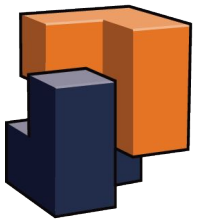


CRISP
Center for Research on Intelligent Storage and Processing in Memory

Sieve: Scalable In-situ DRAM-based Accelerator Designs for Massively Parallel k-mer Matching

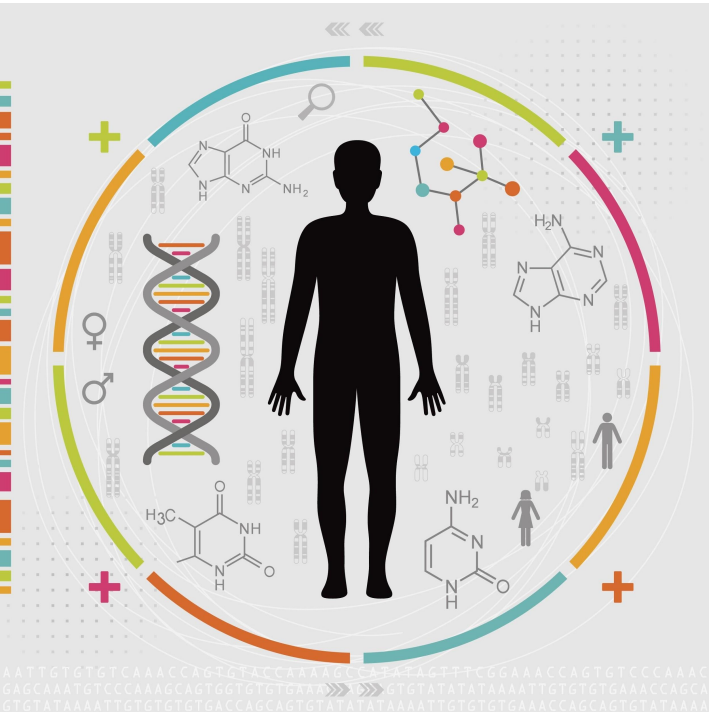
Lingxi Wu, Rasool Sharifi, Marzieh Lenjani, Kevin Skadron, Ashish Venkat
University of Virginia

ISCA 2021
key words: **Processing-in-memory**, **bioinformatics**, **accelerator design**

