Demeter

A Fast and Energy-Efficient Food Profiler Using Hyperdimensional Computing in Memory

Shahroodi, Taha; Zahedi, Mahdi; Firtina, Can; Alser, Mohammed ; Wong, Stephan; Mutlu, Onur; Hamdioui, Said

DOI
10.1109/ACCESS.2022.3195878

Publication date
2022

Document Version
Final published version

Published in
IEEE Access

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 June 2022, accepted 19 July 2022, date of publication 1 August 2022, date of current version 10 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3195878

RESEARCH ARTICLE

Demeter: A Fast and Energy-Efficient Food Profiler Using Hyperdimensional Computing in Memory

TAHA SHAHROODI[1], MAHDI ZAHEDI[1], CAN FIRTINA[2], MOHAMMED ALSER[2], STEPHAN WONG[1], (Senior Member, IEEE), ONUR MUTLU[2], (Fellow, IEEE), AND SAID HAMDIOUI[1], (Senior Member, IEEE)

[1]Q&CE Department, EEMCS Faculty, Delft University of Technology (TU Delft), 2628 CD Delft, The Netherlands
[2]SAFARI Research Group, D-ITET, ETH Zürich, 8092 Zürich, Switzerland

Corresponding authors: Taha Shahroodi (t.shahroodi@tudelft.nl) and Mohammed Alser (mealser@gmail.com)

This work was supported in part by the Q&CE Department, Delft University of Technology (TU Delft); and in part by the SAFARI Research Group's industrial partners, especially AMSL, Facebook, Google, Huawei, Intel, Microsoft, VMware, Xilinx, the ETH Future Computing Laboratory, and Semiconductor Research Corporation.

ABSTRACT Food profiling is an essential step in any food monitoring system needed to prevent health risks and potential frauds in the food industry. Significant improvements in sequencing technologies are pushing food profiling to become the main computational bottleneck. State-of-the-art profilers are unfortunately too costly for food profiling. Our goal is to design a food profiler that solves the main limitations of existing profilers, namely (1) working on massive data structures and (2) incurring considerable data movement, for a real-time monitoring system. To this end, we propose Demeter, the first platform-independent framework for food profiling. Demeter overcomes the first limitation through the use of hyperdimensional computing (HDC) and efficiently performs the accurate few-species classification required in food profiling. We overcome the second limitation by the use of an in-memory hardware accelerator for Demeter (named Acc-Demeter) based on memristor devices. Acc-Demeter actualizes several domain-specific optimizations and exploits the inherent characteristics of memristors to improve the overall performance and energy consumption of Acc-Demeter. We compare Demeter's accuracy with other industrial food profilers using detailed software modeling. We synthesize Acc-Demeter's required hardware using UMC's 65nm library by considering an accurate PCM model based on silicon-based prototypes. Our evaluations demonstrate that Acc-Demeter achieves a (1) throughput improvement of 192× and 724× and (2) memory reduction of 36× and 33× compared to Kraken2 and MetaCache (2 state-of-the-art profilers), respectively, on typical food-related databases. Demeter maintains an acceptable profiling accuracy (within 2% of existing tools) and incurs a very low area overhead.

INDEX TERMS Food profiling, emerging memories, in memory processing, analog computing.

I. INTRODUCTION

The urgent need for a real-time, efficient, and accurate food monitoring system is apparent when one considers the economic impacts and health risk issues due to human errors and/or intentional fraud regarding everyday food. For example, a worldwide annual loss of $10 to $24 billion of dollars is estimated only for the frauds happening in the fish industry [1]. The Halal meat scandal [2] and the black fish scandal [3] are just a few other preventable examples that could have been quickly prevented if we had accurately and efficiently monitored all the food productions in real-time.

Food profiling is the first step and the only computationally expensive task in a food monitoring system. The food profiling task entails the functionality of determining the existing species in a food sample and their relative abundance [4], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

Today's food profilers work with sequenced data as we can capture a more accurate profile using the sequences of a food sample. The rapid drop in the cost of DNA sequencing in the past decades and the expectation for a continual trend [6], [7][1] is expected to lead the way for profiling to become the main bottleneck of this pipeline.

Currently, the industry utilizes state-of-the-art (SOTA) taxonomic profilers from metagenomic studies for food profiling due to the similarity of problem statements in food profiling and metagenomics profiling. However, as alluded, such profilers are developed as the first step of metagenomic studies [9]–[11]: a new, yet different, line of research that allows us to study many species that are taken directly from their environment altogether, as opposed to studying them individually. Unfortunately, these profilers are overkill for simply profiling a given food sample and, therefore, costly since those taxonomic profilers have been designed for different, more complex goals such as (1) capturing complex operations between organisms or (2) finding insights on species that cannot be clonally cultured in labs. Such profilers are also designed for working on larger, more complex, and randomly mixed genome sequences and demand a significant amount of resources that simply impede real-time monitoring of all food samples after production, shipment, or distribution; the ultimate goal of a food monitoring system. Therefore, a new solution must be sought after specifically for food profiling that is cheaper, faster, more energy-efficient, and yet accurate.

In particular, we pinpoint two critical sources of inefficiency in SOTA profilers currently used for food monitoring, collectively called food profilers or profilers hereafter. First, all current (food) profilers work with significantly large working data structures, e.g., humongous hash tables or sorted lists, that require high-end servers with extensive storage and memory capabilities to be handled. This fundamentally limits performance scaling in par with that in sequencing technologies. Second, current profiling techniques incur a significant number of random accesses to large working datasets, and as a result, unnecessary data movement between their storage and memory plus their memory and compute units which cannot be otherwise done where the data resides due to (1) the size of the final data structures and (2) the required operations for tasks in hand. This directly translates to massive energy consumption and latency. For example, as shown in our evaluations Sections IV, VII, a widely used SOTA profiler takes $\sim$1 minute to profile one high-coverage sequenced food sample. However, it requires a super machine or cluster with at least 300 GB of memory and proportionally scaled-up compute power. These costs add up to an unbearable required time and equipment for real-time monitoring of all existing and producing food samples. Therefore, a healthy economy regarding the food industry cannot keep using these profilers and demands cheaper, faster, more energy-efficient, and accurate food profilers for the years to come.

Our goal in this work is to solve both limitations of previous profilers, namely (1) reliance on high-end servers and scaling problem due to required massive data structures and (2) incurring unnecessary data movement. **To this end,** we propose Demeter, an end-to-end, hardware/software co-designed food profiling framework that efficiently profiles species of a food sample. **The key idea** of Demeter is to reduce the food profiling problem to a multi-object (multi-species) classification problem using hyperdimensional (HD) computing (HDC) followed by an abundance estimation step. Demeter is a platform-independent framework and produces accurate results on any hardware platform such as central processing unit) CPU, graphics processing unit (GPU), or application-specific integrated circuit (ASIC).

Our experiments show that although the accuracy of Demeter is comparable with existing SOTA profilers, typical processing units (CPUs) are not exploiting the full parallelism offered by our HDC-based approach, prohibiting those platforms from outperforming SOTA profilers. Moreover, we find two more optimization opportunities that can be achieved with a wisely-chosen platform: (1) eliminating the cost of existing shift operations and (2) mitigating the significant amount of data movement involved in our HDC-based solution. Therefore, we propose an in-memory hardware accelerator for Demeter, Acc-Demeter, to mitigate the costs mentioned above and simultaneously solve the second problem of profilers as well. Acc-Demeter achieves these by (1) the physical attributes of nanoscale memristive-based devices,[2] (2) Processing-In-Memory (PIM), where the data resides, and (3) zero-overhead shift operation in hardware. It is worth noting that, with the advent of portable sequencing machines, a move from cloud computing with sophisticated infrastructure towards an in-build profiler (or other genomics-related kernels) inside the sequencer is finally in the foreseeable future.

Our paper makes the following main contributions:

- To our knowledge, Demeter is the first framework that enables food profiling via HDC. Demeter provides a five-step approach to determine the relative abundance of a set of the food read sequences at species-level. We design Demeter to (1) address the key problems of food profiling rather than accelerating regular metagenomic profilers and (2) be platform-independent (Section III).
- We propose a PIM-enabled hardware accelerator for Demeter using memristor devices (Acc-Demeter) to extract Demeter's full potentials and solve the data movement problem in Demeter and previous profilers. We propose several optimization techniques for Acc-Demeter based on domain-specific knowledge of

[1]Researchers expect that soon different analyses on the sequenced data become the cost and performance bottleneck rather than sequencing itself, as is currently the case for read mapping vs. DNA sequencing [8].

[2]We choose Phase Change Memory (PCM) devices as members of memristor families due to our accessibility to accurate measurements and models. However, in principle, our proposed techniques can be applied to any memristor-based memory technology, such as ReRAM or STT-MRAM.

food profiling and our background on PCM cells characteristics and HDC operations. To our knowledge, Acc-Demeter is the first (in-memory) hardware accelerator for a food profiler (Section V).

- We rigorously compared Demeter and Acc-Demeter to four SOTA food profilers. We show that Demeter provides an accuracy level comparable with previous food profilers and within the accepted level of food monitoring systems. The default setting of Acc-Demeter enables a (1) throughput improvement of ∼192× and 724× and (2) reduction in the required memory of ∼36× and 33× compared to Kraken2 [12] and MetaCache [4], respectively, when querying on a typical food-related reference genome database, i.e., AFS20 [4]. Our design requires only ∼8.9 mm^2 die area and can process ∼9.45 Mbp per joule for our largest food-related database AFS31 [4] (Section VII).

II. BACKGROUND AND MOTIVATION

This section discusses the necessary background and introduction to (1) the current taxonomic profilers and their shortcomings when used for food profiling, (2) HDC, and (3) the PIM paradigm. We devote the materials mainly to those closely related to or used by Demeter. For more detailed background information, we refer the reader to comprehensive reviews on these topics [13]–[19].

A. METAGENOMIC PROFILERS

Constantly increasing the performance of sequencing technologies and the fast drop in the cost of DNA sequencing [6], [7] catalyzed the metagenomic studies [9]–[11]. These studies enable us to capture the big picture of the environment without isolating or cultivating individual organisms. For this purpose, one needs to perform taxonomic profiling: determining the relative abundances of species in a sample directly taken from the environment. Due to the high cost associated with alignment and assembly for large reference datasets, till this date, we still prefer heuristics statistical-based profilers to assembly- or alignment-based ones. However, even these profilers are not yet cheap or economic and prevent large-scale, real-time studying. Their cost is mainly related to the required memory for profilers' data structure and algorithms. Such large data structures or sophisticated algorithms force us to use high-end servers and are needed to fulfill complex goals of subsequent metagenomic analysis, namely capturing complex operations between organisms and discovering insights on species that cannot be clonally cultured in labs. This high cost of profiling in metagenomics profiler prevents us from efficiently profiling food samples in real-time, the end goal of a food monitoring system.

B. PROBLEMS OF FOOD PROFILERS

We use VTune [20] and profile three SOTA profilers that are currently used for food samples as well, namely Kraken2, CLARK, and MetaCache, using their default datasets and parameters on the original platforms for which they have been designed. We make two main observations, which follow a similar trend reported in previous studies in genomics as well [12], [21], [22].

Observation 1. All these profilers induce large memory requirements for their data structures. For example, Kraken2 requires a minimum of 300 GB memory for its reference data structure. Even for smaller and less complex reference data bases such as those in food industry, Kraken2 still requires more than 50 GB of memory (Section IV-D).

Observation 2. All profilers induce high miss rates in L2 and L3 (∼68 to 90%). The nature of their underlying algorithms causes this inefficiency because they always query a small fraction of keys in a large hash table and/or sorted list, leading to random memory access patterns. In other words, the arithmetic intensity of the profilers is too small to the extent that even increasing the number of threads does not help resolve the CPU stall cycles caused by memory accesses required by these misses.

Overall, current food profilers' large working data structure and their low arithmetic intensity lead to high storage cost, low performance, and high energy consumption. It also demands high-end servers. This motivates designs (such as Demeter and Acc-Demeter) that provide reduced working data structures, eliminate unnecessary data movements, and can liberate us from dependency on the clusters.

