In-Memory Accelerator Architectures for Machine Learning and Bioinformatics

TECHNION Israel Institute of Technology

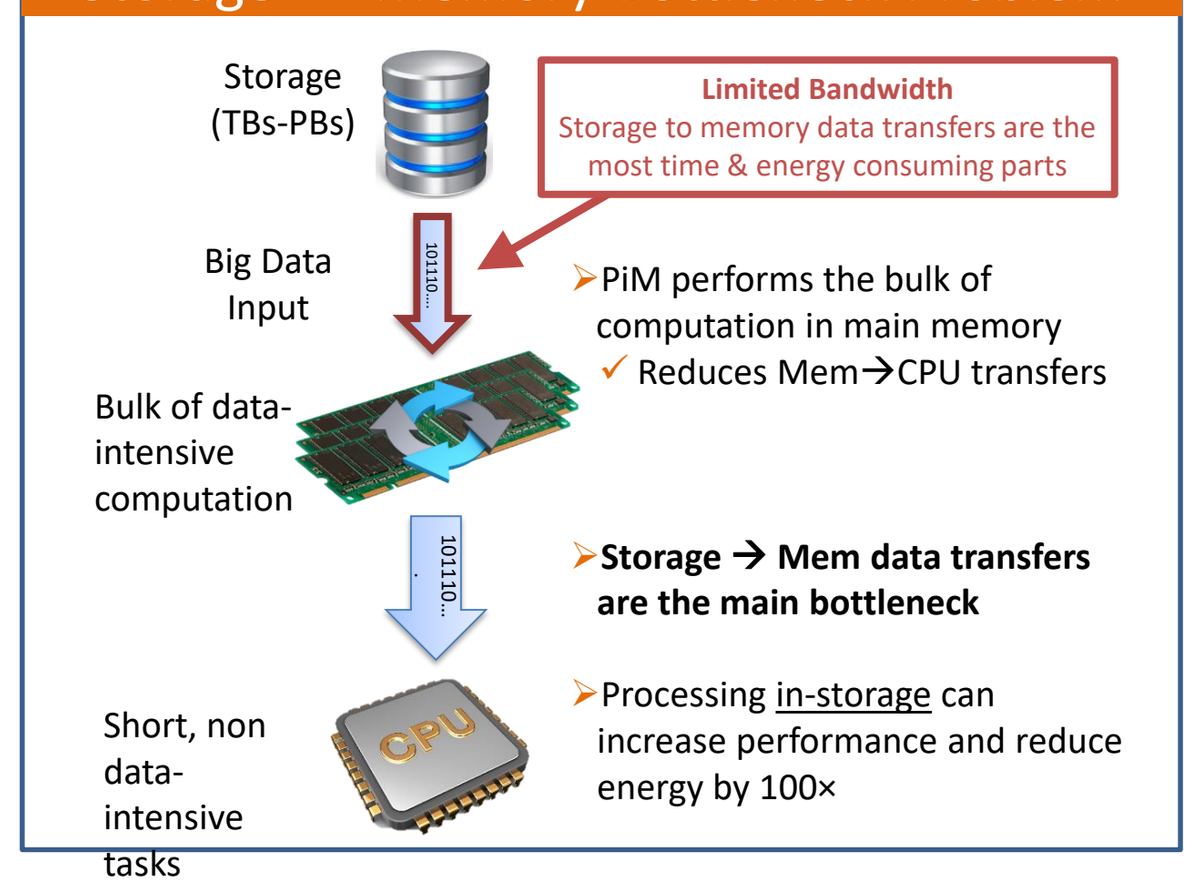
ANDREW AND ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

Roman Kaplan

PhD Advisor: Prof. Ran Ginosar

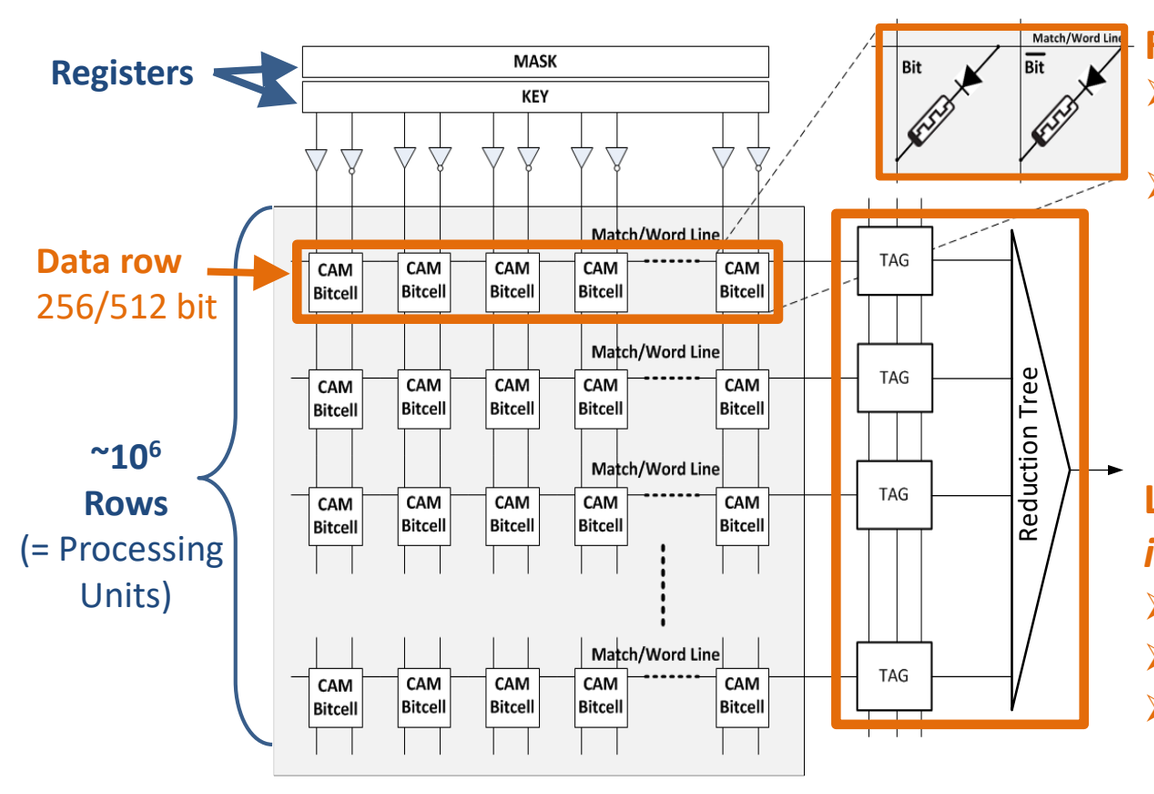
Resistive Content Addressable Memory for Machine Learning and Bioinformatics

Storage ⇔ Memory Bottleneck Problem



- PiM performs the bulk of computation in main memory
  - ✓ Reduces Mem→CPU transfers
- **Storage → Mem data transfers are the main bottleneck**
- Processing in-storage can increase performance and reduce energy by 100×

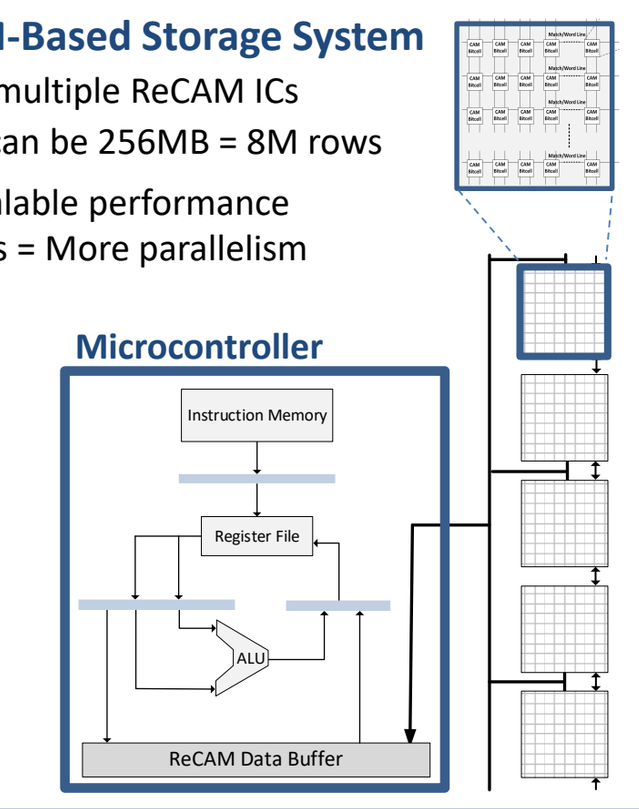
Resistive Content Addressable Memory (ReCAM): A PRinS Device



Resistive Bitcells
- Based on resistive elements (e.g., memristors)
- High density & non-volatile

Logic next to data: *in-situ* processing
- Processing unit (PU) per row
- All PUs can operate in parallel
- **Instruction are based on bit-serial associative processing**