Background

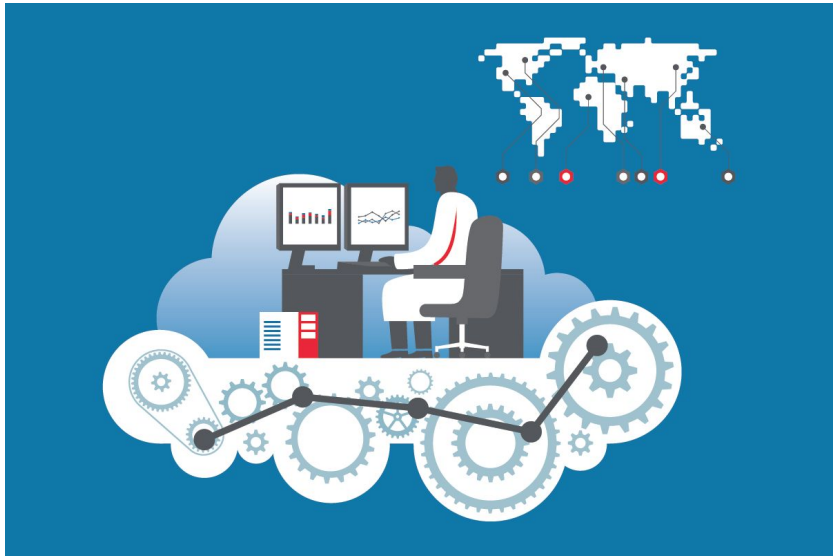
CRISP

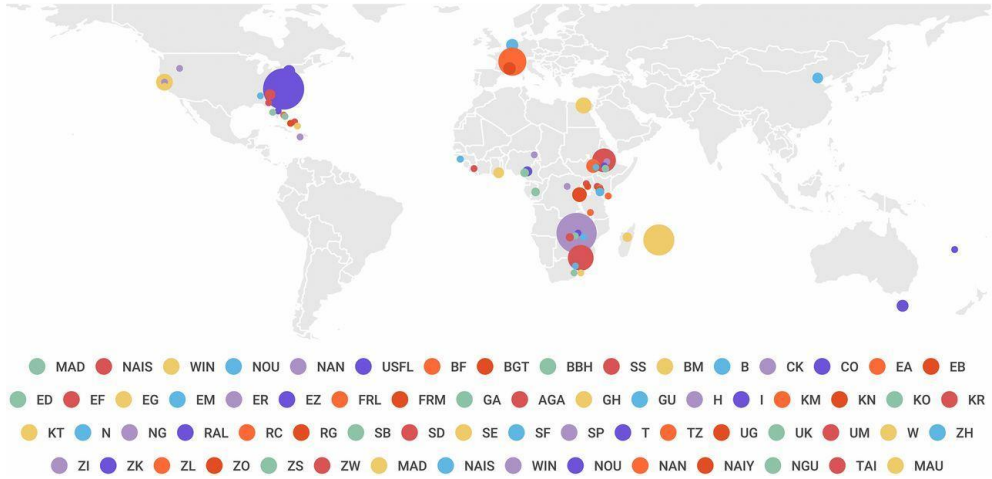
Bioinformatics



Precision Medicine

Disease Surveillance

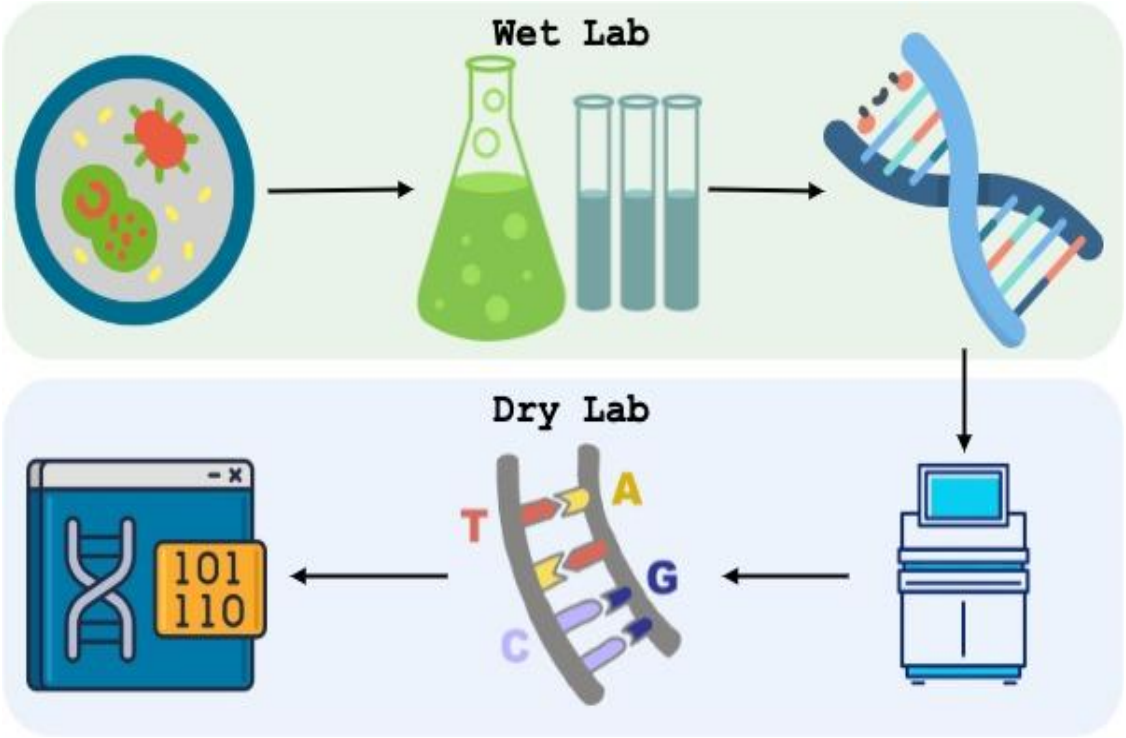




Population Genetics

CRISP

Bio-sequence Search & Classification

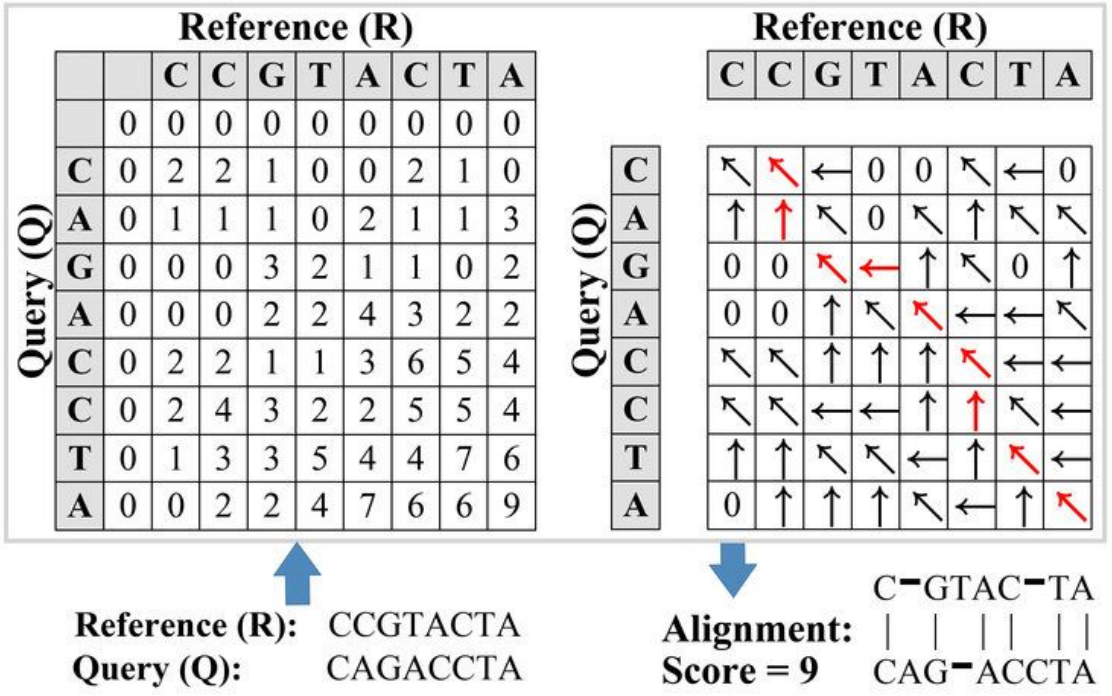


Sequences search & classification:

Identify the **origin** and **functionality** of a DNA sequence.

Traditional approach:

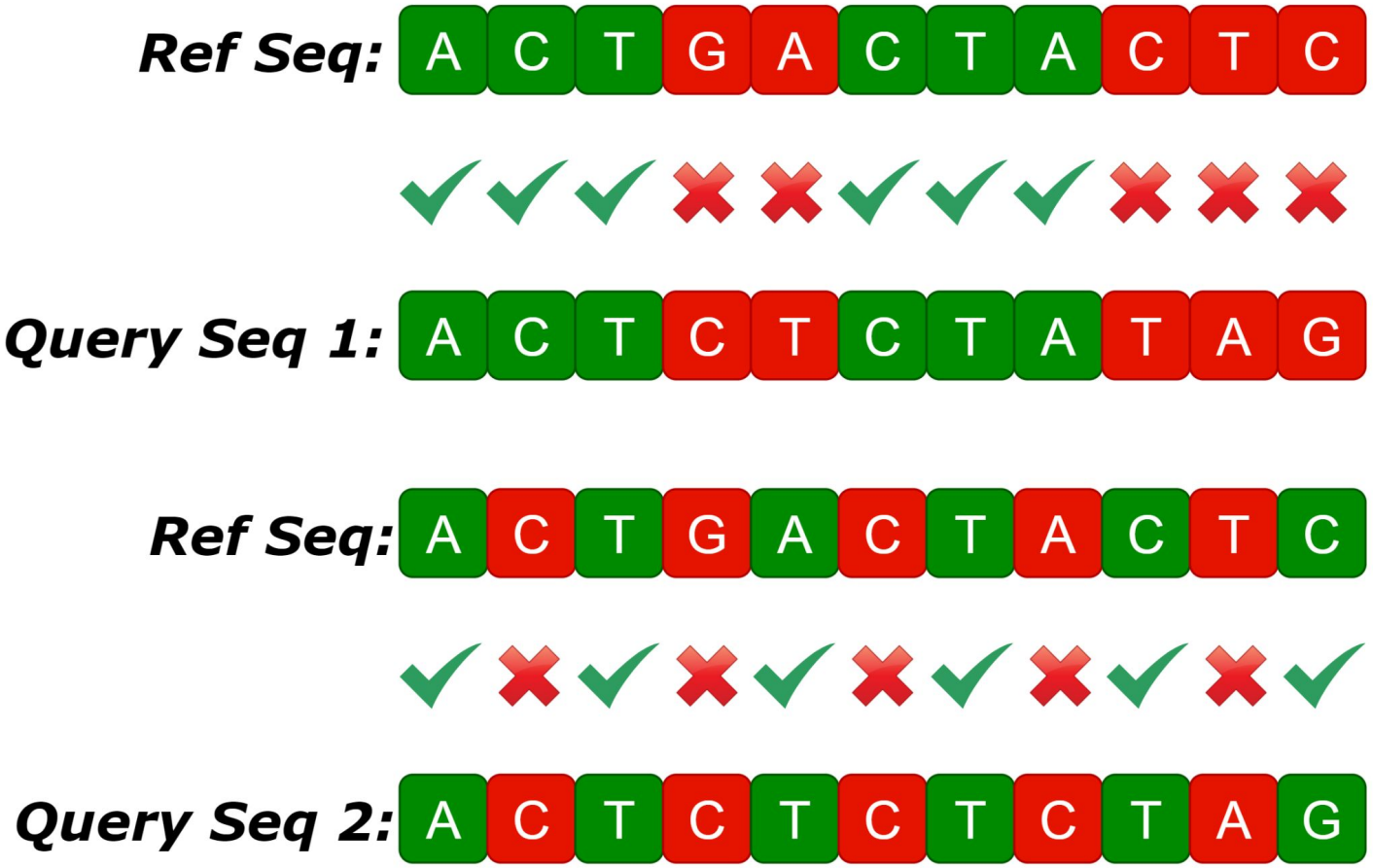
- Sequence alignment
- **Computationally-intensive** **$O(N^2)$**



DNA Sequencing:

Extract raw DNA sequences from living organisms.
Represent them with series of **A**,**C**,**G**,**T**.

K-mer Matching to the Rescue



Query sequence 1 is more likely to be derived from the ***Reference Sequence*** through the process of genetic mutation than ***query sequence 2***

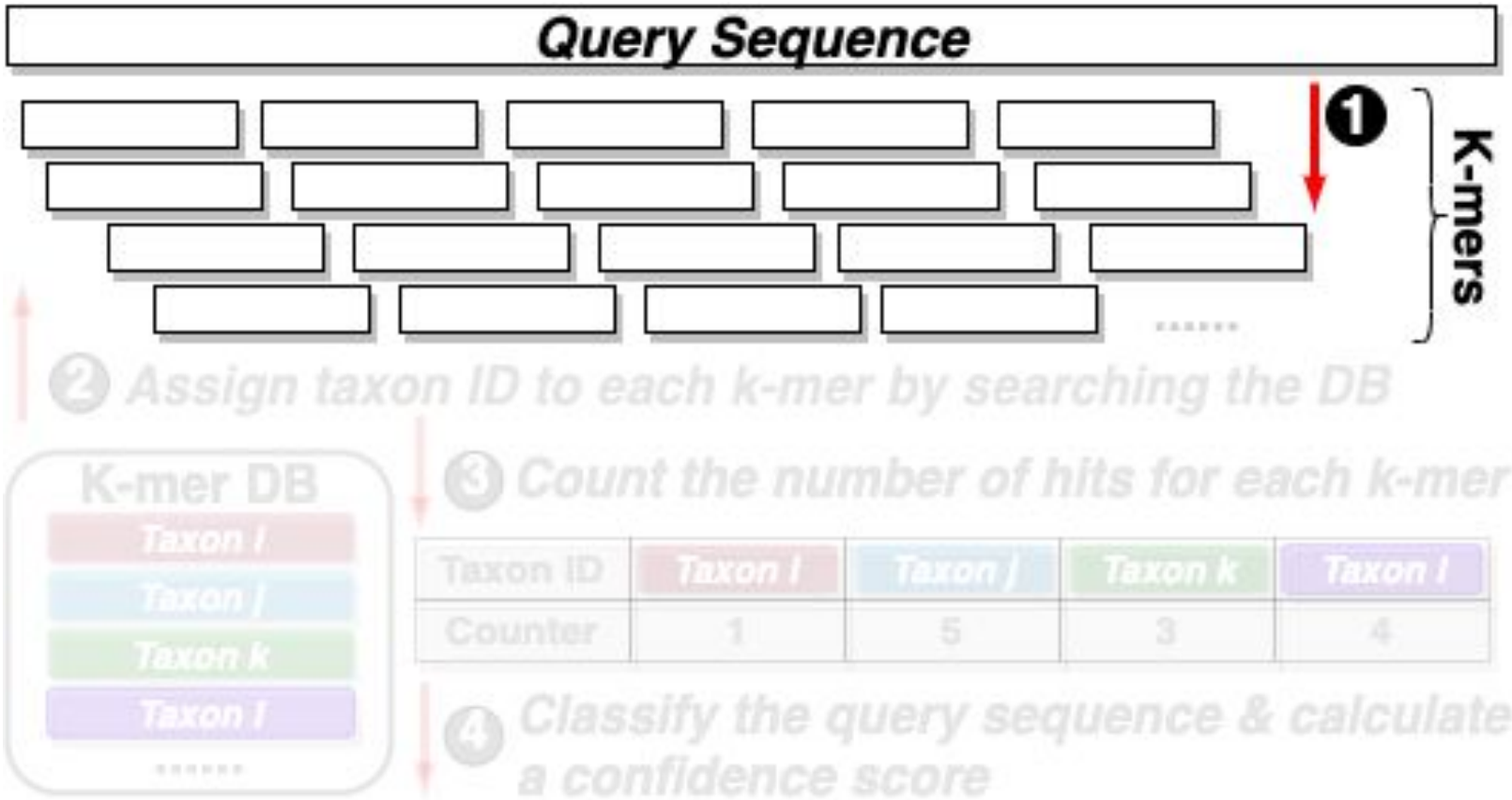
***K*-mers:** Subsequences of size K

Intuition:

1. Biologically correlated sequences share many short lengths of **exact matches**
2. Alignment is not always the best way to capture structural similarity and divergence
3. **Computationally cheap**. Exact match is faster than alignment. (**1000x** faster!)