C. HYPERDIMENSIONAL COMPUTING

Hyperdimensional computing (HDC) [23], [24] is a brain-inspired computing paradigm that has been demonstrated to be effective in reference-based learning domains, such as text classification [25]–[27], gesture recognition [28], and latent semantic analysis [29]. Elements of HDC are presented using high-dimensional vectors, hereafter called HD vectors. HD vectors can be composed of real [30]–[32], binary [23], [33], bipolar [28], [34], or complex numbers [35]. Previous works show that binary representations of HD vectors are more practical and efficient for classification problems or one-shot reinforcement learning. This representation is also more hardware-friendly. Therefore, we proceed with binary HD representations. HD vectors also come with other powerful features such as robustness to random errors, holistic representation, and randomness. We refer the enthusiastic readers to previous works for more details on these features [17], [23].

Like other reference-based classifiers, an HDC-based system also takes two steps: (1) training and (2) classification. An encoding mechanism is used in both steps. One famous example is the N-gram encoding mechanism that follows a two-step approach for encoding a string of size L to an HD vector of size D. **Step 1:** It combines N consecutive alphabets of the string and builds an HD vector that is orthogonal to them all and can preserve their relative order. This operation is called binding and is represented in Equation 1, where $\rho^i(X)$ represents the i^{th} permutation of vector X and B_i are once randomly-generated representative HD vectors (also referred to as atomic or basis HD vectors) for the i^{th} character of the

string C_i. The string is a DNA sequence in Demeter.

$$N-gram(C_1, C_2, \ldots, C_N) = B_1 \oplus \rho(B_2) \oplus \cdots \oplus \rho^{N-1}(B_N) \quad (1)$$

Step 2: The encoder performs an element-wise addition between all HD vectors corresponding to consecutive N-grams, called bundling, to present the entire input sequence. To binarize the final HD vector, the encoder applies a majority function over each position. This final vector is stored in associate memory (AM) and is called a prototype HD vector if the input was a reference genome. Otherwise, it is called query HD vector and will use it for classification.

The most common approach for classifying whether the sequence query belongs to any of the classes in AM after using N-gram encoding mechanism is to measure the hamming distance between the query HD vector (Q) and each of the prototype HD vectors (Ps) and decide based on a fixed distance or threshold (T). This can be easily performed with an XNOR of Q and each P followed by a pop-count[3] and thresholding operation, as shown in Equation 2.

$$Classification(i) = \begin{cases} 1, & \text{if } \sum_{j=1}^{D} Q(j) \bar{\oplus} P_i(j) \geq T \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

D. COMPUTING INSIDE/NEAR MEMORY

For decades, the processing units have been developed at a faster rate than memory units, causing memory units to become a bottleneck, especially in data-intensive workloads. The Processing-In-Memory (PIM) (interchangeably also referred to as Computation-In-Memory (CIM)) is a promising paradigm that aims to alleviate the data movement bottleneck. In essence, PIM advocates for avoiding unnecessary data movement and redesigning systems such that they are no longer processor-centric. Previous works show the potential of various memory technologies for implementing PIM-based architectures [36]–[39]. Resistive memories or memristive devices, such as ReRAM and PCM [37], [40], [41], have recently been introduced as a suitable candidate for both storage and computation units that can efficiently perform vector-matrix multiplication [42] and bulk bit-wise logical operations [43], [44] since they can follow Kirchhoff's law inherently. They also enjoy non-volatility, high-density, and near-zero standby power. Due to the inherent high parallelism, simplicity of the required operations, and intrinsic robustness and error tolerance of an HDC-based system, such a system fits well with the PIM paradigm. A few recent works propose application-specific hardware accelerators based on memristive devices [41], [45]–[47] for such HDC-based systems. We describe the differences between our implementation and closest previous proposals in Sections V and IX.

[3]Pop-count (population count) of a vector or specific value is the process of finding the number of set bits (1s) in that value.

III. DEMETER

Demeter is a configurable framework for food profiling and is based on three main insights: (1) current food profilers, whereas accurate, are neither memory- nor energy-efficient, (2) the primary sources of high cost and inefficiency in current food profilers is their large reference data structures and working sets, and (3) one can profile food samples quickly and accurately using HDC. Fig. 1 provides an overview of the five key steps in Demeter: ❶ defining an HD space, ❷ building an HD reference database (HD-RefDB), ❸ converting sample reads into HD space, ❹ determining the possible species assignment per sample read, and ❺ performing abundance estimation. We describe each step in more detail next.

A. STEP 1: DEFINE THE HD SPACE

As the first step (❶ in Fig. 1), Demeter defines an HD space for all subsequent operations and steps. This is a crucial step as it determines the operations in the remaining steps. Unfortunately, many previous HDC-based proposals did not support the user's input for determining the HD space and designed their space statically. Hence, such designs are more limited.

Demeter defines the HD space in 4 stages. **Stage 1:** Demeter fixes two hyperparameters: (1) The dimension of the HD space; i.e., the dimensionality of the HD vectors (element representations), and (2) The sparsity of each element (HD vector). **Stage 2:** Demeter generates a few atomic HD vectors and stores them in memory (commonly called Item Memory (IM)). These vectors can be (1) the HD vectors that represent our genome alphabets or (2) the one-time randomly generated HD vectors that some encoding mechanisms use, for example, to introduce the concept of order between alphabets of one input. **Stage 3:** Demeter decides on the encoding mechanism to build the space with. This very encoding mechanism will be used throughout Steps ❷ and ❸ of Demeter. **Stage 4:** Demeter fixes the similarity metric and any other associated parameters (such as thresholds) based on the user's input or a common choice considering previous stages. Demeter stores a default value for each stage in a configuration file. Once the user summons Demeter, Demeter quickly checks if a configuration file matches with the user's requested HD space or not. Demeter only runs this step if such file does not exist or the user asked for a change.

B. STEP 2: BUILD DEMETER's REFERENCE DATA STRUCTURE

Demeter takes two sets of inputs in step ❷: (1) HD space parameters defined in Step ❶ and (2) a reference genome database. Subsequently, Demeter builds a new reference database in its HD space out of all the considered reference genomes. This new database, called HD-RefDB, consists of one (or few) prototype HD vector(s) from any given reference genome in the original reference database and is stored in AM. HD-RefDB can be as varied as the number of combinations of possible hyperparameters, atomic

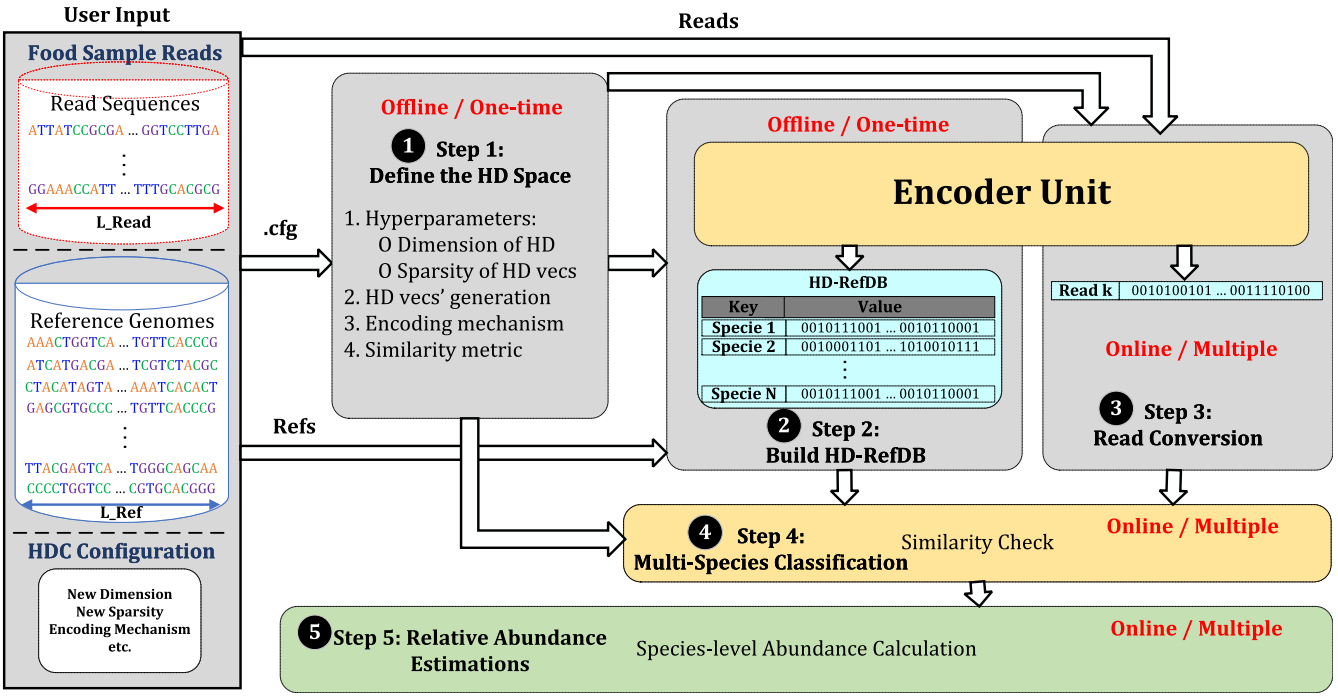


FIGURE 1. Overview of demeter framework.

vectors, and encoding mechanisms in Step ❶. This step aims to reduce the size of the working set for the classification task while avoiding accuracy drop. Since this step requires only simple arithmetics and is also highly parallelizable, it can still be accelerated on our proposed PIM-enabled accelerator (Section V).

C. STEP 3: DEMETER's READ CONVERSION

Demeter again takes two inputs in Step ❸: (1) HD space configuration and (2) read sequences of the food sample under study. Demeter translates each of these read sequences into one query HD vector. To prevent any extra storage cost and to pipeline computations of Step ❸ and Step ❹, Demeter forwards each query HD vector to the next step instead of storing them inside a memory unit while waiting for all of them to be constructed first.[4] Query HD vectors created in this step can require larger or smaller space than a read, depending on the initial length of the read sequences and the dimension of the HD space. Therefore, although Steps ❷ and ❸ share the encoding mechanism, their input and how Demeter treats the outcome are pretty different. Step ❸ neither introduces a new operation nor a procedure other than those that already exist from Step ❷. Therefore, it enjoys similar benefits as Step ❷, namely high parallelization and in-memory suitability. Demeter runs Step ❸ every time it profiles a new read of a food sample.

D. STEP 4: MULTI-SPECIES CLASSIFICATION PER READ

In this step, Demeter takes (1) the query HD vector (Step ❸), (2) HD-RefDB (Step ❷), and (3) similarity function and its corresponding parameters (Step ❶) as inputs. To determine the specie(s) that each read belongs to, Demeter performs a similarity check between the query HD vector and each of the prototype HD vectors in HD-RefDB. The similarity measure can vary depending on the vector representations and encoding approaches. Demeter allows various famous mechanisms for the similarity check, such as Hamming distance [17] and dot product [16]. Usually, this step can be implemented only with simple operations (Section II-C). It also enjoys high parallelization, similar to the previous steps. Although a similarity metric and its related parameters highly relate to (1) the encoding mechanism and (2) hyperparameters of the HD space, such as representations, sparsity, and the dimension of HD vectors, and therefore it makes sense not to let them change arbitrarily, Demeter supports changing them in Step ❹ as well, without needing to re-run Steps ❷ or ❸. This is because some studies show that different similarity metrics and thresholds may outperform others depending on your application and data for a fixed set of hyperparameters and encoding mechanisms. Therefore, if one decides to change their reference database, they may need to play with these to find the right match, and Demeter allows such investigations. Currently, Demeter provides a default option.

[4]This is the default behavior in Demeter. However, we also provide the option for the user to keep and store these query HD vectors (HD-ReadDB) in case one needs to analyze them further. If one uses this option, the stored query HD vectors create another database representing the reads in our food sample, called HD-ReadDB hereafter.

