Content Addressable Memories and Associative Processors

Asif Ali Khan

Fall Semester 2024

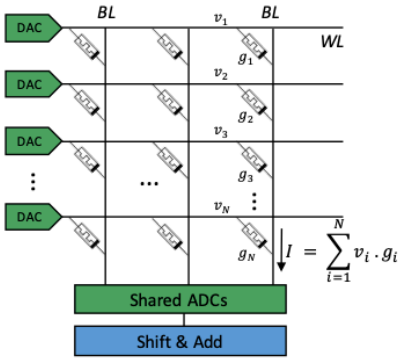
Department of Computer Systems Engineering

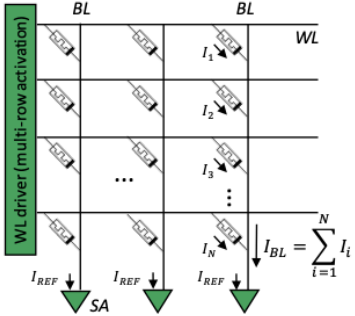
UET Peshawar, Pakistan

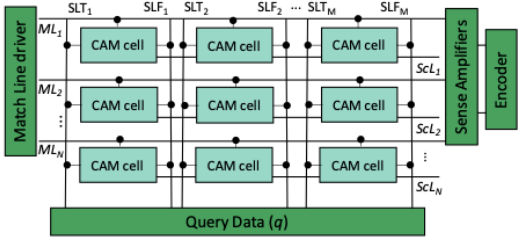
Dec 5, 2024

Recap: Compute-in-memory (CIM)

- The CIM paradigm aims to completely eliminate the data movement
- The fundamental idea is to exploit the physical properties of the memory devices to perform computations
- Not every computation can be performed with every technology

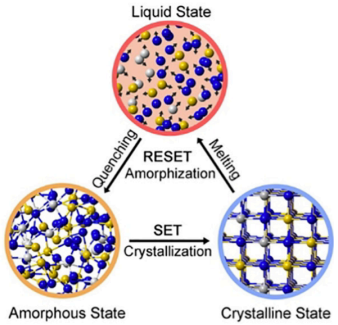




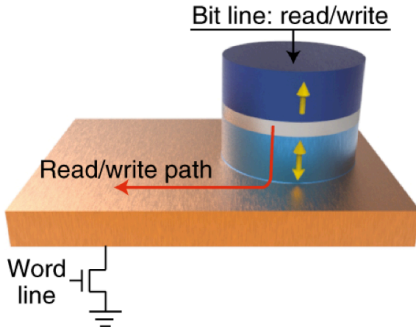


Khan et al., Arxiv 2024

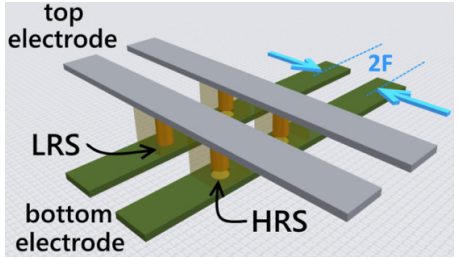
Recap: Emerging nonvolatile memories (NVMs)



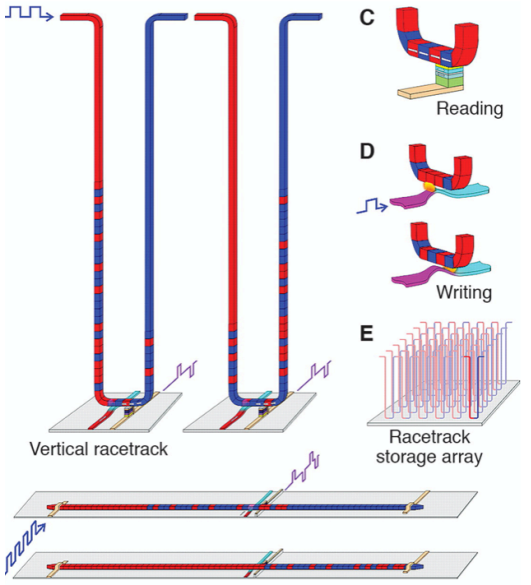
Zhang et al., 2020



G. Yu, 2020



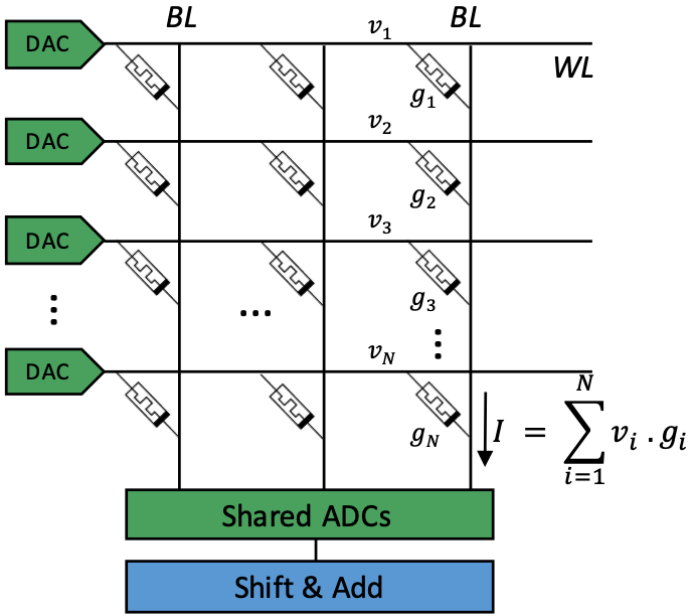
Lim et al, 2015



Parkin et al, 2008

- Other options: FeFET, FeRAM etc
- Each technology has its strengths and challenges
- PCM and MRAM receive a lot of traction in industry

Recap: CIM crossbar

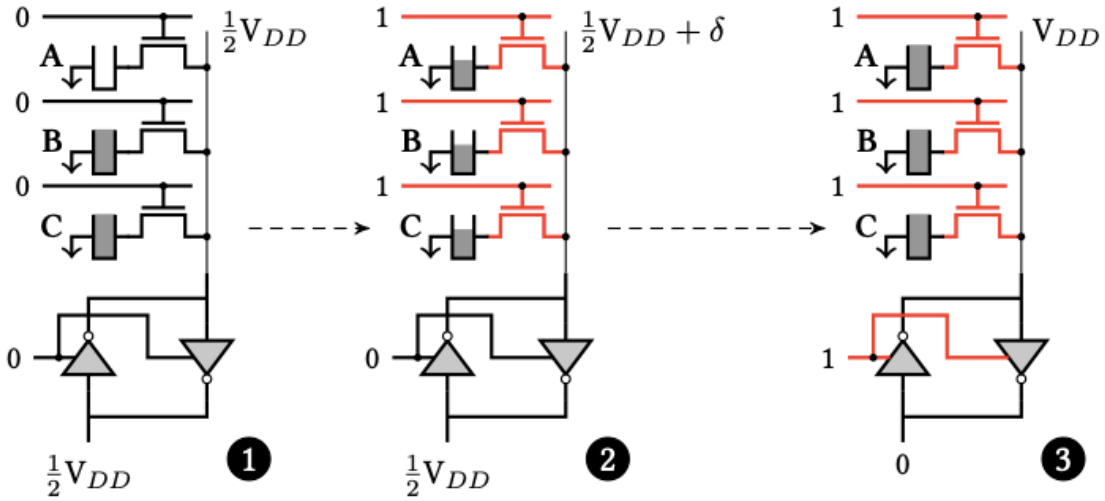


- Program one operand into memristors devices (conductance)
- Enable all wordlines simultaneously and apply another operand as input
- The accumulated current at the bitlines using Kirchoff's law produces the outcome of dot product
- Analog domain computation – results are approximate

