Modular Block-RAM-Based Longest-Prefix Match Ternary Content-Addressable Memories

Ameer M.S. Abdelhadi[*], Guy G.F. Lemieux[+], and Lesley Shannon[*]

SFU SIMON FRASER UNIVERSITY

* School of Engineering; Simon Fraser University; Canada

+ Dept. of ECE; The University of British Columbia; Canada

UBC THE UNIVERSITY OF BRITISH COLUMBIA

Reconfigurable Computing Lab

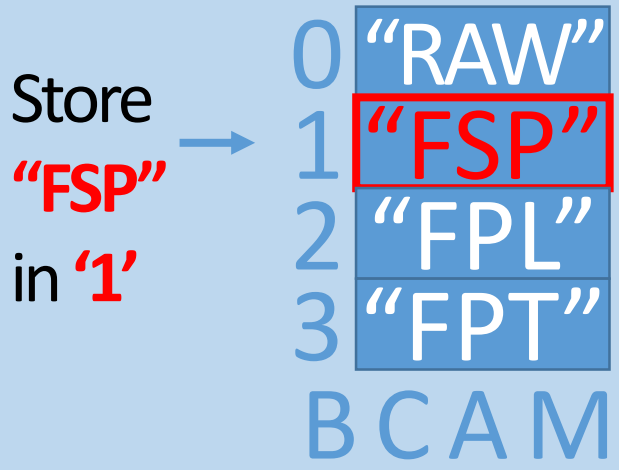
ece Electrical & Computer Engineering

Binary Content-Addressable Memories

Hardware-based Single-Cycle Parallel Search Engines

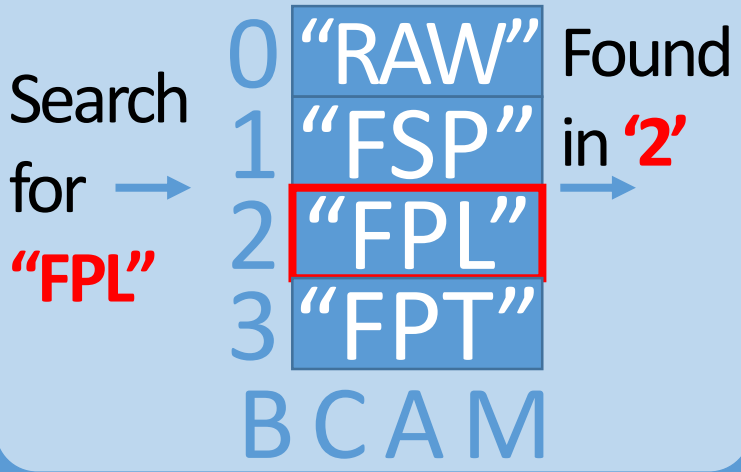
Write

Stores new data at specific address



Match

Search all addresses for a given data (pattern)

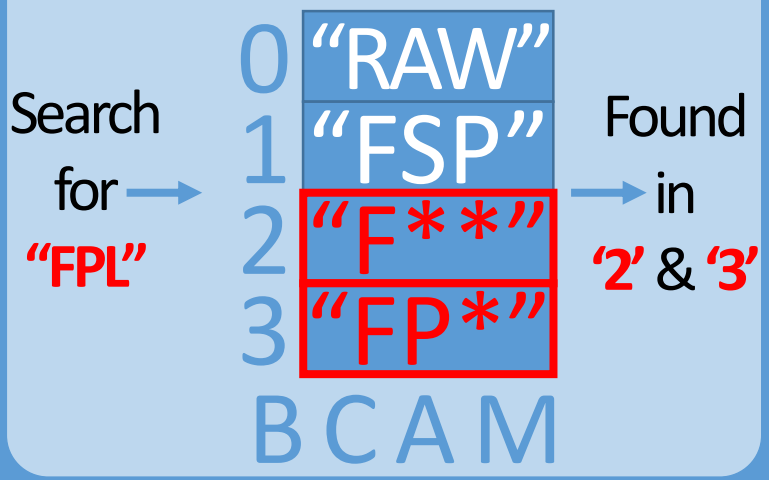


Ternary Content-Addressable Memories

Adds the ability to use wildcards (X's)

Match

Search all addresses for a given data (pattern)



Encoding

Priority Encoder (PE) (First occurrence)

The smallest address were "FPL" is found is '2'

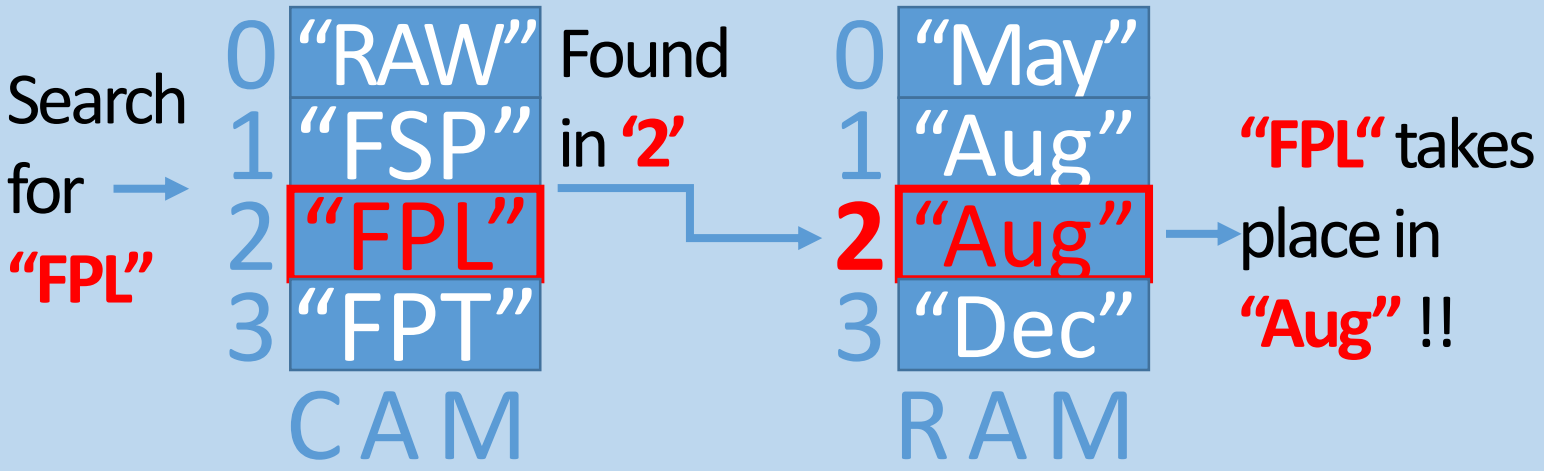
Longest-Prefix Match (LPM)

The longest matching prefix of "FPL" is found in '3'

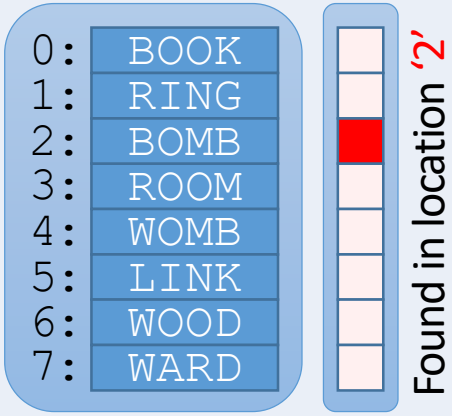
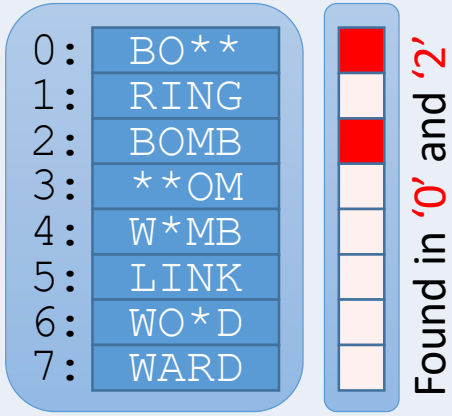
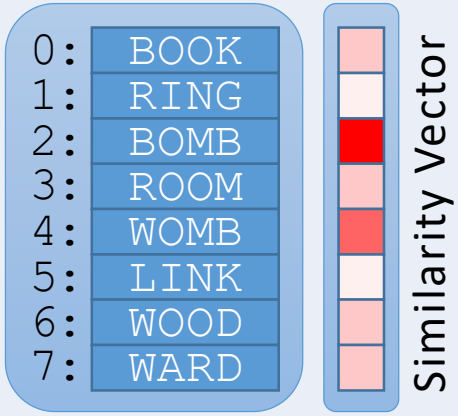
Associated Memory
AKA CAM/RAM

