DRAM-CAM: General-Purpose Bit-Serial Exact Pattern Matching using DRAM

Students: **Lingxi Wu** (Speaker), Rasool Sharifi
Advisors: Ashish Venkat, Kevin Skadron
University of Virginia

CRISP Annual Review Nov 2022
Student Deep Dive Theme 1

Accepted by: ***IEEE Computer Architecture Letters*** **21 (2), 89-92**

Background

Exact pattern matching is a widely used computation kernel:

Searching/matching patterns in a set of reference patterns

- Text/web processing
  - Enterprise Computing → ***String Match*** [1]
  - Web Indexing → ***Word Count*** [1]
- Image processing
  - Pixel ***Histogram*** [1]
- Data mining
  - Associative rule mining → ***Apriori*** [2]
- Embedded control system
  - Automotive → ***Bitcount*** [3]
- Bioinformatics
  - *Matching short DNA sequences*
  - Tremendous societal and economical benefit: $1.4B market for metagenomics

1. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system (IISWC '09)
2. MineBench: A Benchmark Suite for Data Mining Workloads (IISWC '06)
3. MiBench: A free, commercially representative embedded benchmark suite (WWC-4 '01)

CRISP

Problem Statement

Exact matching requires:

- key value stores → retrieve a payload based on a query (pattern)
- population count → count the number of matches

Software approach: lookup/hash tables

- Large memory footprint
- Random access
- Memory-bound

Hardware approach: content-addressable memory (CAM)

- SRAM- or NVM-based CAM is not area-efficient
- Power hungry
- Throughput is limited

Existing approaches are not efficient or scalable for large workloads

Our Proposal

Enabling **CAM** functionality inside **DRAM** to search/match patterns

DRAM offers:

- Smaller cells → potential **area** efficiency
- Low power intensity → potential **energy** efficiency
- Plenty parallelism → potential **throughput** advantage

Prior DRAM-based PIMs suggest good potential:

- Ambit (MICRO '17)
- DRISA (MICRO '17)
- ComputeDRAM (MICRO '19)
- Fulcrum (HPCA '20)
- SIMDRAM (ASPLOS '21)
- Sieve (ISCA '21) ⇐ **This work is an extension of Sieve**

CRISP

DRAM-PIM Flavors - TRA

Option one: charge-sharing, triple-row-activation (**TRA**)-based processing

Power intensive:

~22% additional power to raise additional WL

Long latency:

- Multiple operand row copy required due to special decoder
  - 3 row copy to setup R_{Ref}, R_{Query}, R_{Ctrl} and 1 row copy to WB R_{Result} per TRA (~340 ns)
- Exact matching require XNOR = 2 x TRA

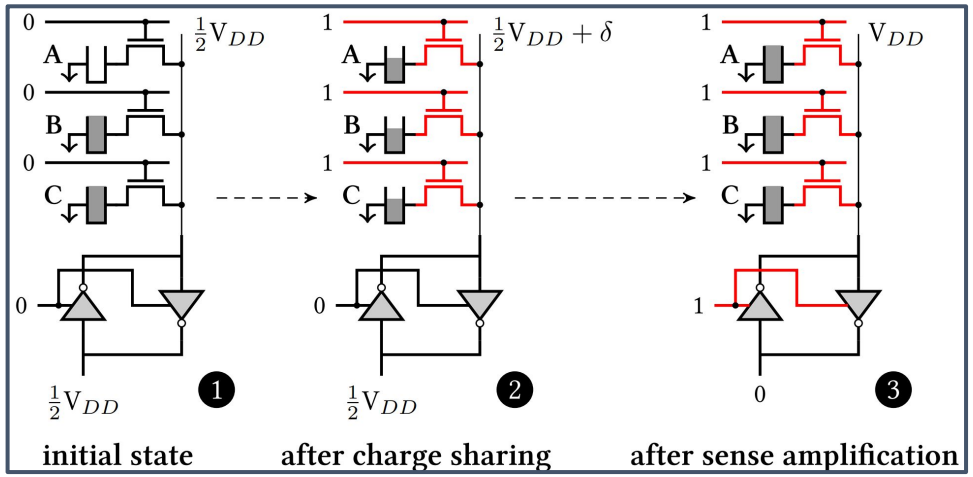


Fig. TRA processing in Ambit (MICRO '17)