Khan et al., Arxiv 2024

Recap: CIM-logic using DRAM

- Simultaneous activation of three rows in a DRAM array results in bit-wise majority
  - At least two cells have to be one, for the output to be one
  - The operation is called *triple row activation* (TRA)



Seshadri et al., MICRO 2017

Recap: CIM-logic using DRAM

- Lets say `A, B, C` represent the state of the three cells
- The final state of the bitline is: `AB + AC + BC`
- This can be rewritten as: `C.(A+B)+C'.(AB)`
- By controlling C, we can implement both `AND` and `OR` operations
  - `C=1`, to implement `OR` and `C=0`, to implement `AND` operation
- Important: TRA destroys contents of the involved cells
  - Contents need to be copied to a different place first, if important/needed

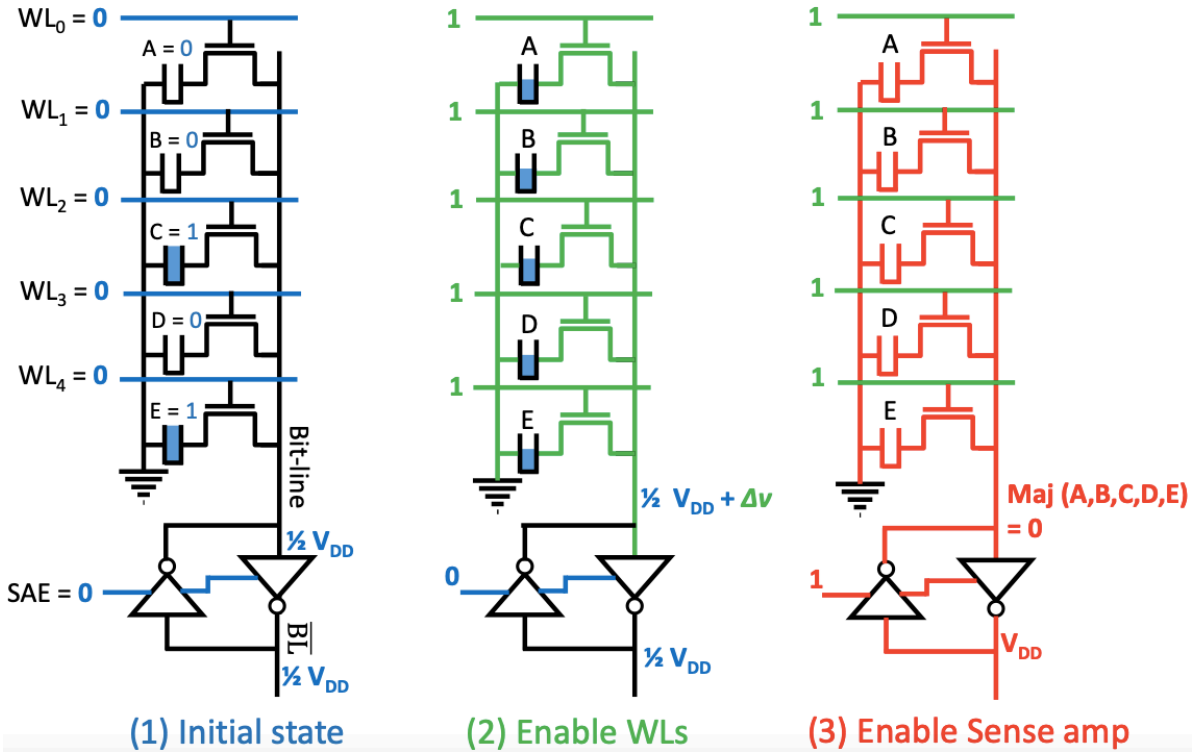
Seshadri et al., MICRO 2017

Recap: In-DRAM addition

$$Cout = Majority(A, B, Cin)$$

$$Sum = Majority(A, B, Cin, \overline{Cout}, \overline{Cout})$$

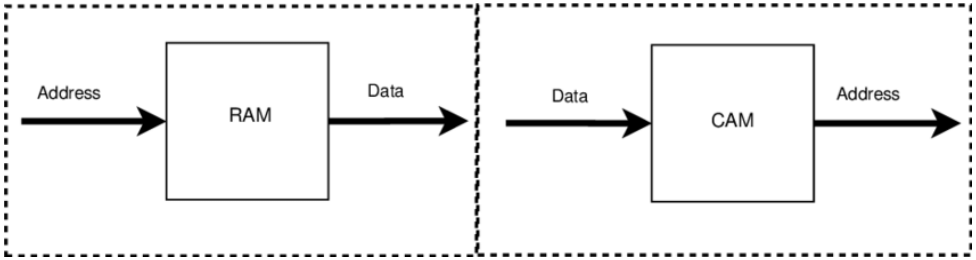
- Note that the operations are bit-serial
- … but word-parallel
- Particularly useful for bulk additions on low-precision numbers



Roy et al., IEEE JESTiC 2021

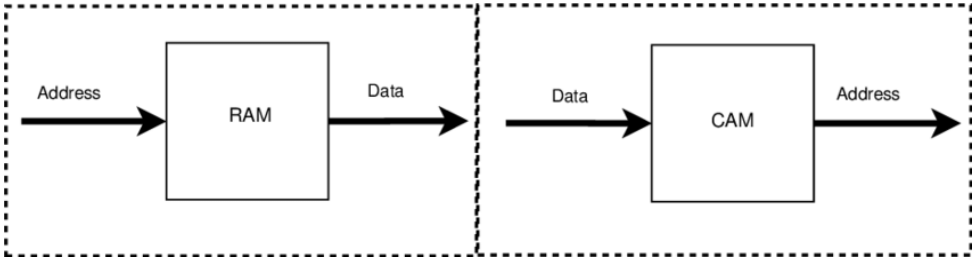
Content-addressable memory (CAM)

- Computer memory that allows data to be accessed based on content rather than specific addresses