CAM/RAM structure to read associated data

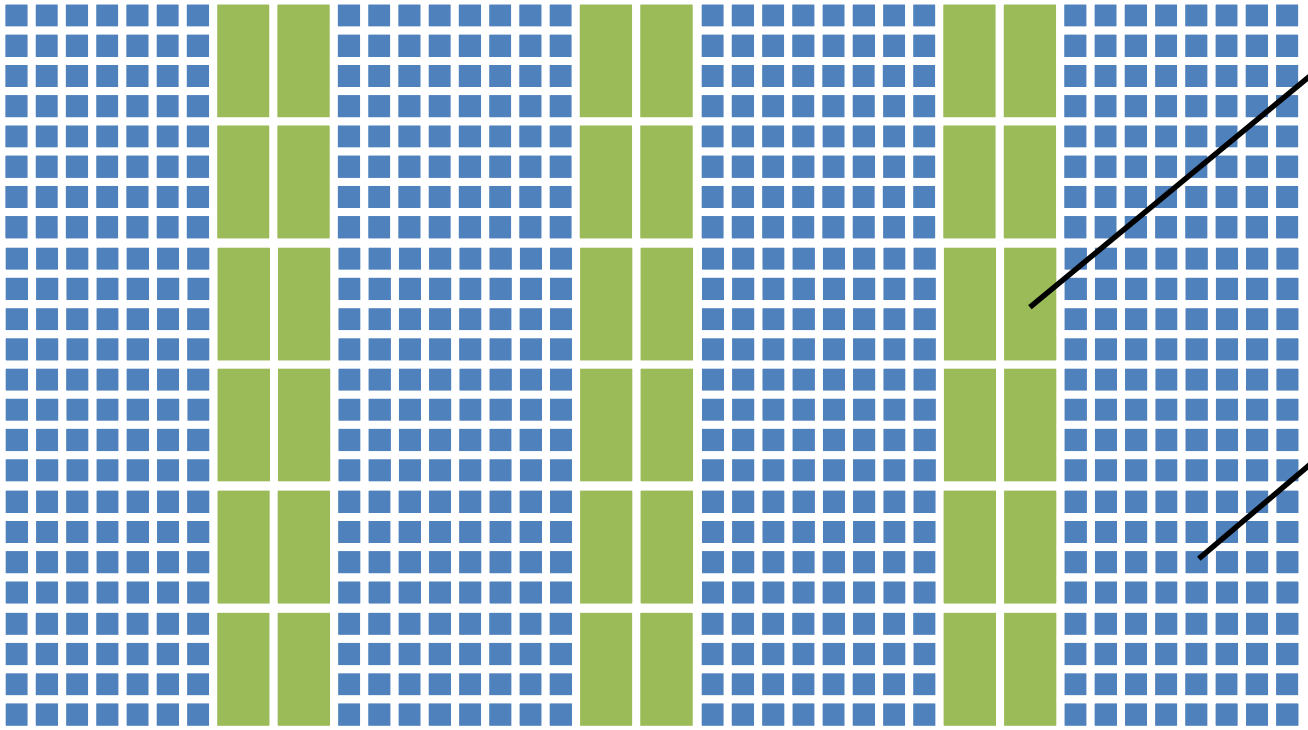
Example: search when a conference takes place



CAM Classification & Applications

Binary	Ternary	Differentiable
Search the memory for exact match	Wildcards may be used	How close is each item to the searched data?
Search for "BOMB" 	Search for "BOMB" 	Search for "BOMB" 
Expensive! Power hungry!	More expensive!	Computing intensive!
• Memory Management • Data Compression • Package classification	• IP forwarding • The internet BGP routers	Memory-augmented neural networks • Neural Turing Machines (DeepMind) • Memory Networks (Facebook AI)
FPT'14 & FCCM'15	Current work	Research in progress