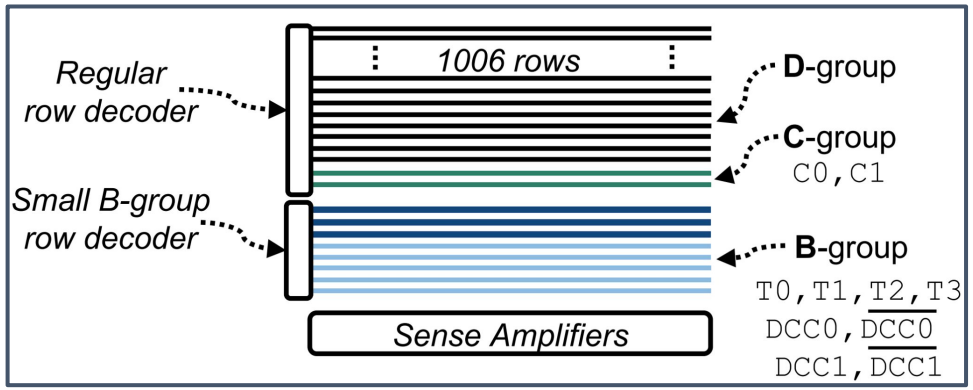


Fig. SubArray partition in SIMDRAM (ASPLOS '21)

CRISP

DRAM-PIM Flavors - Logic in SubArrays

Option Two: integrating **logic** at the local **row buffer** level after sense amplifiers

- Exact matching only requires **simple circuit**
- Operating on digital data, avoiding expensive charge-sharing TRA → **faster and less power hungry**

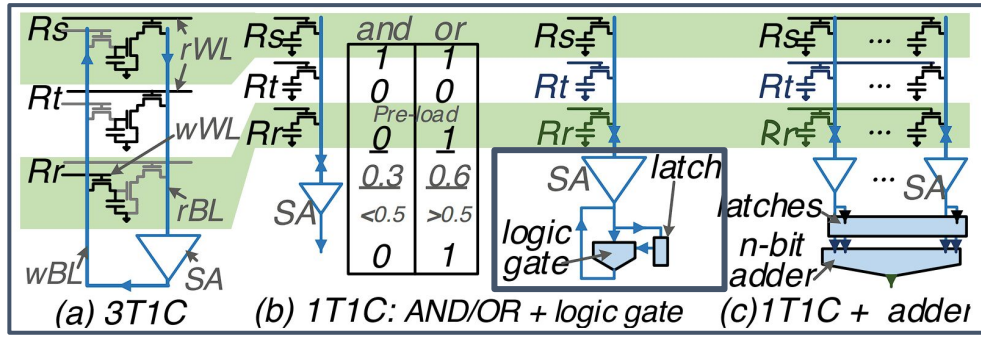


Fig. 1T1C + logic computing in DRISA (MICRO '17)

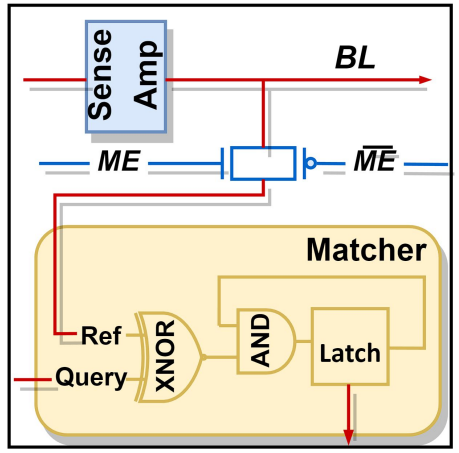


Fig. 1-bit exact matcher in Sieve (ISCA '21)