Content-addressable memory (CAM)

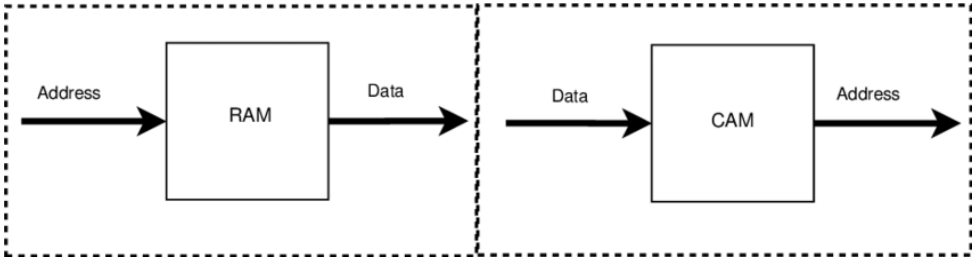
- Computer memory that allows data to be accessed based on content rather than specific addresses



- Search entire memory for matching content word
  - Outputs locations of matching content

Content-addressable memory (CAM) or Associative Memory

- Computer memory that allows data to be accessed based on content rather than specific addresses



- Search entire memory for matching content word
  - Outputs locations of matching content
- Typical use-cases: Database searches, packet forwarding in networking, ML, etc.

CAM types

- CAM array can be:
  - Binary (BCAM)
  - Ternary (TCAM)

CAM types

- CAM array can be:
  - Binary (BCAM)
  - Ternary (TCAM)
- BCAMs only allow for exact match

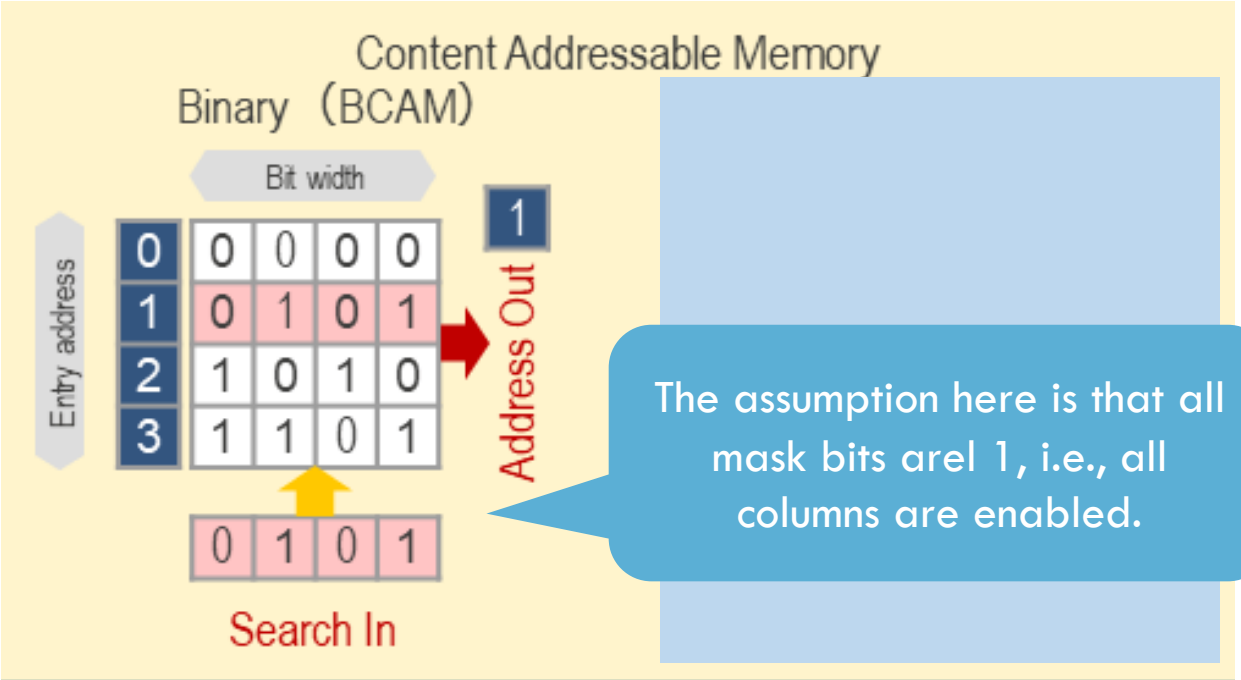
CAM types

- CAM array can be:
  - Binary (BCAM)
  - Ternary (TCAM)
- BCAMs only allow for exact match
- In TCAMs, a bit can be in `set`, `reset`, `don't care (x)` state

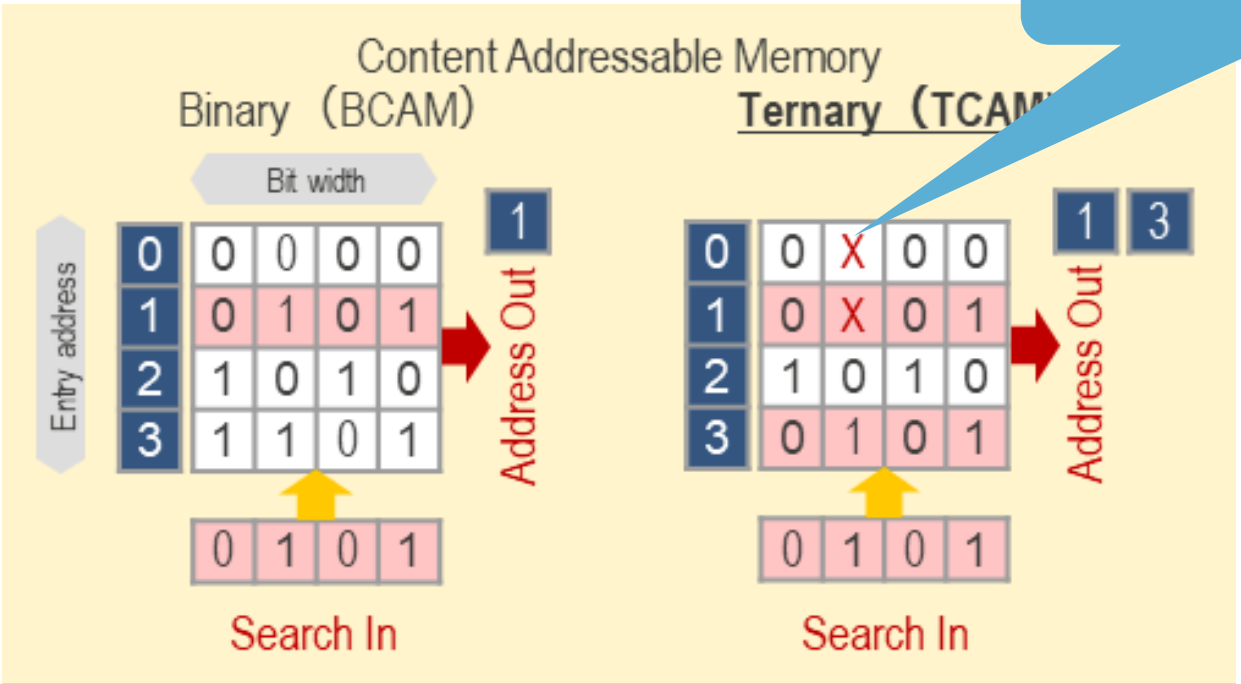
CAM types

- CAM array can be:
  - Binary (BCAM)
  - Ternary (TCAM)
- BCAMs only allow for exact match
- In TCAMs, a bit can be in `set`, `reset`, `don't care (x)` state
- This opens the possibility to do approx. and range based searches in TCAMs