Motivation - FPGAs



No dedicated CAM resources in FPGAs

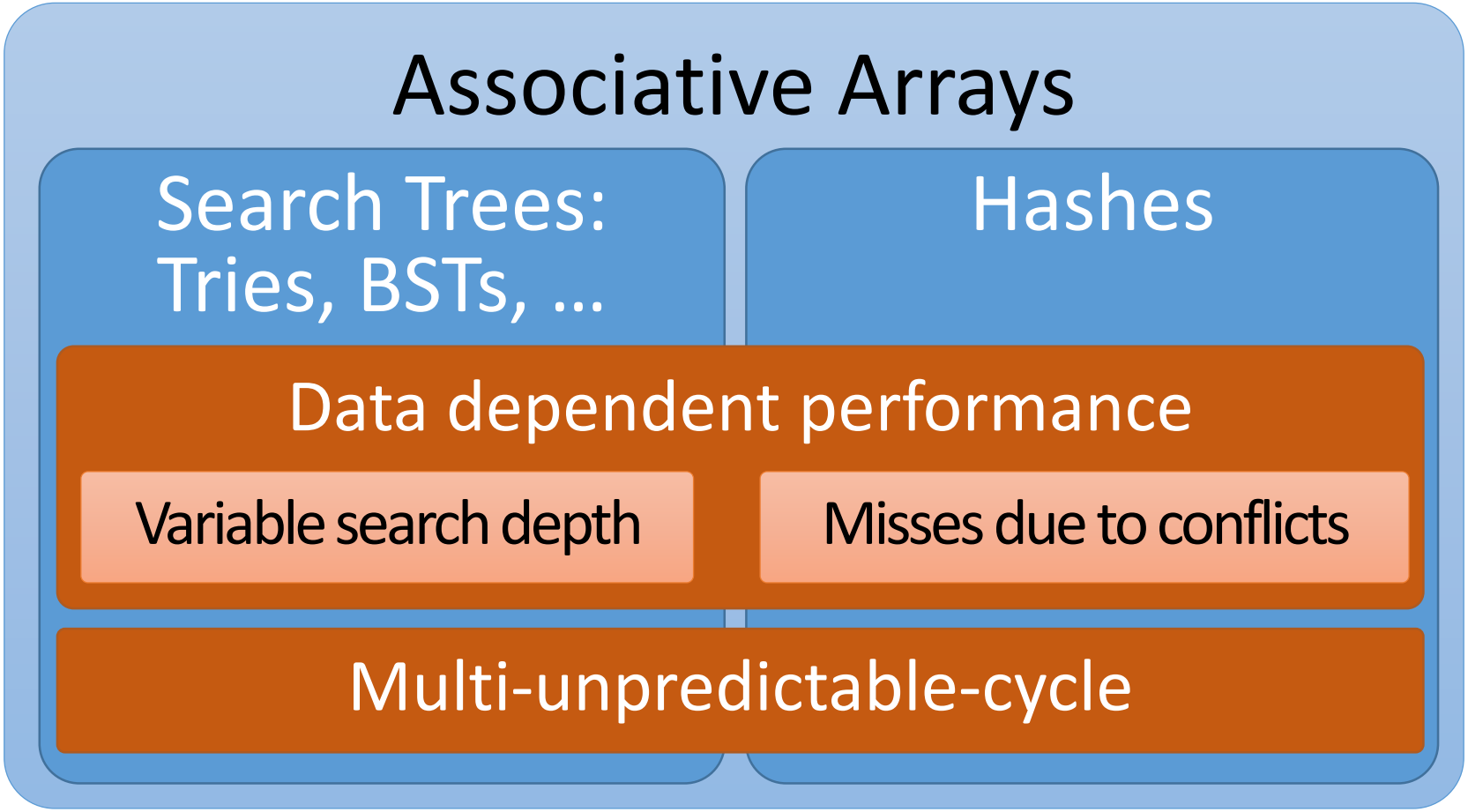
Objectives

Use BRAMs to construct

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

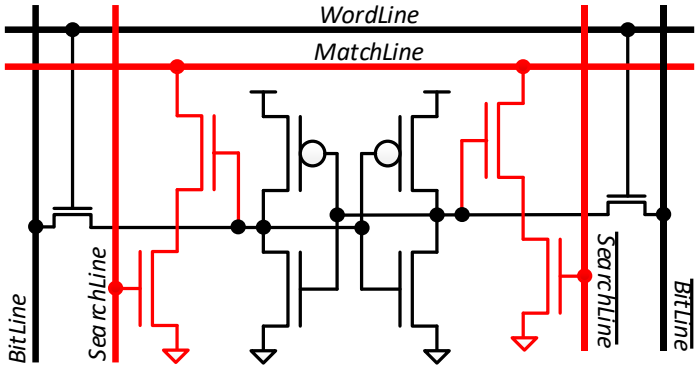
CAMs

Algorithmic Heuristics



Custom-designed CAMs

Modified SRAM cell – Custom-design in transistor level



Renesas TCAM device

- 20Mbit
- 360M Searches/Sec

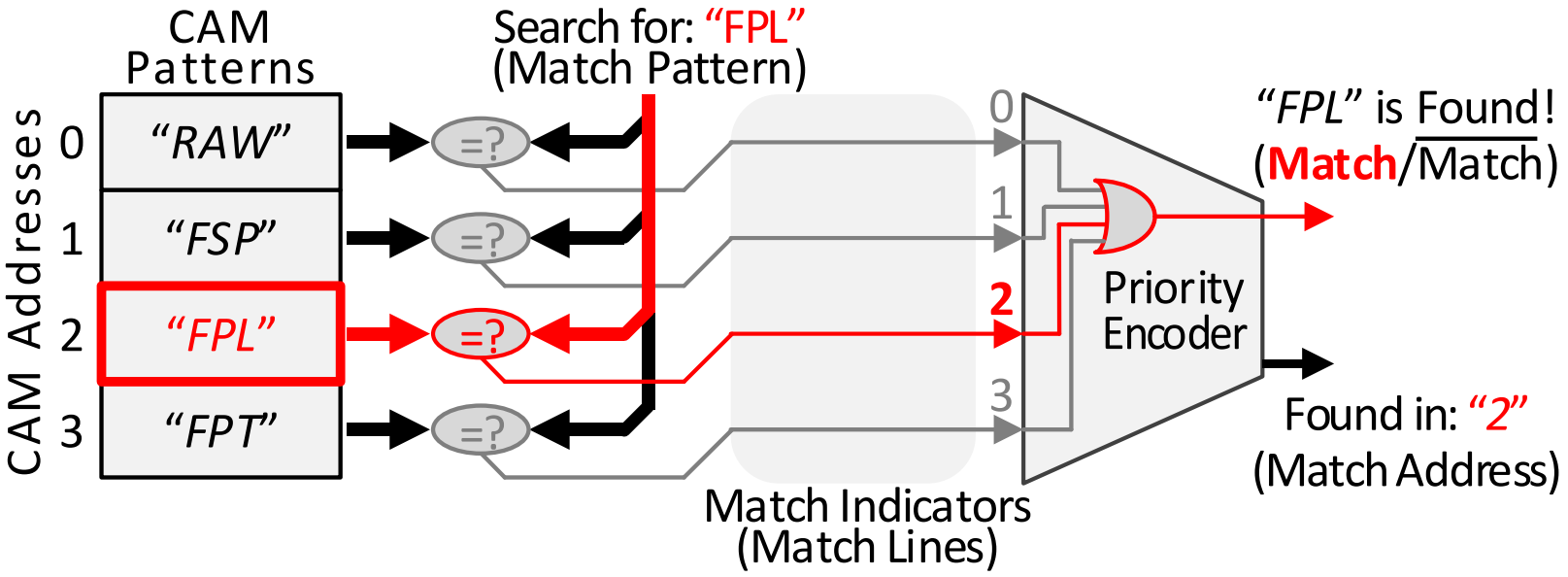
Performance

Cost

Integration overhead

Register-Based CAMs: PE-BCAM

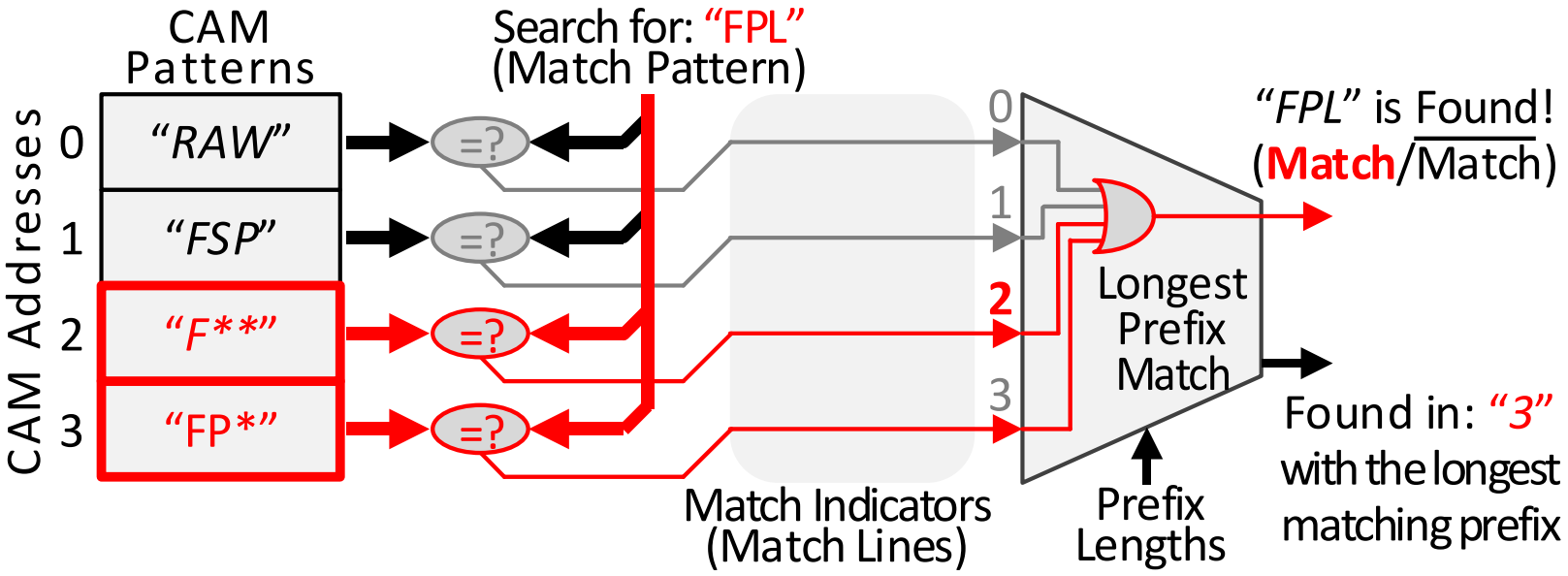
Concurrent register read and compare



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

Register-Based CAMs: LPM-TCAM

Concurrent register read and compare



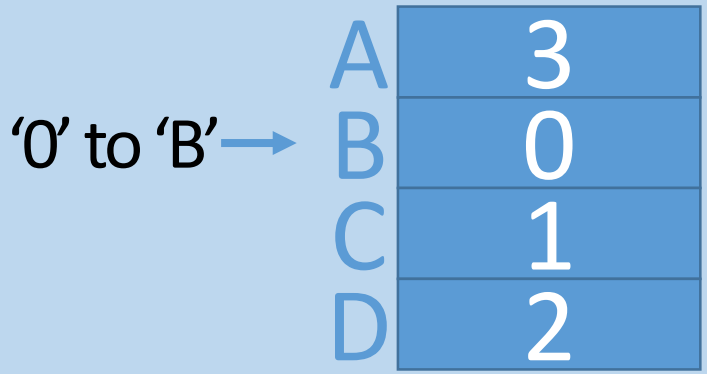
Single-cycle | Limited resources | Complex routing | Fits small CAMs

Brute-Force Transposed-RAM
A Traditional BRAM-based CAM

Key idea: Transposed RAM - data becomes addresses

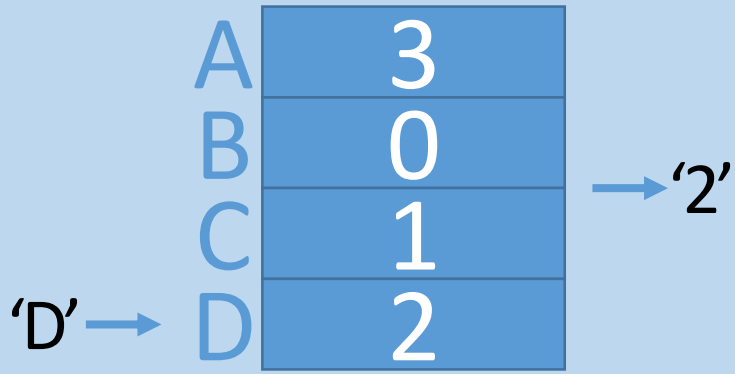
Write

Write '0' to location 'B'



Match

Read location 'D' for match

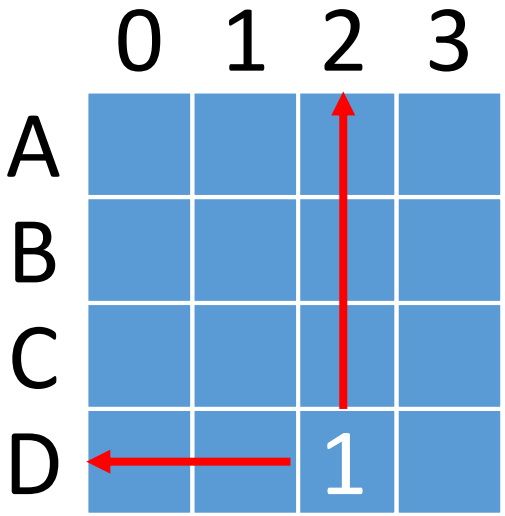


* Xilinx App Notes

Brute-Force Transposed-RAM
A Traditional BRAM-based CAM

- 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

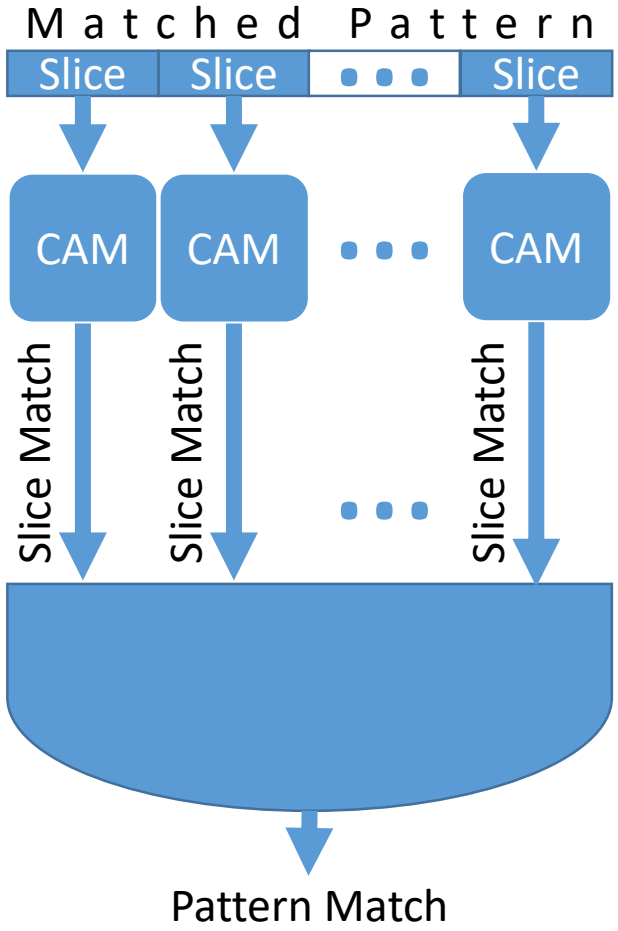
CAM Cascading

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

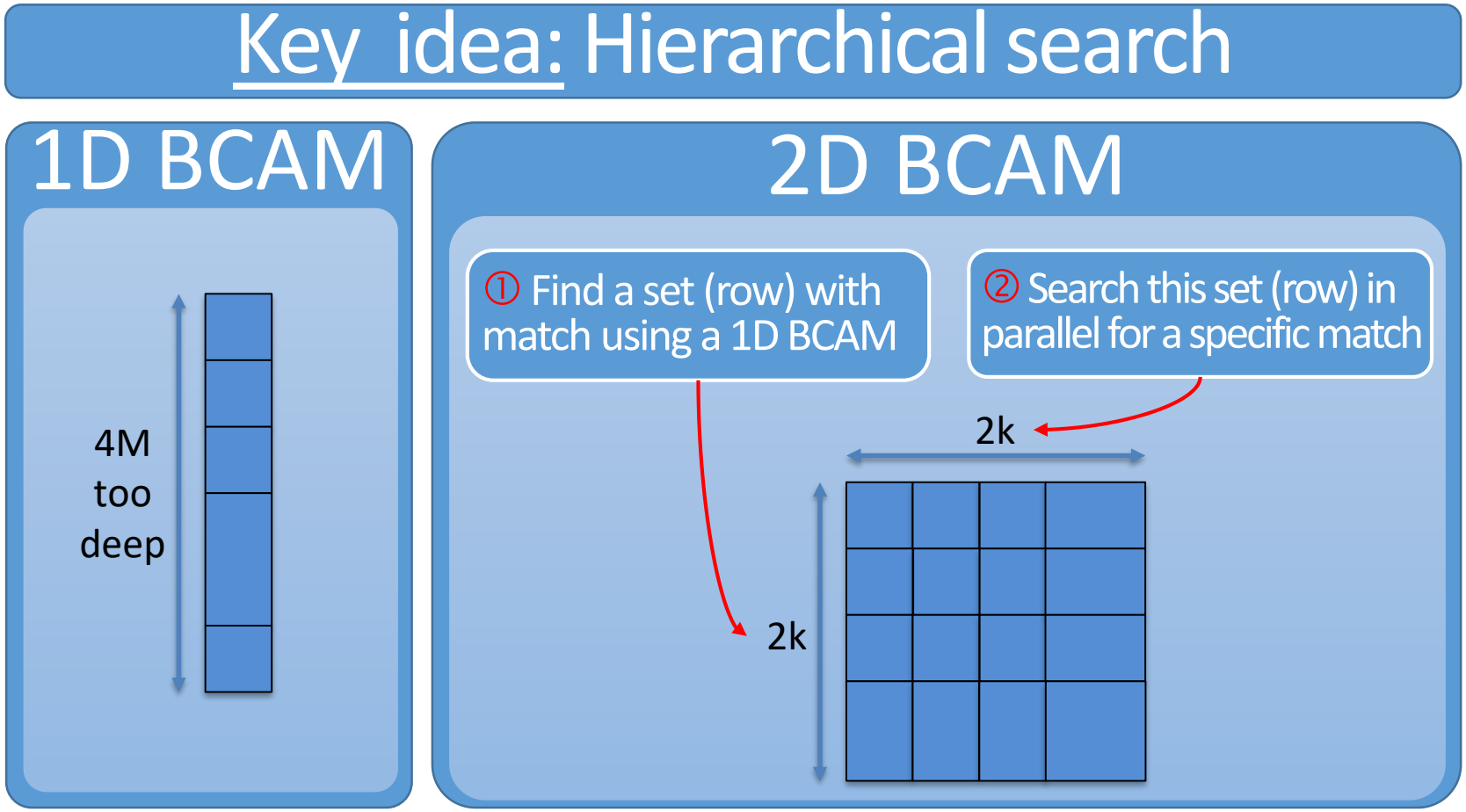
$$\text{RAM Depth} = 2^{\text{Pattern Width}}$$

