FindeR: Accelerating FM-Index-based Exact Pattern Matching in Genomic Sequences through ReRAM Technology

Farzaneh Zokaee and Lei Jiang

Indiana University Bloomington

3th HPCA Workshop on ACCELERATOR ARCHITECTURE IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

Executive summary

1. **Designing PIM: for genome sequence analysis**
   - Read alignment uses FM-Index algorithm to find exact locations of reads in reference genome.
2. **Problems:**
   - Accessing and finding exact matches for huge amount of generated reads by FM-Index (Billions of reads).
3. **Proposed solutions: speeding up FM-Index**
   - FindeR: ReRAM-based process-in-memory architecture
   - Remove cost of data transferring between cpu and memories
   - Hardware/algorithm co-design → operation parallelism ↑
4. **Results:**
   - Throughput: **83% ~ 30k×** over the state-of-the-art.
   - Throughput/power : **3.5× ~ 42.5k×** over the state-of-the-art.

Genome sequencing pipeline

organic DNA

A T

C G

~3.2B bps

Illumina

Nanopore

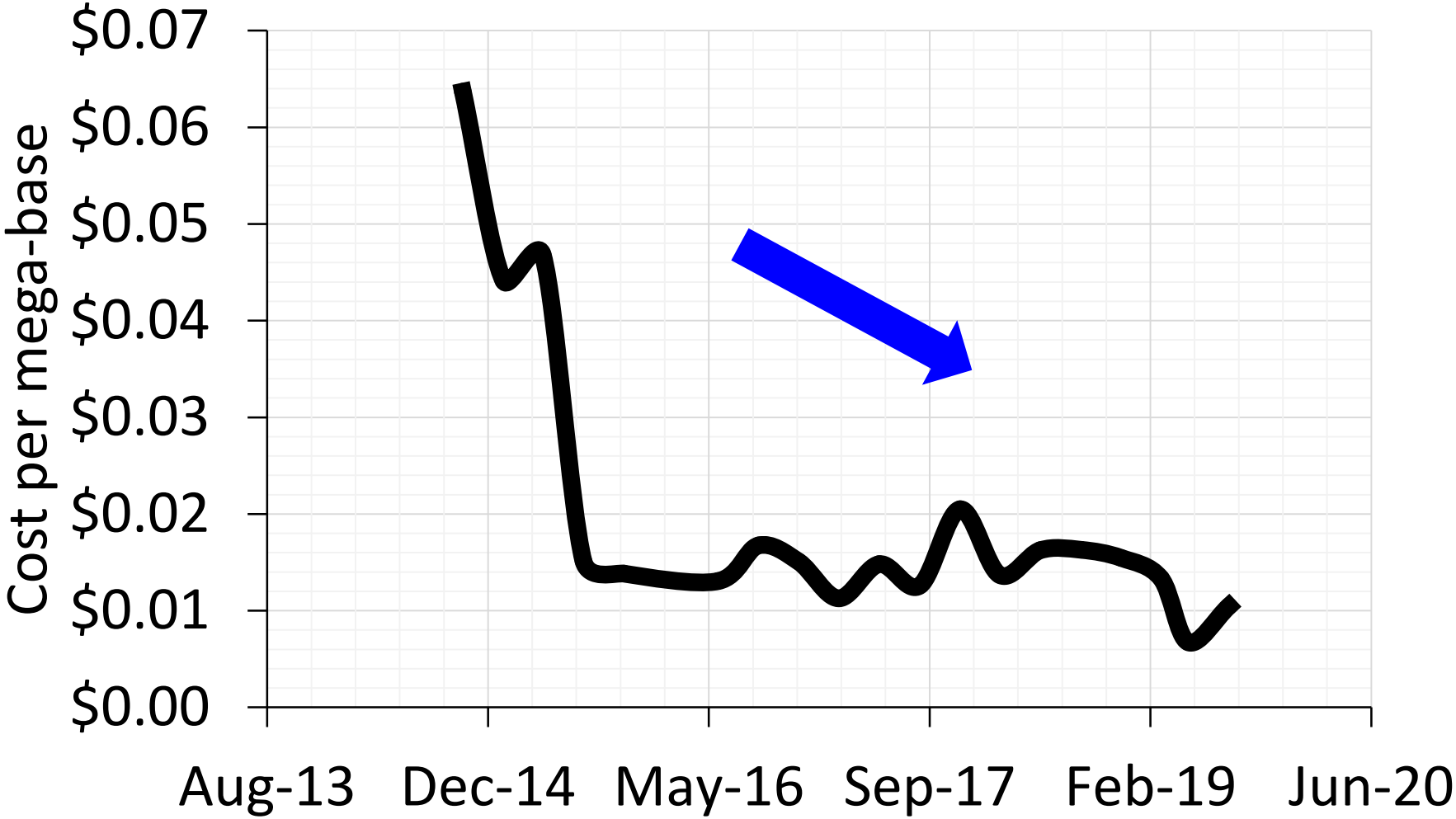
PacBio

CCCCCTATATATACGTACTAGTACGT
ACGACTTTAGTACGTACGT
TATATATACGTACTAGTACGT
ACGTACGCCCCTACGTA
TATATATACGTACTAGTACGT
ACGACTTTAGTACGTACGT
TATATATACGTACTAAAGTACGT
TATATATACGTACTAGTACGT
ACGTTTTTAAAACGTA
TATATATACGTACTAGTACGT
ACGACGGGGAGTACGTACGT

