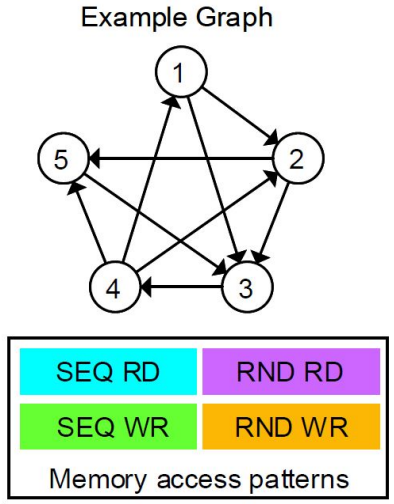
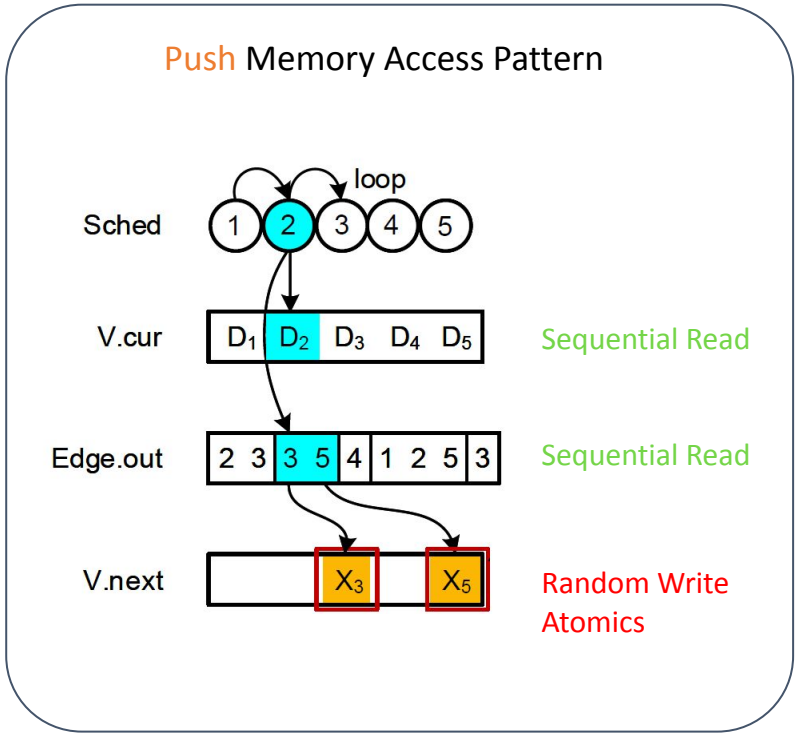
EECS 627 Final Presentation

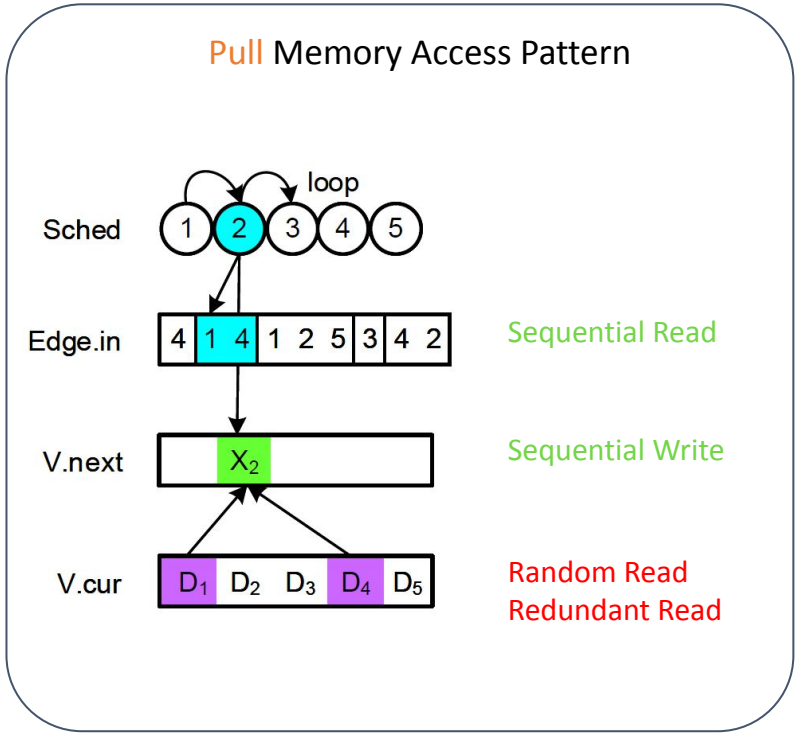
Group 6 - An Event-Driven Graph Processing Accelerator

Motivation

- Vertex-centric Graph Processing Models either Push/Pull updates to/from neighbours
  - Both involves expensive random memory access

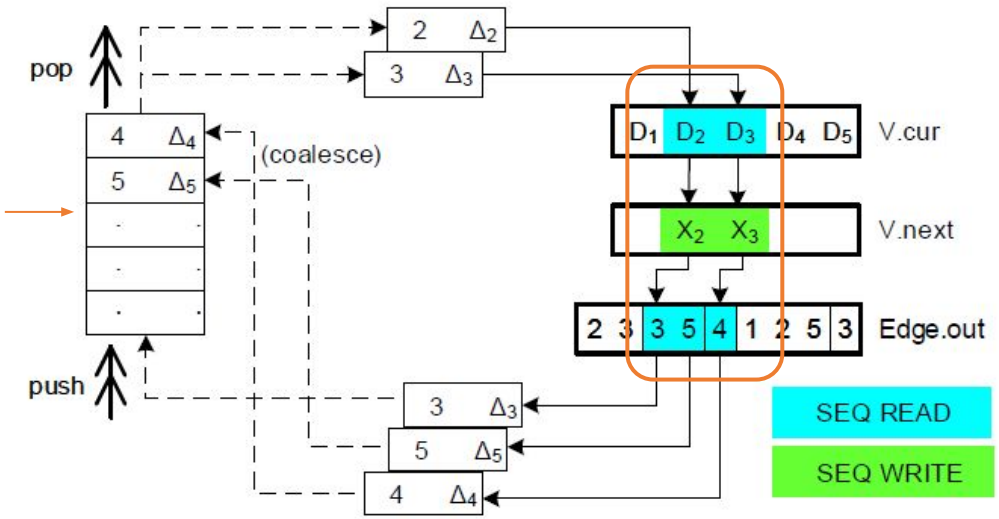






S. Rahman, N. Abu-Ghazaleh and R. Gupta, "GraphPulse: An Event-Driven Hardware Accelerator for Asynchronous Graph Processing," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 2020, pp. 908-921, doi: 10.1109/MICRO50266.2020.00078.

Random Access Reduction



Algorithmic insight:

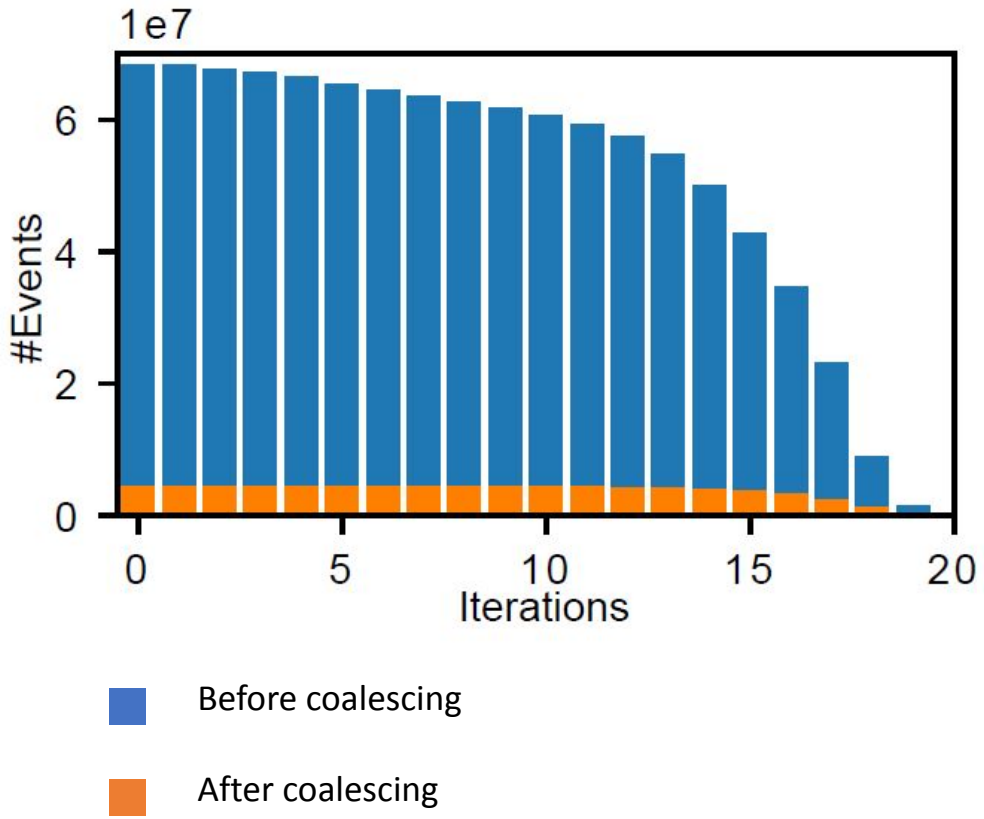
- Reduce function is commutative and associative for many algorithms
- Events targeting the same vertex can be processed in any order

Solution

- Sort the events with target vertex in the event queue
- Issue events with consecutive target vertices at the same time
- Convert random access to sequential access

S. Rahman, N. Abu-Ghazaleh and R. Gupta, "GraphPulse: An Event-Driven Hardware Accelerator for Asynchronous Graph Processing," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 2020, pp. 908-921, doi: 10.1109/MICRO50266.2020.00078.

Event Coalescing



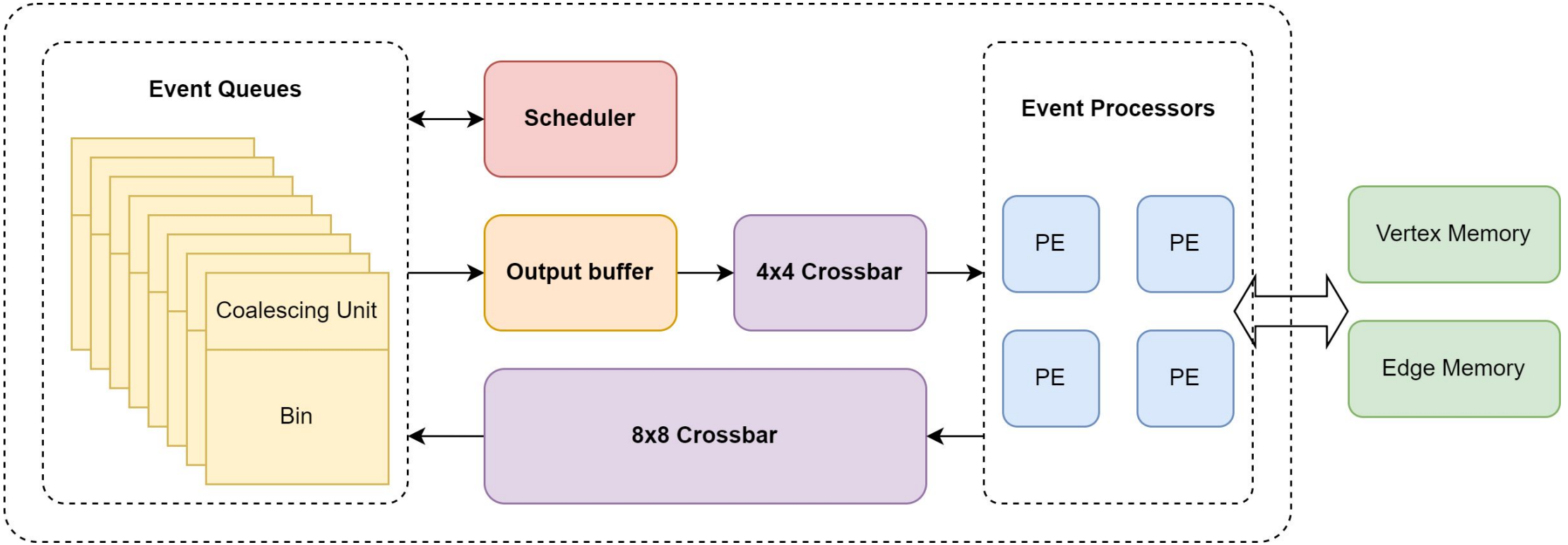
Problems
- A huge number of events are generated in each iteration
- Massive storage of events
- Atomic operations

Algorithmic insight
- Propagation function is distributive for many algorithms

Solution: Coalesces events targeting the same vertex using Reduce function in the event queue
- Less storage of events
- Less workload to be processed
- Only one event targeting each vertex at a time
- Faster convergence

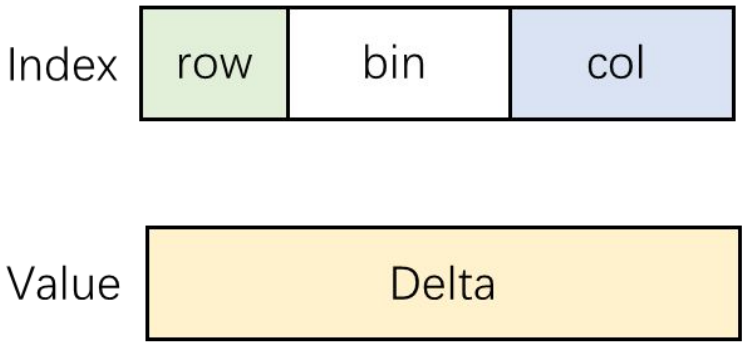
S. Rahman, N. Abu-Ghazaleh and R. Gupta, "GraphPulse: An Event-Driven Hardware Accelerator for Asynchronous Graph Processing," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 2020, pp. 908-921, doi: 10.1109/MICRO50266.2020.00078.