- 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

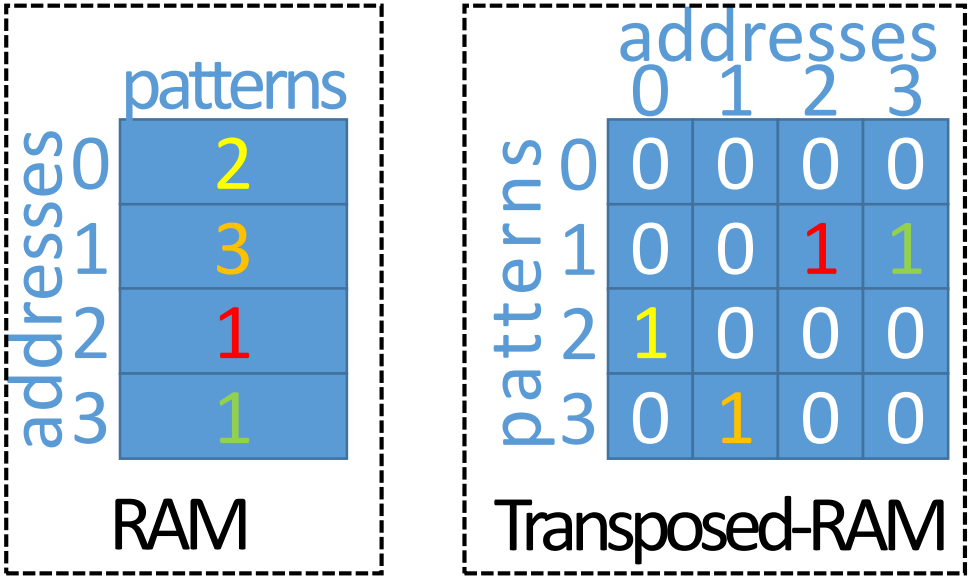
$$\text{RAM Depth} = 2^{\text{Slice Width}} \times (\text{Pattern Width} / \text{Slice Width})$$



Hierarchical Search 2D BCAM: Narrow and Deep BCAM

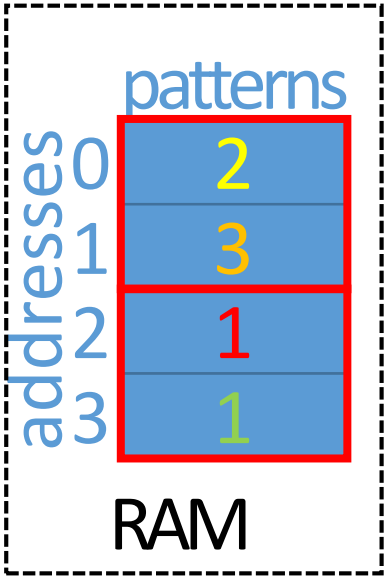


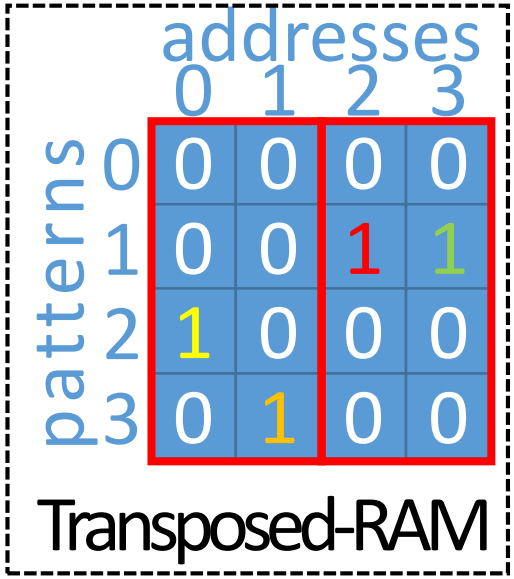
Hierarchical Search 2D BCAM: Example



Hierarchical Search 2D BCAM: Example

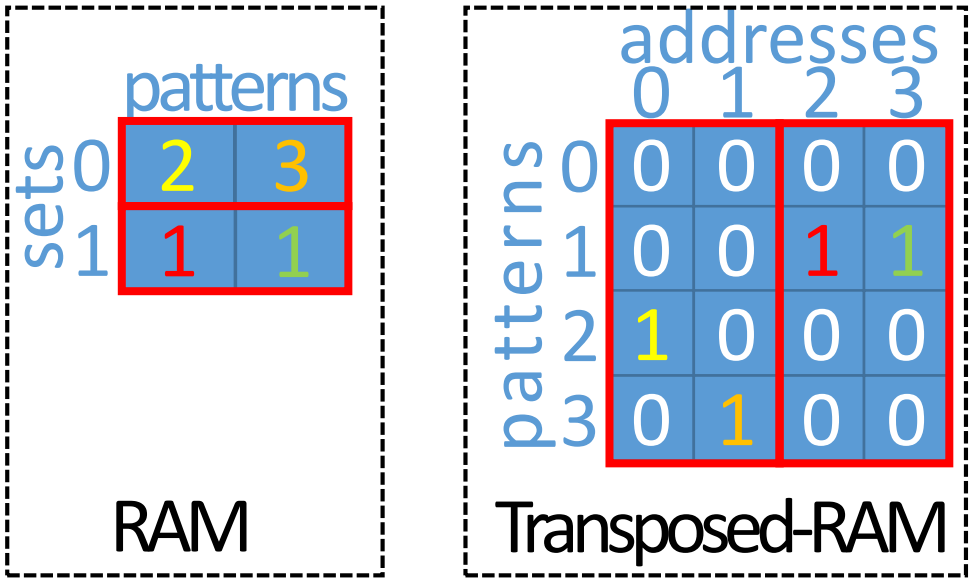
- Divide address space into sets





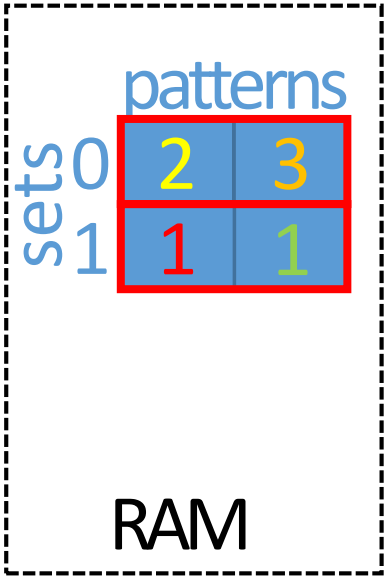
Hierarchical Search 2D BCAM: Example

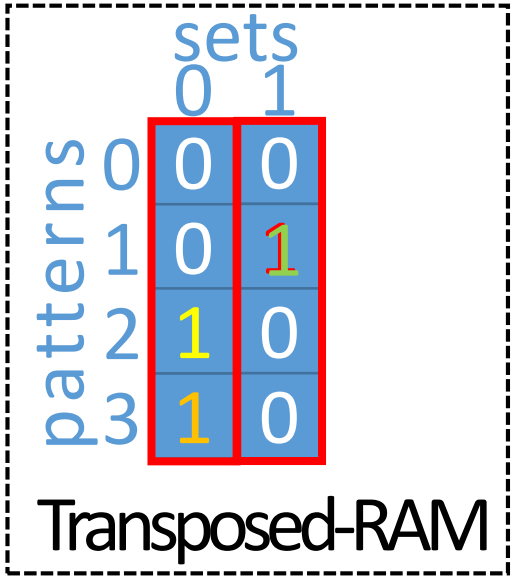
- Divide address space into sets
  - RAM: each set in a line



Hierarchical Search 2D BCAM: Example

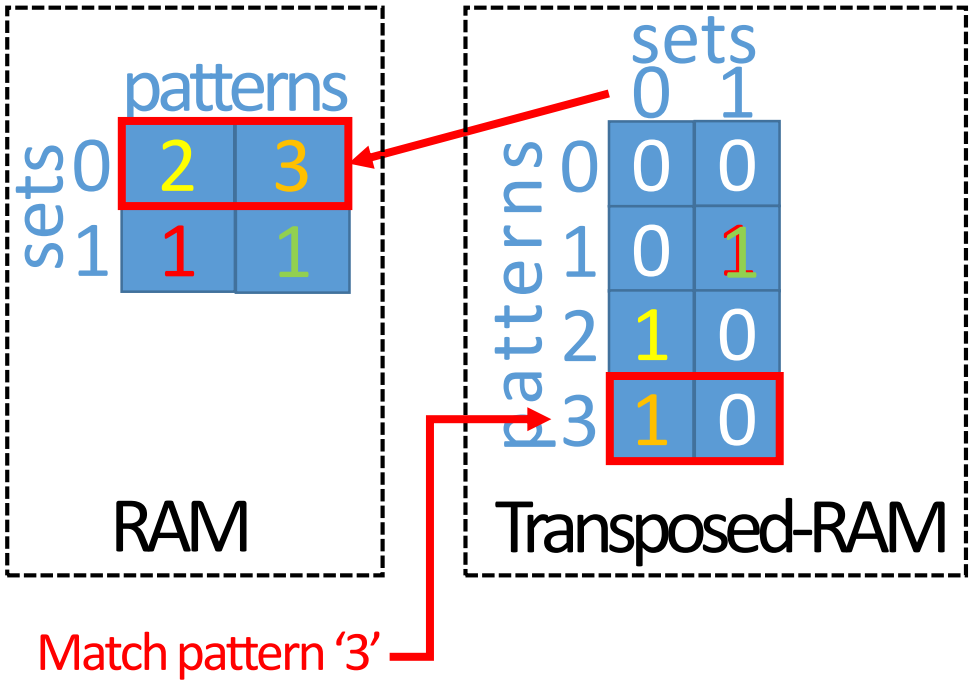
- Divide address space into sets
  - RAM: each set in a line
  - Transposed-RAM: indicates "pattern in set?"