Background: Search with BCAM



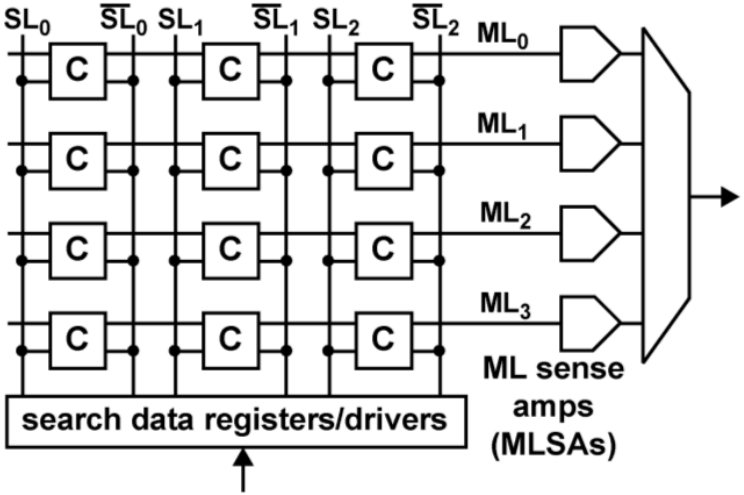
Background: Search with TCAM



The don't care bit results in a match

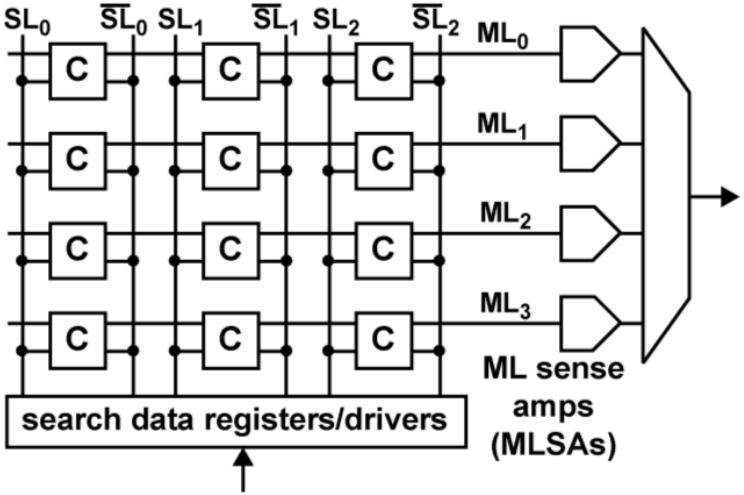
CAM architecture and functionality

- Initially the data to be searched is stored into the search data reg.



Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

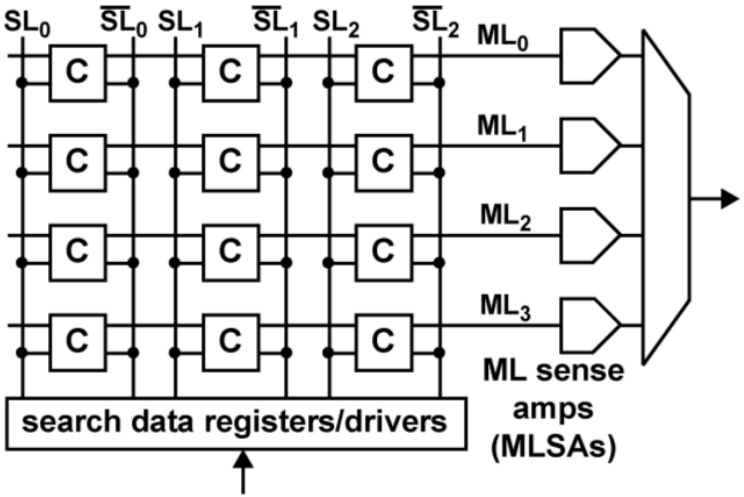
CAM architecture and functionality



- Initially the data to be searched is stored into the search data reg.
- All matchlines (MLs) are precharged to the match state (high)

Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

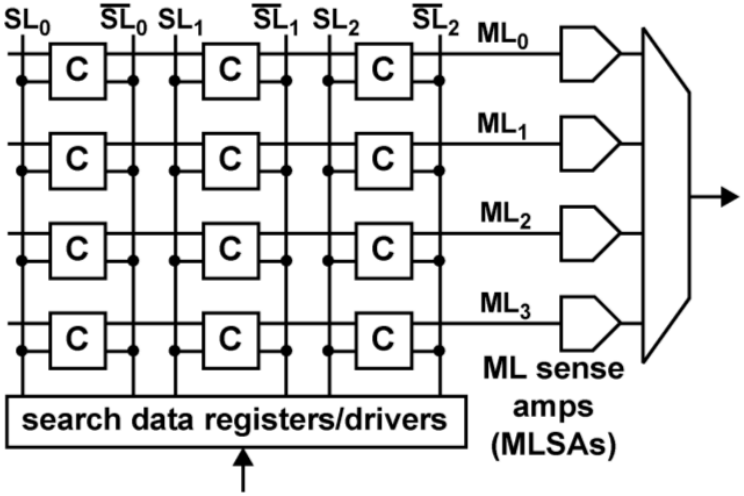
CAM architecture and functionality



- Initially the data to be searched is stored into the search data reg.
- All matchlines (MLs) are precharged to the match state (high)
- The searchline (SL) drivers broadcast the search word onto the differential SLs

Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

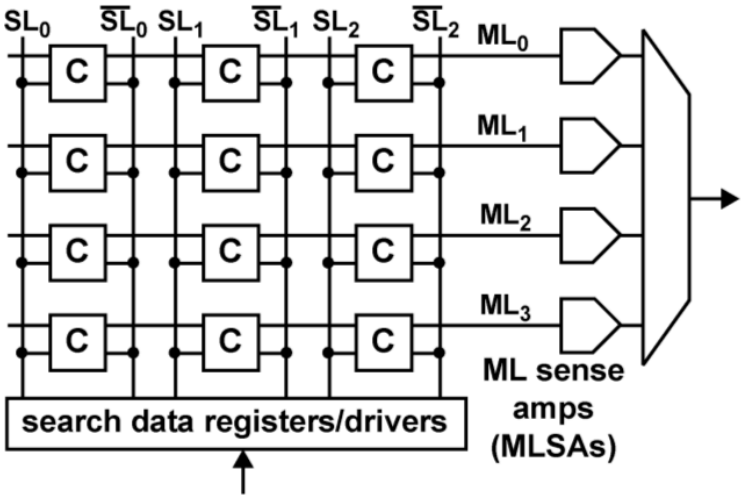
CAM architecture and functionality



- Initially the data to be searched is stored into the search data reg.
- All matchlines (MLs) are precharged to the match state (high)
- The searchline (SL) drivers broadcast the search word onto the differential SLs
- Each CAM cell compares its stored value to the value on the SLs

Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

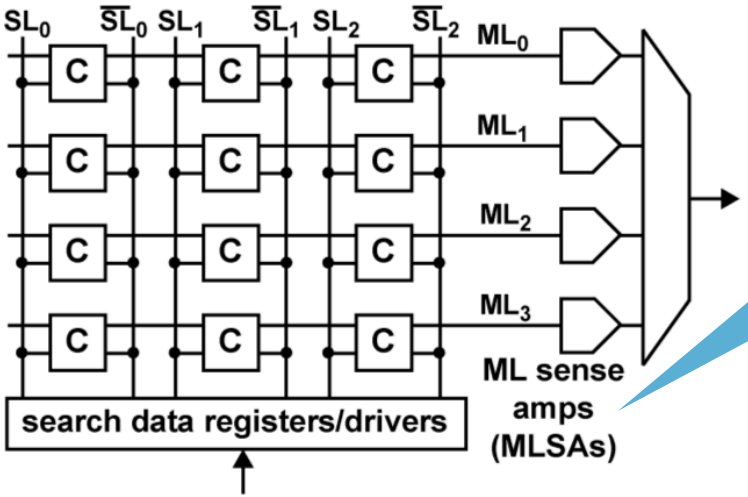
CAM architecture and functionality



- Initially the data to be searched is stored into the search data reg.
- All matchlines (MLs) are precharged to the match state (high)
- The searchline (SL) drivers broadcast the search word onto the differential SLs
- Each CAM cell compares its stored value to the value on the SLs
- Only MLs where all bits match remain in high state, all others discharge to ground

Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality



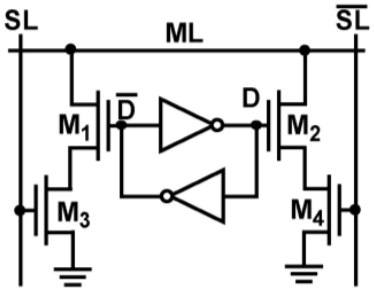
- Initially the data to be searched is stored into the search data reg
- rged to the
- Th chline (SL) drivers broadcast the arch word onto the differential SLs
- Each CAM cell compares its stored value to the value on the SLs
- Only MLs where all bits match remain in high state, all others discharge to ground

The matchline sense amplifier (MLSA) finally sense the match/mismatch state

Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality

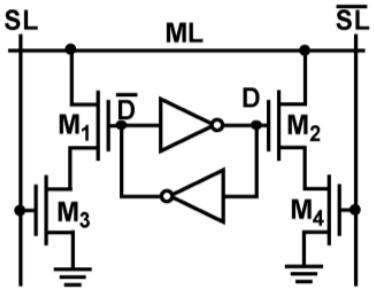
- A CAM cells both stores data and performs the XOR-like comparison



Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality

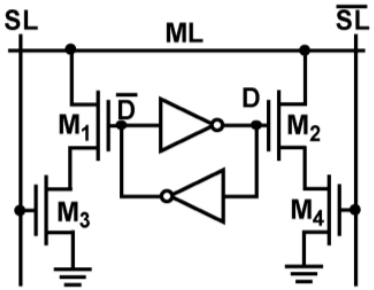
- A CAM cells both stores data and performs the XOR-like comparison
- Consider the SRAM-based CAM cell in the figure
  - Does not show the bitlines and wordlines for simplicity



Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality

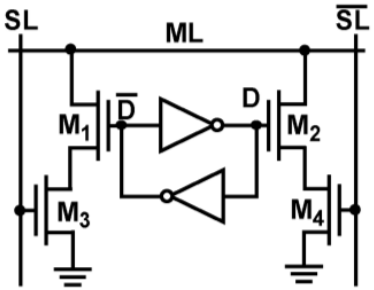
- A CAM cells both stores data and performs the XOR-like comparison
- Consider the SRAM-based CAM cell in the figure
  - Does not show the bitlines and wordlines for simplicity
- The four comparison transistors, M1 through M4, performs the matching and implements the pull-down path



Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality

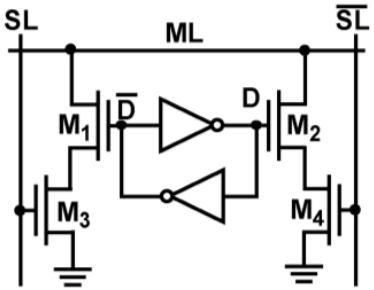
- A CAM cells both stores data and performs the XOR-like comparison
- Consider the SRAM-based CAM cell in the figure
  - Does not show the bitlines and wordlines for simplicity
- The four comparison transistors, M1 through M4, performs the matching and implements the pull-down path
- A match of SL and D disables both pull down path

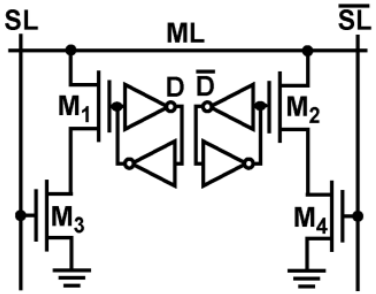


Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

CAM architecture and functionality

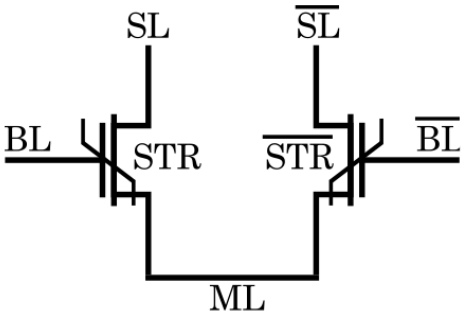
- A CAM cells both stores data and performs the XOR-like comparison
- Consider the SRAM-based CAM cell in the figure
  - Does not show the bitlines and wordlines for simplicity
- The four comparison transistors, M1 through M4, performs the matching and implements the pull-down path
- A match of SL and D disables both pull down path
- For ternary cells, we need to store two bits (3 states in total)