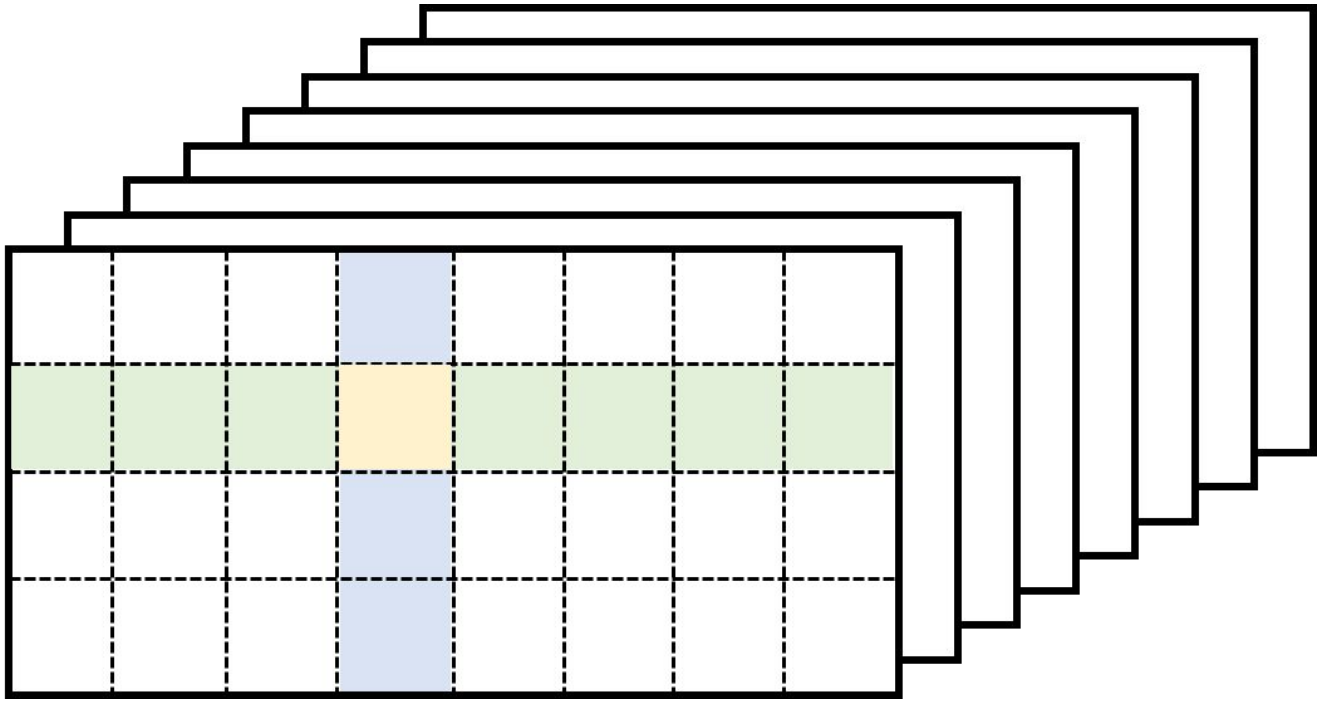
Project Overview



Event Queues

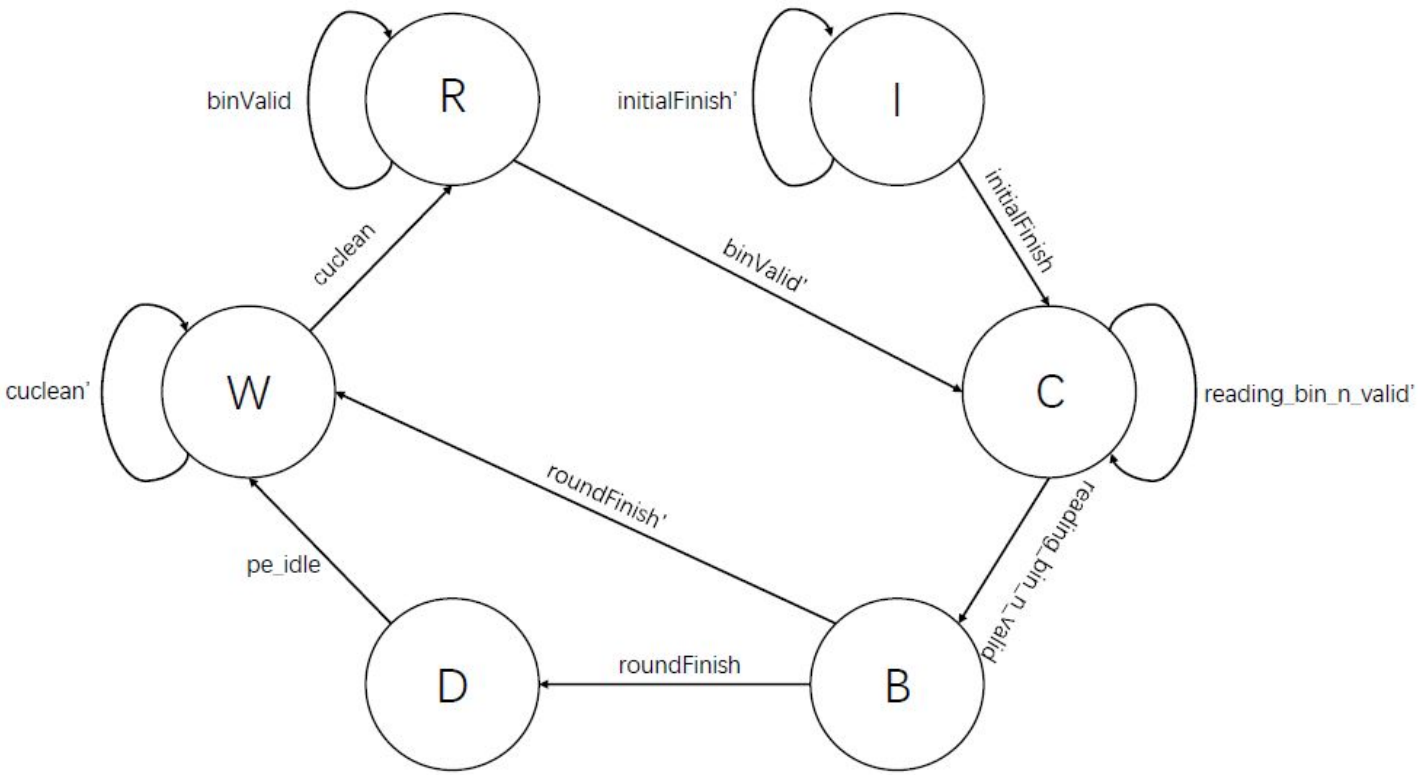
- Direct mapped event storage
  - 8 bin
  - 4 row
  - 8 column





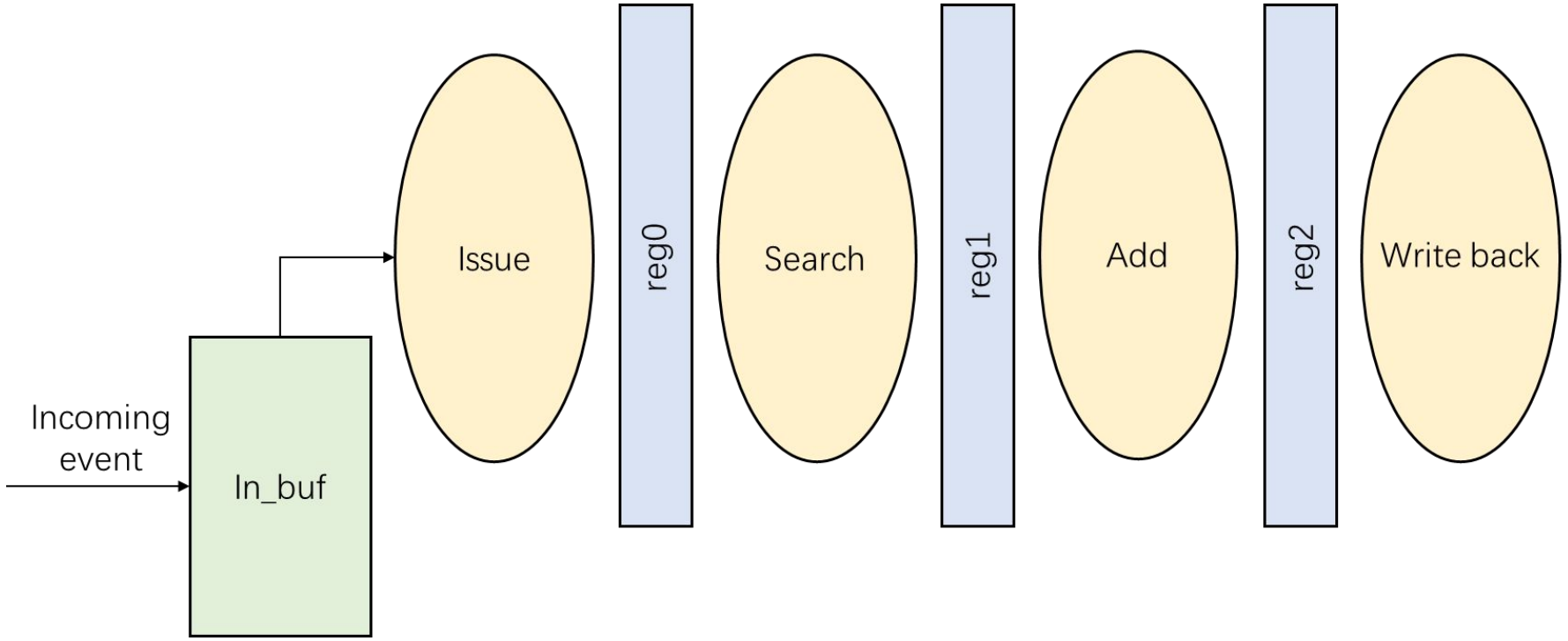
Scheduler FSM Modified for Atomicity

- 6-state FSM
  - **I**: initialization
  - **C**: start to select the next reading bin
  - **B**: stop taking new events from the input buffer into CU registers of the selected reading bin
  - **D**: wait until PE all idle when one bin is selected again
  - **W**: wait for CU to be clean
  - **R**: read enable

