Lake:
CGRA Memory Generator for Unified Buffer

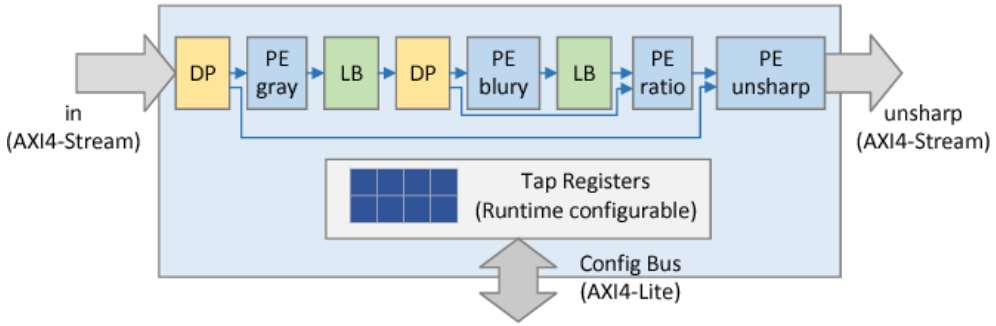
STANFORD AHA

Qiaoyi(Joey) Liu

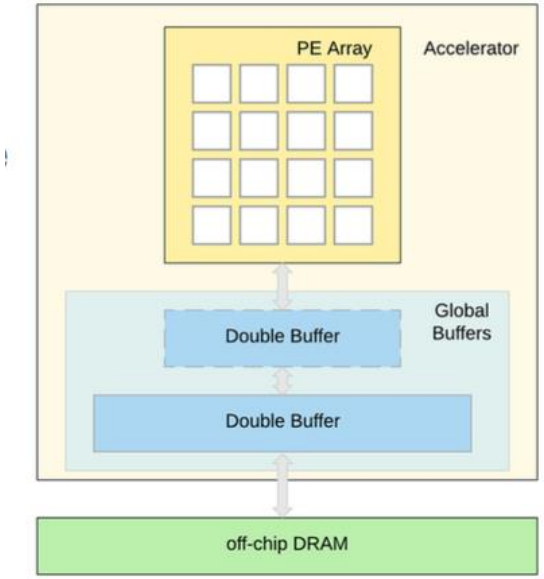
Motivation

Image Processing Hardware are built in separate methods

- Line buffer pipeline for image processing
- Double buffer pipeline for deep neural network



Line buffer Pipeline

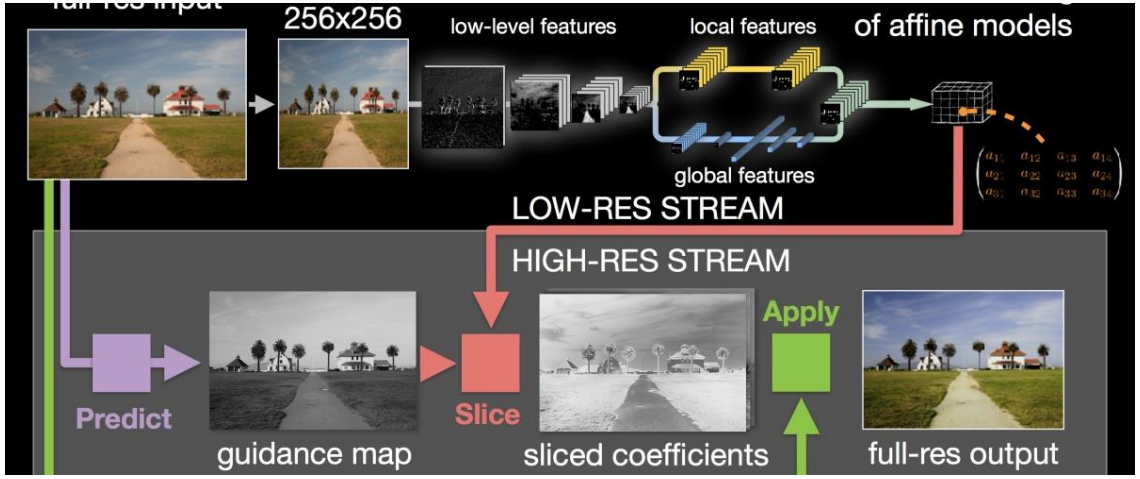


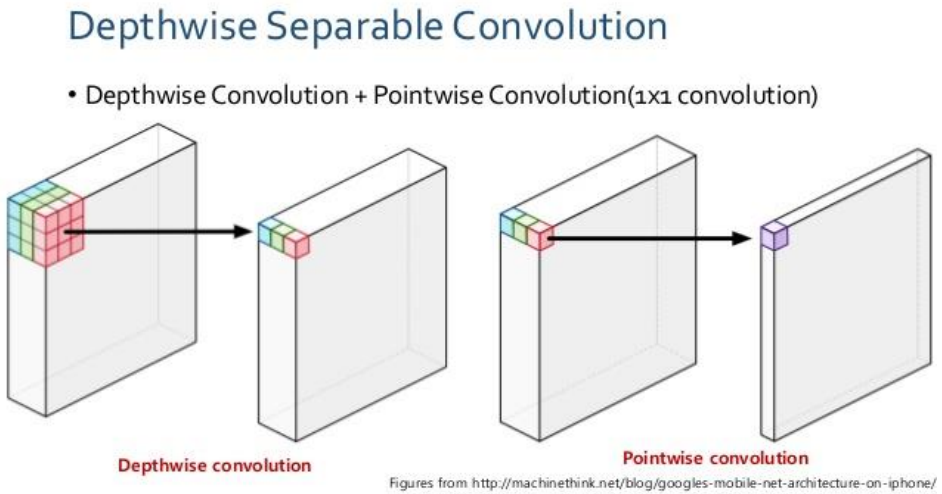
Double buffer Pipeline

Motivation

Application of NN and image processing are merging

- **HDR net:**
  Adding NN operation in image processing application
- **MobileNet:**
  Using light-weight conv layer more similar to traditional convolution





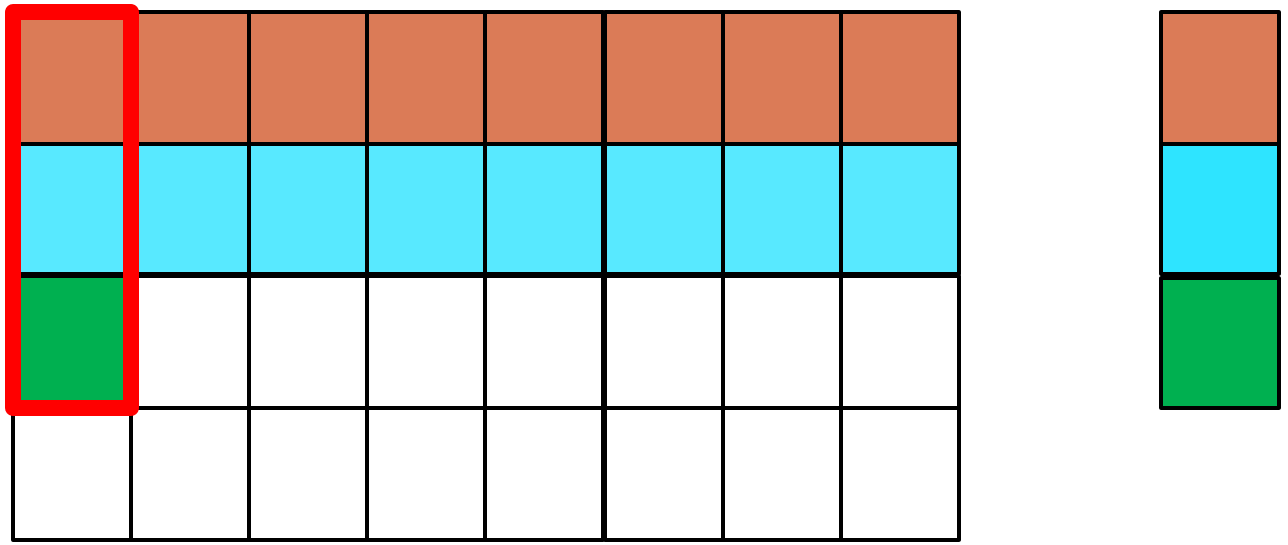
Motivation

- Why not merge the memory structure
  - unified buffer
- Manually mapping?
  - Changing of functional model
  - Modified backend hardware
- Create a generator
  - Generate mapping and schedule from functional model to HW
  - Generate the unified memory backend

Overview: Unified Buffer

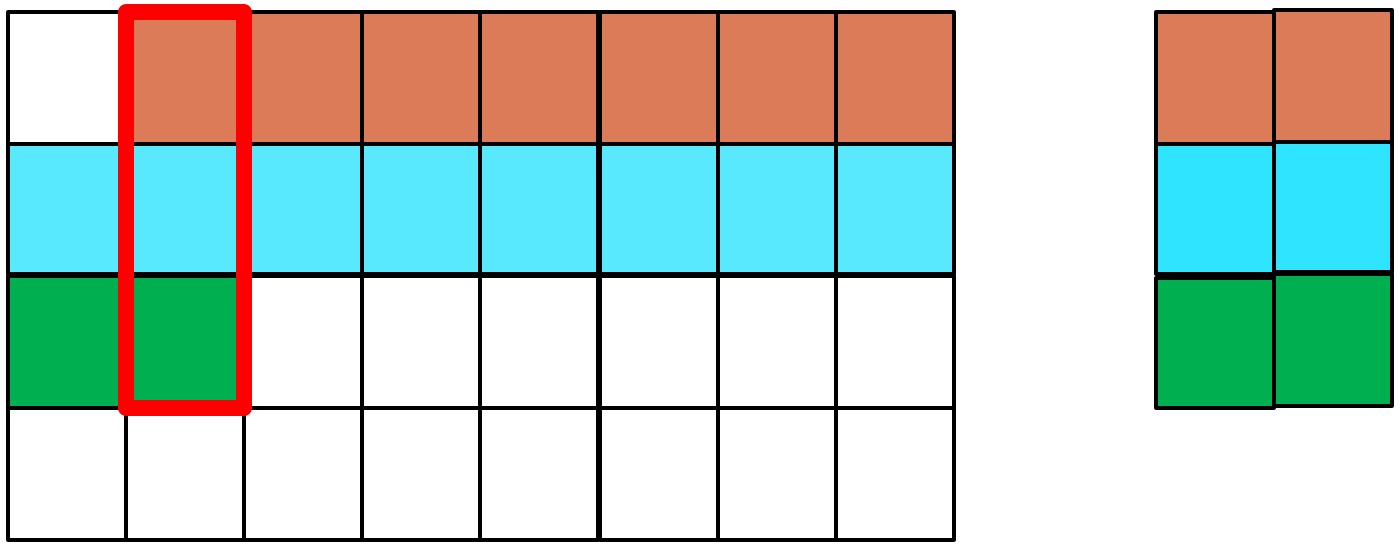
SRAM | FIFO | LINE BUFFER | DOUBLE BUFFER

Line Buffer Stencil Shift Reg



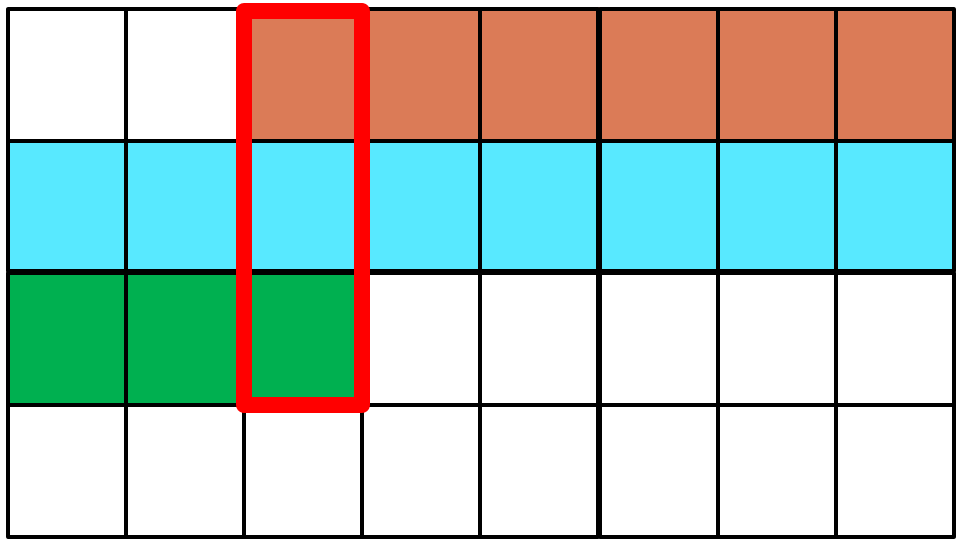
SRAM | FIFO | LINE BUFFER | DOUBLE BUFFER