Demeter may find out that the query HD vector is close to one, multiple, or none of the prototype HD vectors in HD-RefDB. This variety in possible outputs differentiates Demeter from many previous HDC-based designs [17], [41], [48]–[50]. In such works, mostly due to the characteristics of applications under study, researchers always assume that (1) the query HD vector can only belong to one of the prototype HD vectors, and (2) the class of the query HD vector will exist in the AM. However, none of these assumptions hold for a food profiler. One read from the food sample can be related to one, multiple, or none of the reference genomes in the original reference genome database. This is because the read sequences are mostly short strings with a reasonably high probability of existence in longer reference genome sequences. It is also not uncommon that the query HD vector does not belong to any of the reference genomes in the initial reference genome database. This case can happen when, for example, (1) there is either an unknown species in the food sample, (2) one incorrectly excludes the corresponding reference genome in the initial reference genome database, or (3) an uncorrected sequencing error has happened. A food profiler should capture such cases. This difference between how many prototype HD vectors in HD-RefDB can be assigned to one query HD vector is a key difference that affects both the following abundance estimation step and final results. It also distinguishes this work further from previous HDC-based proposals for different applications. Step ❹ also enjoys high parallelization and in-memory suitability features similar to previous steps.

E. STEP 5: SPECIES LEVEL ABUNDANCE ESTIMATION

In Step ❺, Demeter performs a relative abundance estimation based on the results of Step ❹. This step is particularly needed for a food profiler in which one query HD vector can be similar to one or more classes/species. Demeter categorizes each query HD vector into (1) uniquely-mapped, (2) multi-mapped, and (3) unmapped, taking a two-step approach. In the first step, Demeter assigns the uniquely mapped query HD vectors to the species that they are similar to. In the second step, Demeter assigns the multi-mapped query HD vector to multi-species proportionally to the number of reads that have been uniquely aligned to in the first step divided by the length of species (reference genome). Demeter's Step ❺ can be extended to support different assignment policies for the multi-mapped reads. We leave investigating the effect of such methods for future work.

IV. DEMETER's EVALUATION

A. METHODOLOGY

We implement a multi-threaded highly-parallelized version of Demeter in C++ using SeqAn library [51], called C-Demeter. SeqAn library is an open-source optimized library for biological data. C-Demeter verifies the accuracy of Demeter. We also implement a GPU version of Demeter, G-Demeter. G-Demeter uses CUDA streams for parallelizing data copy operation between shared memory and global memory with other computations as much as possible. It implements the similarity check using parallel reduction technique introduced by Harris *et al.* [52] in the shared memory. All of our experiments run on a 128-core server with AMD EPYC 7742 CPUs [53] and with 500 GB of DDR4 DRAM. G-Demeter runs on an NVIDIA RTX 2080Ti GPU. Our sensitivity analysis shows that binary HD vectors of size 40,000, with dense distributed representation (DDR [17]) and N-gram-based encoding mechanism, strike a sweet spot in the tradeoff between accuracy, required memory, and performance. Therefore, unless otherwise stated, our evaluations use these setups.

1) ACCURACY METRICS

We capture the four fundamental rates from a (food) profiler when considering the presence and absence of each species in the output, i.e., True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) Rate. Based on these rates, Demeter reports two standard metrics of Precision and Recall [12], [54], [55] to assess the accuracy of our (food) profilers.

2) PERFORMANCE METRICS

Performance analysis consists of three experiments: (1) Build time, (2) Query time, and (3) Query throughput or speed. This separation has two main reasons. (1) Build time is normally a one-time job and does not affect the overall profiler's performance. Therefore, it is only fair to separate build time and query time. (2) Query time is simply the required time for profiling one single read. However, throughput is measured by million reads per minute ($\frac{MR}{m}$) and should be differentiated as it can get affected easily by other factors such as the size of the data structure, the classifier's parallelization capability, or the infrastructure's computation and storage/memory limitations (e.g., duplicating capabilities).

3) DATASETS

We have two sets of datasets. (1) Genome sequences used as a reference database. (2) Genomes sequences used as food samples and input queries. We consider AFS20 and AFS31 [4], [5] as our reference genome datasets. These datasets are two-commonly used datasets consist of 20 and 31 food-related reference genomes related to animals whose sizes vary from 12 MB to 14 GB. AFS31 is currently also the biggest reference dataset used in food profiling. Food sample reads or queries are from calibrator sausage samples from ENA project ID PRJEB34001 [56] and PRJNA271645 [57]. These reads are real short-read sequences from a mixture of food ingredients such as chicken, turkey, etc., sequenced on an Illumina HiSeq machine.

4) BASELINES

We compare Demeter against MetaCache [4] (the most accurate food profiler) Kraken2 [12], Kraken2+Bracken [58], and CLARK [59], the top 3 alignment-free and fastest

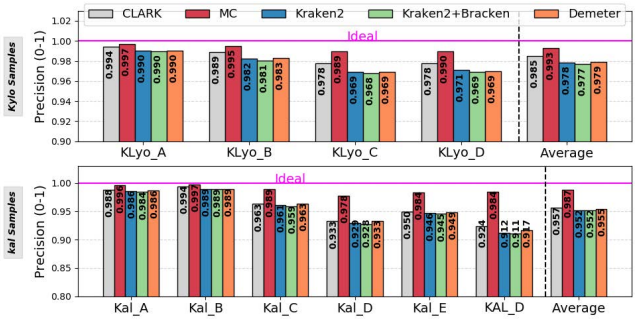


FIGURE 2. Precision rate for Kylo and Kal samples on AFS20.

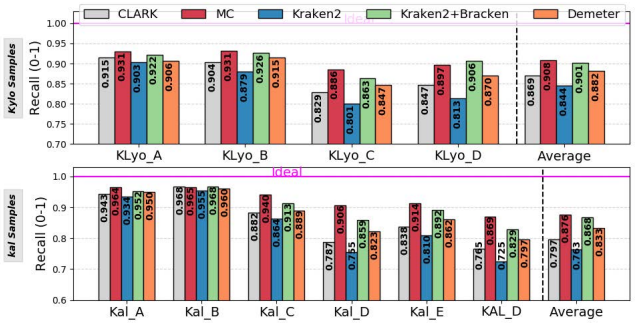


FIGURE 3. Recall rate for Kylo and Kal samples on AFS20.

metagenomic profilers that are also commonly used for food profiling.

B. DEMETER's ACCURACY ANALYSIS

Figures 2 and 3 present the results for the precision and recall of all evaluated food profilers on the species levels over AFS20 for Kylo and Kal food samples [56], [57], respectively. Note that the relative abundance of higher taxonomy levels is not of importance in food profiling. Additionally, those calculations highly depend on the propagation method from species level to those levels. Therefore, they have been excluded from this study.

We observe that Demeter stands very close to the most accurate profiler, MetaCache, and has only 1.4% and 2.6% less precision and recall, respectively, for KLyo samples. Moreover, Demeter achieves similar results on AFS31 and Kal samples. Note that accuracy is very much data-dependent, and indeed this accuracy drop is acceptable for a food profiler. The results of the latest comparison between current (metagenomics) profilers [60] show an Std error of the mean ranging from 0 to 5% regarding the precision and recall among various widely-used profilers on different datasets.

We conclude that Demeter is accurate and achieves high precision and recall for food samples. These results show that Demeter's HDC-based classification approach followed by our abundance estimation technique does not hurt the accuracy of the profiler compared to baselines.

C. DEMETER'S SOFTWARE PERFORMANCE ANALYSIS

Fig. 4-a and Fig. 5-a present the time that each profiler takes to query one (short) read from the query food sample and classify its specie(s) over AFS20 and AFS31, respectively.

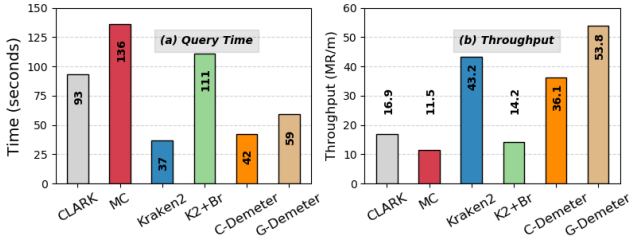


FIGURE 4. (a) Query time and (b) query throughput on AFS20.

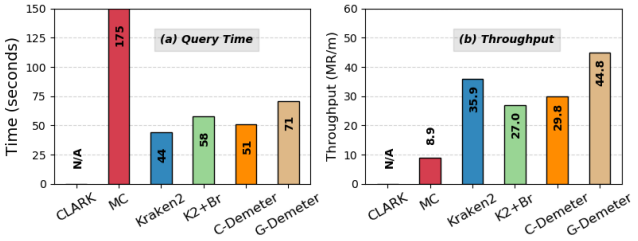


FIGURE 5. (a) Query time and (b) query throughput on AFS31.

We observe that both C-Demeter and G-Demeter, whereas accurate, require higher query time compared to Kraken2. The time breakdown, using Intel VTune [20] and cudaEvents, reveals that both implementations are memory bound, meaning there exists a significant percentage of under-utilized slots due to data access issues.

We believe that there are two main reasons behind this problem. First, the shift operation per processed character in the encoding mechanism of Demeter. Both of these implementations store the large HD vectors into multiple registers. Every shift operation translates to multiple copy operations among those registers, which can become costly in terms of time and energy consumption. This is why the query time is higher than expected. Second, not all prototype HD vectors fit in the caches. Therefore, the software versions take a few cycles to read prototype HD vectors in batches, compare them to query HD vector, save the results, and continue with the next batch. Note that these also put a limit on the expected throughput.

Fig. 4-b and Fig. 5-b present throughput of different profilers over AFS20 and AFS31. We make three observations. First, C-Demeter achieves a lower throughput compared to Kraken2. The reasons behind this are similar to what was discussed for its longer query time. Second, we observe that G-Demeter improves the throughput by up to 24% (depending on the reference dataset) and therefore can be used for food profiling in the industry in the near future. Third, we observe that simply increasing the number of working threads by moving from C-Demeter to G-Demeter does not improve the throughput considerably. We ask to use the commodity GPUs to perform the food profiling to cut the cost in the short term. In the long term, we propose extending Demeter to ASIC designs (such as those we present next) that solve the new sources of inefficiency we discussed above.

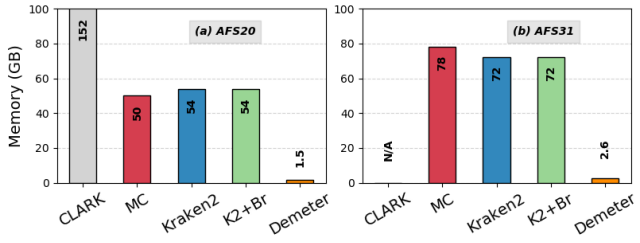


FIGURE 6. Required memory for (a) AFS20 and (b) AFS31.

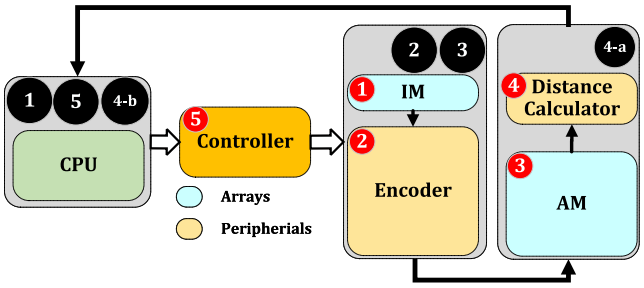


FIGURE 7. Overview of Demeter's in-memory accelerator.

However, our analysis also shows that even a massively-parallel implementation of Demeter, G-Demeter, does not fully utilize the parallelism offered by vector operations of HDC classification of Demeter, while also suffering from expensive copy-pasting among registers and its inability to perform the classification efficiently on a large vector in software.

D. DEMETER's MEMORY ANALYSIS

To show a key source of improvement in Demeter (and an enabler for Acc-Demeter), we compared the memory requirement of Demeter with the other food profilers. Fig. 6 presents the required memory for each profiler on AFS20 and AFS31.

We make the following two observations. First, Demeter requires ∼33× and 36× less memory than Kraken2 and MetaCache for AFS20 database and ∼27× and 30× less memory for them for AFS31 database, respectively. This makes Demeter the most efficient food profiler from a memory usage perspective. Second, the reduction in memory requirement for Demeter is to the extent that, for the first time, the data structure of the food profiler can fit into a standard size memory and does not require a colossal RAM managing further queries. This reduction is the main enabler behind Acc-Demeter. We conclude that Demeter is very memory efficient.

