SFU SIMON FRASER UNIVERSITY

X-Cache: A Modular Architecture for Domain-Specific Caches

Ali Sedaghati, Milad Hakimi,
Reza Hojabr, Arrvindh Shriraman

Session 5B June 21@9:30, ISCA 2022

Executive Summary

Observation: Emerging DSAs work with indexed data structures

Non-Affine, Pointers-Chasing, Dynamic accesses, Irregular access pattern

Challenges: How to capture reuse? How to orchestrate data movement?

Our Proposal : A portable cache template for DSAs.

Meta-Tag: Tag on-chip data with DSA fields +No walk on hits

Cache Walkers: Microcoded cache controller +DSA programmable

Outcome:

Toolflow to generate cache RTL for your DSA (github.com/sfu-arch/X-Cache)

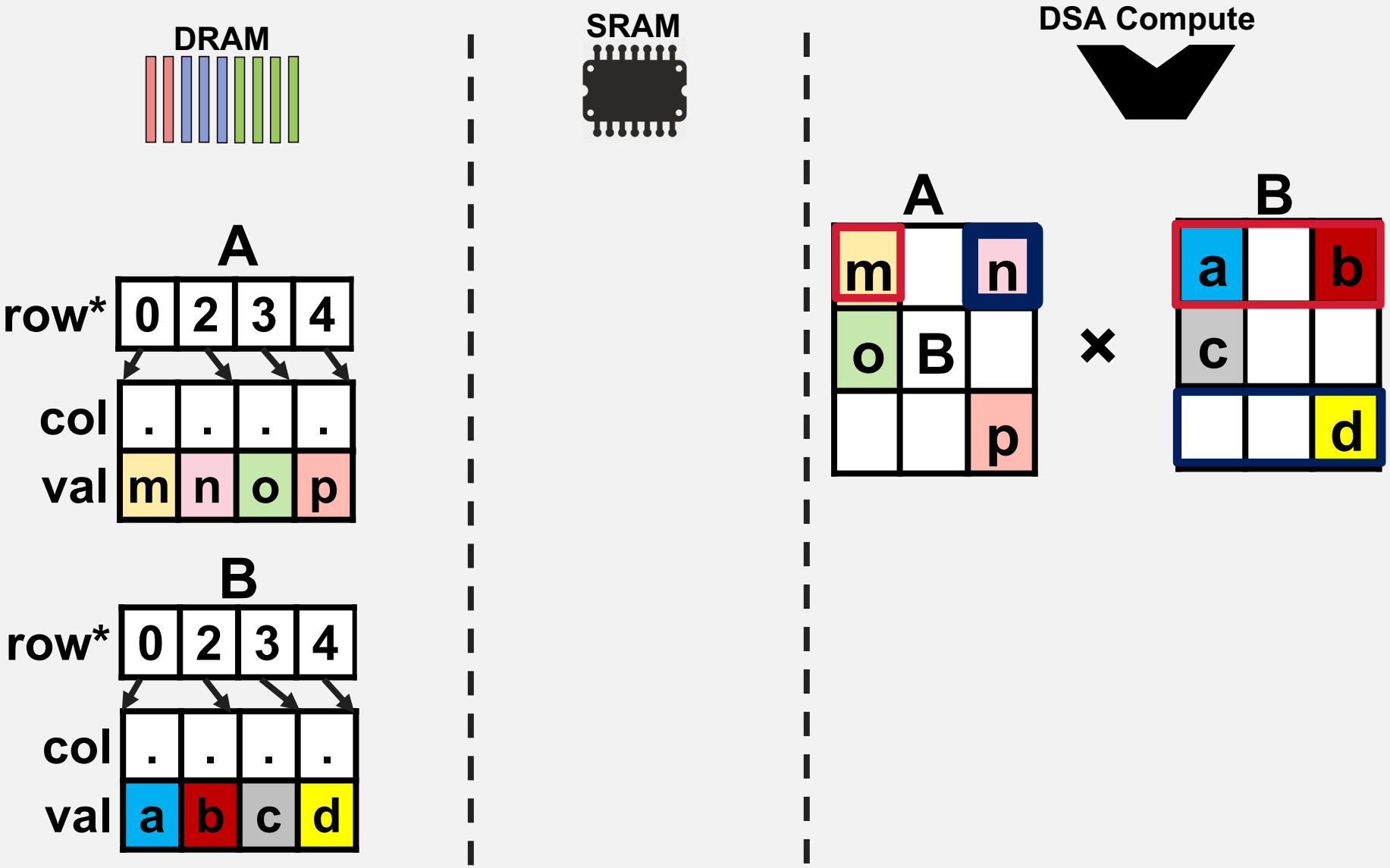
vs Conventional caches. 1.7x performance, 50% lower power, 2-8x lower bandwidth

Outline

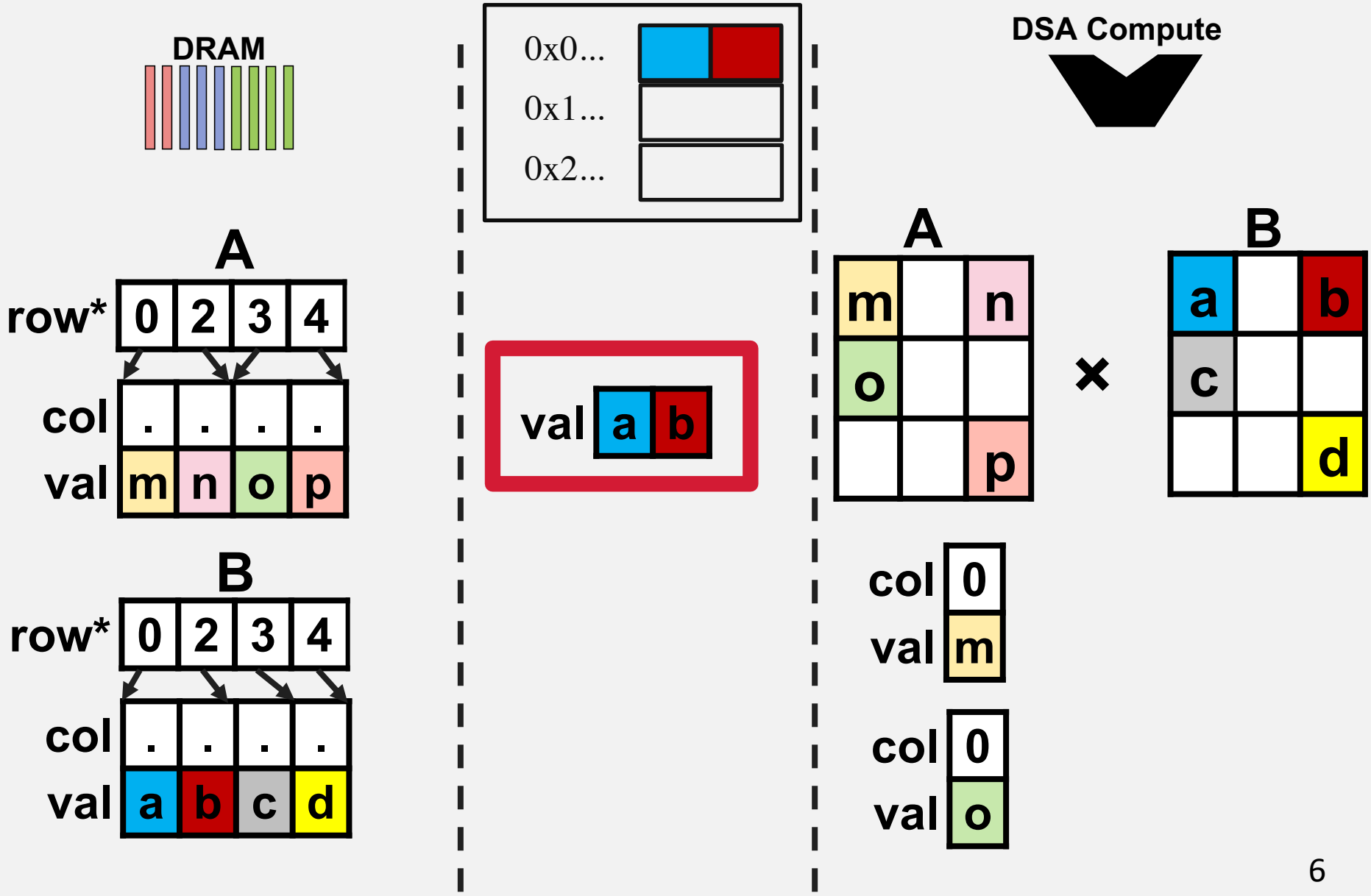
- Challenges in designing a SpMM DSAs
- X-Cache : Architecture and Execution Model
- Evaluation Summary