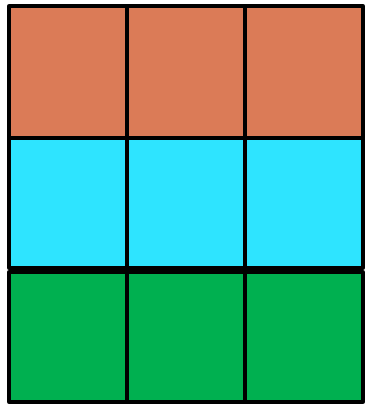
Line Buffer Stencil Shift Reg



SRAM | FIFO | LINE BUFFER | DOUBLE BUFFER

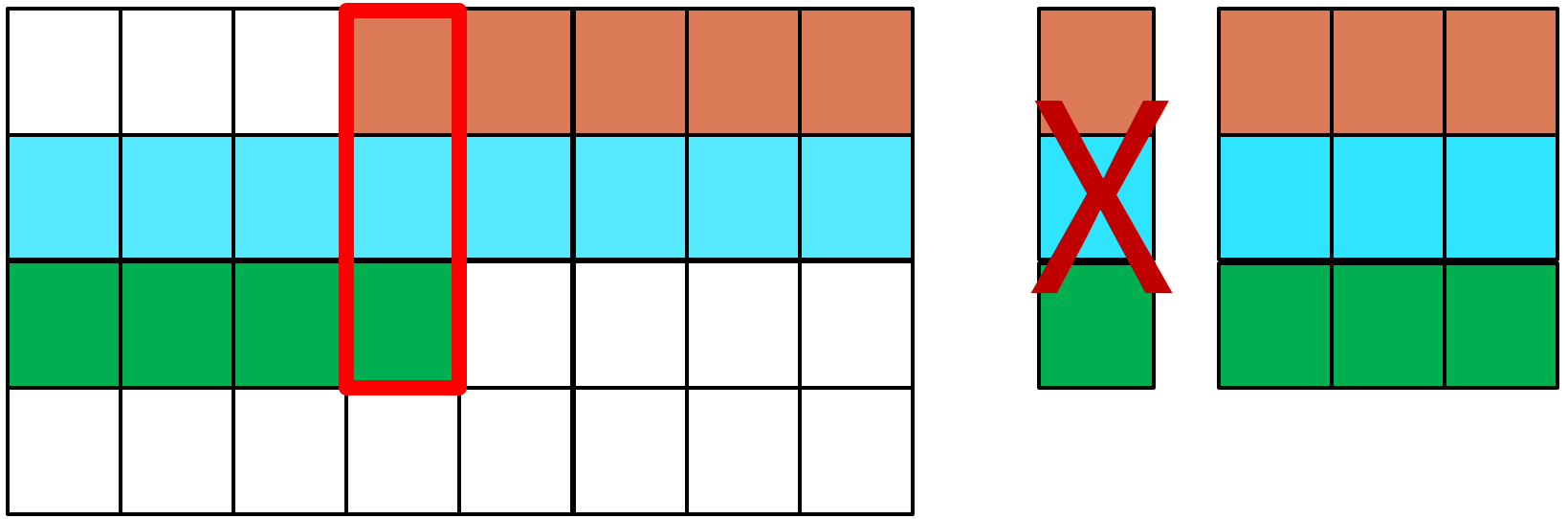
Line Buffer Stencil Shift Reg





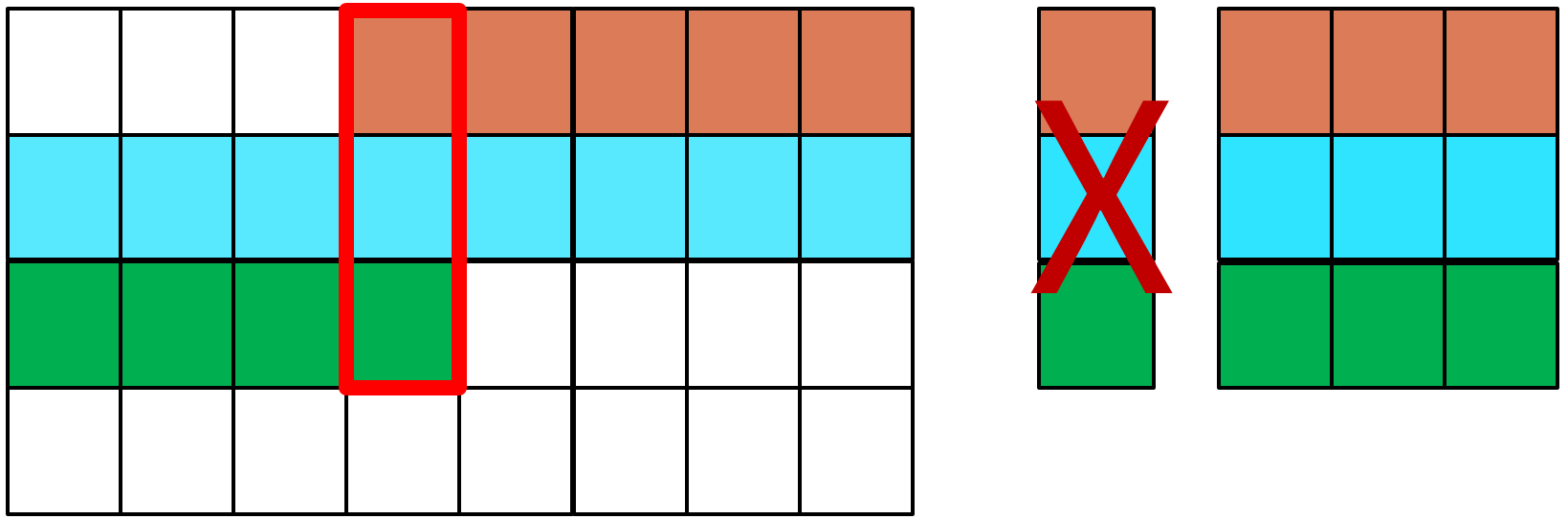
SRAM | FIFO | LINE BUFFER | DOUBLE BUFFER

Line Buffer Stencil Shift Reg



SRAM | FIFO | LINE BUFFER | DOUBLE BUFFER

Line Buffer Stencil Shift Reg



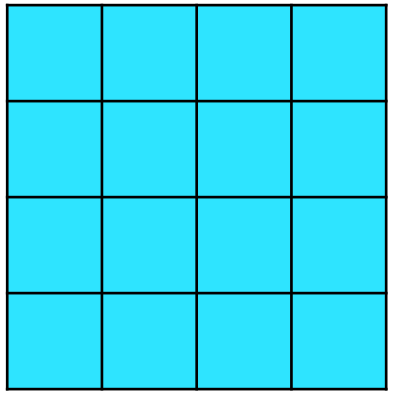
Contiguous Access !