Option three: Two **SubArrays** share one **ALU**

Limited bandwidth: Processing a word at a time

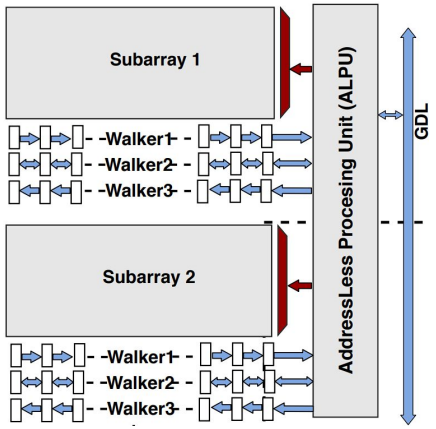


Fig. Fulcrum in-situ ALU (HPCA '20)

Option two achieves best **performance and overhead balance** for in-DRAM exact pattern matching

Data Layout Considerations

Horizontal data layout:

- Each pattern **spans** across multiple **columns** (BLs)
- Each row act brings **a set of patterns** to compare
- *Element-serial, bit-parallel*

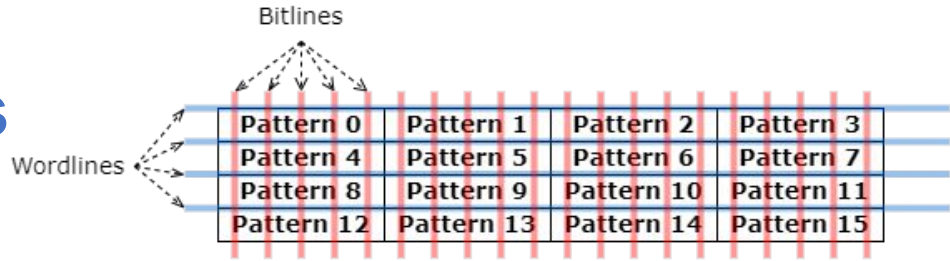


Fig. horizontal data layout

Vertical data layout:

- Each pattern **spans** across multiple **rows** (WLs)
- Each row act brings **a set of bits** of different patterns to compare
- *Element-parallel, bit-serial*