Challenges in emerging DSAs

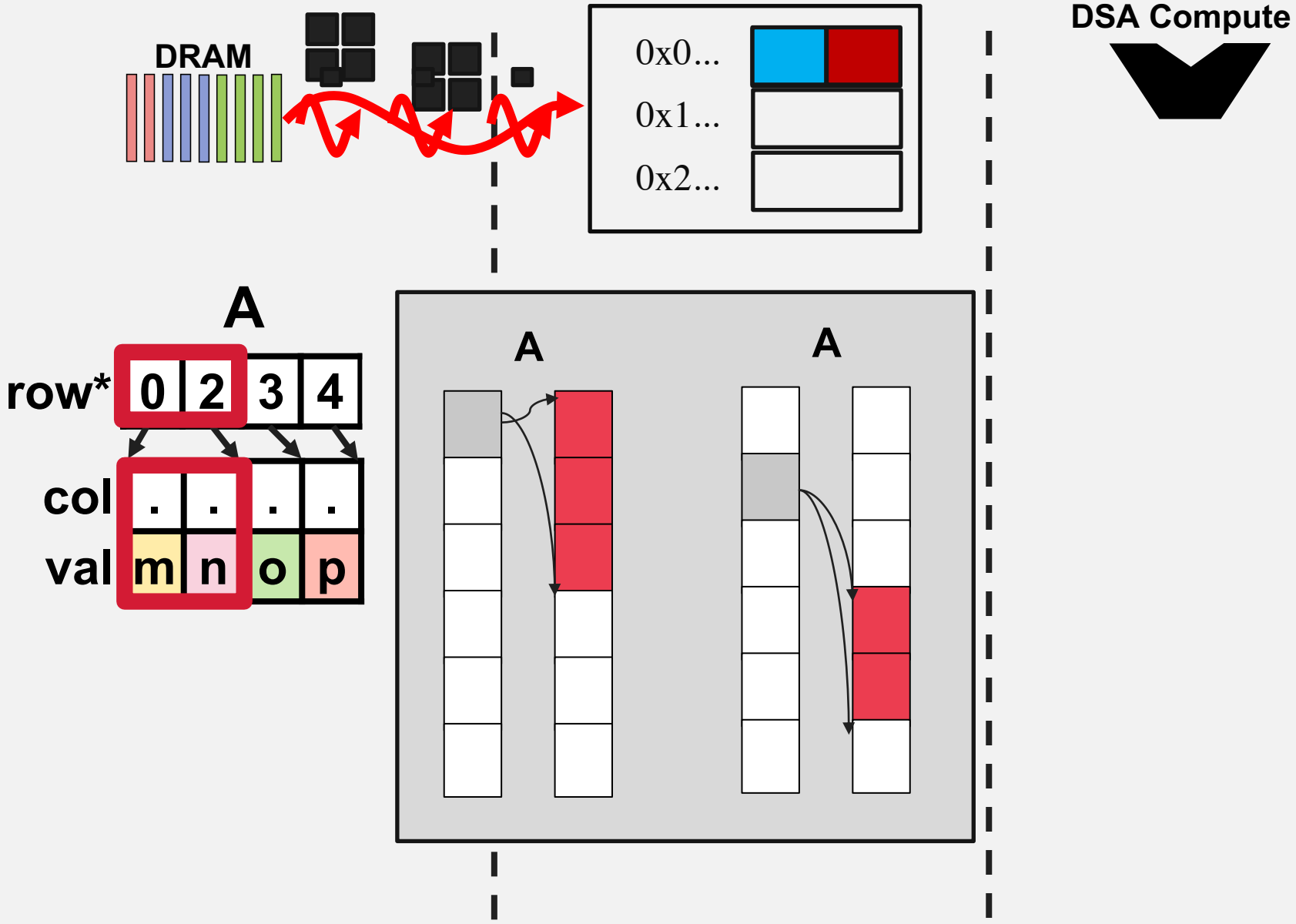
Building an SpMM



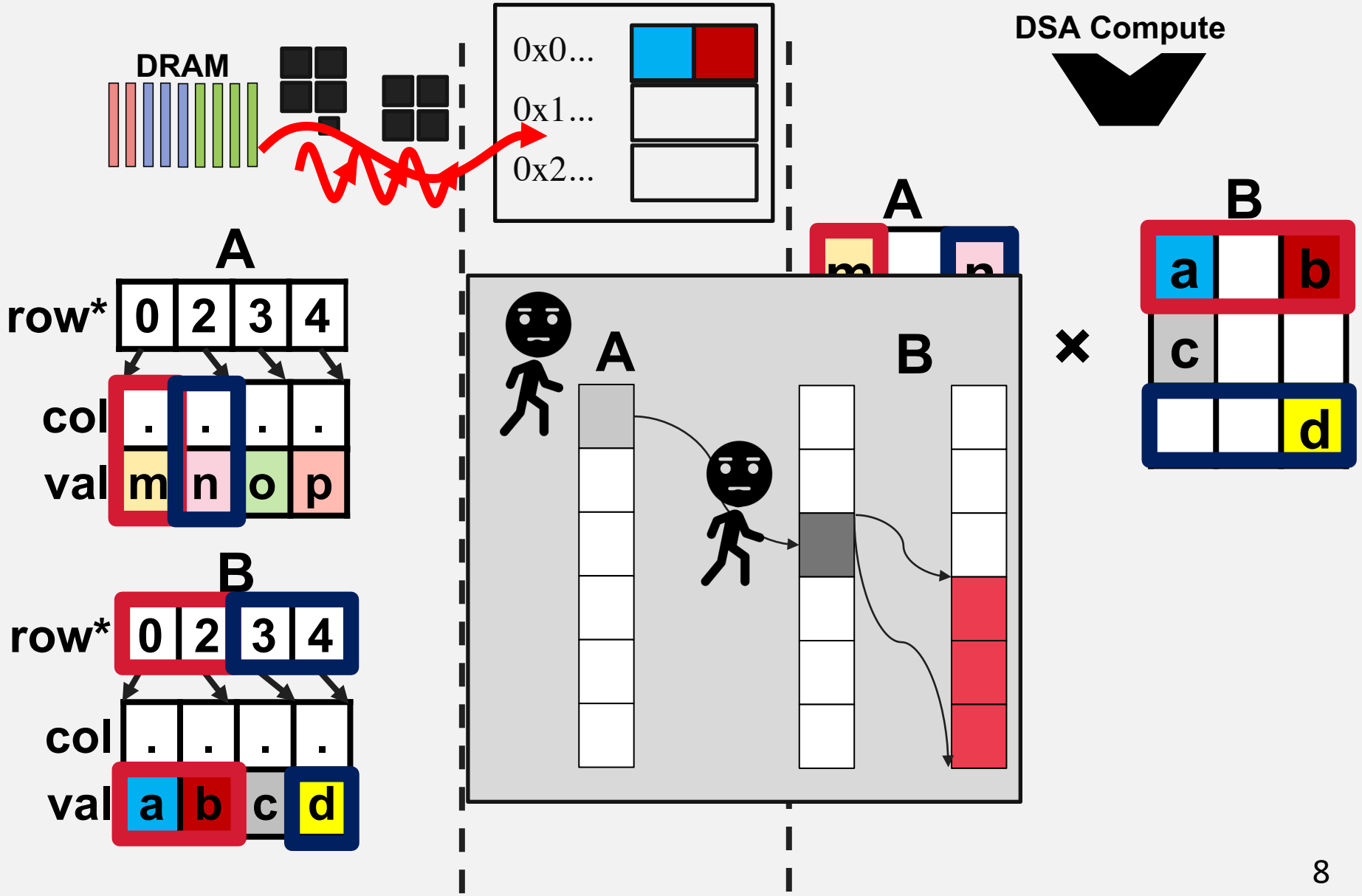
SpMM Challenge 1 : Reuse



SpMM Challenge 2: Non-affine DS



SpMM Challenge 3: Dynamic Access



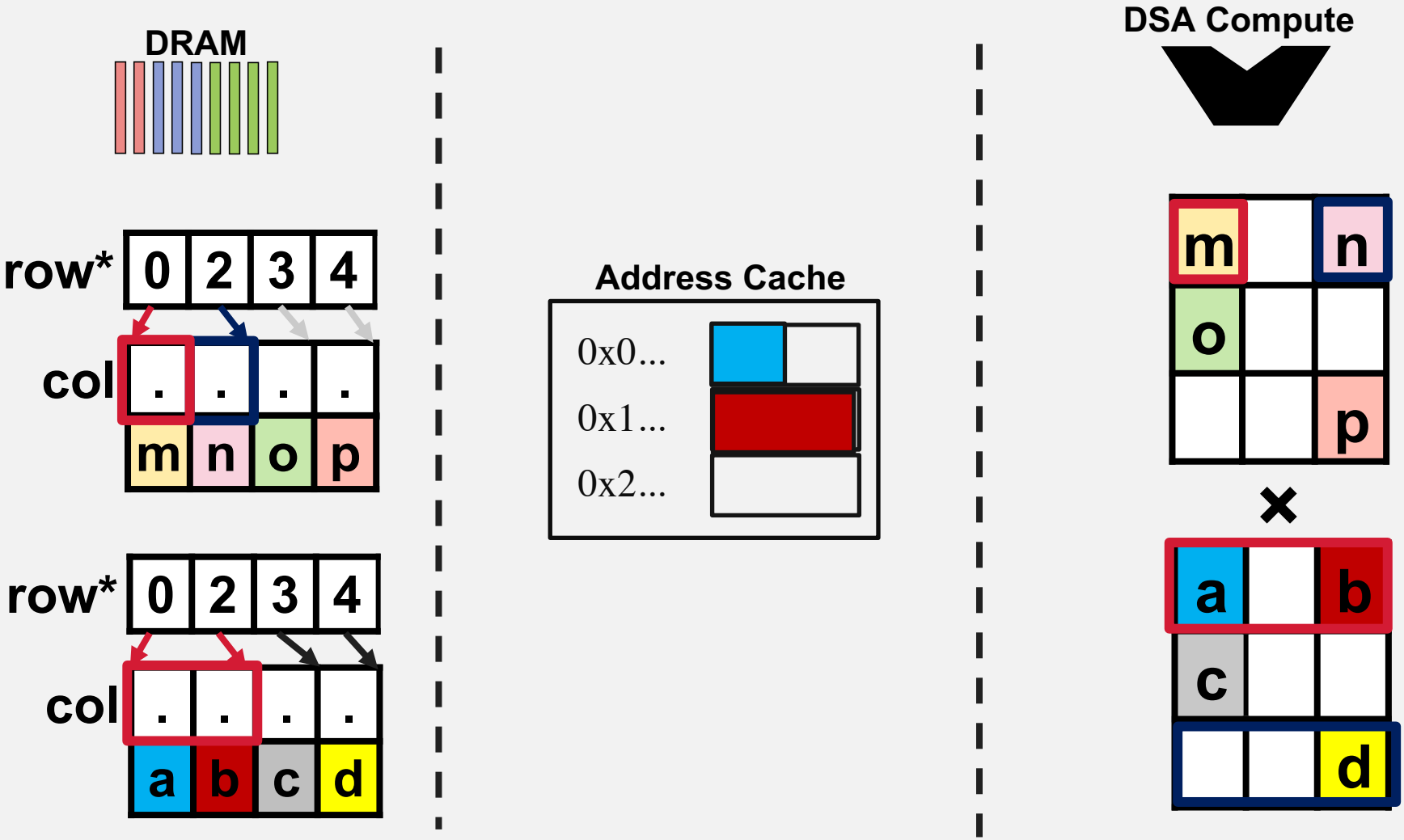
