[0],"write_data_stride":[1,0,0,0]},
  "tb2out_0":{
    "cycle_starting_addr":[1600],"cycle_stride":[1,14,196,588],
    "dimensionality":4,"extent":[12,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,0,1,0]}
},"mode":"lake"}
```

Config file



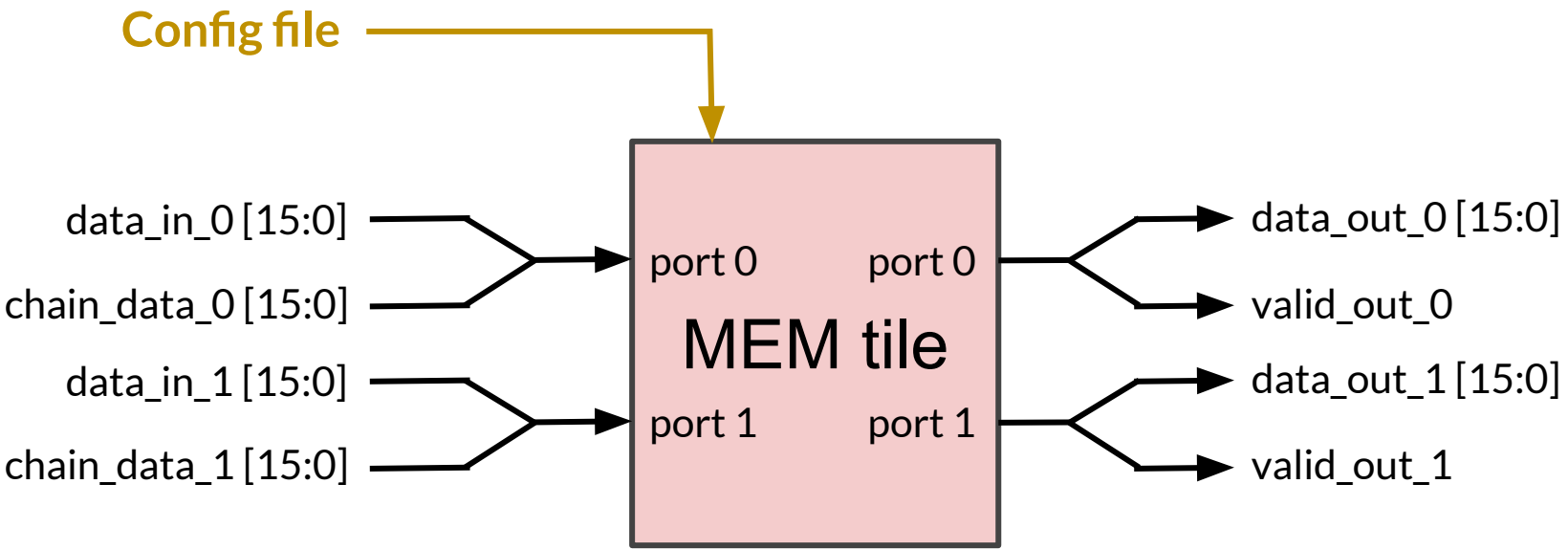

```
"metadata":{"config":{
  "in2agg_0":{
    "cycle_starting_addr":[0],"cycle_stride":[1,14],
    "dimensionality":2,"extent":[14,14],
    "write_data_starting_addr":[0],"write_data_stride":[1,0]},
  "agg2sram_0":{
    "agg_read_padding":[3],"cycle_starting_addr":[4],"cycle_stride
    "dimensionality":2,"extent":[4,14],"mode":[0],
    "read_data_starting_addr":[0],"read_data_stride":[1,0],
    "write_data_starting_addr":[0],"write_data_stride":[1,4]},
  "sram2tb_0":{
    "cycle_starting_addr":[1596],"cycle_stride":[4,14,196,588],
    "dimensionality":4,"extent":[4,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,4,0,4],
    "write_data_starting_addr":[0],"write_data_stride":[1,0,0,0]},
  "tb2out_0":{
    "cycle_starting_addr":[1600],"cycle_stride":[1,14,196,588],
    "dimensionality":4,"extent":[12,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,0,1,0]}
},"mode":"lake"}
```

Config file



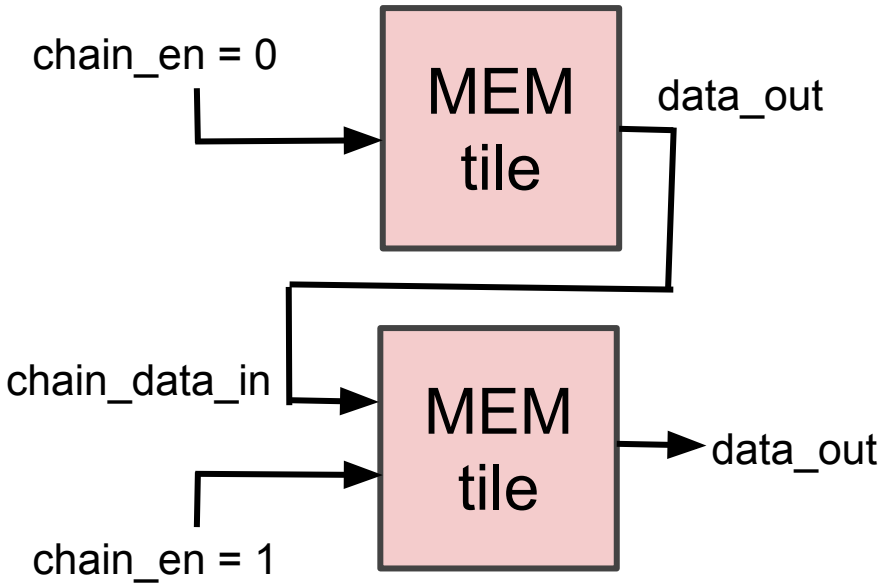

```
"metadata":{"config":{
  "in2agg_0":{
    "cycle_starting_addr":[0],"cycle_stride":[1,14],
    "dimensionality":2,"extent":[14,14],
    "write_data_starting_addr":[0],"write_data_stride":[1,0]},
  "agg2sram_0":{
    "agg_read_padding":[3],"cycle_starting_addr":[4],"cycle_stride