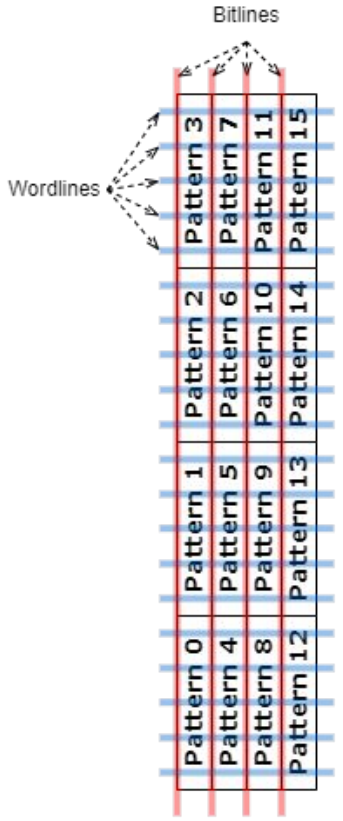


Fig. vertical data layout

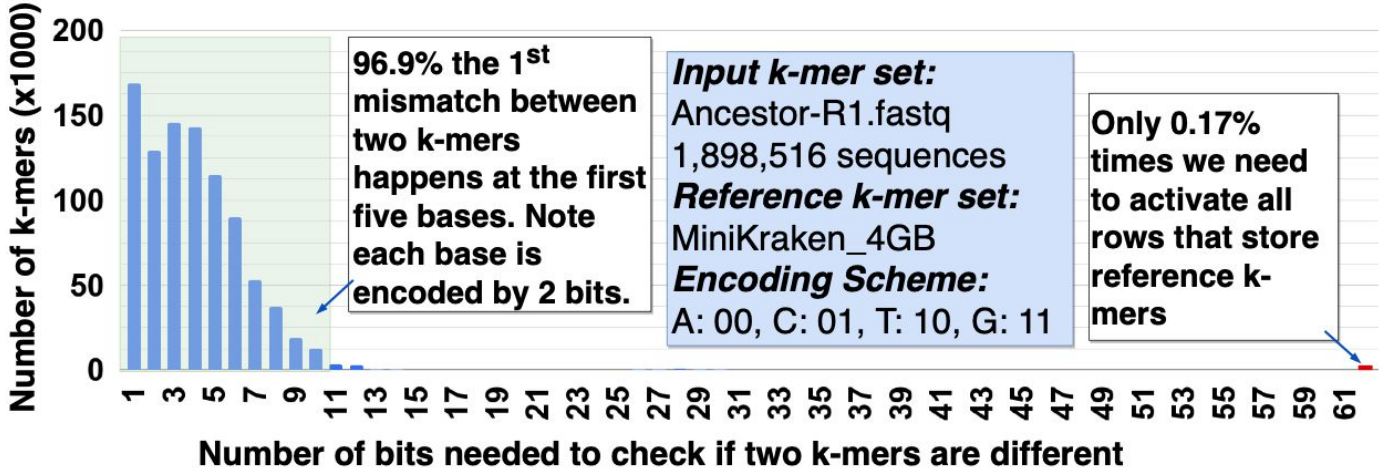
CRISP

Data Layout Considerations

DRAM-CAM adopts *element-parallel, bit-serial* (vertical) data layout

Parallelism: checking more patterns simultaneously

Early-termination: prune unnecessary row activations



For DNA sequence patterns, nearly 97% of the 1st mismatch is within the first 5 bases (**first 10 bits**)
No need to compare full pattern

Amortize query loading cost: for horizontal data layout, setting up a query pattern to search requires replicating it along the entire DRAM row

