BitMAC: An In-Memory Accelerator for Bitvector-Based Sequence Alignment of Both Short and Long Genomic Reads

Damla Senol Cali[1], Gurpreet S. Kalsi[2]. Lavanya Subramanian[3], Can Firtina[4], Anant V. Nori[2], Jeremie S. Kim[4,1], Zulal Bingöl[5], Rachata Ausavarungnirun[6,1], Mohammed Alser[4], Juan Gomez-Luna[4], Amirali Boroumand[1], Allison Scibisz[1], Sreenivas Subramoney[2], Can Alkan[5], Saugata Ghose[1], and Onur Mutlu[4,1]

[1] Carnegie Mellon University, USA [2] Intel, USA [3] Facebook, USA [4] ETH Zürich, Switzerland
[5] Bilkent University, Turkey [6] King Mongkut's University of Technology North Bangkok, Thailand

Problem

- **Read mapping** is the **critical first step** of the genome sequence analysis pipeline.
- In read mapping, each read is aligned against a reference genome to verify whether the potential location results in an alignment for the read (i.e., ***read alignment***).
- Read alignment can be viewed as an approximate (i.e., fuzzy) string matching algorithm.
- Approximate string matching is typically performed with an expensive quadratic-time dynamic programming algorithm, which consumes over 70% of the execution time of read alignment.

String Matching with 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.

Limitations of Bitap

(1) The algorithm itself cannot be parallelized due to data dependencies across loop iterations,
(2) Multiple searches can be done in parallel, but are limited by the number of compute units available in a CPU,
(3) Even if many compute units are made available (e.g., a GPU), Bitap is highly constrained by the amount of available memory bandwidth.

Also, standard Bitap algorithm *cannot*
(1) perform the **traceback step** of the alignment.
(2) work effectively for **both short and long reads**.

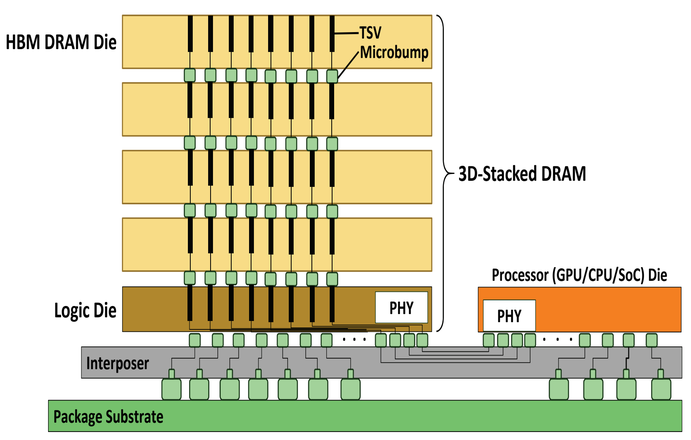
Our Goal and Contributions

Goal: Designing a fast and efficient customized accelerator for approximate string matching to enable faster read alignment, and therefore faster genome sequence analysis for both short and long reads.

Contributions:

- Modified Bitap algorithm to perform efficient genome read alignment in memory, by
  - ✓ parallelizing the Bitap algorithm by removing data dependencies across loop iterations,
  - ✓ adding efficient support for long reads, and
  - ✓ developing the first efficient Bitap-based algorithm for traceback.
- **BitMAC**, the first in-memory read alignment accelerator for both short accurate and long noisy reads.
  - ✓ Hardware implementation of our modified Bitap algorithm, and
  - ✓ Designed specifically to take advantage of the high internal bandwidth available in the **logic layer of 3D-stacked DRAM** chips.

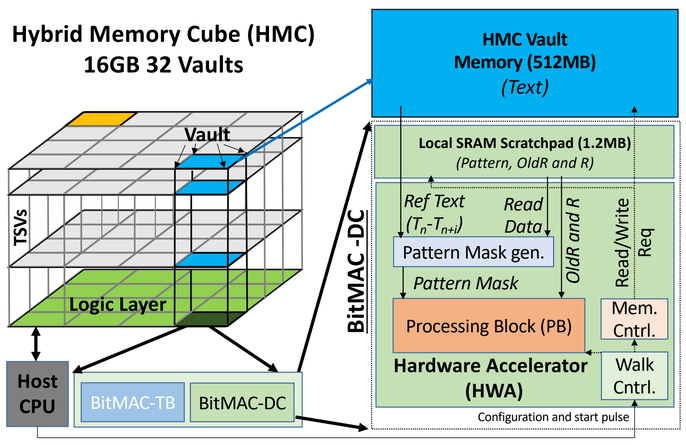
3D-Stacked DRAM and Processing-in-Memory (PIM)



- Recent technological advances in memory design allow architects to tightly **couple memory and logic within the same chip** with very high bandwidth, low latency and energy-efficient vertical connectors.
  → 3D-stacked memories (Hybrid Memory Cube (HMC), High-Bandwidth Memory (HBM))
- A **customizable logic layer** enables fast, massively parallel operations on large sets of data, and provides the ability to run these operations near memory at high bandwidth and low latency.
  → Processing-in-memory (PIM)

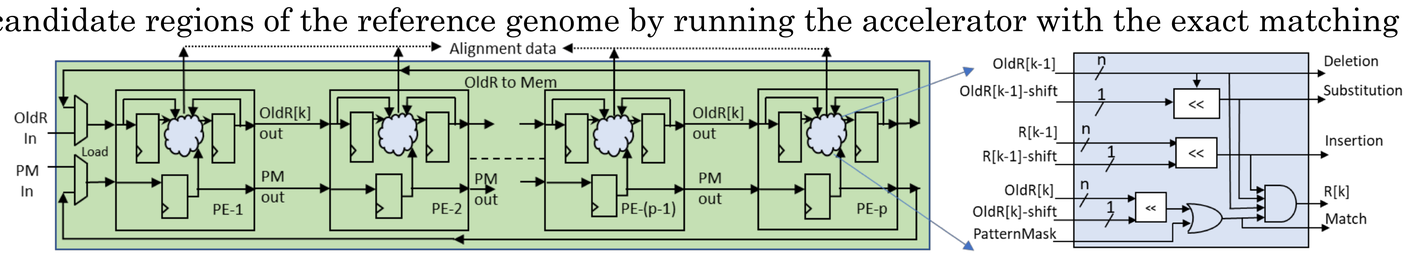
BitMAC: Efficient In-Memory Bitap

- **BitMAC**: implementation of our modified PIM-friendly Bitap algorithm using a dedicated hardware accelerator that we add to the logic layer of a 3D-stacked DRAM chip.
  (1) achieves high performance and energy efficiency with specialized compute units and data locality,
  (2) balances the compute resources and available memory bandwidth per compute unit,
  (3) scales linearly with the number of parallel compute units,
  (4) provides generic applicability by performing both edit distance calculation and traceback for short and long reads.
- **BitMAC** consists of two components that work together to perform read alignment:
  (1) **BitMAC-DC**: calculates the **edit distance** between the reference genome and the query read, and
  (2) **BitMAC-TB**: performs **traceback** using the list of edit locations recorded by BitMAC-DC.



BitMAC-DC:

- **Systolic array based configuration** for an efficient implementation of edit distance calculation. This allows us to provide parallelism across multiple iterations of read alignment.
  (1) computes the edit distance between the whole reference genome and the input reads,
  (2) computes the edit distance between the candidate regions reported by an initial filtering step and the input reads, and also
  (3) finds the candidate regions of the reference genome by running the accelerator with the exact matching mode.



(a) Processing Block (PB) (b) Processing Core (PC)

BitMAC-TB:

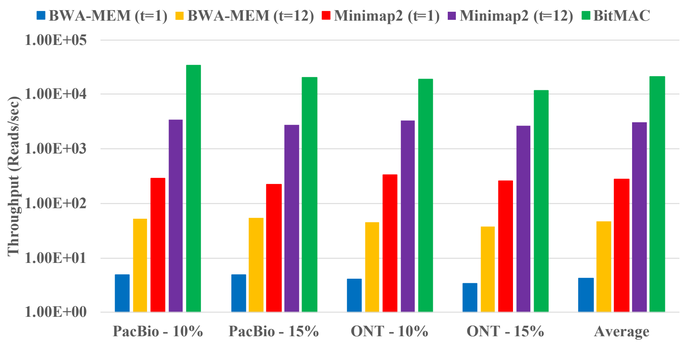
- Makes use of a **low-power general-purpose PIM core** to perform the traceback step of read alignment.
- New, efficient algorithm for traceback, which exploits the bitvectors generated by BitMAC-DC.
- Divides the matching region of the text (as identified by BitMAC-DC) into multiple windows, and then performs traceback in parallel on each window. This allows us to utilize the full memory bandwidth for the traceback step.

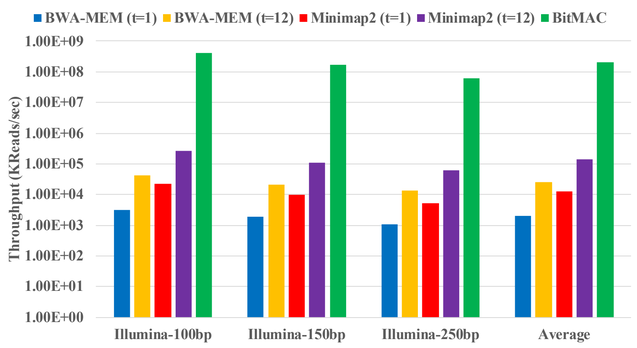
Results

When compared with the alignment steps of BWA-MEM and Minimap2, **BitMAC achieves:**

For simulated PacBio and ONT datasets:
- ✓ 4997× and 79× throughput improvement → over the single-threaded baseline
- ✓ 455× and 7× throughput improvement → over the 12-threaded baseline
- ✓ 15.9× and 13.8× less power consumption

For simulated Illumina datasets:
- ✓ 102,558× and 16,867× throughput improvement → over the single-threaded baseline
- ✓ 8162× and 1445× throughput improvement → over the 12-threaded baseline





Future Work

We believe it is promising to explore:

- coupling PIM-based filtering methods with BitMAC to reduce the amount of required read alignments,
- enhancing the design of BitMAC to support different scoring schemas and affine gap penalties,
- analyzing the effects of a larger alphabet on BitMAC, and
- examining the benefit of using AVX-512 support for multiple pattern searches in parallel.