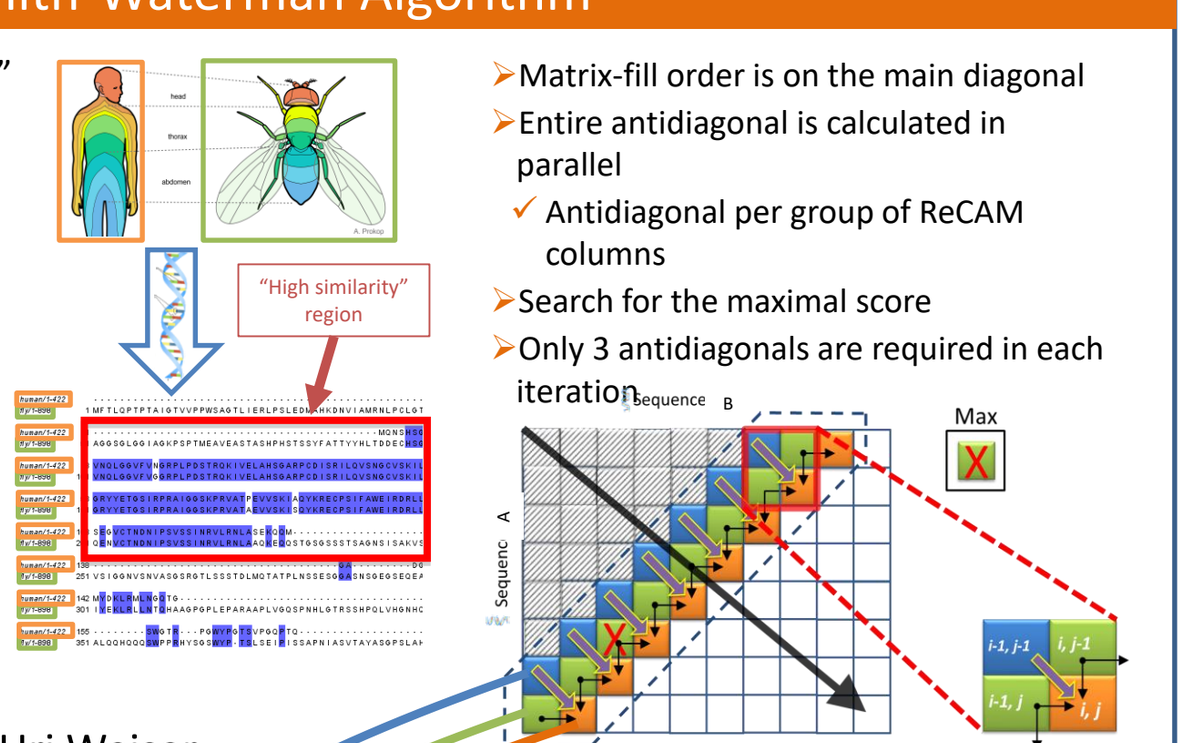
Full ReCAM-Based Storage System
- Divided to multiple ReCAM ICs
  - ✓ E.g., IC can be 256MB = 8M rows
- Linearly scalable performance
  - → More ICs = More parallelism



PRinS Application
DNA Local Sequence Alignment

Smith-Waterman Algorithm

- Finds regions with "highest similarity" between two sequences (DNA\protein)
  - ✓ Proven to be optimal
- Every "match", "mismatch" & "gap" has a score
- S-W is based on dynamic programming
  - ✓ Fills a $m \times n$ matrix
  - ✓ Has a quadratic computational complexity $O(m \cdot n)$
- We focus on the computationally-heavy scoring step

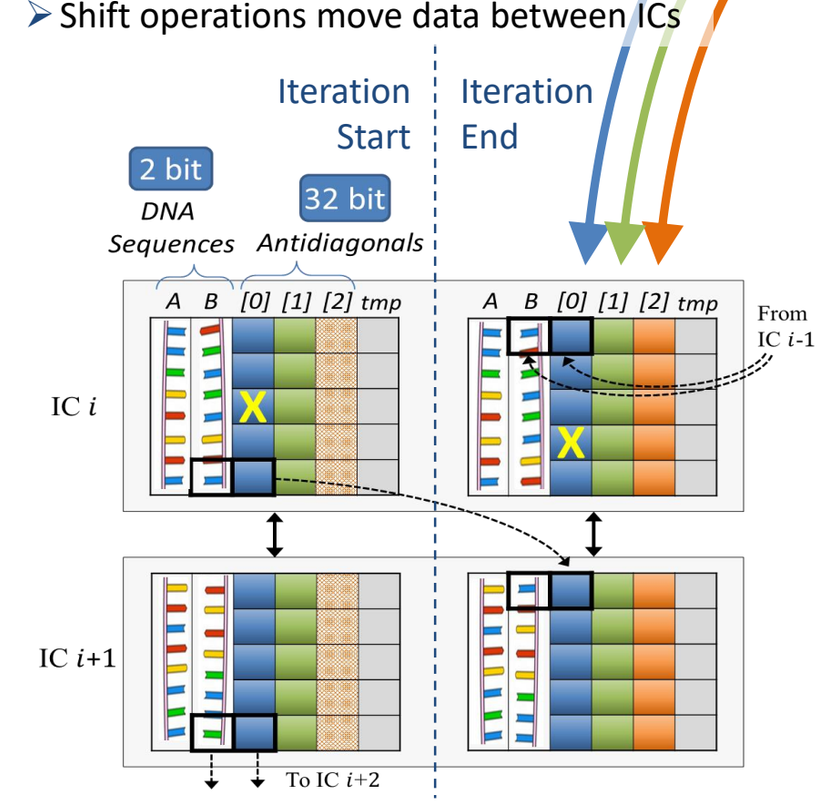


- Matrix-fill order is on the main diagonal
- Entire antidiagonal is calculated in parallel
  - ✓ Antidiagonal per group of ReCAM columns
- Search for the maximal score
- Only 3 antidiagonals are required in each iteration

* This work was done with prof. Uri Weiser

PRinS Implementation & Performance Comparison

In-Memory Computation
- IC = Integrated Circuit
- Shift operations move data between ICs

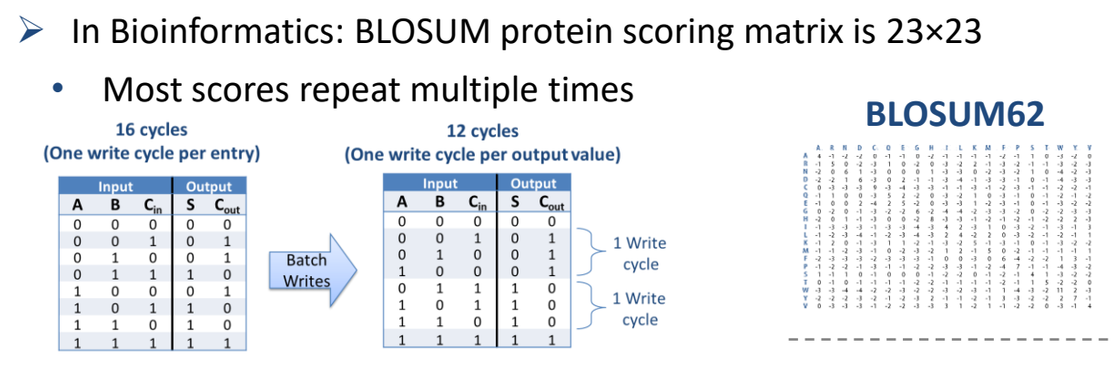


Performance Comparison
- Cycle-accurate simulator: 8GB of storage running at 500MHz
- Compared to multi-accelerator state-of-the-art solutions: FPGA, Xeon Phi and GPU
- ReCAM shows **4.7×** higher performance than a 384-GPU solution