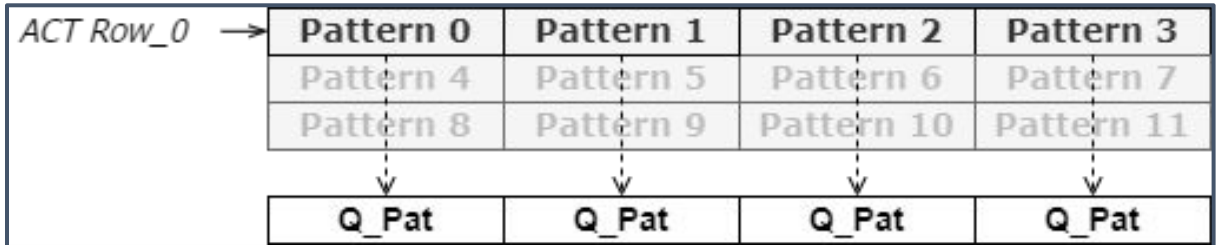


Fig. Query setup in horizontal data layout

CRISP

DRAM-CAM Architecture

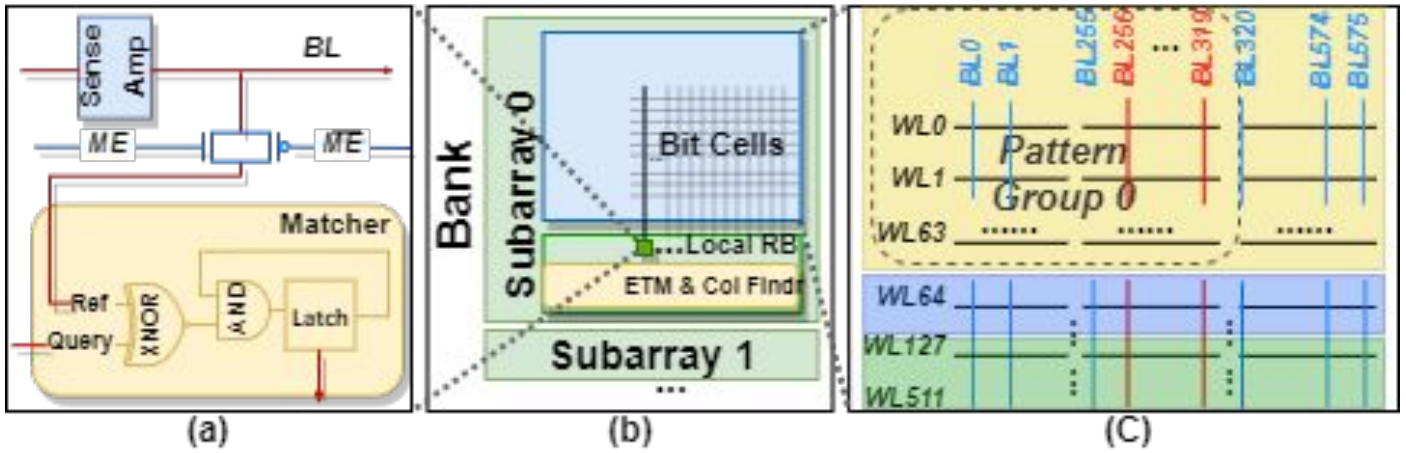


Fig. Overall DRAM-CAM architecture.

Overall Architecture:

- Leveraging **DRAM** technology and architecture
- Lightweight logic (**a**)
- Customized logic placed at the subarray's local row buffer (**b**)
- **Partitioned** cell arrays (**c**)
- **Subarray-level parallelism**

CRISP

Vertical Data layout

Interleave reference patterns with query patterns ⟶ **pattern group** (PG)

- Reference patterns in each PG are different
- Query patterns in each PG are the same

Motivations:

- Query patterns in all PGs are replaced in a batch ⟶ amortize cost
- long wire delay (WL). Need to reduce query bit fan-out latency

Operation:

- Each row_act brings a row of interleaved Q and R bits
- Q bit is delivered through a 1-bit bus within a PG

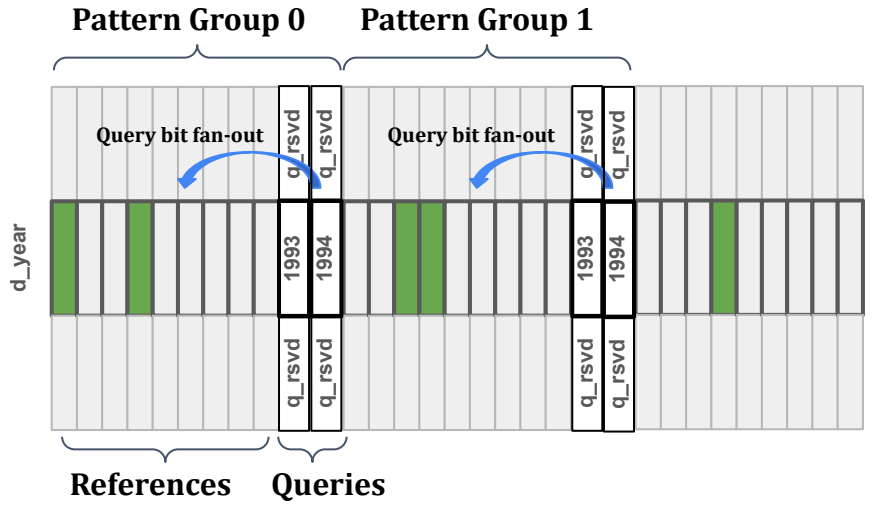


Fig. Subarray-level Data Layout. green = match

Note: DRAM-CAM also utilizes a hardware-based **Data Transposition Unit (DTU)** integrated on the controller to facilitate data layout

CRISP

In-situ Circuit

Row buffer-level Pattern matcher

- Minimalistic digital **matcher** after the sense amp
- Compares a reference pattern bit with a query bit
- **Query bit** arrive through a **1-bit wide bus**
- **Reference bit** arrive through **bitline**
- Stores the running **matching result** in the **1-bit latch**

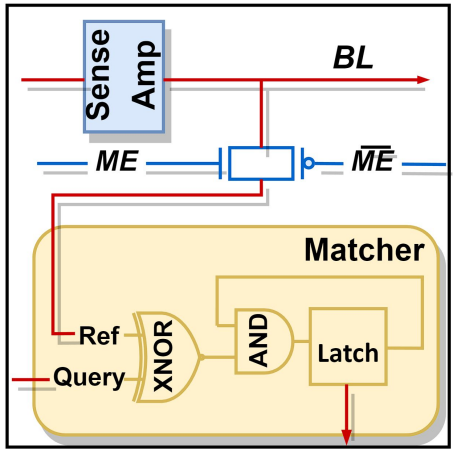


Fig. Bitline-level Matcher

Bank-level **Population Count Logic** → aggregate number of matches

- Aggregate every 64-bit using either LUT or Compressor
- Accumulate into a small register
- Controller aggregate the final result