X-Cache

Idea 1: Replace address tags with meta-tags

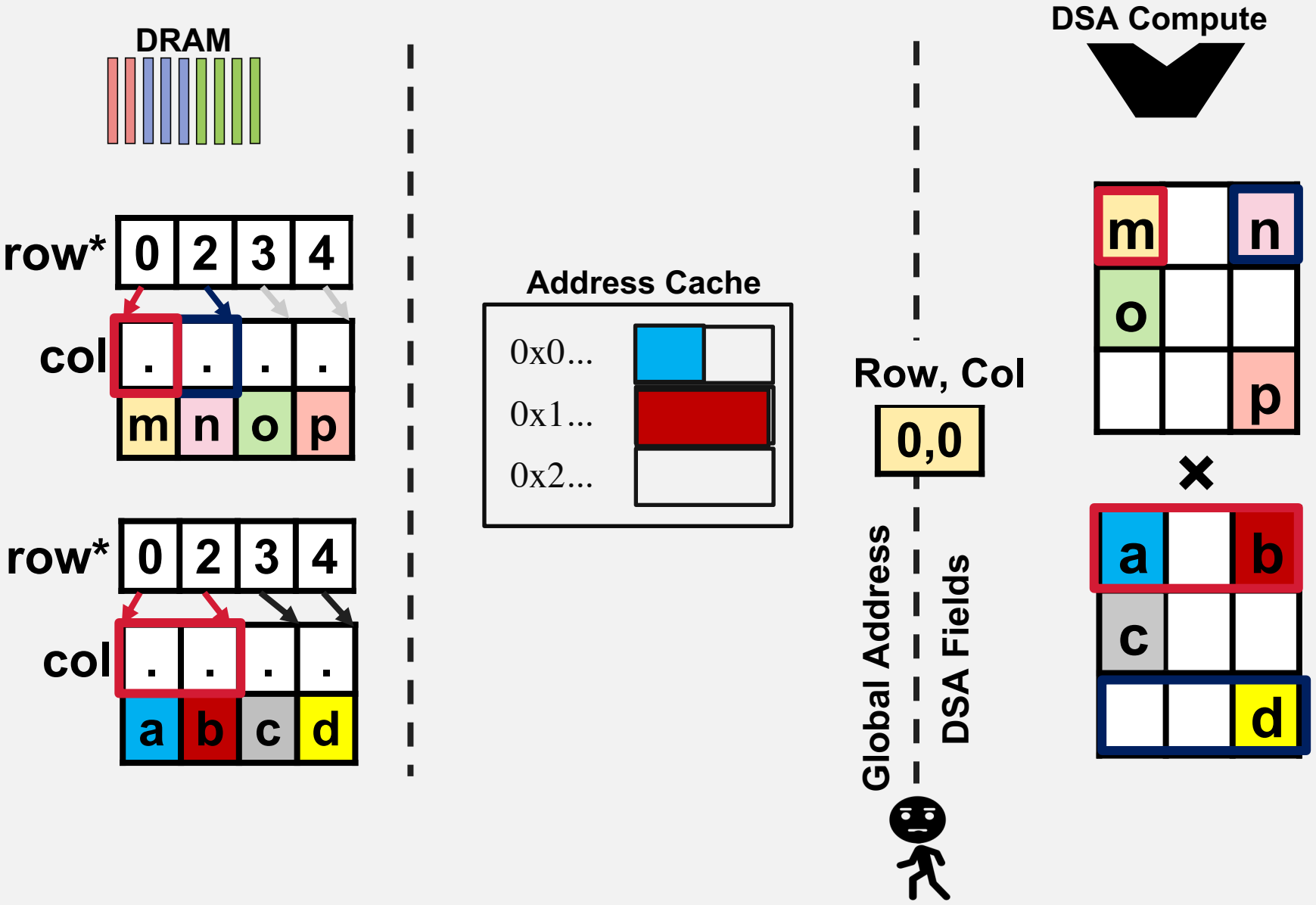
Idea 2: Integrate DSA walkers into cache

Idea 3: Coroutine based cache controller

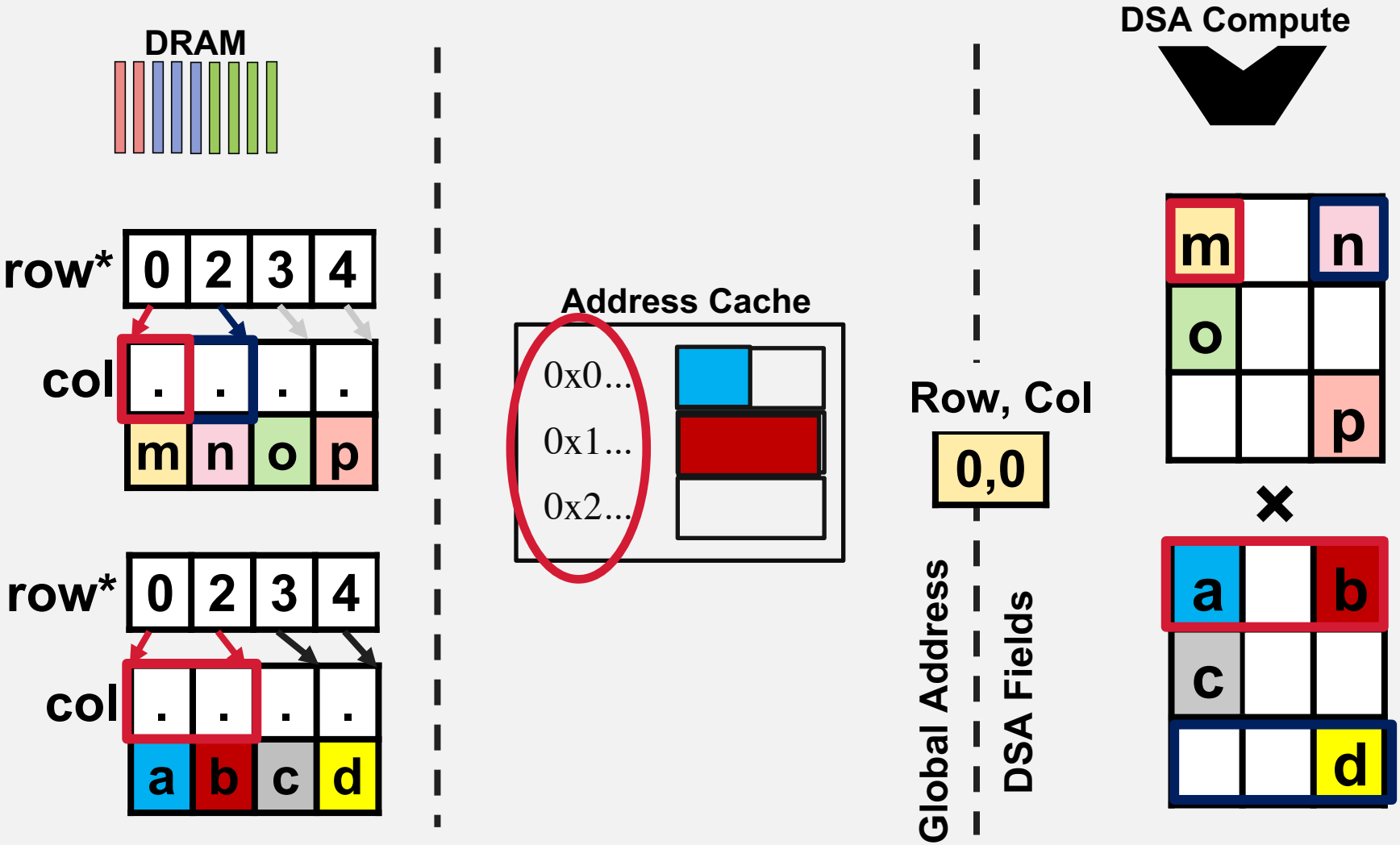
Idea 1: Replace Address Tag with Meta Tag