Illumina HiSeq2000: short reads (100 bp) with error rate 1%

PacBio and Nanopore: long reads (1k bp) with error rate 15-40%

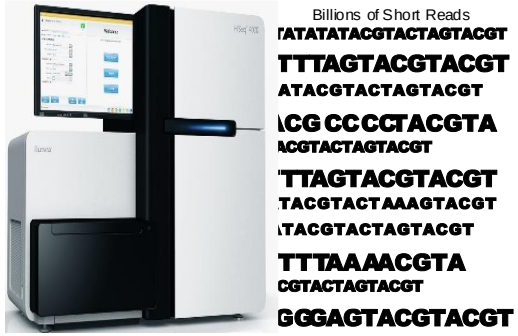
Genome sequencing cost decreases



[Wetterstrand_GSP'19] available at www.genome.gov/sequencingcostsdata

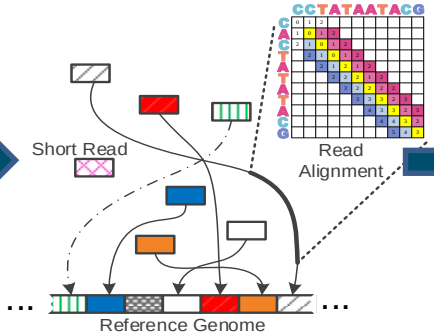
Genome sequencing pipeline

1 Sequencing



Illumina HiSeq2000

2 Read Alignment



3 Variant Calling

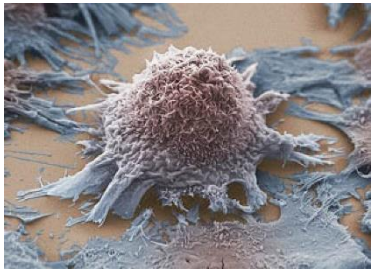
```
reference: TTTATCGCTTCCATGACGCAG
read1:         ATCGCATCC
read2:       TATCGCATC
read3:             CATCCATGA
read4:           CGCTTCCAT
read5:                 CCATGACGC
read6:             TTCCATGAC
```

4 Discovery

The pipeline latency matters!

Genome sequencing for profiling tumor

- Variants → prioritize anti-cancer therapy and direct patient management



which type?

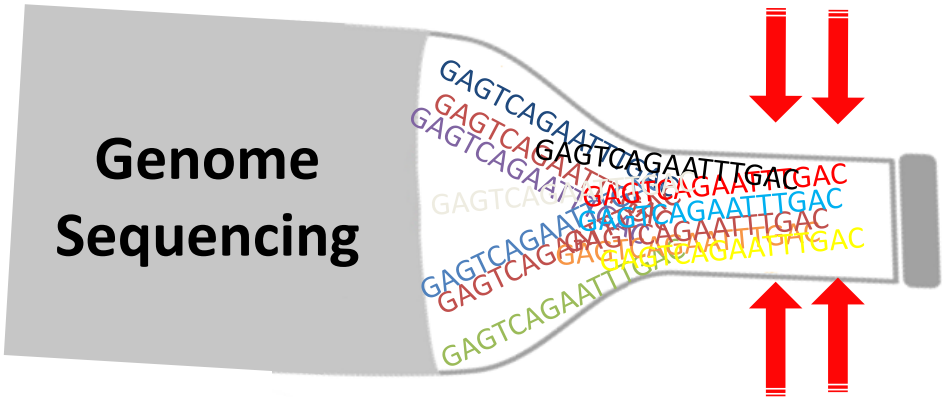
life or death?

Such a test takes

several days to weeks!!!

[MolecularTesting_2019] available at www.mycancergenome.org/content/page/molecular-testing

Bottleneck in genome sequencing pipeline



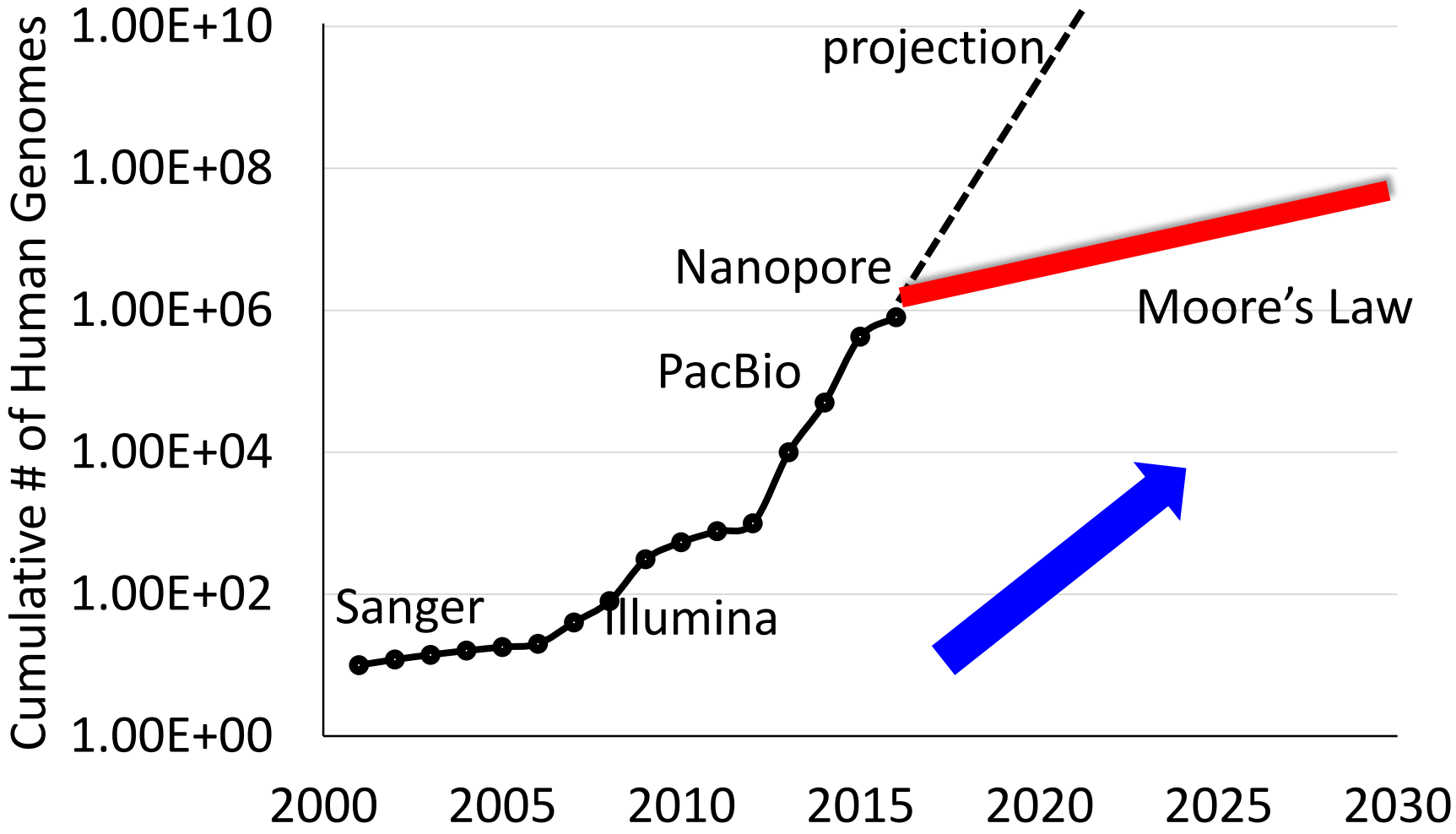
Read Alignment

300 Million bases/minute

2 Million bases/minute

Bottlenecked in Alignment!!

