

# GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping

Mohammed Alser <sup>§</sup>, Hasan Hassan <sup>†</sup>, Hongyi Xin <sup>‡</sup>,  
Oğuz Ergin <sup>†</sup>, Onur Mutlu <sup>◆</sup>,  
and Can Alkan <sup>§</sup>

<sup>§</sup> **Bilkent University**

<sup>†</sup> **TOBB University of Economics & Technology**

<sup>‡</sup> **Carnegie Mellon University**

<sup>◆</sup> **ETH Zurich**

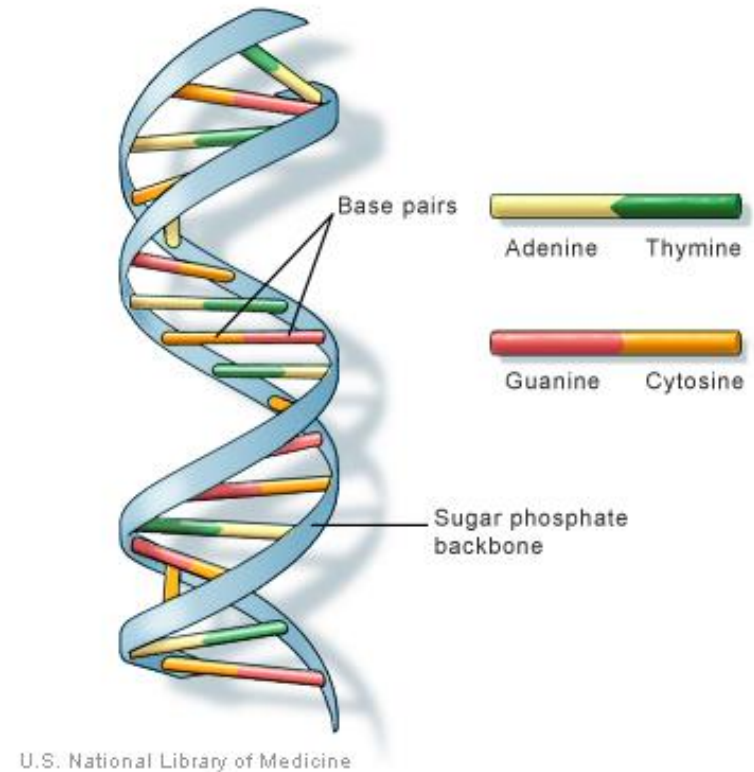
Published in Bioinformatics, 2017.

# Background, Problem & Goal

# Background (1/2)

---

- DNA is the “**source code**” of organisms
- Composed of nucleotides:
  - Adenine (**A**) – Thymine (**T**)
  - Guanine (**G**) – Cytosine (**C**)
- DNA is a **molecule**, but for this problem we abstract it to fit a computer science problem



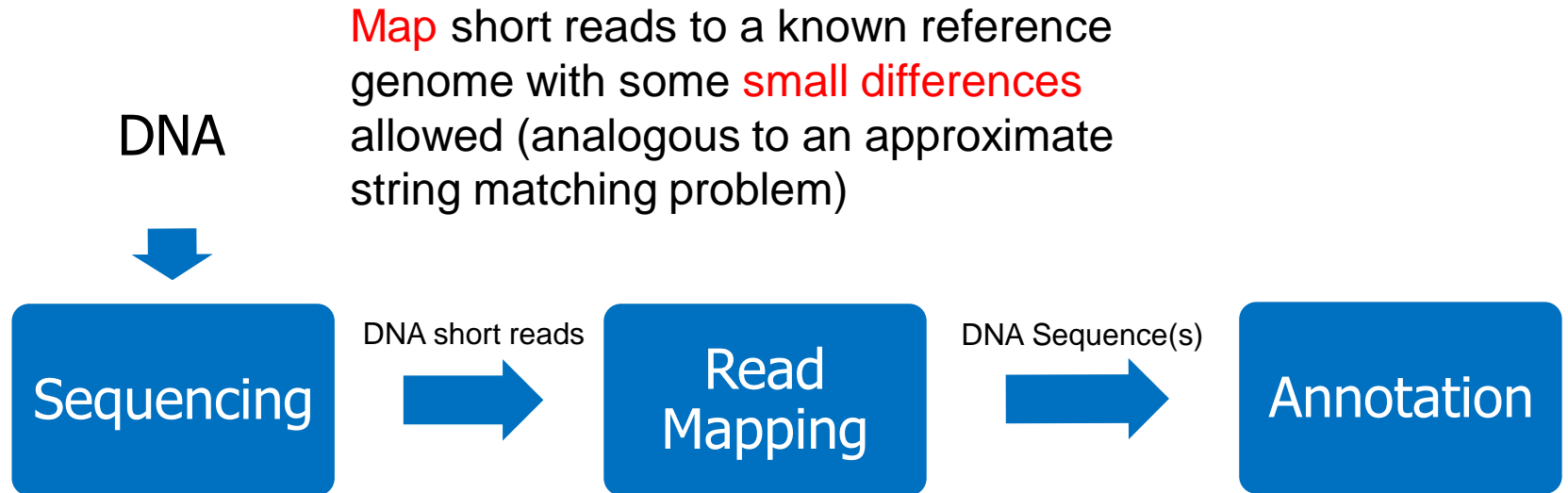
# Background (2/2)

---

- The human species' DNA is organized in **chromosomes**
- Human chromosomes range in size from about 50 million to **300 million base pairs**
- We usually have **23** pairs of chromosomes
- In total, approximately 3.3 billion base pairs for the human genome  $\approx$  **750 megabytes**
- Humans share a DNA similarity of **99.9%**

# Genome Analysis Overview

---



- High throughput DNA sequencing (**HTS**) technologies
- Generate a huge amount of small DNA segments, the “**short reads**”, which are sampled at **random locations**

**Mark** the genes and other biological features in a DNA sequence

# Hash based Read Mapping

---

- Read mapping is constituted of the following steps:
  - Hash based indexing
    - Initialization : Index the reference genome into a hash table
      - For each sequence of DNA (with a set length, e.g. 5) index every location on the genome where this pattern happens
    - When looking up a read, simply find the **candidate locations** of that subsequence (allowing some differences)
  - Alignment Verification
    - Usually a **slow** dynamic programming algorithm that checks with 100% accuracy if the read aligns with the reference

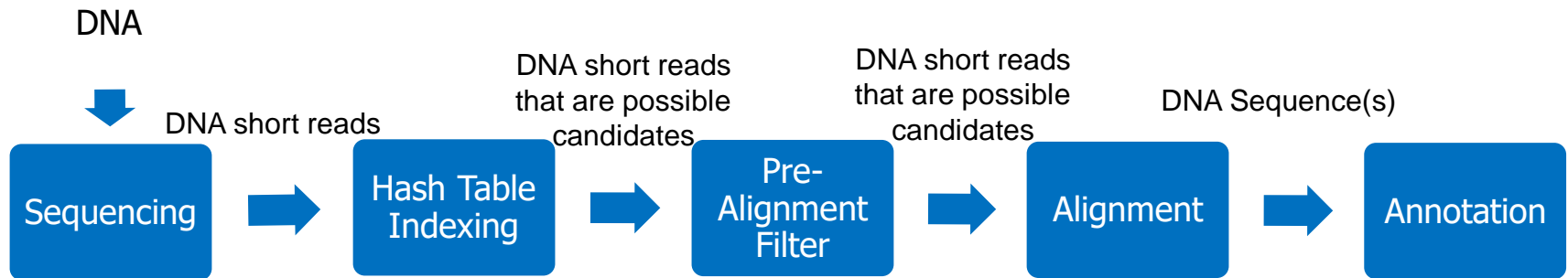
# Problem and Goal

---

- Considering the size of DNA and the **excessive amounts of reads** (up to 400 Gb/day for Illumina HiSeq 4000), the alignment operation of DNA is very computationally intensive.
- Furthermore, state of the art alignment algorithms spend **90%** of their resources checking bad alignments.
- Goal: Provide a **pre-alignment** filter to reduce the number of candidates for the alignment algorithm, therefore avoiding unnecessary computations.

# Genome Analysis with GateKeeper

---



Properties that a pre-alignment filter needs to have:

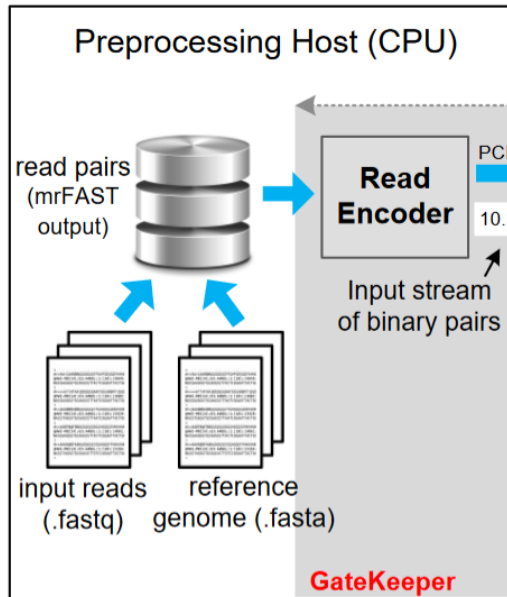
- **Accuracy**, no false negatives
- **Faster than Alignment algorithm**



# Key Approach and Ideas

# Ideas

- Enhance existing state of the art software filter (Shifted Hamming Distance) in hardware by exploiting FPGA's **parallelism** and **fast bit operations**
- FPGA pre-alignment filter



# Novelty

First hardware accelerated system for alignment filtering  
using FPGAs (Field-Programmable Gate Array)

# Mechanisms

# Implementation

---

- We want to compare the reference sequence with the short read one
  - Only accept short reads if the edit distance between the reference sequence and the read sequence are below a user-defined threshold **E**
  - **Hamming Distance** is the minimum number of substitutions required to go from one sequence to the other.  
**Edit Distance** also takes into account insertions and deletions
  - Two cases need to be handled:
    - Substitution/Exact Matching
    - Insertion/Deletion
-

# Substitution/Exact Matching

---

- Fast Approximate String Matching:
  - We simply compute the Hamming mask, which is the **XOR** operation between both sequences and count the number of ones.

M	O	V	E
M	O	R	E
<hr/>			
0	0	1	0

Here we have a Hamming distance of 1, and since the strings have the same length, the edit distance is also 1.

# Insertion & Deletion (1/2)

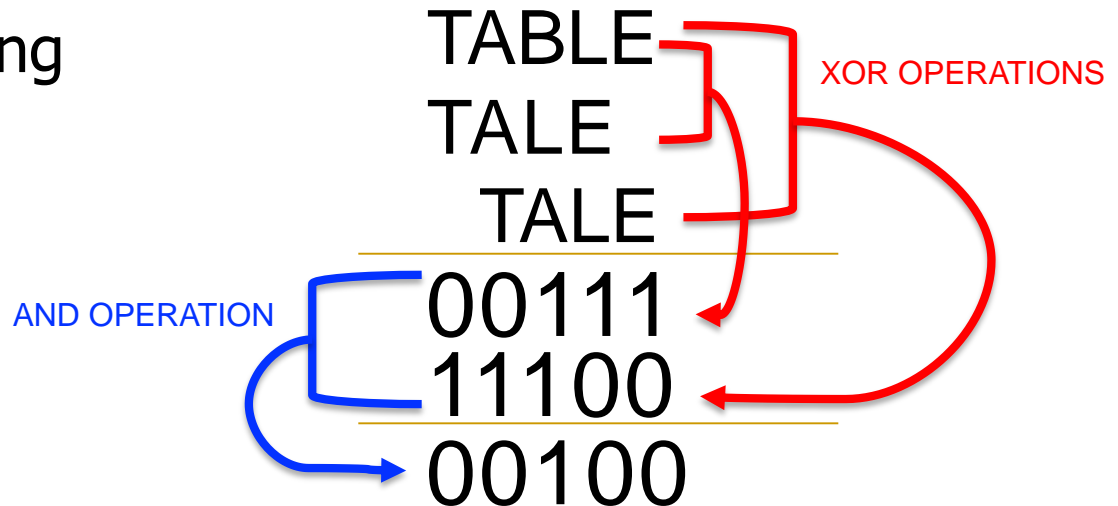
- Let's say we want to compare: TABLE and TALE

TABLE
TALE
00111

Here we have a Hamming distance of 3 but we can clearly see that it has an edit distance of 1 (one deletion)

- Any idea how we could fix this?
- Solution: **Shift** the string

By **AND**ing the result of both **XOR**s, we get the right edit distance



## Insertion & Deletion (2/2)

- Analogously, to work with insertions, we simply shift the sequence to the left.
- Now let's take a look at a real example (here our Edit Threshold  $E = 3$ ):

Query : AAAAAAAAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTCAGCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGGGAAATTGTTGGGCCGGA  
Reference : AAAAAAAAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG

**Hamming Mask :**

[illegible]

2-Deletion Mask : 00000000000000010000000001011011100111111111111011110001110110101101111111110001001010111101101001010

3-Deletion Mask : 00000000000000010111111111101110110011011101101100010010011111111111100101100110101101101101101111

```
1-Insertion Mask : 0000000000000011111111111111101111101111111011101100010010011111111111110010110011000101011101110111110
```

2-Insertion Mask : 0000000000001000000000100111110011111111100100011010101001101011111111111110111001111111000111101100

```
3-Insertion Mask : 0000000000001011111111111011101100110001111111110101101111110011001011101111111101101111010111001000
```

After ANDing, these columns will be “matches”

[illegible]

Needleman-Wunsch  
Alignment:

```

AAAAAAAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTGCAG-CACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGGAACATTGTTGGGCCGG
|||||
AAAAAAAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG

```

Do you see anything wrong? Reminder : We AND all the masks to get the result...



# Amendment

- To fix this, we **flip** these short streaks of zeros:
  - The pattern "101" becomes "111"
  - The pattern "1001" becomes "1111"
- Result:

Query : AAAAAAAAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTCAGCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGGGAAATTGTTGGGCCGGA  
Reference : AAAAAAAAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG

--- Masks after amendment ---

[illegible]

Needleman-Wunsch  
Alignment:

```

AAAAAAAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTCAG-CACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCCGG
|||||
AAAAAAAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCCGG

```

Note: only one of the columns was fixed, GateKeeper **still accepts** it (since  $E = 3$ ) but the Alignment Algorithm **will reject** it (edit distance is 4).

# Minimizing Hardware Resource Overheads

---

DNA base pairs have only **4 possible values** A, T, G or C.  
This requires **2 bits** to encode and also doubles the size  
of the masks and the other components

A → 00  
C → 01  
G → 10  
T → 11

Representation:      Nucleotide

Read:            TCCAT1


Reference:       TCCAG1

Hamming mask:   00001

0: Match    1: Mismatch

# The GateKeeper Algorithm

---

- Step 1 : Check for **perfect matching** and simple **substitutions**, return if it matches
- Step 2 : Detect **Insertions/Deletions** with the shifted Hamming masks 
- Step 3 : Reduce number of bits by half (due to encoding trick)
- Step 4 : **Amend** bits, **AND** the masks
- Step 5 : **Count** number of mismatches, return result
  - Algorithm is very easy to compute in **parallel**; Every short read is independent from the others.
  - Furthermore, the amending of individual bits in each mask can occur at the same time as they are **independent**.

# Why use an FPGA?

---

- The GateKeeper algorithm uses features that are very convenient for FPGAs:
  - GateKeeper can exploit the **parallelism** provided by FPGAs, allowing many concurrent computations
  - The algorithm uses mainly bit operations (XORs, ANDs, ORs) These are very efficient on FPGAs by using **Lookup Tables** (LUT)
    - For example, the amend operation on the FPGA is **407** times faster than the SHD CPU implementation

# Key Results:

## Methodology and Evaluation

# Results

- GateKeeper is **130x** faster than SHD for 100 bp reads and **90x** faster than the Adjacency Filter.
- Additionally, GateKeeper keeps the **same accuracy** of SHD (which is better than Adjacency Filter)

**Table 4: Overall mrFAST mapping time (in hours) with and without a pre-alignment step, with an edit distance threshold of 5%.**

Read length / E	mrFAST version / pre-alignment type	Filtering & Verification time (speed-up)	Overall mapping time (speed-up)
100 bp / 5 edits	2.1 / No Pre-alignment	22.60 h (1x)	24.27 h (1x)
	2.6 / Adjacency Filter	5.65 h (4x)	7.31 h (3.3x)
	2.1 / GateKeeper	0.55 h (41x)	2.50 h (9.7x)
300 bp / 15 edits	2.1 / No Pre-alignment	0.94 h (1x)	1.02 h (1x)
	2.6 / Adjacency Filter	0.04 h (24x)	0.12 h (8x)
	2.1 / GateKeeper	0.003 h (279x)	0.09 h (11x)

Reduced the runtime by a factor of almost 10 of a state-of-the-art mapper.

# Summary

# Summary

---

- GateKeeper is the first open-source FPGA implementation of a **pre-alignment filter** for DNA short read mapping.
- Provides a huge speedup of up to **130x** compared to the previous state of the art software solution.
- Keeps the **same accuracy** as the previously most accurate filter.



# Strengths

# Strengths

---

- New solution to a big problem : solve DNA read mapping in a **faster** manner (this speedup compared to the CPU implementation **could be vital** for medical uses, and others)
- Uses FPGAs and a specially adapted algorithm to use them to the fullest (taking advantage of LUTs)
- Design is **scalable**; could add more processing cores in the future
- Some sequencers use FPGAs as well, so GateKeeper could be integrated into them
- Well-written, interesting and easy to understand paper

# Weaknesses

# Weaknesses

---

- The benefits of such a mechanism require an FPGA and advanced knowledge with computers, this may be **problematic for some biologists/genomicists/geneticists**
- The amendment of the random zeros is a simple “**hack**” to reduce the number of false positives, but there is **no explanation** why GateKeeper only flips the patterns 101 and 1001, what about 10001? And  $10^n1$ ?
- The paper can be **confusing at times** due to the use of a “supplementary material” document that is constantly referred to (but understandable as there was a page limit set by the publication journal)

# Thoughts and Ideas

# Thoughts and Ideas

---

- Use **custom hardware** to make it even faster
- Make a **complete package** that would contain the full DNA sequencing pipeline, this could accelerate the process and make it easier to use
- Make the **amending of zeros more granular** (maybe depending on the edit threshold  $E$ ?) but this would require a lot more hardware on an FPGA

# Takeaways

# Key Takeaways

---

- DNA read mapping is a very important problem that leads to our better understanding of our surroundings and can lead to many important discoveries
- FPGAs can be a very efficient solution for complex problems such as DNA pre-alignment



# Open Discussion/Questions?

# Open Discussion (1/3)

- Is there a more elegant solution for the insertion/deletion of base pairs, instead of amending/ANDing all the masks?
  - ❑ "MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering", Alser et al., AACBB 2018, TIR 2017
  - ❑ Select longest sequence of consecutive zeros, pass it to the result mask, continue on the remaining columns with the **divide-and-conquer** approach

[illegible]

# Open Discussion (2/3)

---

- Another approach:
  - “SLIDER: Fast and Efficient Computation of Banded Sequence Alignment”, Alser et al., ArXiv.org 2018
  - Same idea as MAGNET, but instead of doing a global count of zeros, limit it to a window of 4 bits and then **slide the window** 1 bit to the right.

# Open Discussion (3/3)

---

- Are FPGAs really the “ultimate” tool for pre-alignment?
  - “GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies”, Kim et al., BMC Genomics 2018
  - Use **in-memory processing** for filtering
  - New technology : special RAM needed that has in-memory processing units, this allows to greatly parallelize computations but it requires a large amount of RAM since the data needs to be preloaded into memory.

# Last Remarks

---

- Very recent research (papers presented here were published in 2017-2018)
- Pre-alignment filters are a relatively new idea

# Thank you for listening!

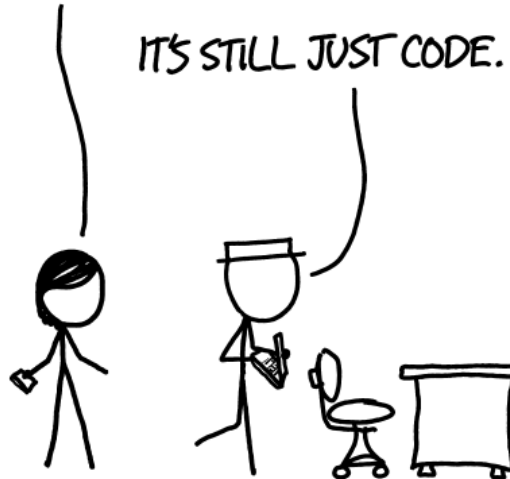
BIOLOGY IS LARGELY SOLVED.  
DNA IS THE SOURCE CODE  
FOR OUR BODIES. NOW THAT  
GENE SEQUENCING IS EASY,  
WE JUST HAVE TO READ IT.

IT'S NOT JUST "SOURCE  
CODE." THERE'S A TON  
OF FEEDBACK AND  
EXTERNAL PROCESSING.



BUT EVEN IF IT WERE, DNA IS THE  
RESULT OF THE MOST AGGRESSIVE  
OPTIMIZATION PROCESS IN THE  
UNIVERSE, RUNNING IN PARALLEL  
AT EVERY LEVEL, IN EVERY LIVING  
THING, FOR FOUR BILLION YEARS.

IT'S STILL JUST CODE.



OK, TRY OPENING GOOGLE.COM  
AND CLICKING "VIEW SOURCE."

OK, I-... OH MY GOD.

THAT'S JUST A FEW YEARS OF  
OPTIMIZATION BY GOOGLE DEVS.  
DNA IS THOUSANDS OF TIMES  
LONGER AND WAY, WAY WORSE.

WOW, BIOLOGY  
IS IMPOSSIBLE.

