Modular SRAM-based Binary Content-Addressable Memories

Ameer M.S. Abdelhadi and Guy G.F. Lemieux

Department of Electrical and Computer Engineering

University of British Columbia

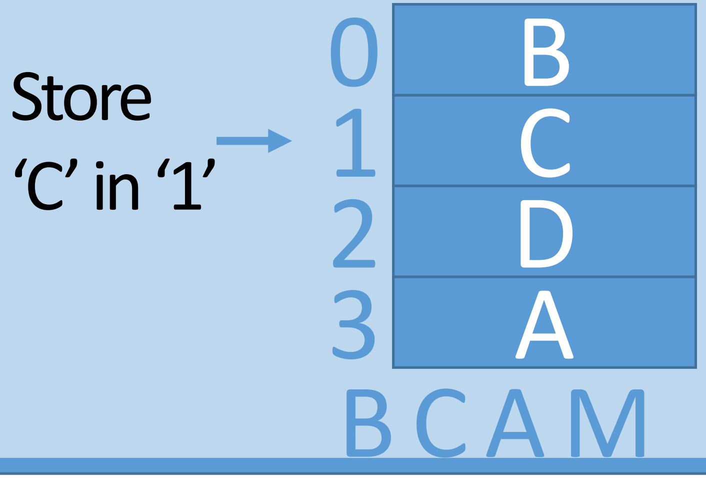
Vancouver, Canada

Binary Content-Addressable Memory (BCAM)

Hardware-based Single-Cycle Parallel Search Engines

Write

Stores new data at specific address



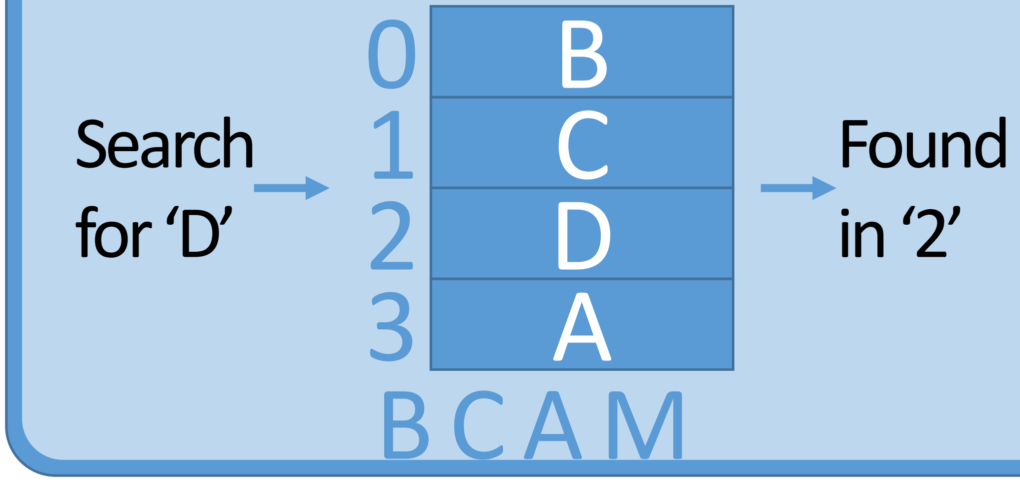
Store ‘C’ in ‘1’

0	B
1	C
2	D
3	A

BCAM

Match

Search all addresses for a given data (pattern)



Search for ‘D’

0	B
1	C
2	D
3	A

Found in ‘2’

BCAM

BCAM Applications

Memory management

- Associative caches
- Translation lookaside buffers (TLBs)

Databases

- Eliminates memory bottleneck

Networking

- IP lookup in routing/ forwarding tables
- Intrusion detection
  - detect predefined suspicious packages
- Packet Classification

Pattern matching

- e.g. DNA sequence lookup

Data compression

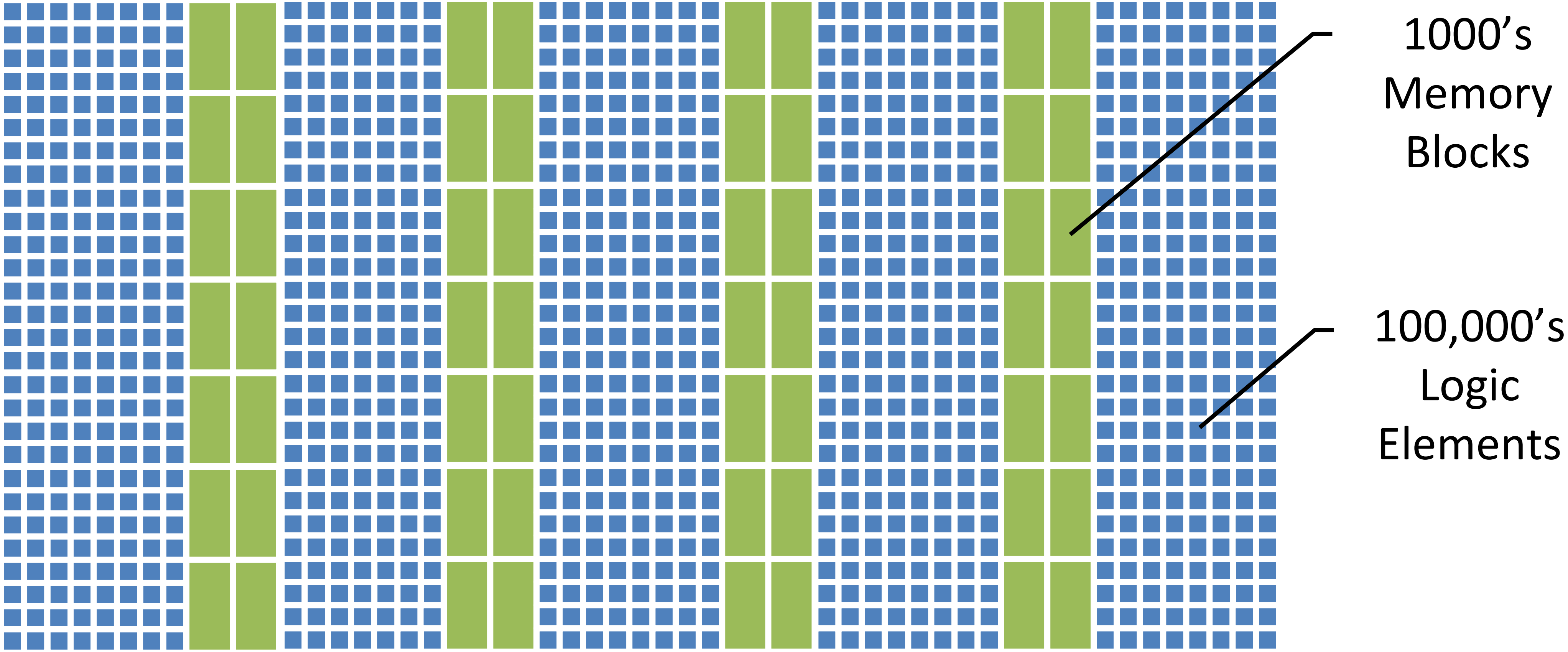
- find and shorten redundant patterns

Data encryption

- find and encrypt specific patterns

• • •

Motivation - FPGAs



No dedicated BCAM resources in FPGAs

Objectives

BCAMs

Massively parallel memory search

Require high memory bandwidth

FPGAs

Block RAMs are main storage

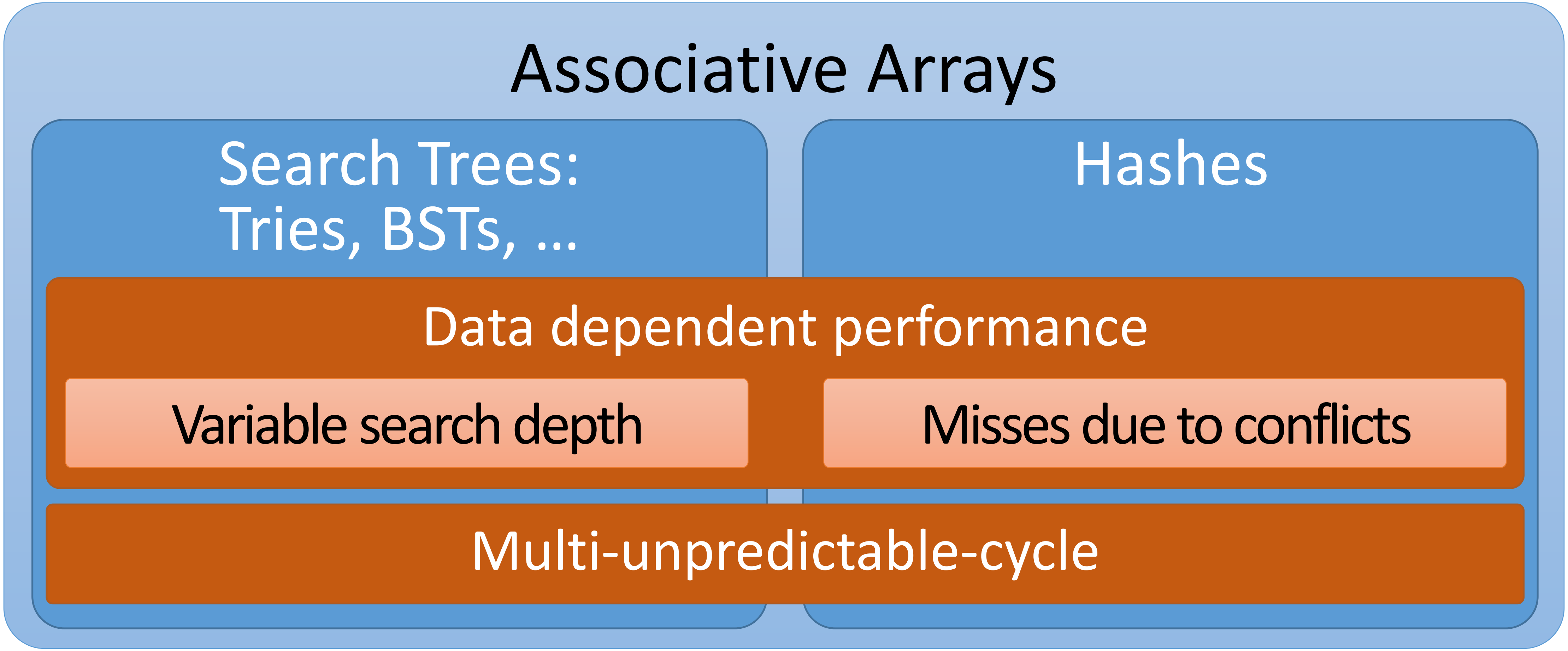
Limited memory bandwidth

Use BRAMs to construct

- Modular and flexible
- Storage efficient
- Single-cycle
- Performance oriented

BCAMs

Algorithmic Heuristics



Associative Arrays

Search Trees:
Tries, BSTs, …

Hashes

Data dependent performance

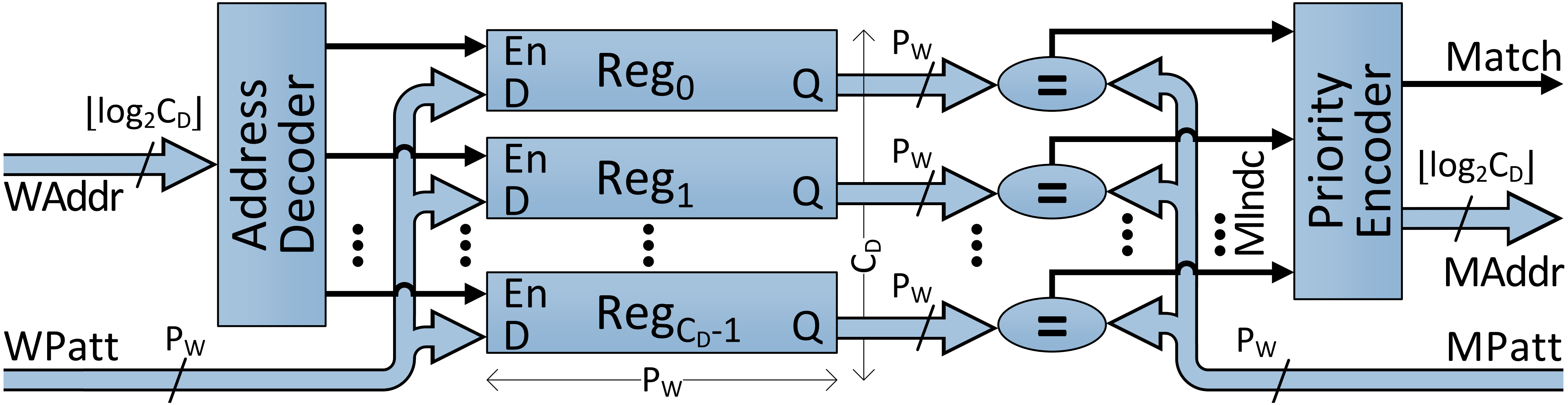
Variable search depth

Misses due to conflicts

Multi-unpredictable-cycle

Register-Based BCAM

Concurrent register read and compare



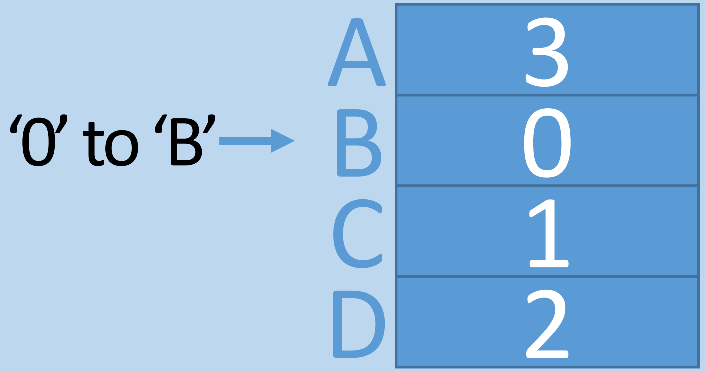
Single-cycle | Limited resources | Complex routing | Fits small BCAMs