The explosion in the genomic data capacity



[Stephens_PLoSBiol2015]

Read alignment

Reference

A T C C G TA C A G A T T T T T C C A T C C G T A

Reads

C G T A
A A G A
T T C A
C A T A

Read alignment

Reference

A T C C G TA C A G A T T T T T C C A T C C G T A

C G T A

Reads

hit

A A G A

T T C A

C A T A

Read alignment

Reference

A T C C G T A C A G A T T T T T C C A T C C G T A

C G T A A A G A

Reads

hit

insert

T T C A

C A T A

Read alignment

Reference

A T C C G T A C A G A T T T T T C C A T C C G T A

Reads

C G T A A A G A T T C A

hit insert delete

C A T A

Read alignment

Reference

A T C C G T A C A G A T T T T T C C A T C C G T A

Reads

C G T A — hit

A A G A — insert

T T C A — delete

C A T A — substitute

Read alignment

Reference

A T C C G T A C A G A T T T T T C C A T C C G T A

Reads

C G T A — hit

A A G A — insert

T T C A — delete

C A T A — substitute

Read alignment Seed-and-Extend : **Seeding** Find exact matches (FM-Index) → **Seed extension** Find inexact matches

Seeding is slow due to FM-Index search algorithm.

Burrows-wheeler transform

Ref: A T C C G T $

0	A	T	C	C	G	T	$
1	T	C	C	G	T	$	A
2	C	C	G	T	$	A	T
3	C	G	T	$	A	T	C
4	G	T	$	A	T	C	C
5	T	$	A	T	C	C	G
6	$	A	T	C	C	G	T

Burrows-wheeler transform

Ref: A T C C G T $

0 A T C C G T $
1 T C C G T $ A
2 C C G T $ A T
3 C G T $ A T C
4 G T $ A T C C
5 T $ A T C C G
6 $ A T C C G T

i	SA	BWT
0	6	$ A T C C G T
1	0	A T C C G T $
2	2	C C G T $ A T
3	3	C G T $ A T C
4	4	G T $ A T C C
5	5	T $ A T C C G
6	1	T C C G T $ A

Burrows-wheeler transform

Ref: A T C C G T $

0 A T C C G T $
1 T C C G T $ A
2 C C G T $ A T
3 C G T $ A T C
4 G T $ A T C C
5 T $ A T C C G
6 $ A T C C G T

i	SA	BWT
0	6	$ A T C C G T
1	0	A T C C G T $
2	2	C C G T $ A T
3	3	C G T $ A T C
4	4	G T $ A T C C
5	5	T $ A T C C G
6	1	T C C G T $ A

BWT: T $ T C C G A

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	0	1	0	2
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

Count

A	C	G	T
1	2	4	5

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

0 1 2 3 4

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	**0**	**1**	**0**	**2**
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

Count

A	C	G	T
1	2	4	5

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

0 1 2 3 4

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	**0**	**1**	**0**	**2**
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

Count

A	C	G	T
1	2	4	5

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

0 1 2 3 4

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	**0**	**1**	**0**	**2**
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

Count

A	C	G	T
1	2	4	5

8 entries!

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

0 1 2 3 4

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	**0**	**1**	**0**	**2**
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

Count

A	C	G	T
1	2	4	5

8 entries!

0

A	C	G	T
1	2	4	5

1

A	C	G	T
1	3	4	7

tag

FM-Index

Ref: A T C C G T $

BWT: T $ T C C G A

0 1 2 3 4

Occ(*S*, i)

i	A	C	G	T
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
3	0	0	0	2
4	**0**	**1**	**0**	**2**
5	0	2	0	2
6	0	2	1	2
7	1	2	1	2

8 entries!

Count

A	C	G	T
1	2	4	5

0

A	C	G	T
1	2	4	5

tag

BWT

T	$	T	C

BWT

1

A	C	G	T
1	3	4	7

BWT

C	G	A	-

2 entries

Backward search

Ref: A T C C G T $ **Query: C G T** **BWT: T $ T C C G A**

```
00 BackwardSearch(BWT, Q){
01   int low = 0;
02   int high = max_occ;
03   for (int i = len; i >= 0; i--){
04    low = LFM(BWT[low/4], Q[i], low);
05    high=LFM(BWT[high/4], Q[i], high);
06     if (low >= high) return;
07   }
08 }
```

tag			
A	C	G	T
1	2	4	5
BWT			
T	$	T	C

tag			
A	C	G	T
1	3	4	7
BWT			
C	G	A	-

```
09 int LFM(BWT[x/4], Q[index], x){
10   int co = 0;
11   int tag = TAG[Q[index]];
12   for (int j = 0; j < x % 4; j++)
13     if (BWT[x/4][j] == s) co ++;
14   return co + tag;
15 }
```

Problem: operations in backward search

- **Random** memory accesses due to pointer chasing

```
low = LFM(BWT[low/4], Q[i], low);
high=LFM(BWT[high/4], Q[i], high);
```

Processing-in-memory!

Problem: operations in backward search

- **Random** memory accesses due to pointer chasing

```
low = LFM(BWT[low/4], Q[i], low);
high=LFM(BWT[high/4], Q[i], high);
```

Processing-in-memory!