V. DEMETER's PIM-ENABLED ACCELERATOR

Demeter is positioned as a platform-independent food profiling framework that uses HDC. Demeter works with large HD vectors, is robust against errors, enjoys high parallelism, and exploits simple operations. These characteristics make Demeter a suitable candidate for hardware acceleration. However, the interest behind accelerating Demeter in a highly parallelizable and energy-efficient platform and specifically a PIM-enabled design goes beyond being simply its suitability and is a requisite for such a platform with two main motives.

Motivation 1: As discussed in Section IV-C, a software version of Demeter incurs a considerable cost on copy operations among registers holding intermediate HD vectors and classification. It also performs the classification poorly due to larger than cache HD-RefDB and low cache hit rate. These costs diminish all the benefits of Demeter that come from its small data structures and memory requirement. However, one can prevent this if Demeter is implemented in hardware as they can (1) realize the shift operation for free by only redirecting the output of each register to the next one and (2) perform the classification efficiently.

Motivation 2: A software-based implementation of Demeter still incurs a lot of unnecessary data movement for Steps ❷, ❸, and ❹. A hardware accelerator, especially a PIM-enabled one, can mitigate this problem greatly.

Therefore, we propose a PIM-enabled hardware accelerator for Demeter using PCM cells. One can accelerate Demeter using a PIM-enabled design on different memory technologies. We choose a memristor-enabled design for three main reasons. First, it is well-known that memristor-based memory technologies can perform vector-matrix multiplication [61]–[64] using Kirchhoff's law efficiently, making them suitable for our design. In this work, we manage to propose a hybrid row-major/column-major data mapping and intelligent data duplication scheme to perform encoding, classification, and profiling efficiently on PCM devices using this operation. Other technologies than memristors do not offer the same features for our hybrid data mapping.

Second, traditional technologies, such as non-memristor-based ones, are generally general-purpose and cost-driven. Moreover, their design does not allow even simple circuit modifications without high penalty on the area and cost. This makes them face a lot of pushback from the industry and unlikely to see future adoption. One of the advantages of memristors over them is their high density and scalability, and previous works show a wide range of accelerators using them.

Third, researchers already show the potential of accelerators based on emerging technologies for other ML-based algorithms [62], [65]. Also, multiple memory technologies already exist in current sequence machines. Therefore, it is not unreasonable to imagine one sort of these emerging memory technologies also be installed in these machines, especially for performing ML-based algorithms such as those for base-calling that are necessary for the sequencers [66].

In this work, we focus on PCM devices, as a member of the family of memristor devices, due to our accessibility to accurate device measurements and models for these devices and leave exploring other technologies for future research.

A. OVERVIEW OF DEMETER's ACCELERATOR

Fig. 7 shows an overview of the proposed PIM-enabled hardware accelerator for Demeter, Acc-Demeter. Acc-Demeter consists of 5 key elements: ❶ Item Memory (IM), ❷ Encoder,

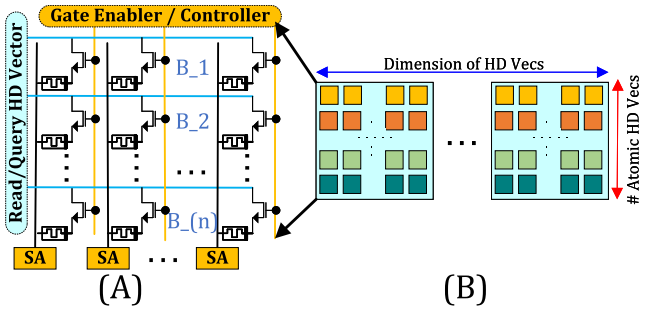


FIGURE 8. (A) IM design. (B) Data mapping and placement of atomic HD vectors in IM.

❸ Associate Memory (AM), ❹ Distance calculator, and ❺ Controller. IM and AM units are memory units, and we implement them as PCM arrays with their control circuitry. However, the encoder and distance calculator units are computing units implemented as the periphery. The controller is a simple FSM designed to harmonize the required steps of Demeter. The CPU initiates Demeter by gathering the user's input (Step ❶) and then booting the controller; i.e., it sends the start command, initializes the registers, and sets the addresses to consider for food samples and/or reference genomes in the controller. In a nutshell, Acc-Demeter accelerates Steps ❷, ❸, and ❹ of Demeter. The controller returns the results of Step ❹ to the CPU for final processing and performing the relative abundance estimation (Step ❺). We discuss these units in more detail next.

B. ITEM MEMORY (IM) DESIGN

We implement our IM using PCM arrays and corresponding circuits, such as decoders. IM stores the atomic HD vectors. Binary "0" and binary "1" in an HD vector translate to amorphous and crystalline states, respectively. In the beginning, the user (or Demeter) generates 4 HD vectors for each DNA alphabet in Step ❶ of Demeter and stores them in the IM. Acc-Demeter reads these atomic HD vectors from IM every time it meets a new symbol. Once Demeter fixes the HD space, IM becomes a read-only memory. This allows us to prevent unwanted changes to the atomic vectors.

Fig. 8-(A) presents the IM design. The gate enabler provides access to cells that the row decoder activated. This way, the design of an entire array is achieved much easier, and the write/read disturbance effect is also mitigated to a great extent. However, this design also blocks the write on a row basis and only allows column-wise programming of IM. This does not complicate IM in any way because the atomic vectors are generated once in the beginning by the host CPU and then stored in the IM for a long time. Note that random number generators are already well-optimized in CPUs. In addition, randomly generated values inside memristors are still in early stages [67]–[69], and Acc-Demeter can be modified later to benefit from a non-intrusive (compatible) random number generator in the future.

Fig. 8-(B) presents (1) data mapping and (2) placement of HD vectors in the IM unit. Note that data mapping is a critical contribution of Acc-Demeter. Acc-Demeter uses a hybrid row-major and column-major data mapping for IM and AM units, respectively. IM enjoys a row-major data mapping for two reasons. First, a row-major data mapping of HD vectors allows Acc-Demeter to read the cells written in one row in one cycle. This is helpful as IM is used in the encoding procedure, which is the bottleneck. Second, the used PCM model provides more #columns than #rows. Therefore, even if there was a method to read column cells all at once but separately, one could only store smaller chunks of an HD vector on that column.

An important design choice regarding IM is related to the limited size of PCM arrays (512×2048 [41]). This limitation of array size (which also exists in mature memory technologies such as DRAM) prevents us from fitting an entire large HD vector in one row or column. Therefore, one needs to break such an HD vector into smaller chunks and store them into separate rows. Three options exist: (1) putting the chunks in the same array, (2) putting them in different arrays, (3) a hybrid approach. As shown in Section VII, encoder is the bottleneck of our operation. Therefore, to prevent exacerbating the overhead of the encoding procedure, IM breaks a HD vector to the largest power of two that is smaller than the number of columns available in an array (2048 in our case) and stores different chunks on different arrays. This is a direct tradeoff between the used area (#arrays) and performance. Fig. 8-(B) also shows this placement.

C. ENCODER DESIGN

The encoder is the main compute unit of Acc-Demeter. The encoder is implemented in the periphery of arrays and executes the binding and bundling operations via a sequence of commands determined by the controller. Demeter is capable of handling different representations (Section III). However, to reduce the complexity and make the design hardware friendly, the current design of Acc-Demeter only supports binary representations. In this setup, the N-gram encoding mechanism is the most common one, which Acc-Demeter supports. We suspect that other choices are also possible with the same hardware or minimal changes. We leave the exploration of those designs for future work.

Based on Equation 1, building an N-gram requires only simple XOR and shift operations. This bitwise XOR operation can be quickly computed after reading the atomic HD vector from the IM with an XOR gate in the periphery. Note that one can also implement XOR using bitwise AND ($\wedge$) and OR ($\vee$). However, this technique requires breaking the XOR operations into minterms whose numbers increase exponentially. Any attempt to reduce them, even if empirically works as in [41], will only produce approximated results and hurt the accuracy. Although some applications can tolerate such extreme accuracy loss, food profiling cannot. Note that the 2-minterm based encoding in [41] also affects the sparsity of N-grams (acknowledged in the paper) and limits the size of the N-gram. However, this is not the case in Acc-Demeter because all the operations accurately use XOR gates. This

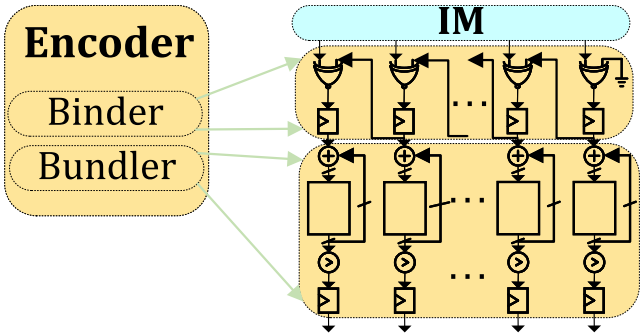


FIGURE 9. Encoder components and schematic.

way, Acc-Demeter can benefit from larger N-grams and does not hurt the density of the HD vectors. As discussed in Section IV-C, the shift operation can quickly become a bottleneck for large HD vectors and strings in a software-based implementation. However, this does not happen here since Acc-Demeter realizes the shift for free by simply redirecting each Flip-Flop (FF)'s stored value to the neighboring one every clock cycle. Fig. 9 depicts a schematic illustration of the encoder unit.

From the hardware perspective, the encoder distinguishes the binding and bundling components completely. For the binding, the SA reads out the value from IM to one input of an XOR gate and uses the previously stored value of neighbor FFs as the second input. The encoder then stores the results in a buffer and repeats the procedure. This design choice provides Acc-Demeter with the cascaded logical operations, with minimum changes to the memory array, and prevents any write back and pressuring the endurance of the PCM substrates. The encoder performs this sequence N times (enforced by the signals from the controller) to build an N-gram. After it finishes creating one N-gram, it passes the N-gram to the bundler unit, resets the buffer, and starts building the next N-gram until it hits either the last character of the input or set limit per final HD vector.

The bundler takes N-grams and adds them to a global HD vector that presents each position with a counter instead of only one bit per position. It then repeats this operation for M N-grams. Finally, the bundler applies a threshold (T) and makes a final binary HD vector representing all the processed characters while building this vector. At this point, the encoder is done. It passes the results to be stored as prototype HD vectors or used as query HD vector in AM and resets both the integer-based and binary HD vectors.

D. ASSOCIATE MEMORY (AM) DESIGN

The AM unit is implemented using PCM arrays and their corresponding circuitry, similar to the IM unit. This unit takes the output of the encoding mechanism (an HD vector) as input. Although the AM and the IM can technically be combined, Acc-Demeter considers separate hardware for three reasons. First, these units serve in subsequent and completely different steps in a profiling pipeline, naming encoding, and classification step. Such a distinct separation enables building a pipeline for them. Second, row-major and column-major data mapping in IM and introduce different parallelism opportunities for encoding and classification steps of a profiling pipeline, respectively. Row-major data mapping of IM parallelizes encoding of all bits in a single HD vector in each clock cycle. On the other hand, column-major mapping of parallelizes the similarity check of one query HD vector with all prototype HD vectors stored vertically in at that clock cycle. Third, separate hardware helps us to simplify IM design by using sense amplifiers instead of ADCs. Doing so brings various benefits in terms of area saving, energy consumption, and read-out time. Note that ADCs are usually the bottleneck of a memristor-based memory in terms of energy, area, and time [70] and that is why one only uses them when VMM or other logical operations such as Scouting Logic [43] are necessary.

Equation 2 shows that for the classification, we need to count the differences between the query HD vector and each prototype HD vector and then decide whether or not it can belong to the corresponding class. Although one can realize this in hardware by performing XNOR operation between the two vectors followed by a pop-count operation all in the periphery, such design comes with two drawbacks: (1) it requires the pop-count operation even after the XNOR, which introduces enormous area cost and significant delay ($log_2 D + 1$ cycles [71]), and (2) the AM unit, similar to the IM, only allows to write columns, not the rows. Since prototype HD vectors are not known from the beginning (unlike atomic HD vectors), this limitation forces us to save them all in another extra unit first and then write them back on a row basis. This is again inefficient.

