Goal: Minimizing Alignment Time

Sequence Alignment is expensive

Our goal is to accelerate read mapping by reducing the need for dynamic programming algorithms
Key Idea

Genomic Strings

Dissimilar Strings

Ignore them if the number of differences exceeds a threshold.

Similar Strings

Find number, location, and type of differences?

EXPENSIVE!
Ideal Filtering Algorithm

1. Filter out most of incorrect mappings.
2. Preserve all correct mappings.
3. Do it quickly.
SneakySnake

- **Key observation:**
  - Correct alignment is a sequence of non-overlapping long matches.

Dot plot, dot matrix (Lipman and Pearson, 1985)
**SneakySnake**

- **Key observation:**
  - Correct alignment is a sequence of non-overlapping long matches

- **Key idea:**
  - Approximate edit distance calculation is similar to *Single Net Routing problem* in VLSI chip
Given two genomic sequences, a reference sequence $R[1 \ldots m]$ and a query sequence $Q[1 \ldots m]$, and an edit distance threshold $E$, we calculate the entry $Z[i, j]$ of the chip maze, where $1 \leq i \leq (2E + 1)$ and $1 \leq j \leq m$, as follows:

$$Z[i, j] = \begin{cases} 
0, & \text{if } i = E + 1, \ Q[j] = R[j], \\
0, & \text{if } 1 \leq i \leq E, \ Q[j-i] = R[j], \\
0, & \text{if } i > E + 1, \ Q[j+i-E-1] = R[j], \\
1, & \text{otherwise}
\end{cases} \quad (1)$$

<table>
<thead>
<tr>
<th>column</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>3rd Upper Diagonal</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2nd Upper Diagonal</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1st Upper Diagonal</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Main Diagonal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1st Lower Diagonal</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2nd Lower Diagonal</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3rd Lower Diagonal</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SneakySnake Walkthrough

Building Neighborhood Map  Finding the Optimal Routing Path  Examining the Snake Survival

\[ E = 3 \]
SneakySnake Walkthrough

Building Neighborhood Map
Finding the Optimal Routing Path
Examining the Snake Survival

SAFARI
This is what you actually need to **build** and it can be done **on-the-fly!**
FPGA Resource Analysis

- FPGA resource usage for a single filtering unit of GateKeeper, Shouji, and Snake-on-Chip for a sequence length of 100 and under different edit distance thresholds (E).

<table>
<thead>
<tr>
<th></th>
<th>$E$ (bp)</th>
<th>Slice LUT</th>
<th>Slice Register</th>
<th>No. of Filtering Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>GateKeeper</td>
<td>2</td>
<td>0.39%</td>
<td>0.01%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.71%</td>
<td>0.01%</td>
<td>16</td>
</tr>
<tr>
<td>Shouji</td>
<td>2</td>
<td>0.69%</td>
<td>0.08%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.72%</td>
<td>0.16%</td>
<td>16</td>
</tr>
<tr>
<td>Snake-on-Chip</td>
<td>2</td>
<td>0.68%</td>
<td>0.16%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.42%</td>
<td>0.34%</td>
<td>16</td>
</tr>
</tbody>
</table>
The Effect of Pre-Alignment (Theoretically)

Filter + Alignment

Pre-alignment saves more than
40% to 80% of the total processing time

Pre-alignment rejected mapping percentage and speed compared to alignment step

SAFARI
Filtering Accuracy

Fig. 10: The execution time of SneakySnake, Parasail, and SneakySnake integrated with Parasail using long sequences, (a) 10Kbp and (b) 100Kbp, and 40 CPU threads. The left y-axes of (a) and (b) are on a logarithmic scale. For each edit distance threshold value, we provide in the right y-axes of (a) and (b) the rate of accepted pairs (out of 100,000 pairs for 10Kbp and out of 74,687 pairs for 100Kbp) by SneakySnake that are passed to Parasail. We present the end-to-end speedup values obtained by integrating SneakySnake with Parasail.
Long Read Mapping (SneakySnake vs KSW2)

10K bp reads

![Graph showing execution time vs edit distance threshold for 10K bp reads.]

100K bp reads

![Graph showing execution time vs edit distance threshold for 100K bp reads.]

Fig. 11: The execution time of SneakySnake, KSW2, and SneakySnake integrated with KSW2 using long sequences, (a) 10Kbp and (b) 100Kbp, and a single CPU thread. The left y-axes of (a) and (b) are on a logarithmic scale. For each edit distance threshold value, we provide in the right y-axes of (a) and (b) the rate of accepted pairs (out of 100,000 pairs for 10Kbp and out of 74,687 pairs for 100Kbp) by SneakySnake that are passed to KSW2. We present the end-to-end speedup values obtained by integrating SneakySnake with KSW2.
Key Results of SneakySnake

- SneakySnake is up to **four orders of magnitude more accurate** than *Shouji* (Bioinformatics’19) and *GateKeeper* (Bioinformatics’17)

- Using short reads, SneakySnake **accelerates Edlib** (Bioinformatics’17) and *Parasail* (BMC Bioinformatics’16) by
  - up to **37.7×** and **43.9×** (>
    **12×** on average), on CPUs
  - up to **413×** and **689×** (>400× on average) with **FPGA/GPU acceleration**

- Using long reads, SneakySnake **accelerates Parasail** and *KSW2* by
  - **140.1×** and **17.1×** on average, respectively, on CPUs
Can We Do Better?

Alleviating Data Movement Bottlenecks
We need to design mapping & filtering algorithms that fit processing-in-memory
Near-memory Pre-alignment Filtering

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gomez-Luna, Henk Corporaal, Onur Mutlu,

"FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications"

[Source Code]
Near-memory SneakySnake

- Problem: Read Mapping is heavily bottlenecked by data movement from main memory

- Solution: Perform read mapping near where data resides (i.e., near-memory)

- We carefully *redesigned the accelerator logic* of SneakySnake to exploit *near-memory computation* capability on modern FPGA boards with high-bandwidth memory
Heterogeneous System: CPU+FPGA

We evaluate two POWER9+FPGA systems:

1. **HBM-based AD9H7 board**: Xilinx Virtex Ultrascale+™ XCVU37P-2
2. **DDR4-based AD9V3 board**: Xilinx Virtex Ultrascale+™ XCVU3P-2

FPGA + HBM on the same package substrate
Key Results of Near-memory SneakySnake

Near-memory pre-alignment filtering improves performance and energy efficiency by $27.4\times$ and $133\times$, respectively, over a 16-core (64 hardware threads) IBM POWER9 CPU.
Near-memory Pre-alignment Filtering

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gomez-Luna, Henk Corporaal, Onur Mutlu,

“FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications”

[Source Code]
Mohammed Alser, Taha Shahroodi, Juan-Gomez Luna, Can Alkan, and Onur Mutlu, "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs" Bioinformatics, 2020.
[Source Code]
[Online link at Bioinformatics Journal]
NVIDIA is claiming a 7x improvement in dynamic programming algorithm (DPX instructions) performance on a single H100 versus naïve execution on an A100.

UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
  - Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth


Onur Mutlu, *Computer Architecture Lecture 2b*, Fall 2019, ETH Zurich
Most speedup comes from parallelism enabled by novel architectures and algorithms.
Near-memory Pre-alignment Filtering

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu

“Accelerating Genome Analysis: A Primer on an Ongoing Journey”

Mohammed Alser, Jeremy Rotman, Dhrithi Deshpande, Kodi Taraszka, Huwenbo Shi, Pelin Icer Baykal, Harry Taegyun Yang, Victor Xue, Sergey Knyazev, Benjamin D. Singer, Brunilda Balliu, David Koslicki, Pavel Skums, Alex Zelikovsky, Can Alkan, Onur Mutlu, Serghei Mangul

"Technology dictates algorithms: Recent developments in read alignment"

Genome Biology, 2021

[Source code]
Feedback From Our Community!

James Ferguson
@Psy_Fer_

This is awesome! I've got my evening reading sorted.

Stéphane Le Crom
@slecrom

Very complete article on the evolution of read alignment algorithms. #NGS #genomics

Svetlana Gorokhova
@SGorokhova

An impressive overview of read alignment methods over the last three decades

BContrerasMoreira @BrunoContrerasM · Sep 10
Replying to @mealser @GenomeBiology and 3 others
Buen hilo de repaso sobre la evolución de los algoritmos de alineamiento de secuencias a medida que ha mejorado la tecnología de secuenciación

https://twitter.com/mealser/status/1435223377644503040
More on Accelerating Genome Analysis ...

- Mohammed Alser,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Talk at RECOMB 2021, Virtual, August 30, 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (27 minutes)]
  [Related Invited Paper (at IEEE Micro, 2020)]
More on Intelligent Genome Analysis …

- Mohammed Alser,
  "Computer Architecture - Lecture 10: Intelligent Genome Analysis"
  ETH Zurich, Computer Architecture Course, Fall 2021, Lecture 10, Virtual, 29 October 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (3 hour 2 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
More on Intelligent Genome Analysis …

- Mohammed Alser,
  "Computer Architecture - Lecture 8: Intelligent Genome Analysis"
  ETH Zurich, Computer Architecture Course, Lecture 8, Virtual, 15 October 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (2 hour 54 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
More on Fast Genome Analysis …

  - Slides (pptx) (pdf)
  - Talk Video (1 hour 37 minutes, including Q&A)
  - Related Invited Paper (at IEEE Micro, 2020)
Two P&S Genomics Courses

Livestream - P&S Genome Sequencing on Mobile Devices (Fall 2021)

9 videos • 75 views • Updated 5 days ago

https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_U2F8yrrNPD9CjcM6CFQXv
## 2021 Meetings/Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Livestream</th>
<th>Meeting</th>
<th>Learning Materials</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>5.10 Tue.</td>
<td>Live</td>
<td>M1: P&amp;S Accelerating Genomics</td>
<td>Required Materials</td>
<td>Recommended Materials</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Course Introduction &amp; Project Proposals</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[YouTube] Video</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td>20.10 Wed.</td>
<td>Live</td>
<td>M2: Introduction to Sequencing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td>27.10 Wed.</td>
<td>Live</td>
<td>M3: Read Mapping</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td>3.11 Wed.</td>
<td>Live</td>
<td>M4: GateKeeper</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W5</td>
<td>10.11 Wed.</td>
<td>Live</td>
<td>M5: MAGNET &amp; Shouji</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W6</td>
<td>17.11 Wed.</td>
<td></td>
<td>M6.1: SneakySnake</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[Video]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M6.2: GRIM-Filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[YouTube] Video</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W7</td>
<td>24.11 Wed.</td>
<td></td>
<td>M7: GenASM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[PDF] [PPT]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[YouTube] Video</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Detailed Lectures on Genome Analysis

- Computer Architecture, Fall 2020, Lecture 3a
  - Introduction to Genome Sequence Analysis (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5

- Computer Architecture, Fall 2020, Lecture 8
  - Intelligent Genome Analysis (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14

- Computer Architecture, Fall 2020, Lecture 9a
  - GenASM: Approx. String Matching Accelerator (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=XoLpzmn-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15

- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - Accelerating Genomics (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCgLwgiDRQDTyId

https://www.youtube.com/onurmutlulectures
Prior Research on Genome Analysis (1/2)


Prior Research on Genome Analysis (2/2)