- **Counting** a symbol *S* in a string

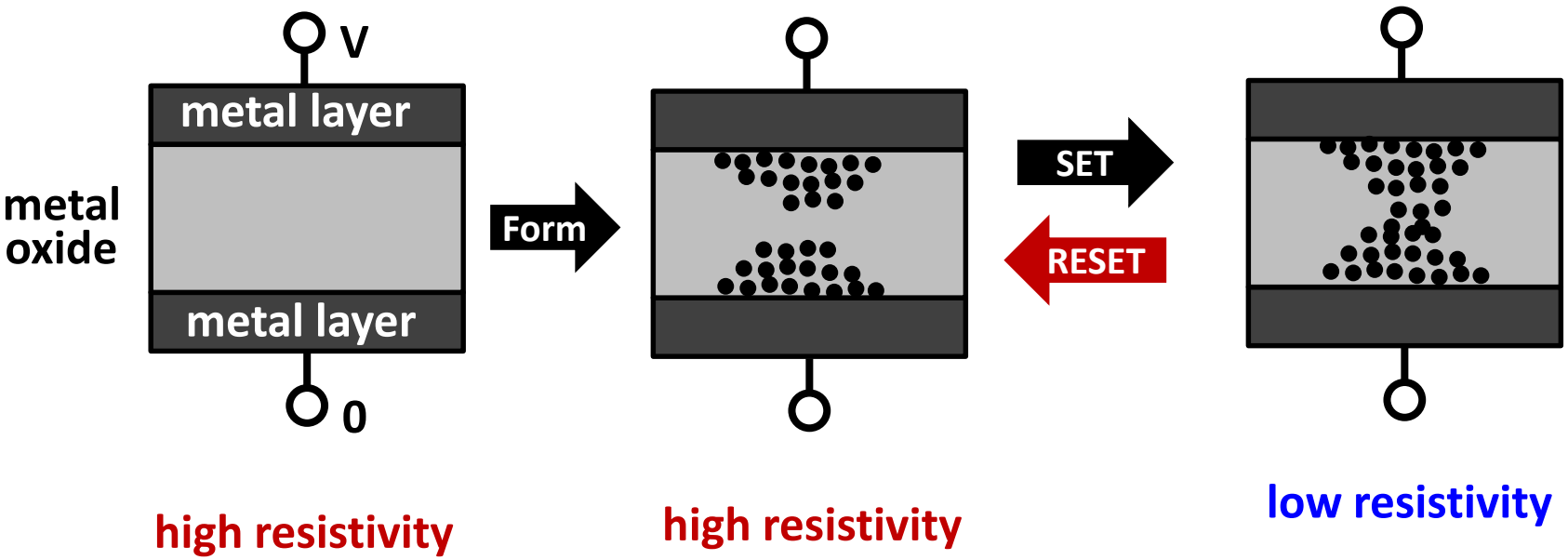
```
for (int j = 0; j < x % 4; j++)
  if (BWT[x/4][j] == s) c ++;
```

Hamming distance between **"SSSSS"** and the string

Hardware/algorithm co-design → operation parallelism ↑

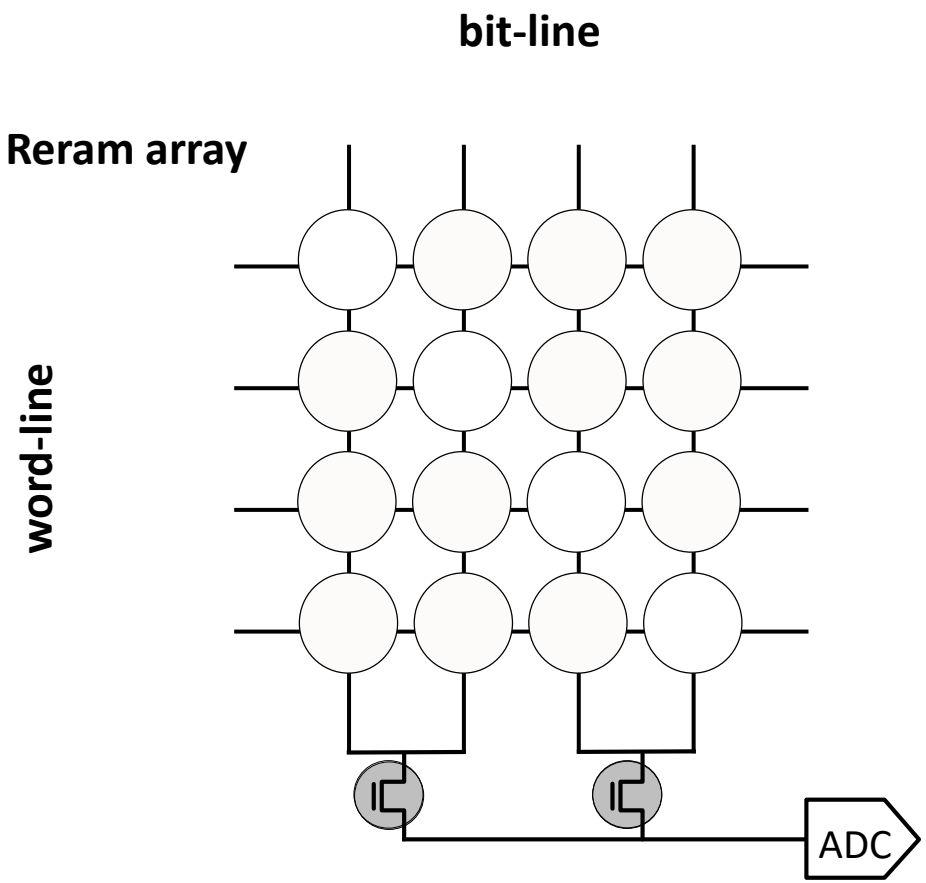
Solution: ReRAM Hamming Distance Unit

ReRAM basics



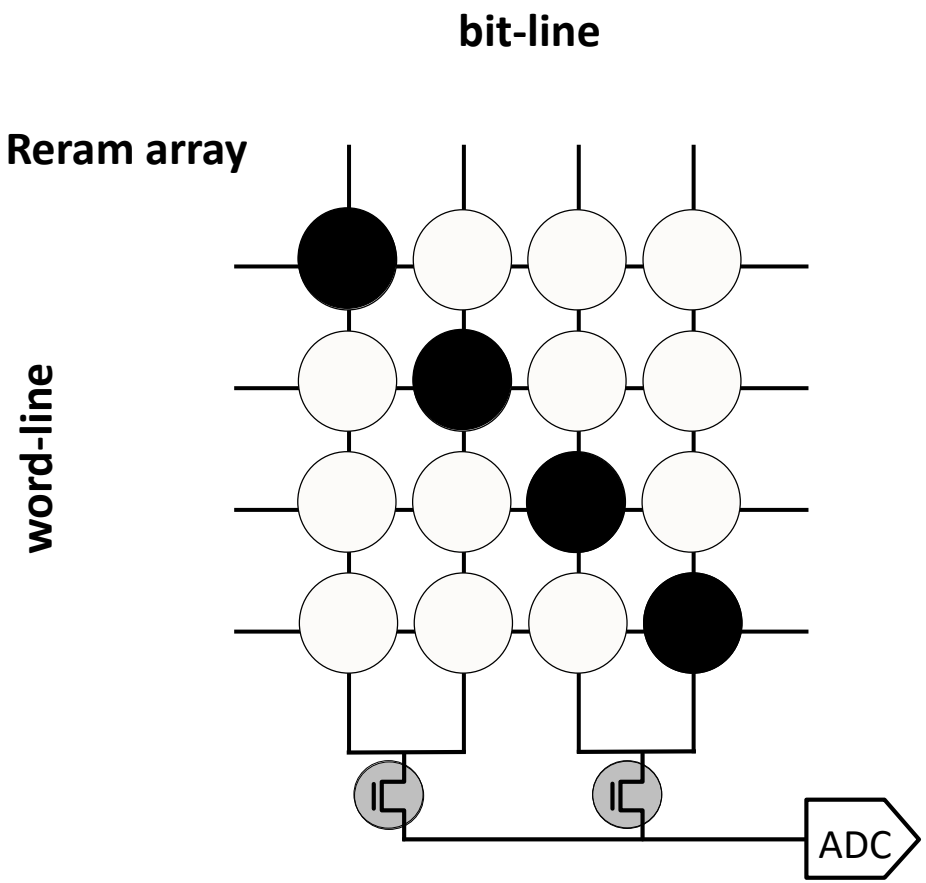
ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "GG" and "CG"