However, Acc-Demeter proposes a new column-major data mapping and intelligent data duplication for this unit and exploits the characteristics of the PCM substrate to solve all these problems for HDC-based classification. It is well-known that memristor-based memory technologies can perform vector-matrix multiplication [61]–[64]. Therefore, Acc-Demeter implements the required XNOR and following pop-count operations in Equation 2 in four steps, three of which happen in the unit and the last one in the Similarity Check unit.

Step 1: Acc-Demeter stores one prototype HD vector (or a chunk of one HD vector) in one column and its complement in the same column number of a second array. Fig. 10-A shows their placement in the AM unit. **Step 2:** Acc-Demeter applies the query HD vector (Q) to the rows of the first array and the complement of the query HD vector ($\bar{Q}$) to the rows of the second array with complement prototype HD vectors (Ps), shown in Fig. 10-B. **Step 3:** Acc-Demeter enables columns consecutively and effectively read out the number of ones in $Q.P$ and $\bar{Q}.\bar{P}$ in ADCs of each array. This way, it performs two vector-matrix multiplications using Kirchhoff's law, one between Q and all Ps in the first array and one between $\bar{Q}$ and all $\bar{P}s$. Section V-E describes **Step 4** that realizes XNOR and pop-count operation simultaneously. Fig. 10-B presents a high-level illustration of AM design.

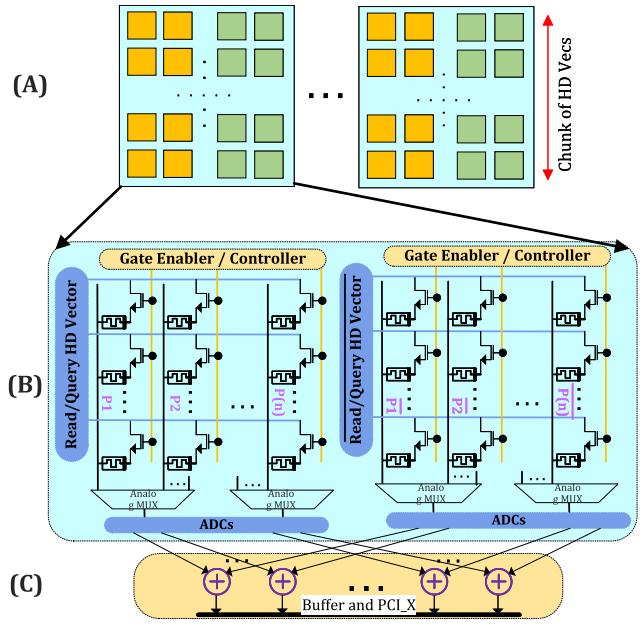


FIGURE 10. (A) Data mapping and placement of prototype HD vectors in, (B) high-level design, and (C) partial hardware for similarity check unit.

Similar to the case in IM, the limited array size of PCM substrates also prevents Acc-Demeter from storing a full HD vector in one row or column of AM. To reduce the required area, and since the encoding is the bottleneck and not the classification (Section VII), in the AM, unlike IM, Acc-Demeter stores the chunks of HD vectors in the same array. Fig. 10-A takes a color-coding approach and depicts the way Acc-Demeter breaks prototype HD vectors into multiple chunks and stores them in columns of AM in and among tiles. It is worth noting that Acc-Demeter only writes to the PCM cells once in both IM and AM units unless either the configuration file in Step ❶ or the user the default reference genome database in Step ❷ changes. This prevents many writes to the devices, which still have limited endurance compared to traditional memory technologies.

E. SIMILARITY CHECK HARDWARE

The similarity check unit is a small computing unit that takes the two ADCs' output of similar columns from the two crossbars and adds them together (**Step 4**). Fig. 10-C depicts all the logic for this unit. The output of this unit is the results of XNOR and pop-count together. At this stage, the similarity check unit sends the results out to the host CPU to determine whether the similarity is close to the threshold and should be considered in the abundance estimation (❹-b, and ❺). The reason behind sending the results out instead of a winner-take-all (WTA) circuit used in previous works [41], [72] is two-folded. First, a WTA circuit assumes that the query matches one and only one prototype HD vector. However, as discussed in Section III-D and III-E, this is not always the case when profiling the genomics data. Second, the relative abundance estimation techniques (Step ❺ in Fig. 1), although simple, require more complex and area-hungry logic circuits, which Acc-Demeter aims to avoid whenever possible. Therefore, since the results will be analyzed outside the PCM-substrate anyway and transferring such small data can be easily handled by interconnects between the host and Acc-Demeter, Acc-Demeter relies on the host CPU to perform the final steps of Demeter. Note that the host is aware of prototype HD vectors' mappings.

F. CONTROLLER UNIT

The controller orchestrates all the operations of Acc-Demeter by generating control signals for other components. It gets the start signal and the address of food samples (or reference genomes) in the memory as its inputs. The controller outputs the results of the similarity check unit back to the host for final steps. The controller is designed as a simple FSM machine and operates based on parameters set in Step ❶.

VI. SYSTEM INTEGRATION OF ACC-DEMETER

This section discussed Acc-Demeter's system integration stack that enables it to operate with the host processing system.

A. ADDRESS TRANSLATION

Acc-Demeter works with physical addresses, instead of virtual ones, and is relieved of address translation challenges that exist and dealt with in previous works [73], [74]. The CPU host uses the same translation lookaside buffer (TLB) lookup mechanism that exist for normal load/store operations to translate instructions's virtual memory addresses into their physical addresses when we have a Acc-Demeter's instruction.

B. COHERENCE

Acc-Demeter may require modified and/or generated atomic vectors (for the IM units) or loaded prototype vectors (for the AM units). Similar to previous works [75]–[77], ensuring that data for Acc-Demeter is up-to-date is a responsibility for programmers and can be achieved easily by flushing cache lines. Acc-Demeter is also capable of leveraging previous PIM coherence optimizations [78], [79] for further performance improvement.

C. INTERRUPTS

We assume that the pages required by Acc-Demeter's AM and IM units are already present. When this is not the case, we rely on the conventional mechanisms for handling the page faults to place this data into the correct arrays. Therefore, Acc-Demeter does not face page fault during the execution of food profiling since pages used by Acc-Demeter are already loaded and pinned into AM and IM units. Acc-Demeter may, however, face an interrupt during a context switch. In such cases, the context of the control unit in Acc-Demeter will be saved and then restored when the profiler resumes.

D. ISA EXTENSIONS AND PROGRAMMING INTERFACE

An expressive and efficient programming interface is a must for Acc-Demeter as it directly impacts the usability of

TABLE 1. Acc-Demeter ISA extensions.

Type	ISA Format
Initialization	*bbop_init*, *address*, *size*, *n*
Input Operation	*bbop_op*, *size*, *n*

TABLE 2. PCM configuration.

Technology	PCM (512*2048 @1bit), Cell Size = 50 F^2
Current on Conducting Devices	0.1 µA
Read Voltage	0.1 V
Read/Write Latency	Read=2.8 ns, write=100 ns
ADC	9 bits resolution, 2 ns, 4 pJ per sample

Demeter framework. To enable easy communication between Acc-Demeter and the programmer, we envision to extend the ISA with a few instructions to allow the control unit knows about the required operations, their timing, and the place where data objects reside in IM and AM units. ISA extension is possible due to the unused opcode space in the host CPU, and has also been adopted in previous PIM-related architectures [36], [73].

Acc-Demeter requires 2 types of instructions: (1) *bbop_initaddress, size, n*: initialization of IM and AM units and (2) *bbop_opsize, n*: instructions for performing different operations in Acc-Demeter. *bbop_init* is the initialization instruction that informs the OS that the memory object is for Acc-Demeter. This way the OS performs virtual-to-physical memory mapping required for AM and IM units. *bbop_init* takes the base physical address, the size of the vector, and the intended value. For Acc-Demeter's operations, we extend the CPU ISA with *bbop_op*. Acc-Demeter utilizes an array-based computation model, i.e., *src* and *dst* are source and destination arrays. *bbop_op* is the opcode, where *size* and *n* are #elements in the array and #bits in each array element, respectively. This paper assumes that the programmer will writing suitable code for Acc-Demeter operations manually. We summarized the required CPU ISA extensions for these operations in Table 1.

VII. ACC-DEMETER's EVALUATION

A. METHODOLOGY

We emulate the execution of Acc-Demeter using a cycle-accurate RTL model and synthesized it using UMC 65 nm technology node in Synopsys Design Compiler [80]. We verify the correct behavior of our memory model using test benches and previous in-memory simulators [41], [62]. We consider a typical operation condition of temperature 25° and voltage 1.2V when evaluating our energy consumption. All the experiments for the PCM-based Acc-Demeter are carried out based on PCM statistical models that capture the variations in the spatiotemporal conductivity of the devices. PCM prototypes and analytical models used for validation and further simulations are based on the results of EU project MNEMOSENE [81], led and concluded by TU Delft in 2020. Table 2 shows the other parameters of our PCM crossbars.

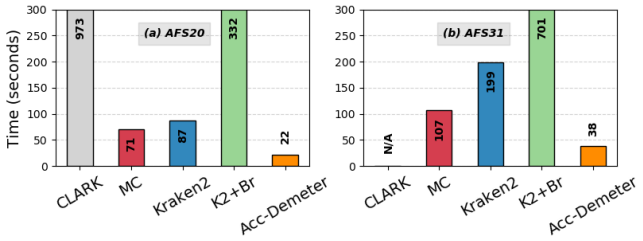


FIGURE 11. Build time on (a) AFS20 and (b) AFS31.

B. ACC-DEMETER's PERFORMANCE ANALYSIS

This section compares the performance of SOTA profilers compared to Acc-Demeter, our PIM-enabled accelerator design of Demeter.

1) BUILD TIME

Fig. 11 shows the build time that each profiler takes to build its initial data structure for two reference databases AFS20 and AFS31.

We make two observations. First, Acc-Demeter has the lowest build time among all previous food profilers. Acc-Demeter builds HD-RefDB corresponding to AFS20 and AFS31 ∼3.2× and 2.8× faster, respectively than MetaCache, the next fastest profiler. Unlike previous HDC-based methods that are faster than their ML competitors due to the one-shot learning ability of HDC paradigm, Acc-Demeter outperforms SOTA profilers due to its highly parallelized performance and simple operations being performed on Acc-Demeter's hardware. SOTA food profilers parse the reference genomes only once, and the one-shot learning of Demeter is not particularly advantageous.

Second, CLARK exceeds the 500 GB memory of the system when running it for AFS31. This is in line with observations in [4]. Therefore, we excluded it from all analyses regarding AFS31 from now on. This case shows an excellent example of where metagenomic profilers, whereas good for lengthy and costly studies, may not be applicable for the scenario of food profiling and later food analysis and monitoring.

2) QUERY TIME

Fig. 12 presents the time that each profiler takes to query one (short) read from the query food sample and classify its specie(s) over AFS20 and AFS31.

We make two key observations. Acc-Demeter improves the query time by ∼74×/88× and 272×/350× compared to Kraken2 and MetaCache, respectively, on AFS20/AFS31. This shows that the acceleration of Demeter pays off and finally makes Demeter not only an accurate but also a fast food profiler.

Second, the query time for Acc-Demeter remains almost the same for both databases and does not change much. We further investigate this and realize a bottleneck shift: Step ❺ or abundance estimation that is being performed inside the CPU is now the bottleneck of Acc-Demeter.

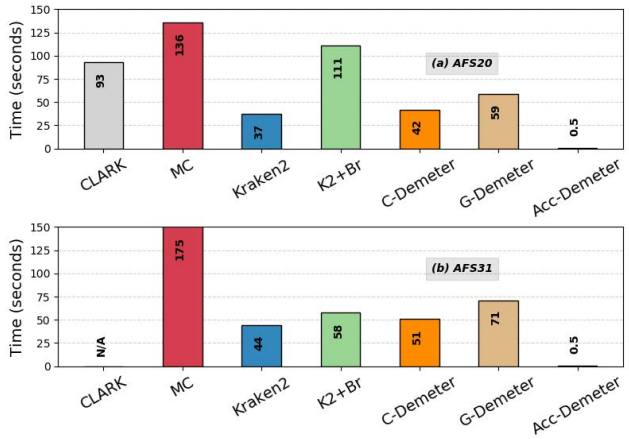


FIGURE 12. Query time on (a) AFS20 and (b) AFS31.