Brute-Force Transposed-Indicators-RAM (1)
A Traditional BRAM-based BCAM

Key idea: Transposed RAM - data becomes addresses

Write

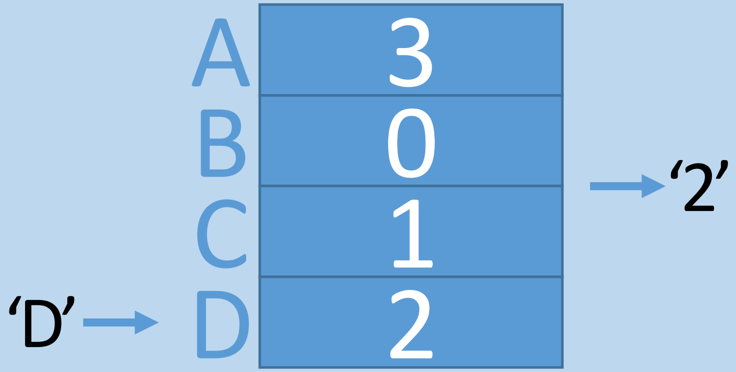
Write ‘0’ to location ‘B’



A	3
B	0
C	1
D	2

Match

Read location ‘D’ for match



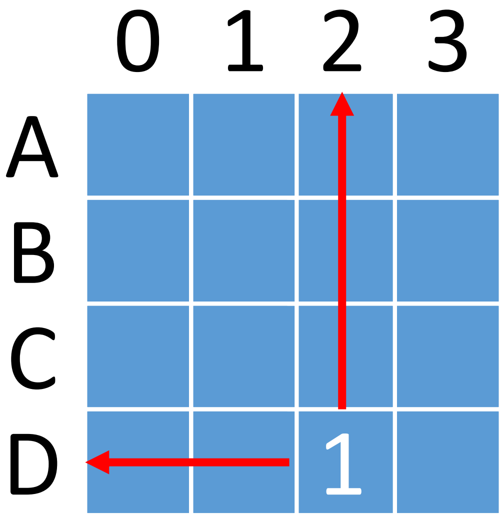
A	3
B	0
C	1
D	2

* Xilinx App Notes

Brute-Force Transposed-Indicators-RAM (2)
Storing Data to Multiple Addresses

- How can we store data to multiple addresses?
  - Specify addresses using one-hot coding
  - Each bit indicates a match or “store at location”

➢ PROBLEM: Depth of CAM is limited by data width of RAM
  - *e.g.* to build 1M deep CAM, we need 1M bits wide
  - In FPGAs: 1000 BRAMs x 32bit wide = 32K deep CAM



BRAM-based

Single-cycle

Depth of CAM is limited by RAM width

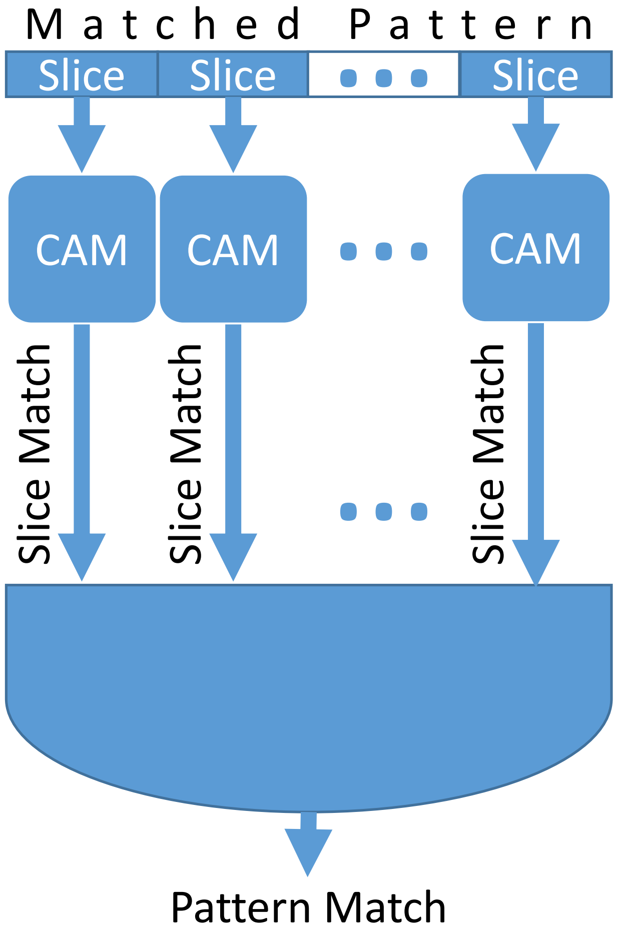
BCAM Cascading

- PROBLEM:
  - Patterns are encoded as RAM addresses
  - ➢RAM depth is exponential to pattern width

$$\text{RAM Depth} = 2^{\text{Pattern Width}}$$

- Solution: Cascading
  1. Divide pattern into smaller slices
  2. Search for each slice separately
  3. If all slices are found → pattern match!
  - ➢RAM depth is linear to pattern width

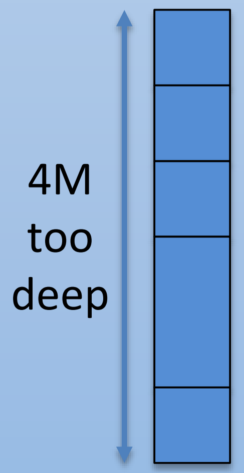
$$\text{RAM Depth} = 2^{\text{Slice Width}} \text{ x } (\text{Pattern Width} / \text{Slice Width})$$



Hierarchical Search 2D BCAM (1)
Narrow and Deep BCAM

Key idea: Hierarchical search

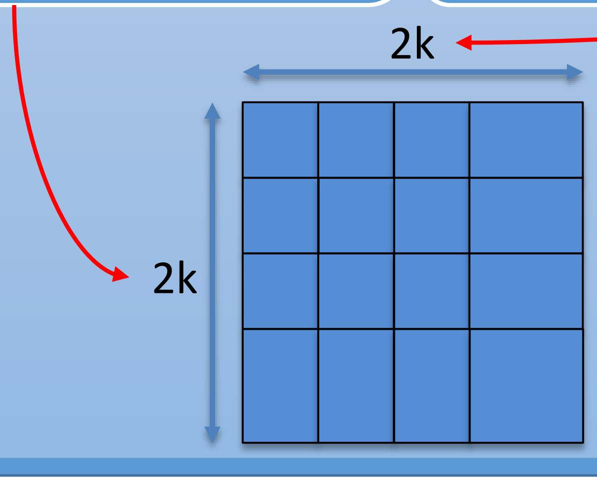
1D BCAM



2D BCAM

① Find a set (row) with match using a 1D BCAM

② Search this set (row) in parallel for a specific match



Hierarchical Search 2D BCAM (2)
Example

addresses	patterns
0	2
1	3
2	1
3	1

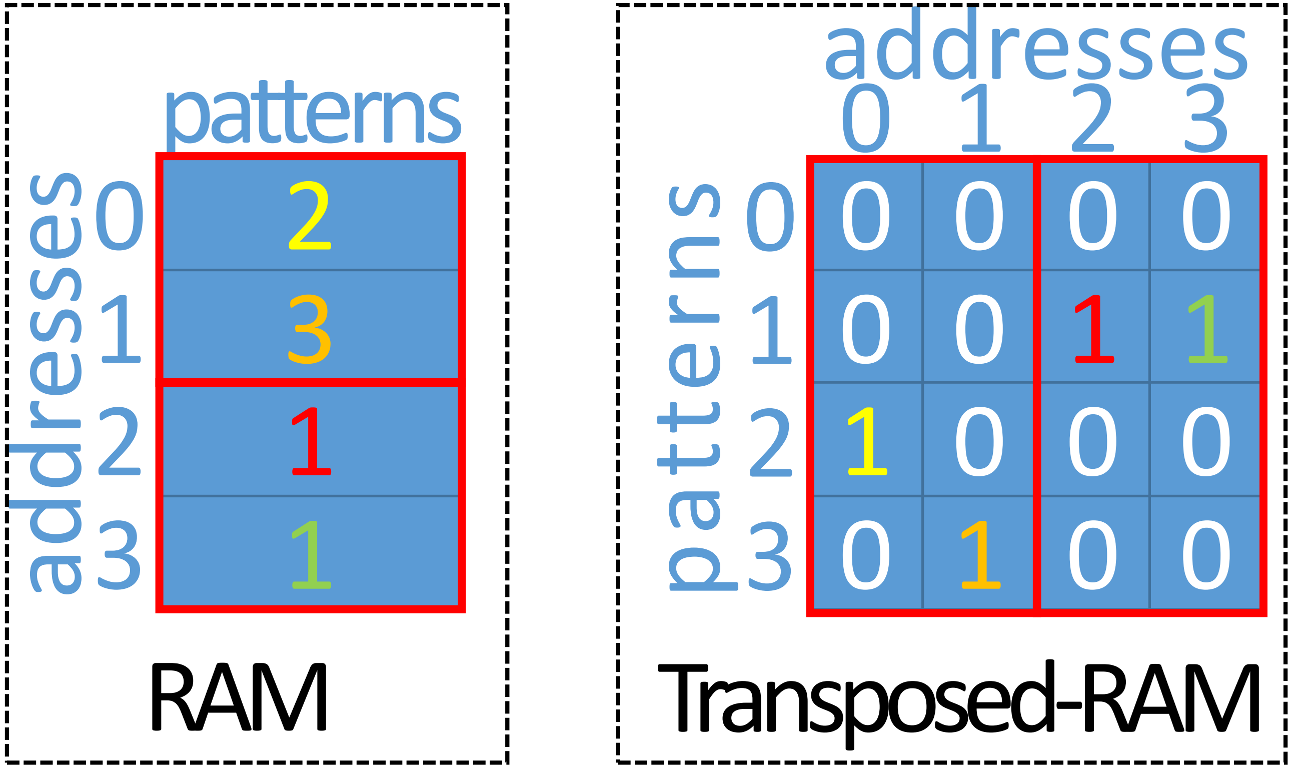
RAM

patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0	1	1
2	1	0	0	0
3	0	1	0	0

Transposed-RAM

Hierarchical Search 2D BCAM (2) Example

- Divide address space into sets



addresses	patterns
0	2
1	3
2	1
3	1

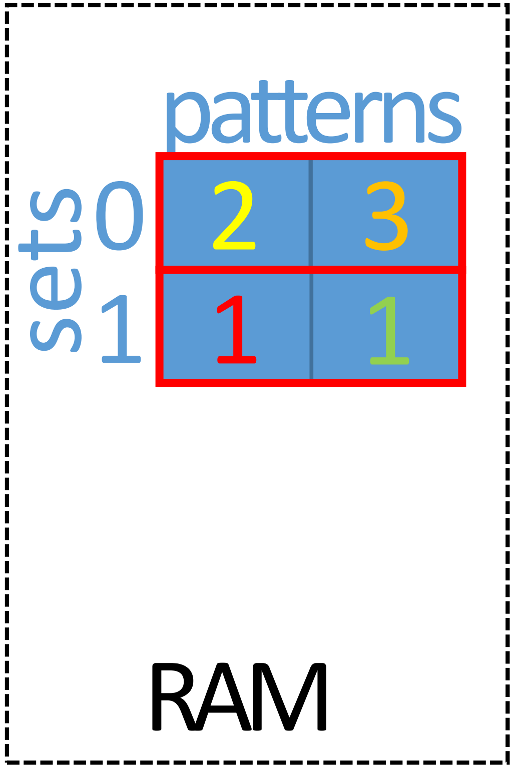
RAM

patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0	1	1
2	1	0	0	0
3	0	1	0	0