CRISP

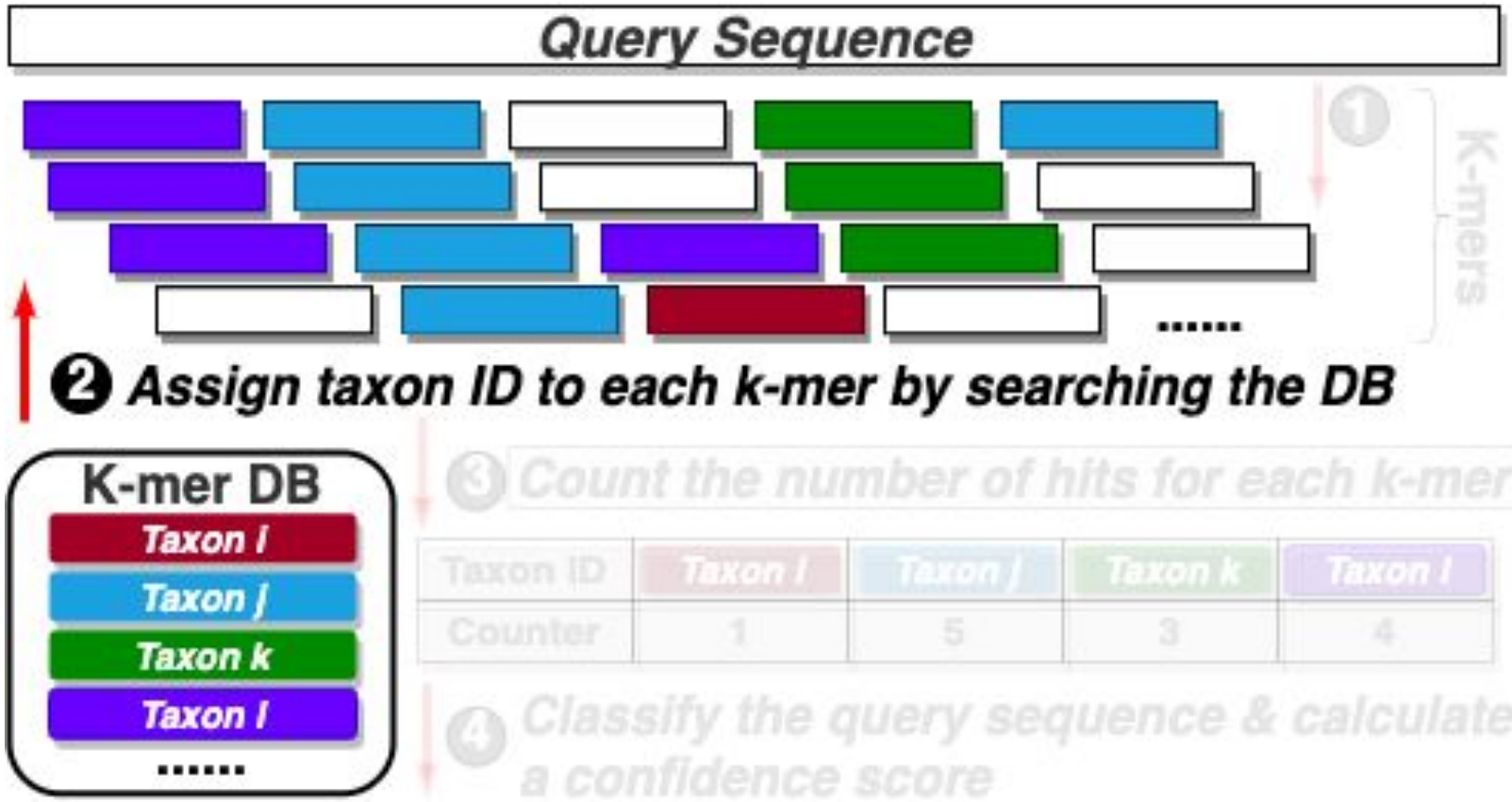
CLARK: *K*-mer Matching Illustration



Each **query** sequence is converted to a set of **k-mers** (default to overlapping 31-mers.)

CRISP

CLARK: *K*-mer Matching (a bottleneck stage)



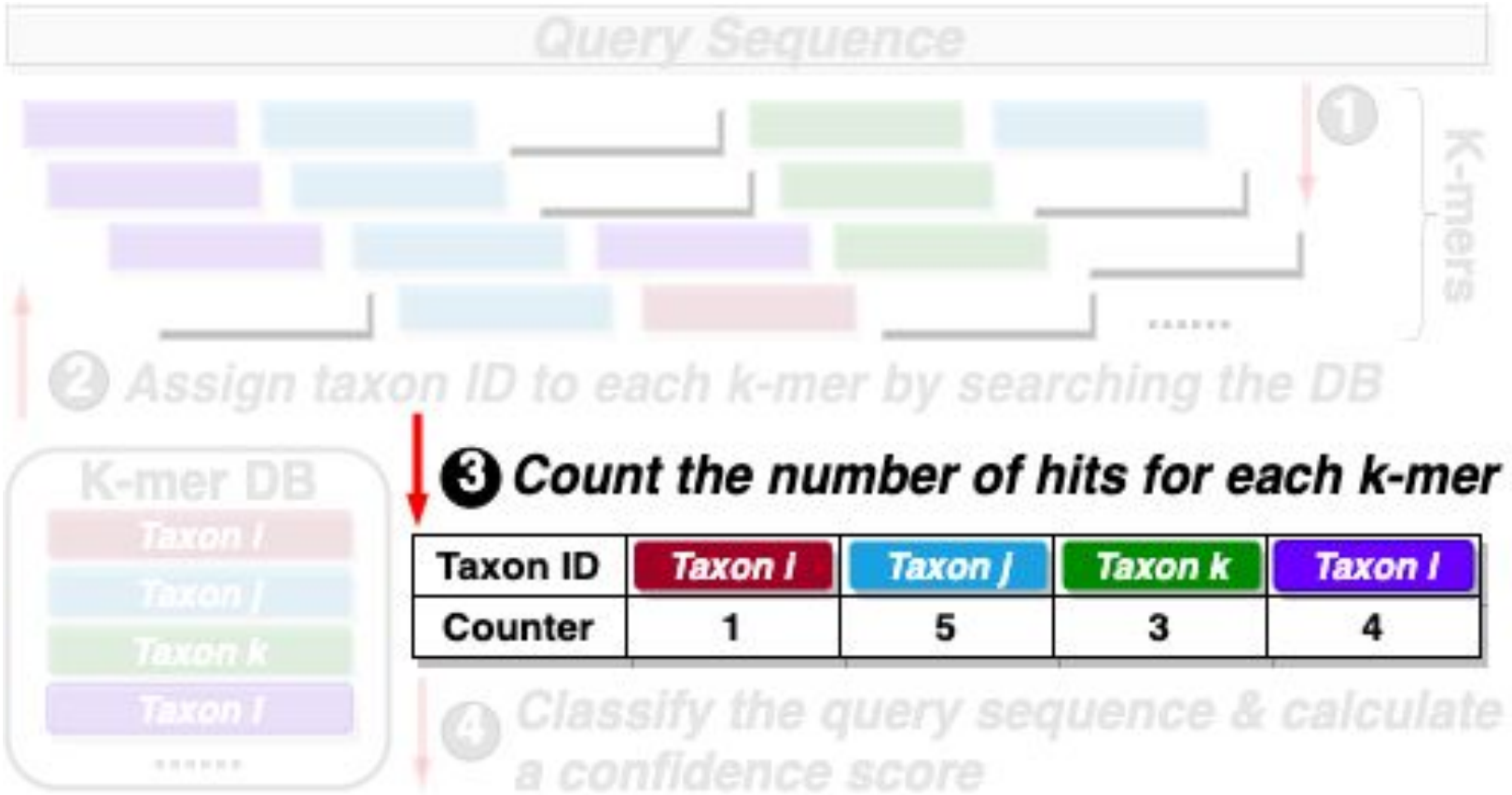
Each k-mer is assigned with a **taxon label** by querying the in-memory database.

Different taxa are color-coded in this Figure and k-mers with no hits are indicated in white.