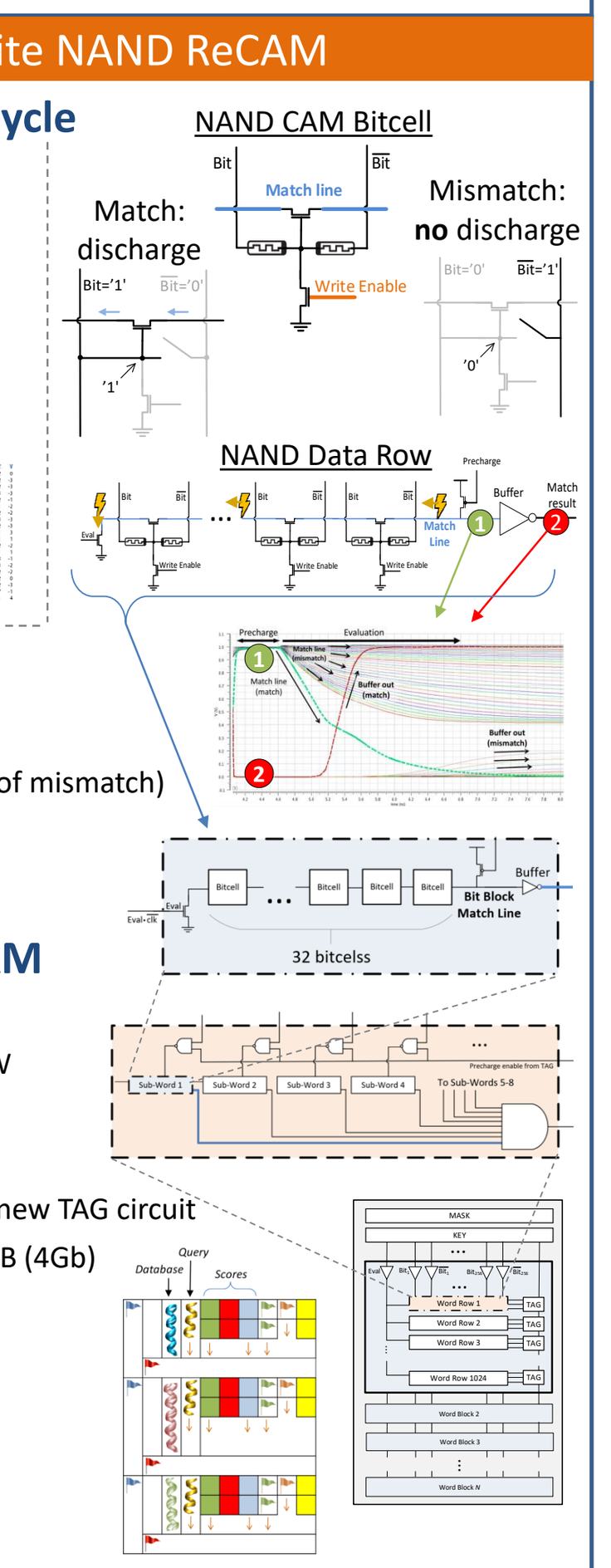
Accelerator	FPGA	Xeon Phi	GPU	ReCAM
Performance (TCUPS)	6.02	0.23	11.08	**52.68**
# of ICs	128	4	**384**	32
Publication	L.Wienbrandt. The FPGA-based High-Performance Computer RIVYERA for Applications in Bioinformatics. In Language, Life, Limits: 10th CiE, pages 383-392. Springer 2014.	Y. Liu and B. Schmidt. 2014b. SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors. In IEEE ASAP. 184–185.	Sandes, E. F. O., et al. "CUDAlign 4.0: Incremental Speculative Traceback for Exact Chromosome-Wide Alignment in GPU Clusters." IEEE	

Improved Architecture: Batch-Write NAND ReCAM

Batch-Write: Write One Output Value in One Cycle
- Associative processor: one write cycle for truth table row
- With Batch-Write: one write cycle for **output value**
  - ➔ All outputs with the same value are batched
- New TAG logic to accumulate consecutive compares
- In Bioinformatics: BLOSUM protein scoring matrix is 23×23
  - Most scores repeat multiple times



NAND Array: Improved Energy Efficiency
- In associative processing, most compares result in mismatch
- NAND CAM bitcell: 2T2R bitcells that discharge on match (instead of mismatch)
- No discharge on mismatch: charge remains on mismatch
  - → Less precharge energy is required for next compare operation



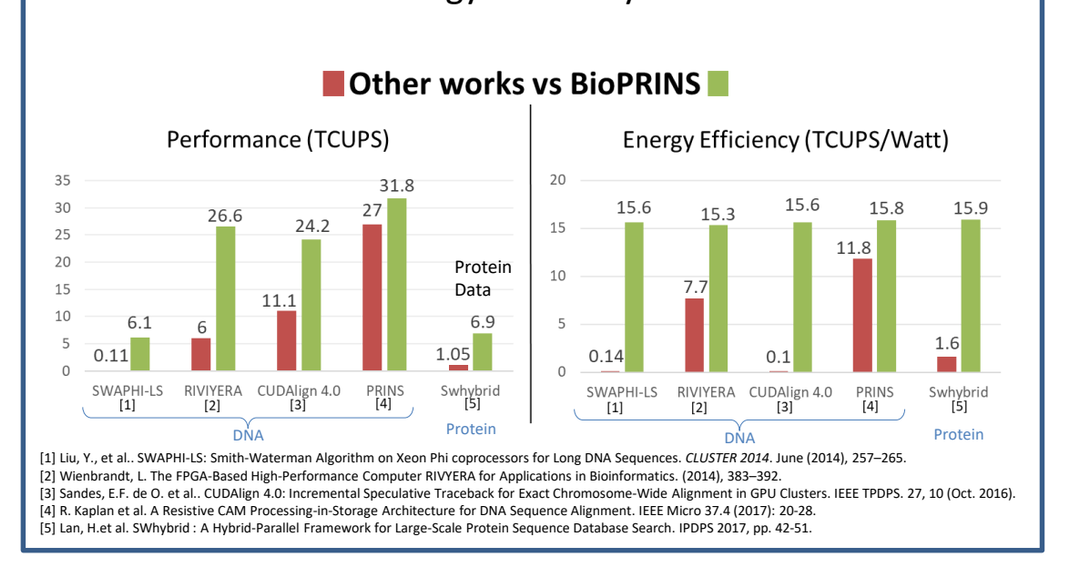
Evaluations: NOR ReCAM vs. BW NAND ReCAM
- BW reduces #cycles per operation:
  - ✓ Logic and arithmetic operations: 25%-37.5% less cycles with BW
  - ✓ BLOSUM62: 48% less cycles for score calculation (1058 vs. 544)

Functionality Area, Performance and Power
- Vs. NOR: +13% in chip area (+40% in TAG area), +3% in energy of new TAG circuit
- Single chip w/ 28nm node process & 300mm^2 die contains 512MB (4Gb)
- Dissipates 220W at 500MHz for logic and arithmetic operations

New Algorithm: Protein Database Search
- Entire protein database (e.g., UniProt) is stored in PRINS
  - Query is a new (unknown?) protein
- Alignment of each database entry with query in parallel

Performance and Energy Efficiency Results

BioPRINS: PRINS for Biological Sequence Search
- BioPRINS compared to five large-scale platforms, including PRINS
- Performance measured in Tera CUPS (TCUPS)
- comparisons over multiple datasets, DNA and protein
  - Results taken from publications (below)
- BioPRINS achieves 2.2-55× higher perf. than other solutions, with 2-156× better energy efficiency

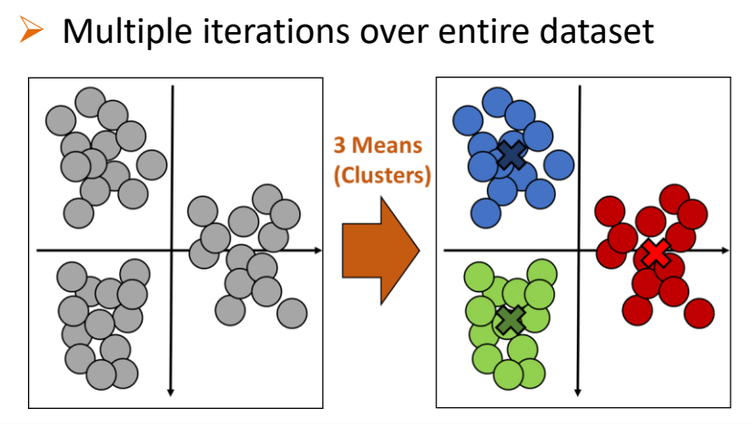


[1] Liu, Y., et al.. SWAPHI-LS: Smith-Waterman Algorithm on Xeon Phi coprocessors for Long DNA Sequences. CLUSTER 2014. June (2014), 257–265.
[2] Wienbrandt, L. The FPGA-Based High-Performance Computer RIVYERA for Applications in Bioinformatics. (2014), 383–392.
[3] Sandes, E.F. de O. et al.. CUDAlign 4.0: Incremental Speculative Traceback for Exact Chromosome-Wide Alignment in GPU Clusters. IEEE TPDPS. 27, 10 (Oct. 2016).
[4] R. Kaplan et al. A Resistive CAM Processing-in-Storage Architecture for DNA Sequence Alignment. IEEE Micro 37.4 (2017): 20-28.
[5] Lan, H.et al. SWhybrid : A Hybrid-Parallel Framework for Large-Scale Protein Sequence Database Search. IPDPS 2017, pp. 42-51.

Machine Learning Application
K-Means and K-Nearest Neighbors

K-Means Clustering Algorithm

- Common Machine Learning algorithm
- For each sample, finds its group among possible K
- Widely used in many fields, including:
  - ✓ Image processing
  - ✓ Anomaly detection
- Data intensitve
  - Multiple iterations over entire dataset



Two Main Loops

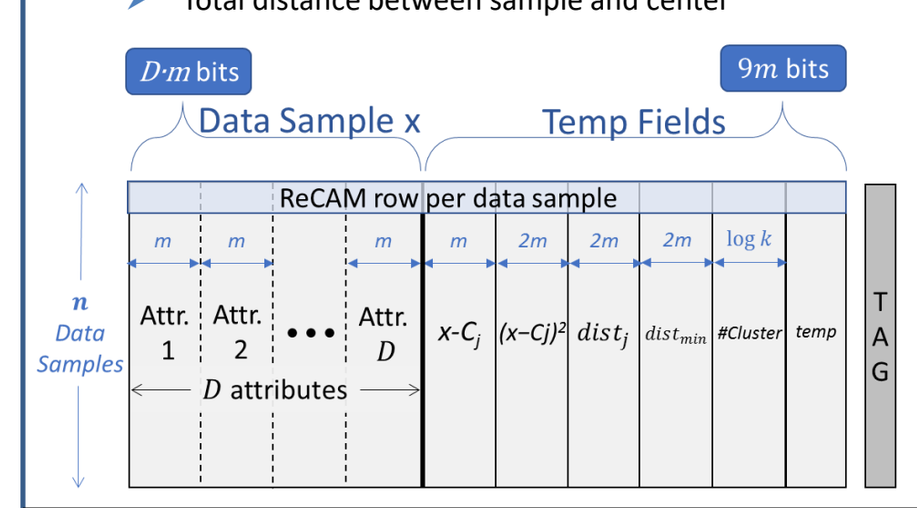
1. For each Sample:
   For each Center:
   Calculate |Sample – Center|
2. For each Center:
   For each Sample:
   Calculate Sum(Sample)

