ReTAP: Processing-in-ReRAM Bitap Approximate String Matching Accelerator for Genomic Analysis

Tsung-Yu Liu*§, Yen An Lu†§, James Yu‡, Chin-Fu Nien‖¶, Hsiang-Yun Cheng*¶
*Academia Sinica, Taiwan †Cornell University, USA ‡Georgia Institute of Technology, USA ‖Chang Gung University, Taiwan
Email: *{tsungyuliu, hycheng}@citi.sinica.edu.tw, †yl3726@cornell.edu, ‡jyu678@gatech.edu, ‖watchmannien@mail.cgu.edu.tw

***Abstract*—Read mapping, which involves computationally intensive approximate string matching (ASM) on large datasets, is the primary performance bottleneck in genome sequence analysis. To accelerate read mapping, a processing-in-memory (PIM) architecture that conducts highly parallel computations within the memory to reduce energy-inefficient data movements can be a promising solution. In this paper, we present ReTAP, a processing-in-ReRAM Bitap accelerator for genomic analysis. Instead of using the intricate dynamic programming algorithm, our design incorporates the Bitap algorithm, which uses only simple bitwise operations to perform ASM. Additionally, we explore the opportunity to reduce redundant computations by dynamically adjusting the error tolerance of Bitap and co-design the hardware to enhance computation parallelism. Our evaluation demonstrates that ReTAP outperforms GenASM, the state-of-the-art Bitap accelerator, with a $153.7\times$ higher throughput.**

I. Introduction

Genome sequencing has revolutionized our understanding of biology and genetics [1]–[3]. In genome sequence analysis, the primary bottleneck is read mapping, a process involving computationally intensive approximate string matching (ASM) on large datasets, mapping terabytes of genome fragments (reads) to the reference genome consisting of billions of nucleotide bases. While algorithmic optimizations [4], [5] have been proposed to alleviate the computational burden of the typical dynamic programming (DP)-based ASM algorithms, they do not fully resolve the inefficiencies associated with moving large volumes of genomic data between computation and memory units.

Processing-in-memory, a rising paradigm in computer architecture that performs computations within memory chips, appears to be promising for addressing the memory wall challenge. While many prior PIM-based ASM accelerators are designed with computationally expensive DP-based algorithms [6], [7], an earlier work introduced GenASM [8], a cutting-edge 3D-stacked PIM accelerator that employs the Bitap algorithm with simple bitwise operations to avoid complex hardware implementation. However, the performance of GenASM is impeded by two limitations. First, its parallelism is restricted by the number of independently operated memory subdivisions capable of performing ASM in parallel (e.g., 32 vaults in 16GB HMC [8]). Second, its implementation of the Bitap algorithm introduces redundant computations due to the conservatively selected high error tolerance for aligning all read-reference pairs.

§ Yen An Lu and Tsung-Yu Liu are co-first authors. ¶ Hsiang-Yun Cheng and Chin-Fu Nien are co-corresponding authors. This work is supported by the National Science and Technology Council of Taiwan (112-2221-E-182-031, 112-2221-E-001-002-MY3)

In this paper, we propose ReTAP, a processing-in-ReRAM Bitap accelerator designed for genomic analysis. In comparison to 3D-stacked PIM, resistive random access memory (ReRAM), capable of performing bitwise logic operations within its memory array by adjusting the input voltage [6], offers increased computation parallelism for Bitap under identical memory capacity. Moreover, we identify an opportunity to dynamically adjust the error tolerance of Bitap for each individual read-reference pair, thus avoiding redundant computations. Through the co-design of the optimized Bitap algorithm with the ReRAM-based PIM architecture, ReTAP achieves $153.7\times$ and $40.88\times$ higher throughput than GenASM, the state-of-the-art Bitap accelerator, for short and long reads, respectively.

II. Proposed ReTAP Architecture

A. Bitap Algorithm Optimization

An ASM algorithm typically comprises two phases: a forward phase called edit distance calculation (EDC) and a backward phase called traceback (TB). In the Bitap algorithm [4], [5], the EDC step involves a nested loop. The outer loop iterates through the length of the genome sequences, while the inner loop assesses the potential occurrence of deletion (D), substitution (S), insertion (I), and match (M) at each position. The Bitap algorithm's error tolerance k controls the number of allowed errors (D, S, and I) of the inner loop.

Concerning hardware design considerations, a naïve implementation of the Bitap algorithm may lead to computational inefficiencies and wastage of cycle time, particularly when a small error tolerance k is sufficient for successful alignment between read and reference sequences. Especially when a pre-alignment filtering step is performed, a naïve implementation of the Bitap algorithm may result in computational inefficiencies. For example, in the 'Illumina' dataset consisting of short reads, over 86.2% of read-reference pairs have error distances of less than 3 bases, and very few exceed 7. Similarly, in long-read datasets like 'Ont' and 'Pacbio,' fewer than 12.0% of pairs have error distances greater than 7 bases, with even fewer exceeding 15. Therefore, we propose an optimized Bitap algorithm co-designed with the architecture detailed in the following section. Our algorithm facilitates dynamic search through k during runtime. It starts with a smaller k value and dynamically increases k whenever the previous k fails to cover the number of errors. This approach allows us to potentially identify a smaller k resulting in shorter execution times. However, if the number of errors surpasses the current k, the results produced by that k are discarded, akin to the misprediction penalty in traditional computer architecture.

979-8-3503-4859-0/DATE24/© 2024 EDAA

B. ReTAP overview

In this work, we propose ReTAP, a processing-in-ReRAM Bitap ASM accelerator for genomic analysis, which incorporates algorithm optimization discussed in the previous subsection. Fig. 1 shows the architecture overview of ReTAP. A ReTAP accelerator *chip* comprises 64 *tiles*. A tile contains 16 *compute units (CUs)*, each serving as the primary unit for scheduling the optimized Bitap alignment task. Within each CU, there are multiple *processing elements (PEs)*, and each PE is equipped with four *crossbars (XBs)*. These PEs serve as the fundamental units for executing the EDC step to align a read sequence to a reference sequence in the proposed optimized Bitap algorithm. To handle different values of k in the EDC step, varying numbers of processing elements (PEs) are employed. For instance, $k = 3$ is handled by a single PE, while $k = 15$ requires four PEs to perform the EDC computation. The XBs in each PE execute bitwise operations by controlling the voltage across the output switching device [6] to realize the EDC calculation. Additionally, besides co-designing ReTAP with the proposed optimized Bitap algorithm, we introduce two techniques to enhance throughput. The first is a memory mapping strategy, where we map two sequences within a row of the 128×128 XB to achieve a $2\times$ throughput. The second is a parallelism enhancement technique, where we allocate different read-reference pairs to the PEs within a CU to keep each PE busy.

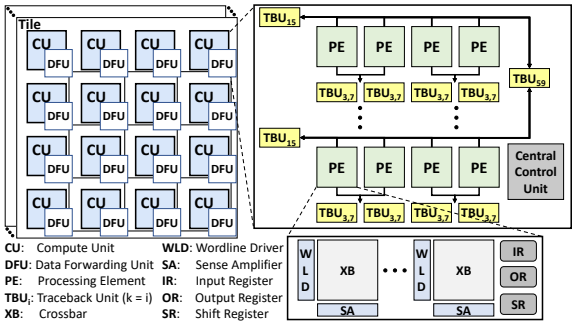

Fig. 1. ReTAP architecture overview

III. Evaluation and Results

A. Evaluation Setup

We simulate the behavior ReTAP architecture by modifying the MNSIM [9] for performing Bitap operations. We use the Design Compiler to synthesize digital components in the TSMC 180 nm process node and then scale them down to 45 nm. We evaluate ReTAP using generated long reads (PacBio and ONT datasets) from PBSIM [2] and generated short reads (Illumina datasets) from Mason [3], following a prior work [8]. We use the X and Y chromosomes of a latest-release human genome assembly, GRCh38 [10], as the reference genome. We compare ReTAP with several baseline methods, including the commonly-used CPU aligners BWA-MEM [1] and Minimap2 [11]. Additionally, we include the state-of-the-art Bitap aligner GenASM [8].

B. Experimental Results

Fig. 2(a) shows the throughput comparison for long reads. Additionally, when compared to Minimap2, ReTAP demonstrates a throughput improvement of $76.0\times$ and outperforms BWA-MEM by a factor of $4332.2\times$. When compared to GenASM, a single chip ReTAP's throughput for long reads slightly falls behind GenASM's performance due to GenASM's 16GB chip capacity, which is much larger than ReTAP's. Scaling ReTAP to 64 chips to match the 16GB capacity results in a throughput surpassing GenASM by a factor of 40.88, thanks to the two design strategies of high ReRAM technology parallelism and the optimization algorithm proposed for ReTAP. For short-read sequences, Fig. 2(b) indicates that a single-chip ReTAP, when compared to Minimap2, achieves an impressive throughput improvement of $349.0\times$ and outshines BWA-MEM with an astonishing factor of $2229.6\times$, while excelling GenASM by $2.4\times$. Scaling to a 64-chip design results in a throughput $153.7\times$ greater than GenASM.

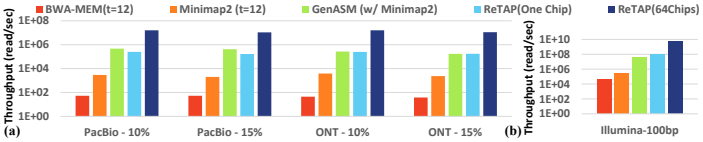

Fig. 2. Throughput comparison for (a) long-read and (b) short-read data.

IV. Conclusion

This paper proposes a novel approach, ReTAP, which leverages processing-in-ReRAM architecture and optimized Bitap algorithm to improve compute parallelism. We address the limitations of existing ReRAM-based genomics accelerators, such as low parallelism and excessive computation. Our architecture co-designs the hardware with the optimized Bitap algorithm, making it efficient in reducing computation workload while maintaining accuracy. To our knowledge, ReTAP is the first ReRAM-based accelerator that accelerates the Bitap algorithm for genomic application. The experimental results demonstrate that the proposed ReTAP achieves $153.7\times$ and $40.88\times$ higher throughput than GenASM, the state-of-the-art Bitap accelerator, for short and long reads, respectively.

References

[1] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," 2013.

[2] Y. Ono *et al.*, "PBSIM: PacBio reads simulator-toward accurate genome assembly," *Bioinformatics*, vol. 29, no. 1, pp. 119–121, 11 2012.

[3] M. Holtgrewe, "Mason-a read simulator for second generation sequencing data," *Technical Report FU Berlin*, 2010.

[4] R. Baeza-Yates and G. H. Gonnet, "A new approach to text searching," *Commun. ACM*, vol. 35, no. 10, p. 74–82, oct 1992.

[5] S. Wu and U. Manber, "Fast text searching: Allowing errors," *Commun. ACM*, vol. 35, no. 10, p. 83–91, oct 1992.

[6] S. Gupta *et al.*, "RAPID: A ReRAM processing in-memory architecture for DNA sequence alignment," in *ISLPED*, 2019, pp. 1–6.

[7] R. Kaplan *et al.*, "BioSEAL: In-memory biological sequence alignment accelerator for large-scale genomic data," in *SYSTOR*, 2020, p. 36–48.

[8] D. S. Cali *et al.*, "GenASM: A high-performance, low-power approximate string matching acceleration framework for genome sequence analysis," in *MICRO*, 2020, pp. 951–966.

[9] Z. Zhu *et al.*, "MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *GLSVLSI*, 2020, p. 83–88.

[10] N. C. for Biotechnology Information, "Genome assembly GRCh38.p13," https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_000001405.39/, 2023.

[11] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 05 2018.