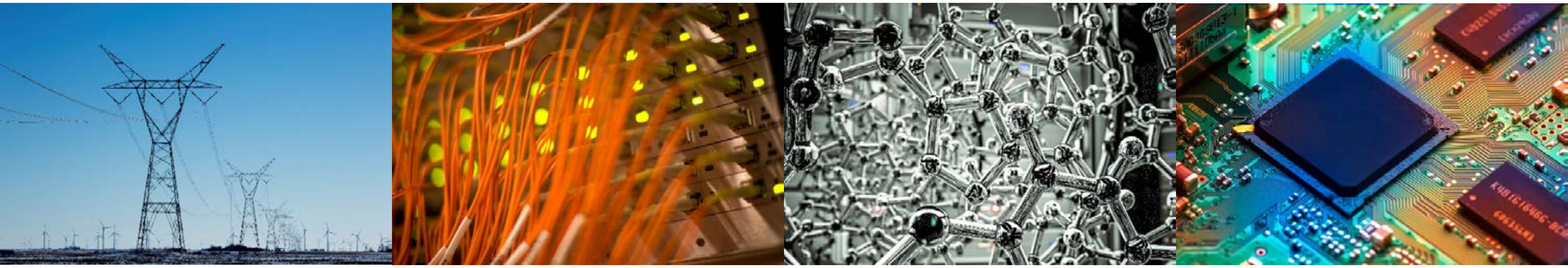
Hardware Acceleration of the Pair HMM Algorithm for DNA Variant Calling

Sitao Huang[1], Gowthami Jayashri Manikandan[1], Anand Ramachandran[1], Kyle Rupnow[2], Wen-mei W. Hwu[1], Deming Chen[1]

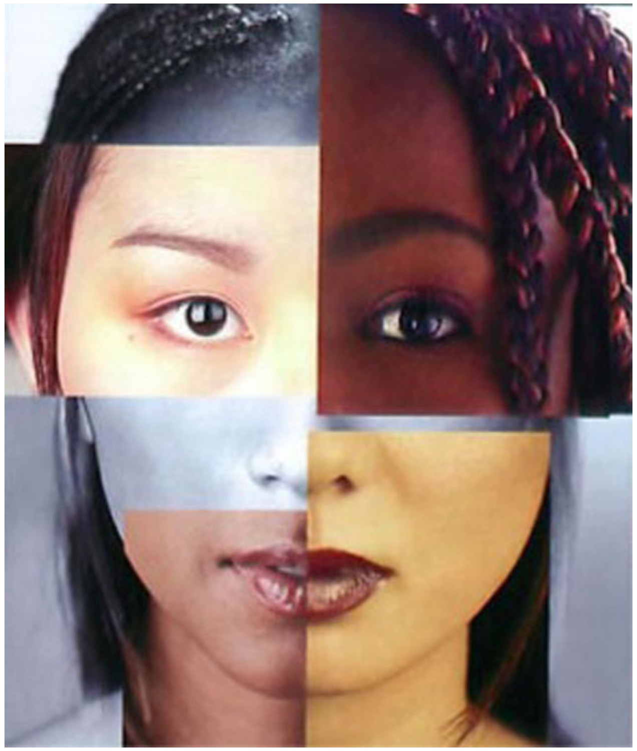
[1]University of Illinois at Urbana-Champaign, USA

[2]Advanced Digital Sciences Center, Singapore



Genomic Variation and Mutations

- Humans have two sets of 3 billion bases in their genomes
- No two humans have identical genome sequences
  - About 0.1 % of genomes are not identical
- These differences lead to people
  - Having different susceptibility or resistance to diseases
  - Responding differently to the same medication
- There are also somatic variations that lead to cancer



The Importance of Mutations and Variant Calling

- The study of mutations is important (e.g. in cancer study)
  - They create cancer
  - They enable cancer to survive
  - They enable cancer to spread
  - They enable cancer to kill
- **Variant calling is a set of analytics that tries to identify mutations in a sequenced genome compared to a standard reference**

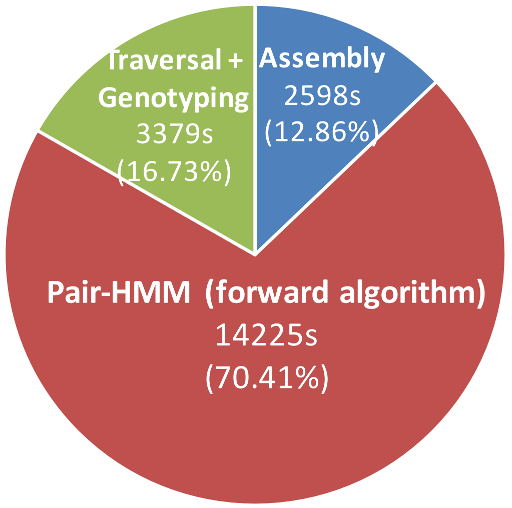
Variant Calling is critical in cancer research and clinical applications

GATK's **HaplotypeCaller** is one of the most popular variant calling tools available today.

Accelerating the Pair HMM in GATK

Why Pair HMM Needs to Be Accelerated?

- Pair HMM computations constitute the bottleneck of HaplotypeCaller
- The full HaplotypeCaller is time consuming
    - Full HaplotypeCaller run on 80xWGS PCR-Free NA12878 dataset: **13 days** on single CPU



Profiling result of a typical HaplotypeCaller run on CPU

Why Using Hardware (FPGA)?

- Parallelism in pair HMM could be better utilized by the fine-grained processing elements in FPGA
- FPGA is good at processing streaming applications (alignment algorithms' nature)

Pair HMM

(candidate to be verified)   (data from sequencing machine)

- Input: two sequences S_h and S_r (S_h: haplotype S_r: read)
- Goal: find a similarity score of S_h and S_r

S_r GTAA
S_h AGGTC

One possible alignment:

S_r	--GTAA
S_h	AGGTC-
$\{a_t\}$	IIMMMD

Another possible alignment:

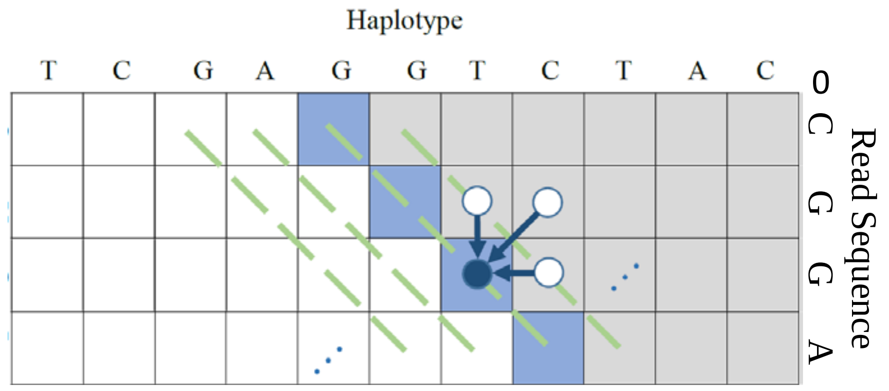
S_r	-G-TAA
S_h	AGGT-C
$\{a_t\}$	IMIMDM

There are many action sequences mapping S_r to S_h.

- Similarity score is defined over a **pair Hidden Markov Model**

Pair HMM – Dynamic Programming

- Similarly:



Coefficients from model

$$f^D(i,j) = a_{MD} f^M(i,j-1) + a_{DD} f^D(i,j-1)$$
$$f^I(i,j) = a_{MI} f^M(i-1,j) + a_{II} f^I(i-1,j)$$
$$f^M(i,j) = prior \cdot \left(a_{MM} f^M(i-1,j-1) + a_{IM} f^I(i-1,j-1) + a_{DM} f^D(i-1,j-1) \right)$$

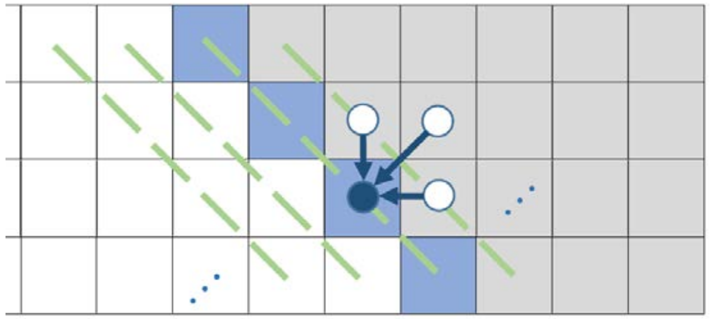
- Output: $score(S_h, S_r) = f^D(N_h, N_r) + f^M(N_h, N_r) + f^I(N_h, N_r)$
- Complexity: $O(M_h \times M_r \times N_h \times N_r)$

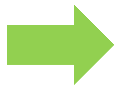
# haplotype sequences

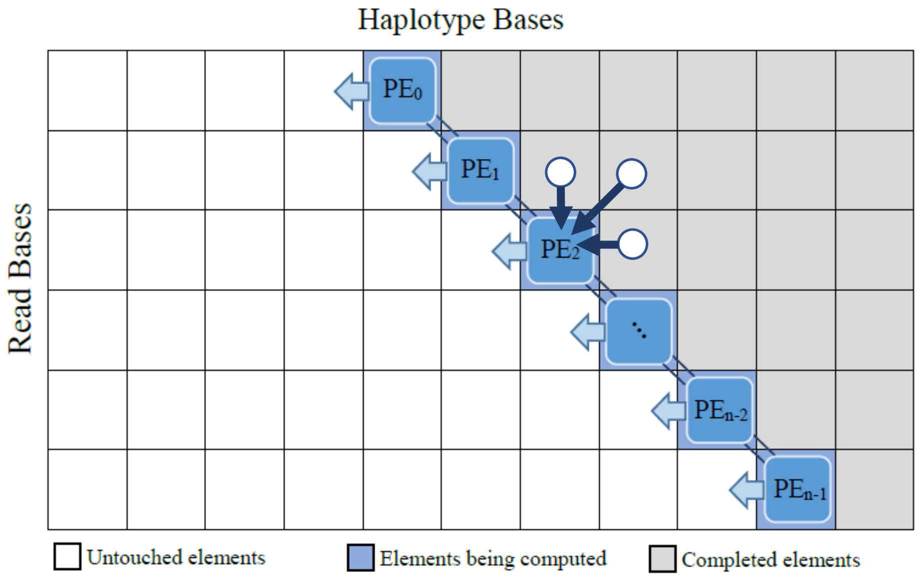
# read sequences

How to Accelerate?