$O(n)$ in a von Neumann machine

$O(1)$ in ReCAM

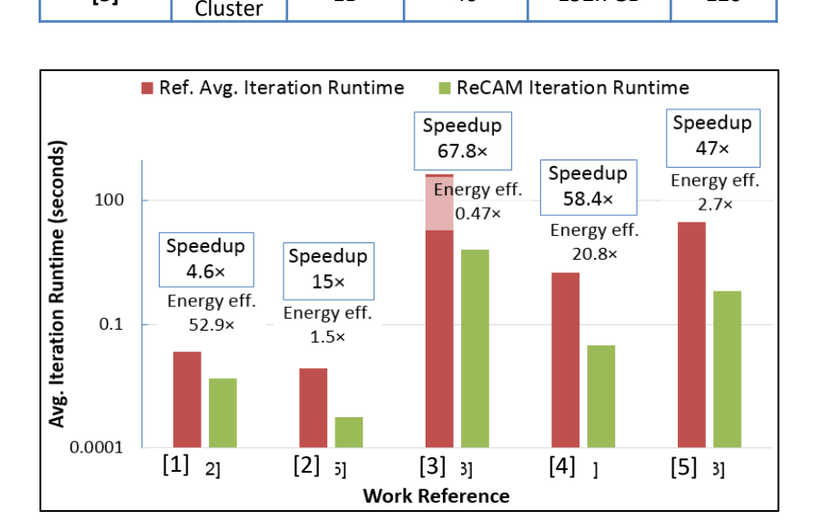
PRinS Implementation & Performance Comparison

In-Storage Computation
- In large datasets: #Data Samples >> #Clusters
- Computation over **ALL** samples in one instruction
- Each sample attributes requires multiple temp fields
  - Difference between center's attribute
  - Squared difference
  - Total distance between sample and center



Comparison to other Works

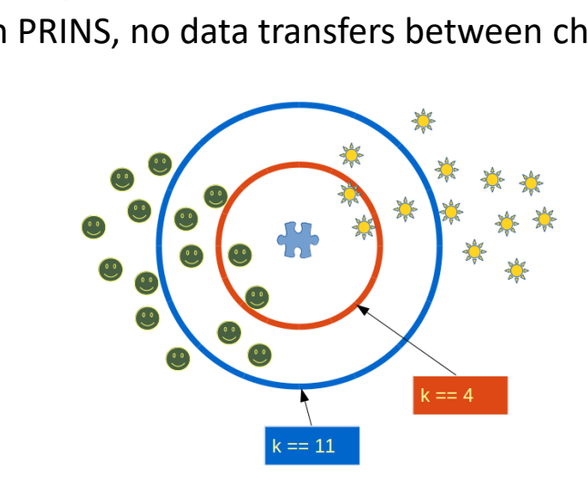
Work Reference	Platform	Dataset: Samples	Dataset: Attributes	Dataset: Size on disk	Clusters
[1]	FPGA	1M	1	4MB	128
[2]	FPGA	2M	4	31.6MB	4
[3]	Intel i7	2.5M	68	318.8MB	10000
[4]	GPU	1.4M	5	21.3MB	240
[5]	10-GPU Cluster	1B	40	152.7GB	120



[1] Li, Zhehao, et al. "High-performance K-means Implementation based on a Coarse-grained Map-Reduce Architecture." arXiv:1610.05601 (2016).
[2] Ramanathan, Nadesh, et al. "A Case for Work-stealing on FPGAs with OpenCL Atomics." SIGDA 2016.
[3] Ding, Yufei, et al. "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup." ICML 2015.
[4] Bhimani, Janki, et al. "Accelerating K-Means clustering with parallel implementations and GPU computing." HPEC 2015.
[5] Rossbach, Christopher J. et al. "Dandelion: a compiler and runtime for heterogeneous systems.", SOSP 2013.

K-Nearest Neighbors Algorithm

- Common classification algorithm
- Given a query sample, find the k most closest dataset samples
- Large datasets cannot fit on memory of one chip
  - ➔ Requires communication between Pes
- On PRINS, no data transfers between chip

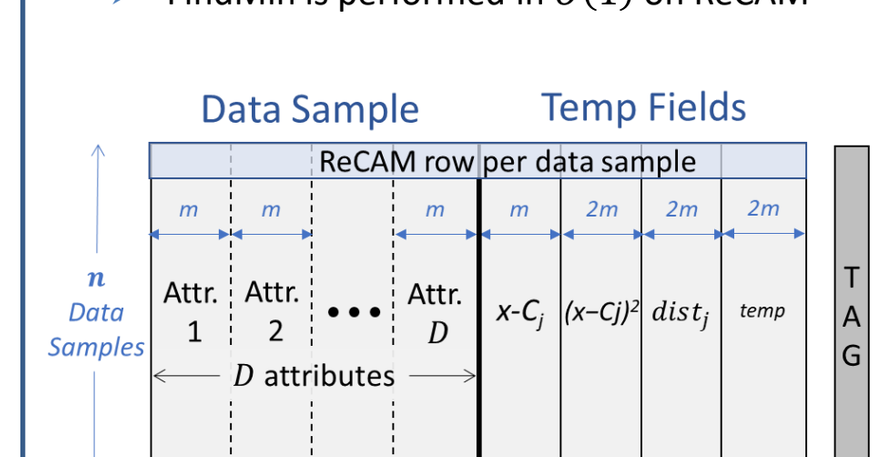


Pseudocode of KNN on PRINS

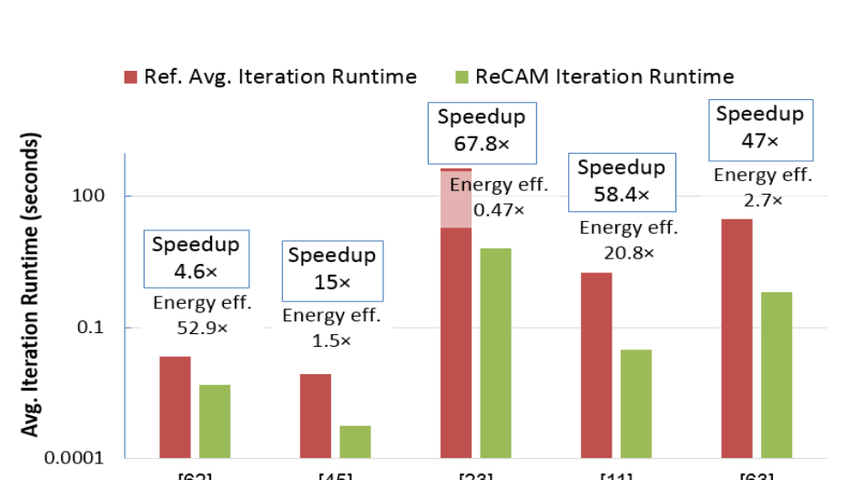
```
//K denotes the number of nearest neighbors.
//Every sample x ∈ X may be stored in several consecutive
RCAM rows; the code assumes one row per sample for
simplicity.
//Each sample is characterized by M attributes
//Calculate distance of each dataset sample from query
1:   For each attr ∈ [1, M]
       Do-all x ∈ X:          In parallel, on entire ReCAM
2:       dist_attr ← Query_attr − x_attr
3:       sqDist_attr ← (dist_attr)^2
4:       sqDist ← sqDist_partial + sqDist_attr
//Assume unique sqDist values

//Find K closest samples
//Histogram of all classes maintained by microcontroller
//Start with all samples unmarked
5:   Loop K times
6:     Tag all unmarked samples                  O(1) time for findMin
7:     Tag and mark first row with min value of sqDist    on ReCAM
8:     Retrieve class of tagged row to microcontroller
9:     On microcontroller: Histogram[class]++
//Classification: Class with highest histogram
```

PRinS Implementation & Performance Comparison

- Distance of each dataset sample from query sample is computed
- Temp fields store intermediate results
  - Distance over each dimensions is accumulated
- Once distance of each dataset sample is found, closest sample is searched serially
  - In all cases: N is usually small (<100)
  - FindMin is performed in $O(1)$ on ReCAM



Work Ref.	Platform	Dataset: Samples	Dataset: Attributes	Dataset: Size on disk	Clusters
[11]	GPU	1.4M	5	21.3MB	240
[45]	FPGA	2M	4	31.6MB	4
[62]	FPGA	1M	1	4MB	128
[23]	Intel i7	2.5M	68	318.8MB	10000
[63]	10-GPU Cluster	1B	40	157.2GB	120



