2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO) | 978-1-6654-6272-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/MICRO56248.2022.00057

BEACON: Scalable Near-Data-Processing Accelerators for Genome Analysis near Memory Pool with the CXL Support

Wenqin Huangfu
University of California, Santa Barbara
wenqin_huangfu@ucsb.edu

Krishna T. Malladi
Samsung Semiconductor, Inc.
k.tej@samsung.com

Andrew Chang
Samsung Semiconductor, Inc.
andrew.c1@samsung.com

Yuan Xie
University of California, Santa Barbara
yuanxie@ece.ucsb.edu

***Abstract*—Genome analysis benefits precise medical care, wildlife conservation, pandemic treatment (e.g., COVID-19), and so on. Unfortunately, in genome analysis, the speed of data processing lags far behind the speed of data generation. Thus, hardware acceleration turns out to be necessary.**

As many applications in genome analysis are memory-bound, Processing-In-Memory (PIM) and Near-Data-Processing (NDP) solutions have been explored to tackle this problem. In particular, the Dual-Inline-Memory-Module (DIMM) based designs are very promising due to their non-invasive feature to the cost-sensitive DRAM dies. However, they have two critical limitations, i.e., performance bottle-necked by communication and the limited potential for memory expansion.

In this paper, we address these two limitations by designing novel DIMM based accelerators located near the dis-aggregated memory pool with the support from the Compute Express Link (CXL), aiming to leverage the abundant memory within the memory pool and the high communication bandwidth provided by CXL. We propose BEACON, Scalable Near-Data-Processing Accelerators for Genome Analysis near Memory Pool with the CXL Support. BEACON adopts a software-hardware co-design approach to tackle the above two limitations. The BEACON architecture builds the foundation for efficient communication and memory expansion by reducing data movement and leveraging the high communication bandwidth provided by CXL. Based on the BEACON architecture, we propose a memory management framework to enable memory expansion with unmodified CXL-DIMMs and further optimize communication by improving data locality. We also propose algorithm-specific optimizations to further boost the performance of BEACON. In addition, BEACON provides two design choices, i.e., BEACON-D and BEACON-S. BEACON-D and BEACON-S perform the computation within the enhanced CXL-DIMMs and enhanced CXL-Switches, respectively. Experimental results show that compared with state-of-the-art DIMM based NDP accelerators, on average, BEACON-D and BEACON-S improve the performance by 4.70x and 4.13x, respectively.

***Keywords*-genome analysis; near-data-processing; software-hardware co-design; accelerator; memory dis-aggregation**

I. Introduction

Genome analysis is getting more and more attention, due to its important usage in evolutionary studies [20], wildlife conservation [51], precise medical care [30], and so on. In fact, genome analysis is closely related to people's health and daily lives. For example, genome analysis is useful in understanding and designing optimal drug cocktail for cancer-causing mutations [10]. In addition, genome analysis also helps a lot in dealing with the Coronavirus Disease 2019 (COVID-19) [27], [50]. However, with the rapid development of the Next Generation Sequencing (NGS) technology [61] and the large amount of sequencing data required for precise medicine [55], the growth rate of genome data becomes much faster than the Moore's law [29], putting forward great challenges for genome analysis [30].

Due to the importance and time-consuming fact of genome analysis, researchers are paying more and more attention to its hardware acceleration. Because of the large amount of data involved, the simple arithmetic operations, and the memory-bound feature, many applications in genome analysis are well-suited for Processing-In-Memory (PIM) and Near-Data-Processing (NDP) [4]. Different PIM and NDP approaches, including ReRAM [28], [36], HMC [52], [67], and Dual-Inline-Memory-Module (DIMM) [29], [30], have been explored to accelerate the applications in genome analysis. Among these PIM and NDP work, the DIMM based designs stand out to be highly promising, because they are more practical and cost-effective.