Idea 1: Replace Address Tag with Meta Tag

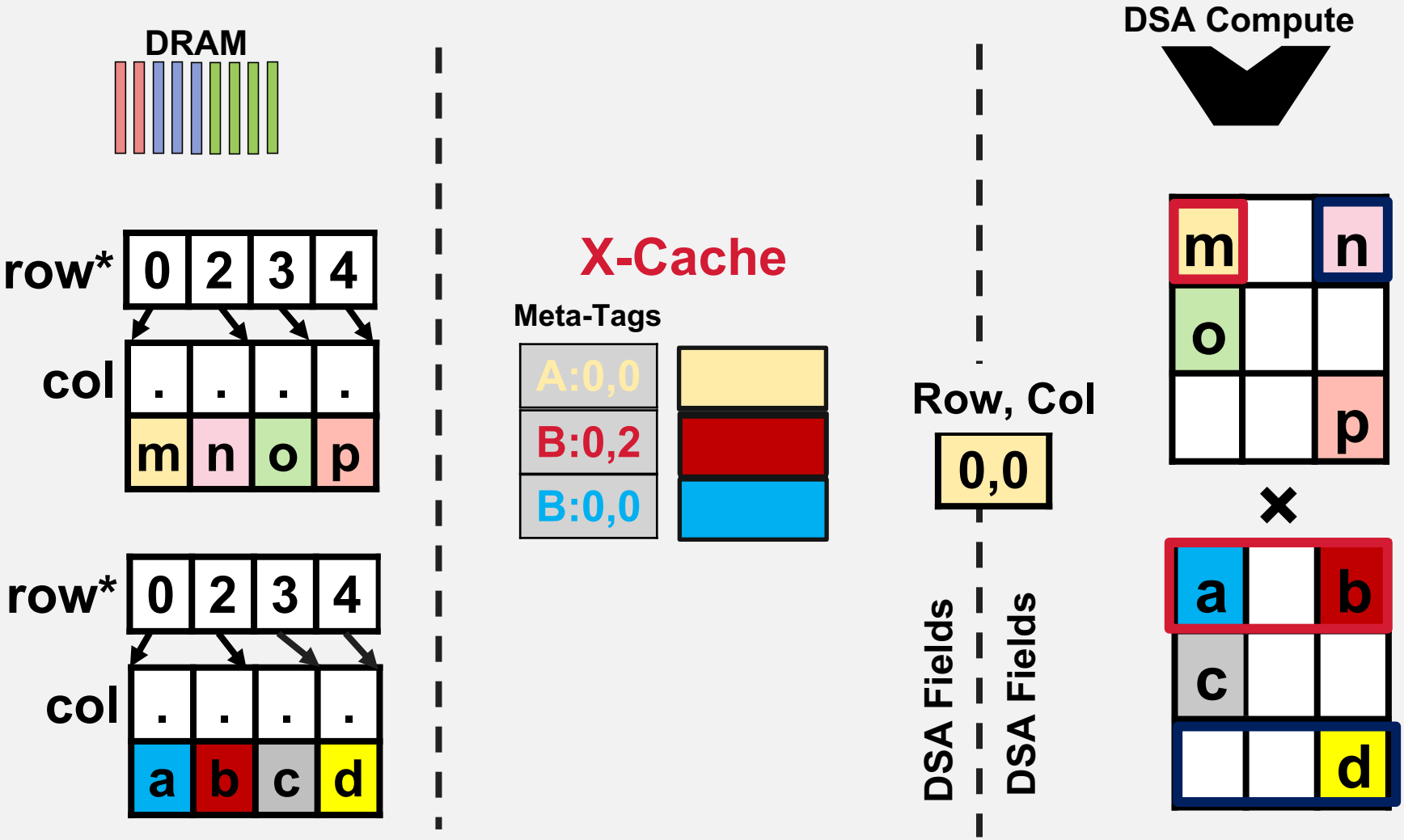


Idea 1: Replace Address Tag with Meta Tag



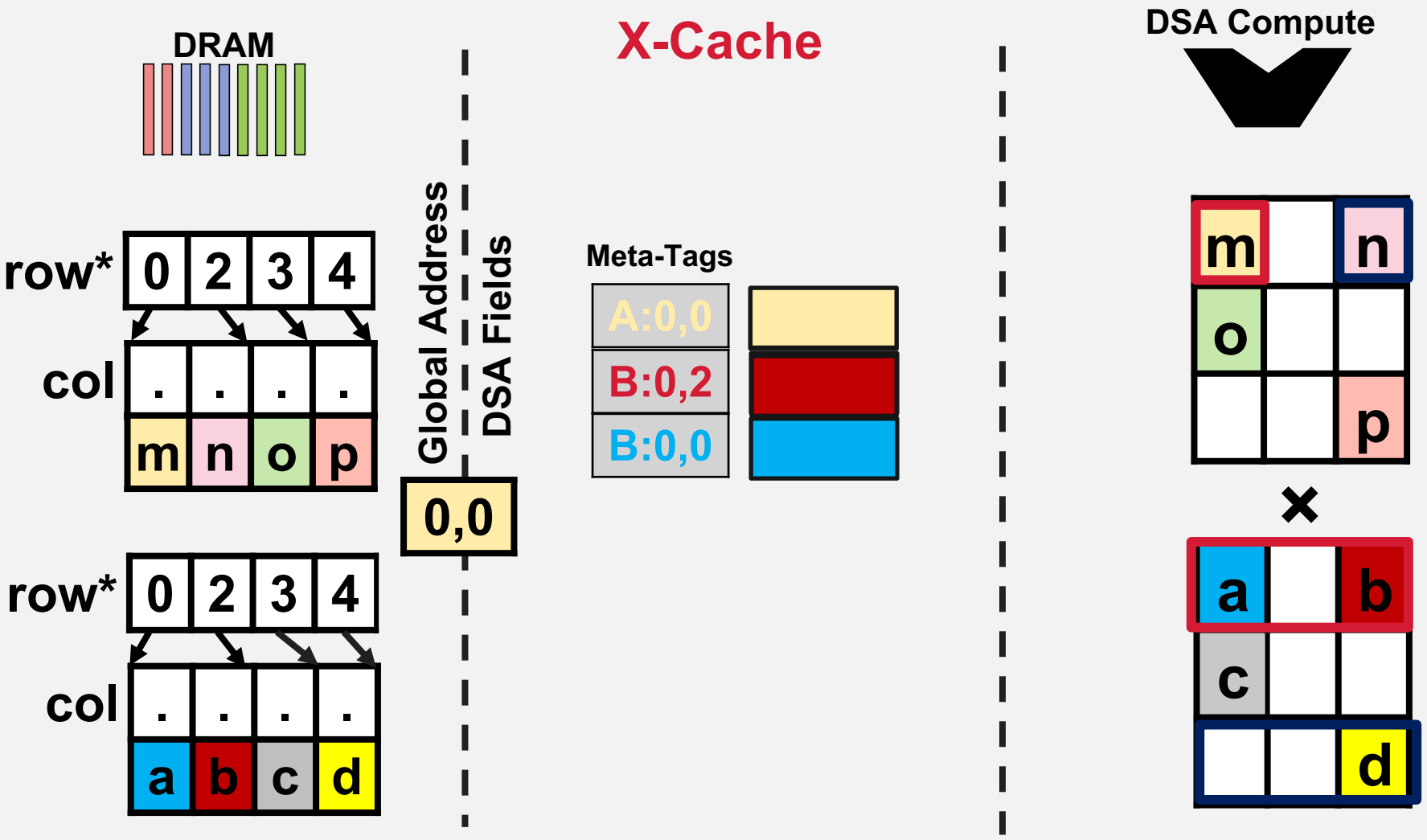
Data in cache, but address tags cannot recognize.