Sieve's goal is to accelerate this step, which is the bulk of the workload

CRISP

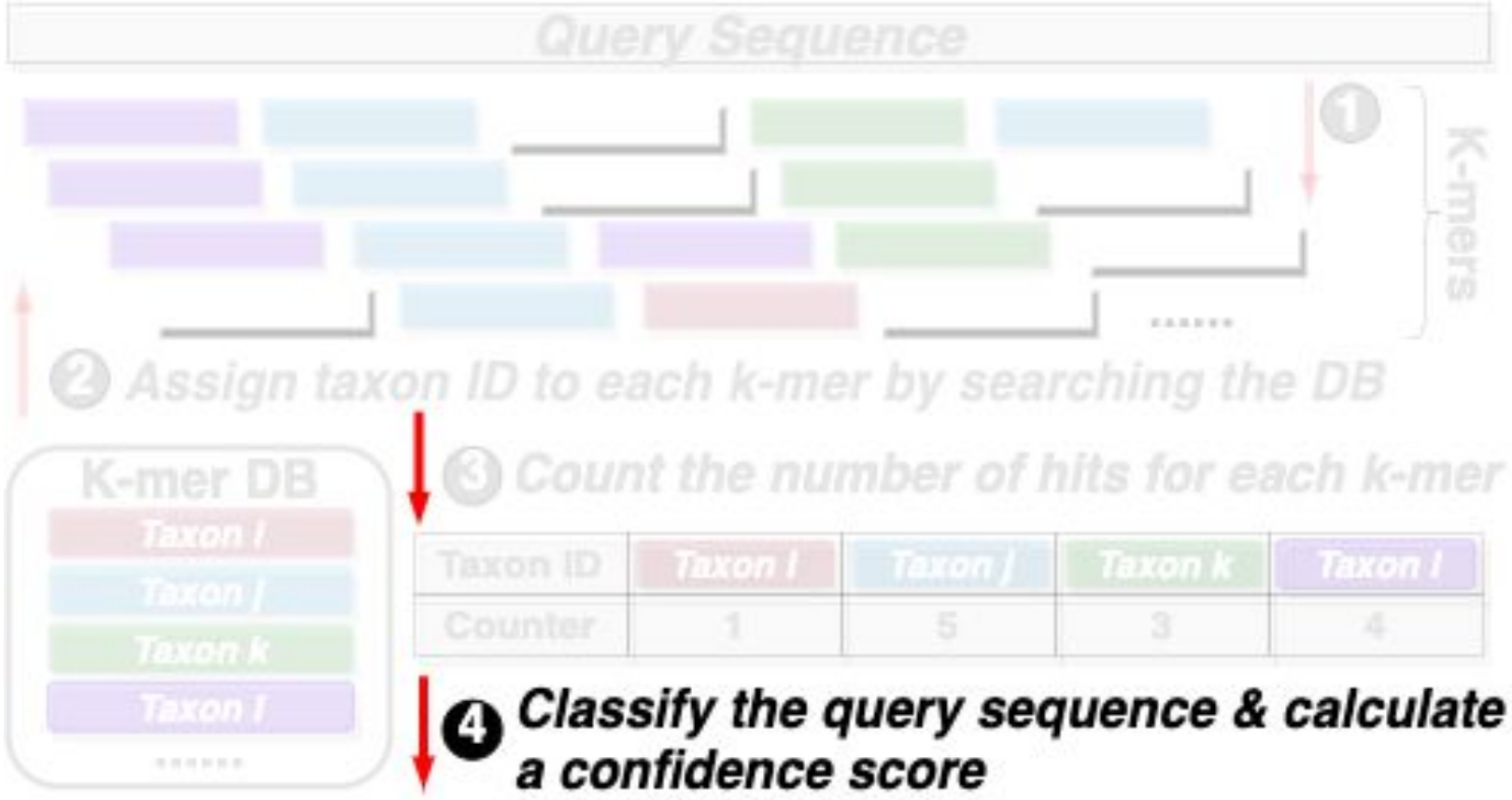
CLARK: Post *K*-mer Matching



The number of hits for each taxon label is accumulated

CRISP

CLARK: Post *K*-mer Matching



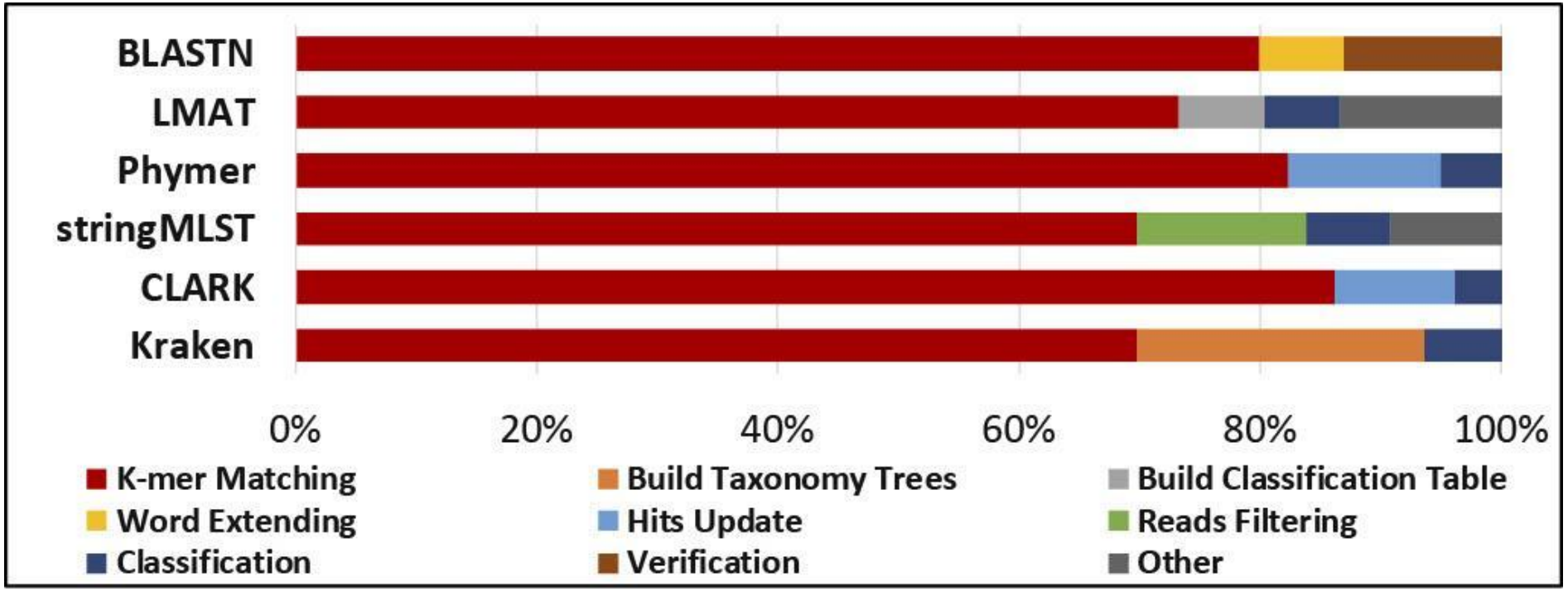
The taxon ID with the most hits is used to **classify the query sequence**

CRISP

Motivation

CRISP

K-mer Matching Becomes a Bottleneck



- *K*-mer matching sits on the **critical paths** of many genome analysis pipelines
- Modern sequencers generate bio data at a rate **surpassing Moore's Law** (more than Twitter, Youtube, astrophysics, etc.)
- Precision medicine:
  - Sequencing **10 TBs** of microbiome and DNA/RNA data → **48 hours**
  - Metagenomics stage (Kraken) → **68 days**

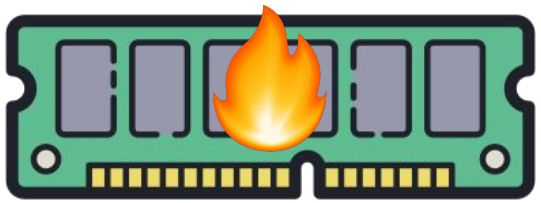
CRISP

Recent Bio Acceleration Efforts

Project	Domain	Year	Conference	Architecture
GenAx (Fuijiki et al.)	Reference-guided assembly	2018	ISCA	ASIC
Darwin (Turakhia et al.)	Reference-guided assembly	2018	ASPLOS	ASIC
INDEL Realignment (Wu et al.)	Insertion deletion realignment	2019	HPCA	FPGA. Cloud
Darwin-WGA (Turakhia et al.)	Whole genome alignment	2019	HPCA	ASIC
MEDAL (Huangfu et al.)	DNA seeding (for alignment)	2019	MICRO	Processing-in-memory
GenCache (Nag et al.)	Sequence alignment	2019	MICRO	Processing-in-memory
GenASM (Damla et al.)	Sequence alignment	2020	MICRO	Processing-in-memory
Genesis (Weng et al.)	Variant calling	2020	ISCA	FPGA. Cloud
SeedEx (Fuijiki et al.)	Read alignment	2020	MICRO	ASIC
EXMA (Jiang et al.)	DNA seeding (for alignment)	2021	HPCA	Processing-in-memory

- Focus on well-studied areas of bioinformatics (i.e., ***alignment-based*** sequence analysis)
- Lack of acceleration effort for ***alignment-free*** or *K*-mer based analysis pipelines
- **Processing-in-memory** is gaining traction

Workload Analysis



K-mer matching is memory-intensive:

- **Poor cache locality** (hash table access, linked list traversal)
  - Optimization: utilizing *minimizer*, which indexes *K*-mers with the same "signature" into the same "bucket"
  - ~ **8%** of consecutive k-mers are indexed into the same bucket → fetch buckets repeatedly
- **Coarse-grained cache line** → bandwidth wastage
- **Low computation intensity** → unable to mask data access latency

Increase system DRAM bandwidth on a multi-core workstation?