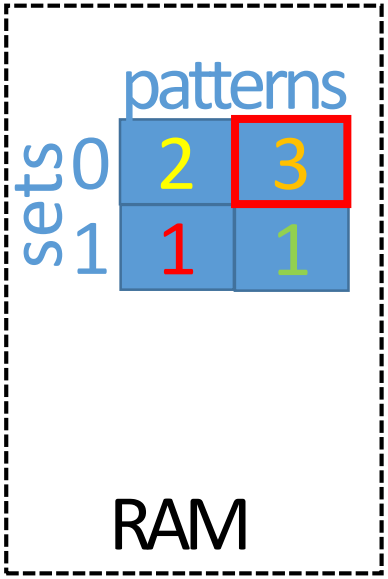
Hierarchical Search 2D BCAM: Example

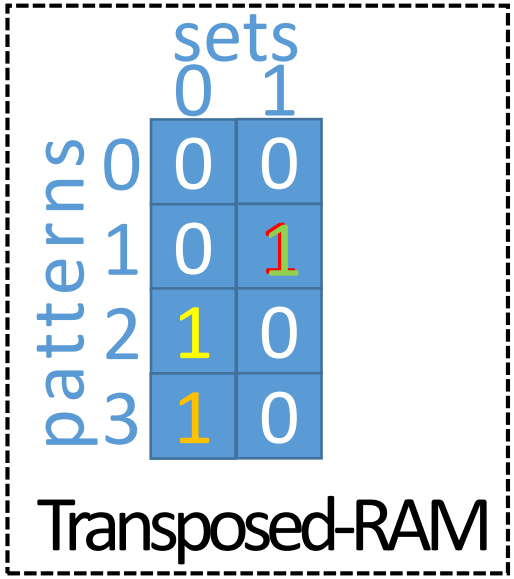
- Divide address space into sets
  - RAM: each set in a line
  - Transposed-RAM: indicates "pattern in set?"
- Hierarchical Search:
  1. Find a set (row) with match using a 1D BCAM



Hierarchical Search 2D BCAM: 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





Hierarchical Search 2D BCAM: Pros and Cons

BRAM-Based

Single-cycle

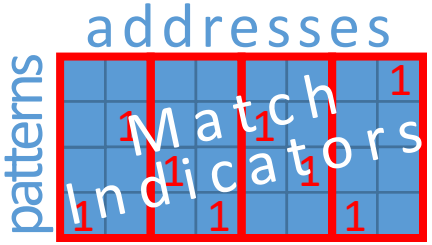
Efficient for deep CAMs

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

Indirectly-Indexed HS BCAM: Cascadable Wide and Deep BCAM

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

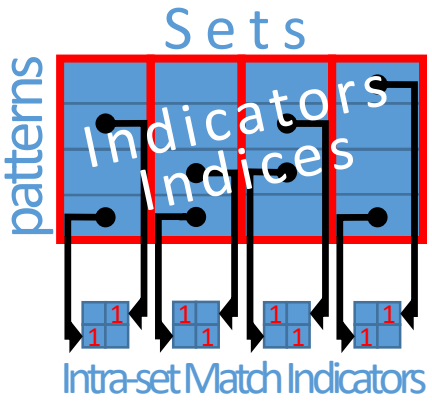
Key observation	
Transposed RAM is a sparse matrix	*n* columns (set of addresses) accommodates *n* matches (1's) at most!



Indirectly-Indexed HS BCAM: 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 HS BCAM: 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

↓

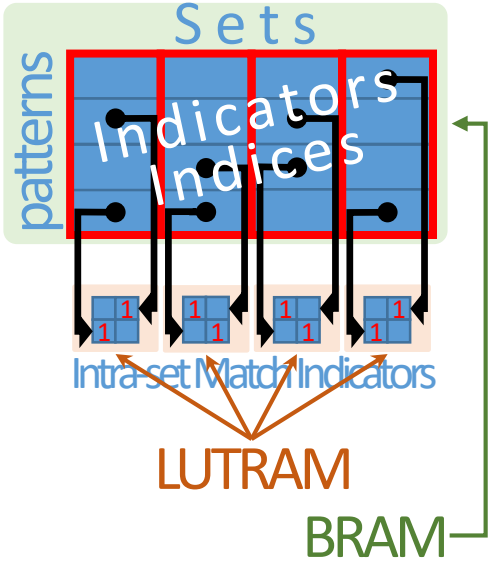
Cascadable

↓

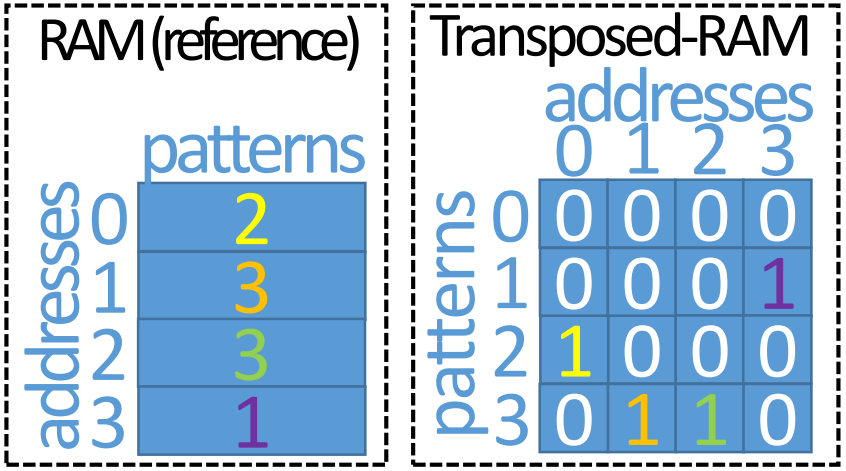
Scalable (linear growth)