Idea 1: Replace Address Tags with DSA Tags



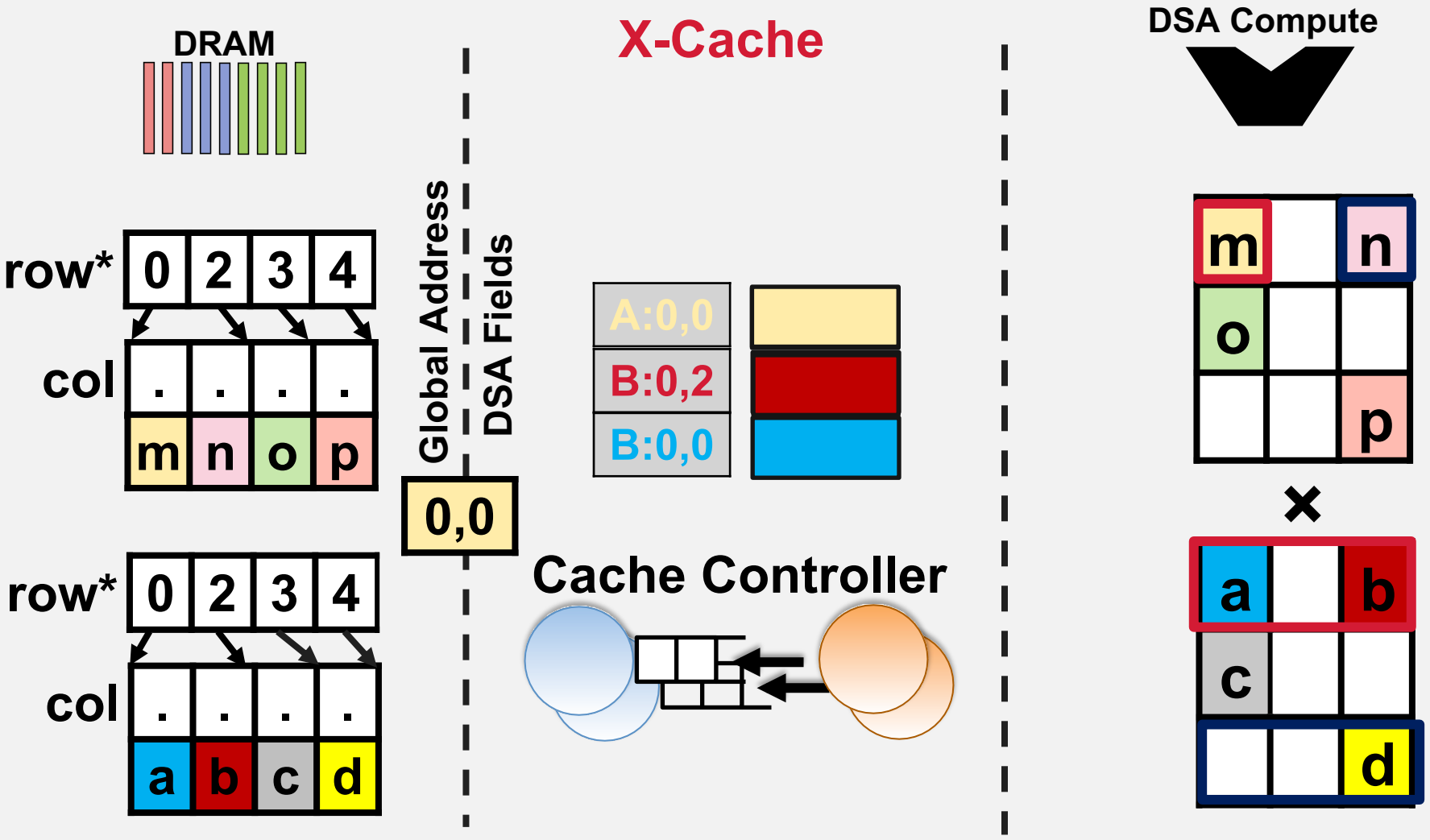
Tag match using DSA-specific fields

Idea 2: Embed Walkers in Cache Controller



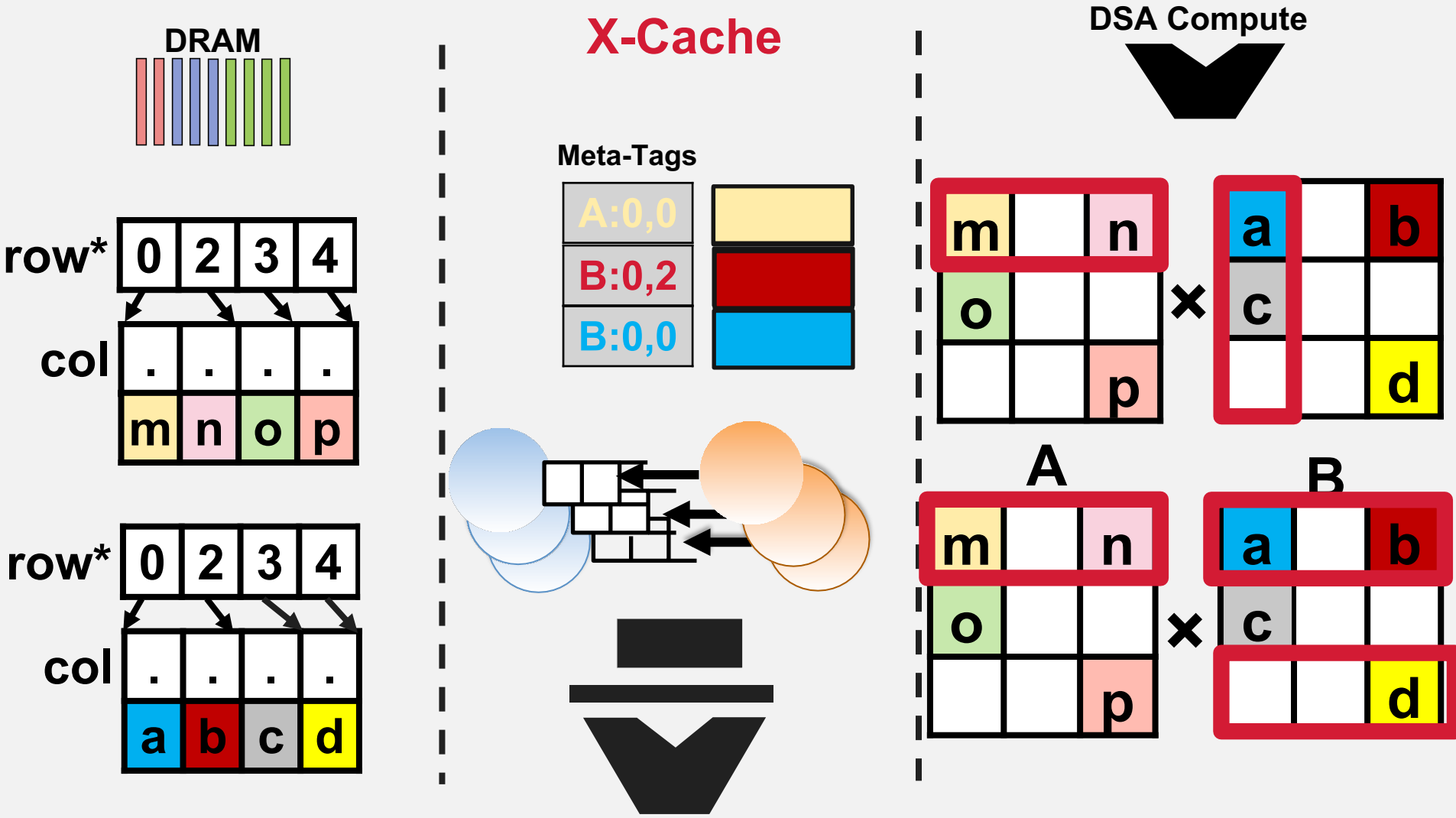
Cache misses now need walks and translation

Idea 2: Embed Walkers in Cache Controller



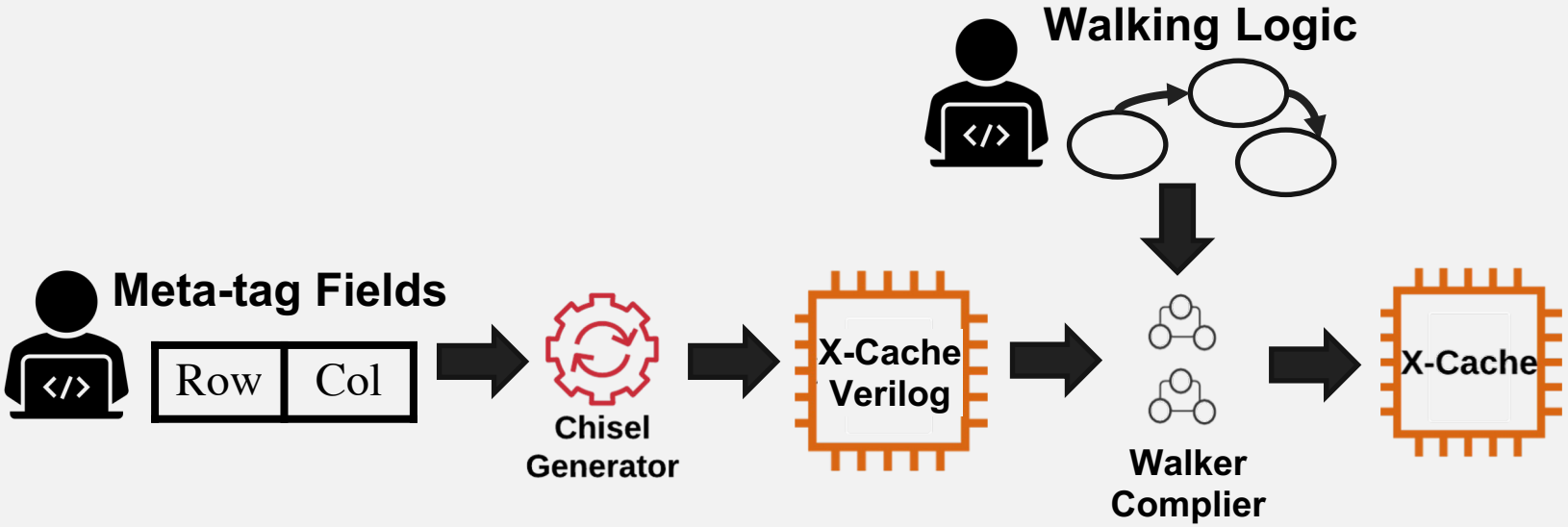
Integrate cache orchestration and DSA Walker