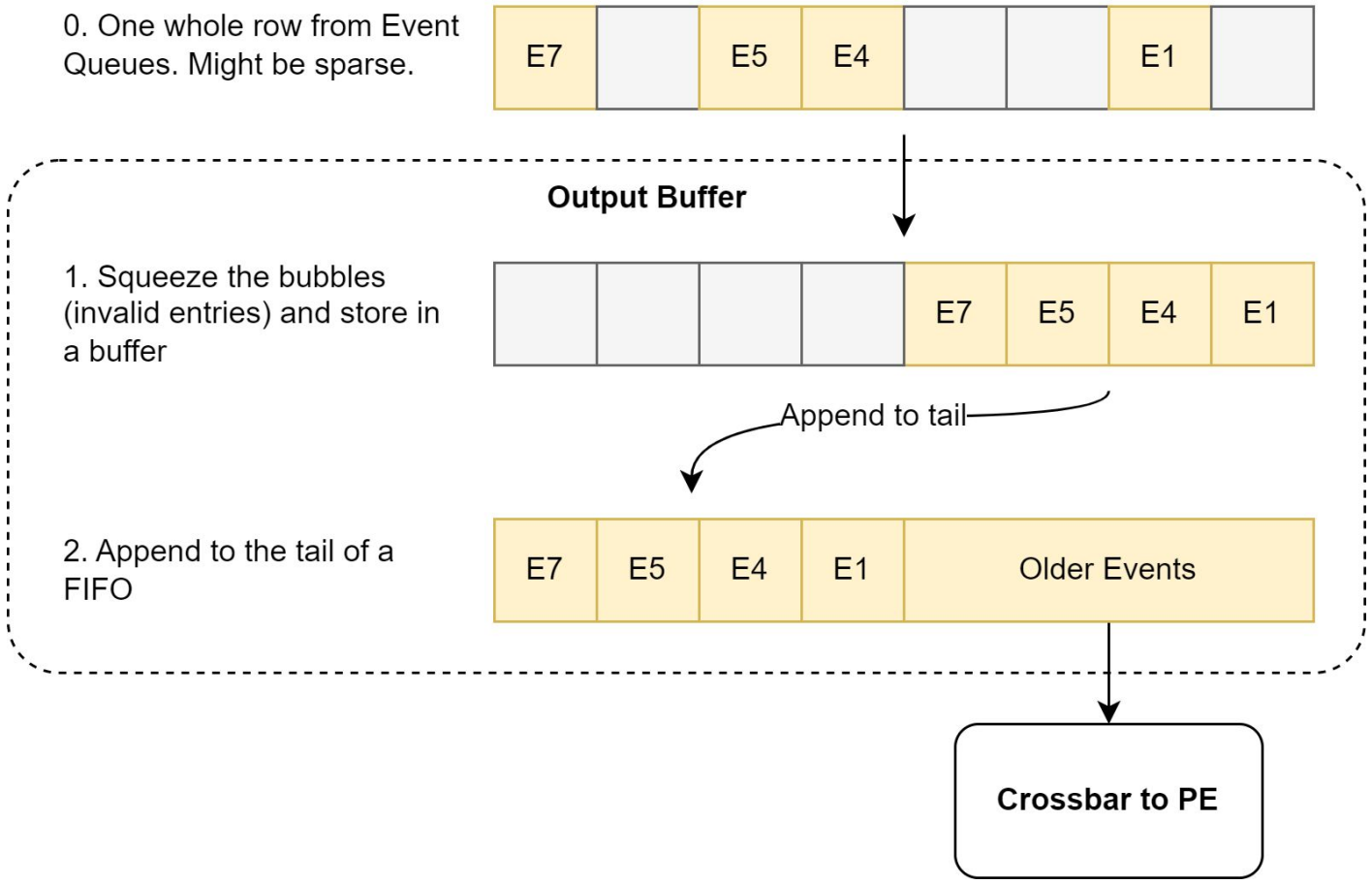


Coalescing Unit

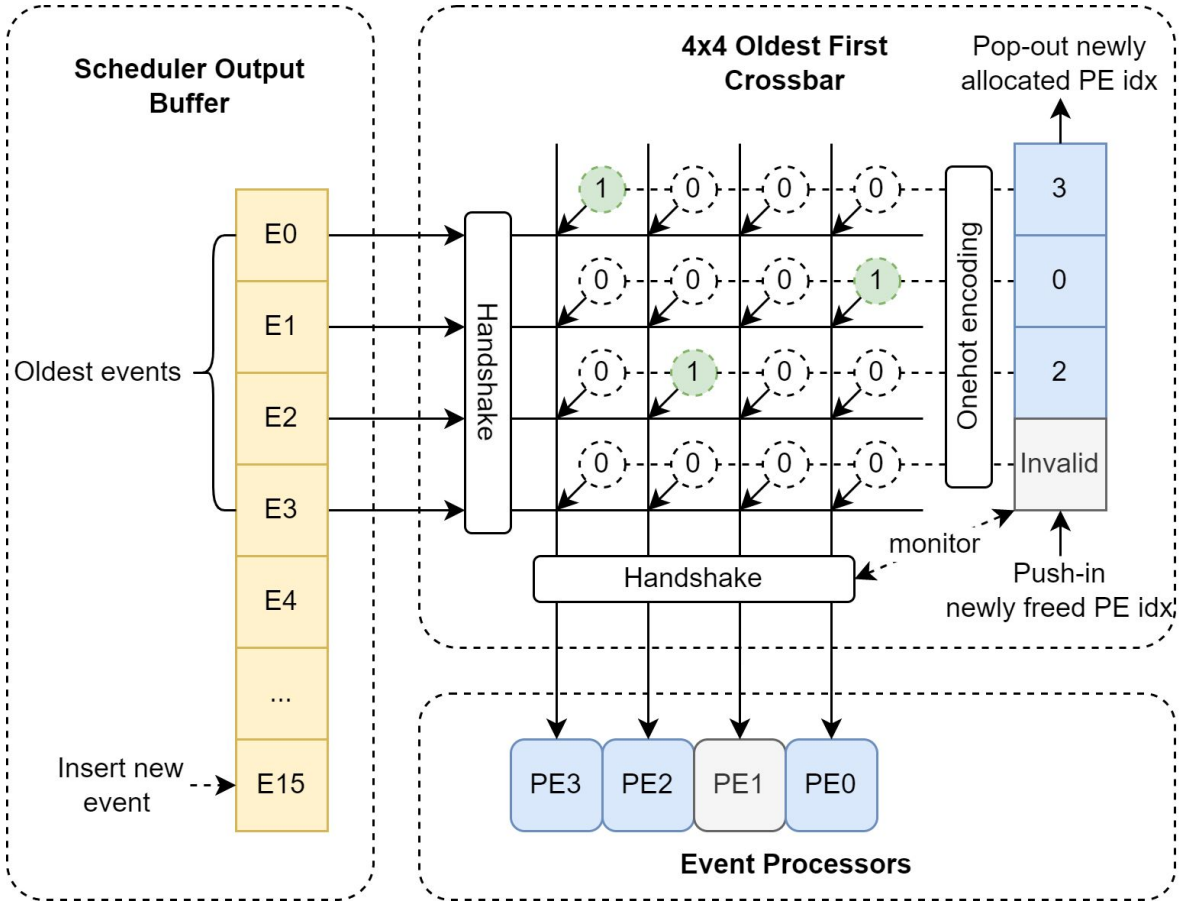
- Pipelined coalescing with input buffer
- Combine multiple events to the same destination