- DRAM bandwidth is under-utilized due to limited MSHRs
- Even with enough MSHRs → needs **215** Broadwell CPU cores!
  - 40 ns avg memory latency and all loads are served concurrently
  - All memory loads are served concurrently

CRISP

PIM is Suitable for *K*-mer Matching

PIM is good for memory intensive applications due to increased **bandwidth**, reduced **access latency** and **data movement**, improved **scalability**

- Graph processing: **Tesseract** (ISCA 15'), **Graphp** (HPCA 18'), etc.
- Data analytics: **Mondrian Data Engin** (ISCA 17'), etc.
- Neural network: **ISAAC** (ISCA 16'), etc.

The bulk of the *K*-mer matching workloads is **exact pattern matching** → **simple** computation logic (i.e., bitwise operations)

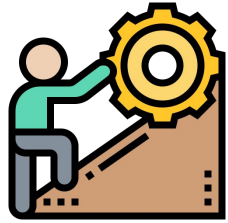
- Can even work with the most aggressive form of PIM: **in-situ: processing at DRAM row buffers**
- **DRISA** (MICRO 17'), **Ambit** (MICRO 17'), **SCOPE** (MICRO 18'), **Fulcrum** (HPCA 20')...

K-mer matching benefits from high data parallelism

- Memory system generally have a deep hierarchy that provides ample parallelism
- **SALP** (ISCA 12') enabled DRAM subarray-level parallelism
- **LISA** (HPCA 16') enabled inter-subarray data movement in DRAM

CRISP

Challenges of in-situ DRAM Acceleration



Area overhead v.s. **Accelerator capacity**: The closer to the actual bits, the more expensive the hardware cost becomes

- Sense amps in row buffers are laid out in a **pitch-matched** manner
- **Limited** metal layer
- State-of-the-art (DRISA) is **half as dense**
- **Bio** accelerators **demands** high **capacity**

Parallelism & bandwidth v.s. **Computation complexity**

- Bandwidth at the row buffer is 10^6**X** higher
- Energy for data access is 10^3**X** lower
- Restricted to **non-flexible** & **simple** row-wide operations
  - Li et al., Seshadri et al., and Marzieh et al. show that in-situ row-wide bitwise computing can be more expensive than CPU

Prior in-situ accelerators do not fully unleash the potential of in-situ k-mer matching, due to **inefficient** data mapping and pattern matching mechanism

Key Ideas

CRISP

Sieve Key Ideas

Design a minimalistic matcher circuit at bitline level

- Comparator: **XNOR** gate
- Accumulator: 1-bit **latch** and an **AND** gate
- Rely on DRAM **row-activation** for query and reference **pattern comparison**

Explore the trade-off of placing such circuits at different DRAM level

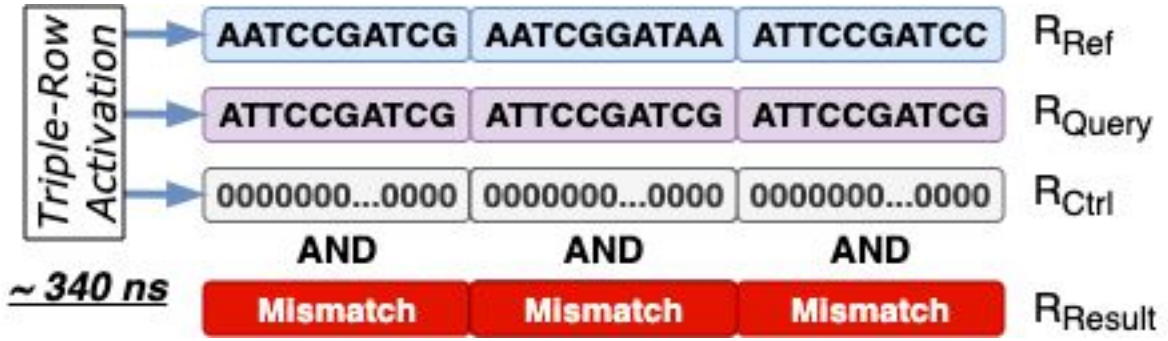
- From the **I/O interface** of the DRAM chips (Type-1) to the **local row buffer** of each subarray (Type-3)
- Each architecture incrementally adds extra hardware complexity to unlock more performance benefits

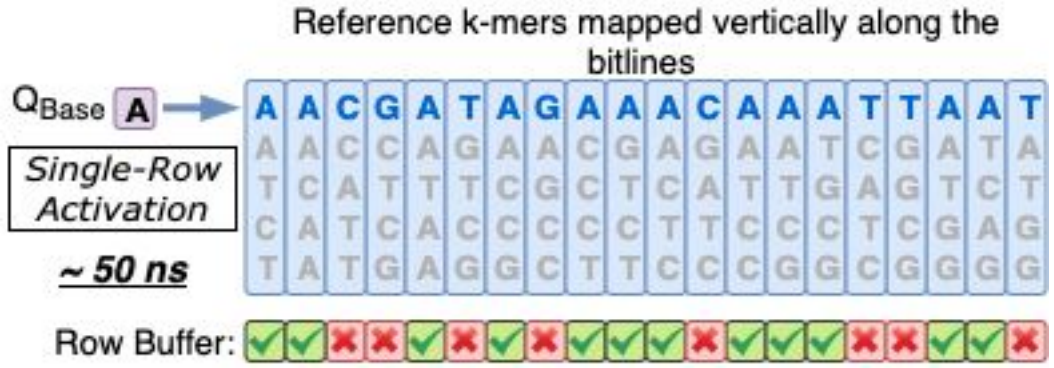
★ Transpose the data layout and provide an Early Termination Mechanism (ETM) to prune unnecessary DRAM row activation

- Element-parallel, bit-serial
- Matching k-mers has no data dependency
- K-mer hit rate is relatively low → abandon unnecessary row activation ASAP

We propose **Sieve**: an in-situ k-mer matching accelerator leveraging DRAM technology. (**Not a memory product**)

Data Layout and Pattern Matching Mechanism





K-mer pattern matching in previous in-situ accelerators using **Triple-row Activation (TRA)** and **horizontal data layout**

2 bits / a DNA base (A: 00, C: 01, T: 10, G: 11), k = 31, DRAM row width = 8192 bits → **128 k-mer patterns / row**

1. Copy 128 different reference patterns to R_{Ref}
2. Copy 128 copies of the same query into R_{Query}.
3. Copy 0s to R_{Ctrl} with 0s from a preset row
4. TRA
5. Copy result bits to row R_{Result}

- **Latency**: 8 *tRAS(~35 ns/tRAS) + 4 *tRP(~15 ns/tRP) = **~ 340 ns**
- **Energy consumption**: raising each additional wordline increases the activation energy by **22%**

K-mer pattern matching in Sieve using **single-row activation** and **vertical data layout** (AKA. element-parallel, bit-serial)

Sieve compares a query with a **more extensive** set of references (~ **8192 k-mer patterns**) in a **shorter** time window (**~50 ns**), but progresses only one bit at a time

Latency: 1 *tRAS+ 1*tRP= ~ **50 ns**

CRISP

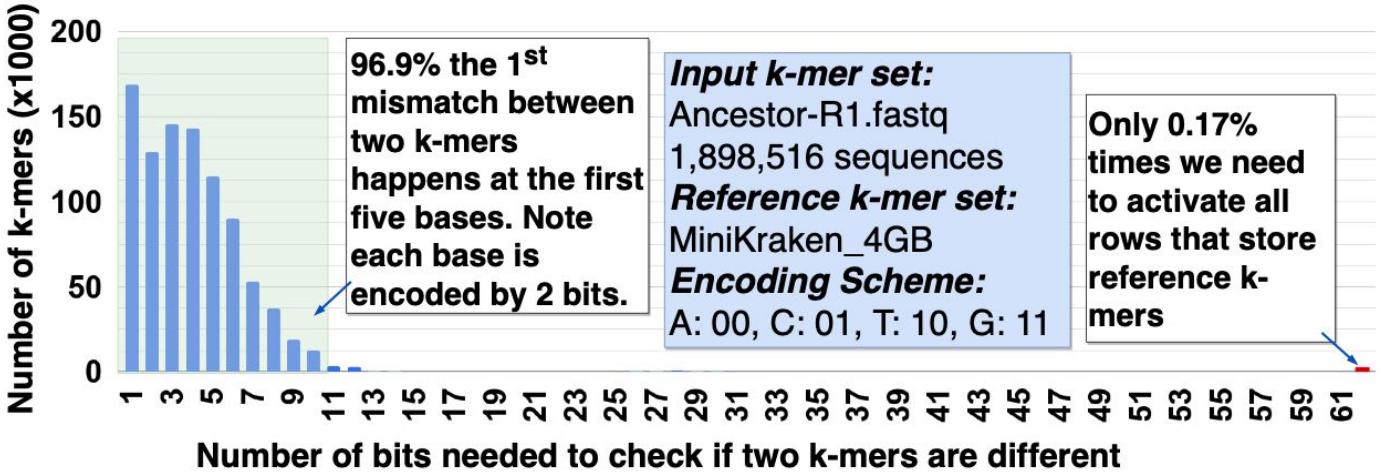
Early Termination Mechanism (ETM)