Idea 3: Programmable Controller



Programmable microcode and parallel coroutines

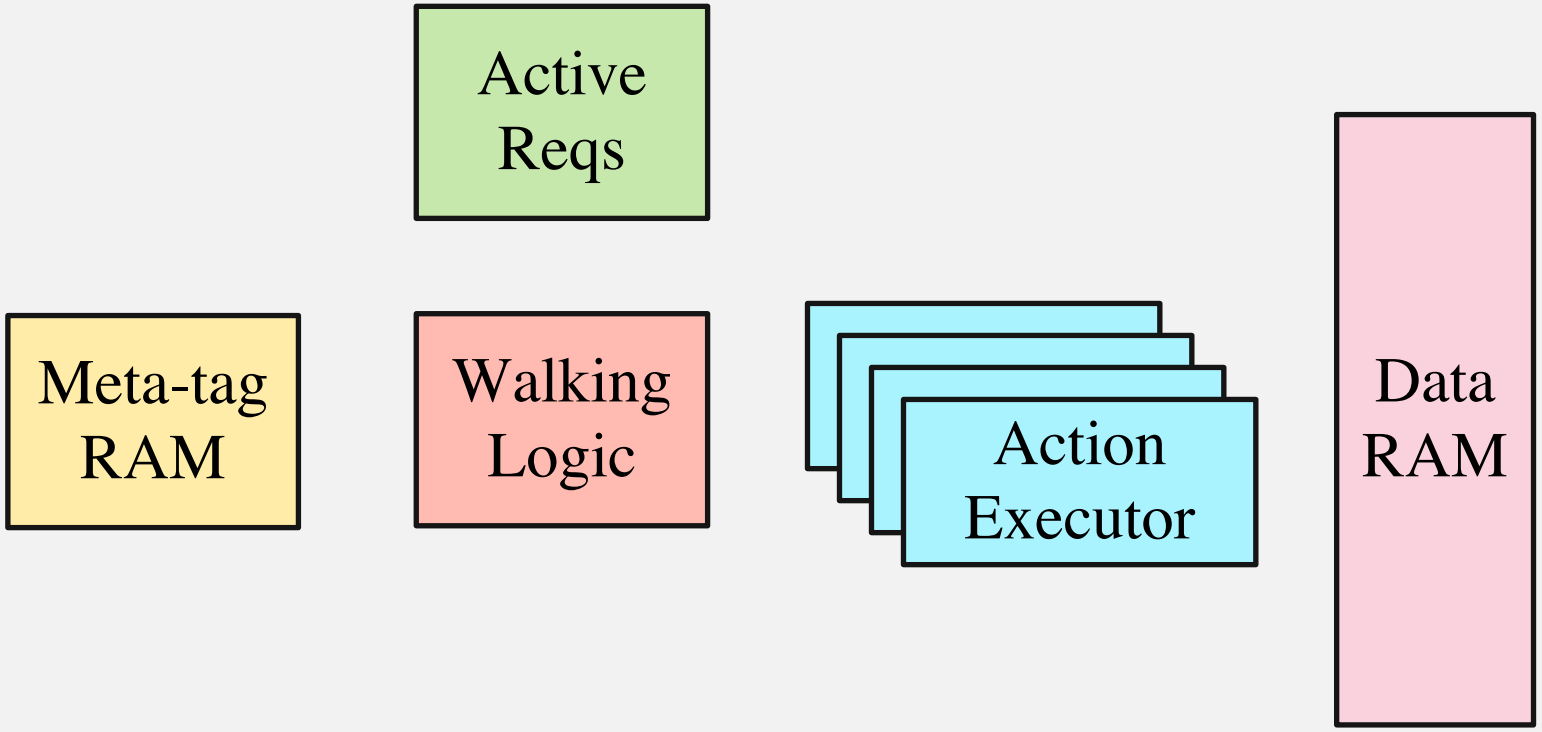
X-Cache Toolchain



A toolchain for:

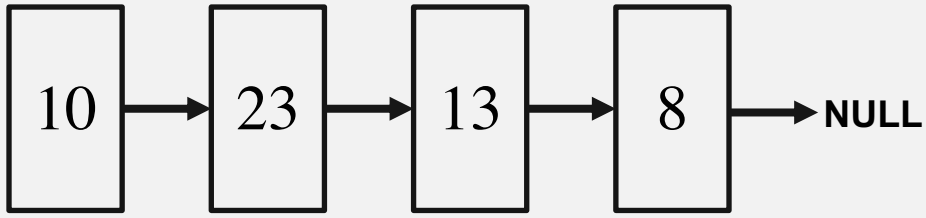
- Generating meta-tag arrays and DSA cache internals
- Programming cache controllers for data movement
- Parameterizing cache controller for DSAs

X-Cache microarchitecture

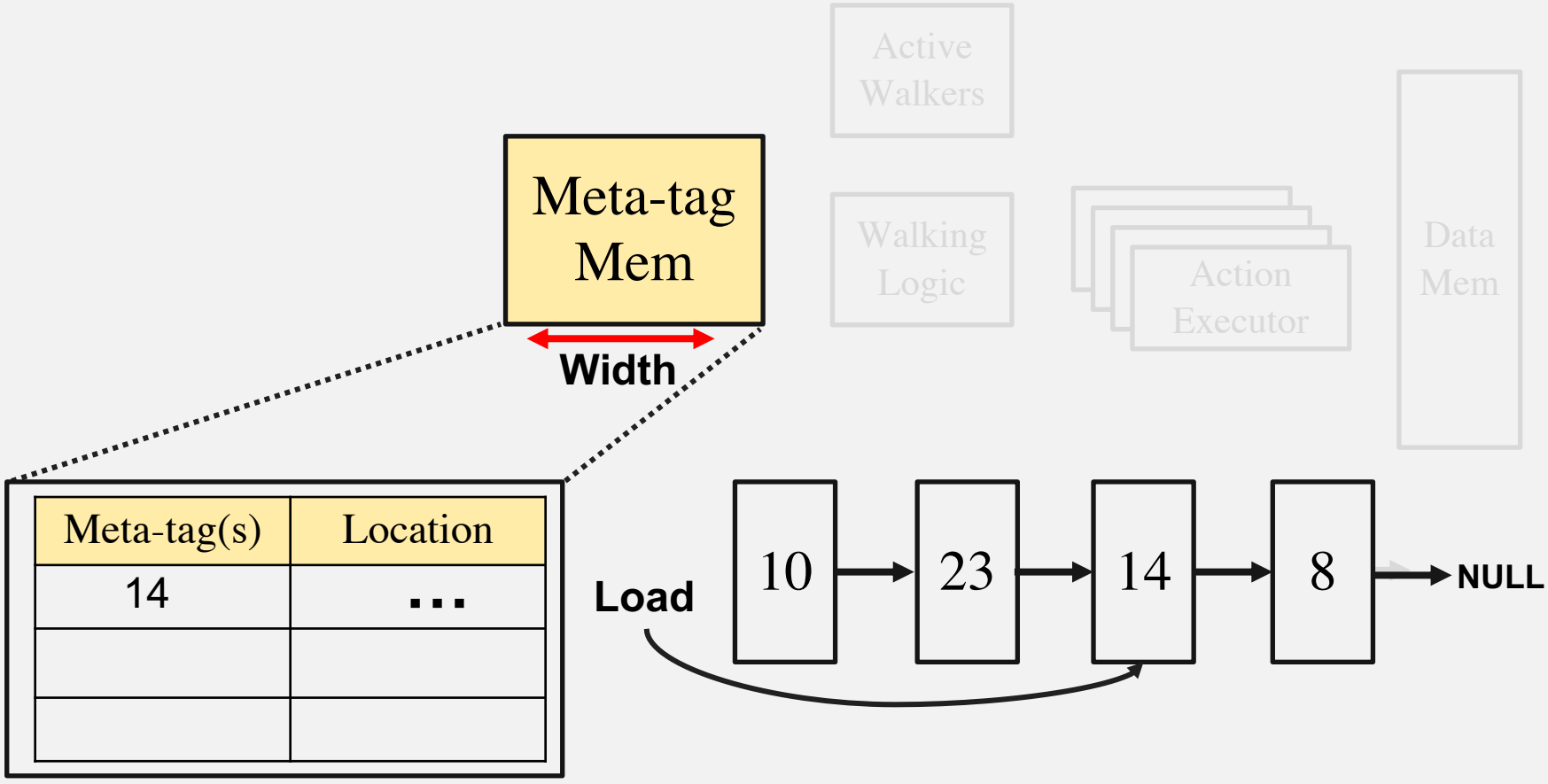


Index-Cache Example

```
while ( cur.key != meta)
        cur = cur → next
return cur.payload
```