SRAM

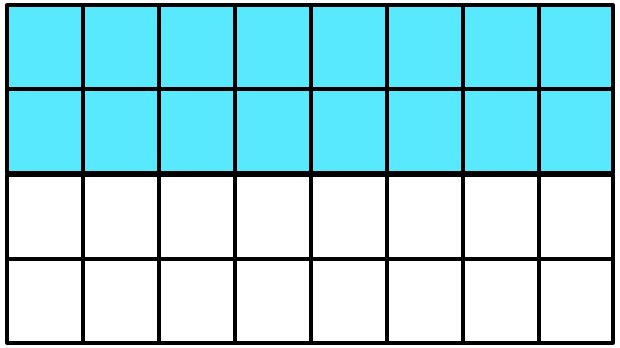
FIFO

LINE BUFFER

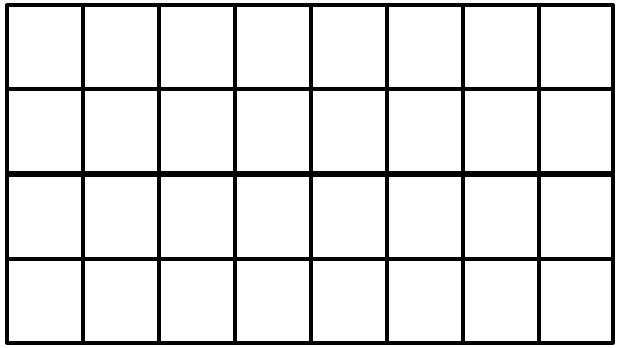
DOUBLE BUFFER



Write



Read

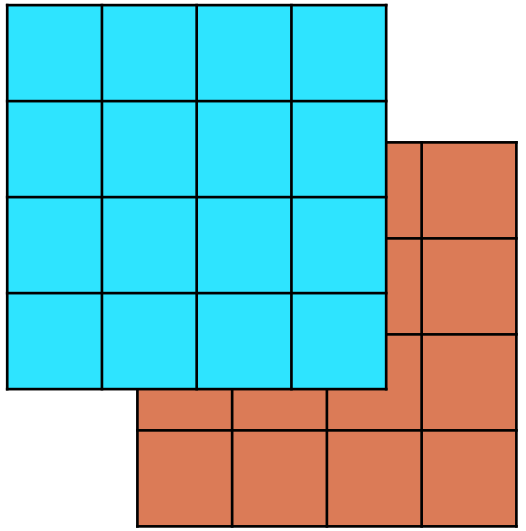


SRAM

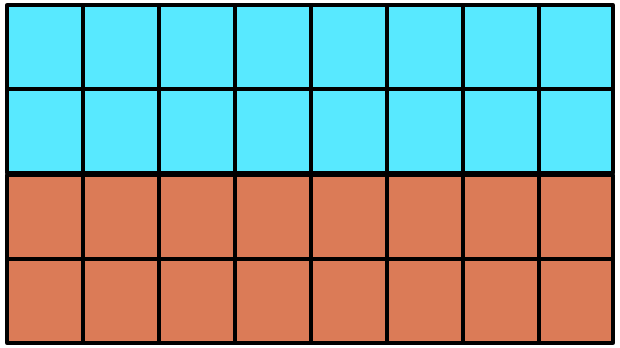
FIFO

LINE BUFFER

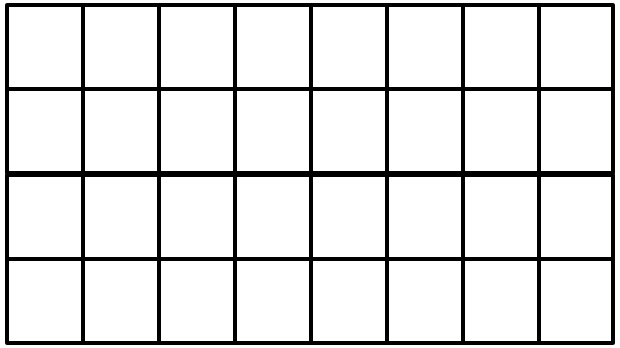
DOUBLE BUFFER



Write



Read

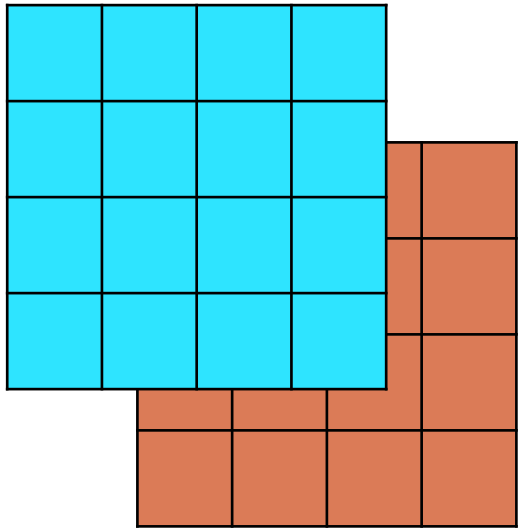


SRAM

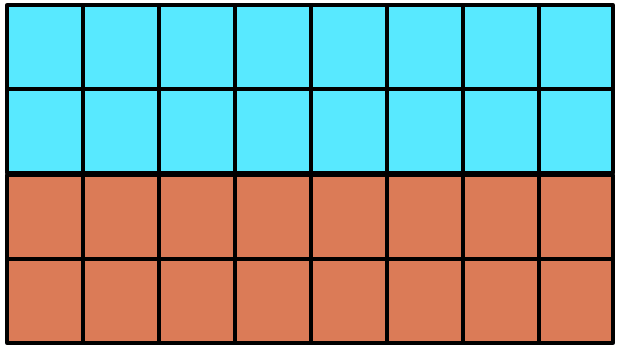
FIFO

LINE BUFFER

DOUBLE BUFFER

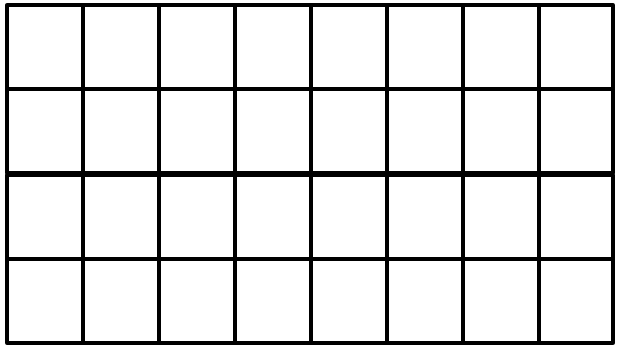


Read



Swap the buffer !