    "dimensionality":2,"extent":[4,14],"mode":[0],
    "read_data_starting_addr":[0],"read_data_stride":[1,0],
    "write_data_starting_addr":[0],"write_data_stride":[1,4]},
  "sram2tb_0":{
    "cycle_starting_addr":[1596],"cycle_stride":[4,14,196,588],
    "dimensionality":4,"extent":[4,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,4,0,4],
    "write_data_starting_addr":[0],"write_data_stride":[1,0,0,0]},
  "tb2out_0":{
    "cycle_starting_addr":[1600],"cycle_stride":[1,14,196,588],
    "dimensionality":4,"extent":[12,12,3,3],
    "read_data_starting_addr":[0],"read_data_stride":[1,0,1,0]}
},"mode":"lake"}
```

I/O of MEM tile



Chaining

- Chain several MEM tiles to increase capacity
- If the buffer capacity in the application exceed the capacity of single MEM tile



Onyx Optimization

- Agg2Sram
  - memory write access of most dense applications is **continuous**
  - SG simplified to **counter of 4**
  - AG can also be a counter as the input data are stored continuously
  - **read_padding** when the data number is not multiplicand of 4
- **Discontinuous** memory accesses
  - Same address but with delay
  - use a small FIFO to store and **delay** the read address and enable

Test Input

- Two input ports and two output ports
  - Use one port, use both
- Affine pattern up to 6-dim
  - Starting_addr, strides, extent
- Chain_en = 0/1
- Onyx Optimization
  - Padding corner case: 0, 2^8 (mode 0)
  - Delay corner case: 0, 2^10 (mode 2,3)
- …

Questions?