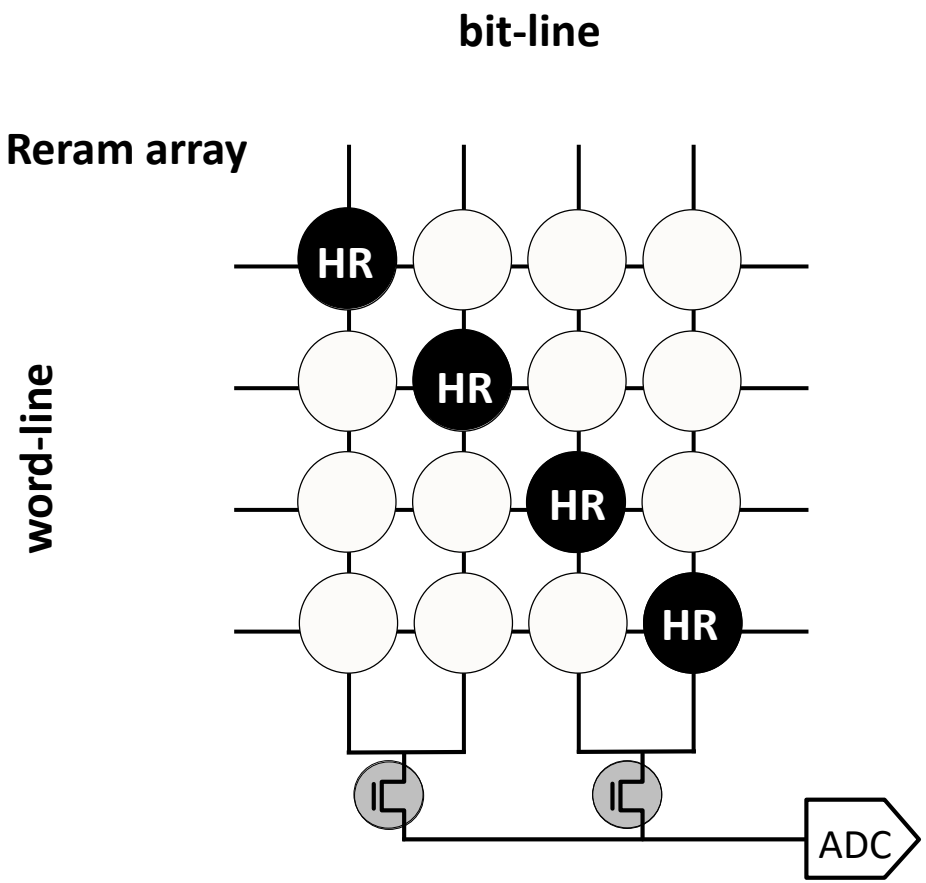
ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "**GG**" and "**CG**"



ReRAM-based Hamming Distance Unit

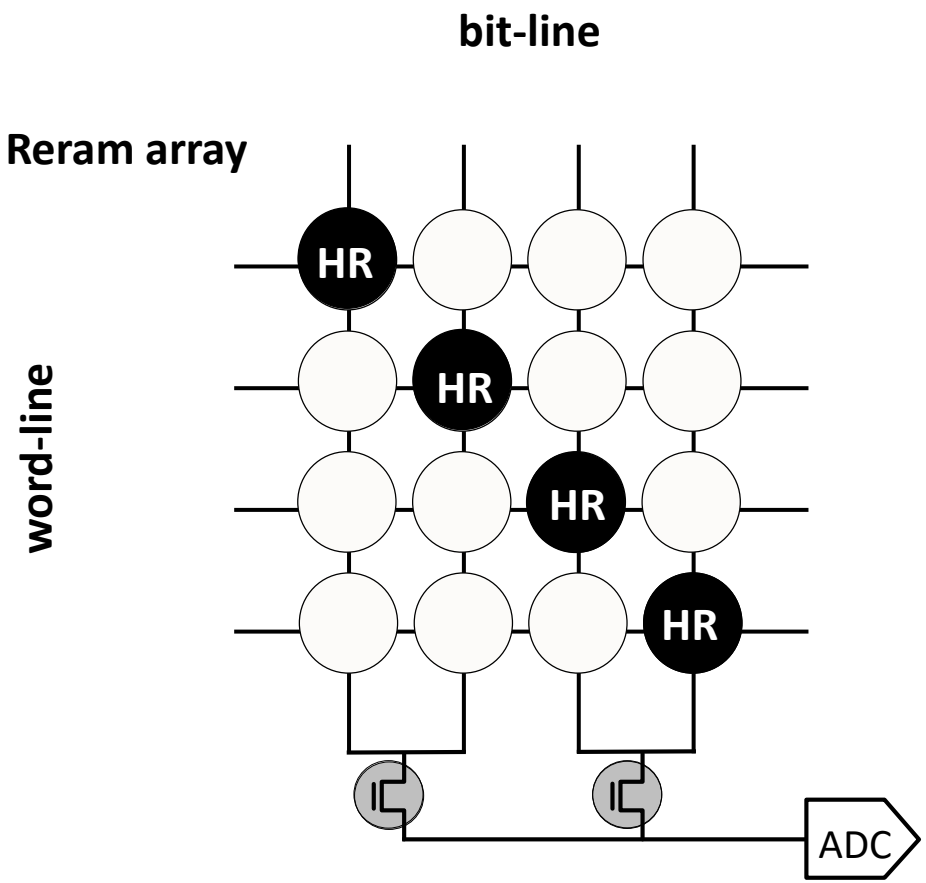
Counting *G* in "CG", Hamming distance between "**GG**" and "**CG**"



ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "**GG**" and "**CG**"

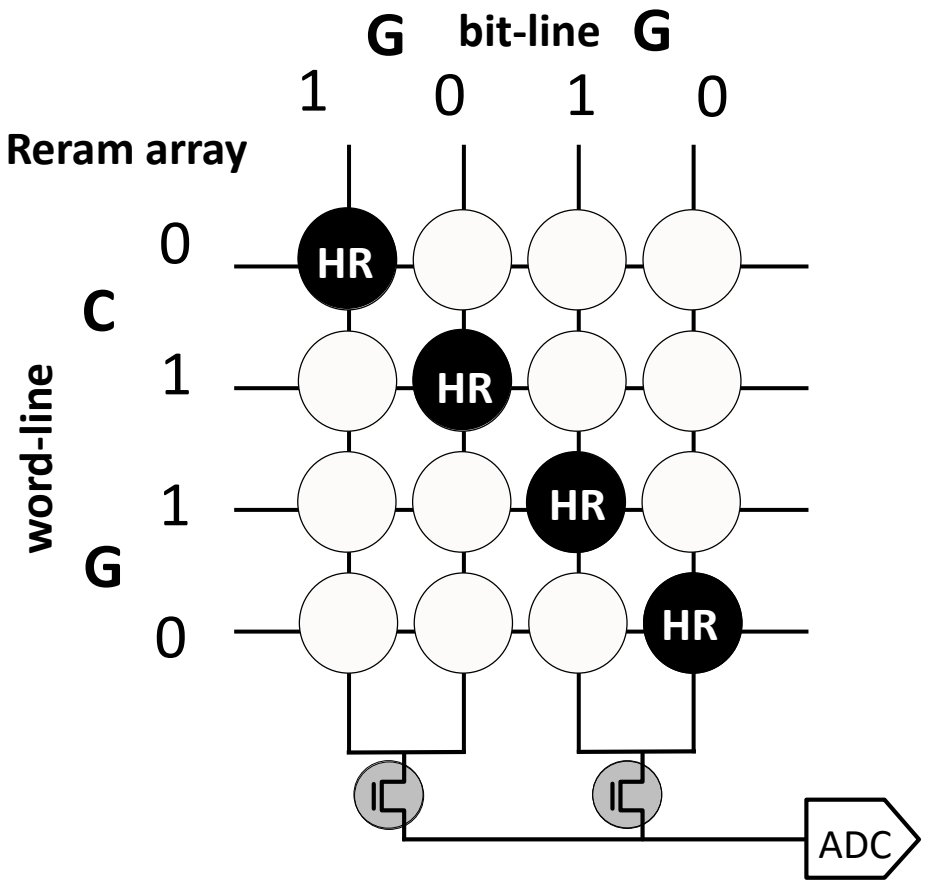
A : 00
C : 01
G : 10
T : 11



ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "**GG**" and "**CG**"

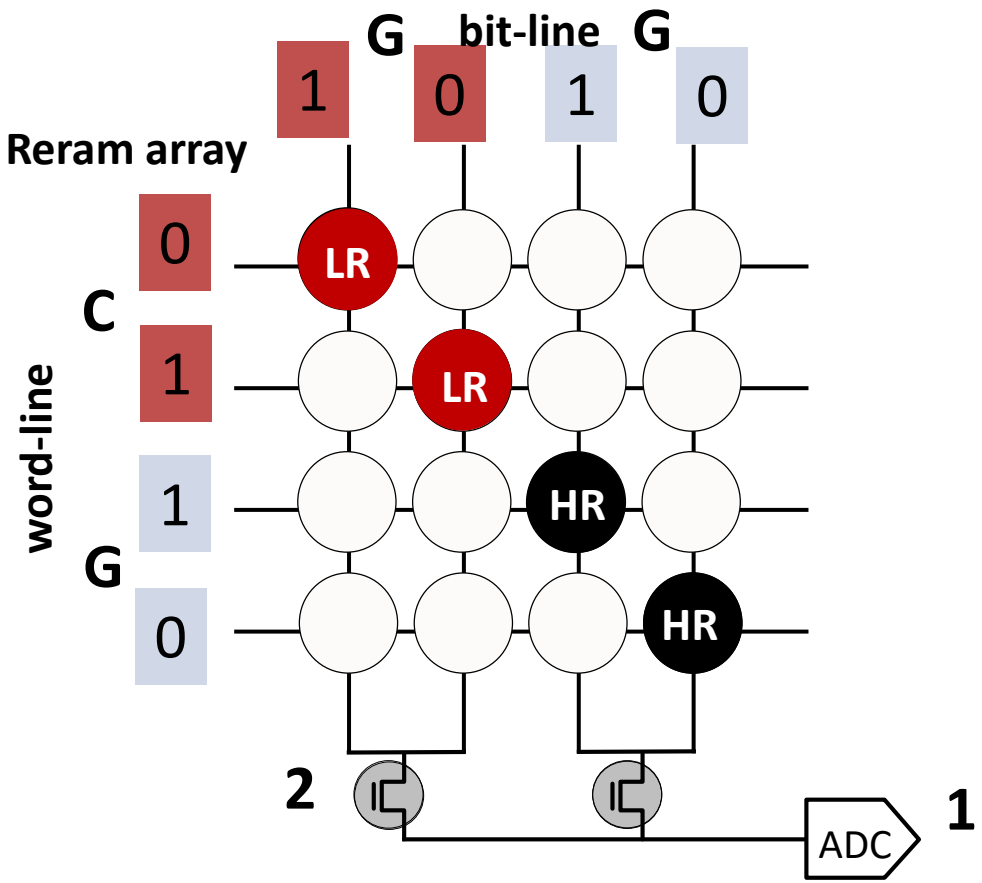
A : 00
C : 01
G : 10
T : 11



ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "GG" and "CG"