↓

Supports wider patterns

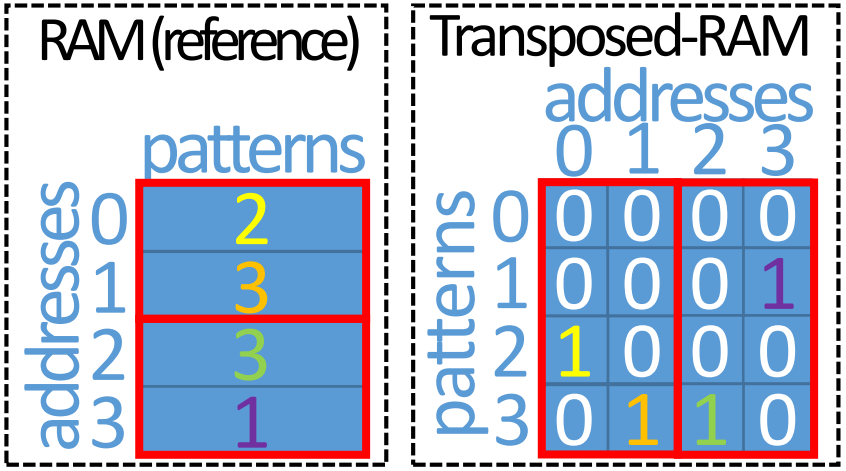


Indirectly-Indexed HS BCAM: Example



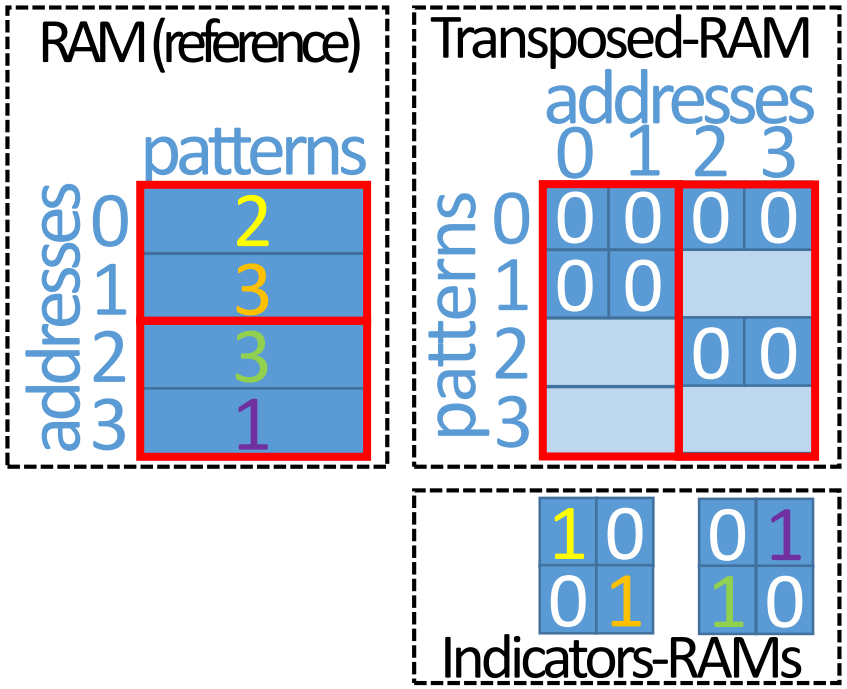
Indirectly-Indexed HS BCAM: Example

- Divide address space into sets



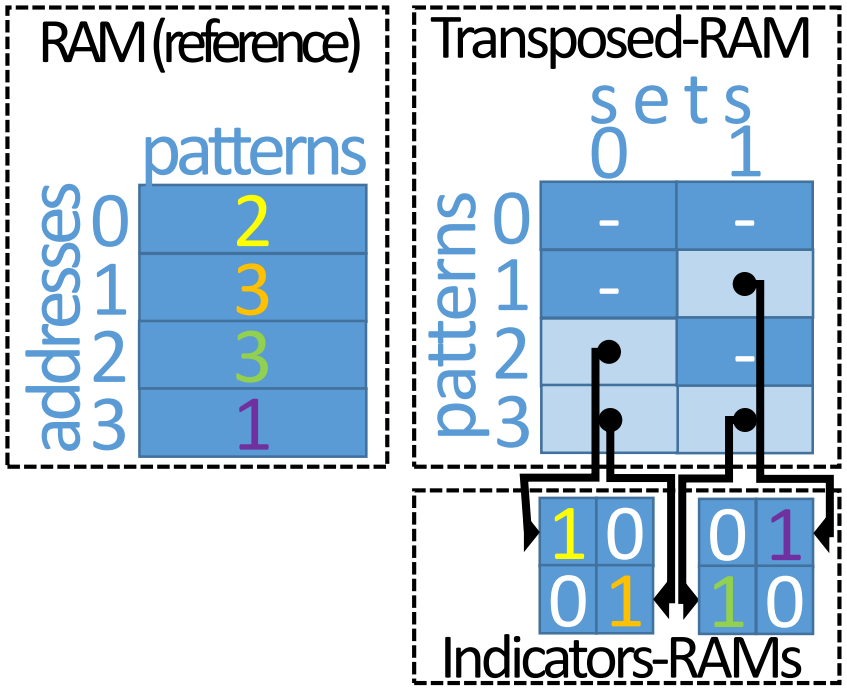
Indirectly-Indexed HS BCAM: Example

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



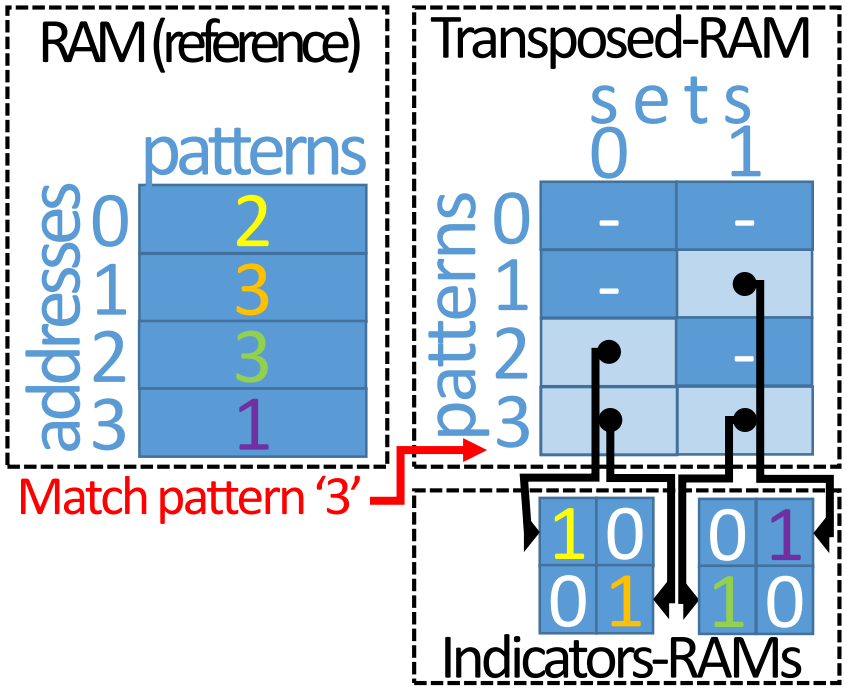
Indirectly-Indexed HS BCAM: Example

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



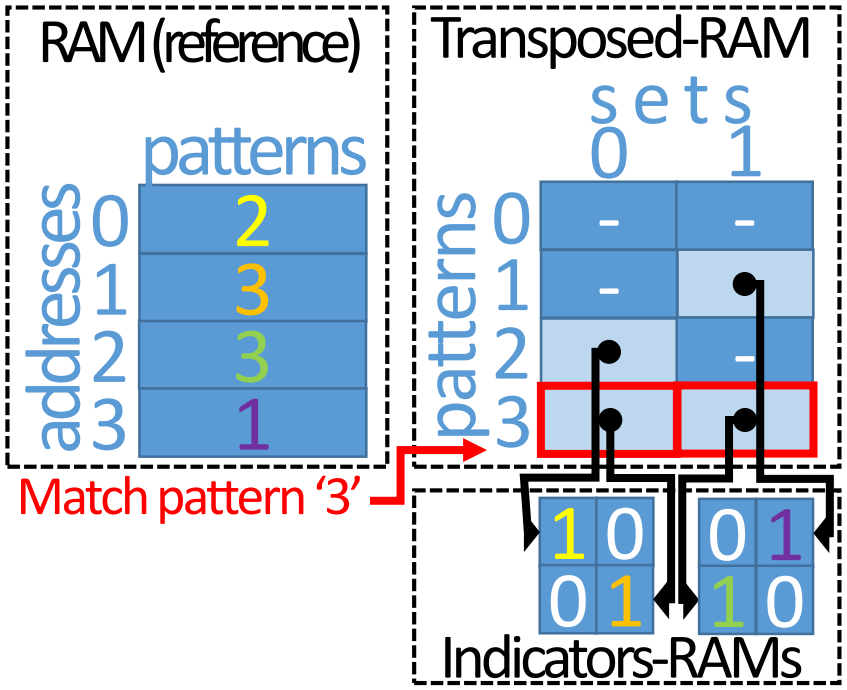
Indirectly-Indexed HS BCAM: Example

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



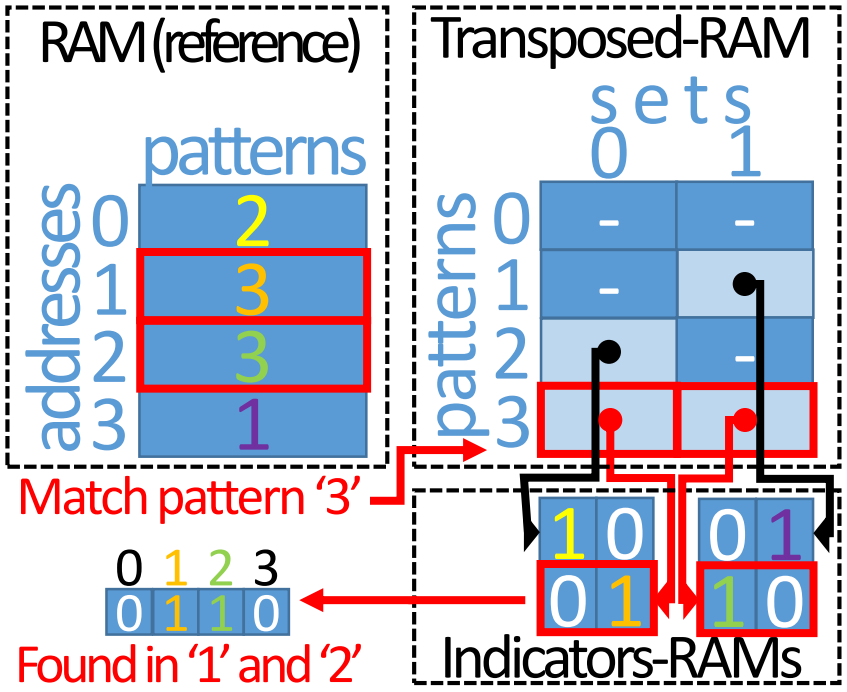
Indirectly-Indexed HS BCAM: Example

- 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 HS BCAM: Example

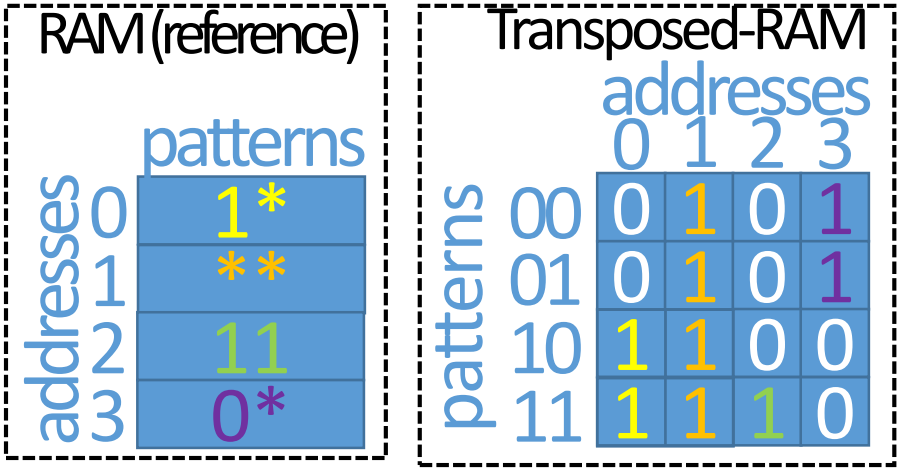
- Divide address space into sets
- Store sets with a match in Indicators-RAM
- Transposed-RAM stores indices to all matches in set
- Hierarchical Search:
  - Find indices of all matching sets in Transposed-RAM
  - Read Indicators-RAM using indices from Transposed-RAM



Indirectly-Indexed HS TCAMs

- Can Indirectly-Indexed HS be applied to TCAMs?