Scheduler Output Buffer

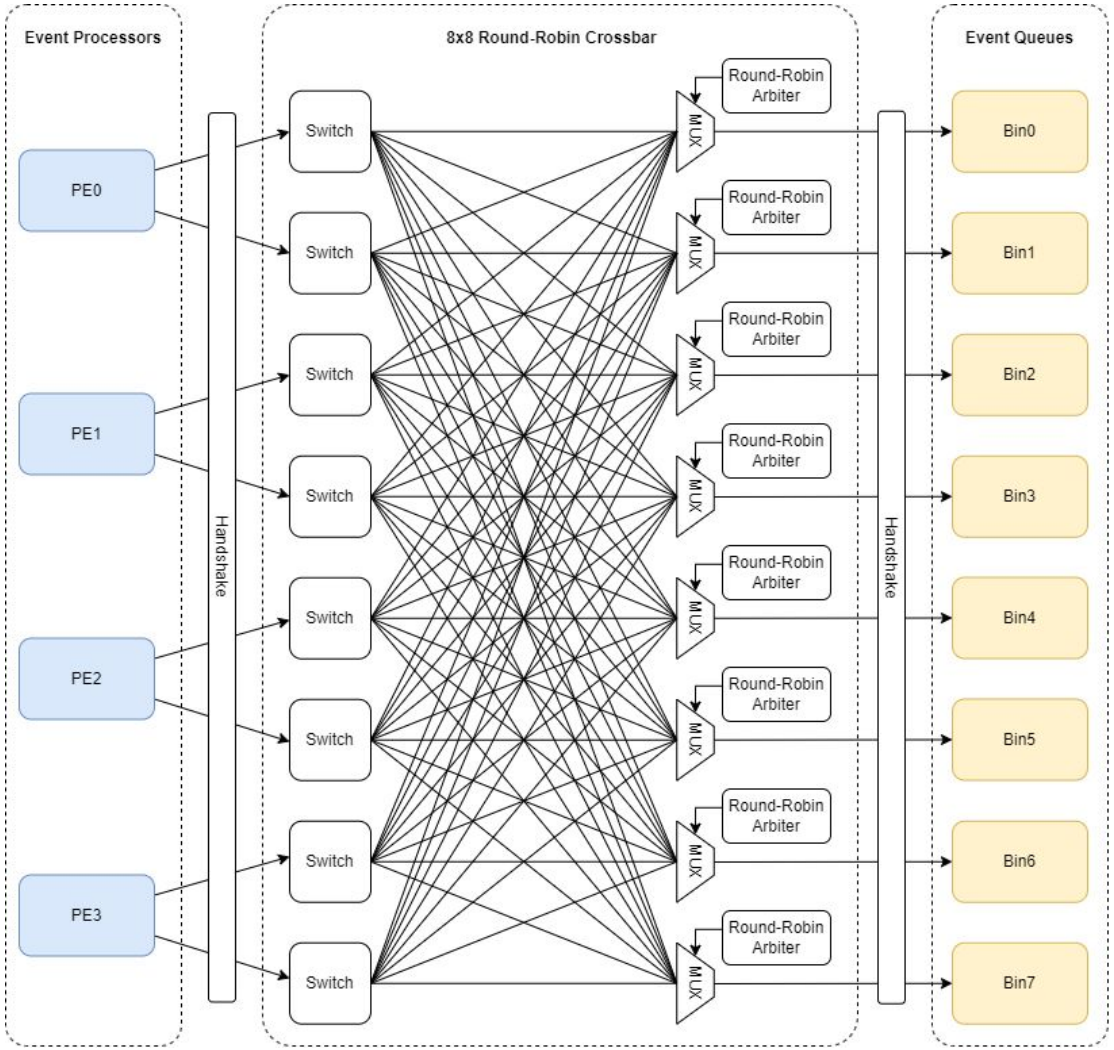


Crossbar - from Scheduler to PEs



- Oldest events are guaranteed to be allocated first.
- Earliest freed PEs are guaranteed to be allocated first.

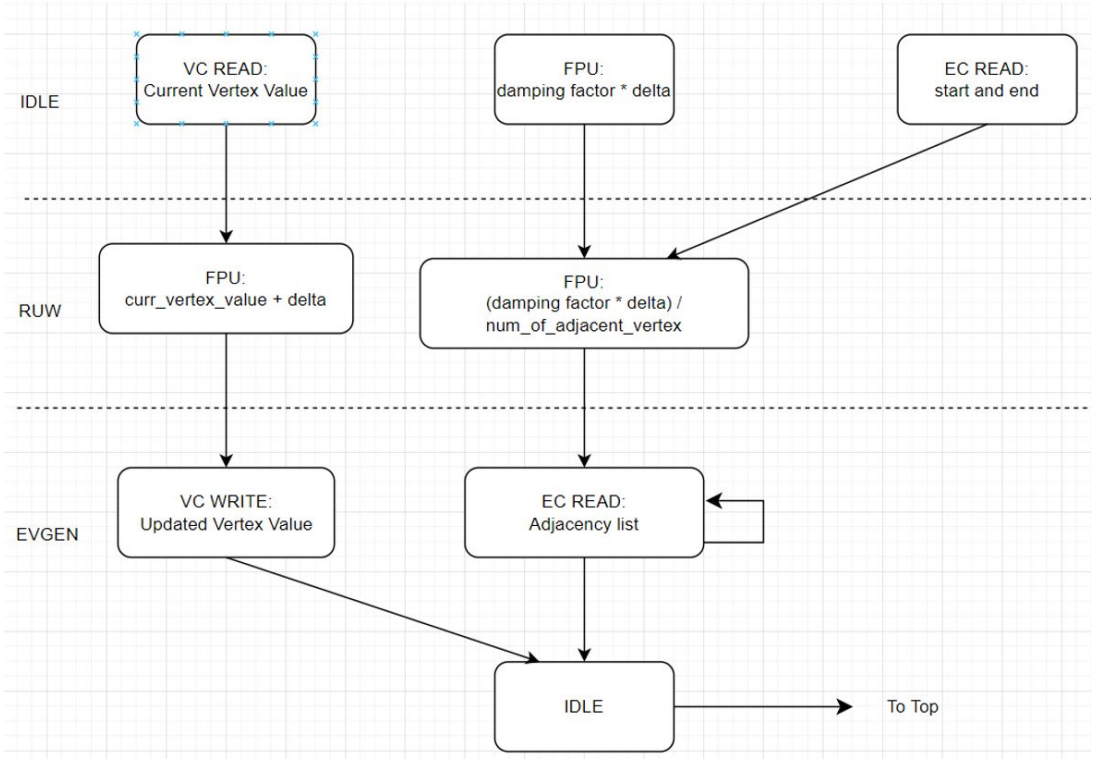
Crossbar - from PEs to Queues



- Switches direct the generated events to a bin according to vertex ID.
- Round-Robin arbiters guarantee fairness among PEs.

Event Processors

- Process existing events and generating new events (propagate)
- 4-state FSM
  - INIT: generating initial event sequence
  - IDLE: waiting for valid incoming event
  - RUW: calculating current event result
  - EVGEN: generating propagation events
- FPU
  - 4-stage pipelined as mentioned in paper
  - 16-bit half-precision (IEEE 754)
- 4 processors in our system