PRinS Application: In-Storage Deduplication

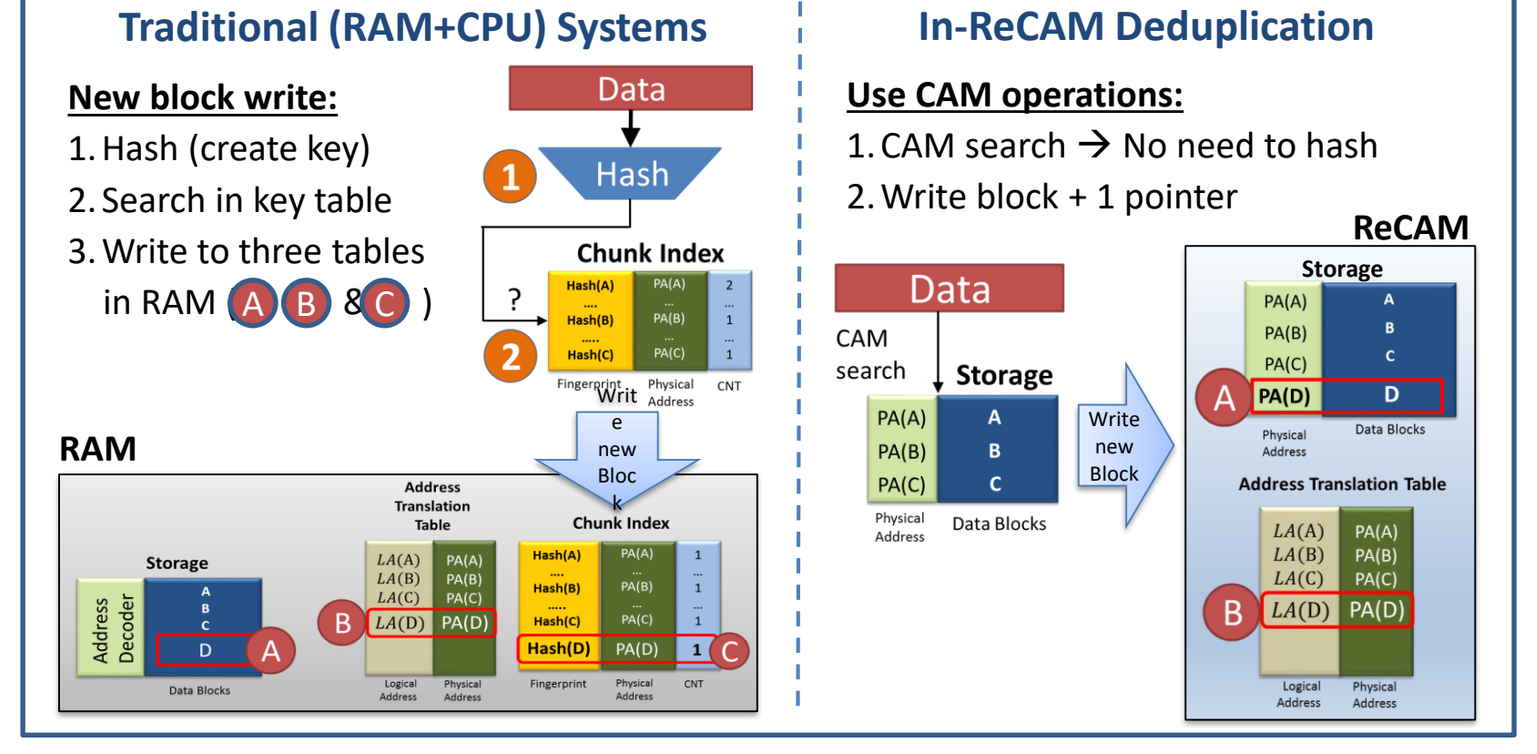
Traditional Systems vs. ReCAM

Traditional (RAM+CPU) Systems

New block write:
1. Hash (create key)
2. Search in key table
3. Write to three tables in RAM (A B & C)

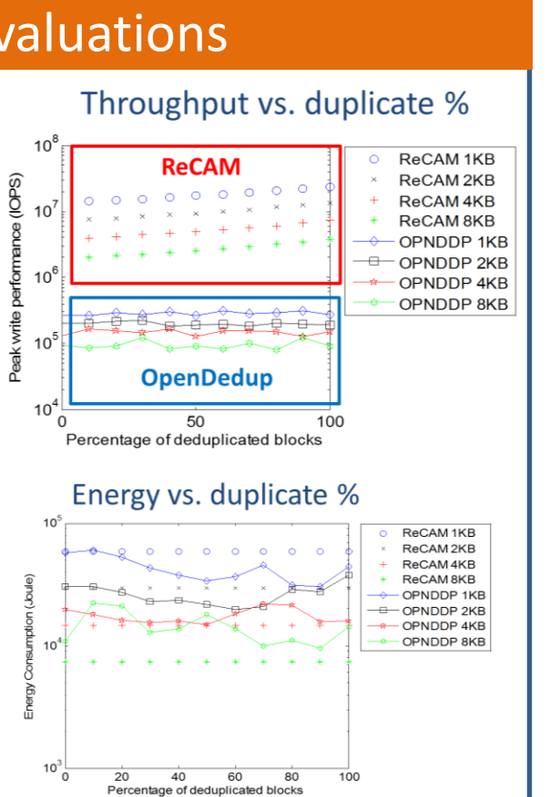
In-ReCAM Deduplication

Use CAM operations:
1. CAM search → No need to hash
2. Write block + 1 pointer



Performance Evaluations

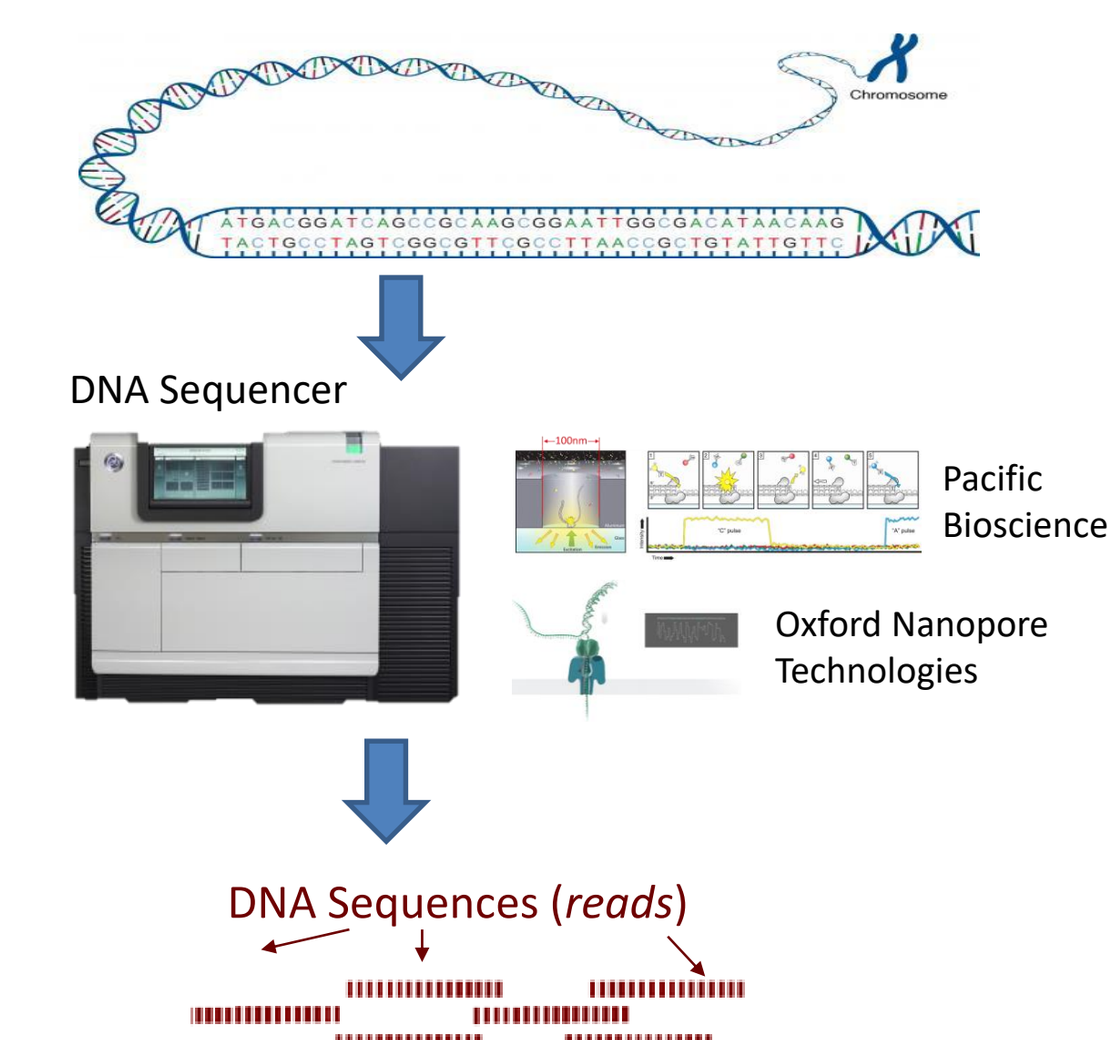
- ReCAM was simulated with a cycle-accurate simulator
- ReCAM Parameters: 256GB @1GHz
- Opendedup executed on high-end server: 4×8 octa-core CPU, 64GB RAM, 800GB SSD drive
- **ReCAM has 100× higher throughput than deduplication with RAM+CPU**
- Energy consumption is similar or lower for the common block sizes (4 & 8KB)