PE: Processing Elements







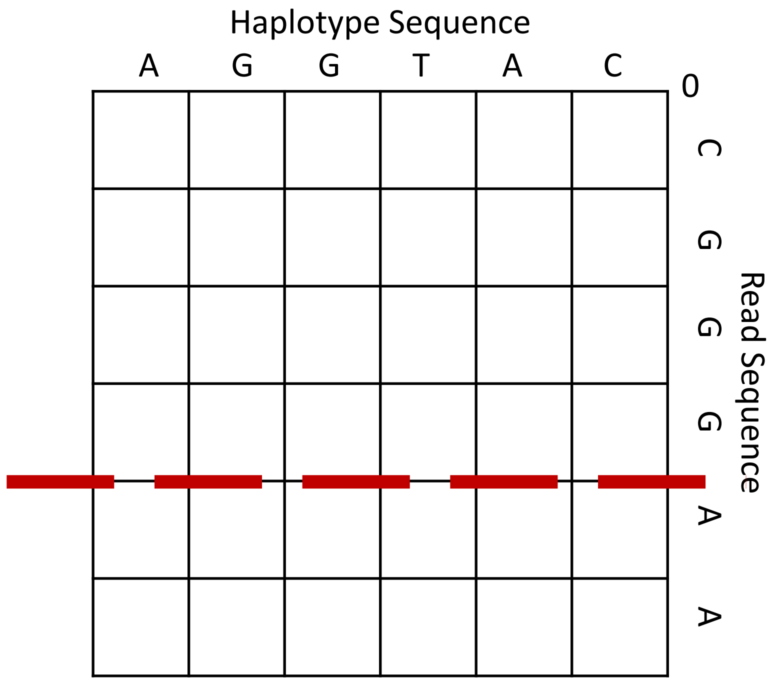
Process "frontier" elements at the same time to maximize parallelism

Number of PEs Needed = Matrix Height

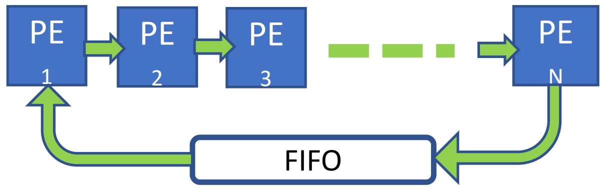
What if matrix height is larger than number of PEs FPGA can host?

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



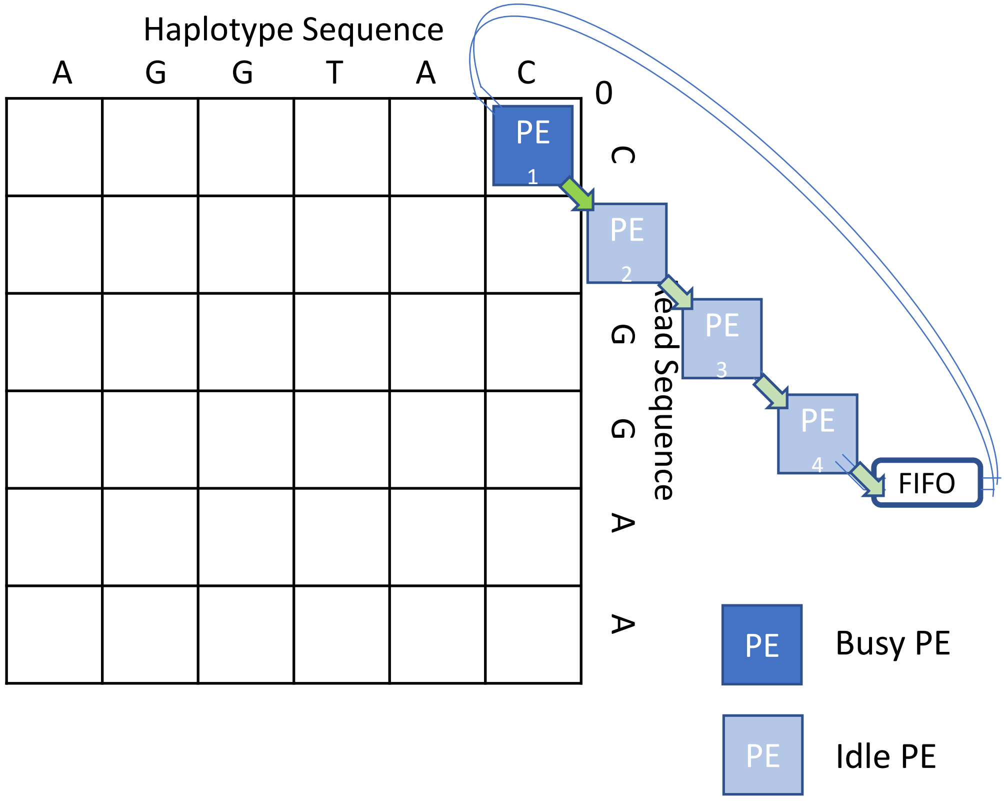
Processing Element (PE) Ring:



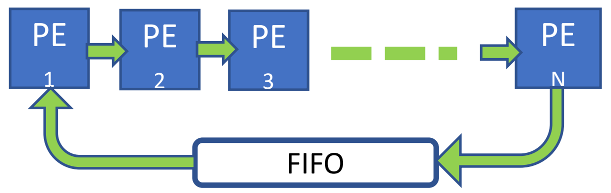
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



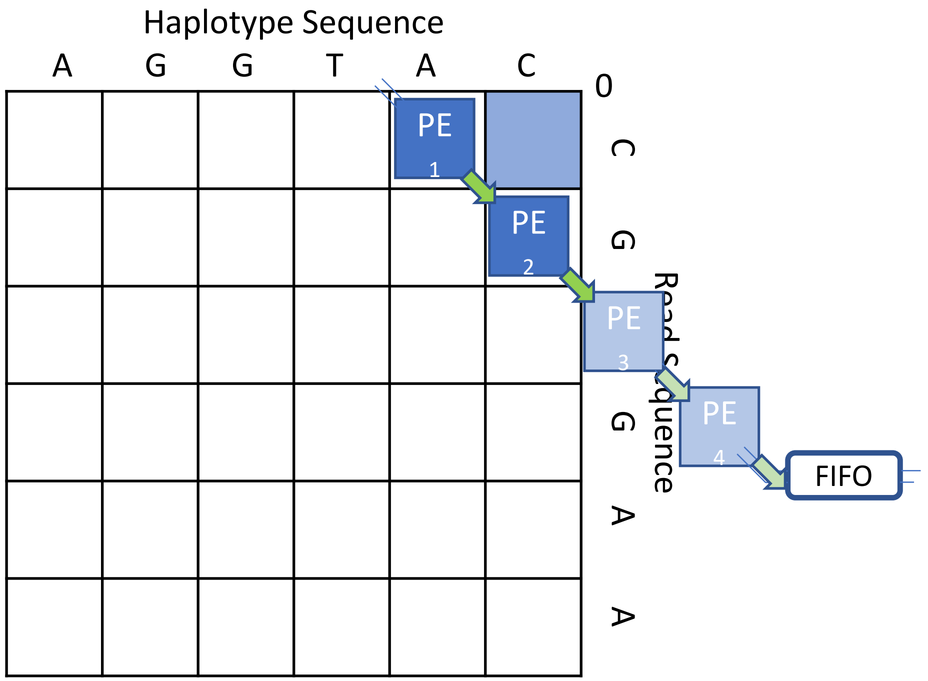
Processing Element (PE) Ring:



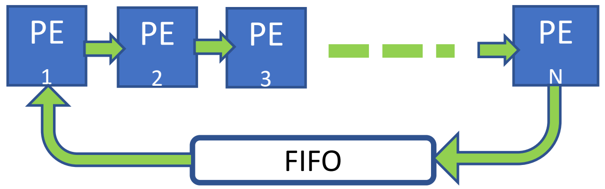
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



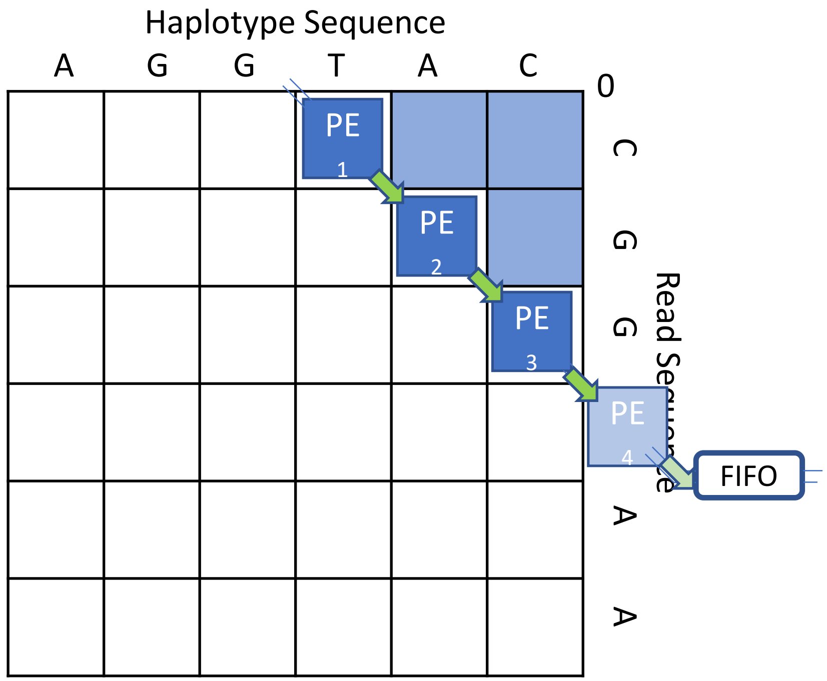
Processing Element (PE) Ring:



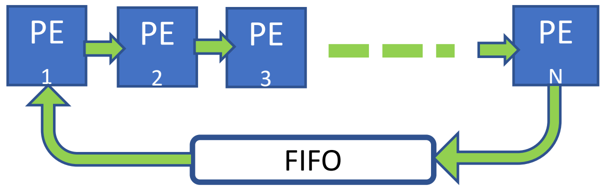
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



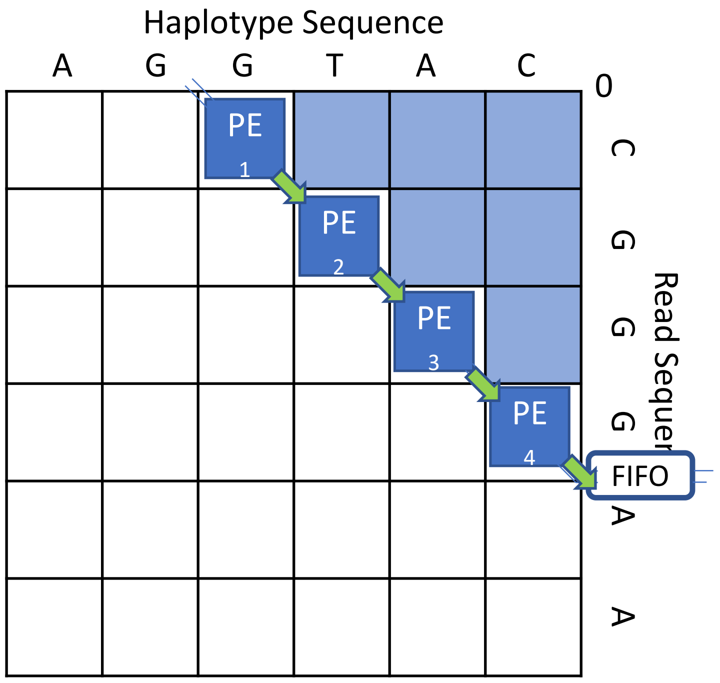
Processing Element (PE) Ring:



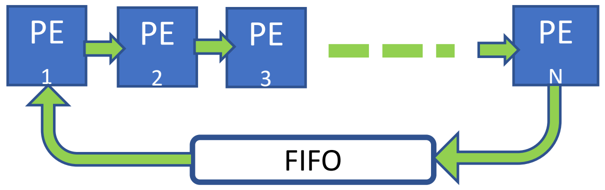
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



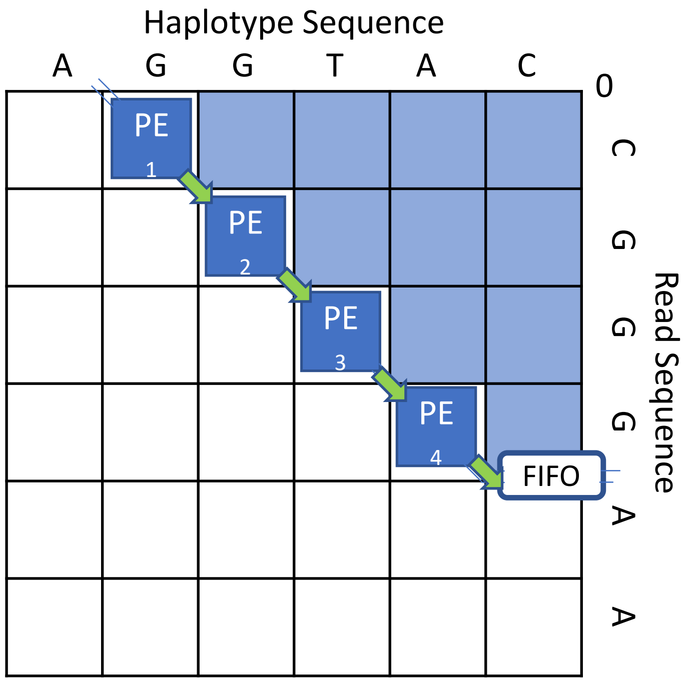
Processing Element (PE) Ring:



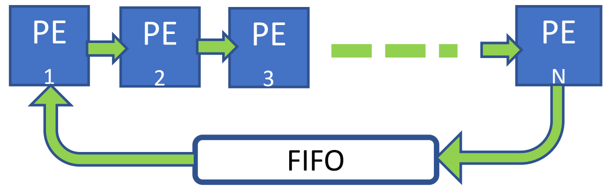
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



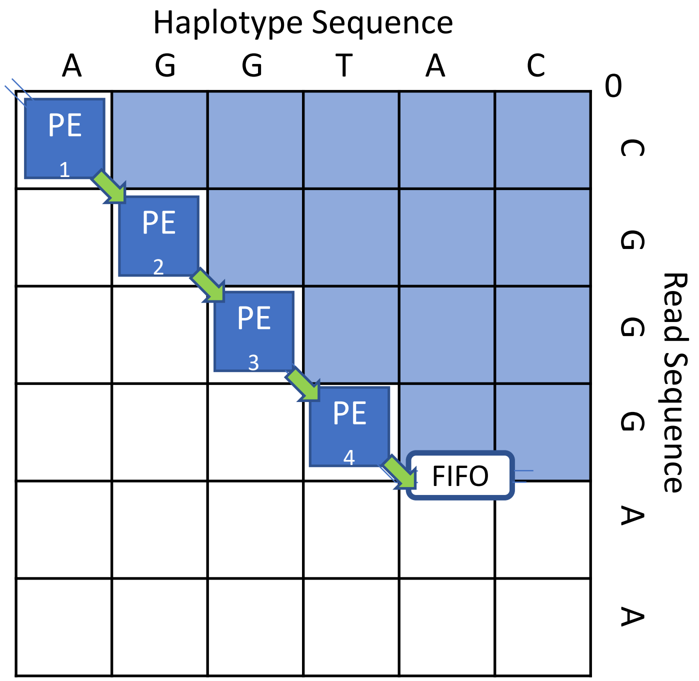
Processing Element (PE) Ring:



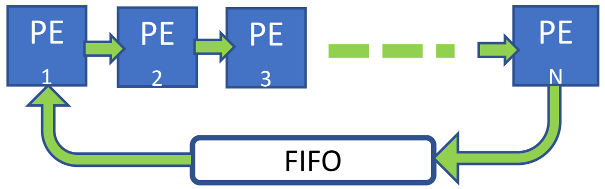
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



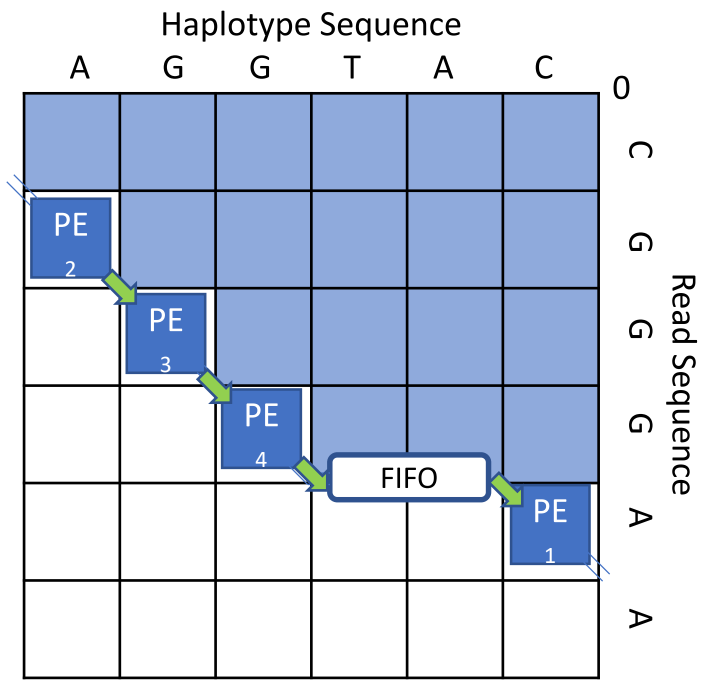
Processing Element (PE) Ring:



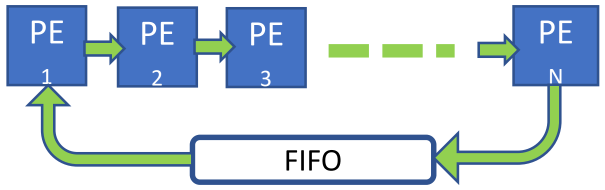
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



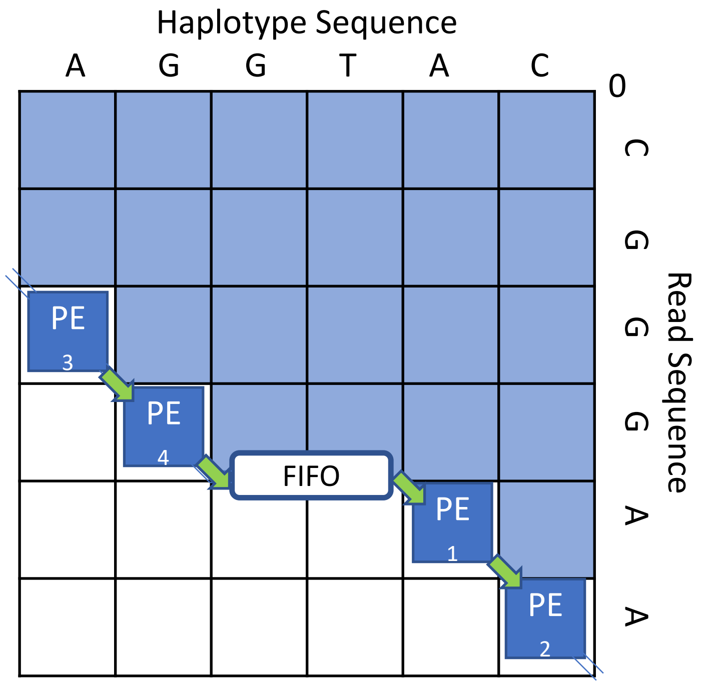
Processing Element (PE) Ring:



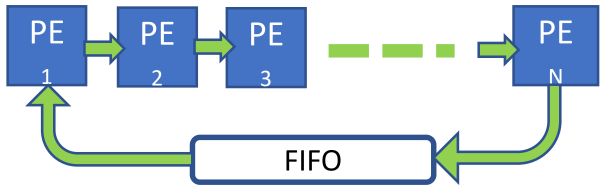
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



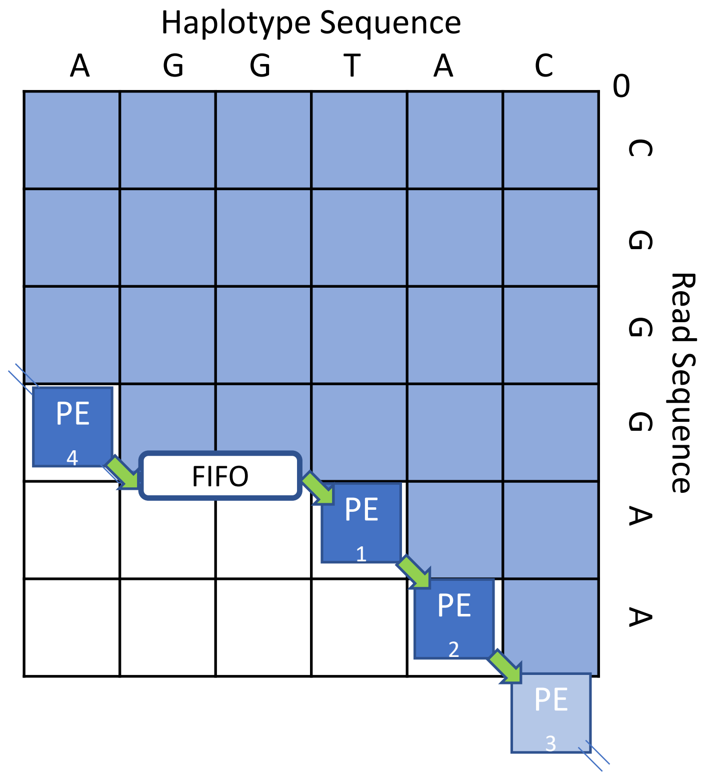
Processing Element (PE) Ring:



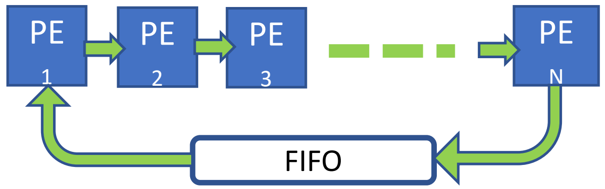
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



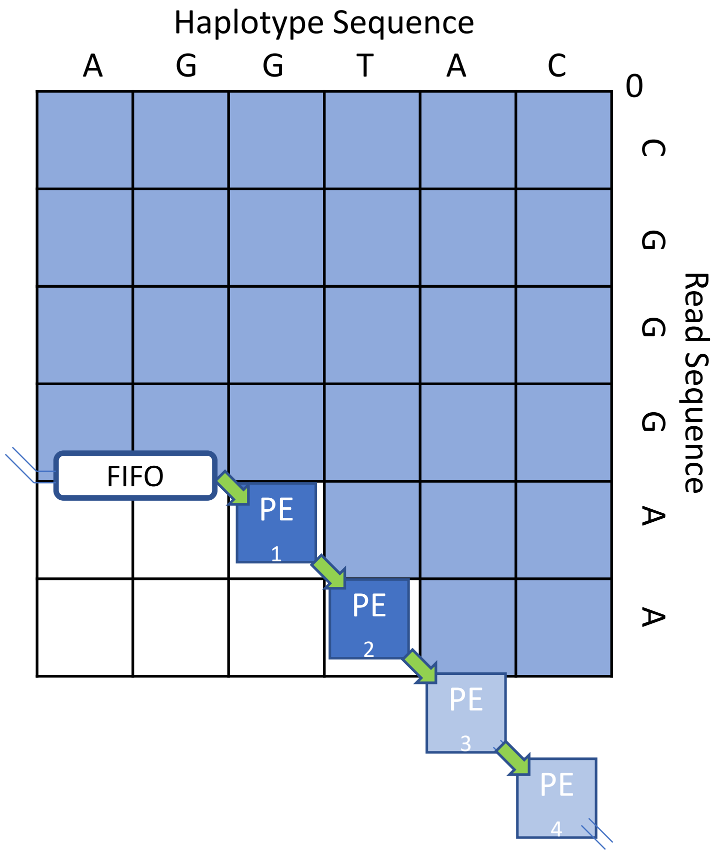
Processing Element (PE) Ring:



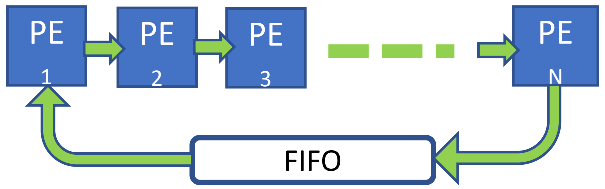
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



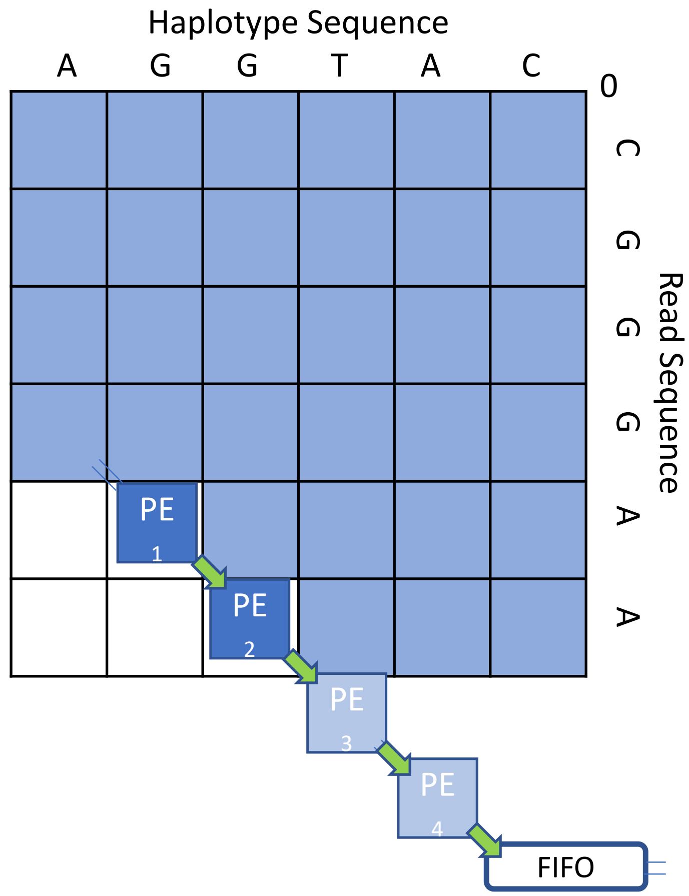
Processing Element (PE) Ring:



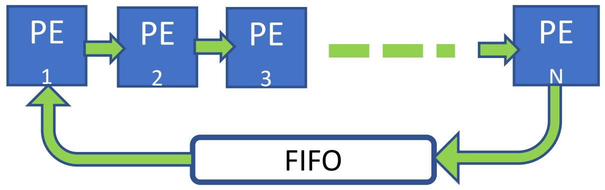
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



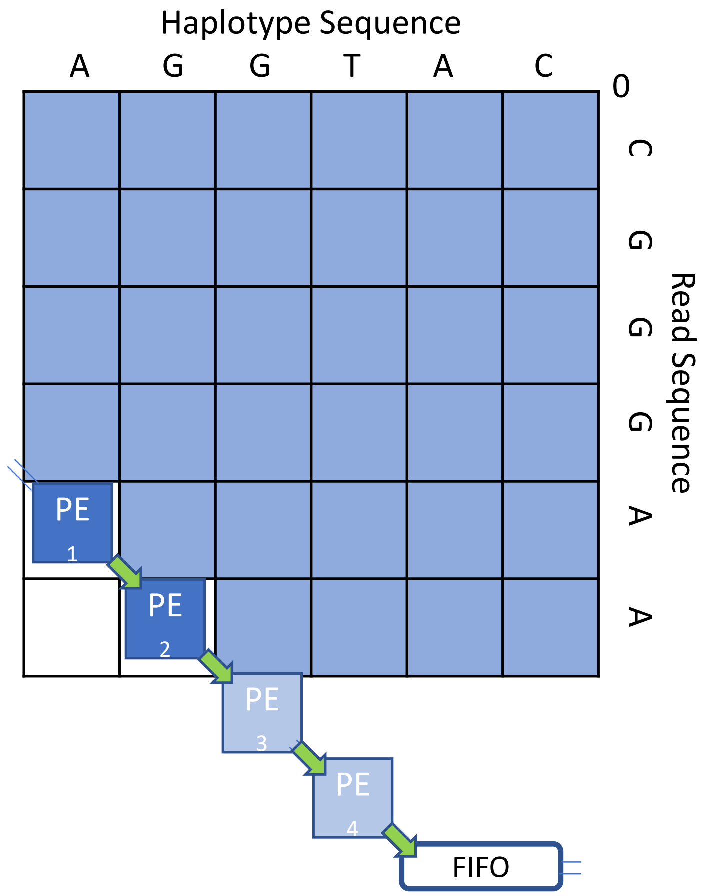
Processing Element (PE) Ring:



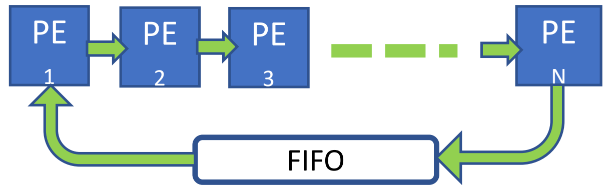
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups



Processing Element (PE) Ring:

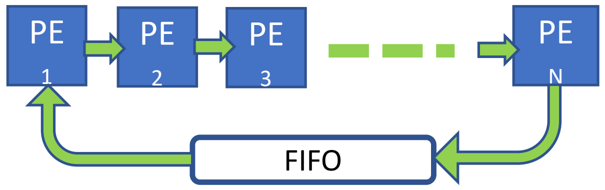


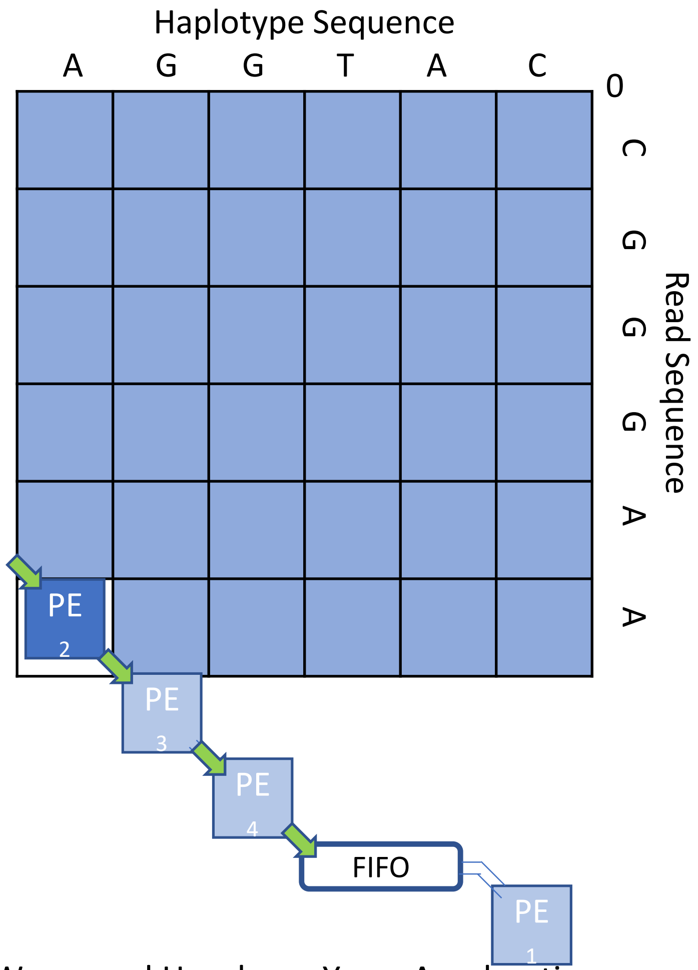
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups

Processing Element (PE) Ring:



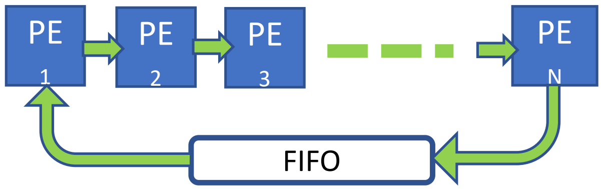


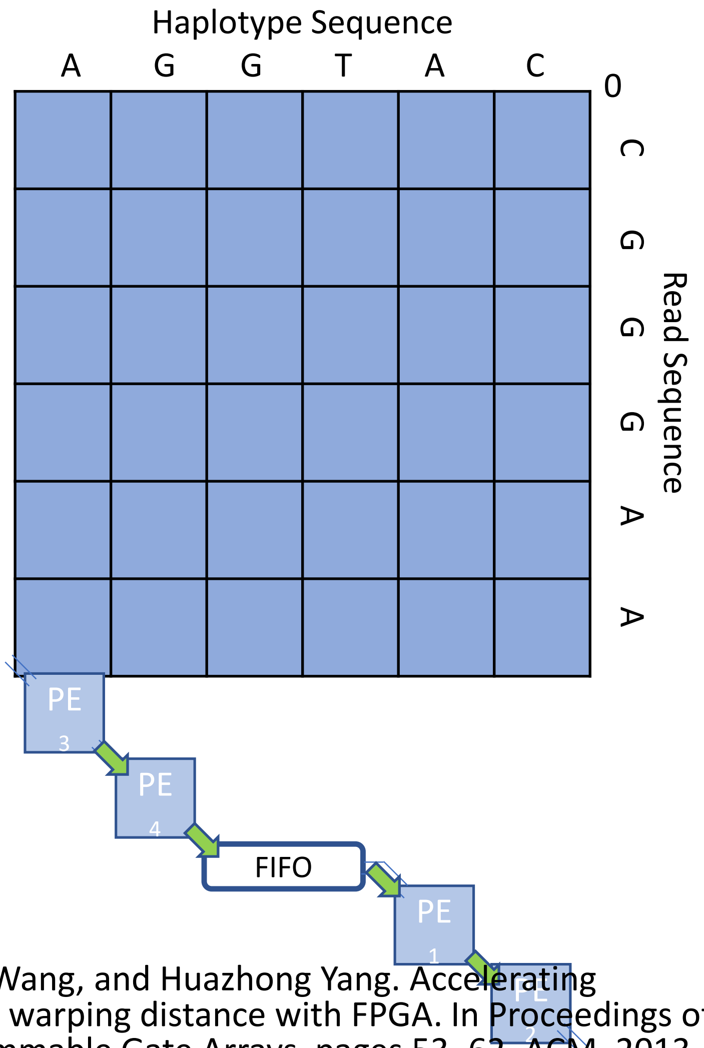
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

PE Ring*

- Connects the first PE and the last PE with FIFO
- Divide matrix rows to groups

Processing Element (PE) Ring:

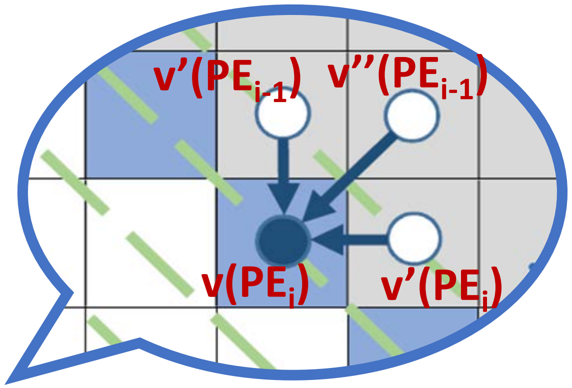




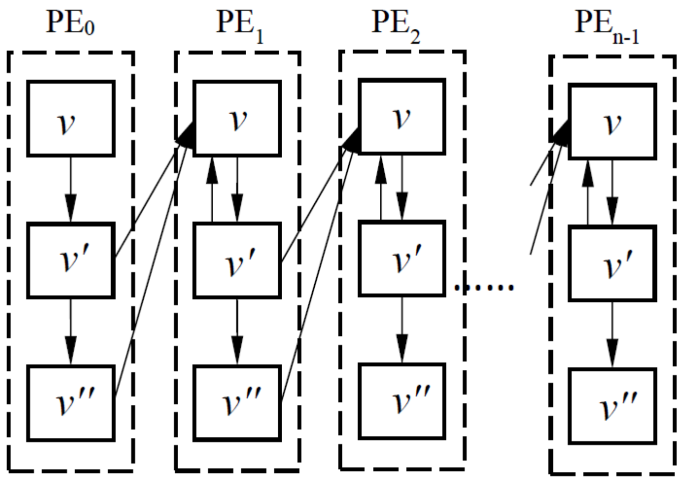
* Zilong Wang, Sitao Huang, Lanjun Wang, Hao Li, Yu Wang, and Huazhong Yang. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, pages 53–62. ACM, 2013.

Challenges in Designing PE for Pair HMM

- PE structure is designed according to the data dependencies in the algorithm
- Each PE passes its intermediate computing result to the next PE

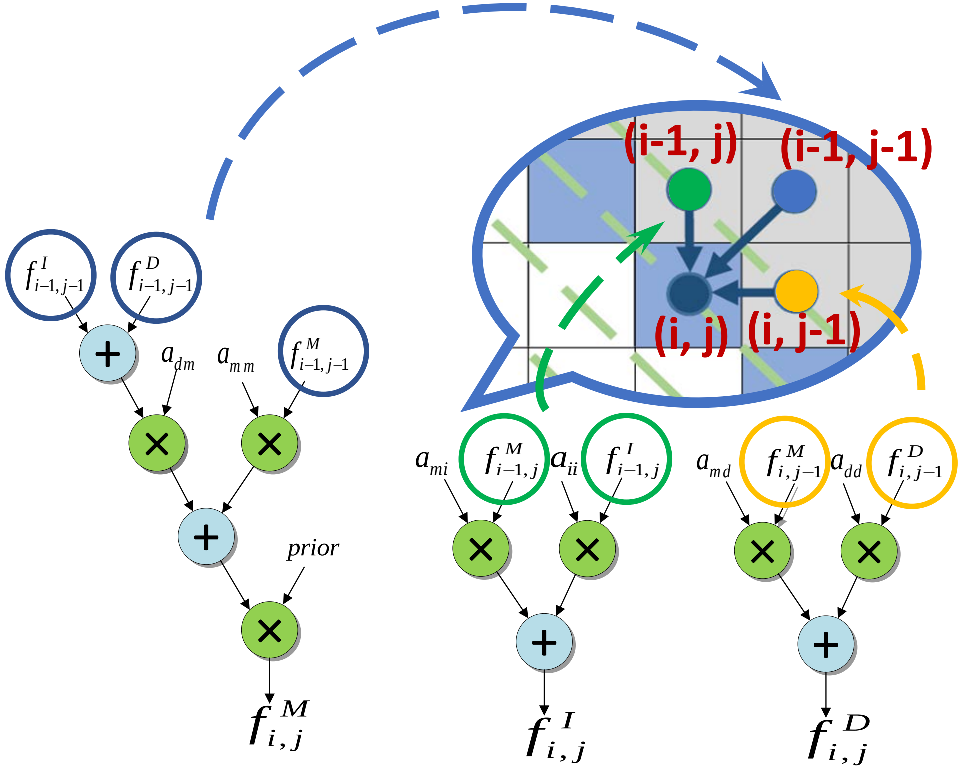


$$v(PE_i) = f(v'(PE_i), v'(PE_{i-1}), v''(PE_{i-1}))$$



Challenges in Designing PE for Pair HMM – Cont.