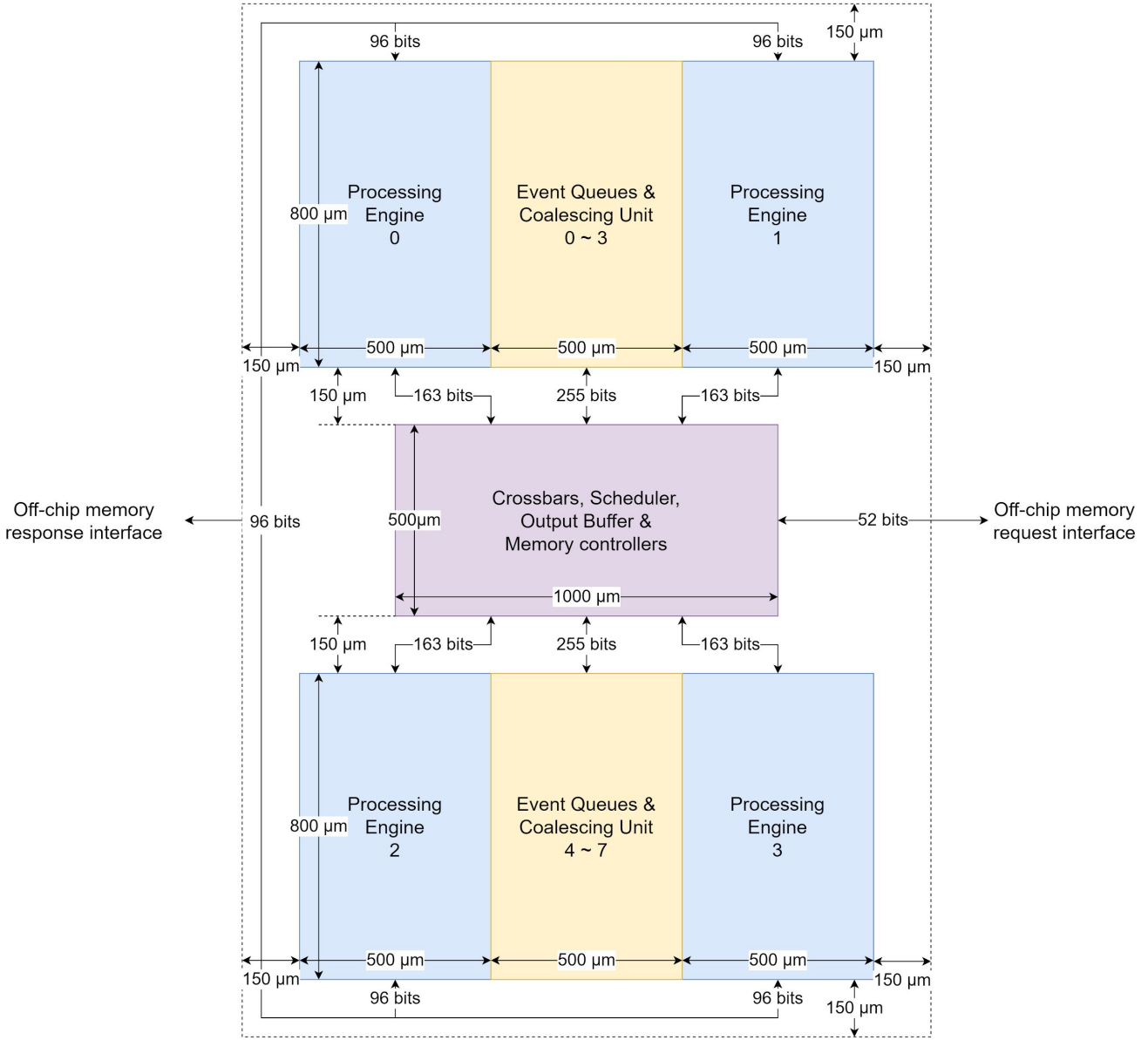
Memory

- No on-chip cache - off-chip memory models from EECS470
  - 64 bits per address
  - Configurable size
  - 5-cycle latency

- Two memories
  - One for vertex, one for edge/adjacency lists
  - Vertex: one 16-bit value per address
  - Edge:
    - CSR
    - up to four 16-bit row indices or eight 8-bit column indices per address

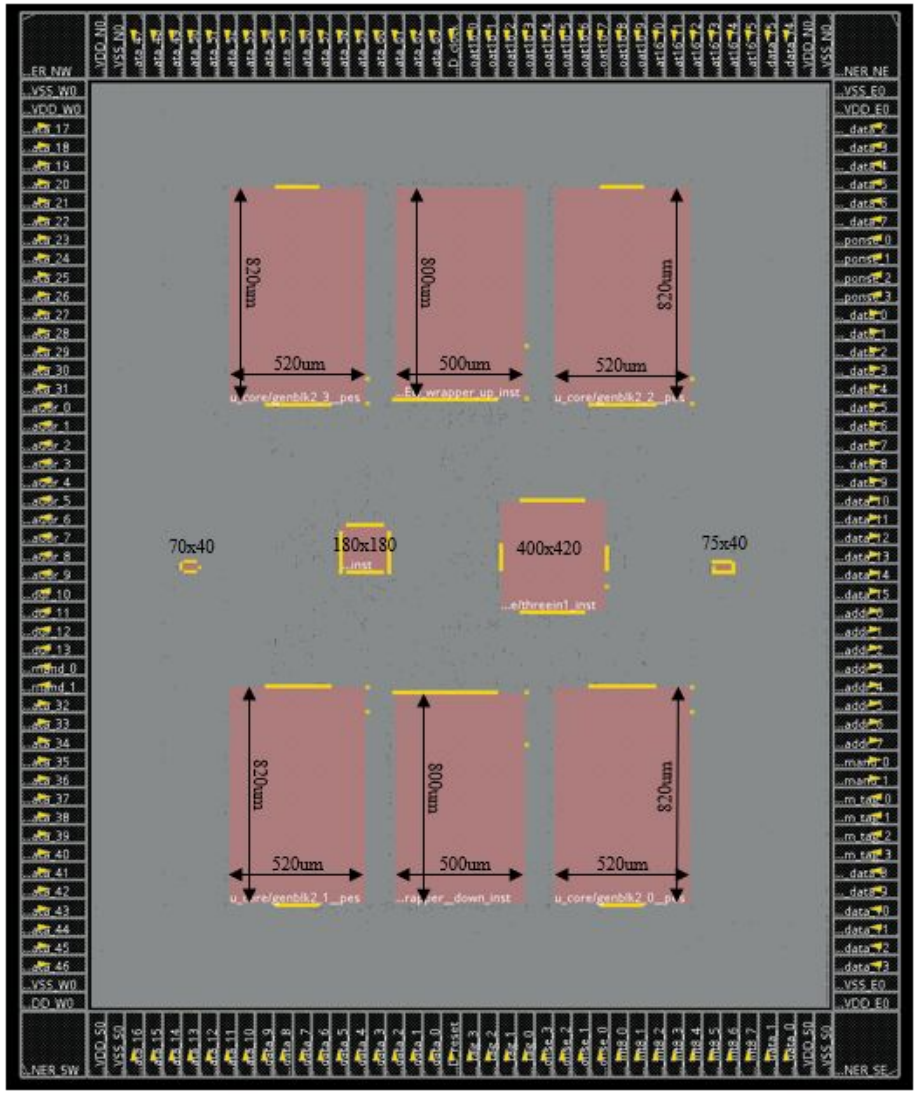
Floorplan

- EQ are divided into 2 groups.
- PEs are of similar distance to shared resources.
- Shared resources a group together to get better area efficiency.
- Gaps are reserved for top level routing.
- Total area estimation
  - Width: 1.8 mm
  - Height: 2.1 mm
  - Area: 3.78 mm^2



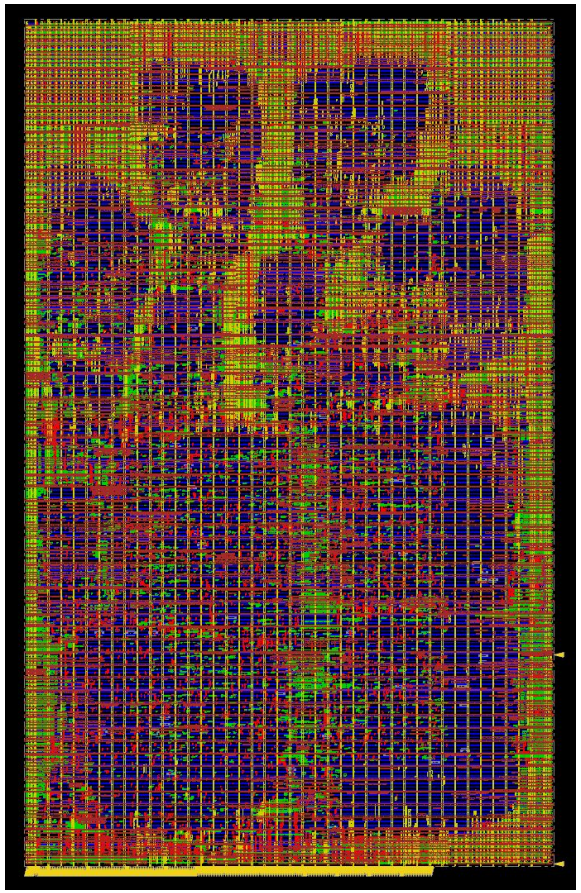
Floorplan

- To resolve DRC by increasing module density, the flattened center module had to be divided into subblocks
- For better routing to and from pads, modules are placed with more gaps in between
- Area with Pad ring:
  - Width: 3.3mm
  - Height: 4mm
  - Area: 13.2mm^2