Transposed-RAM

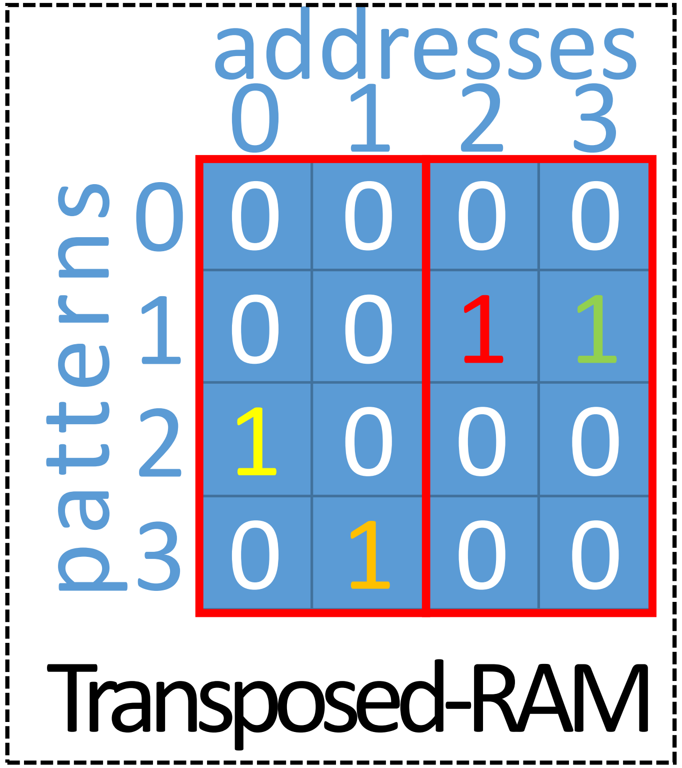
Hierarchical Search 2D BCAM (2) Example

- Divide address space into sets
  - RAM: each set in a line



sets	patterns	
0	2	3
1	1	1

RAM

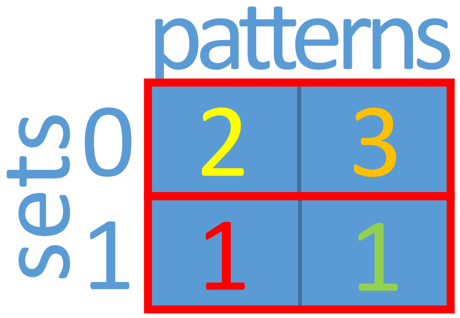


patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0	1	1
2	1	0	0	0
3	0	1	0	0

Transposed-RAM

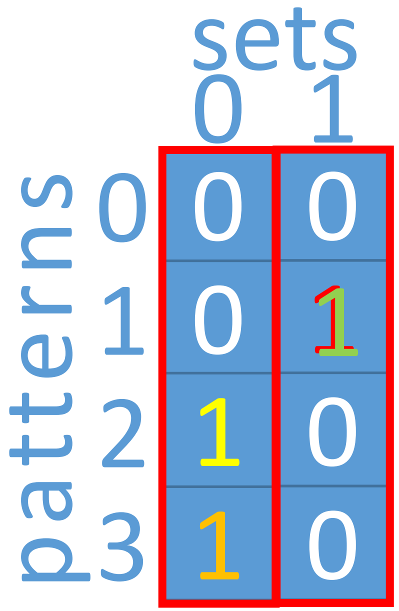
Hierarchical Search 2D BCAM (2) Example

- Divide address space into sets
  - RAM: each set in a line
  - Transposed-RAM: indicates "pattern in set?"



sets \ patterns		
0	2	3
1	1	1

RAM

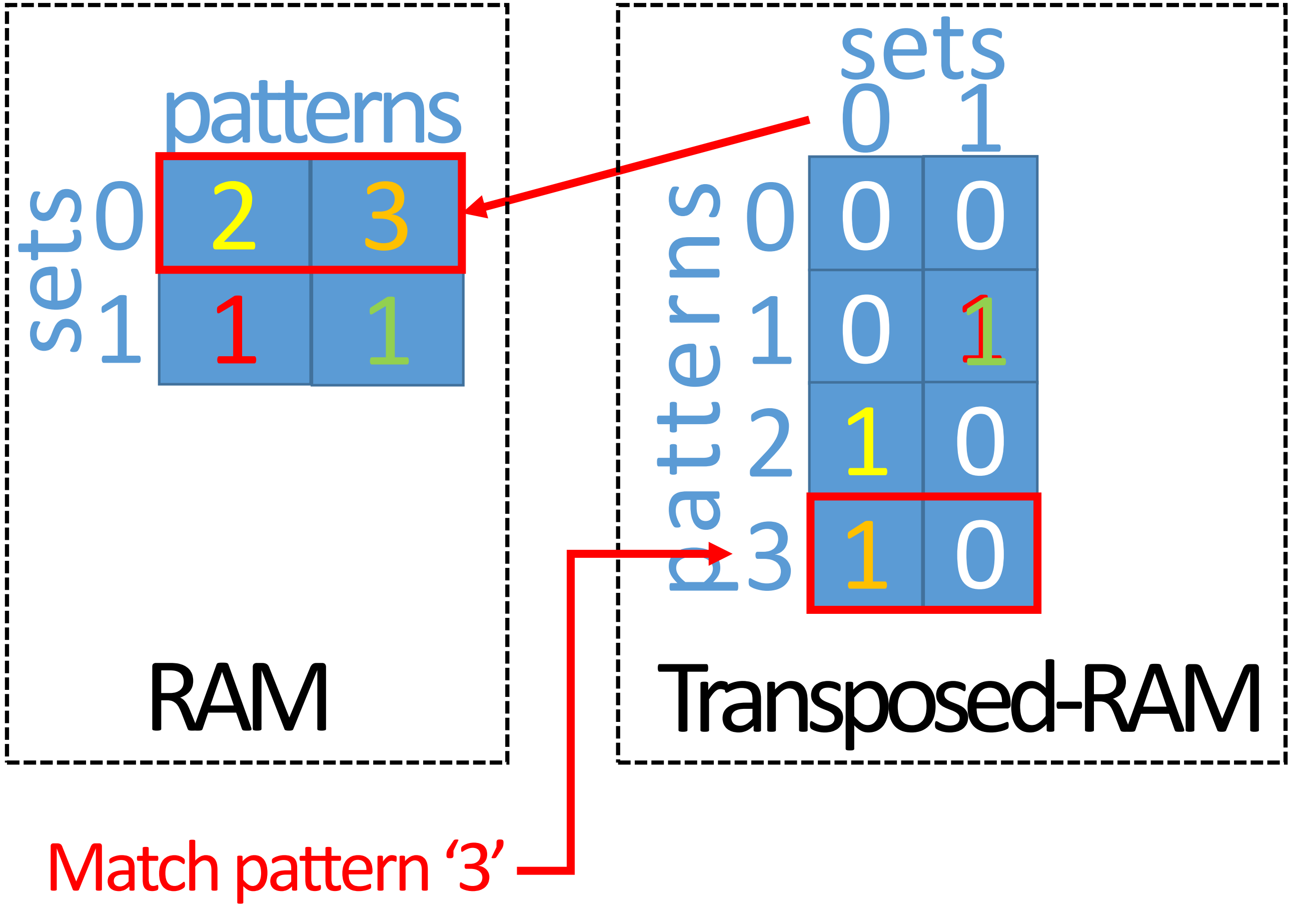


patterns \ sets	0	1
0	0	0
1	0	1
2	1	0
3	1	0

Transposed-RAM

Hierarchical Search 2D BCAM (2) Example

- Divide address space into sets
  - RAM: each set in a line
  - Transposed-RAM: indicates "pattern in set?"
- Hierarchical Search:
  1. Find a set (row) with match using a 1D BCAM



RAM

sets		
0	2	3
1	1	1

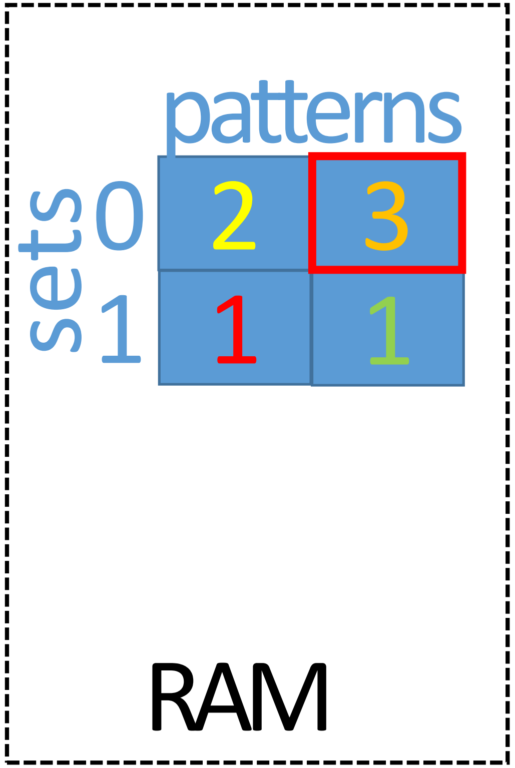
Transposed-RAM

patterns \ sets	0	1
0	0	0
1	0	1
2	1	0
3	1	0

Match pattern '3'

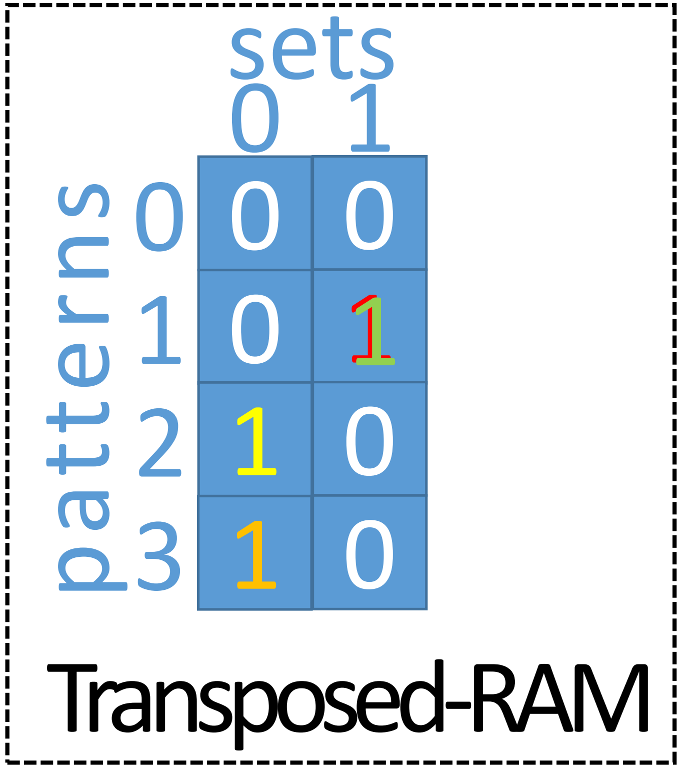
Hierarchical Search 2D BCAM (2) Example

- Divide address space into sets
  - RAM: each set in a line
  - Transposed-RAM: indicates "pattern in set?"
- Hierarchical Search:
  1. Find a set (row) with match using a 1D BCAM
  2. Search this set (row) in parallel for a specific match



sets \ patterns		
0	2	3
1	1	1

RAM

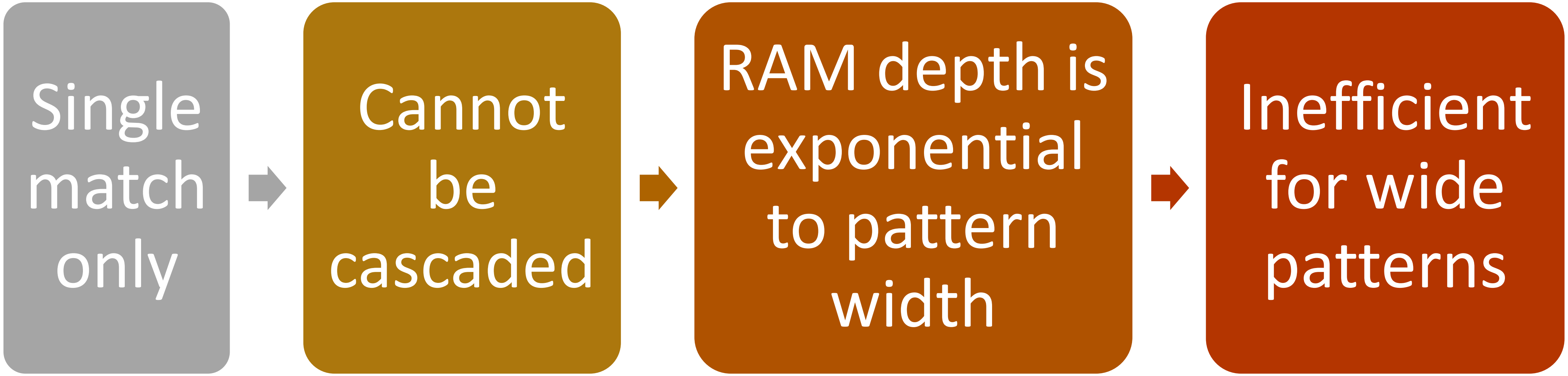


patterns \ sets	0	1
0	0	0
1	0	1
2	1	0
3	1	0

Transposed-RAM

Hierarchical Search 2D BCAM (3)
Pros and Cons

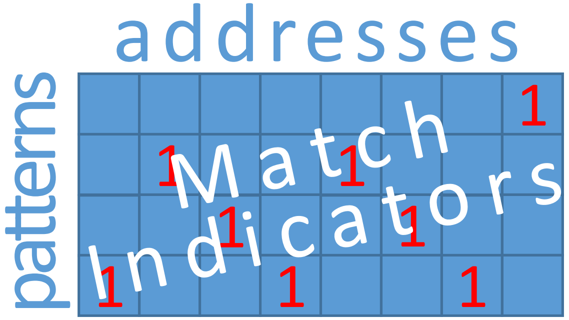
- BRAM-Based
- Single-cycle
- Efficient for deep CAMs



Single match only → Cannot be cascaded → RAM depth is exponential to pattern width → Inefficient for wide patterns

Indirectly-Indexed 2D (II2D) BCAM (1)
Cascadable Wide and Deep BCAM

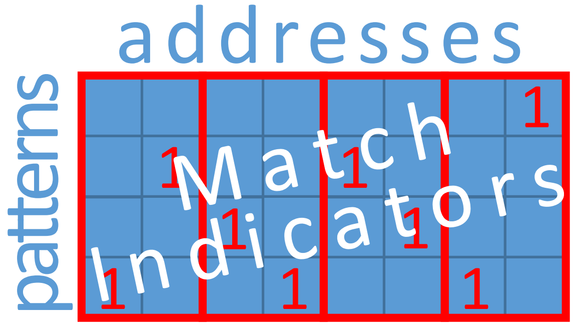
PROBLEM: is it possible to regenerate matches for all addresses?