Observation:
TCAM addresses (columns) may have more than one pattern (due to wildcards)

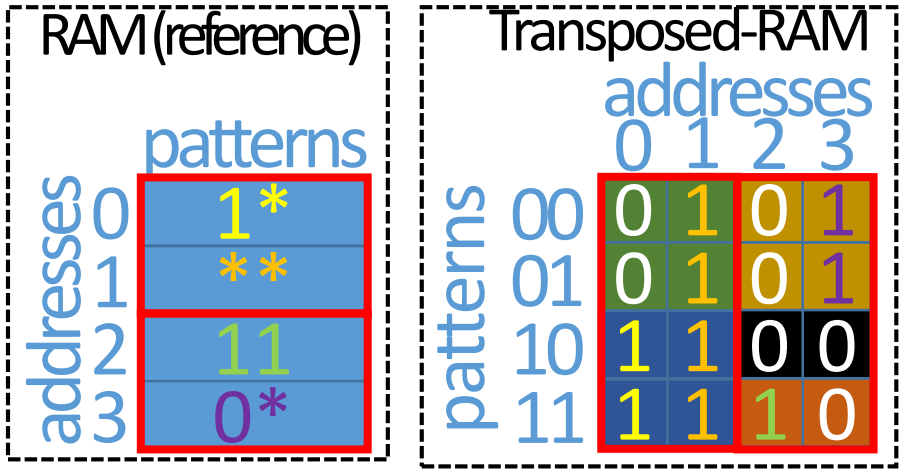


- Can we still do the same set grouping as in II-HS-BCAM? The answer is **YES**!

Indirectly-Indexed HS TCAMs

Key Observation:

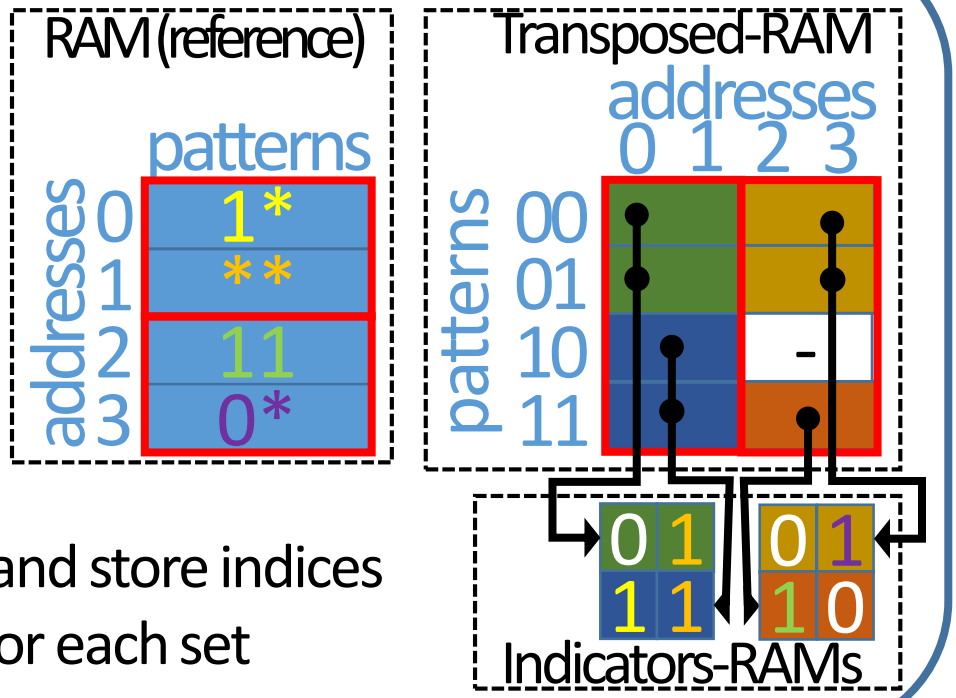
A set of ***n*** addresses (columns) has at most ***n*** different lines (proof in paper)



Indirectly-Indexed HS TCAMs

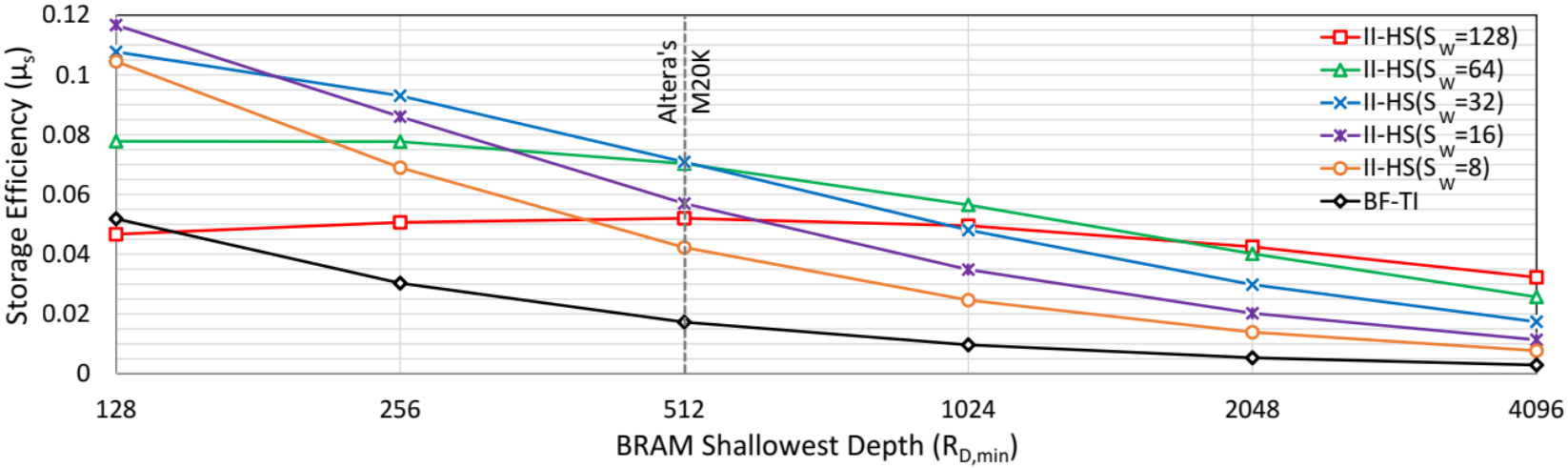
Key Observation:

A set of ***n*** addresses (columns) has at most ***n*** different lines (proof in paper)

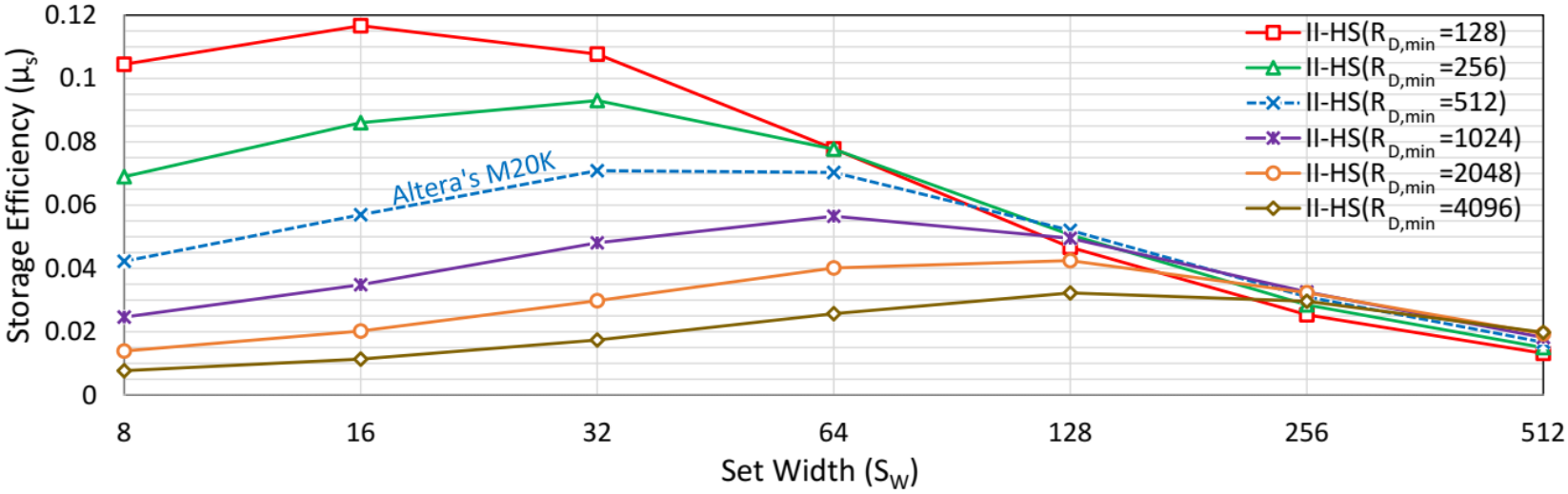


- Move lines to LUTRAMs and store indices
- Requires *n*×*n* LUTRAMs for each set

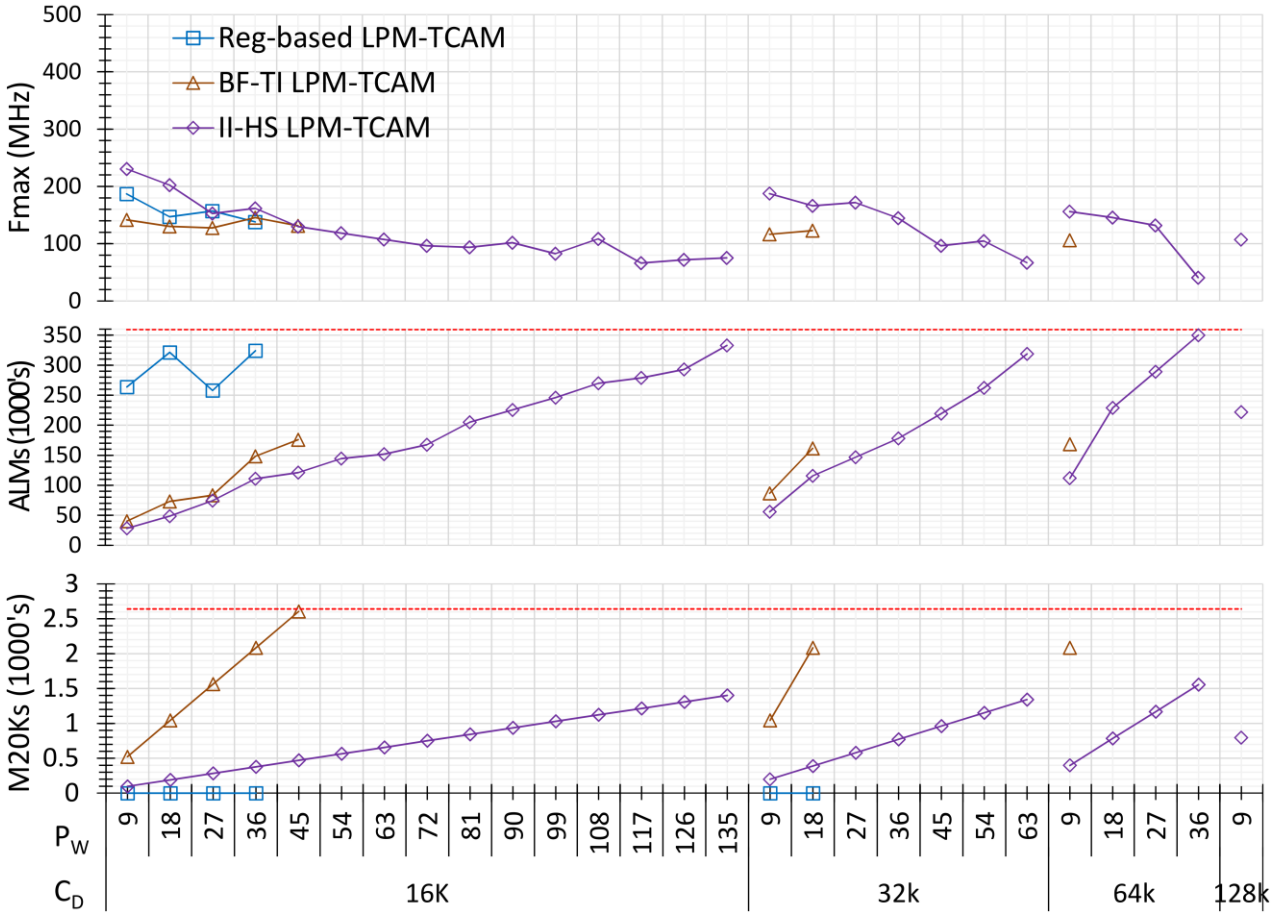
Indirectly-Indexed HS TCAMs: Design parameters