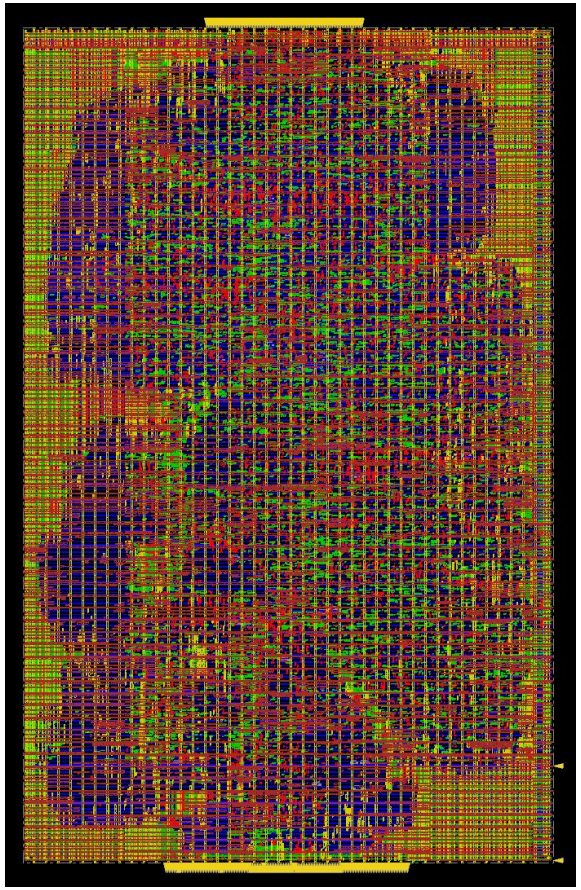


Module Level Layout

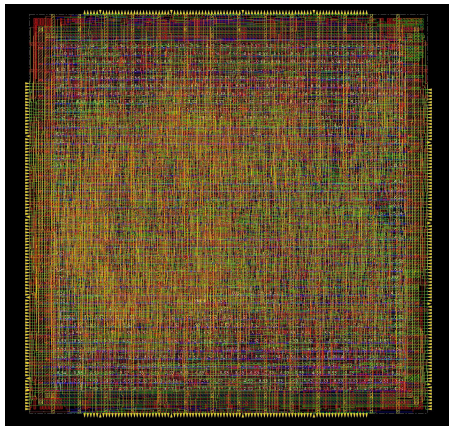
EQ (4 bins): 500x800



PE: 520x820

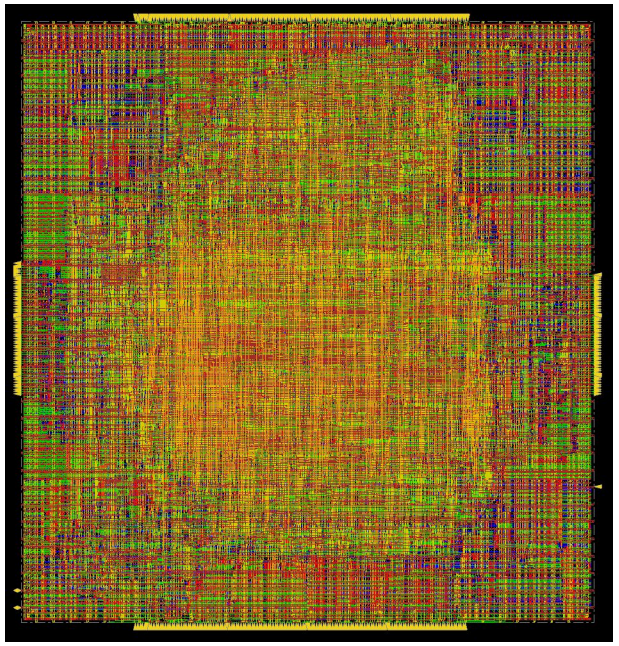


xbar: 180x180

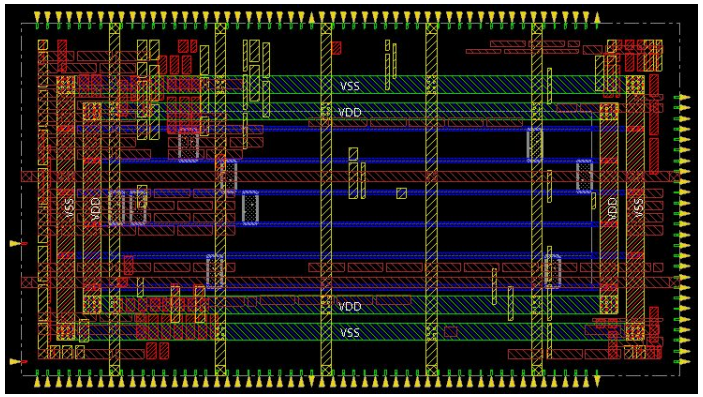


Module Level Layout

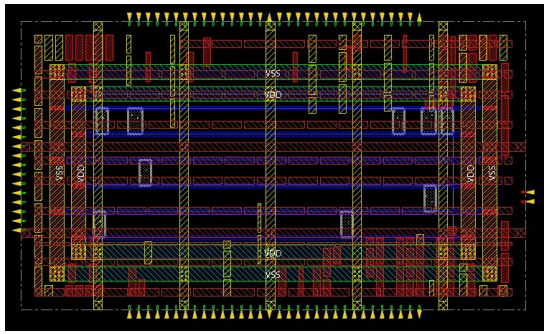
3in1: 400x420



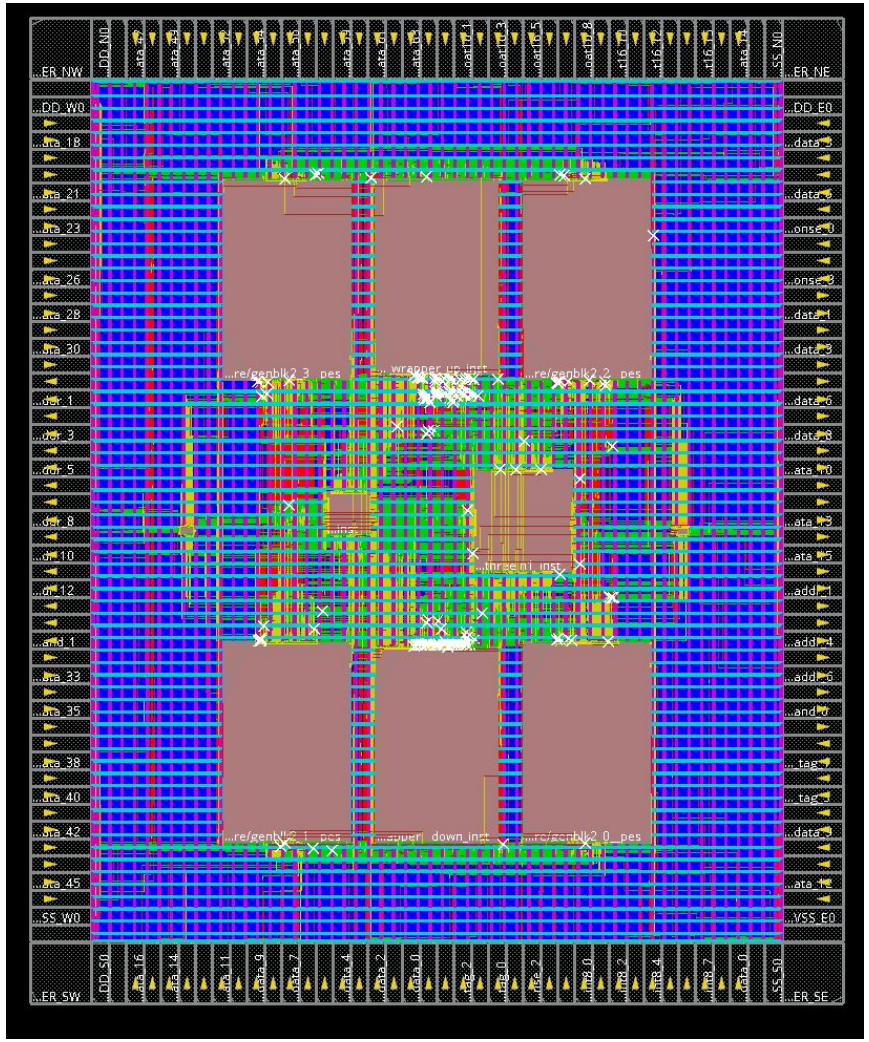
MC_VM: 75x40



MC_EM: 70x40



Top level Layout



Golden Brick Methodology

- A cycle accurate golden brick (Verilog style) in Python
  - global variables in io_port.py