Write



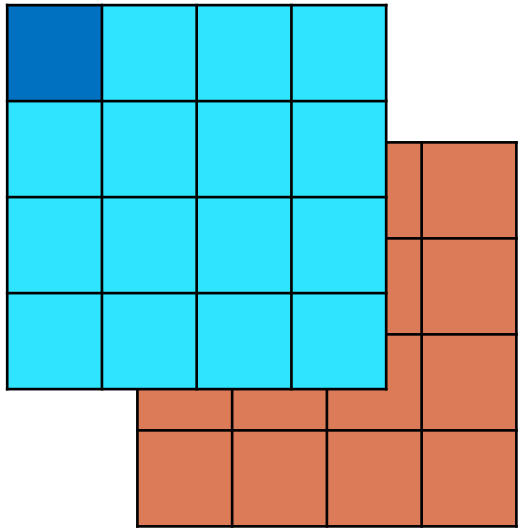
SRAM

FIFO

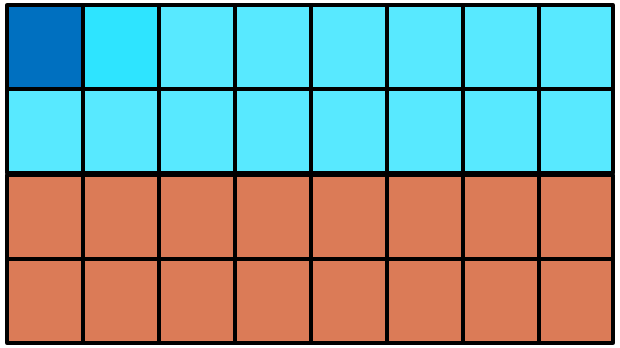
LINE BUFFER

DOUBLE BUFFER

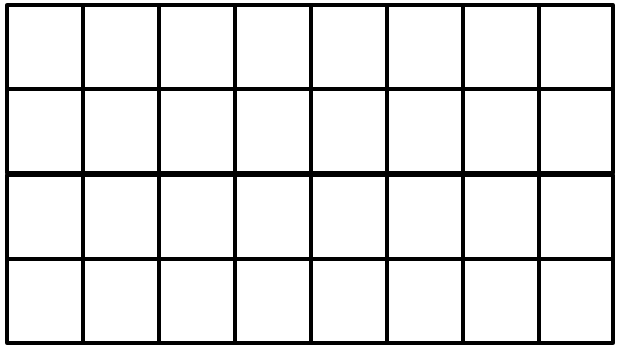
2x2 conv with 2 channels



Read



Write

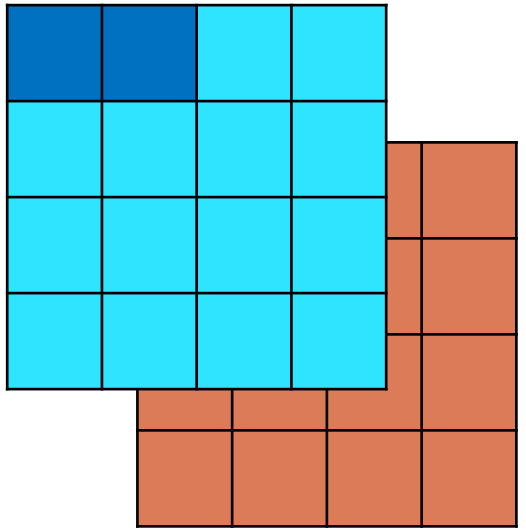


SRAM

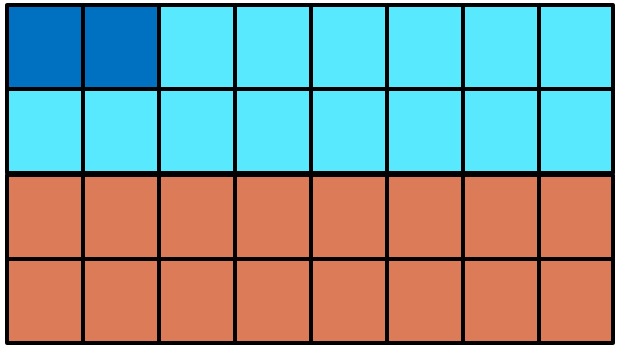
FIFO

LINE BUFFER

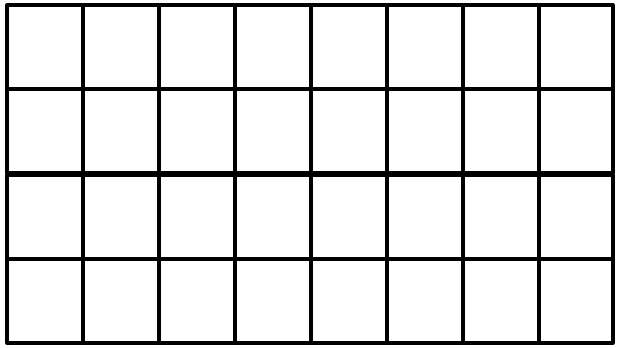
DOUBLE BUFFER



Read



Write

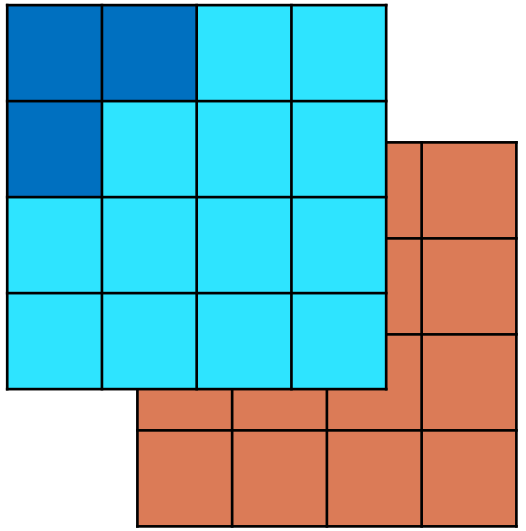


SRAM

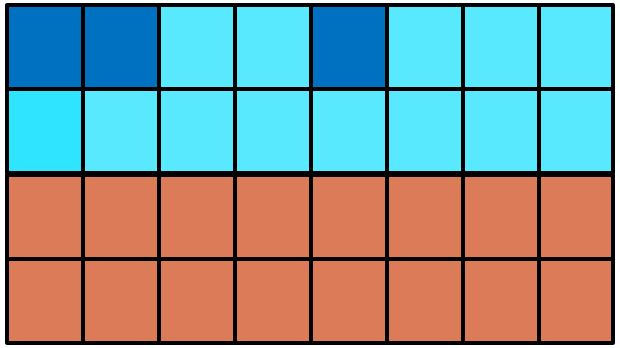
FIFO

LINE BUFFER

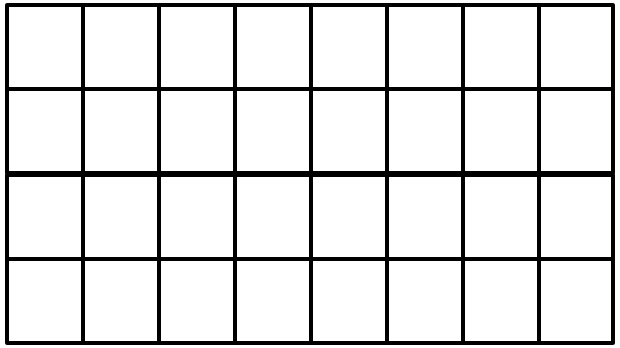
DOUBLE BUFFER



Read



Write

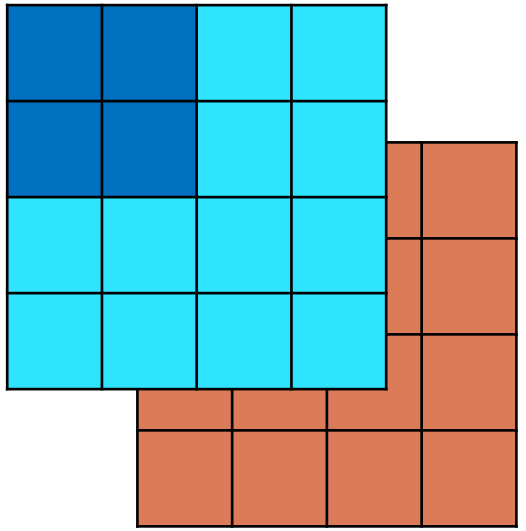


SRAM

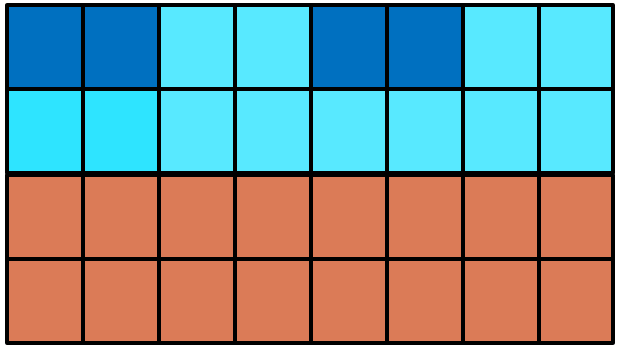
FIFO

LINE BUFFER

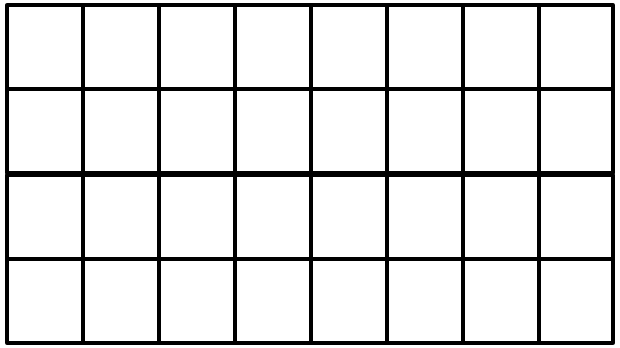
DOUBLE BUFFER



Read



Write



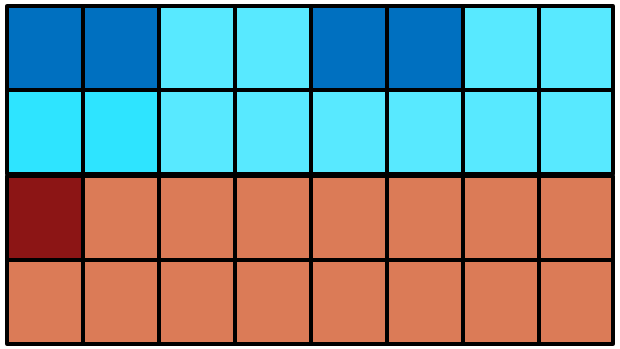
SRAM

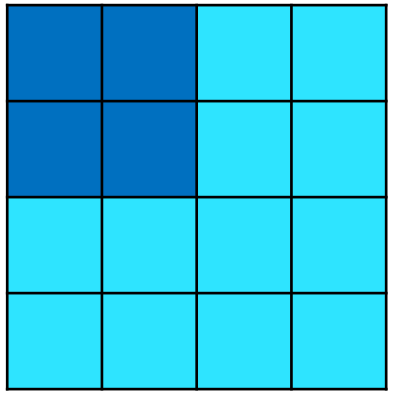
FIFO

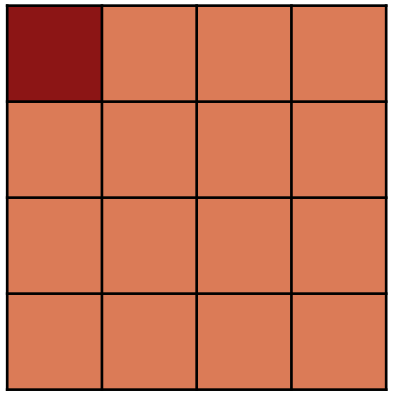
LINE BUFFER

DOUBLE BUFFER

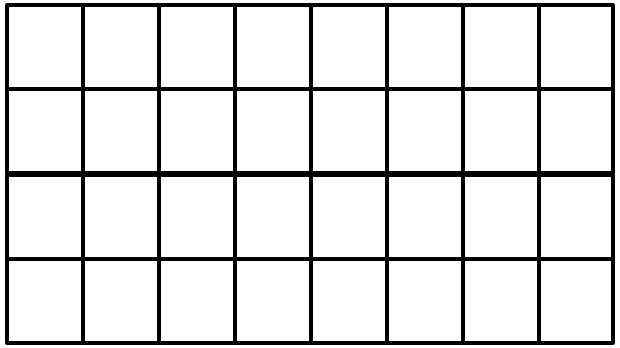
Read







Write



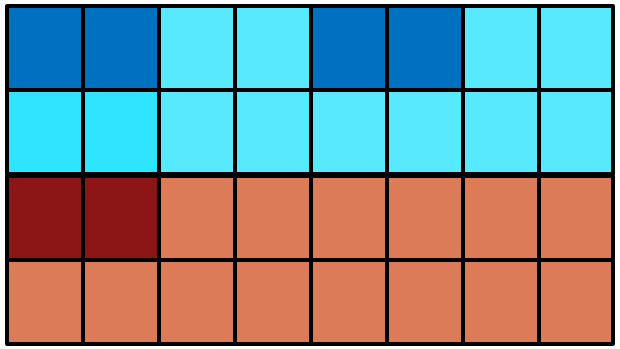
SRAM

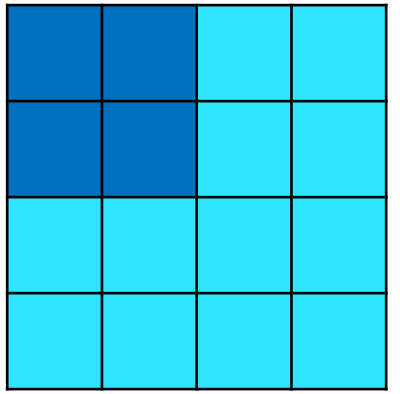
FIFO

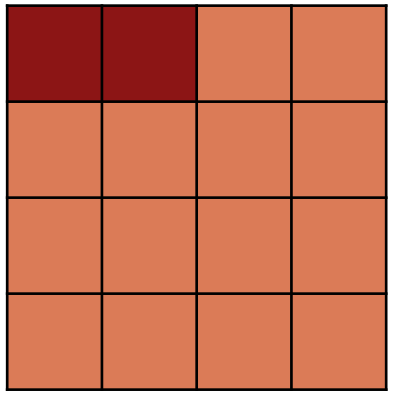
LINE BUFFER

DOUBLE BUFFER

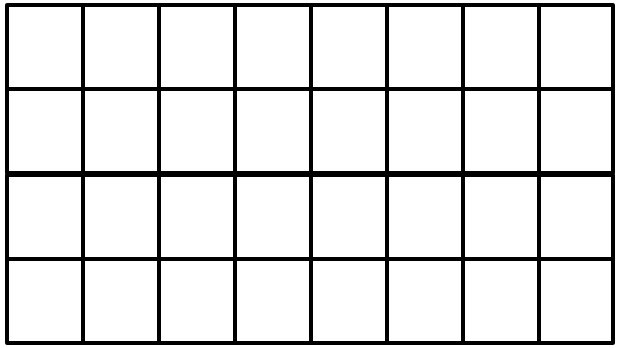
Read







Write



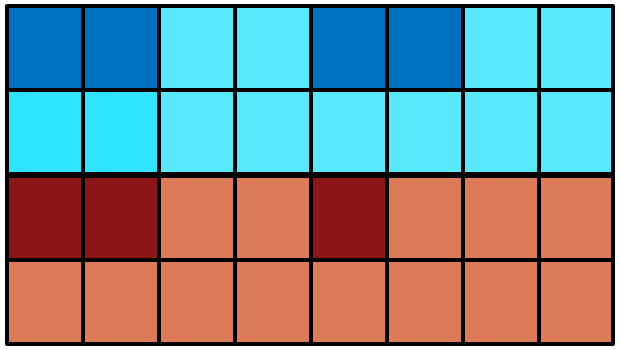
SRAM

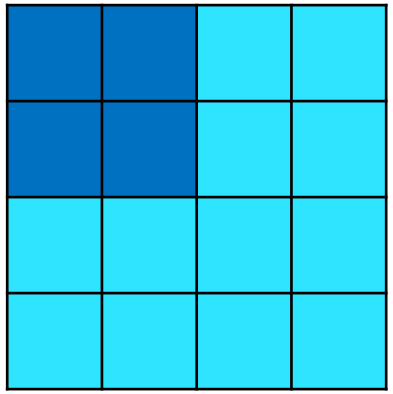
FIFO

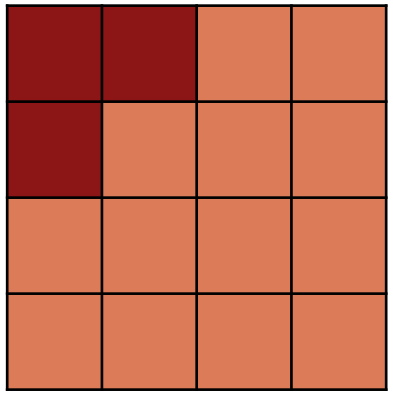
LINE BUFFER

DOUBLE BUFFER

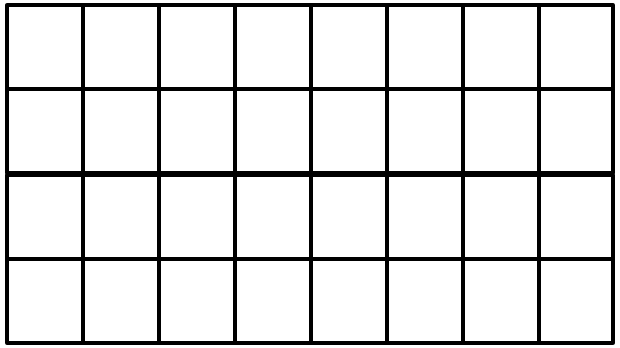
Read







Write



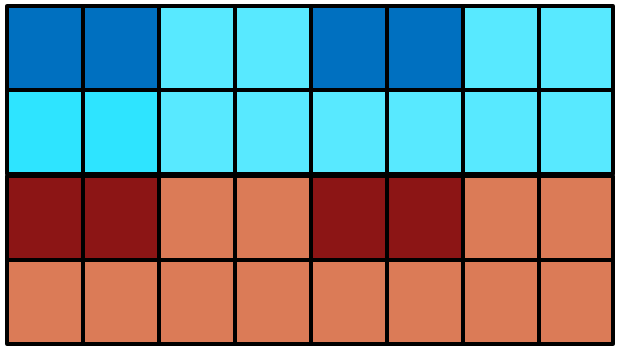
SRAM

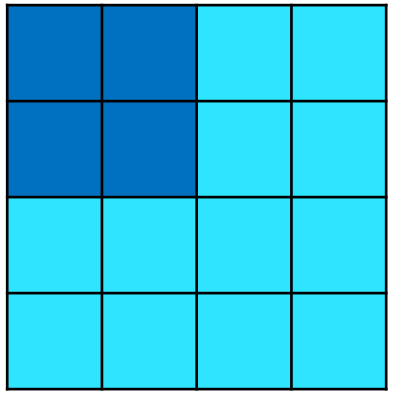
FIFO

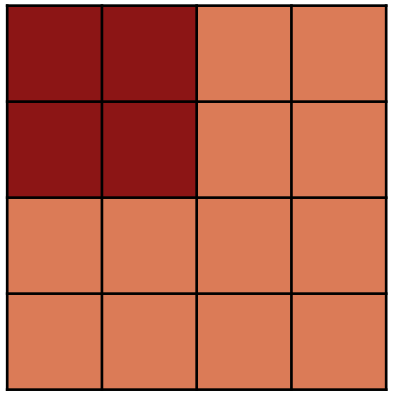
LINE BUFFER

DOUBLE BUFFER

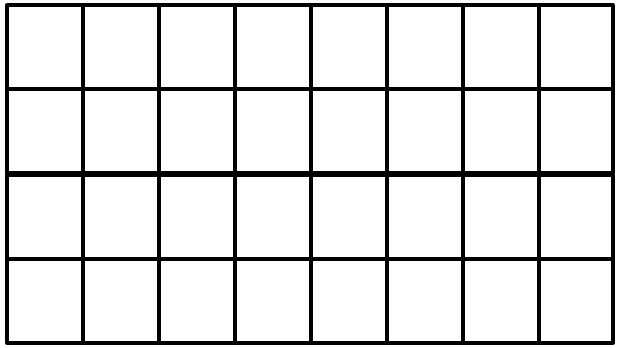
Read







Write



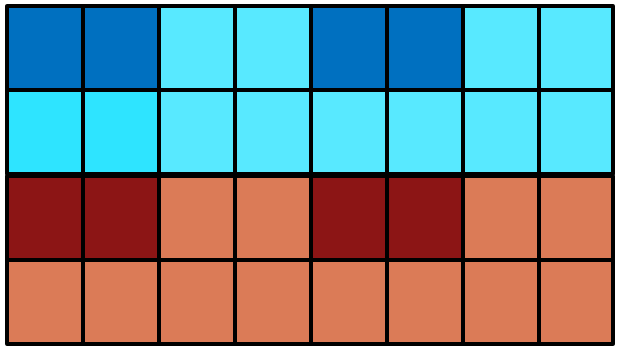
SRAM

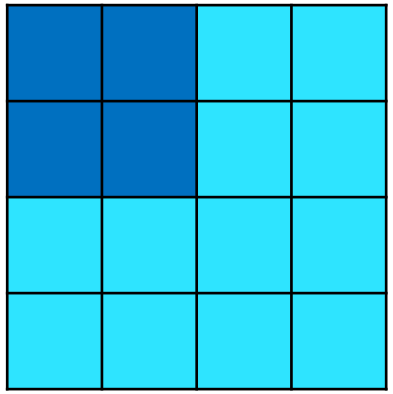
FIFO

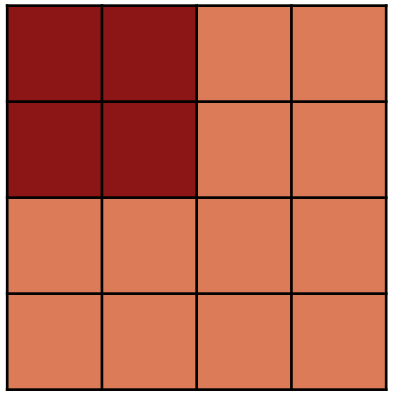
LINE BUFFER

DOUBLE BUFFER

Read







Halide Loop Nest – irregular stride access

```
for y 0:3,
 for x 0:3,
  for r.z 0:2
   for r.y 0:2,
    for r.x 0:2,
       Consume input(x+r.x, y+r.y, r.z)
```