Pagiamtzis et al., "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IJSSC, 2006

NVM-based CAMs (e.g., FeFET based)

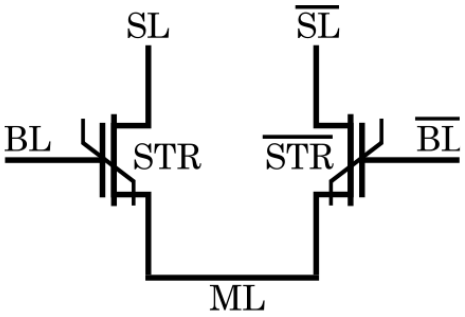


SL (V_D)	STR (V_{PGM})	ML (I_D)
0 V	-5 V	10.9 nA
0 V	+5 V	3.21 pA
50 mV	-5 V	2.79 pA
50 mV	+5 V	13.1 nA

$$ML = XNOR(SL, STR)$$

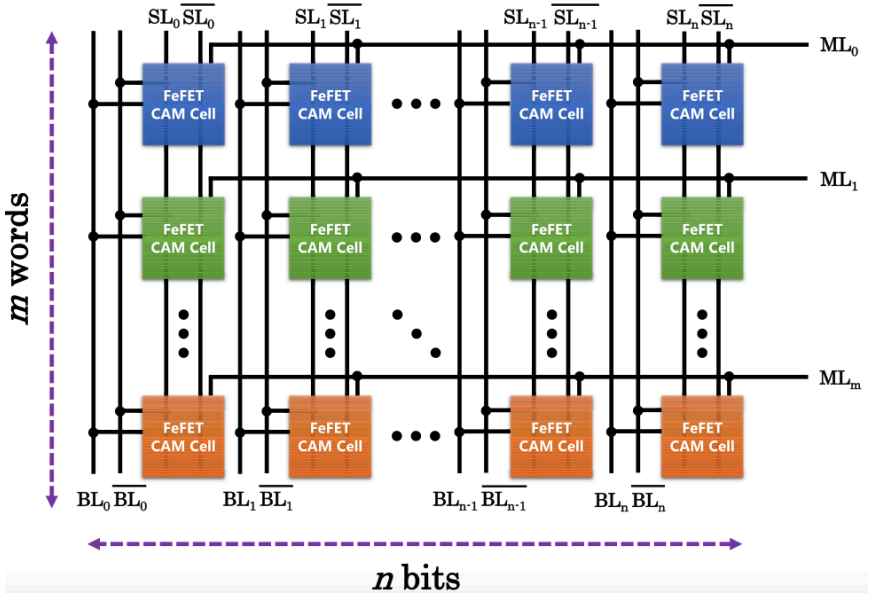
Tan et al., "Experimental Demonstration of a Ferroelectric HfO2-Based Content Addressable Memory Cell", IEEE EDL, 2020

NVM-based CAMs (e.g., FeFET based)



SL (V_D)	STR (V_{PGM})	ML (I_D)
0 V	-5 V	10.9 nA
0 V	+5 V	3.21 pA
50 mV	-5 V	2.79 pA
50 mV	+5 V	13.1 nA

$$ML = XNOR(SL, STR)$$



Tan et al., "Experimental Demonstration of a Ferroelectric HfO2-Based Content Addressable Memory Cell", IEEE EDL, 2020

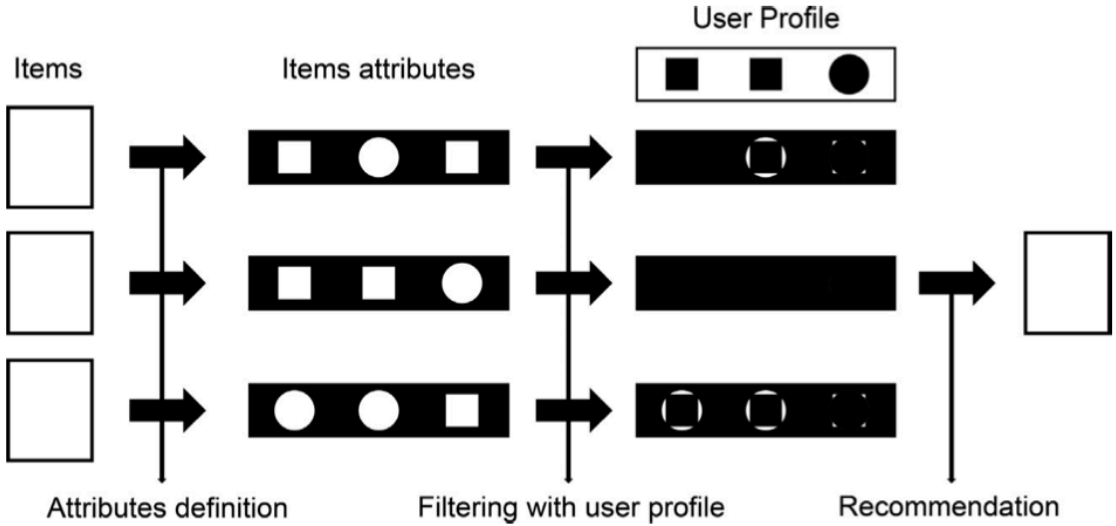
CAM applications

- CAMs can be used to implement applications from different domains

Bobadilla et al., “Recommender Systems Survey!“, Knowledge-based systemsL, 2013

CAM applications

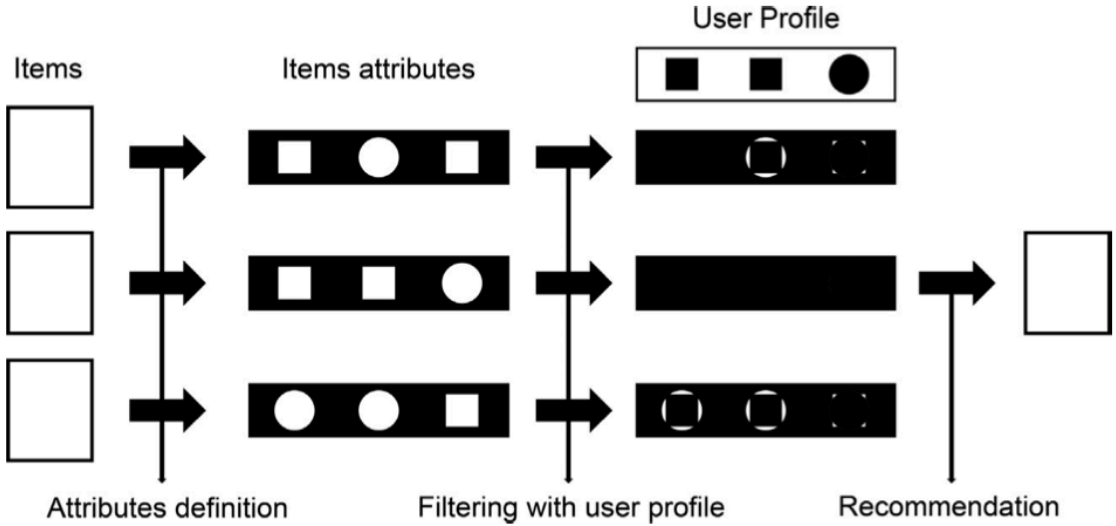
- CAMs can be used to implement applications from different domains