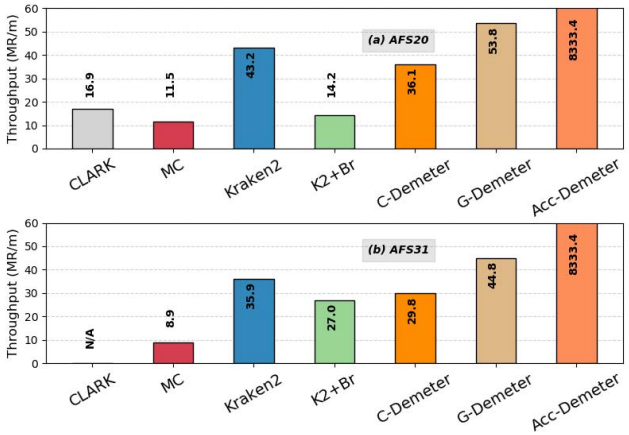


FIGURE 13. Query throughput on (a) AFS20 and (b) AFS31.

This happens because of the high-frequency Acc-Demeter achieved. However, this contrasts with other profilers that spend most of their time querying their massive data structure.

3) QUERY THROUGHPUT

Fig. 13 shows the throughput of different profilers over AFS20 and ASF31.

We make two observations. First, Acc-Demeter provides throughput improvement of ∼192× and 232× for both AFS20 and AFS31, respectively, compared to Kraken2, the second food profiler regarding throughput. This more remarkable improvement in throughput than query time results from Acc-Demeter's ability to classify one query read in parallel with the encoding of the following query. Note that the throughput analysis of the previous profiler does not consider the time for loading their data structure. Second, similar to the query time, throughput is almost the same regardless of the database due to the bottleneck shift. We conclude that Acc-Demeter significantly outperforms all four SOTA baselines for all performance metrics.

C. ACC-DEMETER's POWER AND AREA ANALYSIS

Table 3 provides the area and energy consumption breakdown of different components in Acc-Demeter per query on AFS31.

TABLE 3. Area and power breakdown of Acc-Demeter.

Unit	Area (mm^2)	Area (%)	Energy (nj)	Energy (%)
IM	0.07	3.1	1.179E-06	7.4
Encoder	1.375	78.3	1.43E-05	90.6
AM	0.15	8.4	2.47E-07	1.56
Similarity	0.1815	10.2	6.91E-08	0.4

We make two observations. First, the logic for the encoder unit is the most energy and area hungry unit among all others, more than 90% and 78% energy and area of the whole Acc-Demeter. This is expected because (1) the encoder consists of many CMOS circuits whereas AM and IM are small memory units with PCM technology and (2) the encoder is in the heart of all operations in Demeter, and we spend most of our time in this unit. We argue that this amount of logic around our array is still justifiable. Second, compared to the die area in an Intel Xeon E5-2697 CPU [82], Acc-Demeter only has an area overhead of less than 2%. We conclude that Acc-Demeter is low-cost in terms of die area.

Our evaluations show that Acc-Demeter is capable of performing 9.45Mbp query per joule. Unfortunately, measuring the energy consumption of other profilers and having an apple-to-apple comparison between the energy consumption of this method with other ones is hard. However, Merelli *et al.* [83], [84] show that running Kraken2 with querying an even smaller data structure built from a reduced reference genome dataset, minikraken [83], [85], can incur more energy (maximum of $0.6\frac{Mbp}{j}$). This considerable difference happens because of three reasons: (1) Kraken2 queries a more complex data structure compared to Acc-Demeter and requires more complex operations, (2) Kraken2 queries a bigger data structure for its query, and (3) Kraken2 incurs significant data movement between the memory and the processing unit. All of these limitations exist in similar forms in CLARK and MetaCache. We conclude that Acc-Demeter is more energy-efficient than all four SOTA baselines.

VIII. DISCUSSIONS AND FUTURE WORKS

Capacity. We define the capacity of Demeter as the ratio between the number of reference genomes encoded as prototype HD vectors to the size of HD space for a competitive profiling accuracy target. The higher #prototype HD vectors are, the bigger capacity is needed, resulting in bigger HD space and lower efficiency. Therefore, if one uses Demeter, as is, as a metagenomics profiler, they cannot expect similar improvements compared to SOTA metagenomics profilers (e.g., Kraken2, on those datasets. We are currently investigating the additional techniques to enable Demeter for those cases as well. However, we leave further analysis of required changes to Demeter for supporting metagenomics profiling or other profiling studies with many reference genomes for future work.

A. SUPPORTED FUNCTIONS AND REPRESENTATIONS

As discussed (Sections II, III, and V), Acc-Demeter currently supports only binary representations and N-gram encoding mechanism. This is a design choice made for simplicity and is

based on acceptable accuracy results of the software version. We leave the hardware for other encoding mechanisms and data representations for future work.

IX. RELATED WORKS

To our knowledge, Demeter is the first paper to propose a framework to perform food profiling using HDC. Acc-Demeter is also the first hardware accelerator that enables low-cost and accurate in-memory profiling for a typical reference database in food profilers. We have already compared Demeter and Acc-Demeter extensively to SOTA profilers in Sections IV and VII. This section briefly discusses previous software works for profilers (food or metagenomics), software or hardware of HDC-based systems, and PIM-enabled accelerators.

A. METAGENOMIC PROFILERS

Several recent works propose approaches and techniques to directly or indirectly accelerate or improve the accuracy of metagenomics profiling, the first step of such studies. These works take three approaches: (1) Reducing the reference database's size by pre-alignment filtering [86], [87] or heuristics for taxonomic classification techniques [55], [88]–[91], (2) Accelerating read alignment or assembly (only for alignment-/assembly-based profilers) on CPUs, FPGAs, or GPUs [92]–[98], (3) post-alignment/-assembly/-classification presence and abundance estimation heuristics [54], [55], [99]. Demeter is categorized in the first group, taking a HDC-based approach for the first time. However, compared to the first group, Acc-Demeter is much faster and has a lower cost (regarding both energy and area consumption). Note that Demeter and Acc-Demeter are orthogonal to works in the third group, and their Step ❺ can adapt their proposed techniques for the abundance estimation after the initial classification.

B. HDC-BASED SYSTEMS

Many works exploit the HDC paradigm for specific machine learning applications that require capturing temporal patterns. These works vary from language [100] and voice [101] detection to seizure detection [102]. Demeter is the first work that investigates HDC in the realm of profiling genomics data. Although HDNA [48] and GenieHD [103] propose to use HDC for (partial[5]) sequence alignment of a single reference genome divided into multiple pieces, they never exploit it for any metagenomics or food profiling.

A few works also suggest various hardware platforms such as FPGAs, GPUs, or ASICs [41], [48] to improve the performance of HDC-based designs. Acc-Demeter is different from all of these designs from two important aspects. First, Acc-Demeter performs an exact pop-count operation in one cycle, performing two VMM in parallel and then adding the outputs of ADCs. This is in contrast to previous works [41], [48], [102], [103] that perform an exact pop-count operation in $log_2 D+1$ cycles, where D is the size of an HD vector. Second, unlike [41], Acc-Demeter can perform the required HDC-based operations on long, non-sparse N-grams (discussed in Section V-C). To the best of our knowledge, Acc-Demeter is the only PIM-enabled accelerator for food profiling.

[5]Neither HDNA nor GenieHD is capable of producing the exact type and location of edits between their query and the reference genome as in the typical outputs (.sam file) of a sequence aligner. Hence, the term "partial".

C. PIM-ENABLED ACCELERATORS

Prior works also heavily investigate various forms of compute-capable memories [36], [104], [105]. Among these, only a few use in-memory capability for HDC designs [41], [106]. However, these works are either tuned for single tasks or capable of limited sizes for N-grams like only up to 3-grams [106], or only based on compact models from small prototypes with 256 × 256 ReRAM arrays. Demeter is the first work that proposes food profiling inside memory. In theory, one can accelerate Demeter using a PIM-enabled design on DRAM, SRAM, and other technologies. However, for the reasons iterated in Section V, Acc-Demeter exploits PCM to improve the performance of Demeter.

X. CONCLUSION

This paper introduces Demeter, the first framework that enables profiling of food samples via HDC whereas strictly meeting the accuracy of state-of-the-art profilers. Demeter uses a five-step approach to enable species-level profiling using HDC. This paper also introduces the first PCM-based PIM-enabled hardware accelerator, called Acc-Demeter. We evaluate Demeter on software and Acc-Demeter using a cycle-accurate model based on a small-scale PCM-based prototype. We design Demeter and Acc-Demeter to (1) address the key challenge of HDC-systems when facing a massive input, (2) eliminate the need for a powerful machine with very large memories, and (3) prevent unnecessary data movement between memory and processing units and therefore prevent wasting time and energy. We achieve significant performance and energy benefits over the SOTA CPU implementations whereas achieving the same accuracy. We hope that future work builds on top of our framework and its hardware and extends it to further improve our food profiling systems.

REFERENCES

[1] D. J. Agnew, J. Pearce, G. Pramod, T. Peatman, R. Watson, J. R. Beddington, and T. J. Pitcher, "Estimating the worldwide extent of illegal fishing," *PLoS ONE*, vol. 4, no. 2, Feb. 2009, Art. no. e4570.
[2] R. Smith, "Rural rogues: A case story on the 'smokies' trade," *Int. J. Entrepreneurial Behav. Res.*, vol. 10, no. 4, pp. 277–294, 2004.
[3] R. Smith, "Documenting the UK 'black fish scandal' as a case study of criminal entrepreneurship," *Int. J. Sociol. Social Policy*, vol. 35, nos. 3–4, pp. 199–221, 2015.
[4] R. Kobus, J. M. Abuín, A. Müller, S. L. Hellmann, J. C. Pichel, T. F. Pena, A. Hildebrandt, T. Hankeln, and B. Schmidt, "A big data approach to metagenomics for all-food-sequencing," *BMC Bioinf.*, vol. 21, no. 1, pp. 1–15, Dec. 2020.
[5] F. Ripp, C. F. Krombholz, Y. Liu, M. Weber, A. Schäfer, B. Schmidt, R. Köppel, and T. Hankeln, "All-food-seq (AFS): A quantifiable screen for species in biological samples by deep DNA sequencing," *BMC Genomics*, vol. 15, no. 1, pp. 1–11, Dec. 2014.

[6] Wetterstrand KA. *DNA Sequencing Costs: Data From the NHGRI Genome Sequencing Program (GSP)*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.genome.gov/sequencingcostsdata

[7] M. Barba, H. Czosnek, and A. Hadidi. *Cost in U.S. Dollars Per Raw Megabase of DNA Sequence*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data

[8] M. Alser, Z. Bingol, D. S. Cali, J. Kim, S. Ghose, C. Alkan, and O. Mutlu, "Accelerating genome analysis: A primer on an ongoing journey," *IEEE Micro*, vol. 40, no. 5, pp. 65–75, Sep. 2020.

[9] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman, "Molecular biological access to the chemistry of unknown soil microbes: A new frontier for natural products," *Chem. Biol.*, vol. 5, no. 10, pp. R245–R249, Oct. 1998.

[10] M. R. Rondon, P. R. August, A. D. Bettermann, S. F. Brady, T. H. Grossman, M. R. Liles, K. A. Loiacono, B. A. Lynch, I. A. MacNeil, C. Minor, C. L. Tiong, M. Gilman, M. S. Osburne, J. Clardy, J. Handelsman, and R. M. Goodman, "Cloning the soil metagenome: A strategy for accessing the genetic and functional diversity of uncultured microorganisms," *Appl. Environ. Microbiol.*, vol. 66, no. 6, pp. 2541–2547, Jun. 2000.

[11] J. C. Wooley, A. Godzik, and I. Friedberg, "A primer on metagenomics," *PLoS Comput. Biol.*, vol. 6, no. 2, Feb. 2010, Art. no. e1000667.

[12] D. E. Wood, J. Lu, and B. Langmead, "Improved metagenomic analysis with kraken 2," *Genome Biol.*, vol. 20, no. 1, pp. 1–13, Dec. 2019.