- Floating point operations
  - Long latency
  - Need sophisticated FSM
- Complicated arithmetic operations in DP
  - Elements in three DP matrices depend on each other



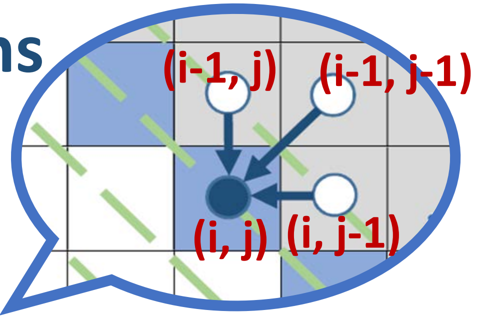
Arithmetic Operations Within a PE (Original)

$$f^D(i,j) = a_{MD} f^M(i,j-1) + a_{DD} f^D(i,j-1)$$

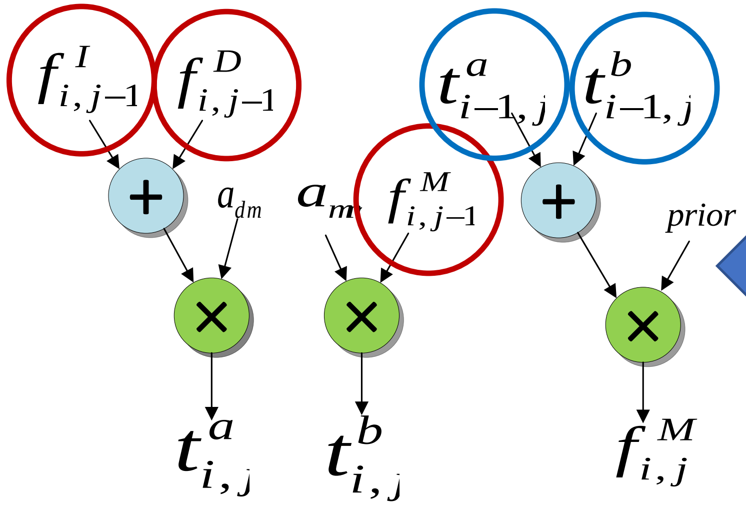
$$f^I(i,j) = a_{MI} f^M(i-1,j) + a_{II} f^I(i-1,j)$$

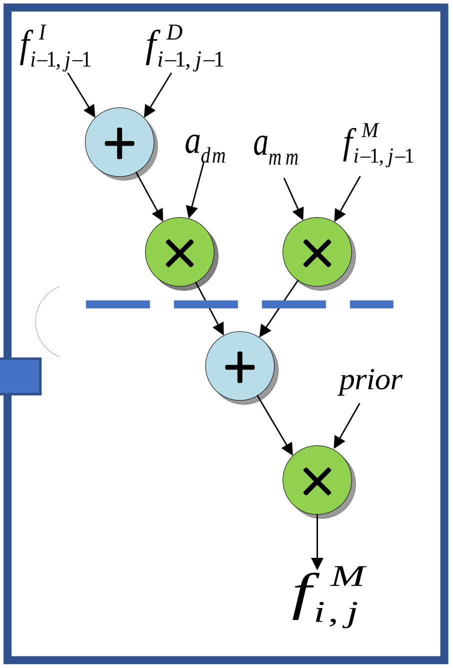
$$f^M(i,j) = prior \cdot \left(a_{MM} f^M(i-1,j-1) + a_{IM} f^I(i-1,j-1) + a_{DM} f^D(i-1,j-1)\right)$$

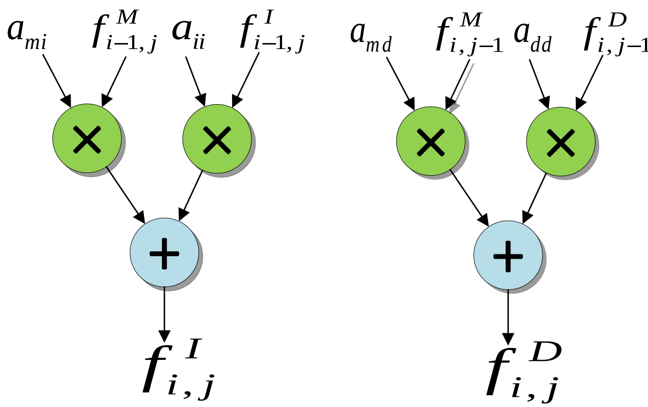
Optimization 1: Shorten critical path in arithmetic operations



Using different input operands





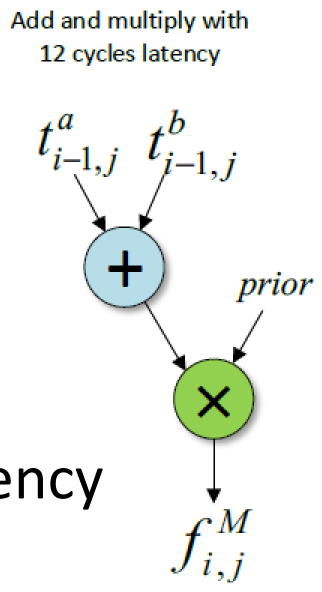


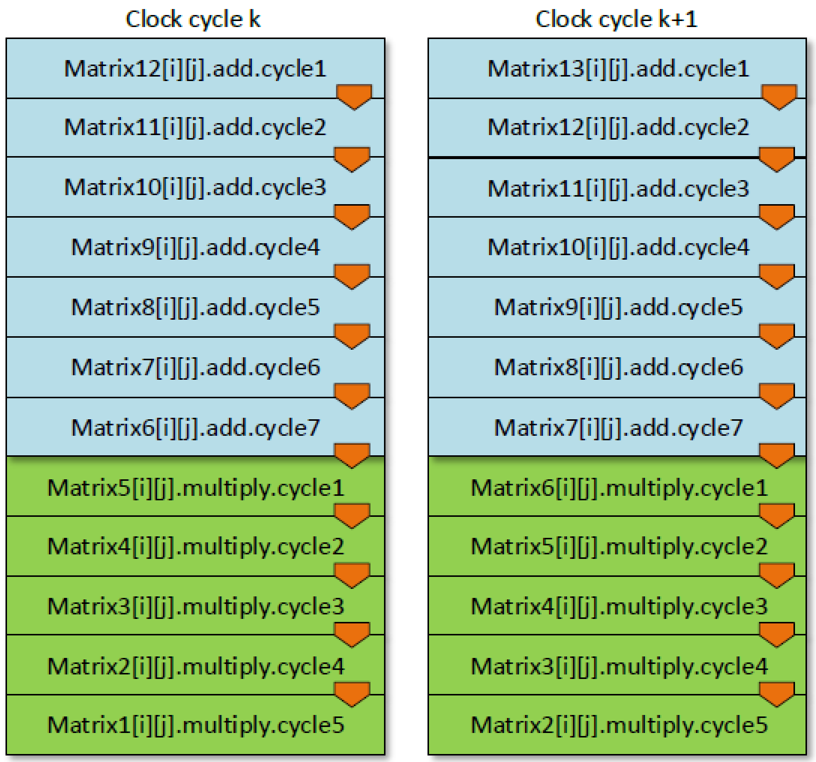
Optimized f^M Calculation

Arithmetic Operations Within a PE (Original)