Consecutive row opening: highly unfavorable

- Long delays: increase in the number of row cycles
- High energy costs: row opening can dominates DRAM energy consumption [1]

Optimization opportunity: *Expected Shared Prefix (ESP)*

- The 1st mismatch location between 2 DNA sequences
- On average, between the 6th and the 8th base [2]



For k-mers, nearly 97% of the 1st mismatch is within the first 5 bases (first 10 bits) □ only need to activate 10 rows

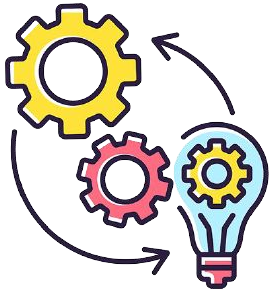
[1]. N. Chatterjee, M. O'Connor, D. Lee, D. R. Johnson, S. W. Keckler, M. Rhu, and W. J. Dally, "Architecting an Energy-Efficient DRAM System for GPUs," in HPCA, 2017.

[2]. E. Fernandez, W. Najjar, and S. Lonardi, "String Matching in Hardware Using the FM-Index," in FCMM, 2011

Implementation

CRISP

Implementation



We implemented three different Sieve types

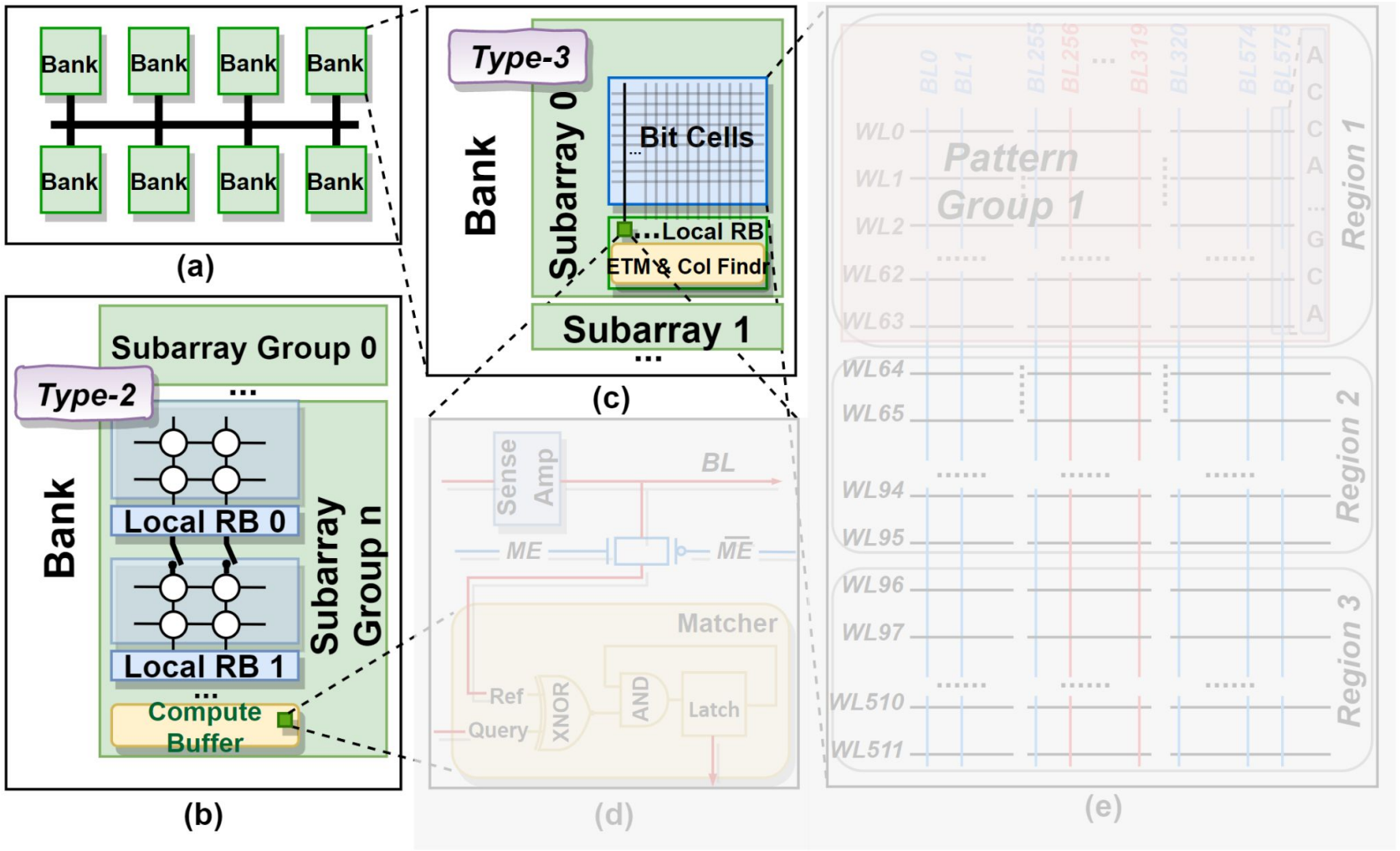
- Explore the **design space**. Optimize for different purposes
  - Type 1: **Minimizes** the hardware **cost**
  - ✓ Type 2: **Balances** between **cost** and **performance**
  - ✓ Type 3: **Maximizes** **speedup** (Subarray-level parallelism)

Implementation of key mechanisms:

- Dispatching *K*-mer queries to the Sieve device
- *K*-mer Indexing: avoid query broadcasting
- Early Termination
- Support for locating and transferring matched genome records (column finder)
- ✓ System Integration

CRISP

Type-2/3 Architecture Walkthrough



a. Sieve adopts the **conventional DRAM organization** and hierarchy, where each chip is made of several banks.

b. Type-2 Zoom-in:

   i. Subarray group

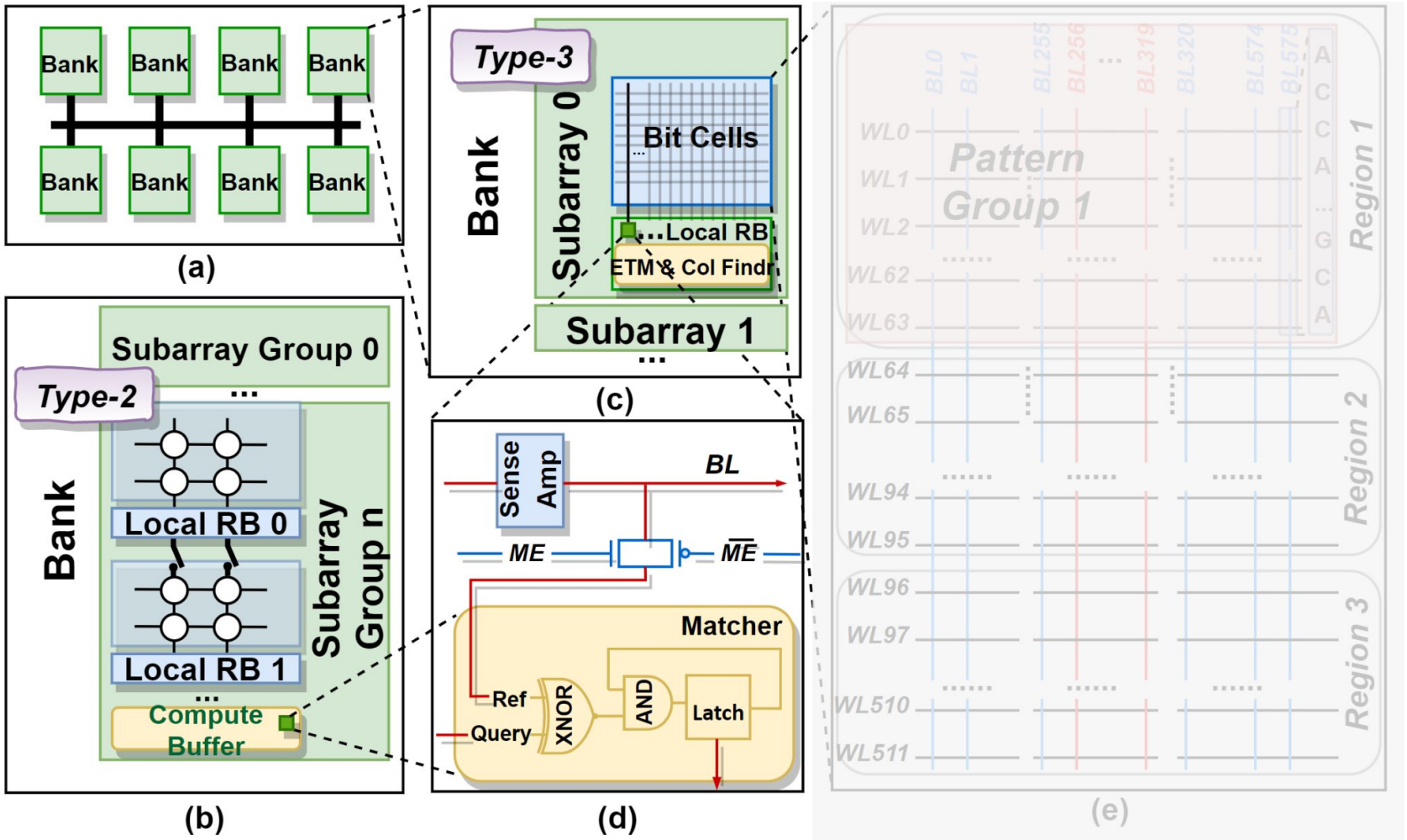
   ii. Each subarray group has a **compute buffer**