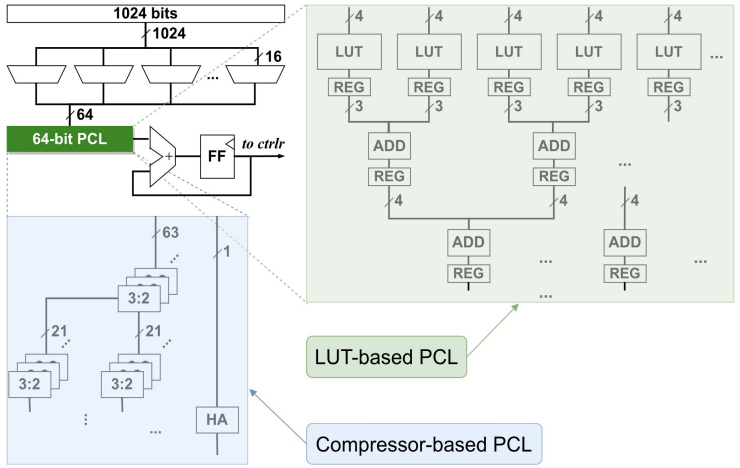


Fig. Bank-level Population Count Logic

CRISP

Optimizations

Hardware: Chip-level Parallelism (CLP)

- A dedicated chip select (CL) wire/chip
- Decouple query dispatch (**Q_W**) and matching operation (**Pat_Match**)

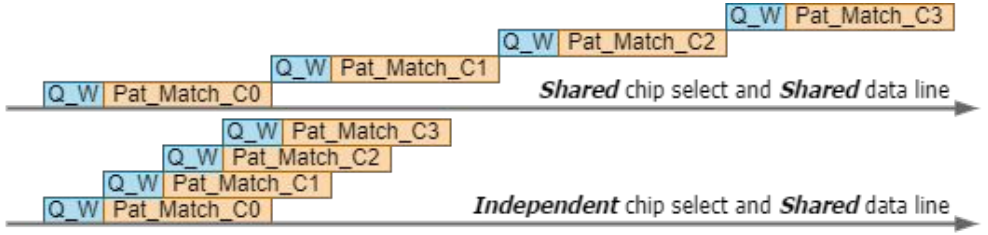


Fig. Chip-level Parallelism Benefit

Runtime Optimizations: To fully leverage the parallelism inside DRAM

Pattern Distribution (**PD**): Distribute patterns to as many subarrays across different banks/ranks/channels as possible

Pattern Replication (**PR**): Clones pattern data multiple times to saturate the capacity of DRAM-CAM → improves concurrency and throughput → more subarrays are processing

Note: Optimizations are not always applicable to every workload:

e.g., workloads with large working set cannot leverage PR and/or PD

CRISP

Case Study - Histogram

Histogram [1]:

Counts the frequencies of distinct pixel values in the red, green, and blue channels of a bitmap image. Each pixel value is stored as a 24-bit binary stream (8-bit per RGB component)

Execution Steps Leveraging the DRAM-CAM:

1. Load the input image file to the **Data Transposition Unit** (**DTU**) on the DRAM-CAM controller
2. Stream the transposed binary pixel into **DRAM-CAM array**
3. Write every **distinct pixel pattern** from 00000000 to 11111111 to DRAM-CAM arrays (query region)
4. Initiate matching requests to DRAM-CAM arrays
5. For each query, aggregate hits using the **Population Count Logic** (**PCL**), and return hits

1. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system (IISWC '09)

CRISP

Evaluations

Application	Benchmark	Domain	Input
String Match (SM)	Phoenix [8]	Text processing	Key and encrypted files.
Histogram (HG)	Phoenix [8]	Img processing	1.4GB bitmap image
Word Count (WC)	Phoenix [8]	Text processing	100MB text file
Bitcount (BC)	MiBench [9]	Automotive	1,125,000 integers
Apriori (AP)	MineBench [10]	Associative rule mining	7,993,604 transactions, 1,000 unique items