Indirectly-Indexed HS TCAMs: Design parameters



Indirectly-Indexed HS TCAMs: Area and Performance



Open Source

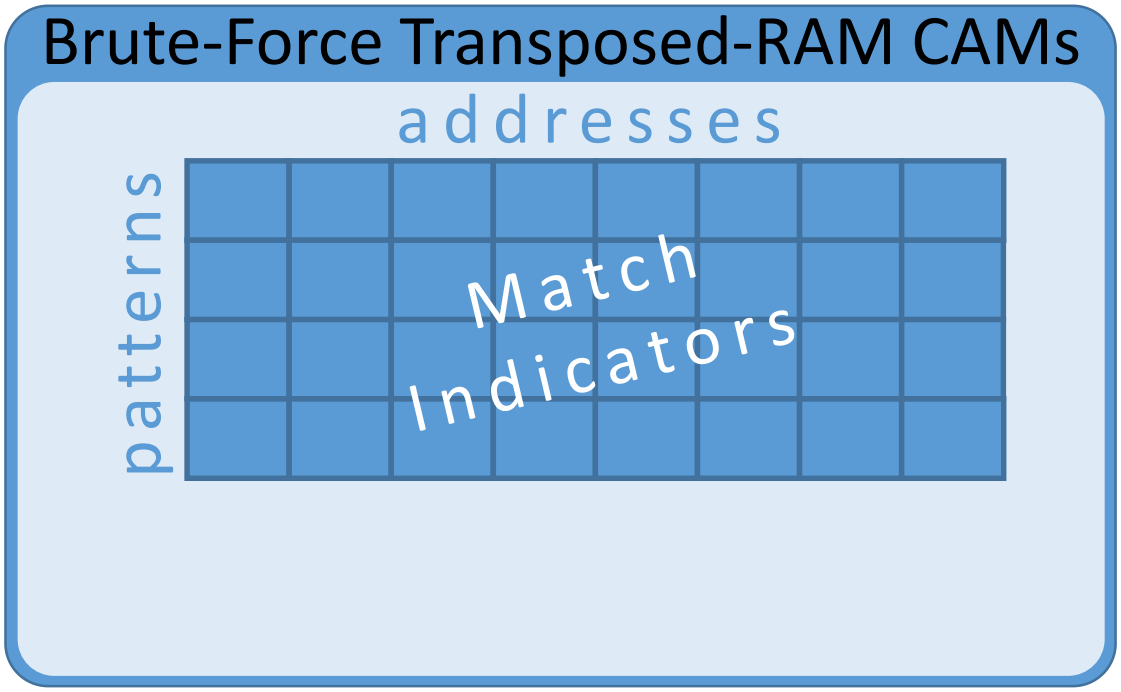
	HS BCAM (FPT'14)	II-HS BCAM (FCCM'15)	II-HS TCAM (FPL'18)
Patterns support	Narrow	Wide	Wide
Match encoding	PE	PE	LPM
Storage efficiency	90%	8%	8%
Fmax (Stratix V)	Up to 550MHz	Up to 300MHz	Up to 200MHz
Cycle/Update	2	2	Shallowest RAM Depth
Search/cycle	1	1	1
Search latency	2	~ $\log_4$(depth)	~ $\log_4$(depth)

Available as open source: **https://github.com/AmeerAbdelhadi**

- Modular and parametric Verilog files
- Run-in-batch simulation and synthesis manager

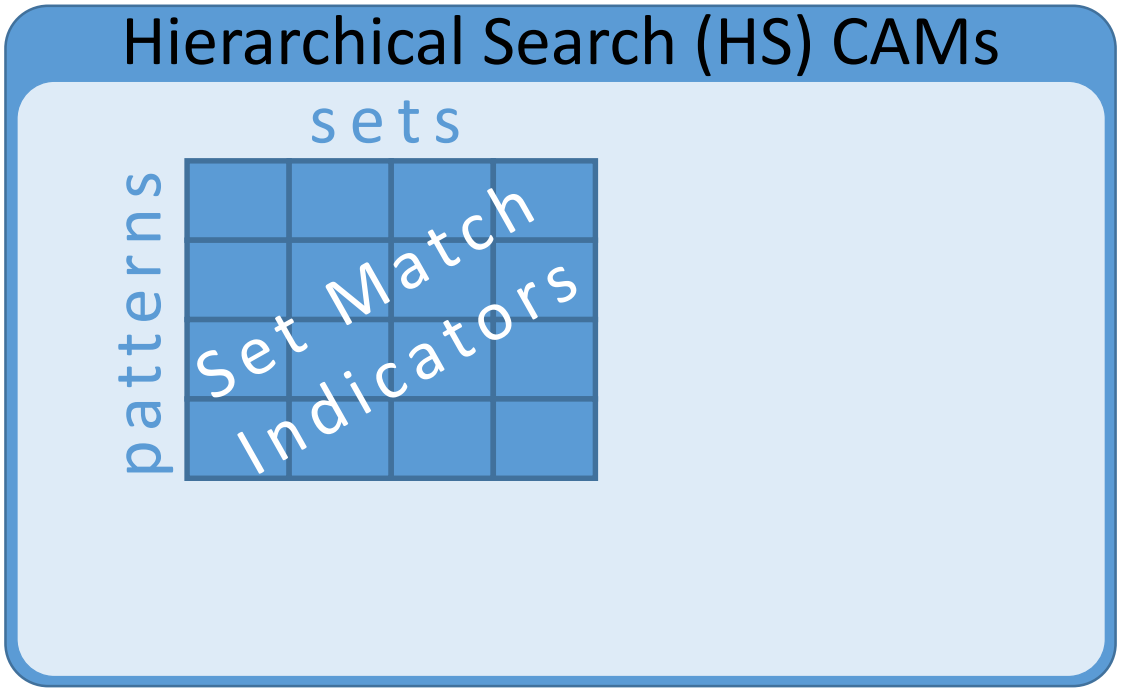
Conclusions

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



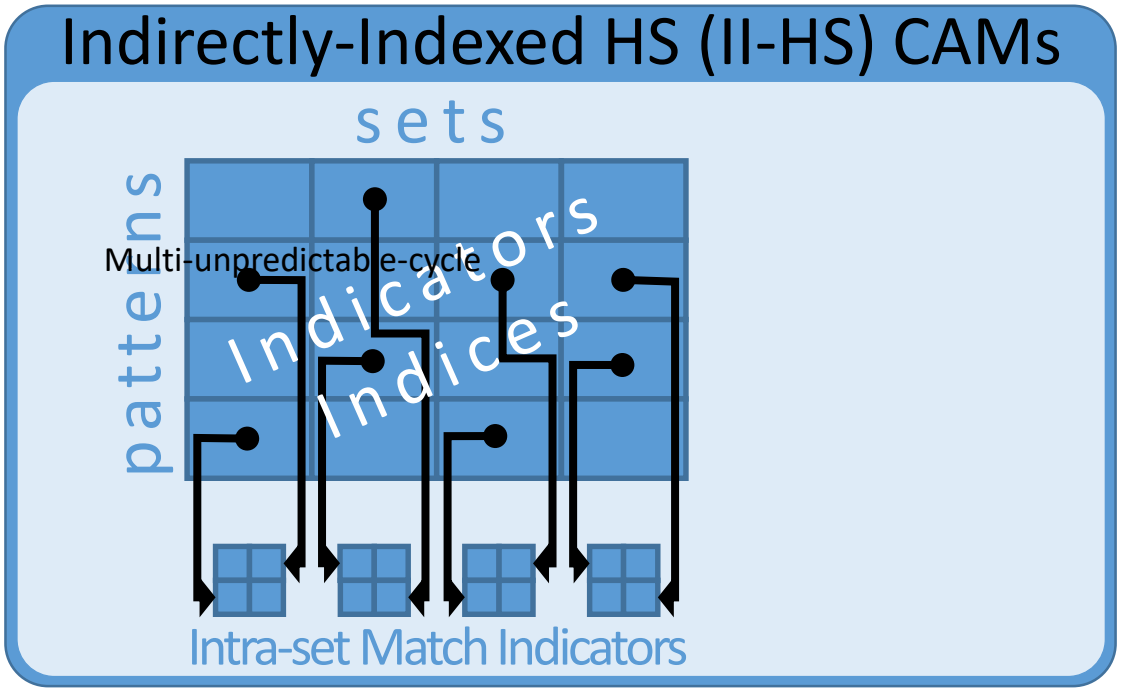
Conclusions

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



Conclusions

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



Thank You!

Backup Slides

Conclusions