    signal, signal_n
  - one_clock() method in each module class

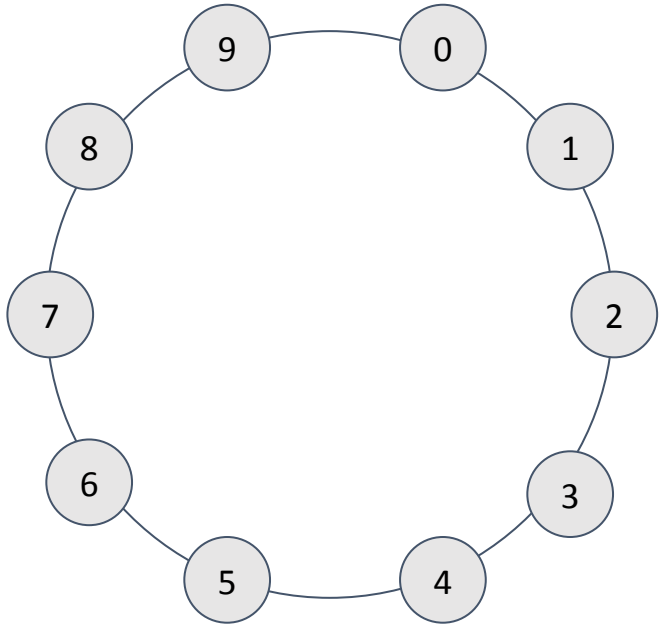
    io_port.signal_n = …
  - update() in the main function

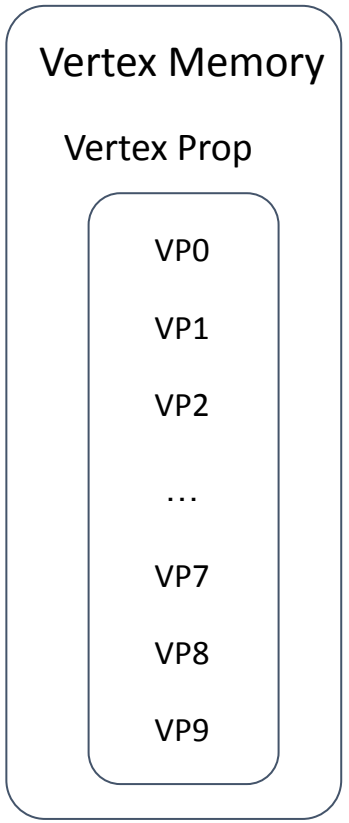
    io_port.signal = io_port.signal_n
- Advantage: easy to transfer into Verilog code
- Disadvantage: inefficient in development

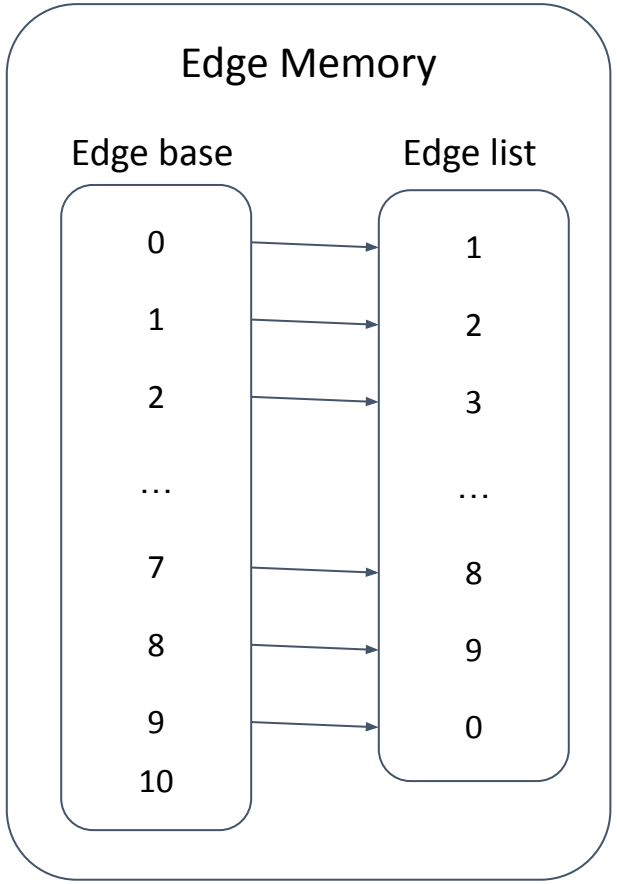
```
in main function:
        module0.one_clock()
        module1.one_clock()
        module2.one_clock()
        …
        update()
```

Simulation Vector

- PageRank on a 10-vertex graph in the shape of a ring







Pre-synthesis Verification

- Verification in SystemVerilog
- Compare the golden output with RTL output
  - All vertex values end up with 0.1

```
vertex mem [          0] = 2e61
vertex mem [          1] = 2e61
vertex mem [          2] = 2e61
vertex mem [          3] = 2e61
vertex mem [          4] = 2e61
vertex mem [          5] = 2e61
vertex mem [          6] = 2e61
vertex mem [          7] = 2e61
vertex mem [          8] = 2e61
vertex mem [          9] = 2e61
Time out @       5001
$finish called from file "testbench.sv", line 174.
$finish at simulation time                250275000
           V C S   S i m u l a t i o n   R e p o r t
Time: 250275000 ps
```

Post-synthesis Verification

- **Problem**
  - Synthesized under 10ns without negative slacks
  - Many X values in netlist
- **Cause - X propagation in netlist**
  - Some values were assigned X in RTL
  - The X values propagated through the netlist
    - Unable to detect in RTL
- With modification on RTL code, the top level netlist pass the simulation under the clock cycle of 22ns

Design Metrics

	Area(um^2)	Density	T_clk(ns)	Setup	Hold
Xbar	180x180	95.97%	10	5.001	0.061
EQ	500x800	52.48%	10	0.289	0.066
MC_EM	70x40	75.84%	10	8.338	0.179
MC_VM	75x40	96.93%	10	8.216	0.178
PE	520x820	61.16%	10	0.431	0.105
3in1	400x420	40.10%	10	0.431	0.105
Top level	3300x4000		11	4.096	0.211

clock uncertainty: 0.1ns

Design Metrics

	Switch Power	Int Power	Leak Power	Total Power
Xbar	0.29	0.762	5.89e+05	1.052
EQ	1.055	5.044	5.36e+06	6.105
MC_EM	4.24e-02	2.87e-02	1.82e+04	7.11e-02
MC_VM	6.06e-02	3.22e-02	2.40e+04	9.28e-02
PE	6.426	5.726	1.31e+07	12.165
3in1	0.571	1.362	1.13e+06	1.934
Top level	0.728	19.549	4.75e+07	20.325

Dynamic Power Units = 1mW, Leakage Power Units = 1pW. Post-synthesis.

Final Status

- All sub-modules' APR are DRC/LVS clean
- Post-synthesis verification passed at clock period of 22ns
- Top Level Integration:
  - Pad ring, floorplan and routing are proved to be feasible
  - Power grid generation is not successful. Possible reason:
    - Sub-modules are configured to use M1~M6 for routing
    - Sub-module power-grids are mistakenly placed on M4 & M5
    - The top-level APR cannot correctly connect top-level power grid (M7 & M8) to sub-module power.
    - Not enough time to rerun all the sub-module APR.

Looking back…

- Graph computation is fun.
- A novel architecture without published implementation work.
- No open-source simulation code. Have to construct golden brick from scratch.
- Complicated RTL design and verification, due to its asynchronous nature.
- DRC/LVS clean sub-modules does not necessarily lead to top-level success.
- Should have paid more attention to power grids.