[13] A. E. Pérez-Cobas, L. Gomez-Valero, and C. Buchrieser, "Metagenomic approaches in microbial ecology: An update on whole-genome and marker gene sequencing analyses," *Microbial Genomics*, vol. 6, no. 8, Aug. 2020, Art. no. mgen000409.

[14] A. B. R. McIntyre, R. Ounit, E. Afshinnekoo, R. J. Prill, E. Hénaff, N. Alexander, S. S. Minot, D. Danko, J. Foox, S. Ahsanuddin, S. Tighe, N. A. Hasan, P. Subramanian, K. Moffat, S. Levy, S. Lonardi, N. Greenfield, R. R. Colwell, G. L. Rosen, and C. E. Mason, "Comprehensive benchmarking and ensemble approaches for metagenomic classifiers," *Genome Biol.*, vol. 18, no. 1, pp. 1–19, Dec. 2017.

[15] F. P. Breitwieser, J. Lu, and S. L. Salzberg, "A review of methods and databases for metagenomic classification and assembly," *Briefings Bioinf.*, vol. 20, no. 4, pp. 1125–1136, Jul. 2019.

[16] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *IEEE Circuits Syst. Mag.*, vol. 20, no. 2, pp. 30–47, 2nd Quart., 2020.

[17] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5880–5898, Dec. 2018.

[18] H. A. D. Nguyen, J. Yu, M. A. Lebdeh, M. Taouil, S. Hamdioui, and F. Catthoor, "A classification of memory-centric computing," *ACM J. Emerg. Technol. Comput. Syst. (JETC)*, vol. 16, no. 2, pp. 1–26, 2020.

[19] S. Hamdioui, L. Xie, H. A. D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike, and J. Van Lunteren, "Memristor based computation-in-memory architecture for data-intensive applications," in *Proc. IEEE Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 1718–1725.

[20] Intel. (2018). *Intel VTune Amplifier 2019 User Guide*. [Online]. Available: https://software.intel.com/en-us/vtune-amplifier-help

[21] L. Wu, R. Sharifi, M. Lenjani, K. Skadron, and A. Venkat, "Sieve: Scalable *in-situ* DRAM-based accelerator designs for massively parallel k-mer matching," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 251–264.

[22] M. Zhou, L. Wu, M. Li, N. Moshiri, K. Skadron, and T. Rosing, "Ultra efficient acceleration for de novo genome assembly via near-memory computing," in *Proc. 30th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Sep. 2021, pp. 199–212.

[23] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognit. Comput.*, vol. 1, no. 2, pp. 139–159, Oct. 2009.

[24] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," 2004, *arXiv:cs/0412059*.

[25] F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2016, p. 1.

[26] D. Kleyko, E. Osipov, N. Papakonstantinou, V. Vyatkin, and A. Mousavi, "Fault detection in the hyperspace: Towards intelligent automation systems," in *Proc. IEEE 13th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2015, pp. 1219–1224.

[27] D. Kleyko, E. Osipov, R. W. Gayler, A. I. Khan, and A. G. Dyer, "Imitation of honey bees' concept learning processes using vector symbolic architectures," *Biologically Inspired Cognit. Archit.*, vol. 14, pp. 57–72, Oct. 2015.

[28] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Oct. 2016, pp. 1–8.

[29] P. Kanerva, J. Kristoferson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proc. Annu. Meeting Cognit. Sci. Soc.*, vol. 22, no. 22, pp. 1–2, 2000.

[30] T. A. Plate, "Holographic reduced representations," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 623–641, May 1995.

[31] R. W. Gayler, "Multiplicative binding, representation operators & analogy," *Workshop Poster*, Jan. 1998.

[32] S. I. Gallant and P. Culliton, "Positional binding with distributed representations," in *Proc. Int. Conf. Image, Vis. Comput. (ICIVC)*, Aug. 2016, pp. 108–113.

[33] D. A. Rachkovskij, "Representation and processing of structures with binary sparse distributed codes," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 2, pp. 261–276, Mar./Apr. 2001.

[34] S. I. Gallant and T. W. Okaywe, "Representing objects, relations, and sequences," *Neural Comput.*, vol. 25, no. 8, pp. 2038–2078, 2013.

[35] T. A. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. Franklin, TN, USA: TNN, 2003.

[36] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 273–287.

[37] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Phase change memory architecture and the quest for scalability," *Commun. ACM*, vol. 53, no. 7, pp. 99–106, Jul. 2010.

[38] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018.

[39] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, D. Druist, D. Lottis, V. Nikitin, X. Tang, S. Watts, and S. Wang, "Advances and future prospects of spin-transfer torque random access memory," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1873–1878, Jun. 2010.

[40] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges," *Adv. Mater.*, vol. 21, nos. 25–26, pp. 2632–2663, Jul. 2009.

[41] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electron.*, vol. 3, no. 6, pp. 327–337, Jun. 2020.

[42] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Mater.*, vol. 18, no. 4, pp. 309–323, 2019.

[43] L. Xie, H. A. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFailakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 176–181.

[44] L. Cheng, Y. Li, K. Yin, S. Hu, Y. Su, M. Jin, Z. Wang, T. Chang, and X. Miao, "Functional demonstration of a memristive arithmetic logic unit (MemALU) for in-memory computing," *Adv. Funct. Mater.*, vol. 29, no. 49, Dec. 2019, Art. no. 1905660.

[45] D. Ielmini and H. S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electron.*, vol. 1, no. 6, pp. 333–343, 2018.

[46] H. Li, T. F. Wu, A. Rahimi, K.-S. Li, M. Rusch, C.-H. Lin, J.-L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, W.-C. Chiu, M.-C. Chen, T.-T. Wu, J.-M. Shieh, W.-K. Yeh, J. M. Rabaey, S. Mitra, and H.-S.-P. Wong, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *IEDM Tech. Dig.*, Dec. 2016, pp. 1–16.

[47] T. F. Wu, H. Li, P.-C. Huang, A. Rahimi, J. M. Rabaey, H.-S.-P. Wong, M. M. Shulaker, and S. Mitra, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 492–494.

[48] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, "HDNA: Energy-efficient DNA sequencing using hyperdimensional computing," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat. (BHI)*, Mar. 2018, pp. 271–274.

[49] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Şekerciogğlu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1250–1262, Jun. 2017.
[50] M. Imani, S. Pampana, S. Gupta, M. Zhou, Y. Kim, and T. Rosing, "DUAL: Acceleration of clustering algorithms using digital-based processing in-memory," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Oct. 2020, pp. 356–371.
[51] K. Reinert, "The SeqAn C++ template library for efficient sequence analysis: A resource for programmers," *J. Biotechnol.*, vol. 261, pp. 157–168, Nov. 2017.
[52] M. Harris, "Optimizing parallel reduction in CUDA," *Nvidia Developer Technol.*, vol. 2, no. 4, p. 70, 2007.
[53] AMD. *AMD EPYC 7742 CPU*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.amd.com/en/products/cpu/amd-epyc-7742
[54] N. LaPierre, S. Mangul, M. Alser, I. Mandric, N. C. Wu, D. Koslicki, and E. Eskin, "MiCoP: Microbial community profiling method for detecting viral and fungal organisms in metagenomic samples," *BMC Genomics*, vol. 20, no. S5, pp. 1–10, Jun. 2019.
[55] N. LaPierre, M. Alser, E. Eskin, D. Koslicki, and S. Mangul, "Metalign: Efficient alignment-based metagenomic profiling via containment min hash," *Genome Biol.*, vol. 21, no. 1, pp. 1–15, Dec. 2020.
[56] Project: PRJEB34001. *Calibration Sausages WGS Containing Mammalian and Avian Species*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.ebi.ac.uk/ena/browser/view/PRJEB34001
[57] Project: PRJNA271645. *Calibration Sausages Metagenome*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.ebi.ac.uk/ena/browser/view/PRJNA271645
[58] J. Lu, F. P. Breitwieser, P. Thielen, and S. L. Salzberg, "Bracken: Estimating species abundance in metagenomics data," *PeerJ Comput. Sci.*, vol. 3, p. e104, Jan. 2017.
[59] R. Ounit, S. Wanamaker, T. J. Close, and S. Lonardi, "CLARK: Fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers," *BMC Genomics*, vol. 16, no. 1, pp. 1–13, Dec. 2015.
[60] F. Meyer, "Critical assessment of metagenome interpretation—The second round of challenges," *Nature methods*, vol. 19, no. 4, pp. 429–440, 2022.
[61] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, and J. P. Strachan, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, Mar. 2018, Art. no. 1705914.
[62] M. Zahedi, M. Mayahinia, M. A. Lebdeh, S. Wong, and S. Hamdioui, "Efficient organization of digital periphery to support integer datatype for memristor-based CIM," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2020, pp. 216–221.
[63] S. Choi, J. H. Shin, J. Lee, P. Sheridan, and W. D. Lu, "Experimental demonstration of feature extraction and dimensionality reduction using memristor networks," *Nano Lett.*, vol. 17, no. 5, pp. 3113–3118, 2017.
[64] M. Zahedi, R. van Duijnen, S. Wong, and S. Hamdioui, "Tile architecture and hardware implementation for Computation-in-Memory," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2021, pp. 108–113.
[65] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-M.-W. Hwu, J. P. Strachan, K. Roy, and D. S. Milojicic, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 715–731.
[66] Q. Lou, S. C. Janga, and L. Jiang, "Helix: Algorithm/Architecture co-design for accelerating nanopore genome base-calling," in *Proc. ACM Int. Conf. Parallel Archit. Compilation Techn.*, Sep. 2020, pp. 293–304.
[67] G. Diarce, Á. Campos-Celador, K. Martin, A. Urresti, A. García-Romero, and J. M. Sala, "A comparative study of the CFD modeling of a ventilated active façade including phase change materials," *Appl. Energy*, vol. 126, pp. 307–317, Aug. 2014.
[68] S. Balatti, S. Ambrogio, Z. Wang, and D. Ielmini, "True random number generation by variability of resistive switching in oxide-based devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 214–221, Jun. 2015.
[69] F. M. Puglisid, N. Zagni, L. Larcher, and P. Pavan, "A new verilog—A compact model of random telegraph noise in oxide-based RRAM for advanced circuit design," in *Proc. 47th Eur. Solid-State Device Res. Conf. (ESSDERC)*, Sep. 2017, pp. 204–207.
[70] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.
[71] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring hyperdimensional associative memory," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 445–456.
[72] R. G. Carvajal, J. Ramirez-Angula, and J. Tombs, "High-speed high-precision voltage-mode MIN/MAX circuits in CMOS technology," in *Proc. IEEE Int. Symp. Circuits Syst. Emerg. Technol. 21st Century.*, May 2000, pp. 13–16.
[73] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Jun. 2015, pp. 336–348.
[74] J. Picorel, D. Jevdjic, and B. Falsafi, "Near-memory address translation," in *Proc. 26th Int. Conf. Parallel Architectures Compilation Techn. (PACT)*, Sep. 2017, pp. 303–317.
[75] N. Hajinazar, G. F. Oliveira, S. Gregorio, J. D. Ferreira, N. M. Ghiasi, M. Patel, M. Alser, S. Ghose, J. Gómez-Luna, and O. Mutlu, "SIMDRAM: A framework for bit-serial SIMD processing using DRAM," in *Proc. 26th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2021, pp. 329–345.
[76] A. Holdings, "Cortex-A8 technical: Reference manual," Tech. Rep., 2010.
[77] P. Guide, "Intel 64 and IA-32 architectures software developer's manual," *Volume 3B, Syst. Program. Guide*, vol. 2, no. 11, p. 1, 2011.
[78] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "LazyPIM: An efficient cache coherence mechanism for processing-in-memory," *IEEE Comput. Archit. Lett.*, vol. 16, no. 1, pp. 46–50, Jan./Jun. 2017.
[79] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, R. Ausavarungnirun, K. Hsieh, N. Hajinazar, K. T. Malladi, H. Zheng, and O. Mutlu, "CoNDA: Efficient cache coherence support for near-data accelerators," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 629–642.
[80] Synopsys, Inc. *Synopsys Design Compiler*. Accessed: Jul. 1, 2021. [Online]. Available: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html
[81] MNEMOSENE Partners. (2020). *The MNEMOSENE Project*. [Online]. Available: http://www.mnemosene.eu/
[82] D. Fujiki, S. Mahlke, and R. Das, "Duality cache for data parallel acceleration," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 397–410.
[83] I. Merelli, L. Morganti, E. Corni, C. Pellegrino, D. Cesini, L. Roverelli, G. Zereik, and D. D'Agostino, "Low-power portable devices for metagenomics analysis: Fog computing makes bioinformatics ready for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 88, pp. 467–478, Nov. 2018.
[84] D. D'Agostino, L. Morganti, E. Corni, D. Cesini, and I. Merelli, "Combining edge and cloud computing for low-power, cost-effective metagenomics analysis," *Future Gener. Comput. Syst.*, vol. 90, pp. 79–85, Jan. 2019.
[85] Ö. Eyice, M. Namura, Y. Chen, A. Mead, S. Samavedam, and H. Schäfer, "SIP metagenomics identifies uncultivated methylophilaceae as dimethylsulphide degrading bacteria in soil and lake sediment," *ISME J.*, vol. 9, no. 11, pp. 2336–2348, Nov. 2015.
[86] H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan, "Accelerating read mapping with FastHASH," *BMC Genomics*, vol. 14, no. S1, pp. 1–13, Jan. 2013.
[87] M. Alser, T. Shahroodi, J. Gómez-Luna, C. Alkan, and O. Mutlu, "SneakySnake: A fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs," *Bioinformatics*, vol. 36, nos. 22–23, pp. 5282–5290, Apr. 2021.
[88] N. Segata, L. Waldron, A. Ballarini, V. Narasimhan, O. Jousson, and C. Huttenhower, "Metagenomic microbial community profiling using unique clade-specific marker genes," *Nature Methods*, vol. 9, no. 8, pp. 811–814, Aug. 2012.
[89] B. Liu, T. Gibbons, M. Ghodsi, T. Treangen, and M. Pop, "Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences," *Genome Biol.*, vol. 12, no. S1, pp. 1–27, Dec. 2011.
[90] D. E. Wood and S. L. Salzberg, "Kraken: Ultrafast metagenomic sequence classification using exact alignments," *Genome Biol.*, vol. 15, no. 3, pp. 1–12, 2014.