Bobadilla et al., "Recommender Systems Surveyl", Knowledge-based systemsL, 2013

CAM applications

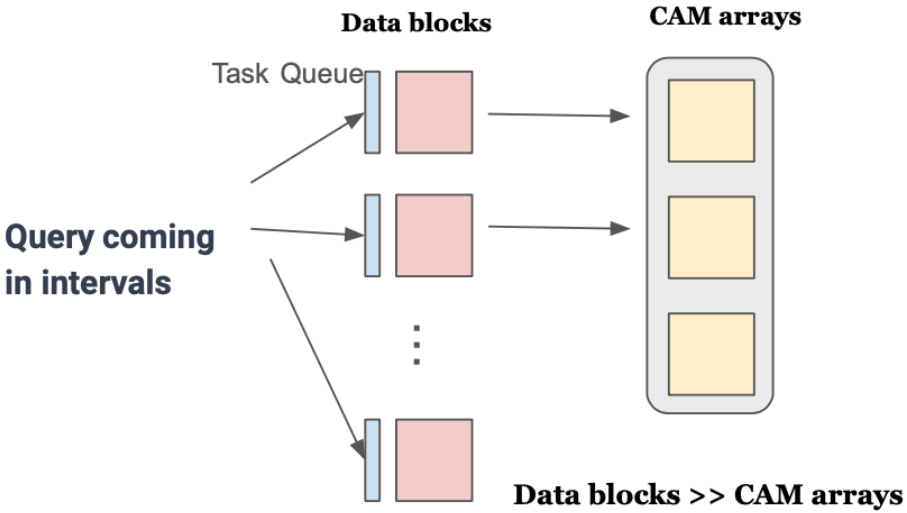
- CAMs can be used to implement applications from different domains



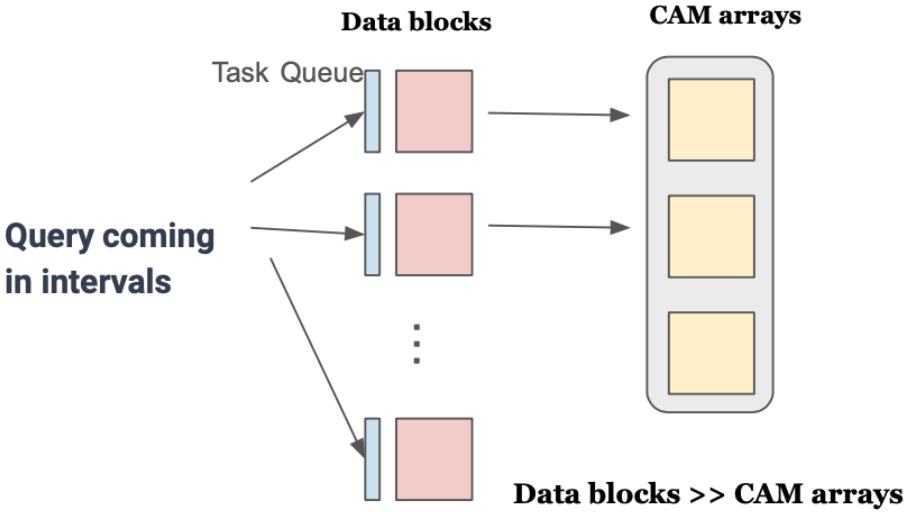
- Other applications include DNA comparison, K-NNs etc.

Bobadilla et al., "Recommender Systems Surveyl", Knowledge-based systemsL, 2013

Application mapping and scheduling on CAMs

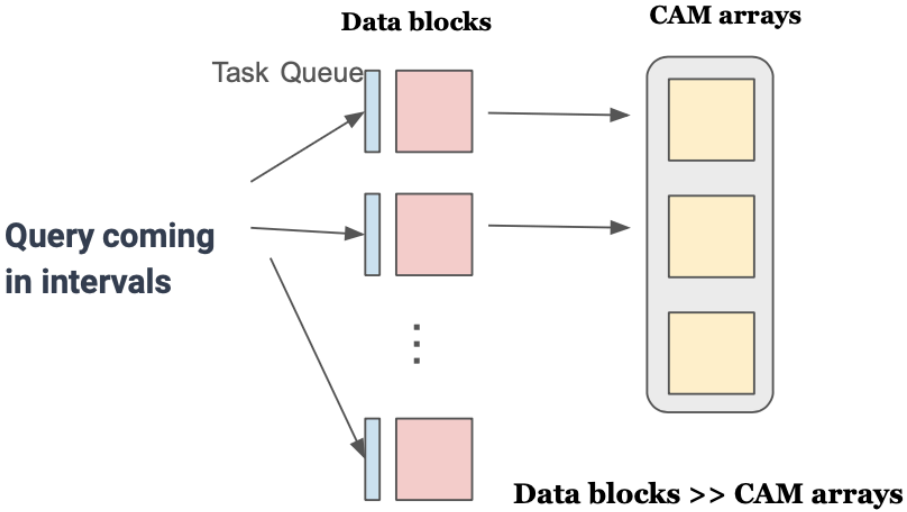


Application mapping and scheduling on CAMs



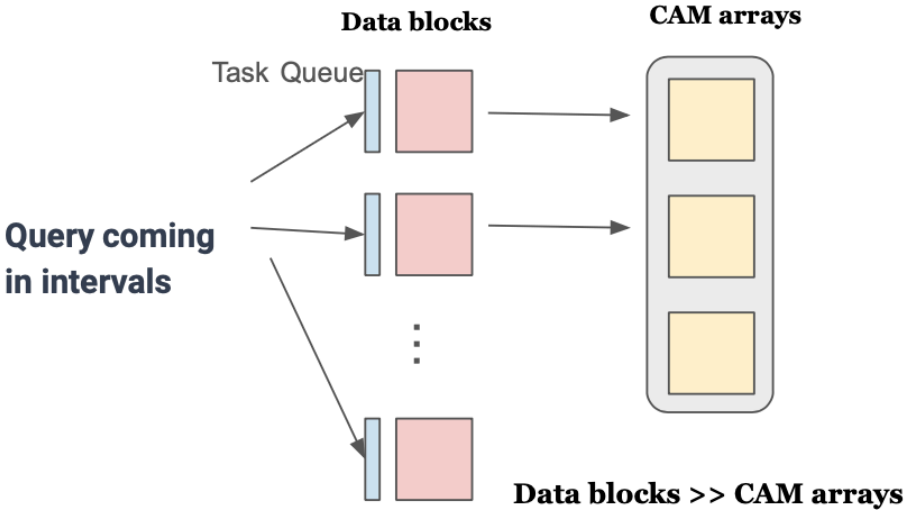
- The objective usually is to optimize for latency, throughput, and/or energy