DNA Long Read Mapping

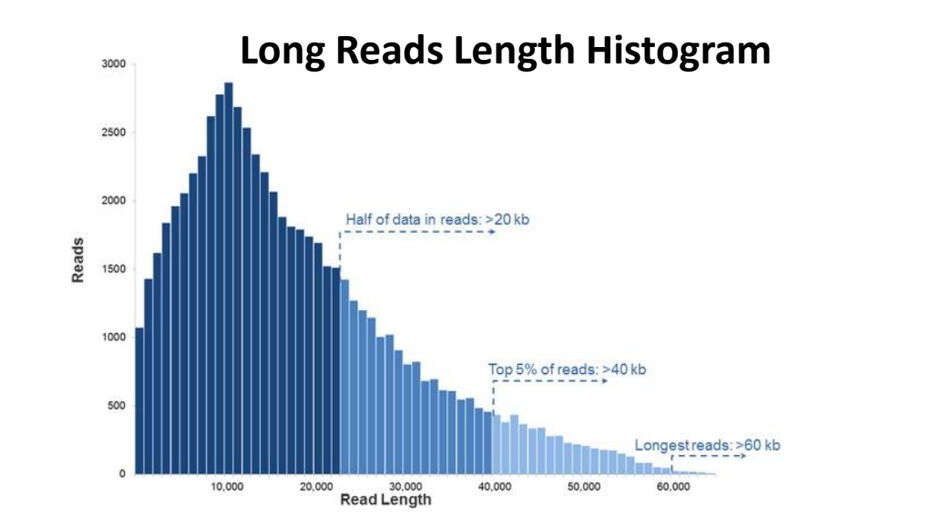
3rd Generation Sequencing: Real-Time Single-Molecule

- DNA is composed of 4 nucleotides: 'A', 'G', 'C', 'T' (base pairs, bps)
- Reading the content of an entire DNA strand at once (e.g., Chromosome) isn't possible
- DNA sequencers output strands of the DNA (*reads*)
- 3rd generation sequencers vs. 2nd generation:
  - Produce outputs faster (hours vs. days)
  - Reads are longer (10k+ bps vs. 100-300 bps) ➔ called *long reads*
  - Simpler preparation (less lab work)
- The downsides: many errors, up to 15%
  - Errors can be insertions, deletions and substitutions

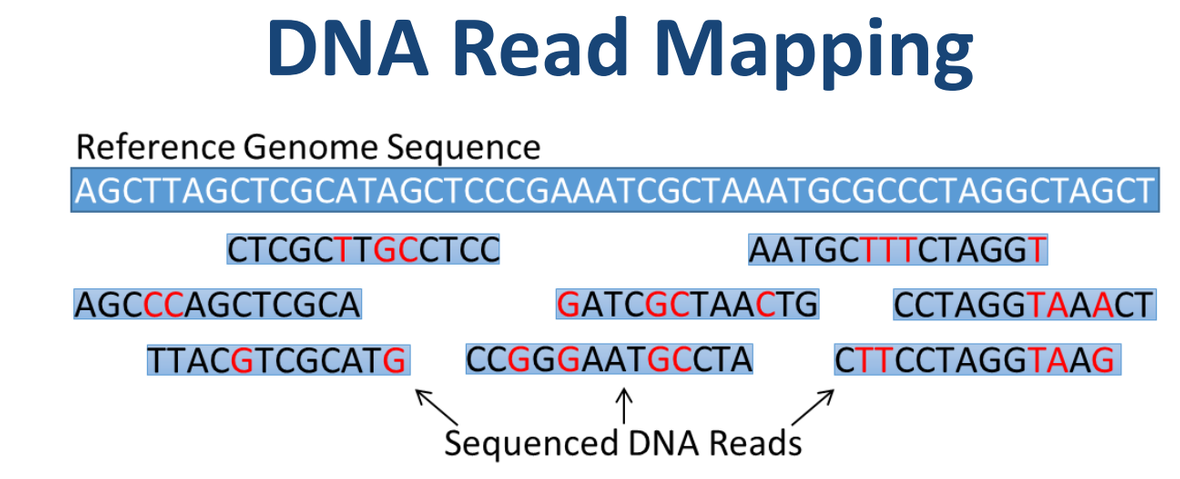


DNA Long Read Mapping

- The problem: "stitching" together all the DNA reads
- When an organism was previously sequenced (e.g., human), this sequence is used as a reference to construct the new organism genome
- Requires mapping 1M+ reads against a reference sequence (e.g., human is 3Gbps)
- Existing read mapping tools use technology-specific heuristics, complex data structures and large memory requirements
- Hardware solutions only addressed short reads with few errors (from 2nd generation)
- Long reads contain many errors and pose a challenge for an effective hardware acceleration solution



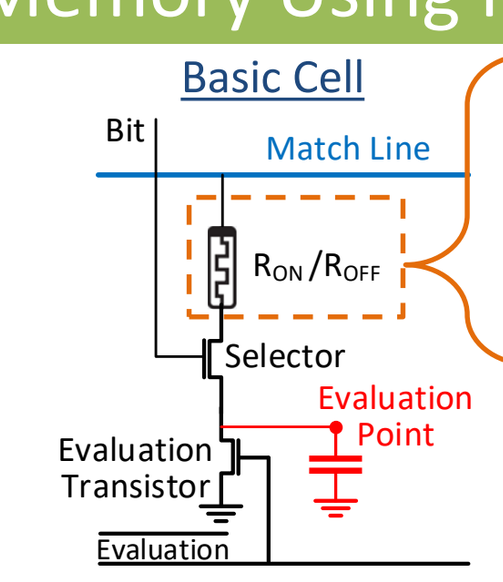
DNA Read Mapping



The Solution
RASSA: Resistive Approximate Similarity Search Accelerator

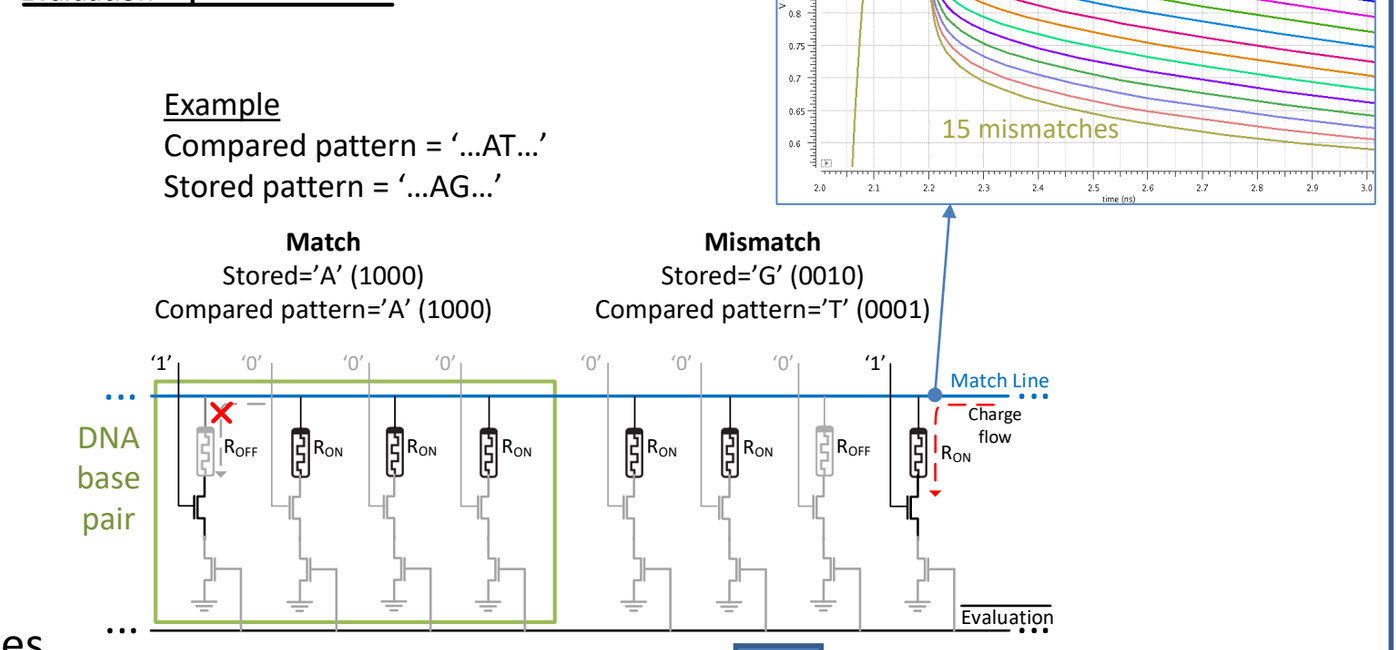
Processing-in-Memory Using Memristors

- Basic cell: 1 memristor, 2 transistors (2T1R)
- The memristor serves as a programmable non-volatile switch
  - Low resistance (R_{on}) – open switch
  - High resistance (R_{off}) – closed switch
- One-hot encoding to store DNA base pairs
  - 4 cells encode one base pair