[91] R. Kobus, C. Hundt, A. Müller, and B. Schmidt, "Accelerating metagenomic read classification on CUDA-enabled GPUs," *BMC Bioinf.*, vol. 18, no. 1, pp. 1–10, Dec. 2017.
[92] A. Brady and S. Salzberg, "PhymmBL expanded: Confidence scores, custom databases, parallelization and more," *Nature methods*, vol. 8, no. 5, p. 367, 2011.
[93] J. Daily, "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments," *BMC Bioinf.*, vol. 17, no. 1, pp. 1–11, Dec. 2016.
[94] S. S. Banerjee, M. El-Hadedy, J. B. Lim, Z. T. Kalbarczyk, D. Chen, S. S. Lumetta, and R. K. Iyer, "ASAP: Accelerated short-read alignment on programmable hardware," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 331–346, 2018.
[95] X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "FPGASW: Accelerating large-scale Smith–Waterman sequence alignment application with backtracking on FPGA linear systolic array," *Interdiscipl. Sci., Comput. Life Sci.*, vol. 10, no. 1, pp. 176–188, 2018.
[96] E. J. Houtgast, V.-M. Sima, K. Bertels, and Z. Al-Ars, "An FPGA-based systolic array to accelerate the BWA-MEM genomic mapping algorithm," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling, Simulation (SAMOS)*, Jul. 2015, pp. 221–227.
[97] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: Accelerating smith-waterman protein database search by coupling CPU and GPU SIMD instructions," *BMC Bioinf.*, vol. 14, no. 1, pp. 1–10, Dec. 2013.
[98] R. Luo, T. Wong, J. Zhu, C.-M. Liu, X. Zhu, E. Wu, L.-K. Lee, H. Lin, W. Zhu, D. W. Cheung, H.-F. Ting, S.-M. Yiu, S. Peng, C. Yu, Y. Li, R. Li, and T.-W. Lam, "SOAP3-dp: Fast, accurate and sensitive GPU-based short read aligner," *PLoS ONE*, vol. 8, no. 5, May 2013, Art. no. e65632.
[99] D. T. Truong, E. A. Franzosa, T. L. Tickle, M. Scholz, G. Weingart, E. Pasolli, A. Tett, C. Huttenhower, and N. Segata, "MetaPhlAn2 for enhanced metagenomic taxonomic profiling," *Nature Methods*, vol. 12, no. 10, pp. 902–903, Oct. 2015.
[100] M. Imani, J. Hwang, T. Rosing, A. Rahimi, and J. M. Rabaey, "Low-power sparse hyperdimensional encoder for language recognition," *IEEE Des. Test*, vol. 34, no. 6, pp. 94–101, Dec. 2017.
[101] Y. Kim, M. Imani, and T. S. Rosing, "Efficient human activity recognition using hyperdimensional computing," in *Proc. 8th Int. Conf. Internet Things*, Oct. 2018, pp. 1–6.
[102] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "Hyperdimensional computing with local binary patterns: One-shot learning of seizure onset and identification of ictogenic brain regions using short-time iEEG recordings," *IEEE Trans. Biomed. Eng.*, vol. 67, no. 2, pp. 601–613, Feb. 2020.
[103] Y. Kim, M. Imani, N. Moshiri, and T. Rosing, "GenieHD: Efficient DNA pattern matching accelerator using hyperdimensional computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 115–120.
[104] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "DRISA: A DRAM-based reconfigurable *in-situ* accelerator," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 288–301.
[105] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," *ACM SIGARCH Comput. Archit. News*, vol. 43, no. 3S, pp. 105–117, Jan. 2016.
[106] H. Li, T. F. Wu, A. Rahimi, K.-S. Li, M. Rusch, C.-H. Lin, J.-L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, W.-C. Chiu, M.-C. Chen, T.-T. Wu, J.-M. Shieh, W.-K. Yeh, J. M. Rabaey, S. Mitra, and H.-S.-P. Wong, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *IEDM Tech. Dig.*, Dec. 2016, pp. 1–16.

TAHA SHAHROODI received the B.Sc. degree in computer engineering from the Sharif University of Technology (SUT), Tehran, Iran, in 2018, and the M.Sc. degree in computer science from ETH Zürich, Zürich, Switzerland, in 2020. He is currently pursuing the Ph.D. degree with the Delft University of Technology (TU Delft), The Netherlands. His current research interests include bioinformatics, computer architecture, and hardware/software co-design.

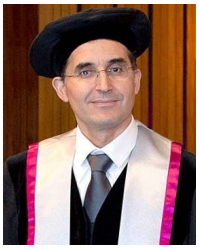
MAHDI ZAHEDI received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Delft University of Technology, Delft, The Netherlands. His current research interests include computer architecture and systems-level HW/SW co-design.

CAN FIRTINA received the B.Sc. and M.Sc. degrees in computer engineering from Bilkent University. He is currently pursuing the Ph.D. degree with the SAFARI Research Group, ETH Zürich, advised by Prof. Onur Mutlu. His current research interests include bioinformatics and computer architecture topics, including accurately and quickly identifying sequence similarities, hardware/software co-design for accelerating bioinformatics applications and genomic data analysis, correcting sequencing errors, and developing computational tools for gene editing. He is generally interested in developing new algorithms and architectures for bioinformatics applications to enable fast and accurate genome analysis.

MOHAMMED ALSER received the Ph.D. degree in computer engineering from Bilkent University, Turkey, where he was awarded the TÜBITAK doctoral fellowship. He is a Senior Researcher and a Lecturer of bioinformatics and computer architecture with D-INFK and SAFARI Research Group, D-ITET, ETH Zürich. He was previously working with ZarLaboratory, UCLA; CfAED, TU Dresden; and PETRONAS. His main research interests include bioinformatics, computational genomics, metagenomics, and computer architecture. His Ph.D. thesis, "Accelerating the Understanding of Life's Code Through Better Algorithms and Hardware Design," received the IEEE Turkey Doctoral Dissertation Award and the HiPEAC Collaboration Grant. During the last year of his Ph.D. study, he was named the Best Palestinian Ph.D. Student in Turkey.

STEPHAN WONG (Senior Member, IEEE) received the Ph.D. degree from the Delft University of Technology, The Netherlands, in December 2002. He is currently an Associate Professor with the Delft University of Technology, The Netherlands. His Ph.D. thesis entitled "Microcoded Reconfigurable Embedded Processor" describes the MOLEN polymorphic processor, organization, and (micro-)architecture. His research interests include reconfigurable computing, distributed collaborative computing, high-performance computing, embedded systems, and hardware/software co-design.

ONUR MUTLU (Fellow, IEEE) received the B.S. degree in computer engineering and psychology from the University of Michigan, Ann Arbor, and the M.S. and Ph.D. degrees in ECE from The University of Texas at Austin. He is currently a Professor of computer science with ETH Zürich. He is also a Faculty Member with Carnegie Mellon University, where he previously held the Strecker Early Career Professorship. He started the Computer Architecture Group at Microsoft Research, from 2006 to 2009; and held various product and research positions at Intel Corporation, Advanced Micro Devices, VMware, and Google. His current research interests include computer architecture, systems, hardware security, and bioinformatics. He, along with his group and collaborators, has invented a variety of techniques over the years have influenced the industry and have been employed in commercial microprocessors and memory/storage systems. He received the IEEE High Performance Computer Architecture Test of Time Award, the IEEE Computer Society Edward J. McCluskey Technical Achievement Award, the ACM SIGARCH Maurice Wilkes Award, the Inaugural IEEE Computer Society Young Computer Architect Award, the Inaugural Intel Early Career Faculty Award, the U.S. National Science Foundation CAREER Award, the Carnegie Mellon University Ladd Research Award, the faculty partnership awards from various companies, and a healthy number of best paper or ''top pick'' paper recognitions at various computer systems, architecture, and security venues. He is a fellow of ACM and an Elected Member of the Academy of Europe (Academia Europaea). His computer architecture and digital logic design course lectures and materials are freely available on YouTube (https://www.youtube.com/OnurMutluLectures) and his research group makes a wide variety of software and hardware artifacts freely available online (https://safari.ethz.ch/). For more information, please see his webpage (https://people.inf.ethz.ch/omutlu/).



SAID HAMDIOUI (Senior Member, IEEE) received the M.S.E.E. and Ph.D. degrees (Hons.) from the Delft University of Technology (TU Delft), The Netherlands. He is currently a Chair Professor of dependable and emerging computer technologies and the Head of the Quantum and Computer Engineering Department. He also serves as the Head for the Computer Engineering Laboratory (CE-Lab), TU Delft. He is the Co-Founder and the CEO of Cognitive-IC, a start-up focusing on hardware dependability solutions. Prior to joining TU Delft as a Professor, he spent about seven years within the industry, including the Microprocessor Products Group with Intel Corporation, Santa Clara, CA, USA; the IP and Yield Group with Philips Semiconductors Research and Development, Crolles, France; and the DSP Design Group with Philips/NXP Semiconductors, Nijmegen, The Netherlands. He is currently involved in different national and EU projects. He owns two patents. He has published one book and contributed to the other two. He has coauthored over 200 conference and journal articles. He has consulted for many companies (such as Intel, ST, Altera, Atmel, and Renesas) regarding in-memory testing. He has collaborated with many industry/research partners in dependable nano-computing and emerging technologies. His research focuses on two domains: dependable CMOS nano-computing (including testability, reliability, and hardware security) and emerging technologies and computing paradigms (including memristors for logic and storage and in-memory-computing for big-data applications). He is strongly involved in the international community as a member of organizing committees or technical program committees of the leading conferences. He is also a member of the Association for European NanoElectronics Activities (AENEAS)/ENIAC Scientific Committee Council. He serves on the Editorial Board for *IEEE Design and Test*, *Microelectronics Reliability* (Elsevier), and the *Journal of Electronic Testing: Theory and Applications*. He delivered dozens of keynote speeches, distinguished lectures, invited presentations, and tutorials at major international forums/conferences/schools and leading semiconductor companies. He is an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.