Indirectly-Indexed 2D (II2D) BCAM (1)
Cascadable Wide and Deep BCAM

PROBLEM: is it possible to regenerate matches for all addresses?

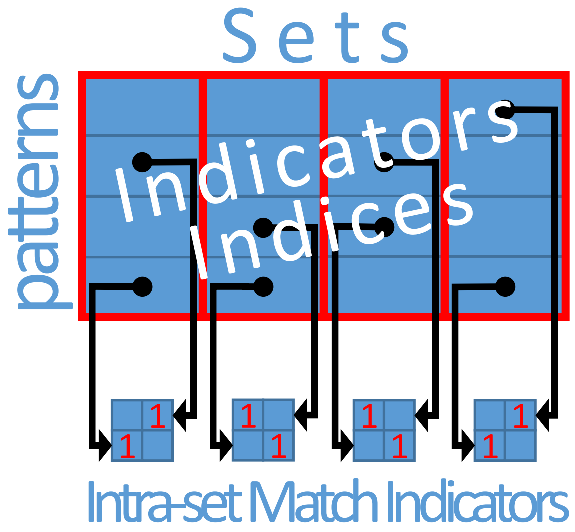
Key observation	
Transposed RAM is a sparse matrix	*n* columns (set of addresses) accommodates *n* matches (1's) at most!



Indirectly-Indexed 2D (II2D) BCAM (1)
Cascadable Wide and Deep BCAM

PROBLEM: is it possible to regenerate matches for all addresses?

Key observation	
Transposed RAM is a sparse matrix	*n* columns (set of addresses) accommodates *n* matches (1's) at most!



Key idea: use indirect indices to point to intra-set matches

Cascadable

Scalable (linear growth)

Supports wider patterns

Indirectly-Indexed 2D (II2D) BCAM (1)
Cascadable Wide and Deep BCAM

PROBLEM: is it possible to regenerate matches for all addresses?

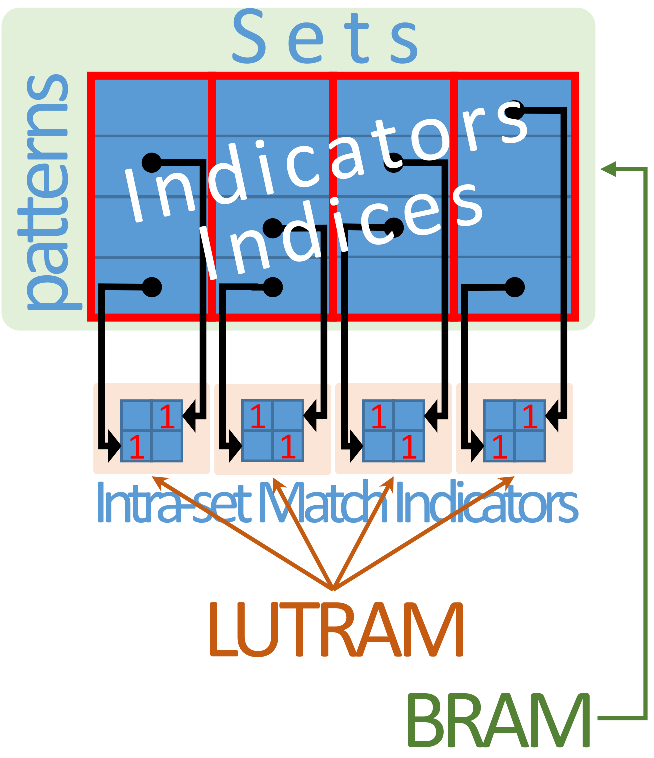
Key observation	
Transposed RAM is a sparse matrix	*n* columns (set of addresses) accommodates *n* matches (1’s) at most!

Key idea: use indirect indices to point to intra-set matches

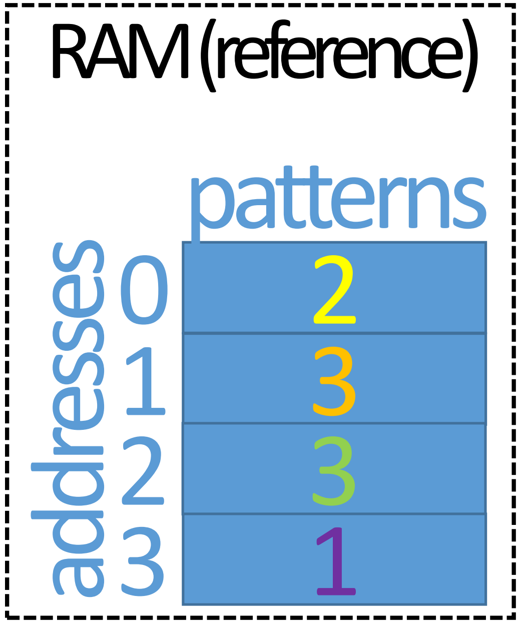
Cascadable

Scalable (linear growth)

Supports wider patterns

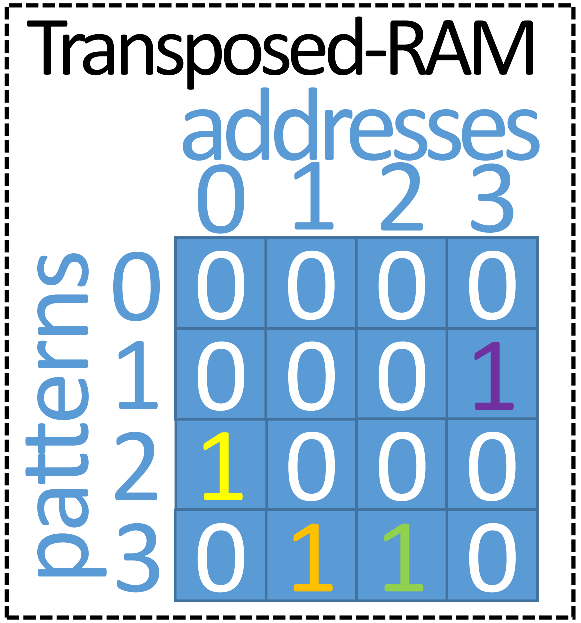


Indirectly-Indexed 2D (II2D) BCAM (2) Example



RAM (reference)

addresses	patterns
0	2
1	3
2	3
3	1

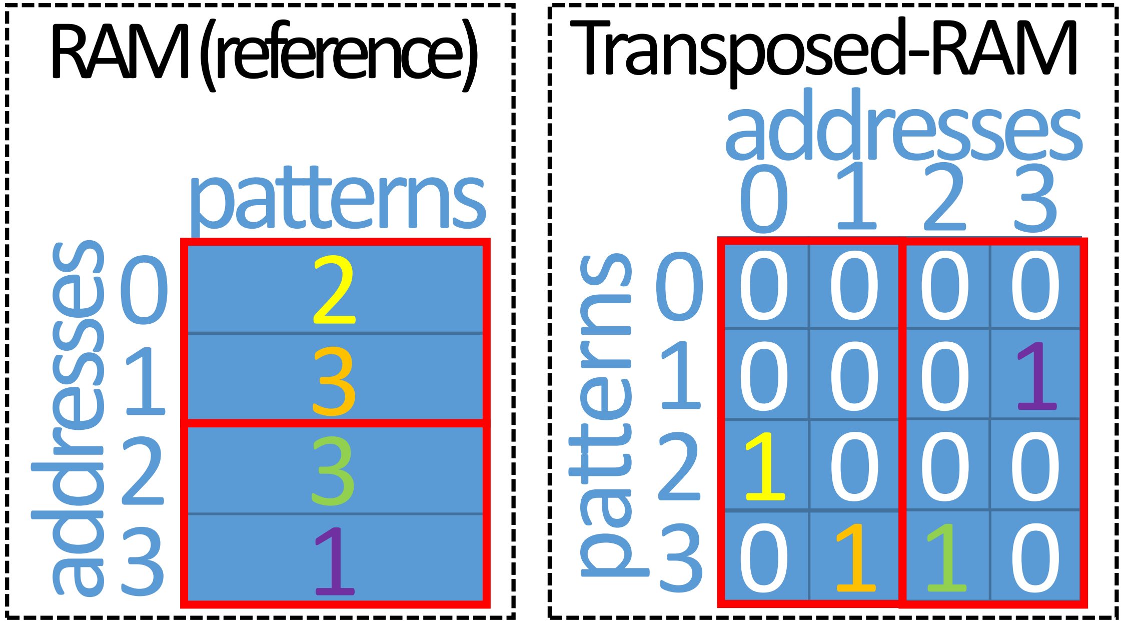


Transposed-RAM

patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	1	0	0	0
3	0	1	1	0

Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets



RAM (reference)

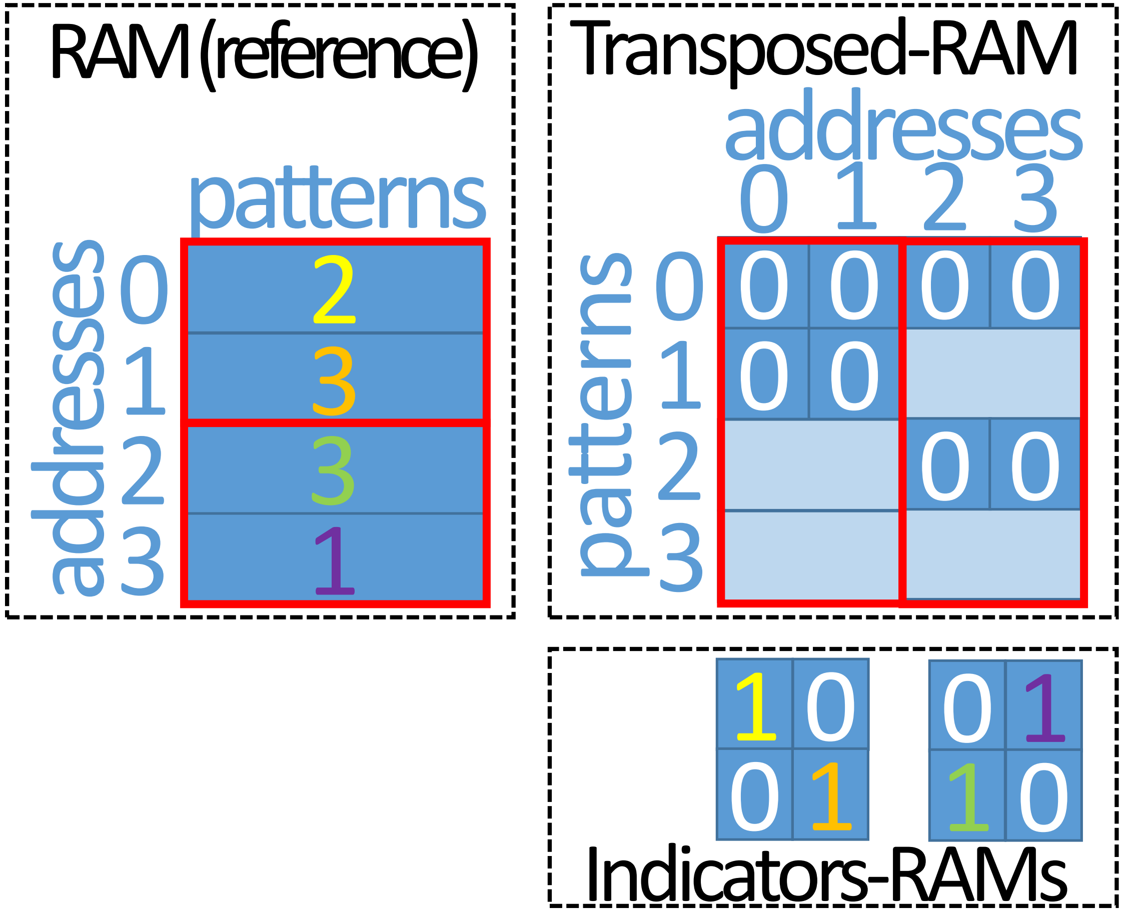
addresses	patterns
0	2
1	3
2	3
3	1

Transposed-RAM

patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	1	0	0	0
3	0	1	1	0

Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets
- Store sets with a match in Indicators-RAM



