Accelerating Approximate Pattern Matching with Processing-In-Memory (PIM) and Single-Instruction Multiple-Data (SIMD) Programming

Damla Senol Cali[1], Zülal Bingöl[2], Jeremie S. Kim[1,3], Rachata Ausavarungnirun[1], Saugata Ghose[1], Can Alkan[2] and Onur Mutlu[3,1]

[1] Carnegie Mellon University, Pittsburgh, PA, USA [2] Bilkent University, Ankara, Turkey [3] ETH Zürich, Zürich, Switzerland

Bitap Algorithm

Bitap algorithm (i.e., Shift-Or algorithm, or Baeza-Yates-Gonnet algorithm) [1] can perform exact string matching with **fast and simple bitwise operations.** Wu and Manber extended the algorithm [2] in order to perform **approximate string matching**.

- **Step 1 – Preprocessing:** For each character in the alphabet (i.e., A,C,G,T), generate a pattern bitmask that stores information about the presence of the corresponding character in the pattern.
- **Step 2 – Searching:** Compare all characters of the text with the pattern by using the preprocessed bitmasks, a set of bitvectors that hold the status of the partial matches and the bitwise operations.

[1] Baeza-Yates, Ricardo, and Gaston H. Gonnet. "A new approach to text searching." Communications of the ACM 35.10 (1992): 74-82.

[2] Wu, Sun, and Udi Manber. "Fast text search allowing errors." Communications of the ACM 35.10 (1992): 83-91.

Problem & Our Goal

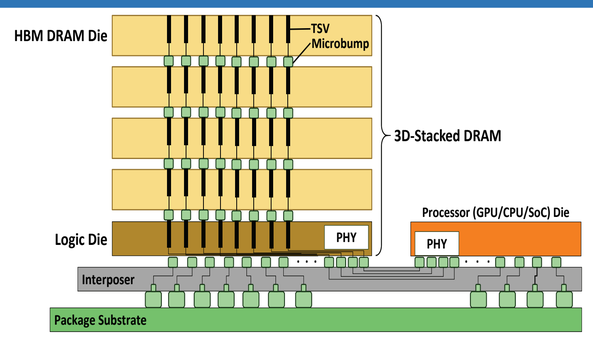
Problem:

- The operations used during *bitap* can be performed in **parallel**, but **high-throughput parallel** *bitap* computation requires a **large amount of memory bandwidth** that is currently **unavailable to the processor.**
- Read mapping is an application of approximate string matching problem, and thus can benefit from existing techniques used to optimize general-purpose string matching.

Our Goal:

- Overcoming memory bottleneck of *bitap* by performing **processing-in-memory** to exploit the **high internal bandwidth** available inside new and emerging memory technologies.
- Using **SIMD programming** to take advantage of the **high amount of parallelism** available in the *bitap* algorithm.

Processing-in-Memory



- Recent technology that tightly **couples memory and logic vertically** with **very high bandwidth** connectors.
- Numerous Through Silicon Vias (TSVs) connecting layers, enable **higher bandwidth** and **lower latency** and **energy consumption**.
- **Customizable logic layer** enables **fast, massively parallel operations** on large sets of data, and provides the ability to run these operations **near memory** to alleviate the memory bottleneck.

Acceleration of Bitap with PIM

Semantics of 0 and 1 are reversed from their conventional meanings throughout the bitap computation.

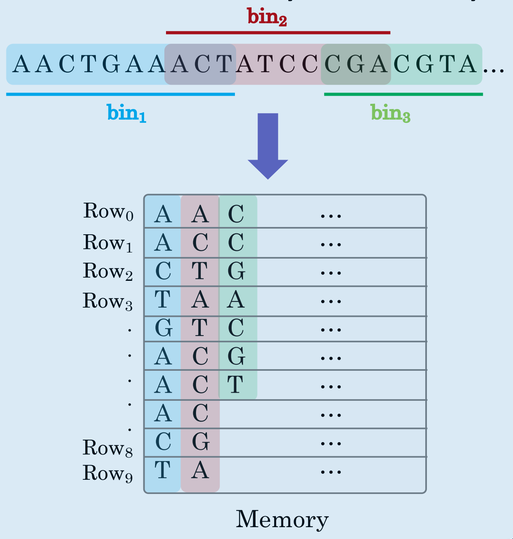
- 0 means match, 1 means mismatch

Text: AACTGAAACTATCCCGACGTA **Pattern:** ACG **Number of allowed errors (k):** 1

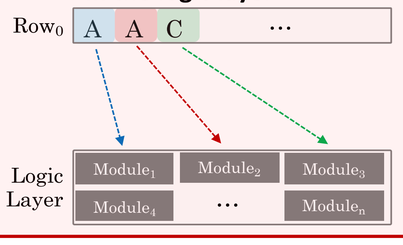
1) Generate the pattern bitmasks, initialize the status bitvectors, and store them within the logic layer

B[A] = 011 R[0] = 111
B[C] = 101 R[1] = 111
B[G] = 110
B[T] = 111

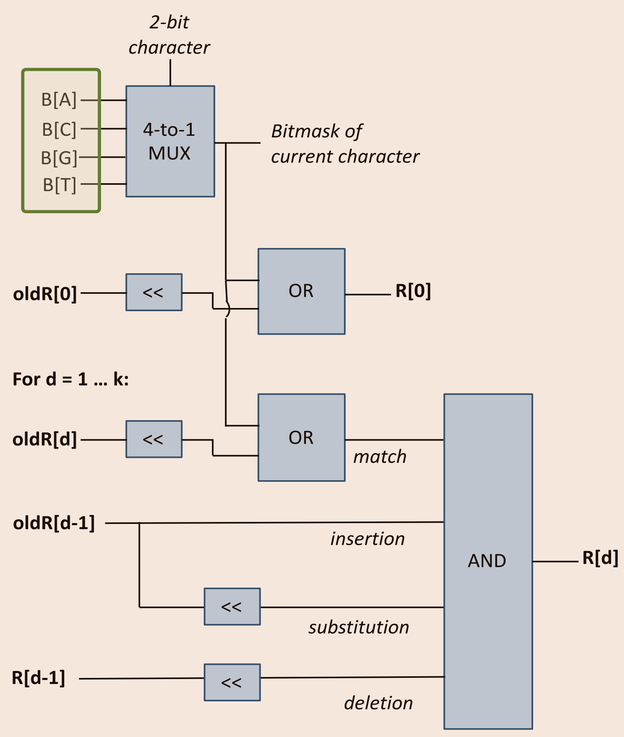
2) Split the text into overlapping bins and store each bin vertically within memory



3) Fetch one memory row and send each character (2-bit) to a separate logic module in the logic layer



4) Perform the computation within the logic module



5) Check the most significant bit of R[0], R[1], … , R[k]. If MSB of R[d] is 0, then there is a match between the text and the pattern with edit distance = d.

NOTES:

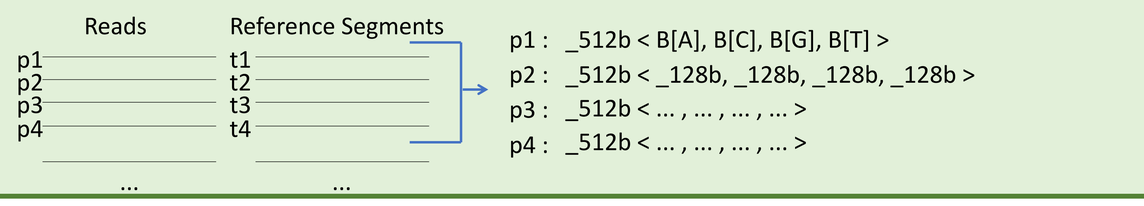
- **7k+2 bitwise operations** are completed sequentially for the computation of a single character in a bin. However, multiple characters from different bins are computed **in parallel** with the help of **multiple logic modules** (i.e., PIM accelerators) in the logic layer.
- If D is the number of iterations to complete the computation of one memory row, **D*(7k+2)** is the total number of bitwise ops per row, where D = (max # of accelerators) / (actual # of accelerators)

Acceleration of Bitap with SIMD

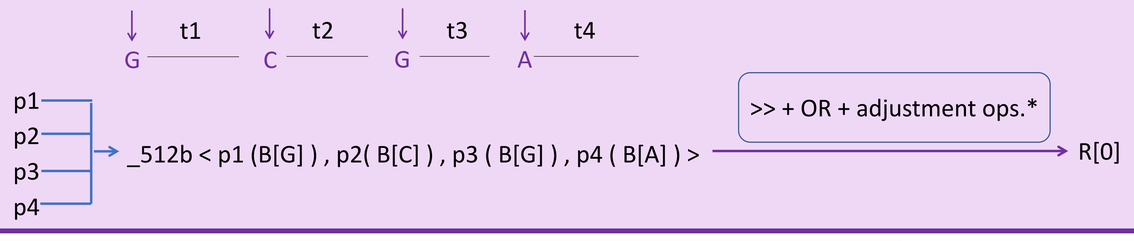
NOTES:

- **Intel Xeon Phi coprocessor** has vector processing unit which utilizes **Advanced Vector Extensions (AVX)** with an instruction set to perform effective SIMD operations.
- Our current architecture is **Knights Corner** and it enables usage of **512-bit vectors** performing 8 double precision or 16 single precision operations per single cycle.
- The recent system runs natively on a single MIC device and the read length must be **at most 128 characters.**

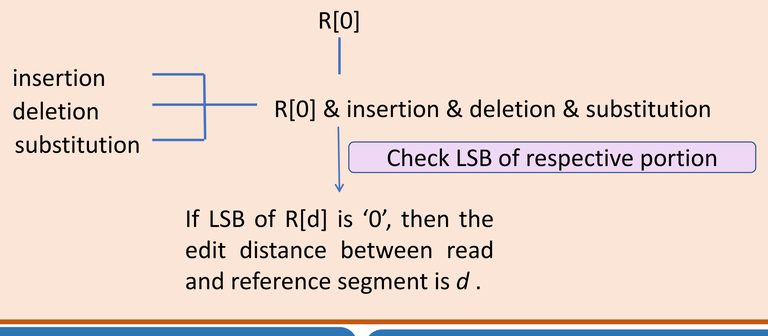
1) Get 4 pairs of reads and reference segments, prepare bitmasks of each read and assemble them into a vector.



2) Initialize status vectors, start iterating over 4 reference segments simultaneously. While iterating, assign the respective bitvectors of the reads as active and assemble them into a vector. Perform the bitwise operations to get R[0].



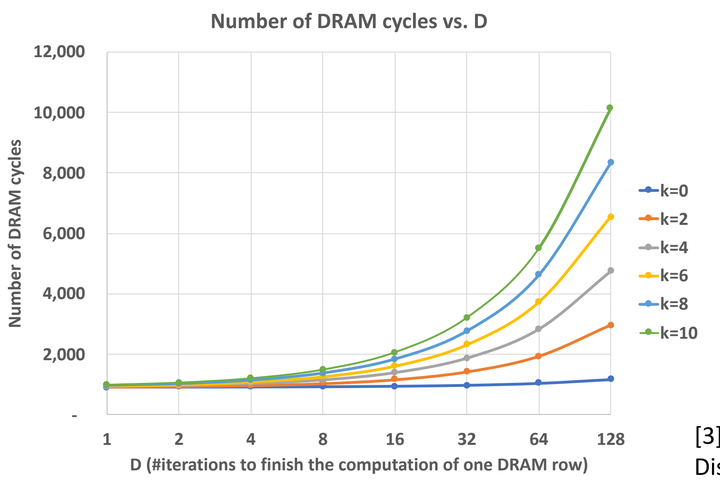
3) Integrate the result R[0] with insertion, deletion and substitution status vectors. Deactivate 128b portion of R[0]...R[d] if the respective *t* ends. Then, perform checking operations on the portion.



***Adjustment ops.:** Since the system represents entries with 128 bits and only 64-bit shift operation is supported by the instruction set, carry bit operations must be performed.

Results - PIM

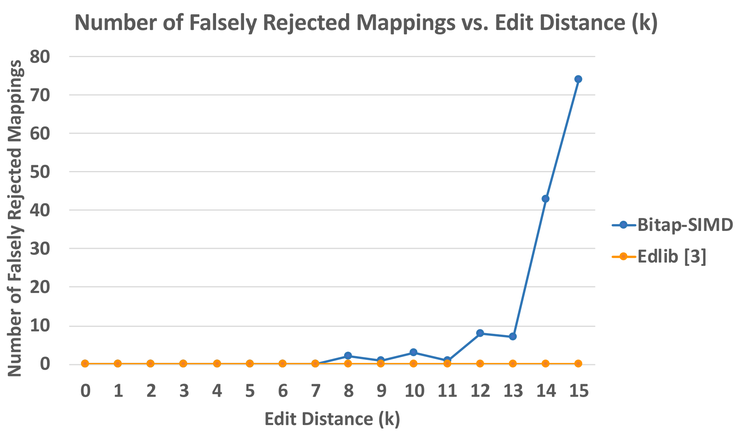
- Assuming a **row size** of 8 Kilobytes (65,536 bits) and a **cache line size** of 64 bytes (512 bits), there are 128 cache lines in a single row. Thus, **Memory Latency (ML)** = row miss latency + 127*(row hit latency) ~ 914 cycles. ML is constant (i.e., independent of # of accelerators).



- For the human chromosome 1 as the text and a read with 64bp as the pattern, Bitap-PIM provides **3.35x end-to-end speedup** over Edlib [3], on average.

Results - SIMD

- We perform the tests with read and reference segment pairs with 100bp long each. The total number of tested mappings is 3,000,000.



[3] Šošić, Martin, and Mile Šikić. "Edlib: A C/C ++ Library for Fast, Exact Sequence Alignment Using Edit Distance." Bioinformatics 33.9 (2017): 1394–1395.

Future Work

Bitap-PIM:

- Improving the logic module in the logic layer in order to decrease the number of operations performed within a DRAM cycle.
- Providing a backtracing extension in order to generate CIGAR strings.
- Comparing Bitap-PIM the with state-of-the-art read mappers for both short and long reads.

Bitap-SIMD:

- Extending the current system to work in offload mode for exploiting 4 MIC devices simultaneously.
- Optimizing the expensive adjustment operations (i.e., carry bit operations) to improve the performance of Bitap-SIMD.