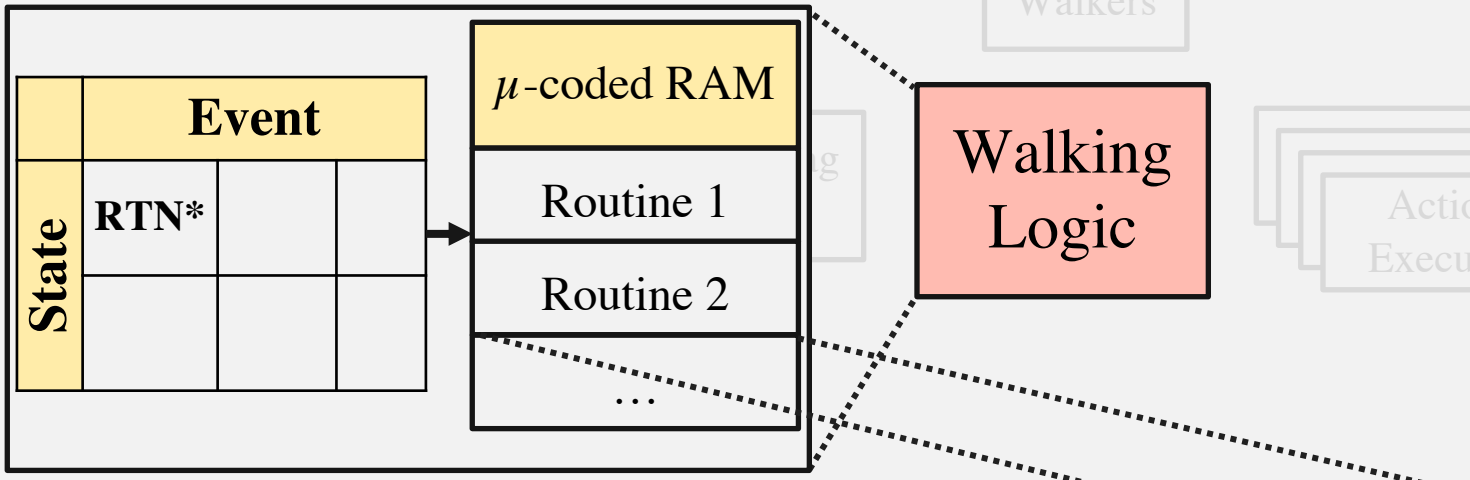


µ-architecture: Meta-tag



No walking when it's a hit

µ-architecture: Walker

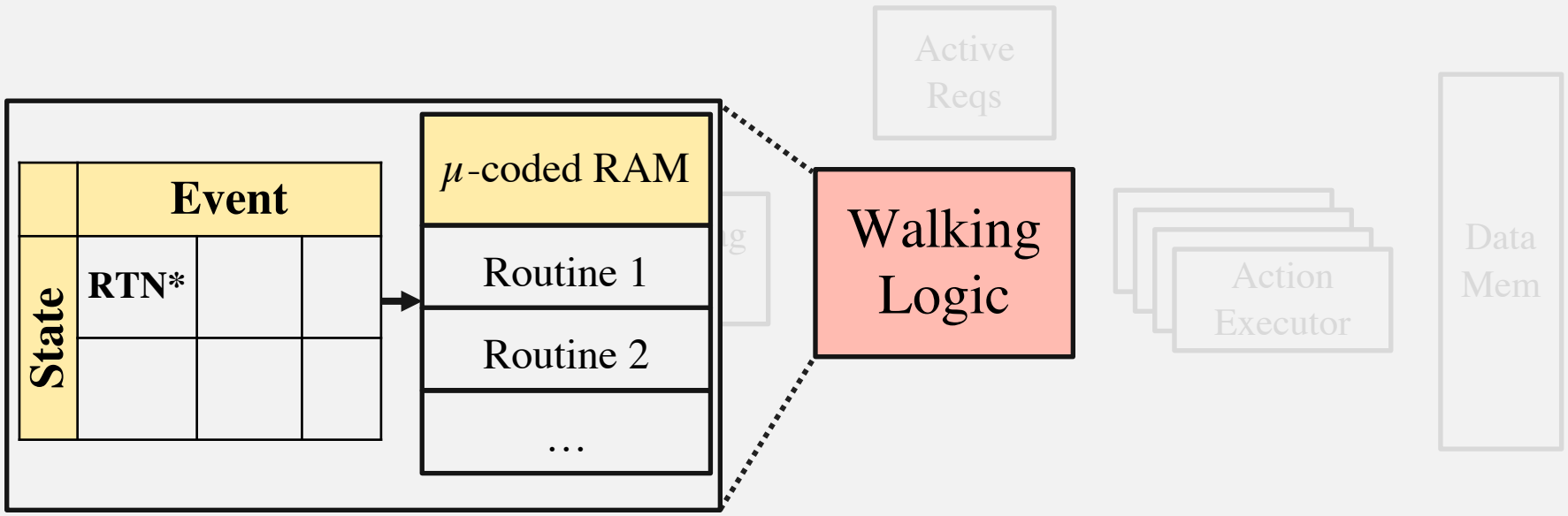


	µcode actions
AGEN	add, and, or, xor, addi, inc, dec, shl, shr, sra, srl, not, allocR
Queues	enq, deq, read-data, write-data, peek
Meta-tags	allocM, deallocM, update, state
Control	bmiss, bhit, beq, bnz, blt, bge, ble
DataRAM	allocD, deallocD, read,write

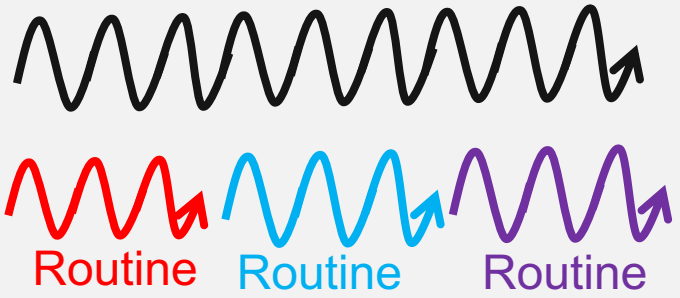
Routine X
Action 1
Action 2
Action 3

Embedding walkers in the cache controller

µ-architecture: Walker




```
while ( cur.key != meta)
    cur = cur → next
return cur.payload
```