RAM (reference)

addresses	patterns
0	2
1	3
2	3
3	1

Transposed-RAM

patterns \ addresses	0	1	2	3
0	0	0	0	0
1	0	0		
2			0	0
3				

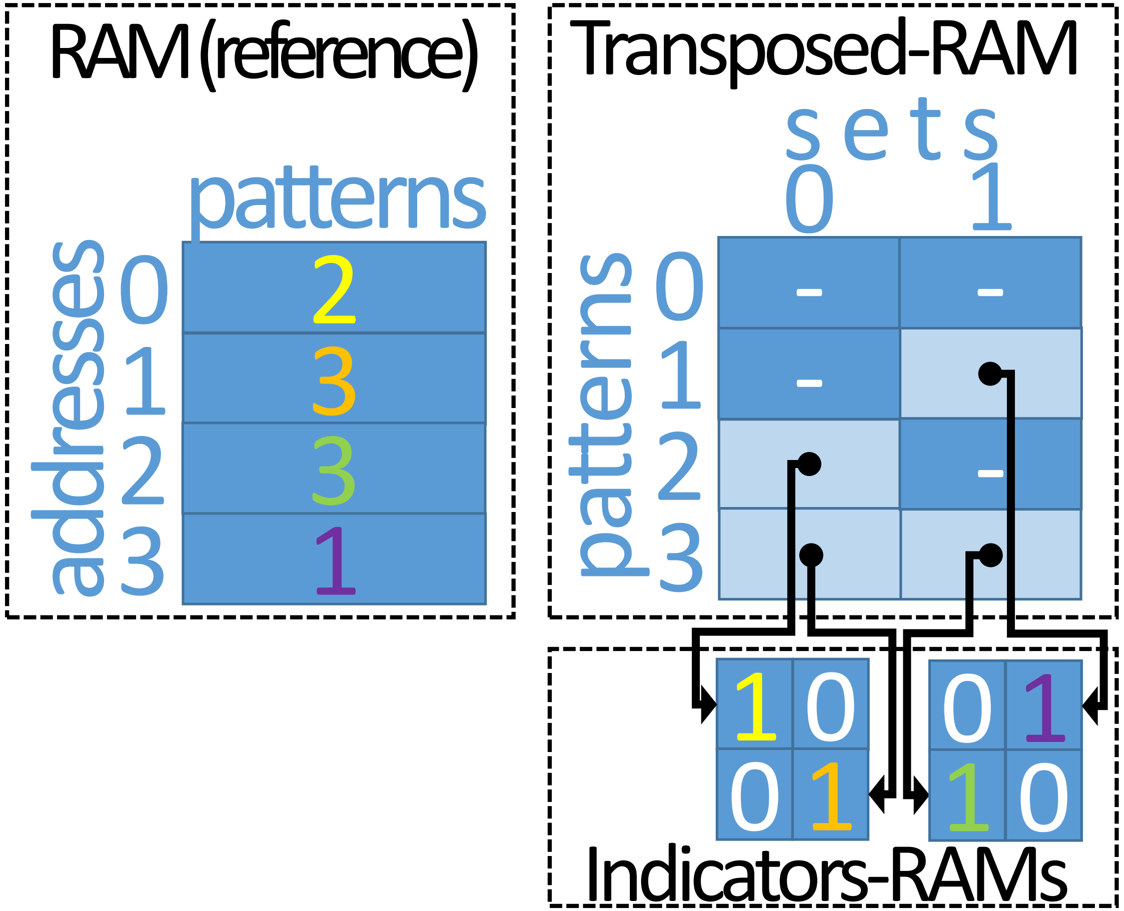
Indicators-RAMs

1	0
0	1

0	1
1	0

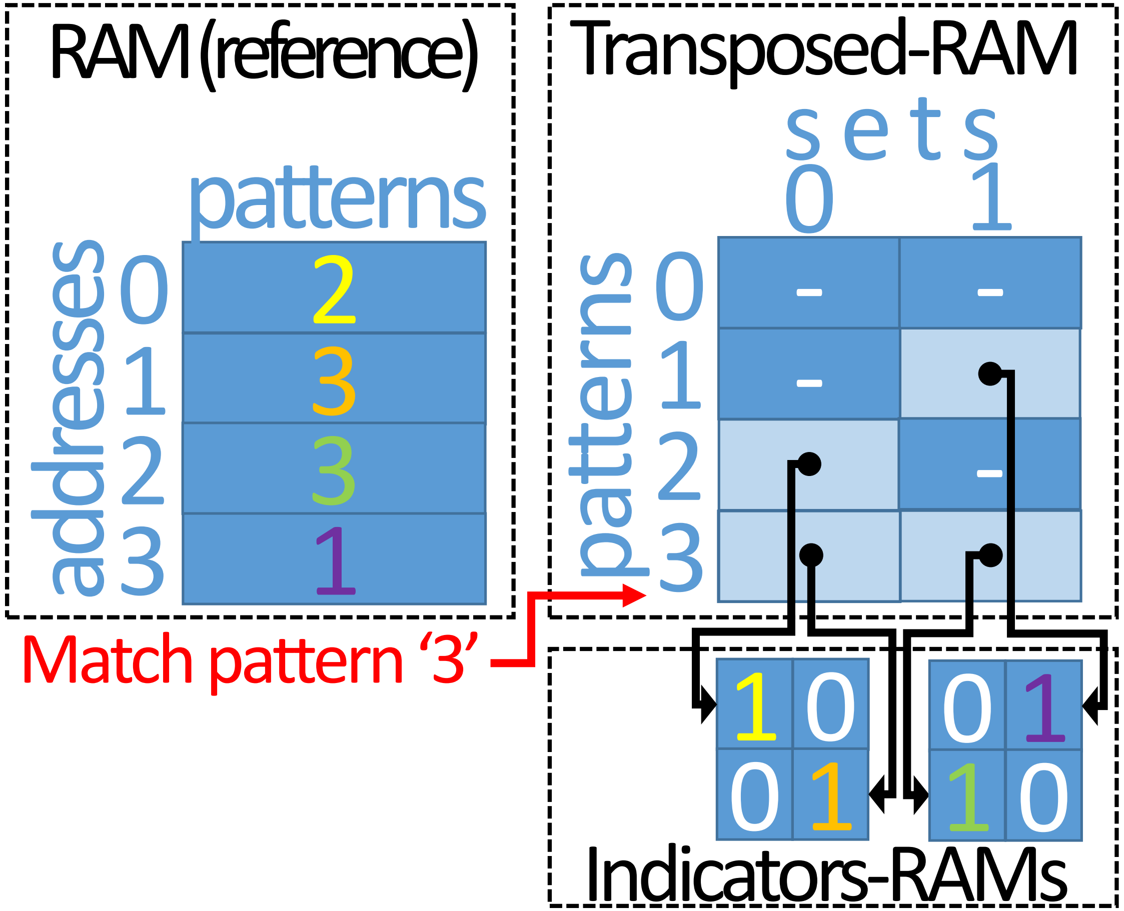
Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets
- Store sets with a match in Indicators-RAM
- Transposed-RAM stores indices to all matches in set



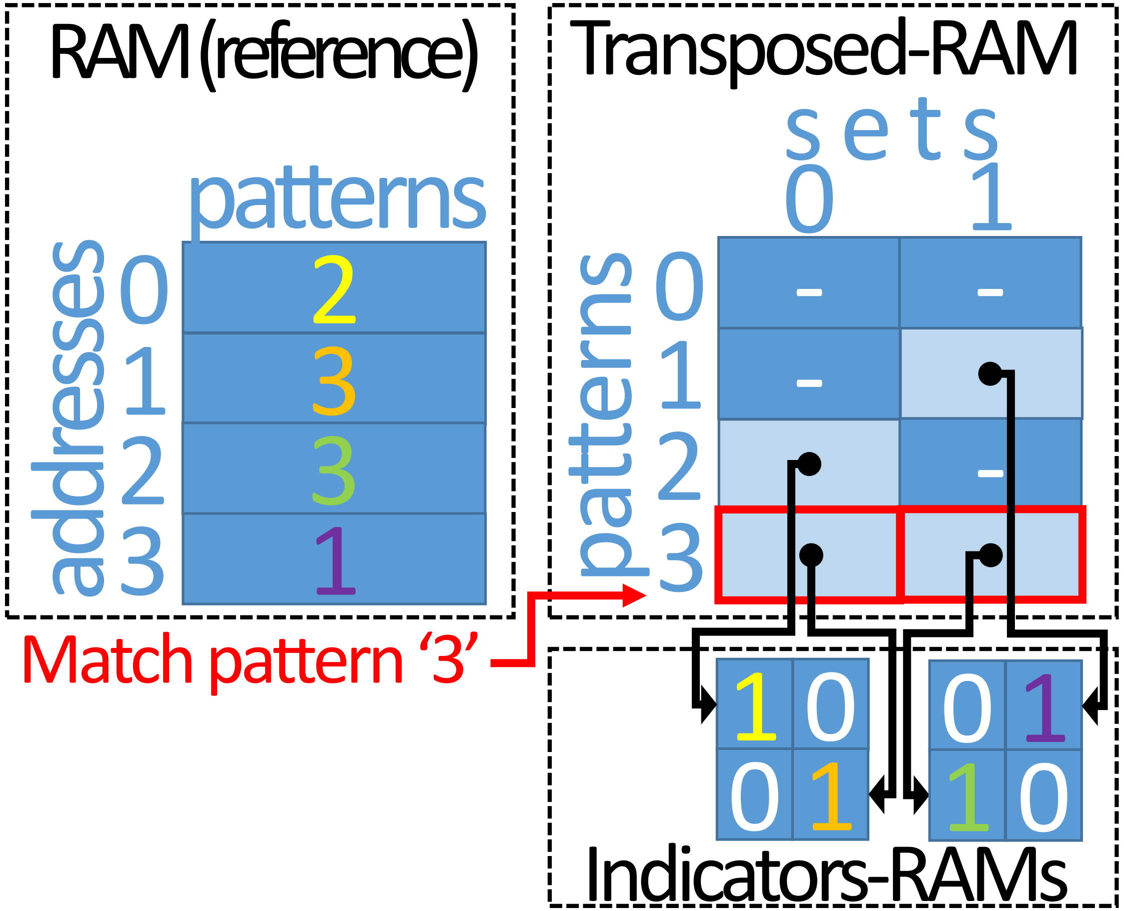
Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets
- Store sets with a match in Indicators-RAM
- Transposed-RAM stores indices to all matches in set
- Hierarchical Search:



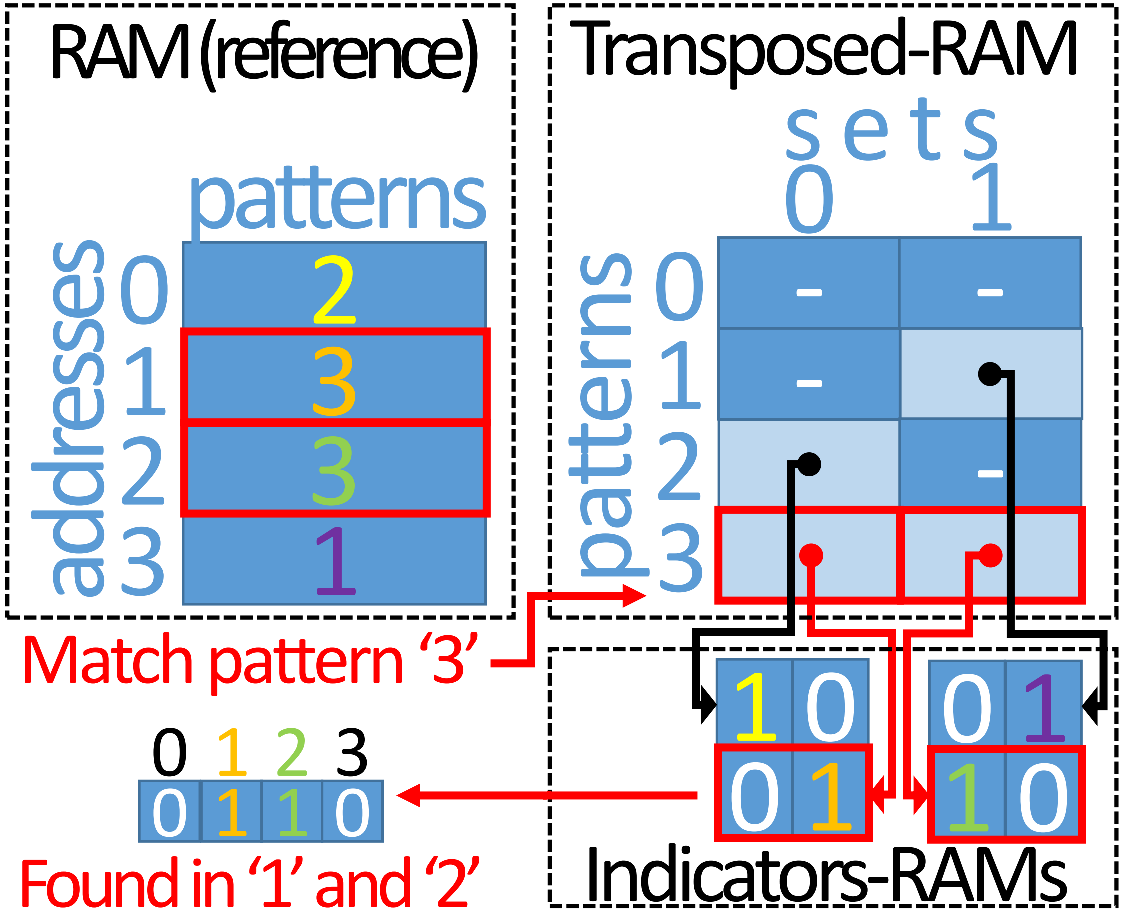
Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets
- Store sets with a match in Indicators-RAM
- Transposed-RAM stores indices to all matches in set
- Hierarchical Search:
  - Find indices of all matching sets in Transposed-RAM