Optimization 2:
Pipelining and resource sharing

- Hide the floating-point arithmetic operations' latency
- Improve throughput

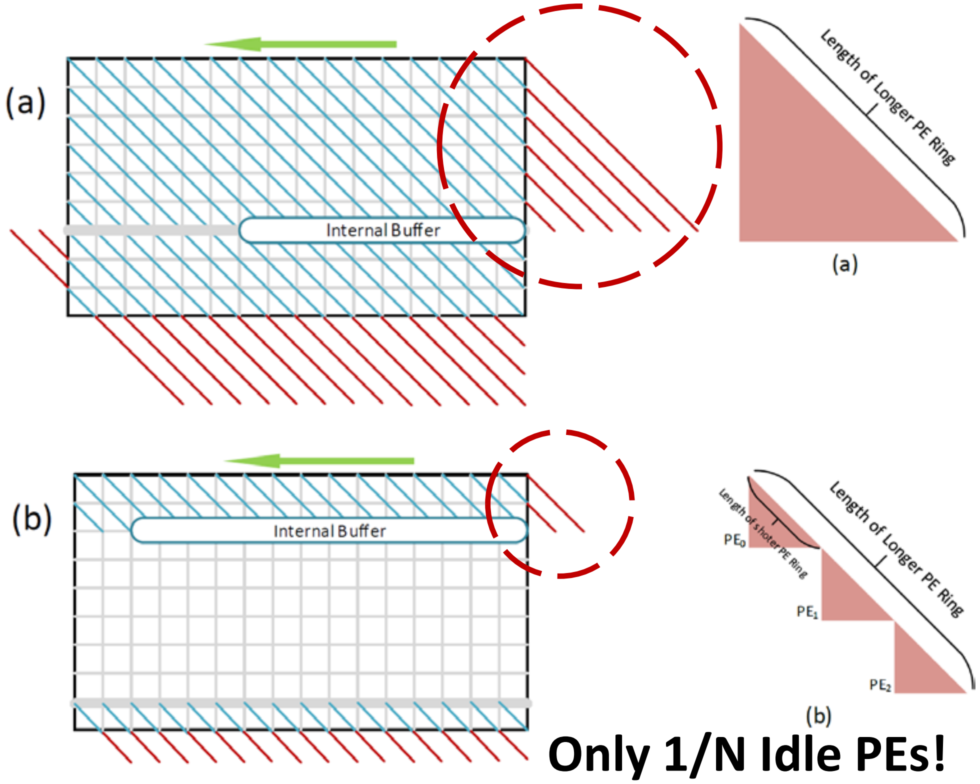




Clock cycle k	Clock cycle k+1
Matrix12[i][j].add.cycle1	Matrix13[i][j].add.cycle1
Matrix11[i][j].add.cycle2	Matrix12[i][j].add.cycle2
Matrix10[i][j].add.cycle3	Matrix11[i][j].add.cycle3
Matrix9[i][j].add.cycle4	Matrix10[i][j].add.cycle4
Matrix8[i][j].add.cycle5	Matrix9[i][j].add.cycle5
Matrix7[i][j].add.cycle6	Matrix8[i][j].add.cycle6
Matrix6[i][j].add.cycle7	Matrix7[i][j].add.cycle7
Matrix5[i][j].multiply.cycle1	Matrix6[i][j].multiply.cycle1
Matrix4[i][j].multiply.cycle2	Matrix5[i][j].multiply.cycle2
Matrix3[i][j].multiply.cycle3	Matrix4[i][j].multiply.cycle3
Matrix2[i][j].multiply.cycle4	Matrix3[i][j].multiply.cycle4
Matrix1[i][j].multiply.cycle5	Matrix2[i][j].multiply.cycle5

Optimization 3:
Tuning PE ring size and number of PE rings

- Same amount of HW resource can accommodate more shorter PE rings (calculating multiple matrices)
- Shorter PE rings have fewer idle PEs



Only 1/N Idle PEs!

Experiment Result 1: Comparison to Other Implementations

- Compared to CPU, vector processor, GPU, multi-core, previous FPGA implementations
- Using "10s" dataset
- Arria 10 has more logic and DSP resources. It also has hard floating-point DSP block

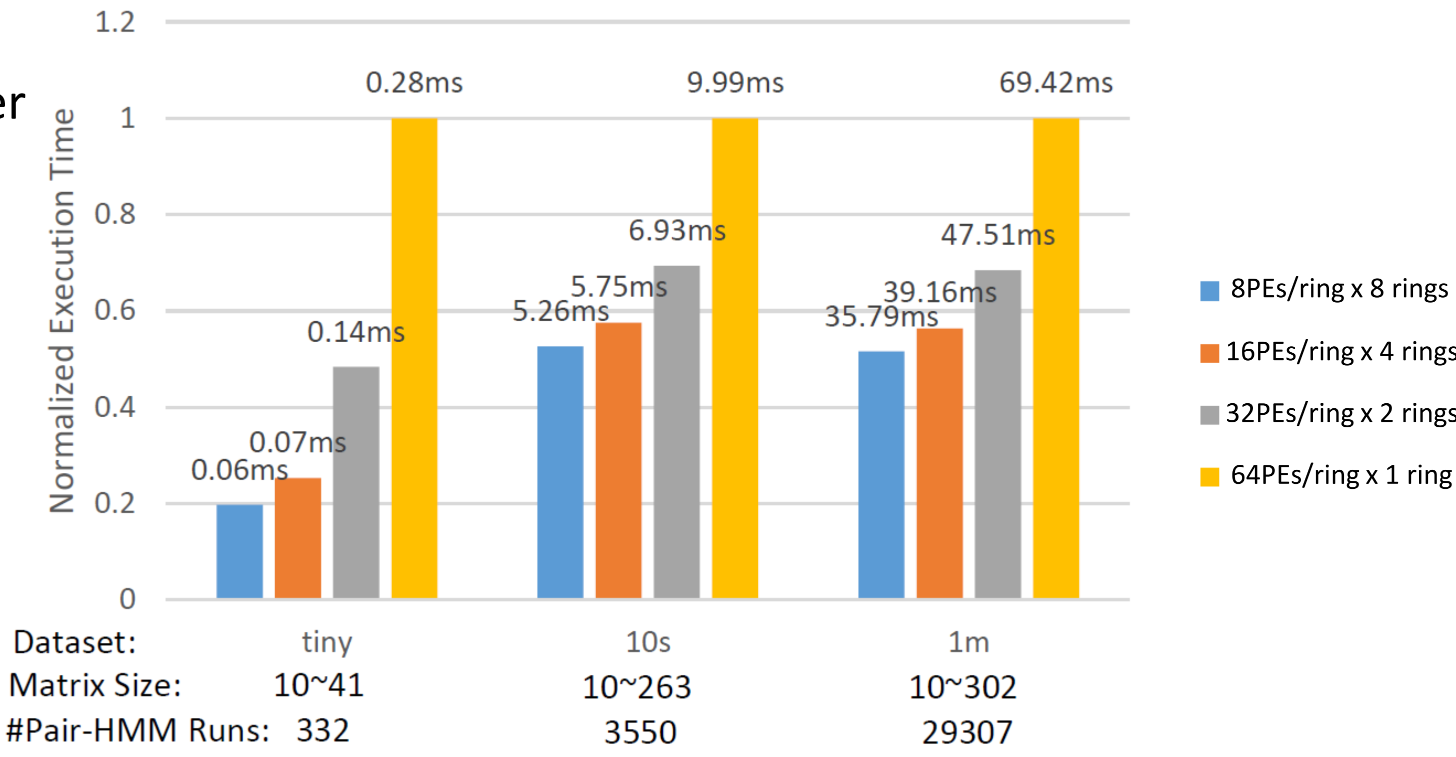
Platform	Runtime(ms)	Speedup
Java on CPU	10800	1×
C++ on CPU	1267	9×
Intel Xeon AVX Single Core	138	78×
NVidia K40 GPU	70	154×
Intel Xeon AVX 24 Cores	15	720×
Altera OpenCL (Stratix V)*	8.3	1301×
Our Design (Stratix V)	**5.3**	**2038**×
Altera OpenCL (Arria 10)*	2.8	3857×
Our Design (Arria 10)	**2.6**	**4154**×

Theoretical runtime lower bound (assuming no idle PE) for 64 PEs: **4.7ms**

* Altera. Accelerating genomics research with OpenCL and FPGAs, 2016.

Experiment Result 2: Impact of PE Ring Size

- Shorter PE rings benefit from higher PE utilization and smaller PE initialization overhead



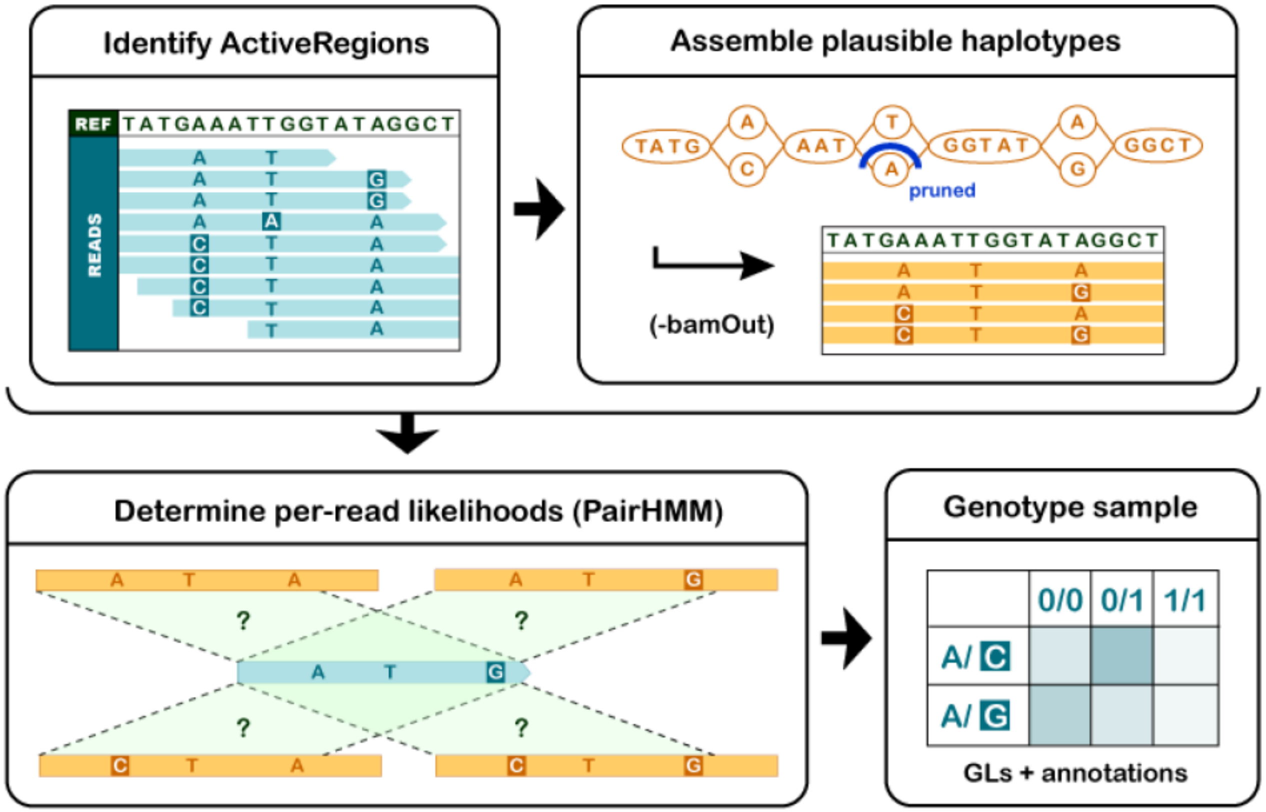
Dataset:	tiny	10s	1m
Matrix Size:	10~41	10~263	10~302
#Pair-HMM Runs:	332	3550	29307

Summary

- Pair HMM forward algorithm is computation-intensive. It is the bottleneck of HaplotypeCaller.
- Ring-based hardware structure exhibits flexibility in configuration and high data reuse.
- PE ring structure based pair HMM implementation can achieve significant speedup compared to the software implementation, and it also outperforms the published best hardware implementation.

HaplotypeCaller

BACKUP SLIDES



Emission and Transition Probabilities

BACKUP SLIDES

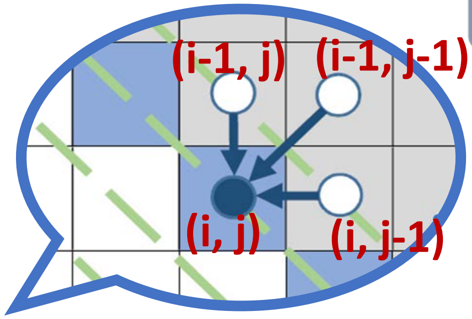
$$prior = \begin{cases} 1 - Q_{base}; & \text{if the bases match} \\ Q_{base}; & \text{if the bases don't match} \end{cases}$$

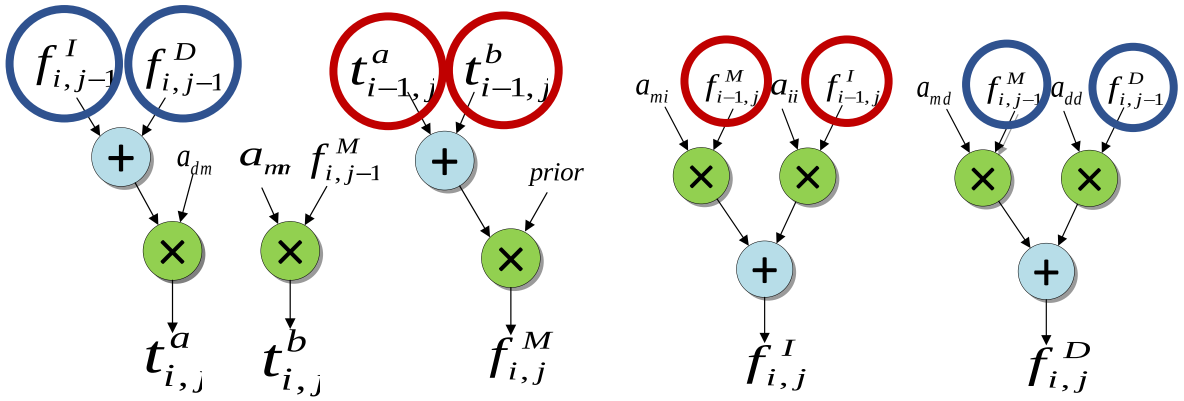
$$\begin{aligned}
a_{MM} &= 1 - (Q_i + Q_d) && \text{– match continuation} \\
a_{IM} &= 1 - Q_g && \text{– insertion to match} \\
a_{DM} &= 1 - Q_g && \text{– deletion to match} \\
a_{MI} &= Q_i && \text{– match to insertion} \\
a_{II} &= Q_g && \text{– insertion continuation} \\
a_{MD} &= Q_d && \text{– match to deletion} \\
a_{DD} &= Q_g && \text{– deletion continuation}
\end{aligned}$$

Q_{base} : Base Error Rate
Q_i : Base Insertion Probability
Q_d : Base Deletion Probability
Q_g : Gap Continuation Penalty

What's in PE?

BACKUP SLIDES





BACKUP SLIDES

Why sequence alignment?

- Comparing genes or regions from different species
  - to find important regions
  - determine function
  - uncover evolutionary forces
- Assembling fragments to sequence DNA
- Compare individuals to looking for mutations

BACKUP SLIDES

Problem Statement

(candidate to be verified) (data from sequencing machine)

- Input: two sequences S_h and S_r (S_h: haplotype S_r: read)
- Goal: find a similarity score of S_h and S_r

S_r	GTAA
S_h	AGGTC

- Similarity score is defined over a **pair Hidden Markov Model**

BACKUP SLIDES

Pair HMM – Action Sequence

- Action(**D**`elete`, **I**`nsert`, **M**`atch/Mismatch`) sequence $\{a_t\}$ s.t. $S_r \xrightarrow{\{a_t\}} S_h$

One possible alignment:

S_r	--GTAA
S_h	AGGTC-
$\{a_t\}$	IIMMMD

Another possible alignment:

S_r	-G-TAA
S_h	AGGT-C
$\{a_t\}$	IMIMDM

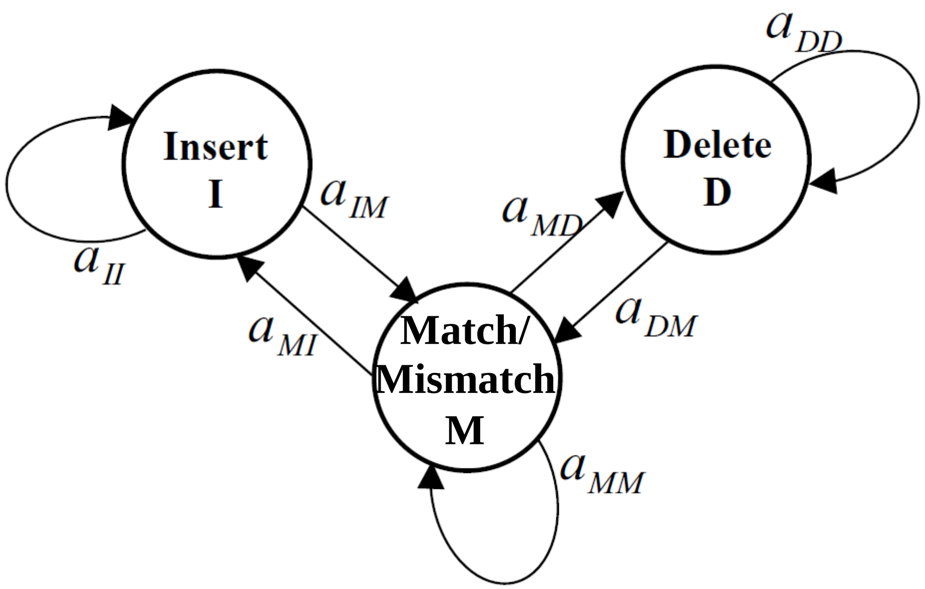
There are many action sequences mapping S_r to S_h.

BACKUP SLIDES

Pair HMM - Probability

- Each action sequence is associated with a probability:

$$P(\{a_t\}) = \prod_t p(a_t | a_{t-1})$$



Similarity score – Dynamic Programming

$$score(S_h, S_r) = \sum_{\{a_t\}:\, S_h \xrightarrow{\{a_t\}} S_r} P(\{a_t\}) = f^D(N_h, N_r) + f^M(N_h, N_r) + f^I(N_h, N_r)$$

- Last action: delete
- Last action: match / mismatch
- Last action: insert

$S_r[0{:}j-1]$	S_r	G?T?AA
$S_h[0{:}i]$	S_h	AGGTA-
$\{a_t\}$	$\{a_t\}$	?????D

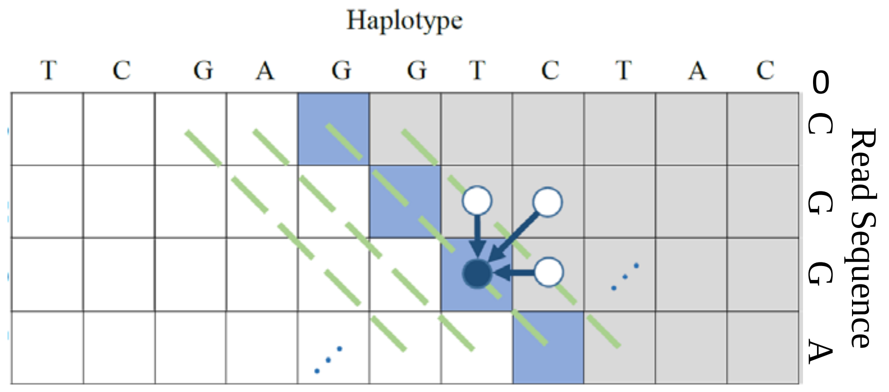
probability dependency (Markov)

$$f^D(i,j) = a_{MD} f^M(i,j-1) + a_{DD} f^D(i,j-1) \cancel{+ a_{ID} f^I(i,j-1)}$$

BACKUP SLIDES

Recursion

- Similarly:



$$f^D(i,j) = a_{MD} f^M(i,j-1) + a_{DD} f^D(i,j-1)$$
$$f^I(i,j) = a_{MI} f^M(i-1,j) + a_{II} f^I(i-1,j)$$
$$f^M(i,j) = prior \cdot \left(a_{MM} f^M(i-1,j-1) + a_{IM} f^I(i-1,j-1) + a_{DM} f^D(i-1,j-1)\right)$$

- Output: $score(S_h, S_r) = f^D(N_h, N_r) + f^M(N_h, N_r) + f^I(N_h, N_r)$
- Complexity: $O(\underbrace{M_h}_{\text{\# haplotype sequences}} \times \underbrace{M_r}_{\text{\# read sequences}} \times N_h \times N_r)$