Access Pattern Generator for unified buffer

High dimensional cube access reduced to 1D

Compact representation:

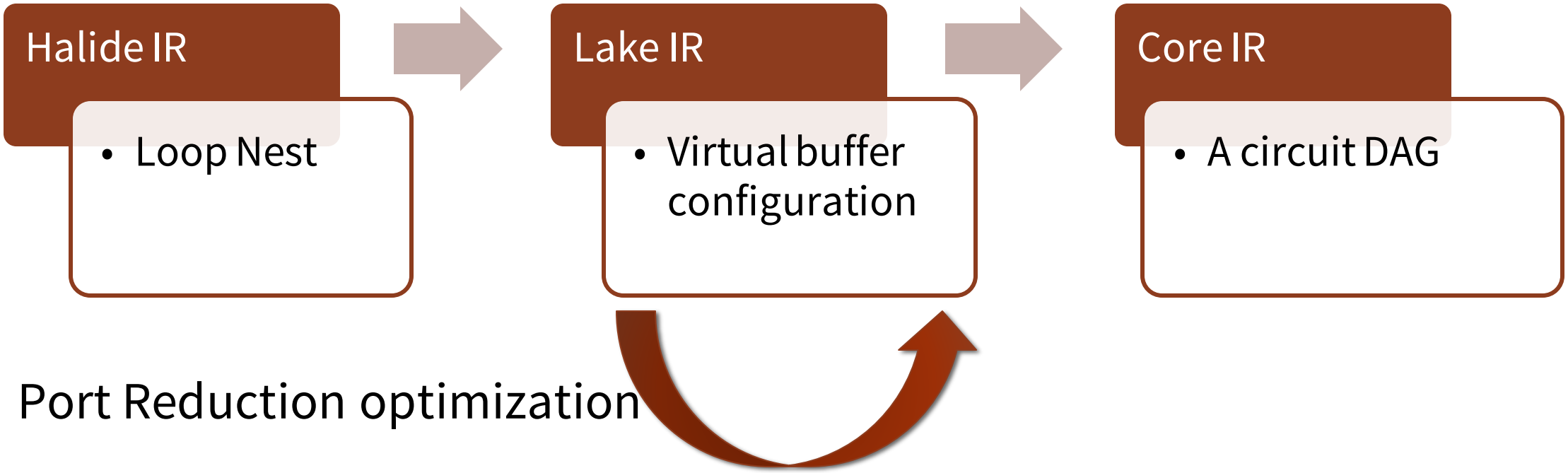
- Range list: `{rng0, rng1, rng2 ..}`
- Stride list: `{st0, st1, st2 ...}`
- Starting addr: `start_addr`

Compact expression for

```
…
for itr2 in [0: rng2]
  for itr1 in [0: rng1]
    for itr0 in [0: rng0]
      Addr = start_addr + itr0*st0 + itr1*st1 + itr2*rng2 + …
```

Lake Rewrite Rule

Lake Unified Buffer Generation Flow -a set of rewrite rules



- Port Reduction optimization
- CoreIR Generation
  - Banking
  - Chaining
  - Flatten the access iterator

  …

How Halide IR represents a Unified Buffer

Halide Loop Nest

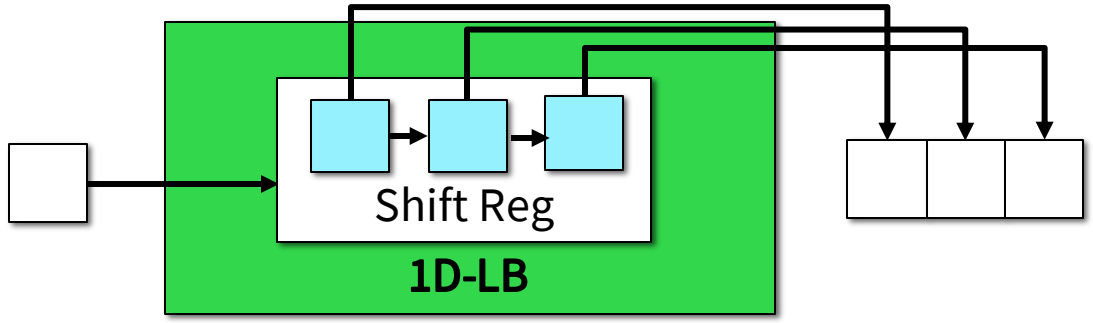
```
for y 0:6,
 for x 0:6,
   for r.y 0:3,
    for r.x 0:3,
       Consume input(x+r.x, y+r.y)
```

Virtual Unified Buffer Configuration(Lake IR)

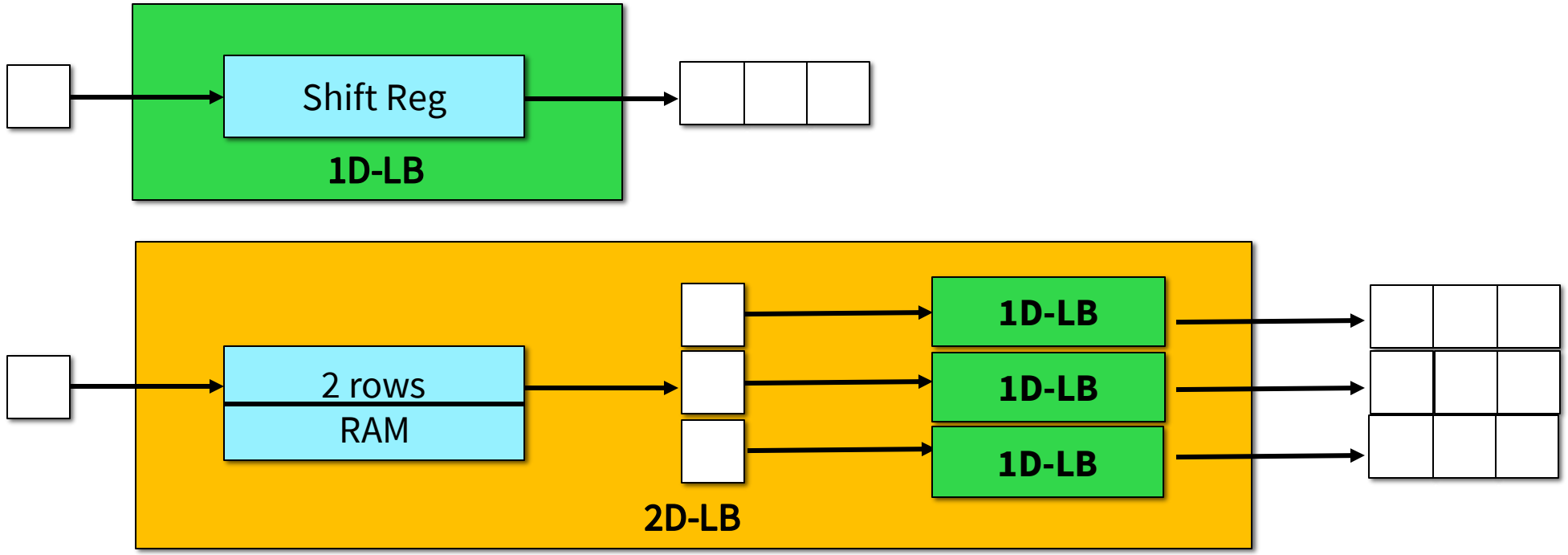
- Input port number
- Output port number
- Access Pattern
  - Range list
  - Stride list
  - Start addr list

- Connection relation

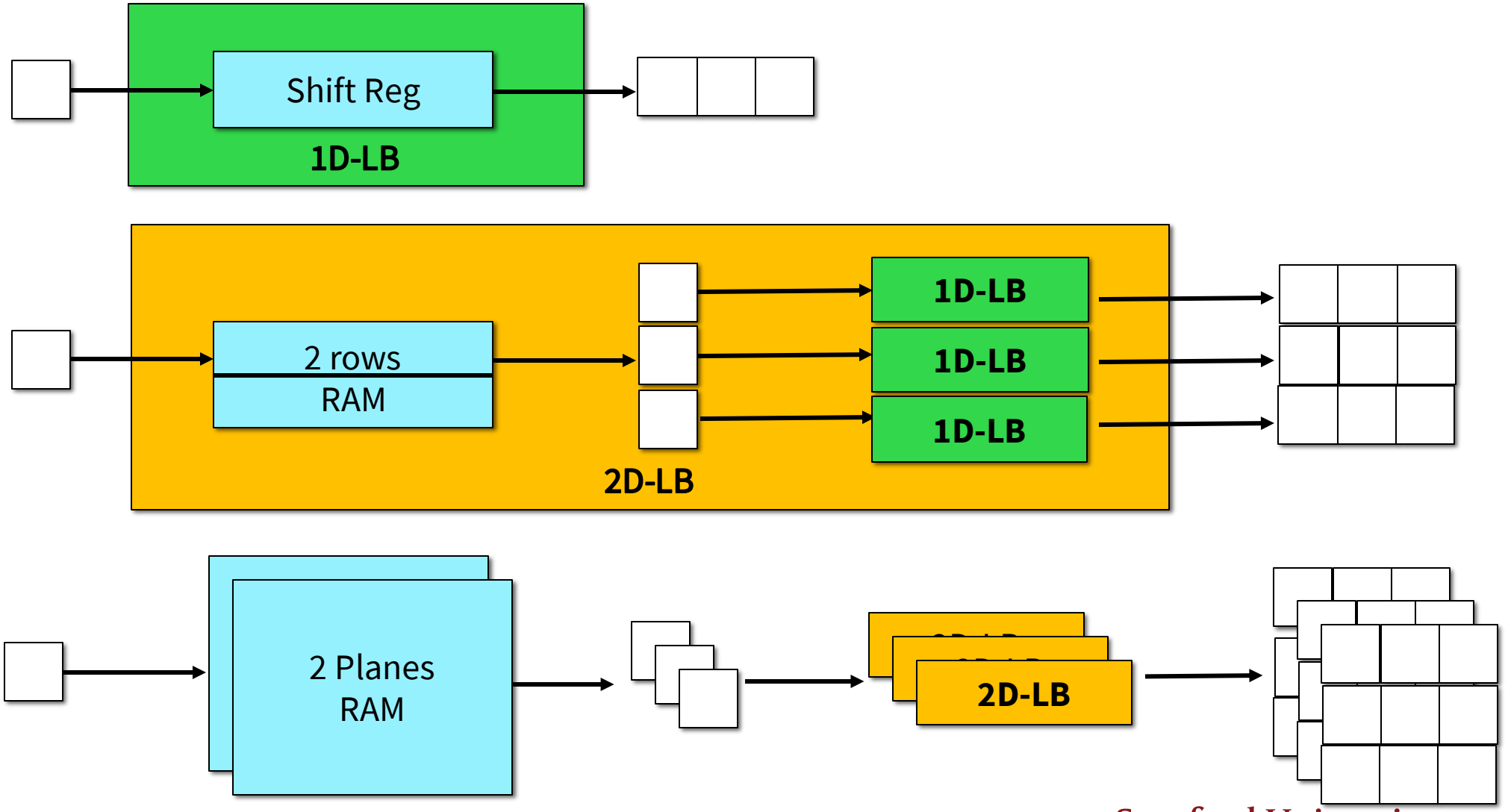
Multi-dimensional Line buffer– recursive structure