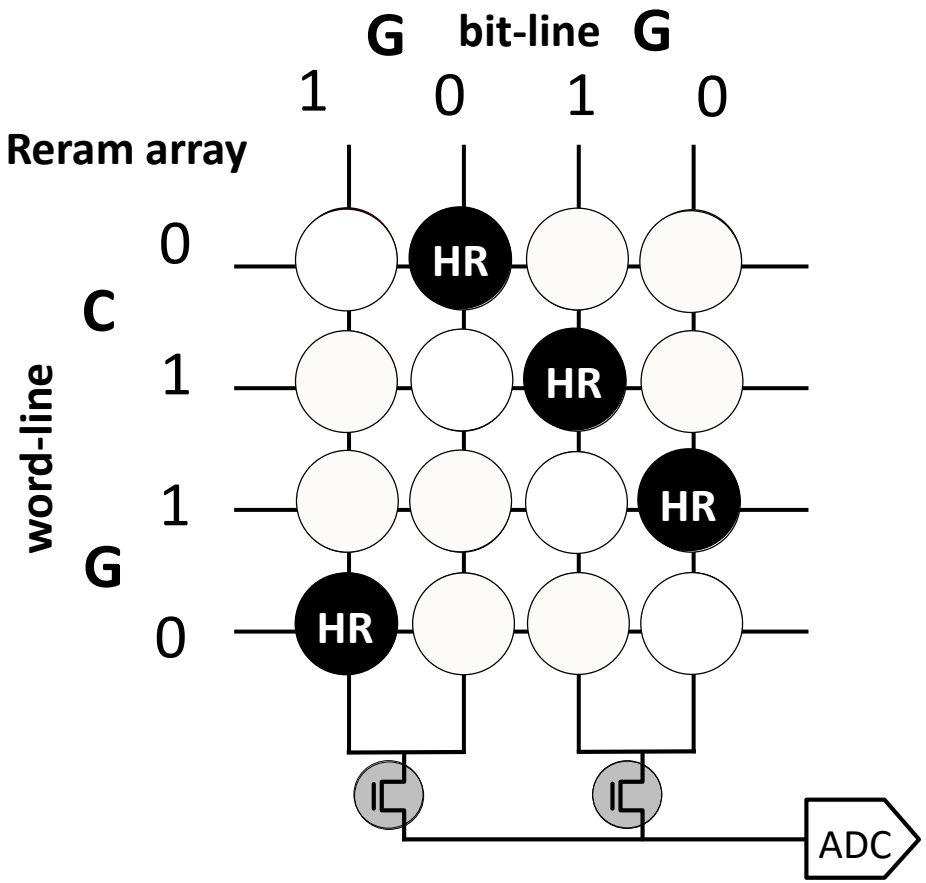
A : 00
C : 01
G : 10
T : 11



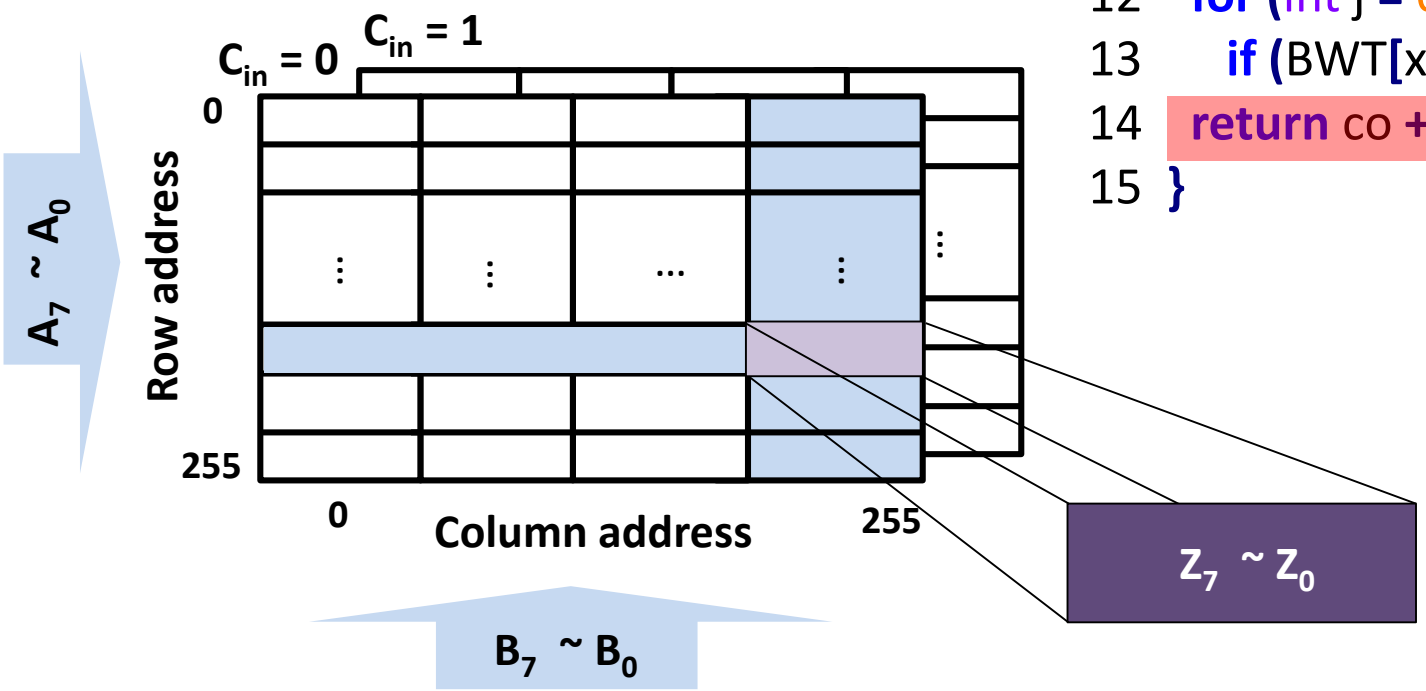
ReRAM-based Hamming Distance Unit

Counting *G* in "CG", Hamming distance between "**GG**" and "**CG**"