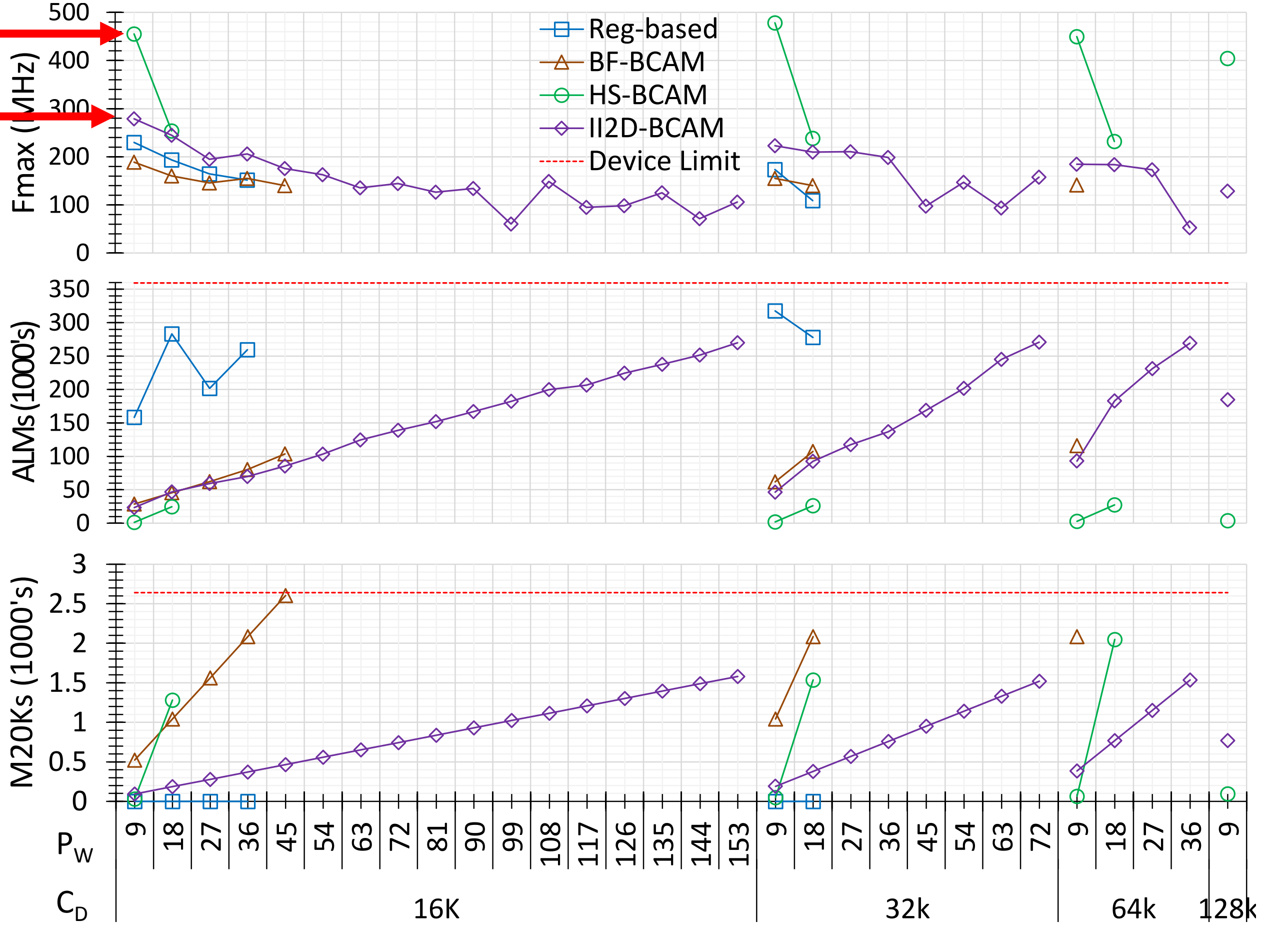
Indirectly-Indexed 2D (II2D) BCAM (2) Example

- Divide address space into sets
- Store sets with a match in Indicators-RAM
- Transposed-RAM stores indices to all matches in set
- Hierarchical Search:
  - Find indices of all matching sets in Transposed-RAM
  - Read Indicators-RAM using indices from Transposed-RAM



Indirectly-Indexed 2D (II2D) BCAM (3) Area and Performance

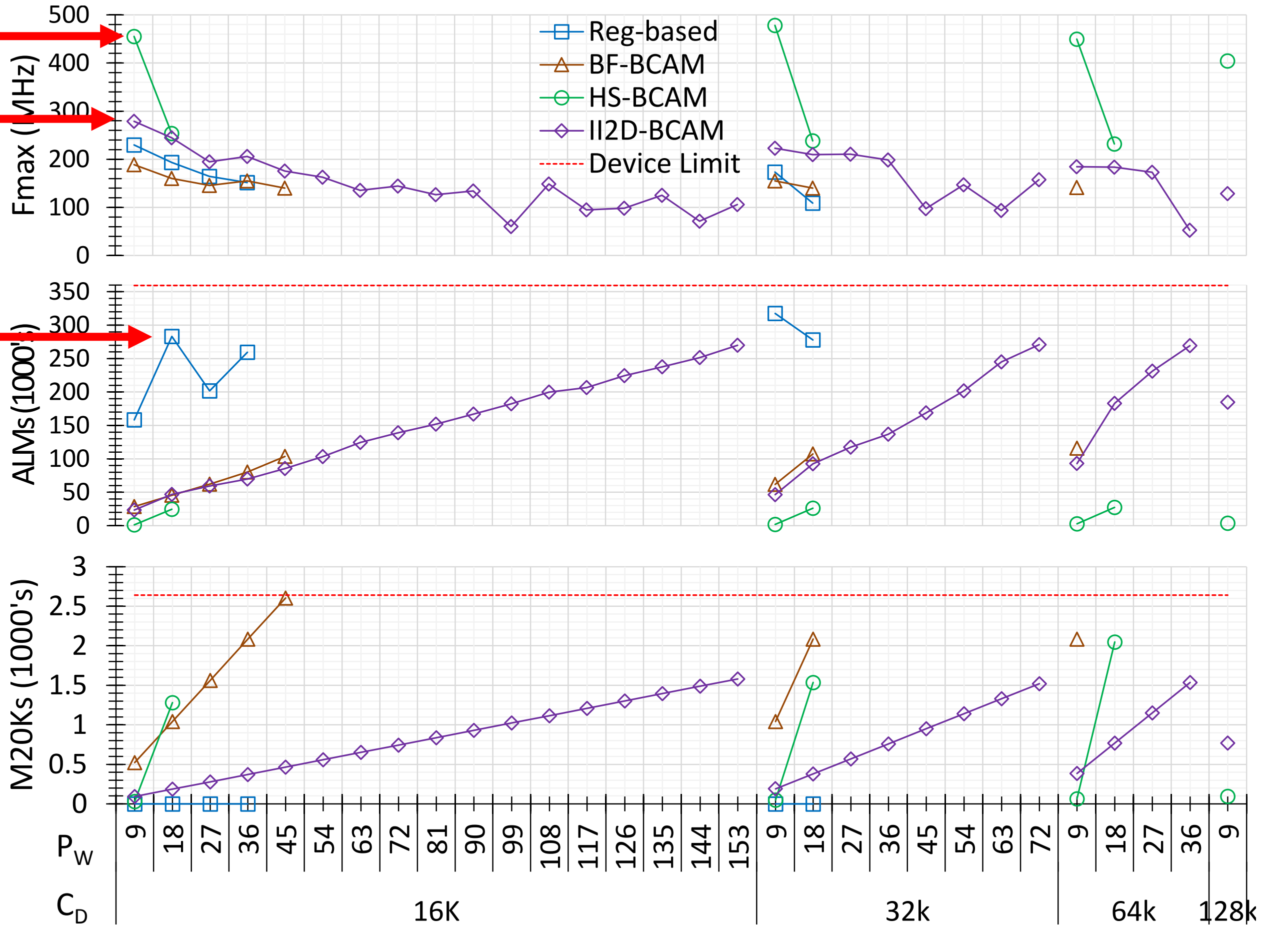
Except for very a narrow HS, II2D exhibits higher Fmax