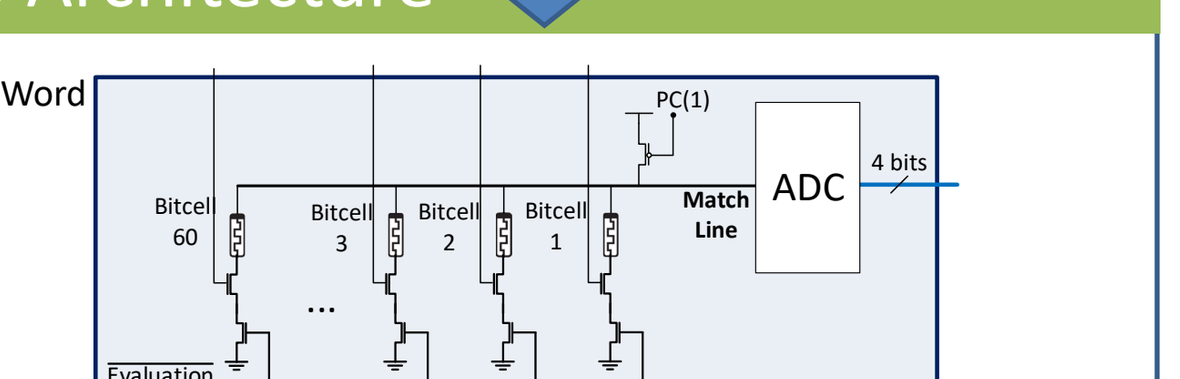
Memristors (Memory Resistor)
- Change resistance if applied voltage higher than a threshold
- High density
- Non-volatile

Comparing a pattern
- Compared pattern is applied on all bitcells
- R_{on} allows charge to flow
  - **Match**: no charge flow → N.C. in Match line
  - **Mismatch**: charge flows through R_{on} memristor → Match line voltage drops
- Charge flow through one bitcell allows to quantify the drop in match line
- Match line voltage level translates to number of matches



The RASSA Bottom-Up Architecture

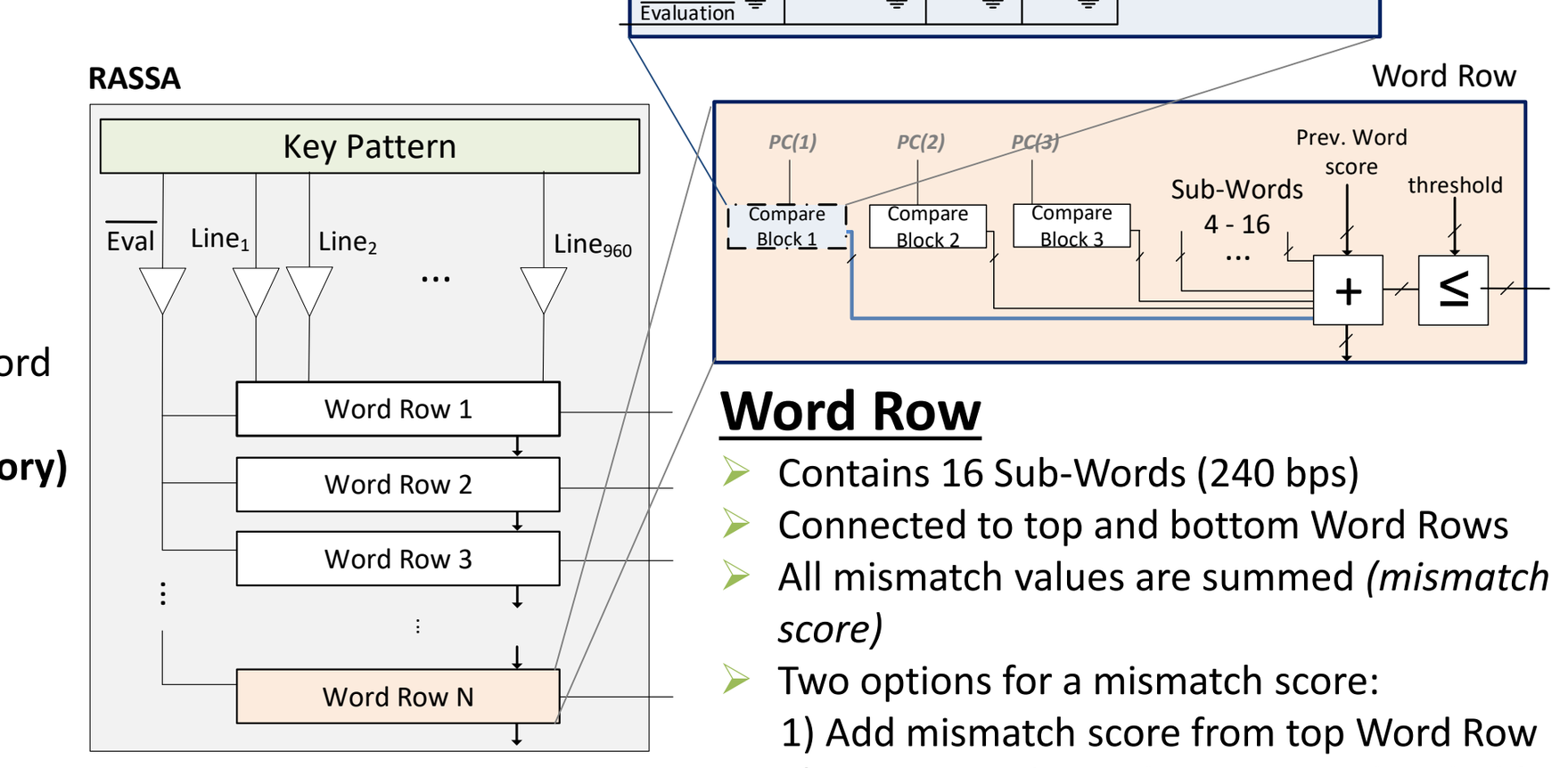
Sub-Word
- Contains 60 bitcells ➔ encodes 15 DNA base pairs (0 through 15 matches = 4 bit)
- Analog-to-Digital converter translates match line voltage level to number of matches



RASSA
- Holds multiple Word Rows
- A compare pattern is applied on all Word Rows
- The comparison takes place in all Word Rows ***in-situ*** and **in parallel**

(Massively parallel processing-in-memory)

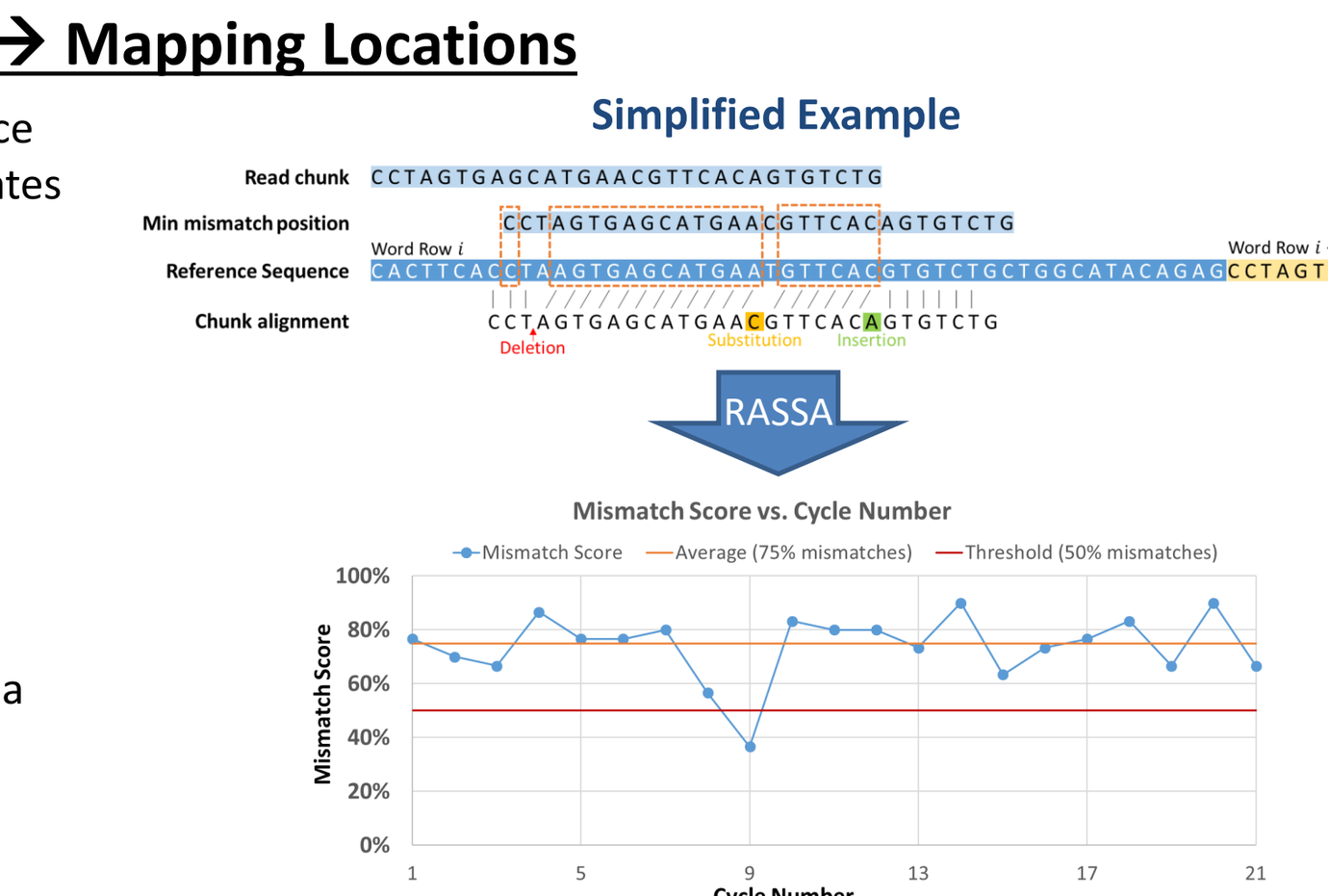
- Scores below a threshold indicate a potential mapping location