Multi-dimensional Line buffer– recursive structure



Multi-dimensional Line buffer– recursive structure



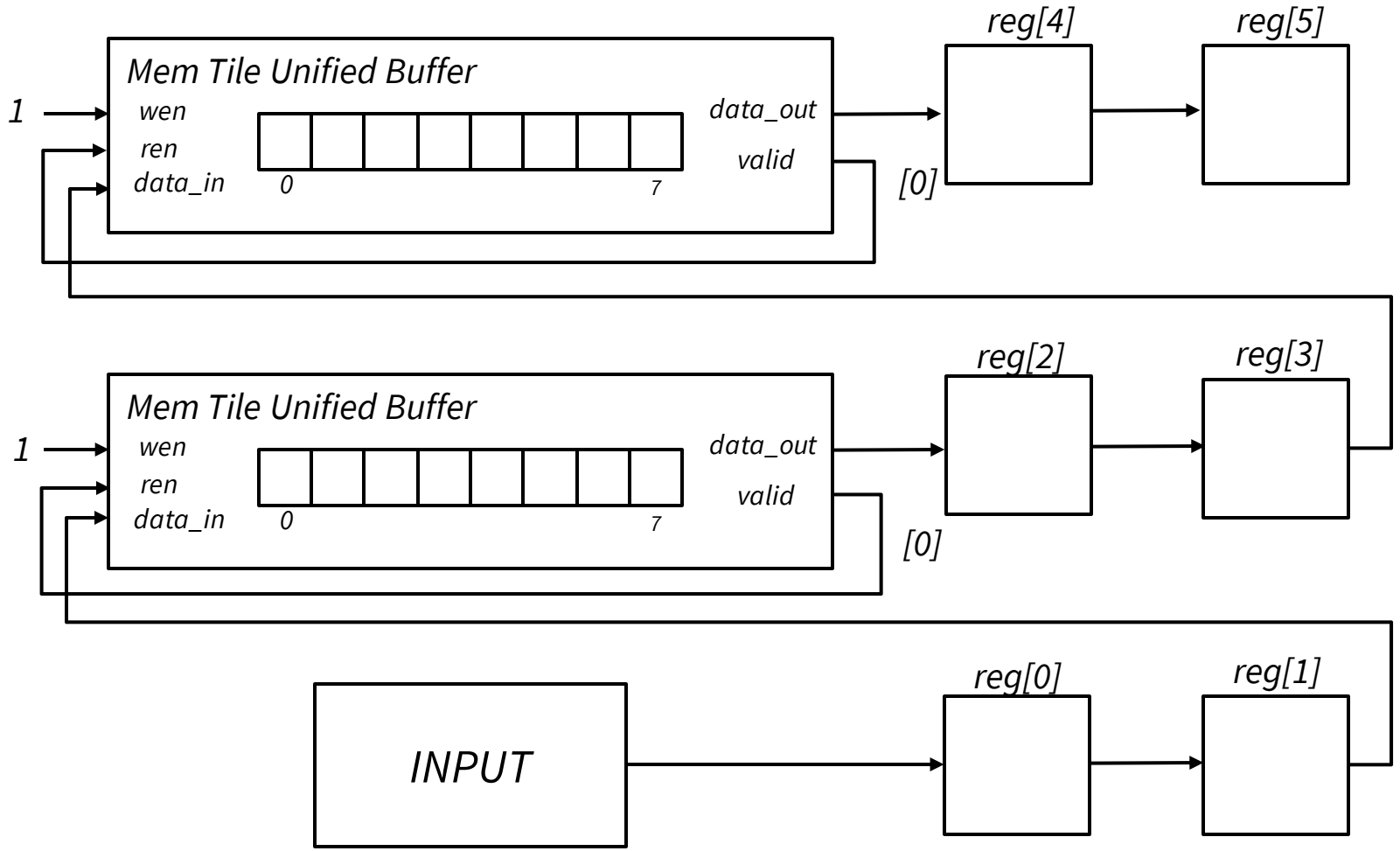
Core IR represent a Unified Buffer: DAG with configuration

Config

Access Pattern Config
Contiguous with different start_addr

Read Delay = 8 (Whole row)