Indirectly-Indexed 2D (II2D) BCAM (3)
Area and Performance

Except for very a narrow HS,
II2D exhibits higher Fmax

register-based BCAM
register consumption

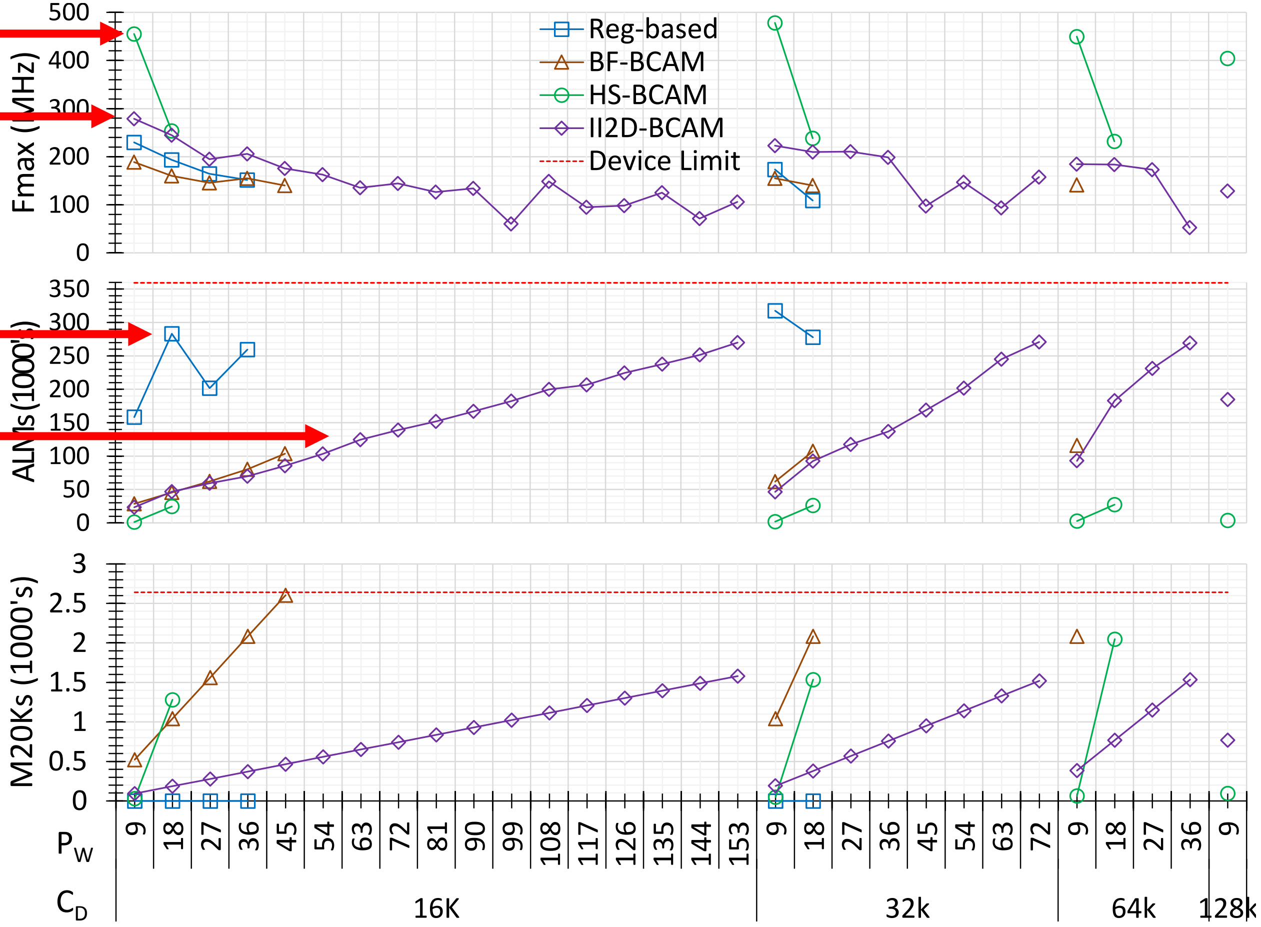


Indirectly-Indexed 2D (II2D) BCAM (3)
Area and Performance

Except for very a narrow HS, II2D exhibits higher Fmax

register-based BCAM register consumption

II2D linear ALM consumption; similar to other methods



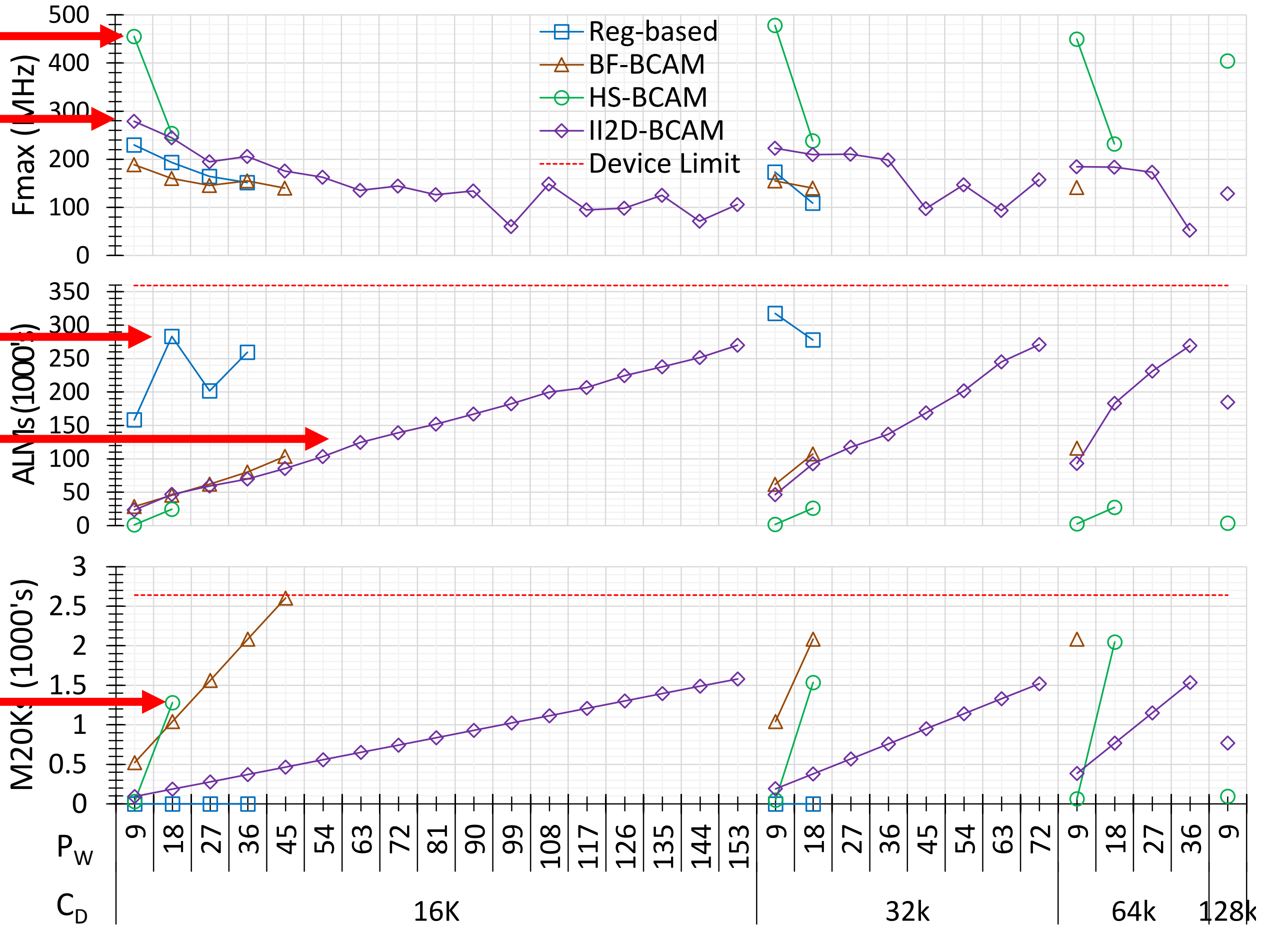
Indirectly-Indexed 2D (II2D) BCAM (3)
Area and Performance

Except for very a narrow HS,
II2D exhibits higher Fmax

register-based BCAM
register consumption

II2D linear ALM consumption;
similar to other methods

HS exponential BRAM consumption



Indirectly-Indexed 2D (II2D) BCAM (3)
Area and Performance

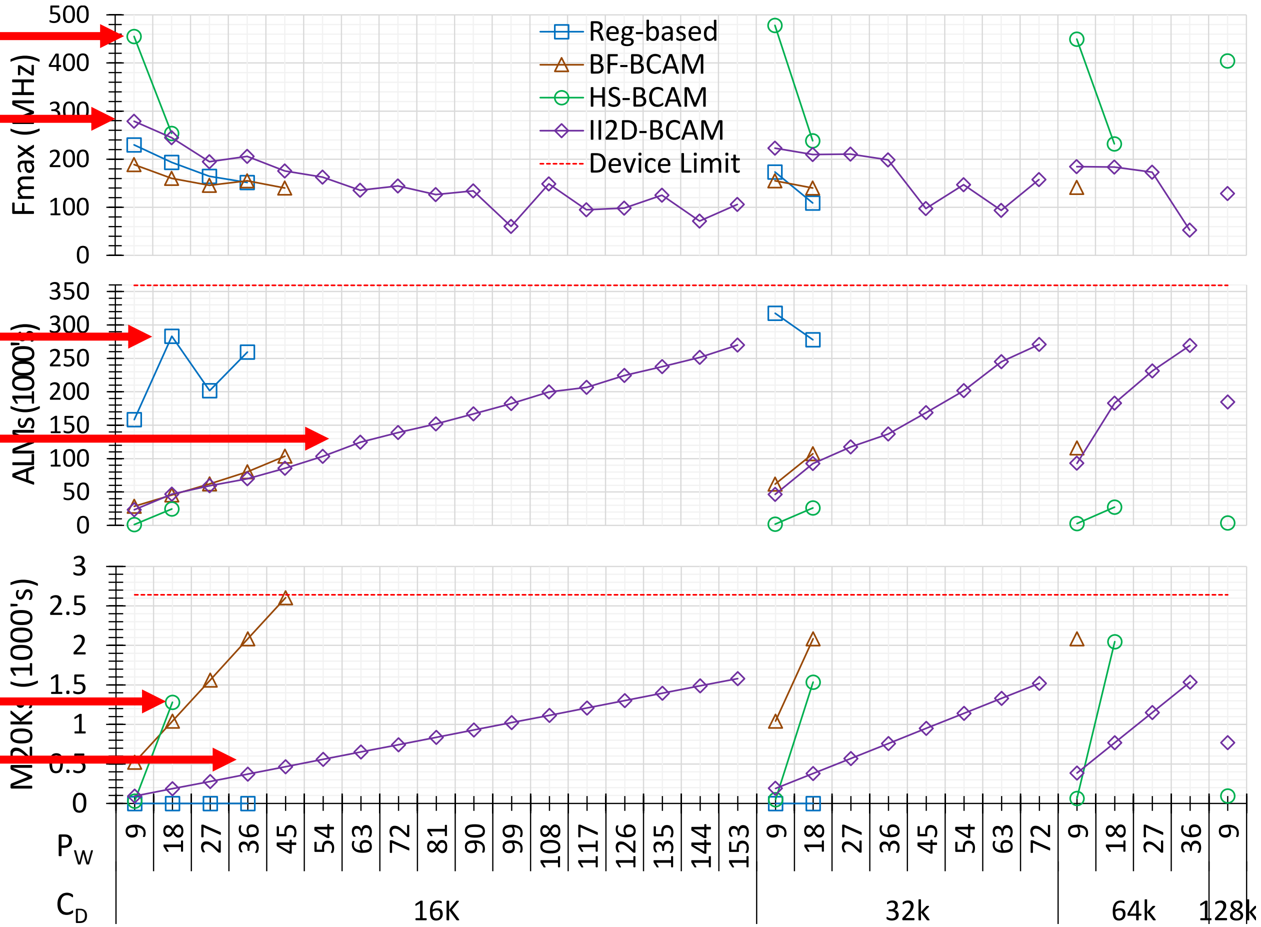
Except for very a narrow HS, II2D exhibits higher Fmax

register-based BCAM register consumption

II2D linear ALM consumption; similar to other methods

HS exponential BRAM consumption

II2D linear BRAM consumption



Indirectly-Indexed 2D (II2D) BCAM (3)
Area and Performance

Except for very a narrow HS, II2D exhibits higher Fmax

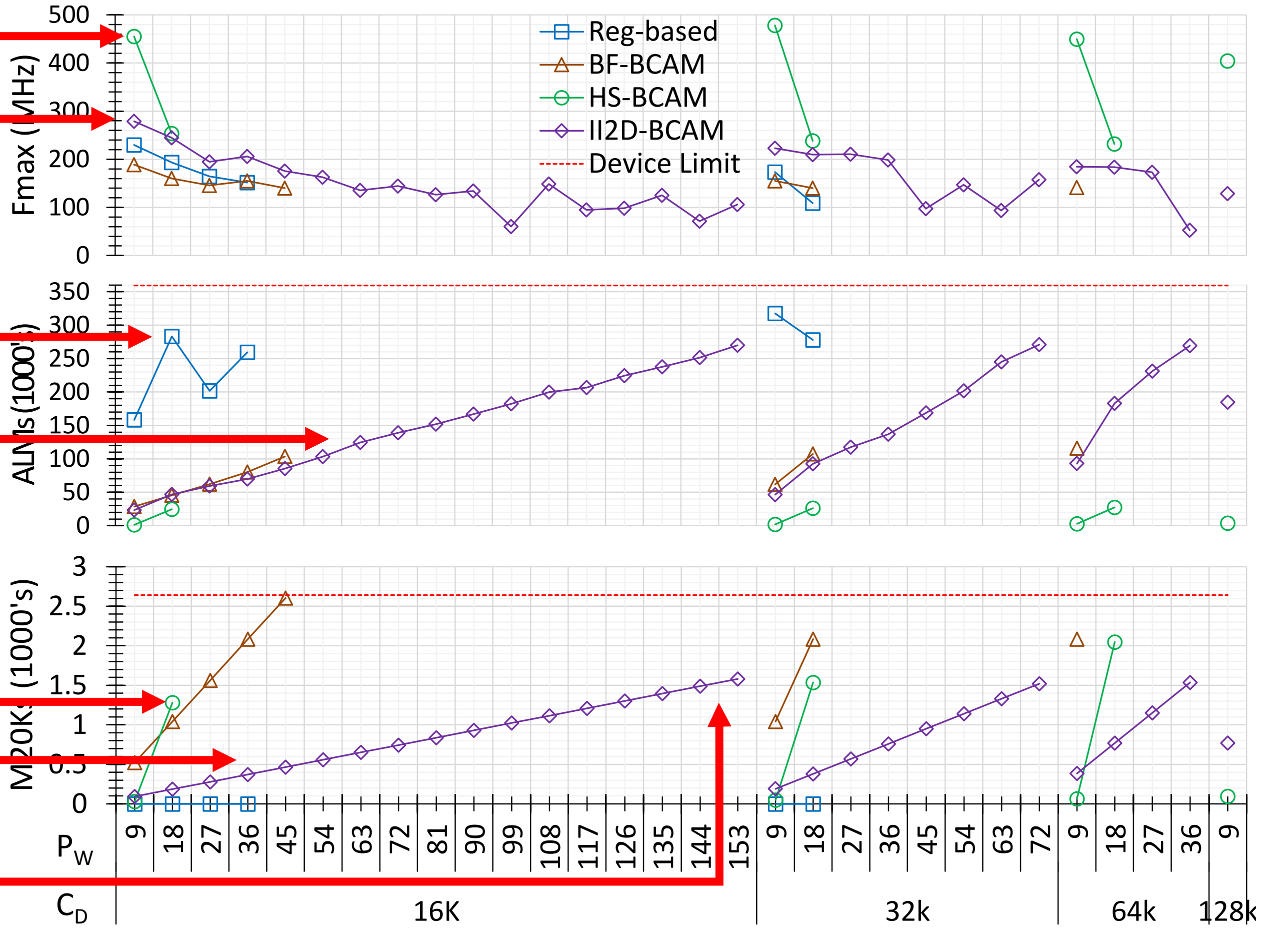
register-based BCAM register consumption