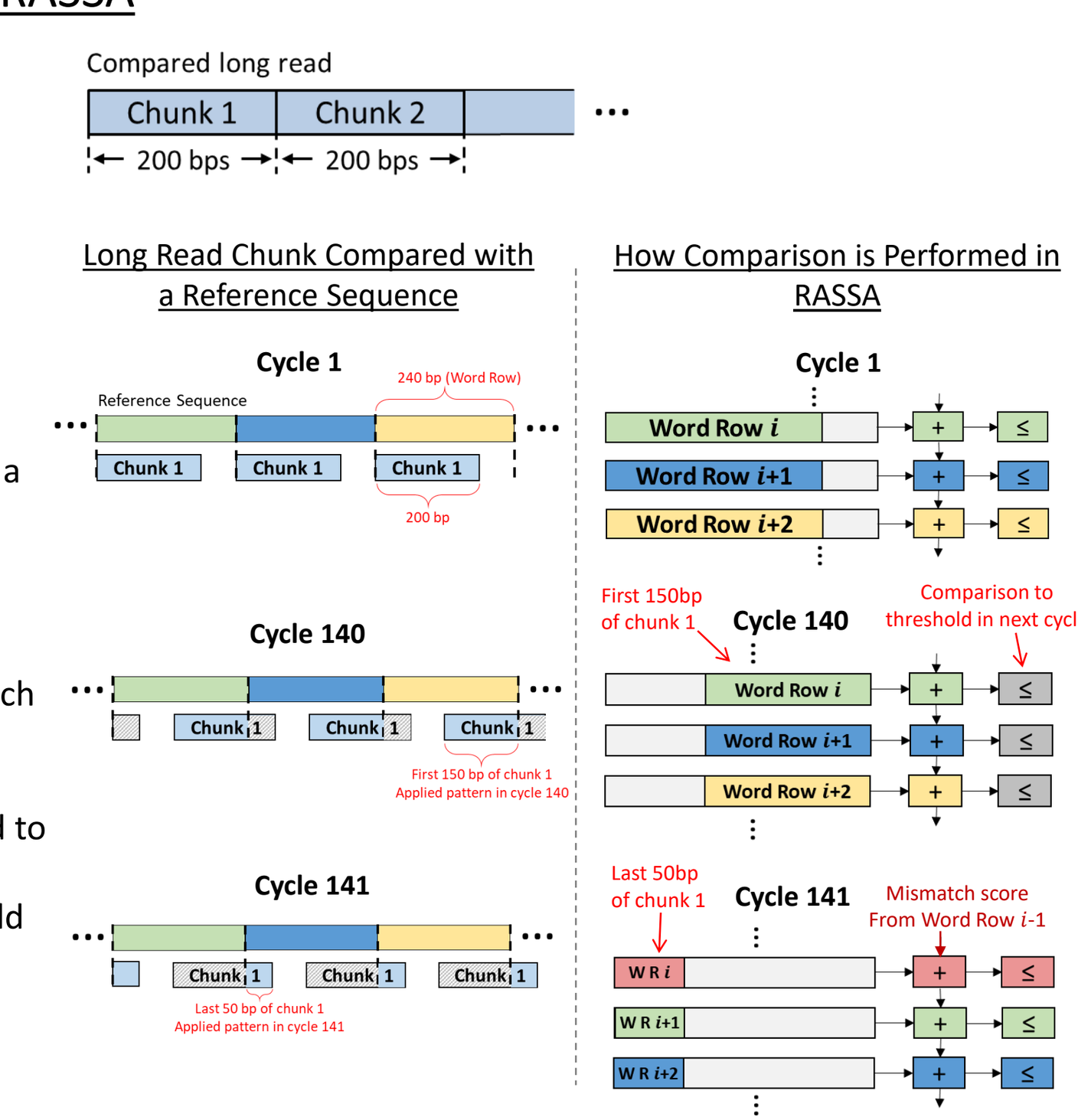
Word Row
- Contains 16 Sub-Words (240 bps)
- Connected to top and bottom Word Rows
- All mismatch values are summed *(mismatch score)*
- Two options for a mismatch score:
  1) Add mismatch score from top Word Row
  2) Compare mismatch score to a threshold

DNA Read Mapping with RASSA

The Main Idea: High-Similarity Regions → Mapping Locations
- A fixed-size chunk is compared against a long sequence
- Counting mismatch score in every position approximates the correlation between the chunk and sequence
- In the example, the read chunk is mismatch score is calculated for every position against the reference sequence
- The mapping location for the chunk is the position of a mismatch score below the threshold



Find Long Read Mapping Locations with RASSA
- Reads are divided to fixed-size chunks
  For example: 100bps / 200bps
- The threshold is set at 40-50% of the chunk length (determined empirically)
- Every chunk is compared against the entire reference sequence (stored in RASSA)
- Chunk with mismatch score below a threshold signals a mapping location
- When chunk is split between two Word Rows, mismatch score for every part is found in separate cycles
  - Two consecutive Word Rows are needed
  - Mismatch score from top Word Row is transmitted to the bottom Word Row
  - Bottom Word Row sums and compares to threshold



Evaluations

Chip Parameters, Performance and Accuracy

Chip Parameters
- A Sub-Word circuit was designed, placed and routed using the Global Foundries 28nm CMOS High-k Metal Gate library for:
  - Transistor sizing
  - Timing
  - Power analysis
- Spectre simulations for the FF and SS corners at 70°C and nominal voltage

Parameter	Value
DNA bps per row (bits)	240 (960)
Words per chip	$131k$ (2^{17})
Memory size (DNA bps)	$31.5M$
Frequency	$1GHz$
Single chip power	235W
Single chip area	$209mm^2$

Performance and Accuracy Comparison with minimap2
- Two organism reference sequences were used: e.coli and yeast
- Input sequences from Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT)
  - For PacBio: regular and high-quality Circular Consensus Sequencing (CCS) reads
- Accuracy and performance compared to minimap2, a state-of-art long read mapper
  - **Executing platform:** 16-core Intel Xeon E5-2650 @2GHz w/ 64GB of RAM
  - Minimap2 is was executed active SIMD extensions and multi-threading enabled
  - Sensitivity: % of reads found by RASSA from those found by minimap2

Reference Sequences

E.coli K-12 MG1655: 4.6Mbp

S.cerevisiae (Yeast) W303: 11.7Mbp

Datasets: #reads, avg len

e.coli PacBio: 100k, 5.2kbp

e.Coli CCS: 260k, 940bp

e.Coli ONT: 165k, 9kbp

Yeast PacBio: 100k, 6.3kbp

Yeast ONT: 30k, 11.3kbp

		200bp chunk			100bp chunk		
		Sensitivity	False Positives	Speedup	Sensitivity	False Positives	Speedup
e.coli	PacBio	79.3%	13.4%	25×	83.2%	13.6%	16×
	PacBio CCS	96.3%	8.9%	43×	96.2%	6.9%	24×
	ONT	88.8%	10.5%	48×	87.6%	12.4%	31×
Yeast	PacBio	69.8%	8.7%	77×	72%	11.8%	51×
	ONT*	85.9%	34.9%	31×	85.1%	39.2%	49×

* minimap2 mapped only 20% of all reads, with 50% of mappings with lower quality score than 60 (indicates a high-confidence mapping). RASSA sensitivity = % of reads mapped from the entire dataset. False positives = % of mapped reads with two or more mapping locations.

Performance Comparison with FPGA
- Gatekeeper [1], a pre-alignment FPGA accelerator
  - Counts number of mismatches between short reads and a reference sequence
  - Implemented in a Virtex-7 FPGA using Xilinx VC709 board, running @250MHz
  - Host machine uses 3.6GHz Intel i7-3820 CPU w/ 8GB of RAM
- Comparison of RASSA vs. GateKeeper throughput
  - Throughput measured in Billion Evaluated Mappings Locations per second (BEML/s)
  - GateKeeper results were taken from [1], RASSA results are normalized to 250MHz

RASSA vs. GateKeeper Throughput Comparison

Read Lengths	GateKeeper	RASSA @250MHz
100bp	1.7 BEML/s	226.8 BEML/s
200bp	-	175.2 BEML/s
300bp	0.2 BEML/s	142.8 BEML/s

[1] Alser, M., Hassan, H., Xin, H., Ergin, O., Mutlu, O. and Alkan, C. "GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping." Bioinformatics, vol. 33, no. 21, pp. 3355-3363, 2017.