Application mapping and scheduling on CAMs



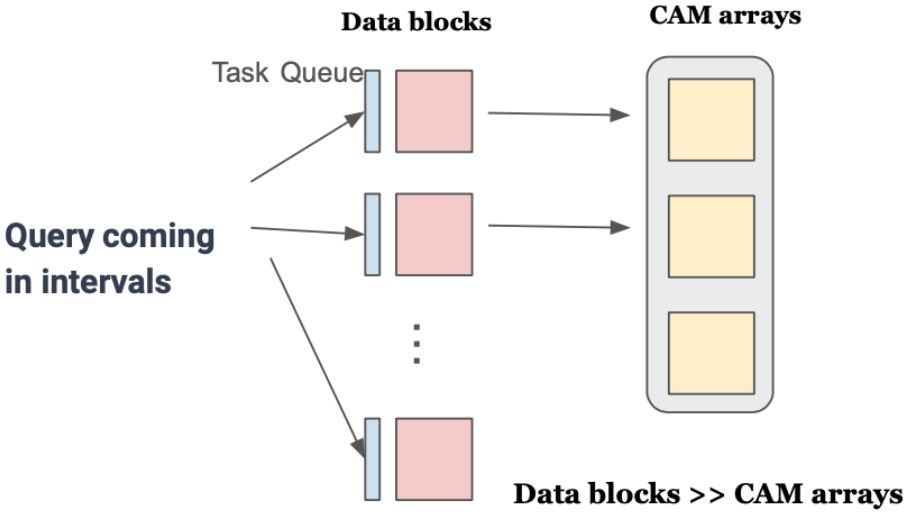
- The objective usually is to optimize for latency, throughput, and/or energy
- The best-mapping/scheduling should minimize the number of reloads to the CAM arrays

Application mapping and scheduling on CAMs



- The objective usually is to optimize for latency, throughput, and/or energy
- The best-mapping/scheduling should minimize the number of reloads to the CAM arrays
- In the worst case, every new query starts when the previous one finishes

Application mapping and scheduling on CAMs



- The objective usually is to optimize for latency, throughput, and/or energy
- The best-mapping/scheduling should minimize the number of reloads to the CAM arrays
- In the worst case, every new query starts when the previous one finishes
- Efficient handling of intermediate results is the trick here

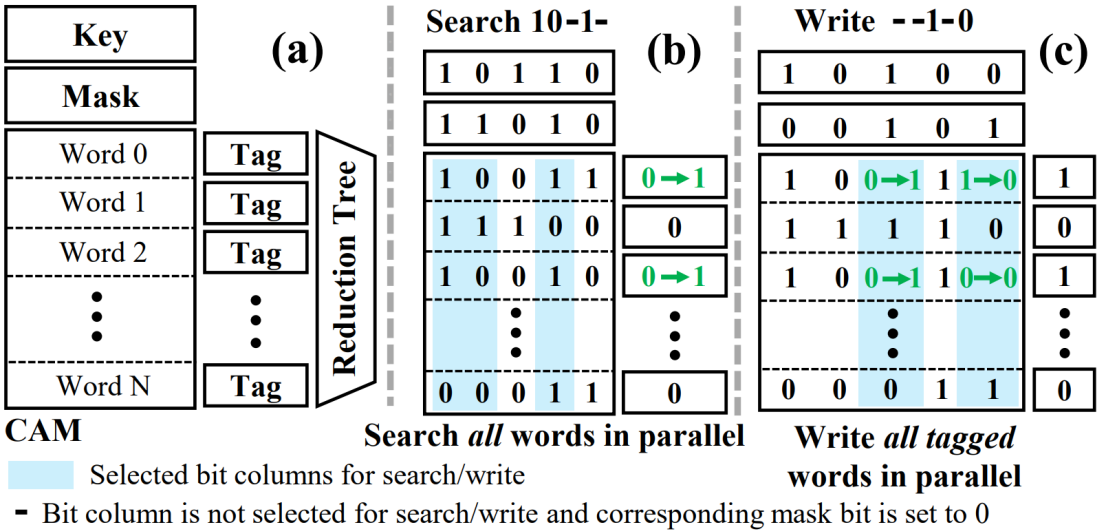
Associative processors

- Perform computations using CAM

Garzon et al., AM4: MRAM Crossbar Based CAM/TCAM/ACAM/AP for In-Memory Computing, IEEE JESTCS, 2023.

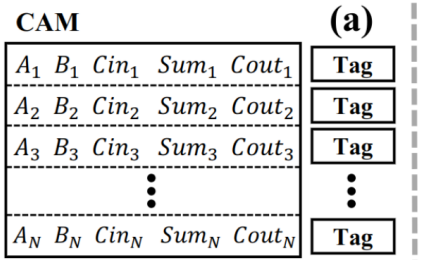
Associative processors

- Perform computations using CAM
- The idea is to replace computations using search-and-write operations



Garzon et al., AM4: MRAM Crossbar Based CAM/TCAM/ACAM/AP for In-Memory Computing, IEEE JESTCS, 2023.

Associative processors (Full adder)



(b)

Input Pattern	Computation Result
A=1, B=0, Cin=0	Sum=1
A=0, B=1, Cin=0	Sum=1
A=0, B=0, Cin=1	Sum=1
A=1, B=1, Cin=1	Sum=1
A=1, B=1	Cout=1
A=1, Cin=1	Cout=1
B=1, Cin=1	Cout=1

Search	100--	Input pattern: A=1, B=0, Cin=0
Write	---1-	Computation result: Sum=1
Search	010--	Input pattern: A=0, B=1, Cin=0
Write	---1-	Computation result: Sum=1
Search	001--	Input pattern: A=0, B=0, Cin=1
Write	---1-	Computation result: Sum=1
Search	111--	Input pattern: A=1, B=1, Cin=1
Write	---1-	Computation result: Sum=1
Search	11---	Input pattern: A=1, B=1
Write	----1	Computation result: Cout=1
Search	1-1--	Input pattern: A=1, Cin=1
Write	----1	Computation result: Cout=1
Search	-11--	Input pattern: B=1, Cin=1
Write	----1	Computation result: Cout=1

14 Operations (c)

Traditional AP

Search a single pattern in one search operation

Write is performed after each search operation

- Bit column is not selected for search or write

Garzon et al., AM4: MRAM Crossbar Based CAM/TCAM/ACAM/AP for In-Memory Computing, IEEE JESTCS, 2023.

Thank you!

asif.ali@uetpeshawar.edu.pk