II2D linear ALM consumption; similar to other methods

HS exponential BRAM consumption

II2D linear BRAM consumption

II2D supports wider patterns



Open Source

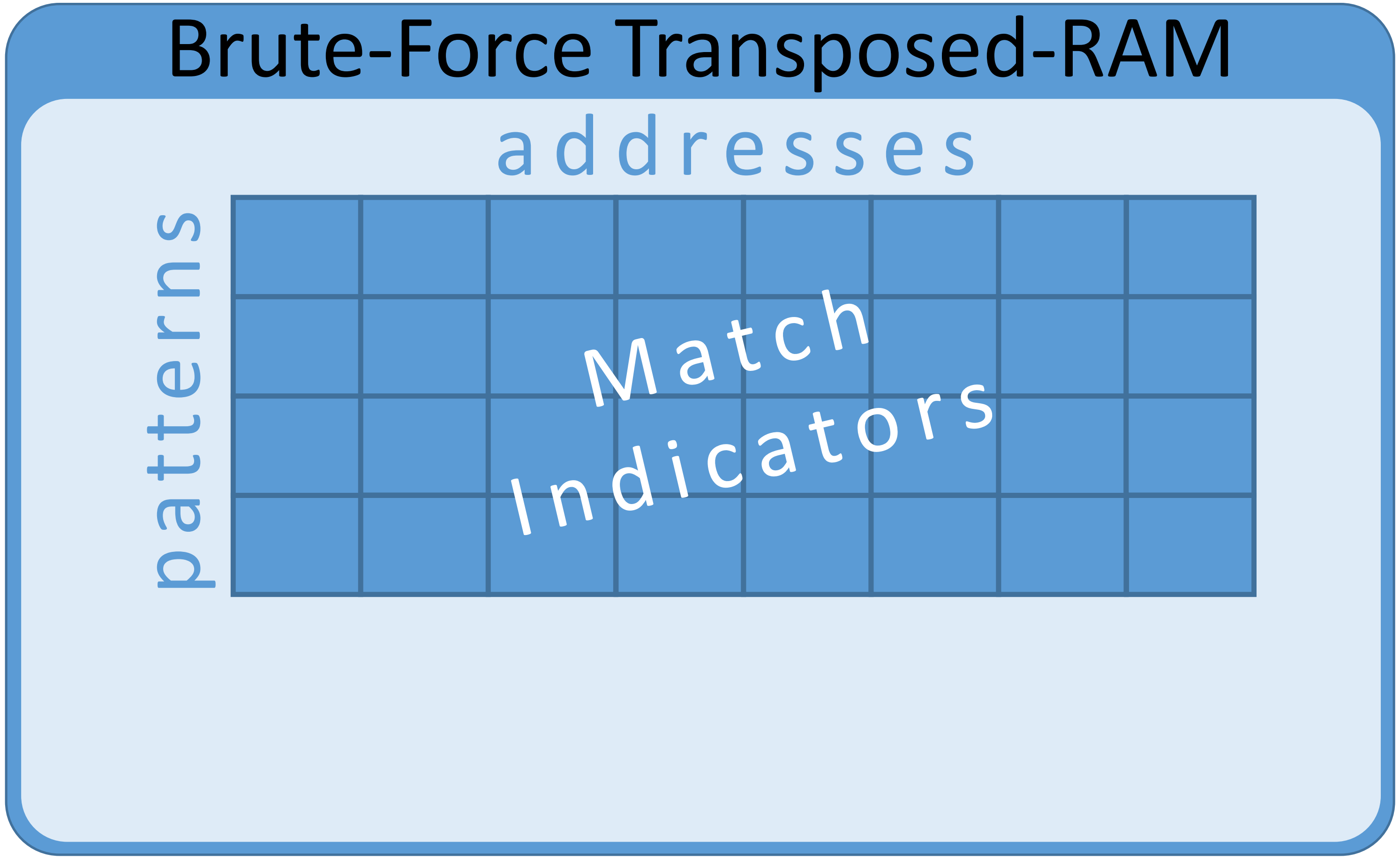
http://ece.ubc.ca/~lemieux/downloads/

Modular and parametric Verilog files

Run-in-batch simulation and synthesis manager

Conclusions

- BRAM-based ✓
- Single-cycle ✓
- Cascadable ✓
- Scalable ✓
- Deep ✗
- Wide ✓



Conclusions

BRAM-based ✓

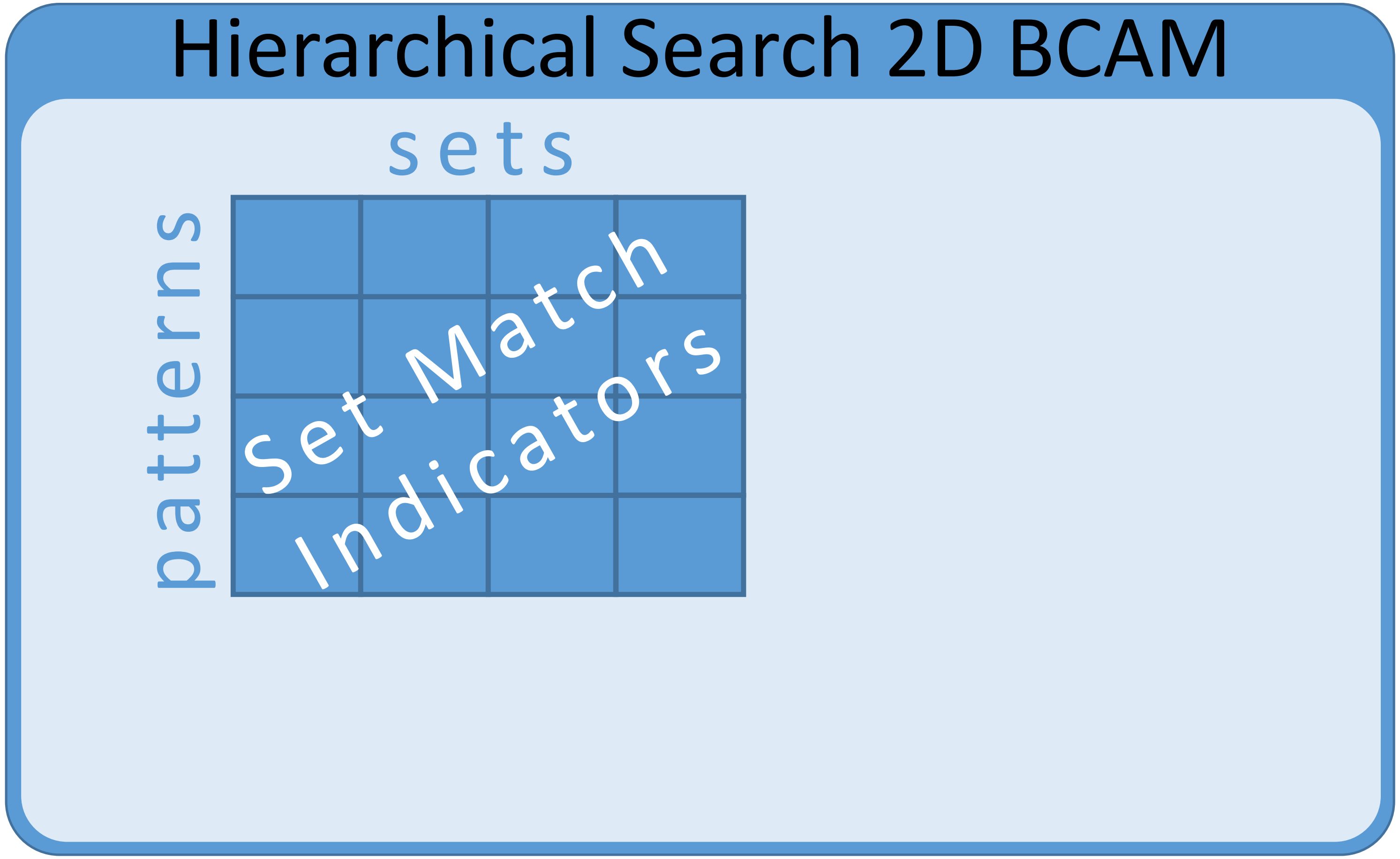
Single-cycle ✓

Cascadable ✗

Scalable ✗

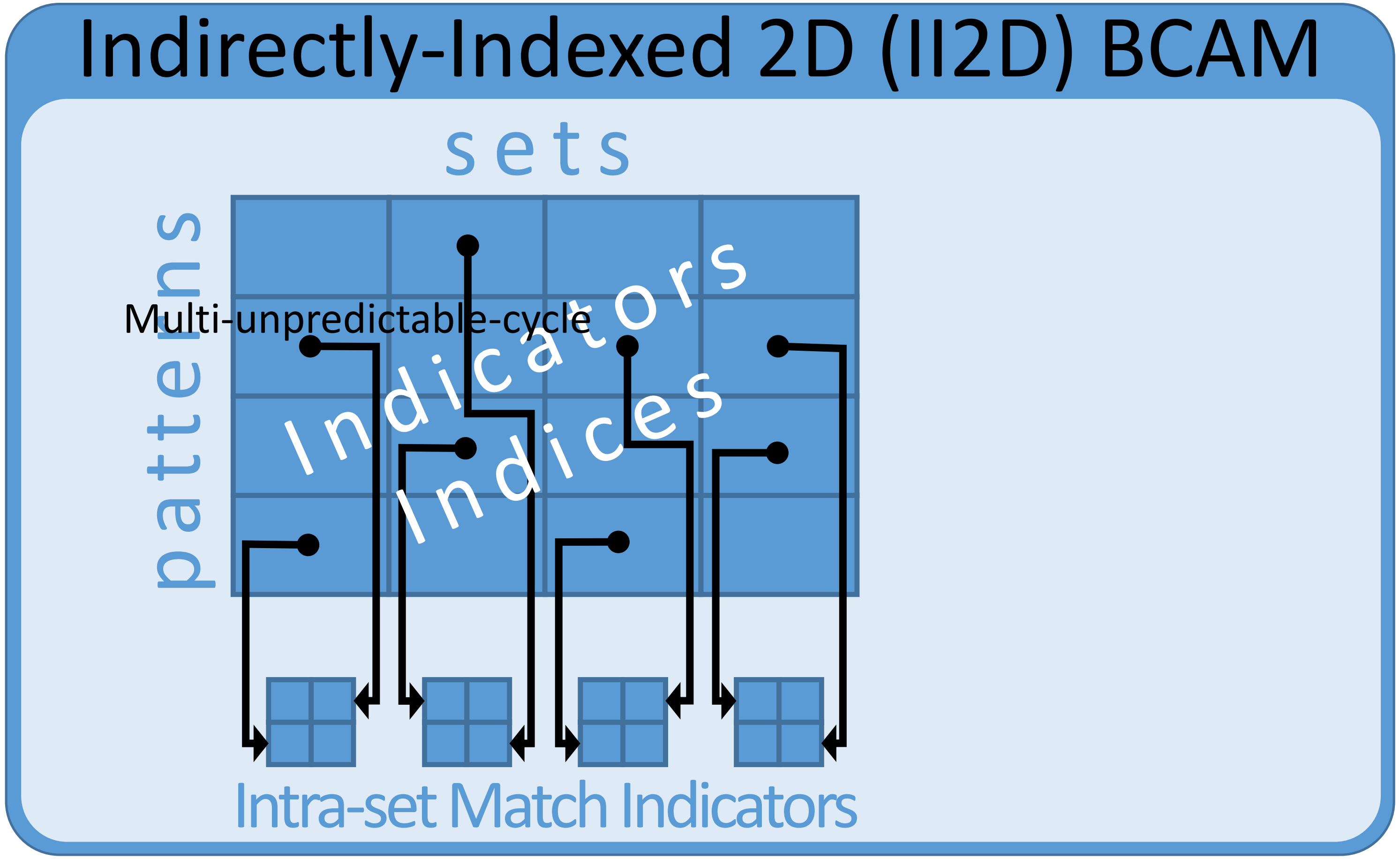
Deep ✓

Wide ✗


Hierarchical Search 2D BCAM

Conclusions

- BRAM-based ✓
- Single-cycle ✓
- Cascadable ✓
- Scalable ✓
- Deep ✓
- Wide ✓



Indirectly-Indexed 2D (II2D) BCAM

sets

patterns

Multi-unpredictable-cycle

Indicators

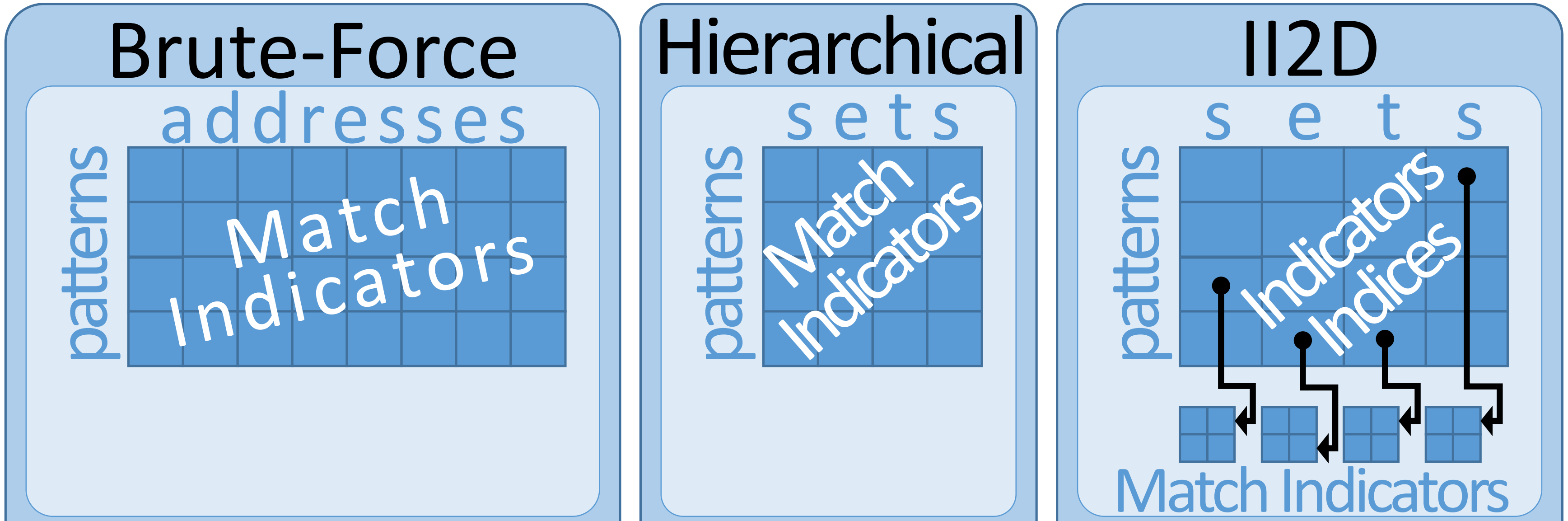
Indices

Intra-set Match Indicators

Thank You!

Backup Slides

Conclusions



	Brute-Force	Hierarchical	II2D
BRAM-based	✓	✓	✓
Single-cycle	✓	✓	✓
Deep	✗	✓	✓
Wide	✓	✗	✓