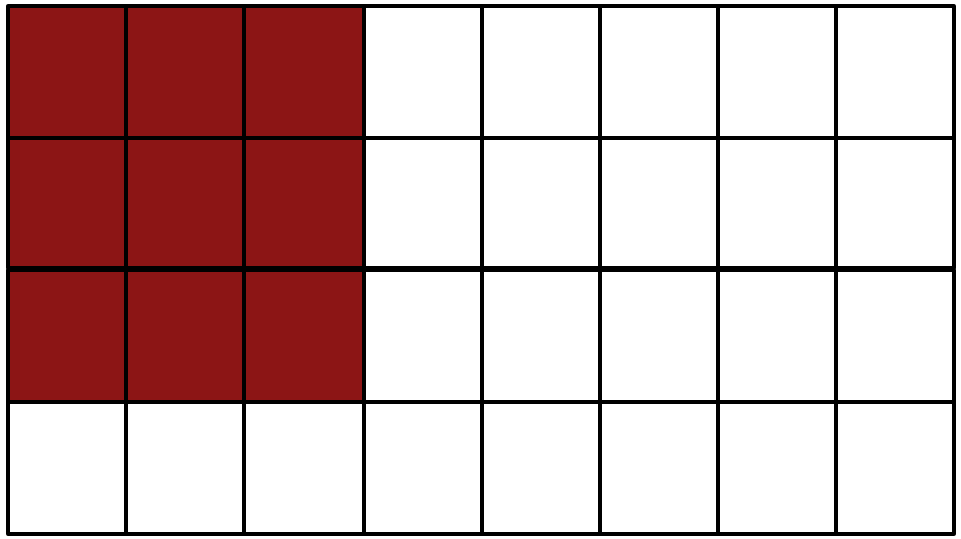
Fifo Depth = 2



Rewrite Rule 1: Port Optimization

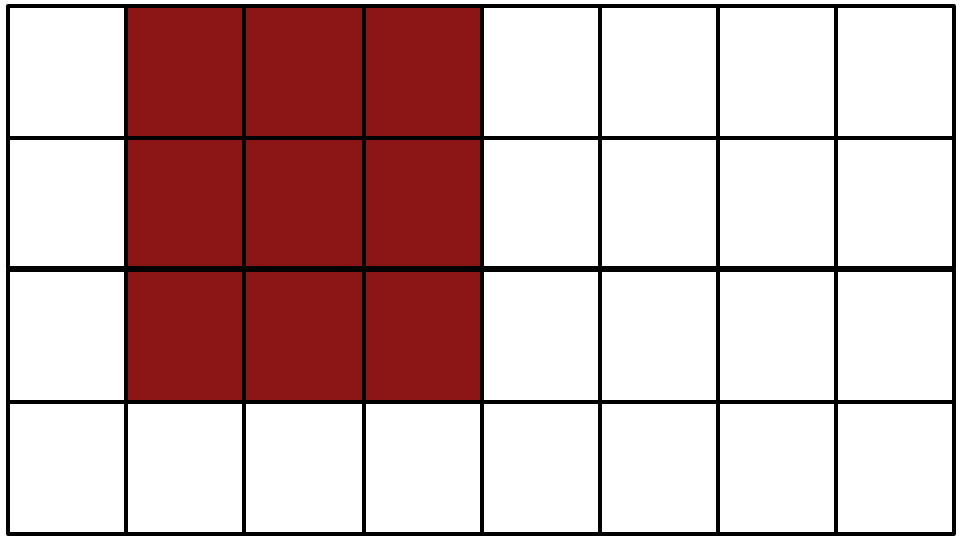
Example 3x3 conv

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}



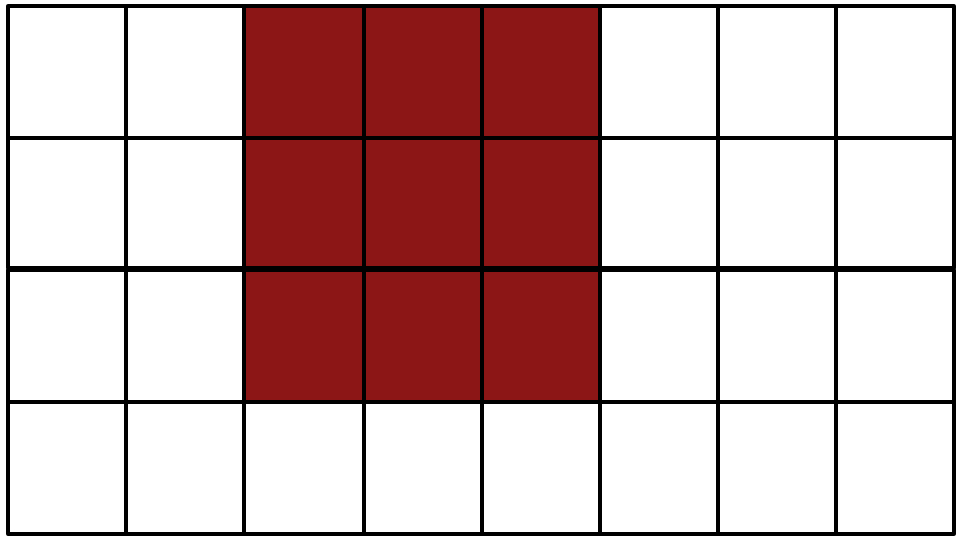
Example 3x3 conv

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}



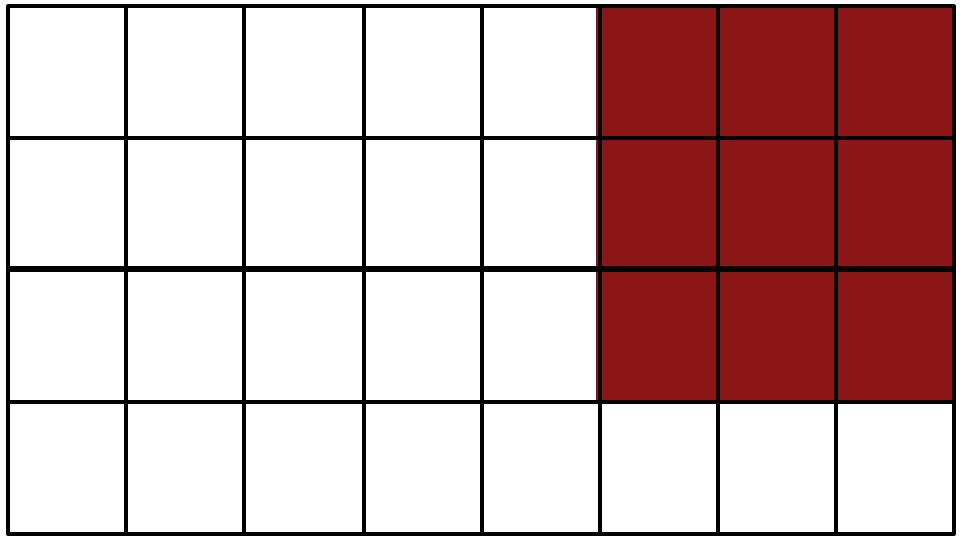
Example 3x3 conv

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}



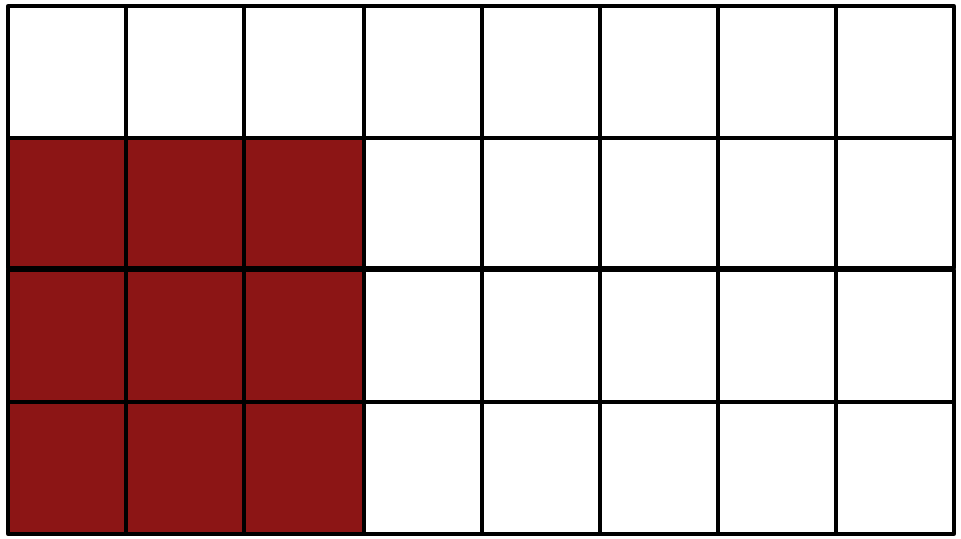
Example 3x3 conv

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}



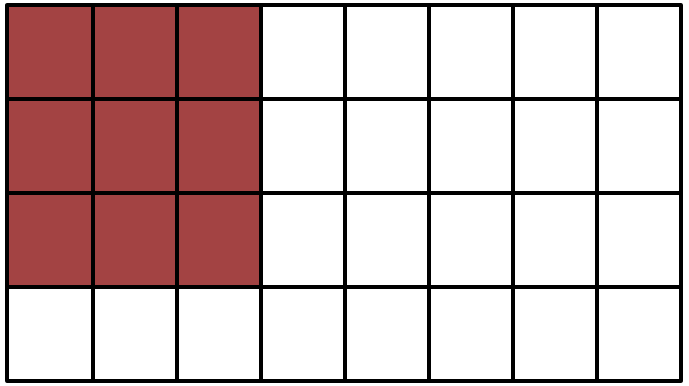
Example 3x3 conv

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}

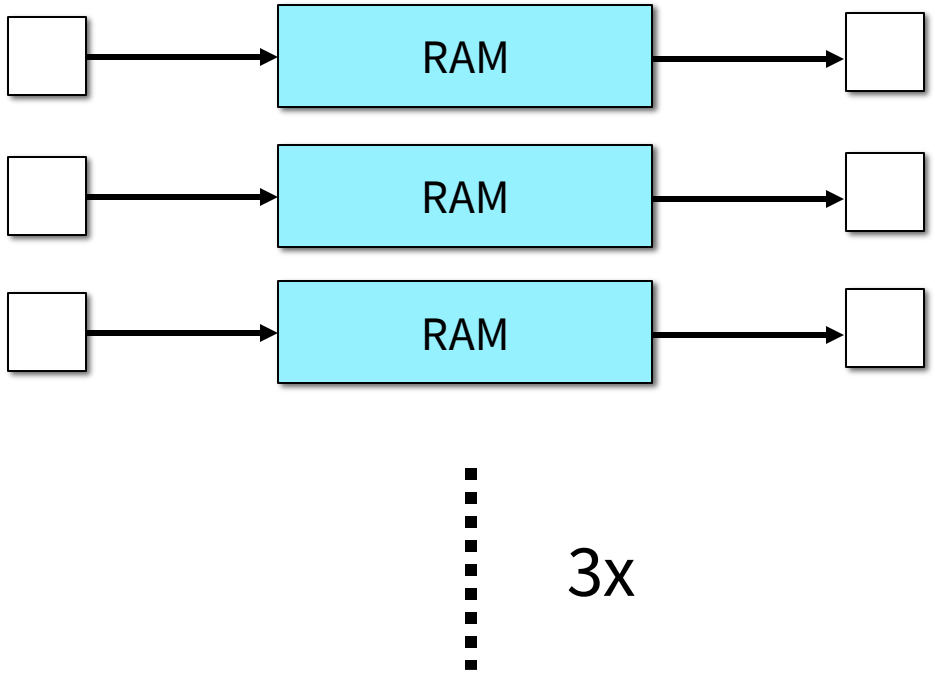


Port Optimization: Recursively Finding Overlap

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}



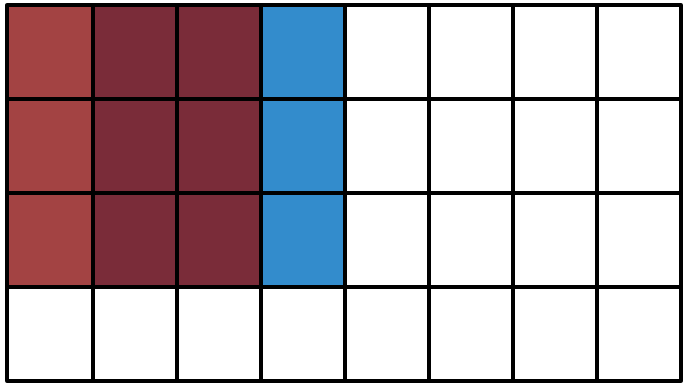
- Generated Hardware
  - 9 ports RAM



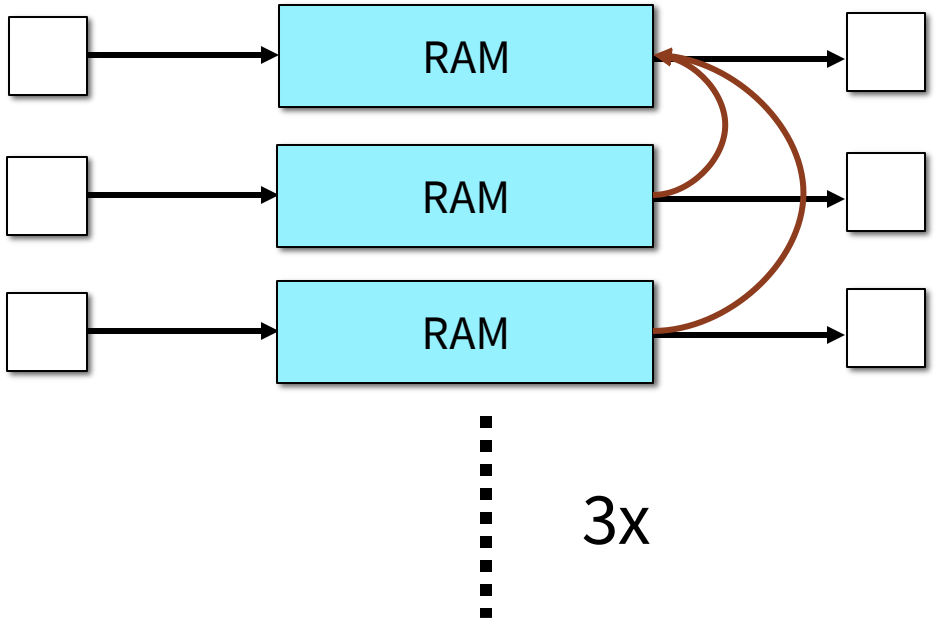
Finding Overlap: Increment stride in dim 1

- Start address: {0, 1, 2, 8, 9, 10, 16, 17, 18}
- Stride: {1, 8}

- Next access: {1, 2, 3, 9, 10, 11, 17, 18, 19}
- Overlapping

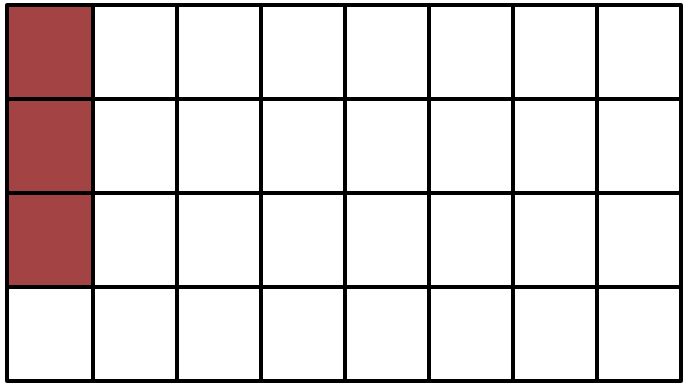


- Merge 9 Port into 3
- Create 3 shift reg
  - Depth = 2
  - reg Size = 1

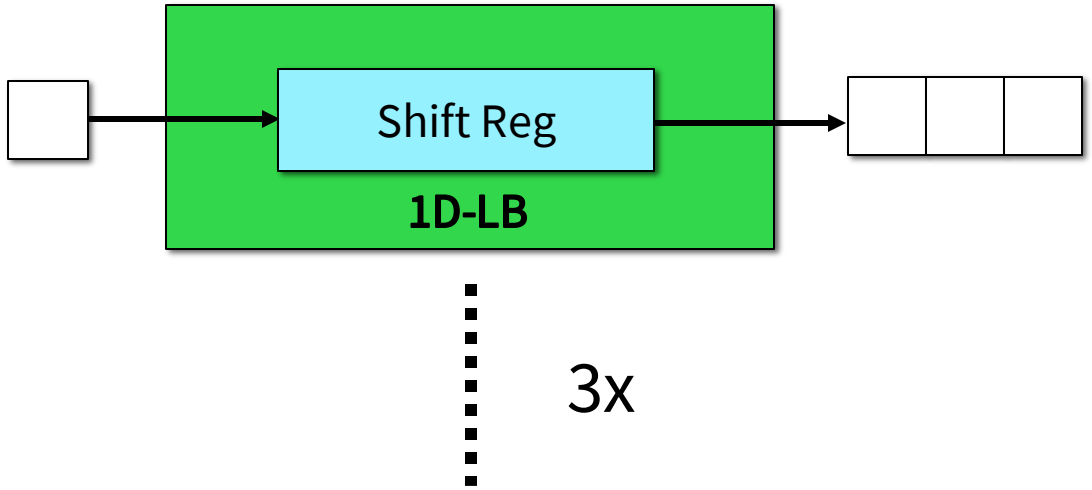


Merge Port: update and create ShiftReg

- Start address: {0, 8, 16}
- Stride: {1, 8}

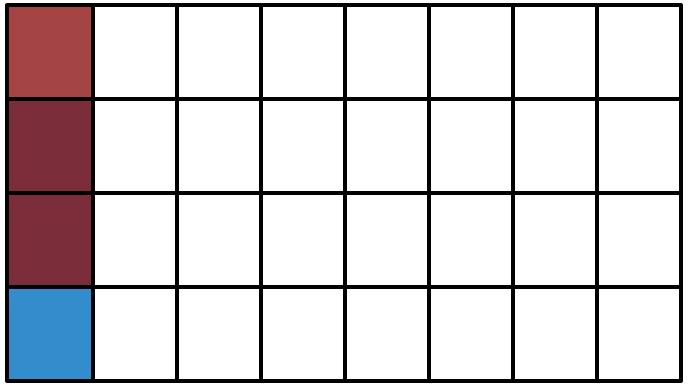


- Generated Hardware
  - 3 ports
  - 3 1D-LB with
    - Depth = 2
    - reg Size = 1