c. Type-3 Zoom-in:

   i. SALP

CRISP

Type-2/3 Architecture Walkthrough

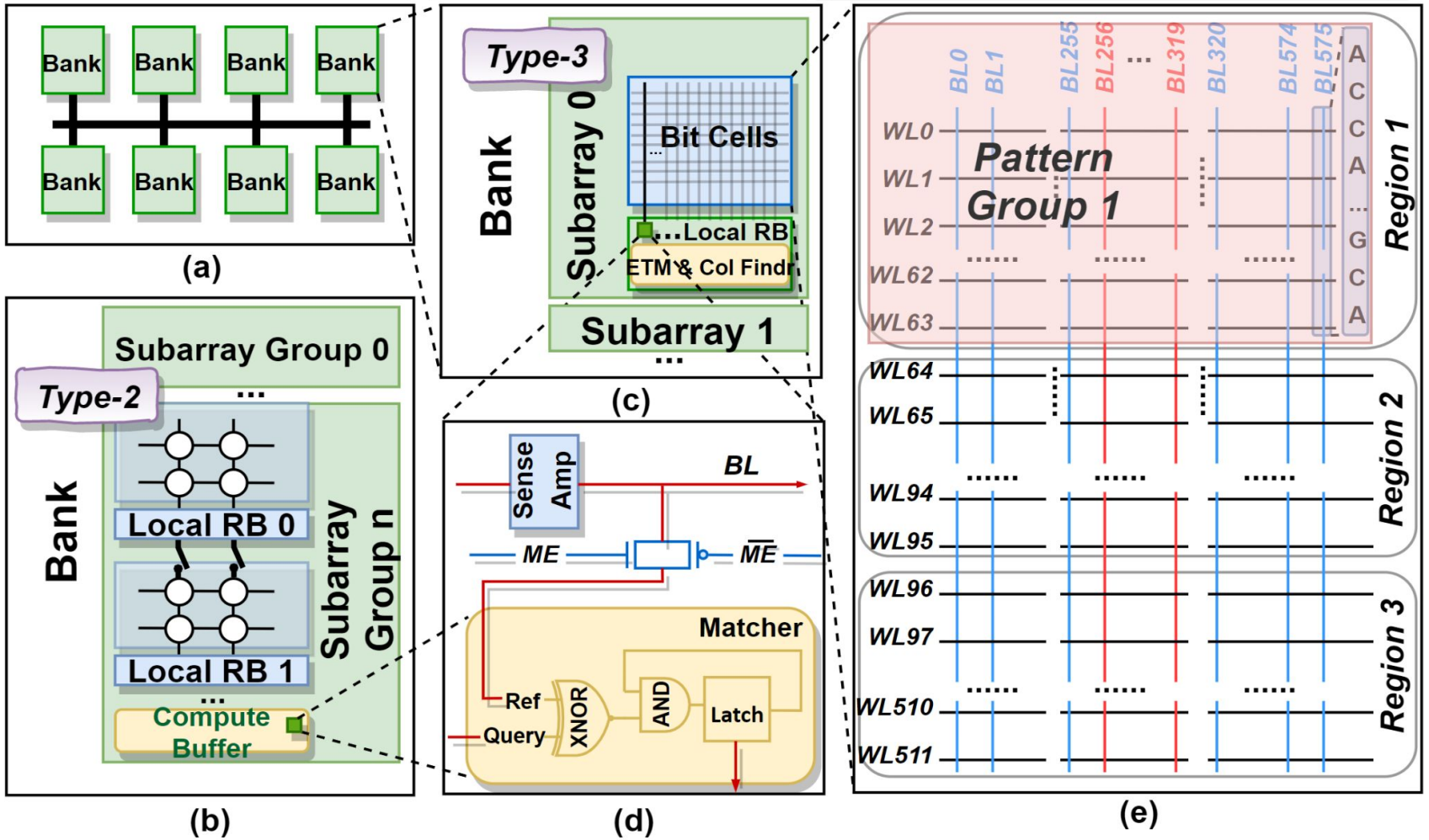


Matcher circuit design:

- **XNOR** gate: compares **reference** bit and the **query** bit
- **1- bit latch:** stores the result of the comparison, preset to 1 initially (default to match).
- The **AND** gate: capturing the running match result bit-by-bit.

CRISP

Type-2/3 Architecture Walkthrough

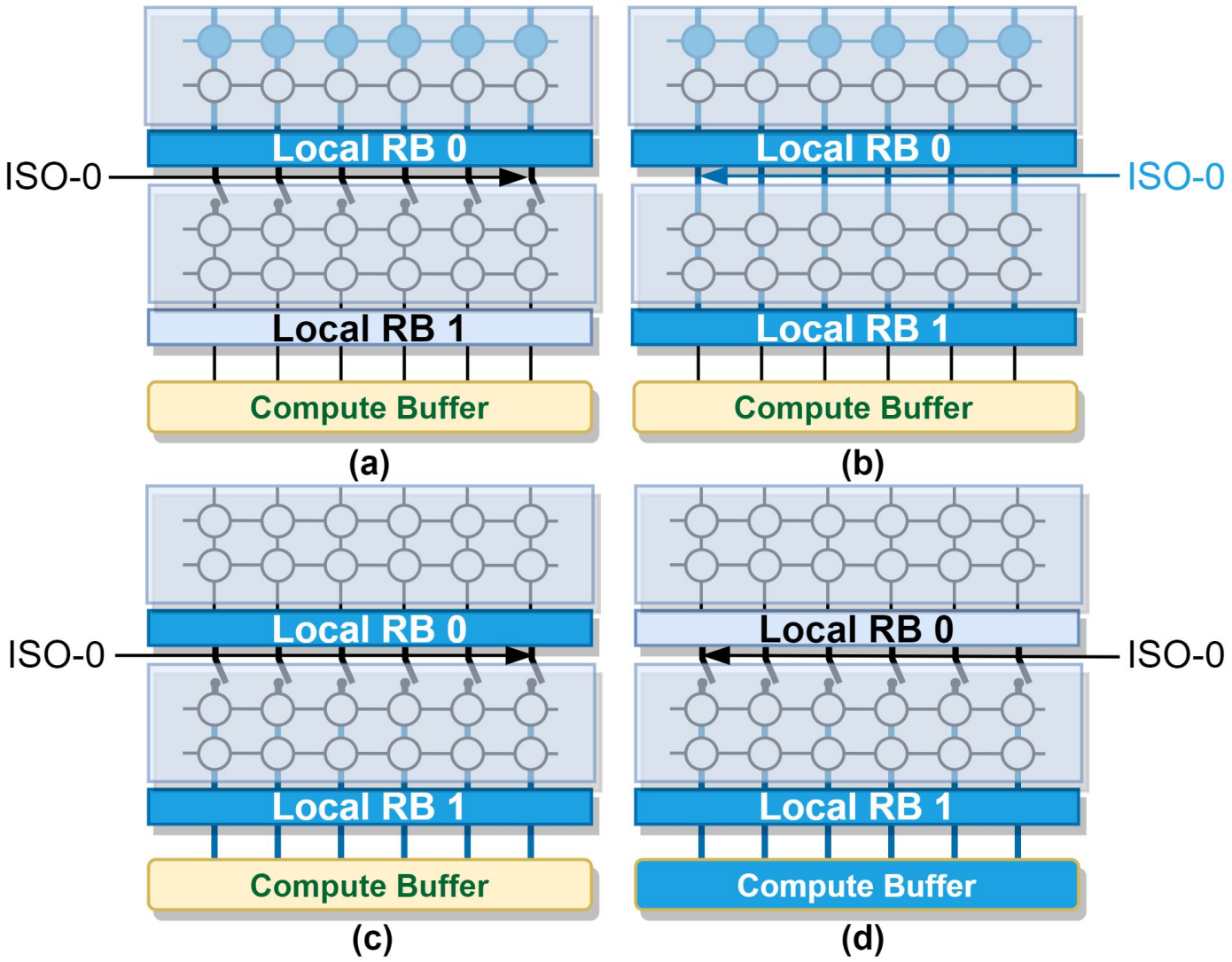


Bit cells within each subarray are divided into **three regions**.