A : 00
C : 01
G : 10
T : 11

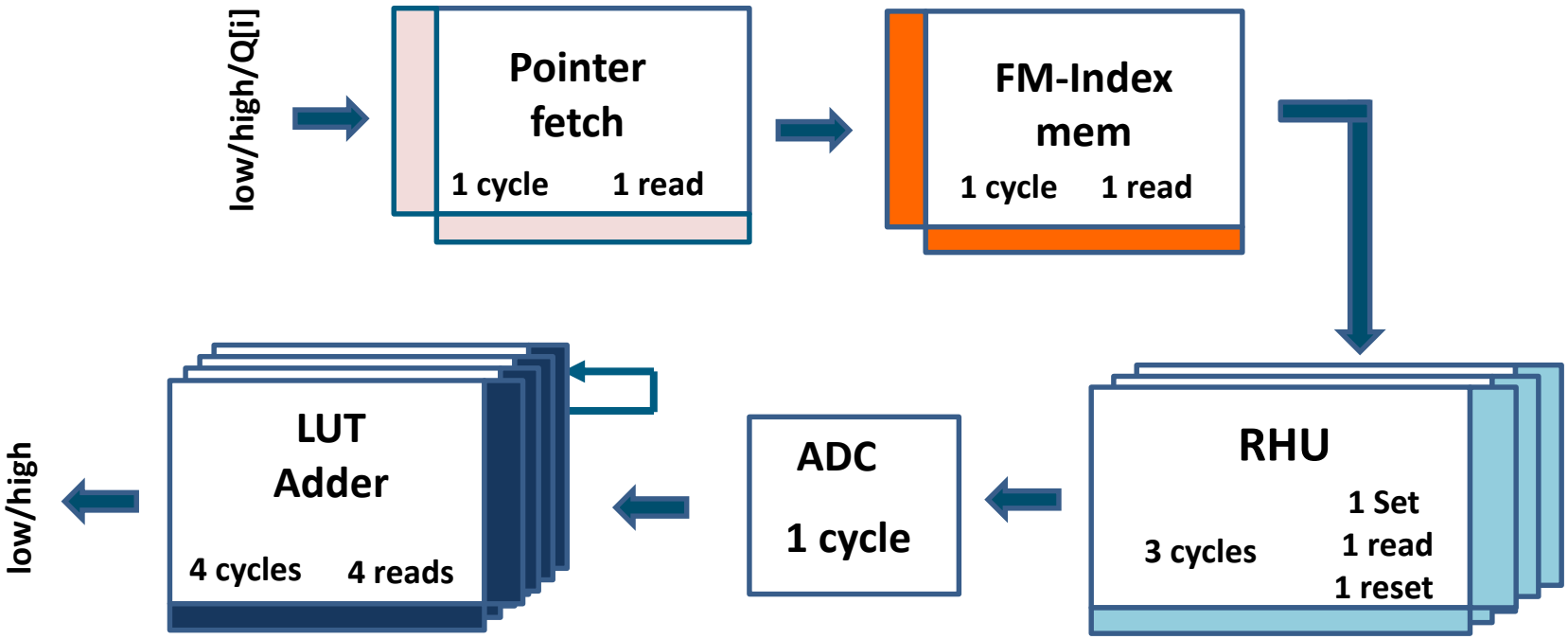


ReRAM Lookup Table-based Adder




```
int LFM(BWT[x/4], Q[index], x){
  int co = 0;
  int tag = TAG[Q[index]];
  for (int j = 0; j < x % 4; j++)
    if (BWT[x/4][j] == s) co ++;
  return co + tag;
}
```

Pipeline Design



Results

Short read alignment

	CPU	GPU	ASIC	FPGA	FindR
Die size (mm^2)	14.3K	1.6K	352	14.8K	1.1K
Main memory(GB)	128	6	1.3	48	0
Power(W)	130	258	3.1	247	9.09 ✓
Throughput	68K	150K	379K	1.5M	10.07M ✓
Throughput/Watt	523	581	121K	6.2K	1.18M ✓

- **FindR** improves throughput by **10x** over **FPGA**
- **FindR** improves throughput/watt by **9.75x** over **ASIC**

Long Read Seeding

	Performance		Quality	
	Throughput	Throughput/Watt	Sensitivity	Precision
Darwin-PAC	**3.9K** ✓	**0.26K**	**99.71%**	**99.91%**
Darwin-ONT			**98.2%**	**99.1%**
FindR-PAC	**2.9K**	**1.64K** ✓	**95.95%**	**95.95%**
FindR-ONT			**98.11%**	**99.10%**

× **Darwin** uses the power hungry SRAM buffers

- **FindR** improves Throughput/Watt by **5.3x**

Long Read Seeding

	Performance		Quality	
	Throughput	Throughput/Watt	Sensitivity	Precision
Darwin-PAC	**3.9K** ✓	**0.26K**	**99.71%**	**99.91%** ✓
Darwin-ONT			**98.2%**	**99.1%**
FindR-PAC	**2.9K**	**1.64K** ✓	**95.95%**	**95.95%**
FindR-ONT			**98.11%**	**99.10%**

- **FindeR** improves quality by using FM-Index-based **error correction technique** with SMEM seeding

Long Read Seeding

	Performance		Quality	
	Throughput	Throughput/Watt	Sensitivity	Precision
Darwin-PAC	**3.9K** ✓	**0.26K**	**99.71%**	**99.91%**
Darwin-ONT			**98.2%**	**99.1%**
FindR-PAC	**2.9K**	**1.64K** ✓	**95.95%**	**95.95%**
FindR-ONT			**98.11%**	**99.10%**

- **FindeR** improves quality by using FM-Index-based **error correction technique** with SMEM seeding

FindR-PAC	**0.87K**	**0.49K** ✓	**99.8%**	**99.95%**
FindR-ONT			**98.31%**	**99.23%**

✓

Short Read Alignment

	Hash Table		Dynamic		Automata	FM-Index
	RADAR	BioCAM	Race	RCAM	GenAx	FindR
Die size (mm^2)	120	9.8K	450	383	4.6K	1.1K
Off-chip memory(GB)	0	0	8	0	120	0
Function	Seeding		Seed Extension			Both
Power(W)	12.5	153	24.3	6.6K	20	9.09 ✓
Throughput	125	186.8K	2.1M	177K	973	3.86M ✓
Throughput/Watt	10	1.2K	86K	26	48.65	424.6K ✓

- **FindR** improves throughput by **83% ~ 30Kx**
- **FindR** improves throughput/watt **by 3.5x ~ 42.5Kx**

Executive summary

1. **Designing PIM: for genome sequence analysis**
   - Read alignment uses FM-Index algorithm to find exact locations of reads in reference genome.
2. **Problems:**
   - Accessing and finding exact matches for huge amount of generated reads by FM-Index (Billions of reads).
3. **Proposed solutions: speeding up FM-Index**
   - FindeR: ReRAM-based process-in-memory architecture
   - Remove cost of data transferring between cpu and memories
   - Hardware/algorithm co-design → operation parallelism ↑
4. **Results:**
   - Throughput: **83% ~ 30k×** over the state-of-the-art.
   - Throughput/power : **3.5× ~ 42.5k×** over the state-of-the-art.

FindeR: Accelerating FM-Index-based Exact Pattern Matching in Genomic Sequences through ReRAM Technology

Farzaneh Zokaee and Lei Jiang

Indiana University Bloomington

3th HPCA Workshop on ACCELERATOR ARCHITECTURE IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS