RADAR: A 3D-ReRAM based DNA Alignment Accelerator Architecture

Wenqin Huangfu, Shuangchen Li, Xing Hu, Yuan Xie
University of California at Santa Barbara
yuanxie@ece.ucsb.edu

ABSTRACT

Next Generation Sequencing (NGS) technology has become an indispensable tool for studying genomics, resulting in an exponentially growth of biological data. Booming data volume demands significant computational resources and creates challenges for 'Sequence Alignment', which is the most fundamental application in bioinformatics. Consequently, many researchers exploit both software and hardware methods to accelerate the most widely used sequence alignment algorithm – Basic Local Alignment Search Tool (BLAST). However, prior work suffers from moving huge DNA databases from the storage to computational units. Such data movement is both time and energy consuming.

Based on the observation that the bottlenecks of BLAST involve a large amount of comparison operations, we propose a 3D Resistive Random Access Memory (ReRAM) based DNA Alignment Accelerator Architecture (RADAR) which performs most computational operations locally without moving DNA databases. To improve the storage density for various lengths of DNA sequences without damaging the performance, we propose a dense data mapping scheme to handle DNA sequences efficiently and a Tail Bits Duplication (TBD) technique to enable fully parallel computation for RADAR. Experimental results show that RADAR can achieve 5114x speedup and 386x energy reduction when compared to a single CPU. Compared to the Multi-Core/FPGA/GPU based accelerators, RADAR outperforms them between 53x and 1896x in processing speed.

1 INTRODUCTION

In the past 30 years, biological data has been growing exponentially, primarily due to Next Generation Sequencing (NGS) technology [18]. For example, GenBank [7] of the National Center for Biotechnology Information (NCBI) has doubled its size approximately every 18 months and now contains more than 245 billion characters. This bio-data explosion introduces challenges to the most fundamental bio-application, 'Sequence Alignment', which is critical for solving diverse tasks of bioinformatics, ranging from motif finding [2] to metagenomics clustering [9]. BLAST, the Basic Local Alignment Search Tool [1], is the most widely used sequence alignment tool. To keep pace with booming bio-data, acceleration of BLASTN, the most heavily used DNA version of BLAST, is necessary.

Software approaches [17] provide speedup for BLASTN by substantially sacrificing sensitivity or by indexing the entire database with large overhead. In response to the shortcomings of the software approaches, many hardware approaches, such as multi-core [3], FPGA [12], and GPU based accelerators [15], have been proposed to further accelerate BLASTN by leveraging its inner parallelism. However, these approaches do not address the issues of energy and latency overheads which are caused by moving the entire DNA database from memory to CPU/FPGA/GPU. Resistive Random Access Memory (ReRAM) based Content Addressable Memory (CAM) PRocessing-in-Storage (PRinS) [11] solves the data movement problem in the Smith-Waterman (SW) algorithm based on associative computing. Unfortunately, lifetime and storage efficiency are serious issues in ReRAM-based CAM (ReCAM) PRinS, because associative processing inherently requires many write operations and uses 75% of the ReRAM cells to store the intermediate data [23]. Last, the SW algorithm, which ReCAM PRinS focuses on, is different from BLASTN: the SW algorithm is used in the last stage of BLASTN and only consumes less than 1% of BLASTN's runtime [12].

Observing that the execution of BLASTN is dominated by comparison operations and introduces huge amount of data movement, we identify that 3D ReCAM [14][22] is extremely suitable for the acceleration of BLASTN due to its high density, low power consumption, and ability to perform parallel comparisons locally. We propose RADAR, a Processing-In-Memory (PIM) architecture which utilizes ReCAM with the aid of ASIC units, to accelerate BLASTN without encountering the problem of write endurance. However, mapping DNA sequences into 3D ReCAMs is non-trivial due to huge variations in the lengths of DNA sequences, which may affect both storage and computation efficiency. Hence, we propose efficient data mapping methods to improve efficiency of both storage and computation in RADAR.

The main contributions of this paper are:

- We propose RADAR, a novel PIM architecture that utilizes 3D ReCAM to accelerate BLASTN efficiently with high storage density and efficiency. RADAR reduces data migration and incurs less burden on write endurance to improve the lifetime of ReRAM.
- We design a dense data mapping scheme to handle DNA sequences with various lengths efficiently in RADAR and we propose a Tail Bits Duplication (TBD) technique to eliminate the row-level, CAM-level, and Unit-level data dependencies and communication to support fully parallel computation.
- We conduct extensive design space exploration to study the trade-offs between response time, energy consumption, energy efficiency, and area of RADAR.

This work was supported in part by DOE DE-SC0013553, NSF 1533933 and 1719160.

DAC '18, June 24–29, 2018, San Francisco, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196098

2 BACKGROUND AND MOTIVATION

This section introduces the basics of BLASTN, 3D ReCAM and the motivation for this work.

2.1 BLASTN

As a DNA sequence alignment tool, BLASTN can be divided into three stages as shown in Fig. 1(a). The input of BLASTN [1] is a query DNA sequence which needs to be compared against the DNA database, and the output of BLASTN are gapped alignments generated after the third stage.

- Stage 1 - Word Matching: Subsequences of fixed length w (typically $w = 11$ for DNA) are precisely located in DNA databases. The located sequence can be precisely matched to the query DNA sequence. These short matches are referred as w-mers and are forwarded to the second stage - 'Ungapped Extension'.
- Stage 2 - Ungapped Extension: w-mers forwarded from the 'Word Matching' stage are extended on both sides to identify longer pairs of sequences around the w-mers, which should have more matches and fewer mismatches. These longer pairs are called High-Scoring Segment Pairs (HSPs) and will be passed to the third stage - 'Gapped Extension'.
- Stage 3 - Gapped Extension: Dynamic programming algorithms, like the SW algorithm [19], are used to extend the HSPs into gapped alignments, which are sequence pairs that differ only by a few mismatches and gaps.

The bottlenecks of BLASTN are 'Wording Matching' and 'Ungapped Extension', which consume 83.9% and 15.9% of the runtime [12], respectively. The dominant operations in these two stages are comparisons. Consequently, the acceleration of BLASTN should focus on those two stages and comparison operations.

Accelerating BLASTN in general purpose computing platforms is not efficient due to the following reasons:

- With a large volume of data, many independent, simple operations with significant parallelism dominate BLASTN, while current general-purpose processors waste efforts by supporting complex operations that are overdesigned for BLASTN [10].
- The whole DNA database needs to be moved from storage to the computational units, introducing huge data movement in current hardware design. Previous work shows that data transfer translates into performance penalties in both processing speedup and energy consumption [10].

2.2 3D ReCAM

Parallel match/mismatch operations provided by CAM make it a good candidate for search-based applications. With a small cell size, non-volatility, zero standby power, and the potential to be stacked in 3D [14] [22], the emerging ReCAM provides an alternative solution to SRAM-based CAM.

The structure of 3D ReCAM is shown in Fig. 1(e). The vertical pillar electrode and the surrounding metal oxide forms the metal-oxide-metal structure of ReRAM cells, when the metal oxide is in contact with the horizontal plane electrodes. Multiple bits can be stored in one pillar, because 3D ReCAM has multiple horizontal layers. The horizontal plane is used as the Source Line (**SrL**) and a row of pillars are connected together to a Match Line (**ML**). The match/mismatch signal from a certain pillar will be sent to the Sense Amplifier (SA) for reading. A comparison key can slide through the input ports (**SrL**) to search the data stored in 3D ReCAM. Only one access transistor is needed per pillar. This is an extremely high-density, multi-layer, and transistor-less design of non-volatile CAM. It works in column-serial, row-parallel mode.

3D ReCAM, which performs parallel comparisons locally, is extremely suitable for accelerating BLASTN. In addition, with extremely high density, 3D ReCAMs can store the entire DNA database and perform computation locally. One important issue of ReCAM is its limited write endurance, thus write operations in ReCAM should be minimized [13][23].

This work aims to leverage the potential of 3D ReCAM to accelerate the 'Word Matching' and 'Ungapped Extension' stages of BLASTN without encountering the endurance issue of ReCAM.

3 RADAR ACCELERATOR

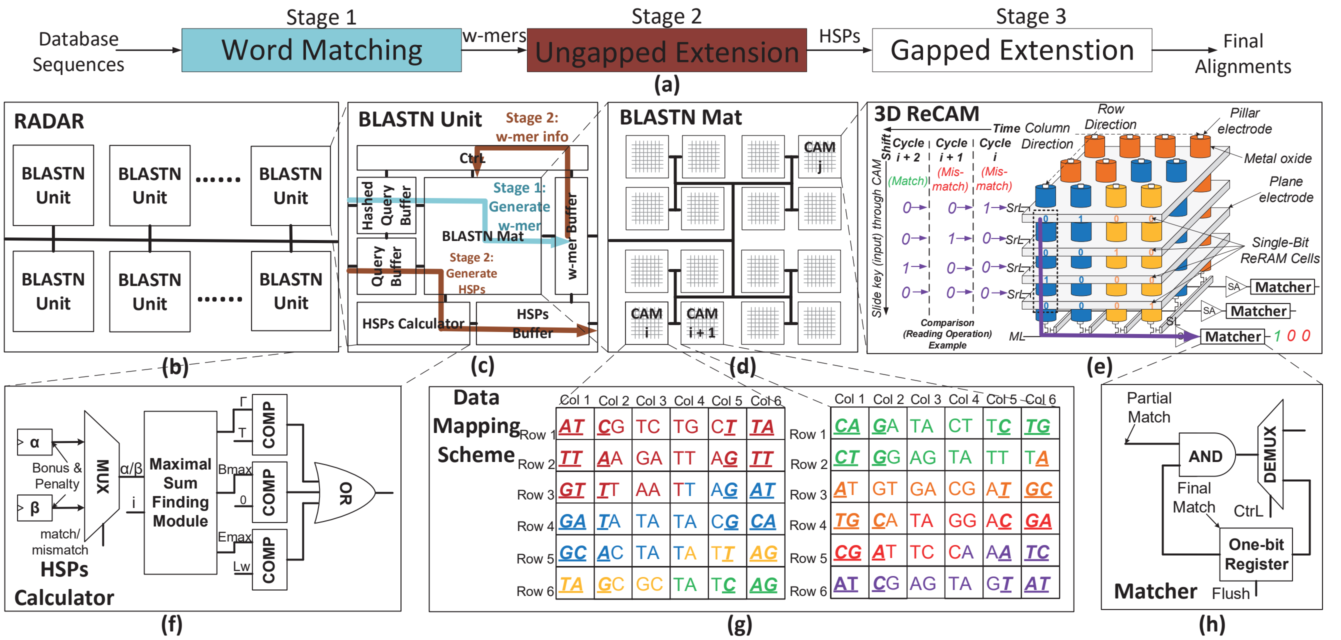
This section introduces the architecture design, presents the data flow in 'Word Matching' and 'Ungapped Extension', and gives some insight into important design factors of RADAR.

3.1 Overview

The high-level architecture of RADAR is shown in Fig. 1(b). RADAR consists of multiple BLASTN Units, the fundamental computational blocks. BLASTN Units receive hashed and original query sequence from the CPU. Hashed query sequences contain hashed words with length w from the query sequence and is used for 'Word Matching'. The original query sequence is used for 'Ungapped Extension'. After those two stages, the BLASTN Units will output results to the CPU. All BLASTN Units can perform computation independently and in parallel, providing unit-level parallelism. A BLASTN Unit consists of a BLASTN Mat, buffers, and a HSPs Calculator. The BLAST Mat stores the DNA database and performs comparison locally in 3D ReCAMs. The buffers in the BLASTN Units store the inputs (hashed and original query sequences), intermediate data (w-mers), and the outputs (HSPs). Verification of HSPs is performed in a customized HSPs Calculator.

3.2 Architecture

BLASTN Unit: The architecture of a BLASTN Unit is shown in Fig. 1(c). The hashed query buffer and the query buffer are connected to a bus to receive the hashed and the original query sequences from the CPU, and to forward the query information to the BLASTN Mat. The BLASTN Mat is connected to the w-mer buffer and the HSPs Calculator. After 'Word Matching', the BLASTN Mat gets the information of w-mers and sends it to the w-mer buffer. During 'Ungapped Extension', the information of the w-mers are re-sent to the BLASTN Mat along with the original query sequence to get HSPs candidates. Then the HSPs Calculator can help verify these HSPs candidates and output the valid HSPs to the local HSPs buffer. The HSPs buffer stores HSPs temporarily and then passes them to host CPU via bus. Finally, a controller is connected to these modules to regulate the computational process.



	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Row 1	**AT**	**C**G	TC	TG	C**T**	**TA**
Row 2	**TT**	**A**A	GA	TT	A**G**	**TT**
Row 3	**GT**	**T**T	AA	TT	A**G**	**AT**
Row 4	**GA**	**T**A	TA	TA	C**G**	**CA**
Row 5	**GC**	**A**C	TA	TA	T**T**	**AG**
Row 6	**TA**	**G**C	GC	TA	T**C**	**AG**

	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Row 1	**CA**	**G**A	TA	CT	T**C**	**TG**
Row 2	**CT**	**G**G	AG	TA	TT	T**A**
Row 3	**A**T	GT	GA	CG	A**T**	**GC**
Row 4	**TG**	**C**A	TA	GG	A**C**	**GA**
Row 5	**CG**	**A**T	TC	CA	A**A**	**TC**
Row 6	**AT**	**C**G	AG	TA	G**T**	**AT**

Figure 1: Three stages of BLASTN & architecture, data flow and data mapping scheme of RADAR. The entire DNA database and the majority of computation is stored and performed in 3D ReCAMs. (a) The first two stages are bottlenecks of BLASTN. (b) Architecture of RADAR. (c) BLASTN Unit. The blue arrow indicates the data flow of 'Word Matching' and the green arrow indicates the data flow of 'Ungapped Extension'. (d) BLASTN Mat, H-tree is used to connect different 3D ReCAMs. (e) 3D ReCAM. DNA database stored inside. Different colors represent different DNA sequences. A comparison key shifts through CAM to perform comparison. The purple arrow indicates the data flow. (f) Circuit of HSPs Calculator. (g) Representation of data mapping scheme, including the dense data mapping method and the TBD technique (The actual *w* should be 11, 4 is used in this figure just as an example for illustration). Multiple DNA bases can be stored in one pillar. Different colors represent different DNA sequences. Bases in bold and with underscores stand for tail bits duplicated in storage. (h) Matcher circuit. Results of partial comparisons are combined together to generate the final comparison result.

BLASTN Mat: The BLASTN Mat consists of multiple 3D ReCAMs connected to each other in an H-tree. 3D ReCAMs store the entire DNA database and implement the in-situ comparison with the search-target subsequence as the input and matching results as the output. The comparison operations can be performed in different ReCAM rows and different ReCAMs concurrently, enabling both row-level parallelism and CAM-level parallelism.

In each CAM, comparisons are performed in column-serial, row-parallel order. An example of a comparison is shown in Fig. 1(e), in which the comparison key is shifted by one position per cycle. If a match occurs, the output of the corresponding row will be 1. Otherwise, the outputs of all rows will be 0. The data flow of the example comparison operation is indicated by a purple arrow. If a comparison involves data stored in different vertical pillars, which will happen when the comparison window crosses multiple pillars, partial comparisons will be performed each time. A Matcher in Fig. 1(h) is designed to merge the partial comparison results. The final comparison result will be available when all partial comparisons have been performed. The data mapping scheme exerts influence on searching and will be discussed in detail in Section 5.

HSPs Calculator: First, the HSPs Calculator finds the maximal sum of scores in a scoring pair within a fixed-size window. Then, the HSPs Calculator compares the maximal sum with a preset threshold and determines if the generated scoring segment pairs are valid HSPs. The circuit design is shown in Fig. 1(f). The scores in scoring pairs are either a bonus α for matches or a penalty β for mismatches. The maximal sum finding process is performed in the Maximal Sum Finding Module. After verification, the verified HSPs will be stored in the local HSPs buffer, then further passed to the CPU to perform 'Gapped Extension'.

3.3 Data Flow

Word Matching: The data flow of 'Word matching' is indicated by the blue arrow in Fig. 1(c). The hashed query sequence will be passed to the hashed query buffers in each BLASTN Unit from the CPU through the bus. Then, the hashed word with length w will be transferred to different CAMs through the H-tree to perform 'Word Matching'.

RADAR uses the input word with length w as a comparison key and shifts it through the DNA sequences stored in each CAM to perform searching. If the final comparison result of an input word is a match, this word will be recognized as a w-mer and will be stored in w-mer buffers for 'Ungapped Extension'.

Ungapped Extension: The data flow of 'Ungapped Extension' is indicated by the green arrows in Fig. 1(c). The original query sequence will be passed to the query buffers in each BLASTN Unit and be the input to the BLASTN Mat.

The information of w-mers will be transferred to the controller (CtrL) to determine where 'Ungapped Extension' stage should start. Then, characters within a fixed-sized window with size L_w, centered on the target w-mer, will be transferred from the query buffer to CAMs to perform pair-wise character comparison. These pair-wise match/mismatch information will be passed to the HSPs Calculator, as shown in Fig. 1(f).

The HSPs Calculator will find the max sum of HSPs candidate, i.e. character pairs. If the maximal sum passes a preset threshold, this HSPs candidate will become a valid HSP and be passed to the host to perform 'Gapped Extension'.

Reduction in Data Movement: Unlike previous solutions [3, 12, 17], RADAR moves the query sequences into CAMs instead of moving the entire DNA database into the computing components to perform comparisons. The DNA database remains in CAMs and computation is performed locally. In this way, the amount of data movement is reduced from tens of gigabytes to a few bytes for a query task, leading to speedup and energy saving.

3.4 Discussion

3.4.1 Design Configuration. RADAR provides row-level, CAM-level, and unit-level parallelism. Adjustments can be made between these three levels of parallelism, enabling either fine-grain control or coarse-grain control, by changing the size of 3D ReCAMs, number of CAMs per BLASTN Mat, and number of BLASTN Units.

3.4.2 Endurance Problem. RADAR will not confront the lifetime issue of ReRAM [23] since RADAR only leverages read operation to perform DNA alignment. There are no write operation in RADAR other than initially writing the DNA database into the ReCAMs.

3.4.3 Scalability. Due to the high density of 3D ReCAM, RADAR is able to store the entire DNA database in a single chip, as our experiments demonstrate. When the target database happens to be too large to fit in a single chip, scaling-out is an option. We can use distributed RADAR to deal with extremely huge databases. Different RADAR nodes perform BLASTN locally and the local results will be merged to generate the final result after all nodes complete their own computation.

3.4.4 Further Extension. Although RADAR currently utilizes 3D ReCAM, RADAR can also be implemented with other CAMs, including other NVM-based CAMs [16] and traditional SRAM-based CAM by making minor modifications to the RADAR architecture.

Because searching is a commonly used operation in various applications, we can further extend RADAR to make it suitable for accelerating other applications such as text searching and BLASTP [8], the protein version of BLAST.

To summarize, RADAR can be extended to various applications in addition to different devices.

4 DATA MAPPING SCHEME

Since there are huge gaps between the lengths of long and short DNA sequences, it is challenging to map DNA sequences into RADAR due to the following reasons:

- Managing sequences in an aligned manner in 3D ReCAM and leaving the remaining bits unused will cost significant storage overhead.
- Segmenting sequences into multiple rows and CAMs will deteriorate parallelism due to data dependencies and communication.

In this section, we introduce the data mapping scheme in RADAR to address these challenges.

4.1 Dense Data Mapping Scheme

In order to address the first issue, we design a dense data mapping scheme to avoid wasting storage. This dense data mapping scheme sequentially allocates the DNA sequences one after another in the 3D ReCAMs as shown in Fig. 1(e) and (g). Different DNA sequences are represented using different colors. Since there are only 4 nucleic acids, i.e. 'A/T/C/G', and CAM cannot distinguish words with different numbers (> 0) of 1, if there is 1 in a word, the corresponding search result will be an unconditional match. For example, '0001', '0011', '0111', and '1111' are indistinguishable. RADAR uses 4 single-bit ReRAM cells to encode 1 nucleic acid.

Unfortunately, solely using dense data mapping scheme will damage the comparison parallelism since the DNA sequences can cross multiple rows, multiple CAMs, and even multiple BLASTN Units. Boundary crossing will introduce data dependencies and communication in the 'Word Matching' stage when the comparison key slides to the end of each row. This is the second issue we mentioned above.

4.2 Tail Bits Duplication (TBD)

To address the second issue in data mapping and improve parallelism, we propose the TBD technique, shown in Fig. 1(g). The TBD technique duplicates the tail bits in each row. If the DNA sequences cross multiple rows or multiple CAMs, the head bits in the following row or following CAM are set to the duplicated bits. The number of the tail bits will be at most $w - 1$.

TBD eliminates data dependencies and communication, enabling full row-level, CAM-level and, unit-level parallelism. Assume that the word size is w and that each 3D ReCAM has M rows, N columns, and L layers. The resulting storage overhead, i.e. Redundancy Ratio, has an upper bound given by the equation:

$$Redundancy\_Ratio \leq \frac{w-1}{N*(L/4)} \times 100\% \qquad (1)$$

Considering the fact that the typical value for w is 11 for BLASTN and $N*(L/4)$ is on the order of a thousand, the $Redundancy\_Ratio$ is typically far below 1%.

5 EXPERIMENTAL RESULTS

The experimental setup, results, and analysis are presented in this section.

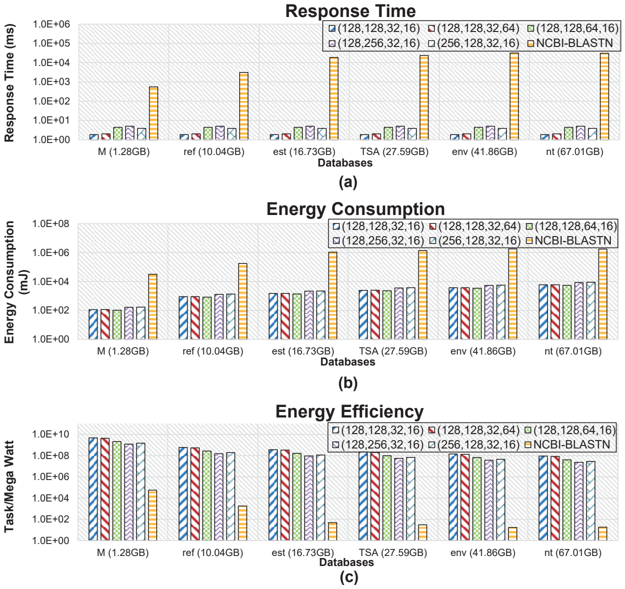


Figure 2: Performance of NCBI BLASTN and RADAR. (a) Query response time. (b) Energy consumption. (c) Energy efficiency.

5.1 Experimental Setup

The query response time and the energy consumption of BLASTN are related to the size of the DNA database. The representative length of a query sequence is 100 [6]. Therefore, we evaluate 6 different-sized databases using query sequences of length 100. For each of the 5 configurations of RADAR, experiments are performed 10 times for 6 databases with 10 query sequences.

To validate the effectiveness of RADAR, we use three evaluation metrics. These metrics are: the response time of a single query, the energy consumption of a single query, and the energy efficiency (Task/MegaWatt) of RADAR.

The baseline in our experiments is NCBI BLASTN 2.6.0+, running in a server with an Intel Xeon E5-2680 v3 CPU.

We build a simulator in C++ to simulate the performance of RADAR. The parameters of 3D ReCAM are extracted from NVSim [5]. ASIC circuits are realized in Verilog and synthesized in Design Complier [21]. We evaluate 5 configurations of RADAR with different numbers of CAM Rows, CAM Columns, CAM Layers, and CAMs per BLASTN Mat. As shown in Fig. 2 and Fig. 3, these configurations are represented in the following format: (*Row Number*,*Column Number*,*Layer Number*, *CAMs per Mat*).

5.2 Analysis of Experimental Results

Speedup: In Fig. 2(a), the x-axis represents the 5 RADAR configurations and CPU with different databases. The y-axis shows the response time for each experimental setup. All RADAR configurations have extremely good performance on accelerating BLASTN.

Fig. 2(a) shows that the query response time of CPU grows dramatically with larger databases, while that of the RADAR configurations remains relatively stable because of the large parallelism in RADAR. This means RADAR is extremely suitable for handling the ever-growing DNA databases. As a result, speedup of the first RADAR configuration over CPU grows from 299x up to 16,900x when the size of the database grows from 1.28GB to 67.01GB.

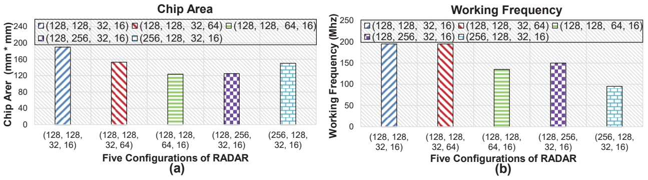


Figure 3: Parameters of 5 RADAR configurations. (a) Chip area. (b) Working frequencies.

Table 1: Speedup of Different Accelerators for BLASTN Baseline: NCBI BLASTN

Design	RADAR	Paracel BLAST [3]	Mercury BLAST [4]	TUC BLAST [20]	CUDA-BLAST [15]
Hardware	3D ReCAM	32-CPU Cluster	FPGA	FPGA	GPU
Speedup	5114x	96x	11x	37x	2.7x

Energy Consumption: The energy consumption (y-axis) of CPU and RADAR configurations is shown in Fig. 2(b). Experimental results demonstrate that all configurations of RADAR are very energy efficient compared to the CPU. The energy efficiency of RADAR comes from its reduction in data movement.

Energy consumption of both CPU and RADAR configurations grow with the size of the DNA database (x-axis). Compared to the CPU, the first RADAR configuration shows that energy is reduced by 199x to 724x.

Energy Efficiency: Fig. 2(c) indicates that the energy efficiency (y-axis) of both CPU and RADARs will decrease as the size of the DNA database (x-axis) increases, because larger databases lead to longer query response time. However, the energy efficiency of the CPU decreases much faster than the energy efficiency of RADAR configurations. The significant speedup and energy reduction of RADAR lead to its huge improvement in energy efficiency, which ranges from 85,200x up to 8,260,000x for the first configuration.

Trade-Offs: Trade-offs between response time, energy consumption, energy efficiency and chip area are viable by adjusting configuration parameters in RADAR, as shown in Fig. 2 and Fig. 3.

Configurations with more CAMs per BLASTN Unit, layers, columns, and rows have the advantage of reducing the chip area, but decrease the working frequency of RADAR. Larger CAMs and larger BLASTN Mats result in longer read time within the CAM, increasing the response time and the energy consumption, which leads to reduced energy efficiency.

Comparison to Other Accelerators: As shown in Table. 1, compared to the Multi-Core/FPGA/GPU based accelerators of BLASTN, RADAR has the best performance and outperforms them between 53x and 1896x in processing speed.

5.3 Performance of Tail Bits Duplication

TBD eliminates data dependencies and communication, improving the performance of comparison. To evaluate the effectiveness

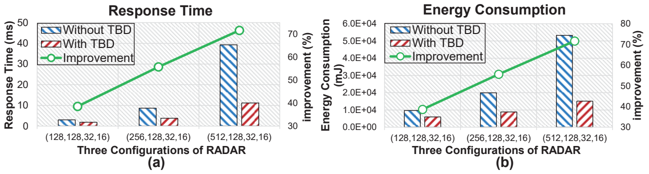


Figure 4: Performance of TBD. (a) Response time. (b) Energy consumption.

of TBD, we conduct experiments on RADAR with and without TBD. Fig. 4 shows that reduction in both response time and energy consumption ranging from around 40% to more than 70% can be achieved with TBD. The performance gain of TBD grows with the number of rows per CAM, because additional rows introduce larger inter-row data dependencies. These dependencies harms the performance of RADAR greatly, while TBD can totally eliminate data dependencies and communication.

5.4 Energy and area breakdown

We breakdown the energy consumption and area for the first configuration of RADAR, as shown in Fig. 5. Results show that 99.95% of the energy is consumed by CAMs and that the leakage energy consumption of CAMs is only 5.18%. As for the area, 74.13% of the area is occupied by CAMs. The buffers and HSPs Calculators occupy 21.75% and 4.12% of the area, respectively. This shows that RADAR is a memory-centric accelerator with low leakage power and RADAR only introduces very small extra hardware overhead on 3D ReCAM.

6 CONCLUSION

To address the problem of frequent data movement in BLASTN, we propose RADAR, an energy efficient PIM architecture to accelerate BLASTN. We also propose a dense data mapping method for efficiently handling DNA sequences and a TBD technique to enable fully parallel computation.

Experimental results show that the proposed TBD technique improves the response time and reduces the energy consumption of RADAR from approximately 40% to more than 70%, with insignificant storage overhead that is less than 1%. Regarding the bottlenecks in BLASTN, 5114x speedup and 386x energy reduction can be achieved with RADAR, compared to a single CPU. Compared with Multi-Core/FPGA/GPU based accelerators, RADAR outperforms them between 53x and 1896x in processing speed.

REFERENCES

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. 1990. Basic local alignment search tool. *Journal of molecular biology* 215, 3 (1990), 403–410.

[2] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research* 25, 17 (1997), 3389–3402.

[3] Paracel BLAST. 2015. Paracel BLAST. http://www.paracel.com/index.html. (2015).

[4] Jeremy D Buhler, Joseph M Lancaster, Arpith C Jacob, Roger D Chamberlain, et al. 2007. Mercury BLASTN: Faster DNA sequence comparison using a streaming hardware architecture. *Proc. of Reconfigurable Systems Summer Institute* (2007).

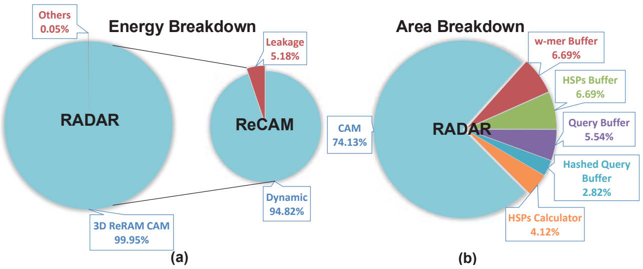


Figure 5: (a) Energy breakdown of RADAR. (b) Area breakdown of RADAR.

[5] Xiangyu Dong, Cong Xu, Norm Jouppi, and Yuan Xie. 2014. NVSim: A circuit-level performance, energy, and area model for emerging non-volatile memory. In *Emerging Memory Technologies*. Springer, 15–50.

[6] Edward B Fernandez, Walid A Najjar, Stefano Lonardi, and Jason Villarreal. [n. d.]. Multithreaded FPGA acceleration of DNA sequence mapping. In *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*. 1–6.

[7] National Center for Biological Information. 2017. GenBank and WGS Statistics. https://www.ncbi.nlm.nih.gov/genbank/statistics/. (2017).

[8] Arpith Jacob, Joseph Lancaster, Jeremy Buhler, Brandon Harris, and Roger D Chamberlain. 2008. Mercury BLASTP: Accelerating protein sequence alignment. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* (2008).

[9] Manjari Jha, Raunaq Malhotra, and Raj Acharya. 2016. A generalized lattice based probabilistic approach for metagenomic clustering. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2016).

[10] Jung-Yup Kang, Sandeep Gupta, and J-L Gaudiot. 2004. Accelerating the kernels of BLAST with an efficient PIM (processor-in-memory) architecture. In *Computational Systems Bioinformatics Conference, 2004*. IEEE, 552–553.

[11] Roman Kaplan, Leonid Yavits, Ran Ginosar, and Uri Weiser. 2017. A resistive CAM Processing-in-Storage architecture for DNA sequence alignment. *arXiv preprint arXiv:1701.04723* (2017).

[12] Joseph Lancaster, Jeremy Buhler, and Roger D Chamberlain. 2009. Acceleration of ungapped extension in Mercury BLAST. *Microprocessors and microsystems* 33, 4 (2009), 281–289.

[13] Shuangchen Li, Ping Chi, Jishen Zhao, Kwang-Ting Cheng, and Yuan Xie. 2015. Leveraging nonvolatility for architecture design with emerging NVM. In *Non-Volatile Memory System and Applications Symposium (NVMSA), 2015 IEEE*. IEEE, 1–5.

[14] Shuangchen Li, Liu Liu, Peng Gu, Cong Xu, and Yuan Xie. 2016. Nvsim-cam: a circuit-level simulator for emerging nonvolatile memory based content-addressable memory. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 2.

[15] Cheng Ling and Khaled Benkrid. 2010. Design and implementation of a CUDA-compatible GPU-based core for gapped BLAST algorithm. *Procedia Computer Science* 1, 1 (2010), 495–504.

[16] Shoun Matsunaga, Sadahiko Miura, Hiroaki Honjou, Keizo Kinoshita, Shoji Ikeda, Tetsuo Endoh, Hideo Ohno, and Takahiro Hanyu. 2012. A 3.14 um 2 4T-2MTJ-cell fully parallel TCAM based on nonvolatile logic-in-memory architecture. In *Symposium on VLSI Circuits (VLSIC)*. 44–45.

[17] Zemin Ning, Anthony J Cox, and James C Mullikin. 2001. SSAHA: a fast search method for large DNA databases. *Genome research* 11, 10 (2001), 1725–1729.

[18] Jay Shendure and Hanlee Ji. 2008. Next-generation DNA sequencing. *Nature biotechnology* 26, 10 (2008), 1135–1145.

[19] Temple F Smith and Michael S Waterman. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

[20] Euripides Sotiriades, Christos Kozanitis, and Apostolos Dollas. 2006. Some initial results on hardware BLAST acceleration with a reconfigurable architecture. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 8–pp.

[21] Synopsys. 2017. Design Complier (DC). https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test.html. (2017).

[22] Cong Xu, Pai-Yu Chen, Dimin Niu, Yang Zheng, Shimeng Yu, and Yuan Xie. 2014. Architecting 3D vertical resistive memory for next-generation storage systems. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. 55–62.

[23] Leonid Yavits, Shahar Kvatinsky, Amir Morad, and Ran Ginosar. 2015. Resistive associative processor. *IEEE Computer Architecture Letters* 14, 2 (2015), 148–151.