Workloads: 5 kernels from 3 benchmark suites

Systems:

- CPU
- **1** unoptimized (**UNOPT**) DRAM-CAM
- **3** optimized DRAM-CAM w/ pattern distribution (**PD**), pattern replications (**PR**), chip-level parallelism (**CLP**)
- **6** different prior DRAM-based PIM setups

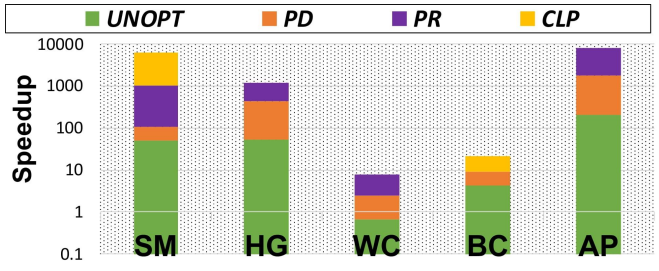


Fig. Speedup over CPU

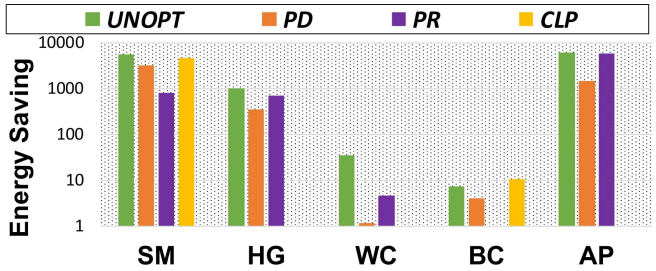


Fig. Energy saving over CPU

Compare to CPU:

Unoptimized DRAM-CAM:

62.99X/207.98X avg/max speedup

2552.75X/6174.77X avg/max energy saving

W/ Optimizations:

2467.09X/6217.42X avg/max speedup

1416.55X/5888.30X avg/max energy saving

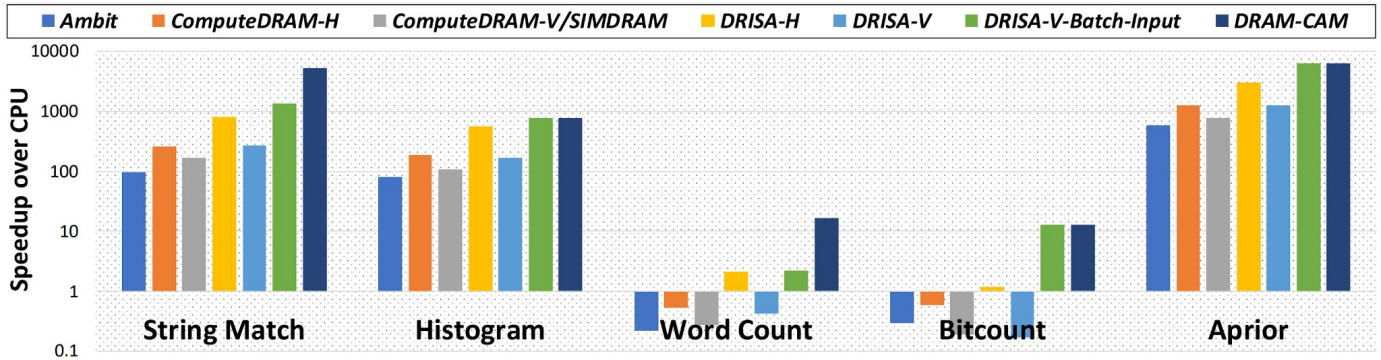


Fig. Speedup over other DRAM PIM

Compared to other DRAM-PIM configs:

Thoroughly compared to various combinations of setup TRA + H/V, Digital Logic + H/V, etc.

DRAM-CAM wins by employing faster comparison mechanism and a more efficient way of setting up queries

CRISP

JUMP

Joint University Microelectronics Program

www.src.org/program/jump

Semiconductor Research Corporation

@srcJUMP