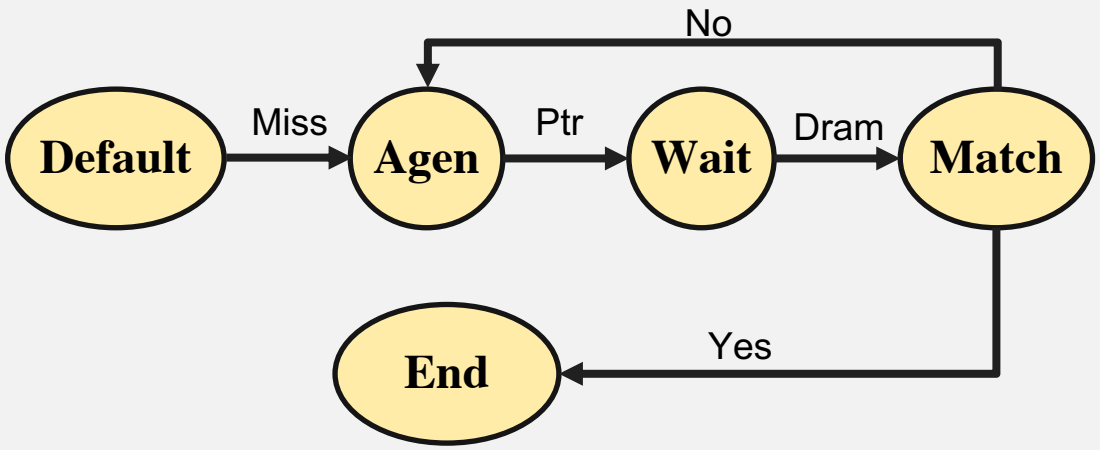


Breaking down the walking into Routines

µ-architecture: Walker Example

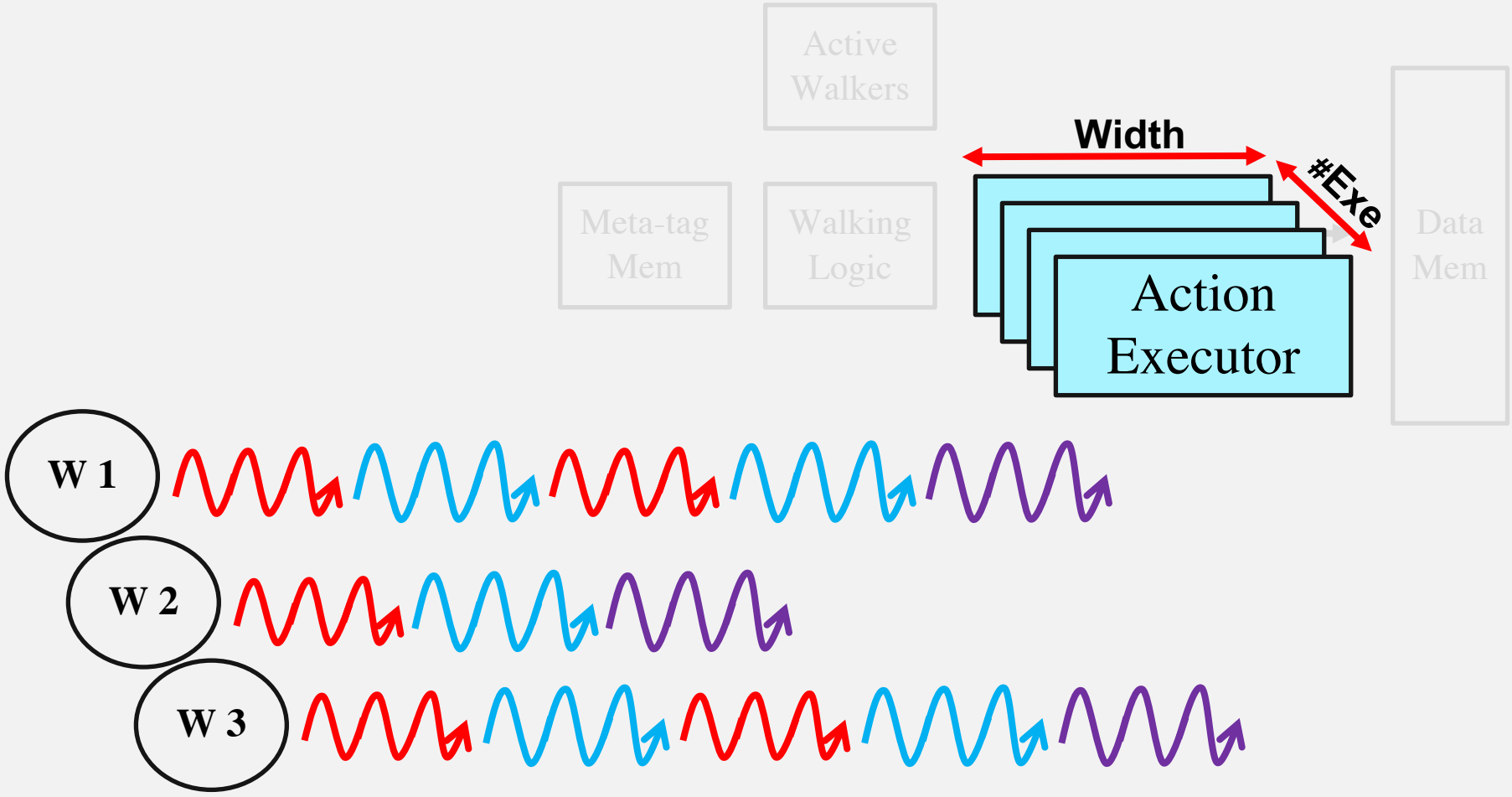
C Walker

while (cur.key != meta)
cur = cur → next
return cur.payload



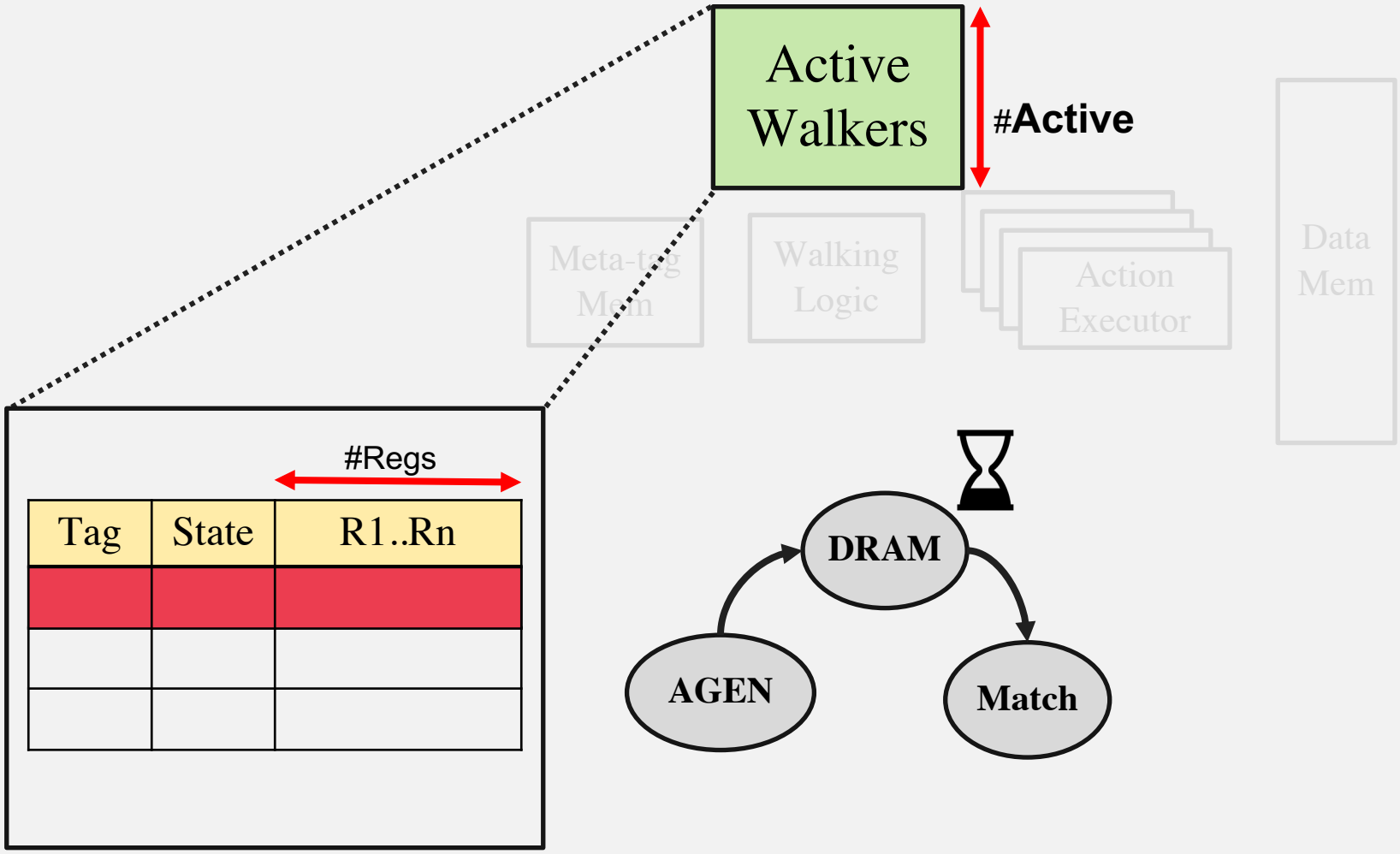
(State, Event)→ Routine : Sequence {Actions}		
(Default, Miss)	→ MISS:	{allocD, allocM, enq Ptr, state Agen}
(Agen, Ptr)	→ AGEN:	{allocR, add, enq DRAM, state Wait}
(Wait, Dram)	→ PEEK:	{…}
(Match, Check)	→ CHECK:	{…}

µ-architecture: Executors

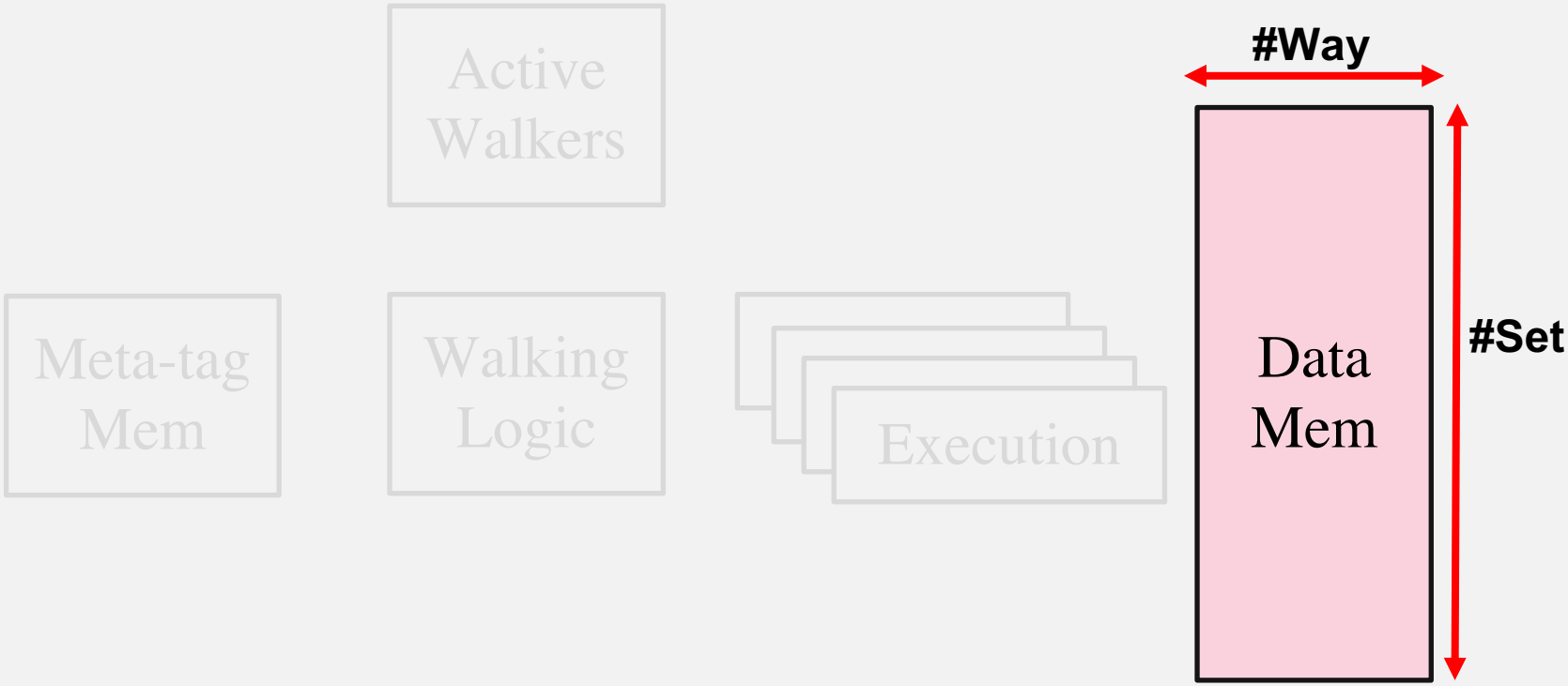


Executing multiple parallel walkers

µ-architecture: Active Walkers



μ-architecture: Data Mem



Evaluation

Evaluation

Setup: Verilator RTL Simulation, FPGA synthesis

DSAs: Widx [MICRO'10], Sparch[HPCA'21],
DAS-X [ICS'15], Graphpulse [Micro'21] Gamma [ASPLOS'20]

Performance:

- *1.7x higher vs. address-based caches.*
- *50% higher for some DSAs by shorting walks*

Bandwidth:

- 2x to 8*x lower vs. address-based cache (no walks on hits)*

Power:

- *25-80% lower vs. address-based cache*
- *Meta-tags 7% of total, Programmable Controller, 6% of total*

Thanks! Index Cache on 1 Slide

State Transitions

```
val transitions = Array[Transitions] (
    Transition ( Routine(MISS), Trigger(Miss, Default)),
    Transition ( Routine(AGEN), Trigger(Ptr , Agen))          ,
    Transition ( Routine(PEEK), Trigger(Dram, Wait)),
    Transition ( Routine(CHECK), Trigger(Check , Match))
}
```

RTL Parameters

```
trait XCacheParams {
        val nways = …
        val nsets = …
        val xregDepth = …
        val lockDepth = …
        val nExe = …
        val nCache = …
        val nWords = …
}
```

Microcode Routines

```
val routines = Array[Routine] (
    Routine (“MISS” , Seq( “allocD, allocM, enq “Ptr”, state “Agen”)),
    Routine (“AGEN”, Seq( “allocR, add(base, offset) , enq DRAM, state “Wait”)),
    Routine (“PEEK”, Seq( “peek (data), load (R, data.meta), enq “Check”, state “Match”)),
    Routine (“CHECK”, Seq( beq (R, input_key, 2), enq “Ptr”, State  “Agen”,  Write, State “End” ))
)
```

Q/A