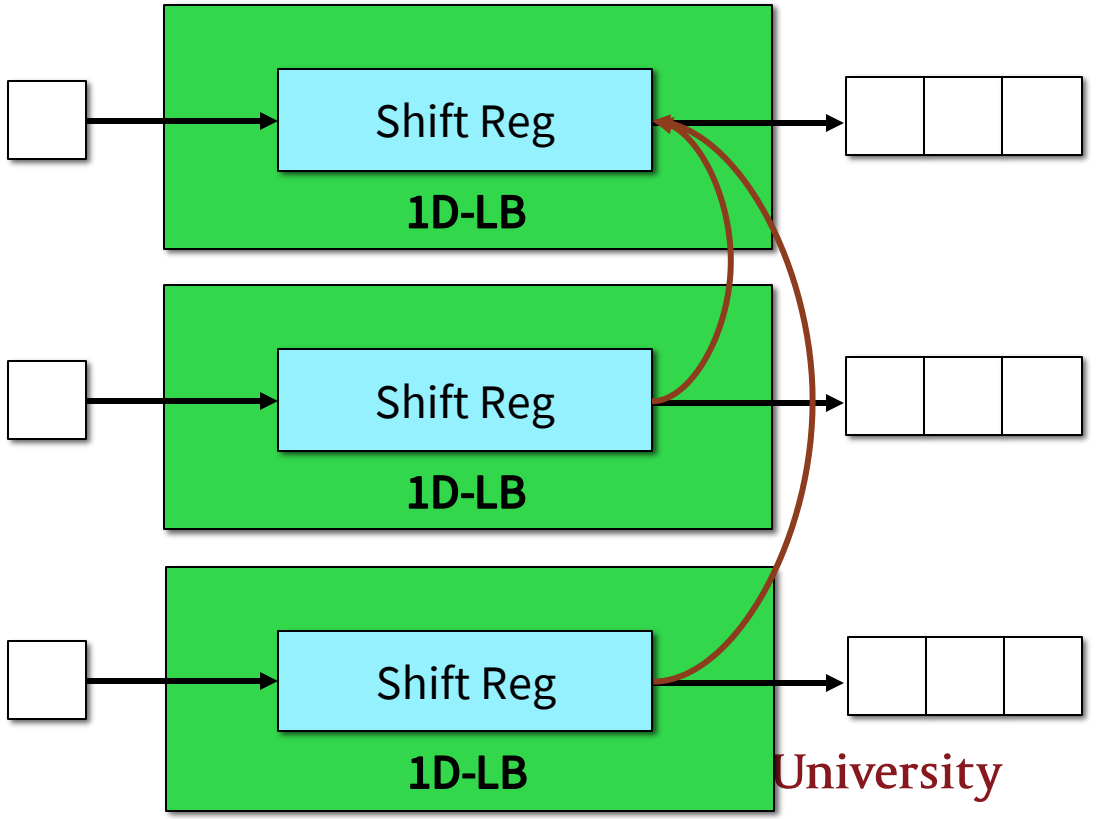


Finding Overlap: Move to the next dimension

- Start address: {0, 8, 16}
- Stride: {1, 8}
- Next address: {8, 16, 24}



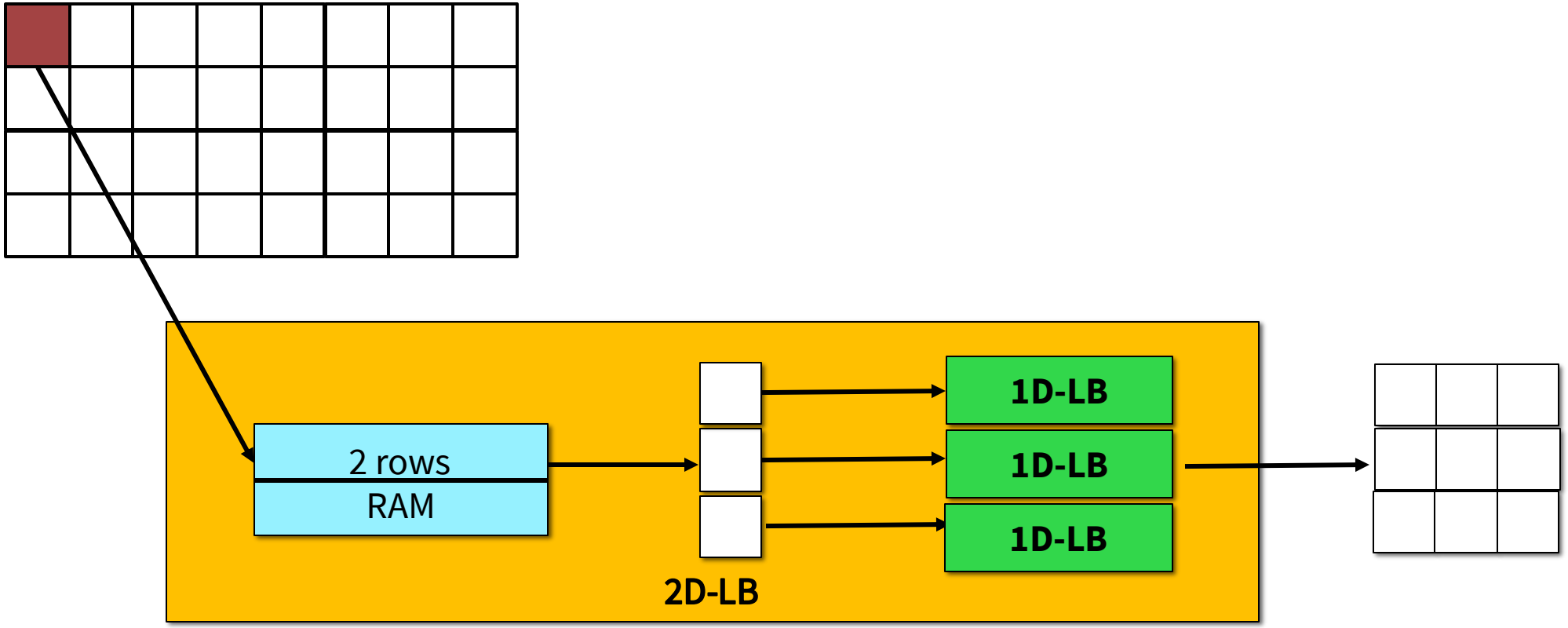
- Merge 3 Ports into 1
- Create Meta shift reg(Row buffer) with
  - Depth = 2
  - reg Size = 8



Merge port: Create a Meta-ShiftReg(Row buffer)

- Start address: {0}
- Stride: {1, 8}

- Merge 3 Ports into 1
- Create 2 Meta shift reg(Row buffer) with
  - Depth = 2
  - reg Size = 8



Rewrite Rule 2: CoreIR Generation

CoreIR Generation

- Map **abstract memory functional model** to **technology-aware hardware**
- Wired up all hardware components

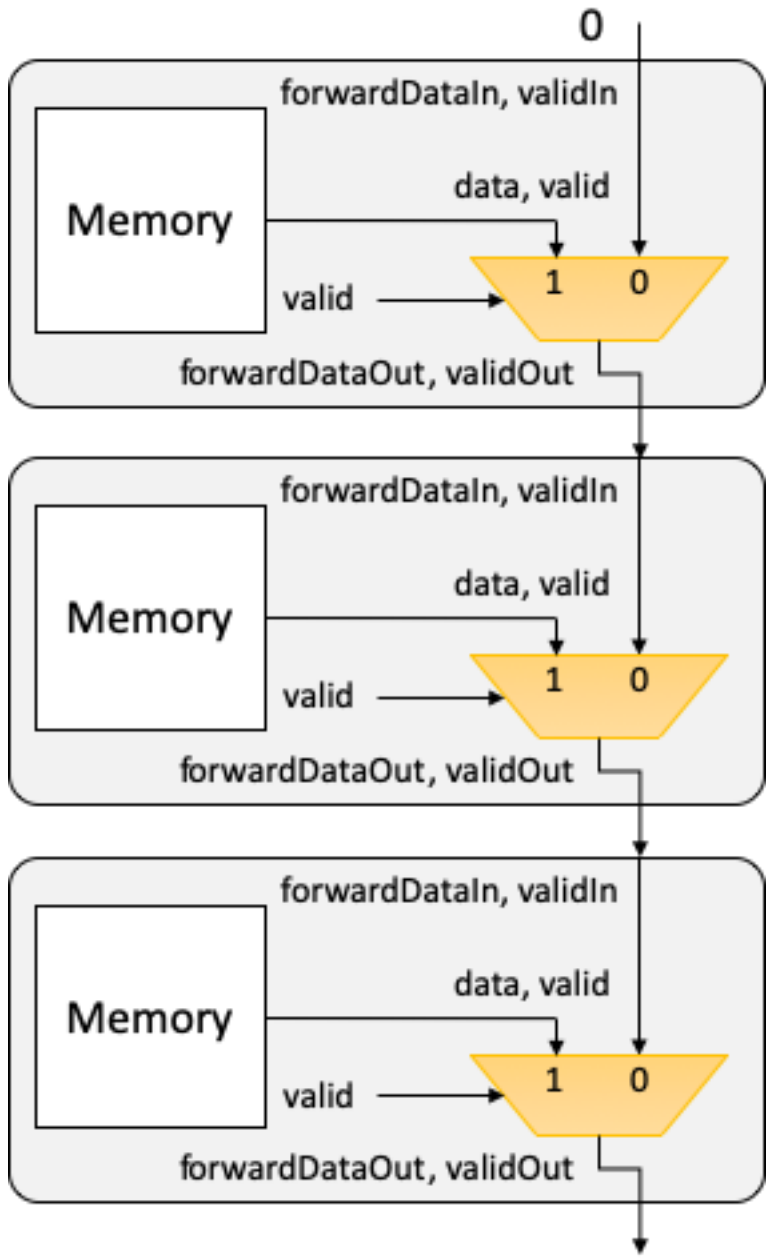
- Different configuration in memory functional model:
  - Double buffer
  - Meta Shift Reg in line buffer
- Different backend:
  (With different capacity and port number)
  - CGRA Memory tile
  - FPGA BRAM

Chaining

Virtual buffer capacity > mem tile capacity

Map to multiple mem tiles

- Share: global access pattern and iterator
- Unique: range of data
- Valid signal & datapath

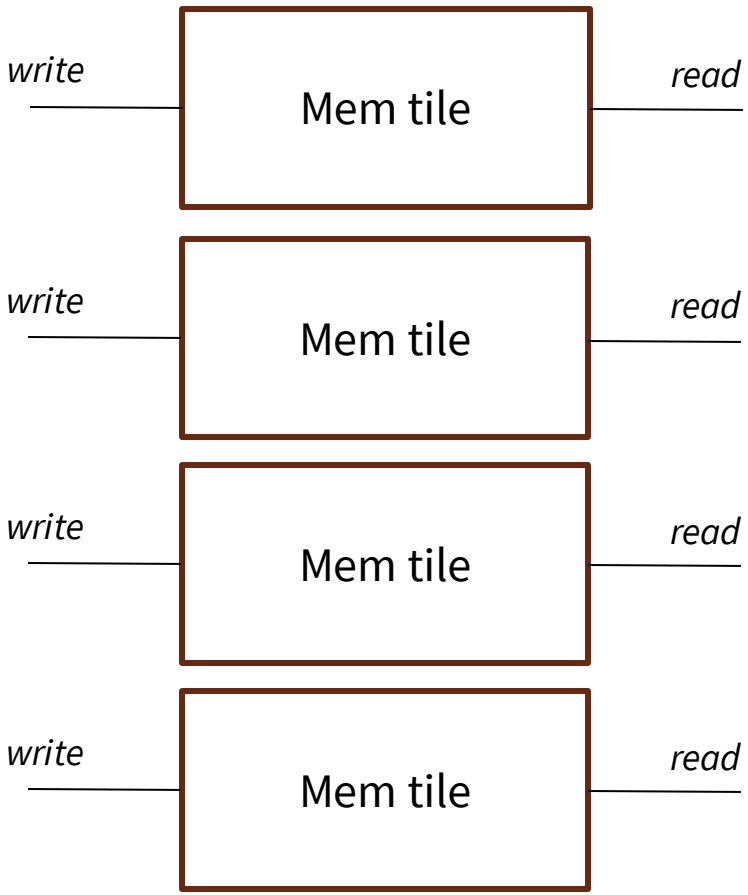


Banking

Virtual Buffer Port # > Mem tile Port #

Map to multiple mem tiles

- Parallel
- All access pattern should be same



Conclusion & Future Work

- Create and understand the rewrite rule
  - Port optimization
  - CoreIR Generation (mapping)
- Apply those on Halide output and map to our current generation memory tile

- In the future(Being agile)
  - More rewrite rules can be added for specilized buffer
  - Generate the memory tile hardware for future version

Thanks

Q & A