- **Region-1:** stores the **interleaved** reference (blue) and query k-mers (red)
  - **transmission delay** of long wires
- **Region-2:** stores the offsets to the starting address of payloads (genomic records)
- **Region-3:** stores the actual payloads

CRISP

Type-2 Transferring a Row of Bits



To transfer bits from a subarry **Local Row Buffer** to the **Compute Buffer**, we adopt a mechanism from **LISA** (Chang et al. HPCA 16')

Bits are **relayed** from one local row buffer to another in a sequential manner.

(a) DRAM row activation → data is latched onto its **Local RB 0**

(b) The bitlines of subarray 0 are fully driven → **ISO-0** enabled → charge sharing between subarrays 0 and 1 → **Local RB 1** amplifies it further

(c) **Local RB 0** and **Local RB 1** drive their bitlines to the same voltage levels

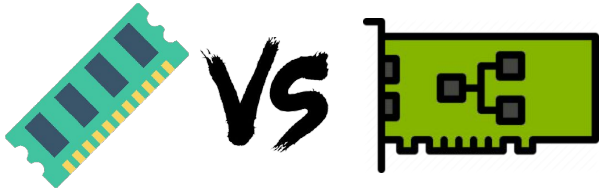
(d) **ISO-0** disconnected and the **Local RB 0** are precharged

CRISP

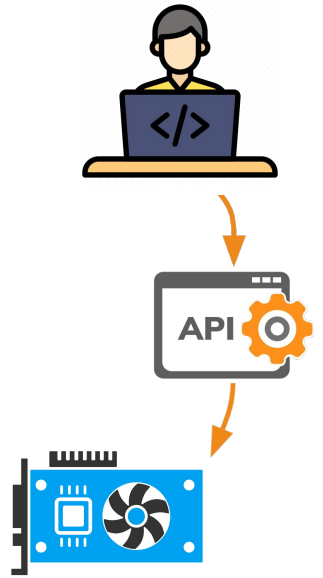
System Integration

Hardware Integration

- **Dual-Inline Memory Module** (DIMM) v.s. PCIe.
  - DIMM: limited power supply and bandwidth
  - DIMM (DDR4) is **sufficient** for Type-1
- **PCIe**: extra **communication overhead**
  - Type-2/3 exceeds the **power cap** and **bandwidth** requirements of DIMM
  - 3 on-chip queues for storing incoming/outgoing PCIe packets and serviced k-mer requests



Software Integration: **Sieve API**: a user-level **library** and an associated kernel **module** or **driver** to interface to the Sieve hardware (future work)



- **Transpose** a conventional database
- **Load** a database into the Sieve
- Make k-mer **queries**

Evaluation and Results

CRISP

Methodology

Workloads:
- Kraken 2 (CPU), CLARK (CPU), cuCLARK (GPU)

Input:
- **Reference Dataset**: MiniKraken_4GB, MiniKraken_8GB , NCBI Bacteria (2785 full bacterial genomes)
- **Input Sequences**: 6 input sequences that come with Kraken and CLARK

Circuit-level SPICE Validation
- Sense amplifier and matcher circuits are implemented using 45nm PTM transistor models to ensure that matcher does not cause any bit flip

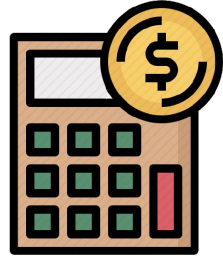
Energy, Area, and Latency Modeling:
- FreePDK45 and OpenRAM (**power** and **latency**). 3D-DATE (DRAM **area** model). Scale down to **22 nm** technology.

Architecture Baselines:
- **PIM: row-major** (DRISA/Ambit and computeDRAM), Sieve-like **column-major** w/o ETM, and **computeDRAM**
- **CPU**: Intel(R) Xeon(R) E5-2658 v4
- **GPU**: Pascal NVIDIA Titan X

Simulator:
- Trace-driven, in-house simulator with an custom DRAMSim2-based front-end
- Model PCIe communication overhead

CRISP

Energy, Latency, & Area Cost



Energy Evaluation

- **Type-2/3** consume only **6%** more energy/DRAM row activation
- Area and the load of the extra transistors is small compared to the sense amplifier and the bitline drivers
- **Type-1** adds **no overhead** on top of the regular DRAM row activation (no modification made to the row buffer) → less energy-intensive than Type-2/3.

Latency Evaluation

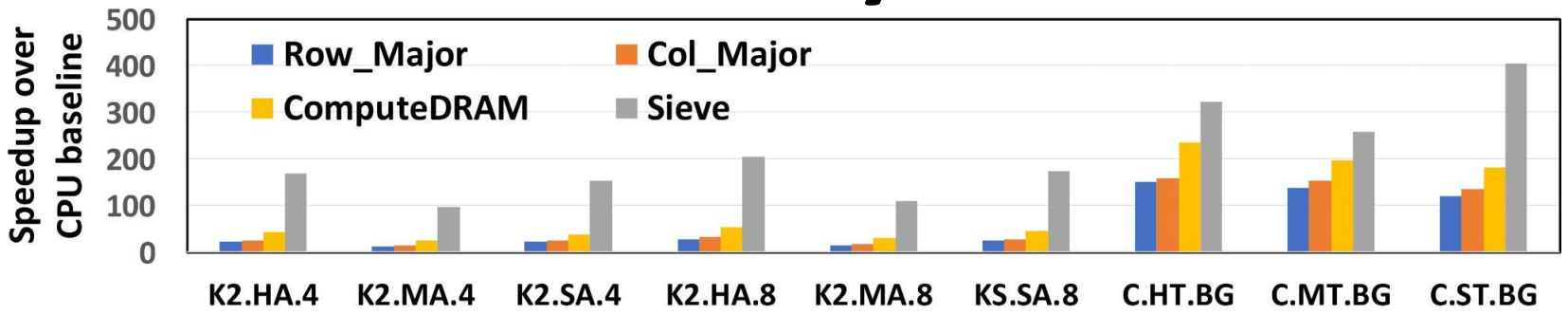
- **Type-2/3** pattern matching and etc. are on the **critical path**, but add **negligible** overhead (**~0.5 ns**) to a DRAM row cycle (**~50 ns**)
- **ETM** and etc. are composed of **simple shifters**, **fast** (**<0.625 ns**)

Area Evaluation

- **$4F^2$** DRAM layout. **short side**/**long side** of the sense amps are **6F**/ **90F**
- **Type-3** adds **340F** to the long side (ETM, Column Finder) → **10.90%** overhead
- **Type-2** adds additional **60F** for **link** → **1/64/128** compute buffers incurs **1.03%**/**6.3%**/**10.75%** overhead (8-bank DRAM chip)
- **Type-1** fits logic at the **center strip** of the DRAM. **Matching circuit** in each bank **0.08%**. Total **2.48%**

CRISP

Performance Analysis



Comparison Against Prior In-Situ Accelerators

- **X-axis**: benchmarks. **Y-axis**: speedup over CPU. **Capacity**: 32 GB
- **Row_Major** (DRISA/Ambit): **horizontal data layout** + **Triple-row activation**
- **Col_Major** (Sieve Type-3): **vertical data layout** w/o **ETM**
- **ComputeDRAM** (Gao et al. MICRO 19'): **horizontal data layout** + **Fast Triple-row activation**
- **Sieve** (Sieve Type-3): **vertical data layout** + **ETM**

Results Summary

1. **Row_Major ≅ Col_Major**
   a. **Row_Major** has lower setup cost but needs to activate more rows
   b. **Col_Major** stops with *K*-mer hits but higher setup cost
2. **ComputeDRAM > Row_Major** or **Col_Major**
   a. Outperforms both both due to faster Triple-row activation
3. **ETM** (**Sieve**) provides additional 5.2X to 7.2X performance gain
   a. Due to data mapping issue, **ComputeDRAM** and **Row_Major** lack such opportunity
4. **Col_Major > Row_Major**
   a. Both design opens roughly same amount of rows.
   b. **Row_Major** performs better with higher *K*-mer hits. But real data has low hit rate.

CRISP

Conclusions

CRISP

Conclusions and Future Works

Identify ***K*-mer matching** as a bottleneck for its **memory-intensive** nature

Propose **Sieve**, a set of **DRAM-based in-memory** accelerators

- Storing reference k-mer patterns **vertically**
- **Minimal** set of Boolean logic for pattern matching
- Optimize Sieve with an Early Termination Mechanism (**ETM**)
- Design space exploration for different concerns

Column-major eliminates multi-row activation and enables ETM that prunes unnecessary operations ⟶ **performance gain** and **energy efficiency**

Sieve helps in invaluable life-saving tasks

- Development of vaccines
- Therapeutics against bioterror
- Surveillance of pathogens

Future work, investigating a general-purpose Sieve that aids other application domains.

CRISP

JUMP

Joint University Microelectronics Program

www.src.org/program/jump

Semiconductor Research Corporation

@srcJUMP