However, as shown in Fig. 1, the previous DIMM based accelerators are built upon the DDR-DIMMs attached to the host and rely on the DDR memory channel for the inter-DIMM communication. These DDR-DIMM based designs have two critical limitations: First, communication has become the performance bottleneck. As shown in Fig. 1, the large gap, e.g., 12x in MEDAL [29], between the intra-DIMM memory bandwidth and the inter-DIMM communication bandwidth seriously degrades the performance [29], [30]. Second, the potential for memory expansion is limited. The memory capacity requirements for different applications in genome analysis vary significantly, depending on the specific application, the dataset, the algorithm to use, and the parameters of the algorithm. For example, BWA-MEM uses 64GB memory for FM-index based DNA seeding [68] and SMUFIN uses near 2TB memory for k-mer counting [9]. Ideally, it's desired to be able to conveniently adjust the

978-1-6654-6272-3/22/$31.00 ©2022 IEEE
DOI 10.1109/MICRO56248.2022.00057

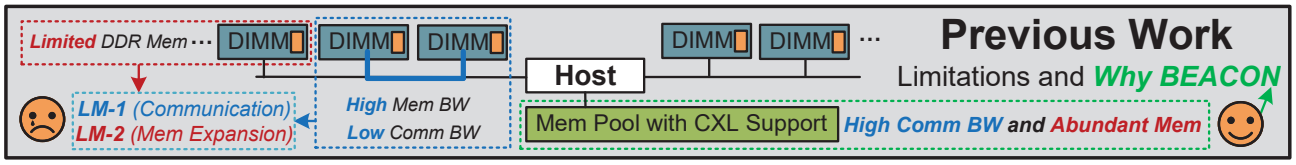

Figure 1. Architecture and limitations of the previous work, together with the motivations for designing BEACON.

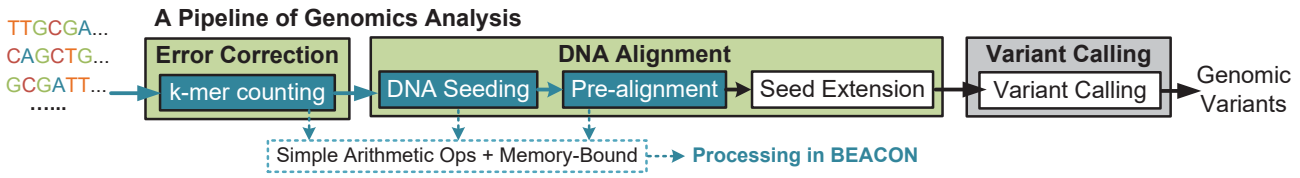

Figure 2. A pipeline of genome analysis.

memory capacity of the DIMM based accelerators with unmodified DIMMs on-demand to deal with different usage scenarios for genome analysis [29], [30]. Unfortunately, as shown in Fig. 1, the constraints of the DDR-DIMMs, e.g., the limited number of memory channels and slots [41], [46], and the trend of memory dis-aggregation, i.e., migrating local DDR memory to a memory pool, greatly reduces the potential of memory expansion in the previous work. For example, NEST provides 512GB memory in the DDR memory channel [30], which cannot meet the 2TB memory requirement for SMUFIN [9]. As a comparison, the memory pool easily provides 4.5TB memory and can scale-out far beyond this [42]. In addition, memory expansion with unmodified DIMMs totally contradicts with the design philosophy of the previous DDR-DIMM based accelerators, i.e., modifying the DIMM to reduce the expensive inter-DIMM communication and leverage the high intra-DIMM memory bandwidth with intra-DIMM data manipulation enabled by DIMM-customization.

*The **goal** of this paper is to build DIMM based accelerators for genome analysis under the scenario of memory dis-aggregation, supporting efficient on-demand memory expansion with unmodified DIMMs and eliminating the performance bottleneck of communication.* To this end, we propose BEACON, i.e., CXL-DIMM based NDP accelerators for genome analysis located near the dis-aggregated memory pool with the CXL support. BEACON enables four key features: First, it embraces the emerging trend-of memory dis-aggregation [6], [7], [57] with CXL as one of the most promising enabler [41], [47], [48]. Second, BEACON leverages the abundant memory in the memory pool to enable on-demand memory expansion with unmodified CXL-DIMMs. Third, BEACON fully leverages the high communication bandwidth provided by CXL to address the challenge of communication. Fourth, BEACON maintains the promising and non-invasive feature, i.e., no modifications to the cost-sensitive DRAM dies, of the previous DIMM based designs. BEACON provides two design choices, i.e., BEACON-D and BEACON-S. BEACON-D and BEACON-S perform the computation within the enhanced CXL-DIMMs and enhanced CXL-Switches, respectively.

To achieve the design goal, BEACON adopts a software-hardware co-design approach. Specifically, the architecture of BEACON is designed to set the foundation for efficient memory expansion and communication by reducing data movement and leveraging the high bandwidth provided by CXL. Based on the BEACON architecture, the memory management framework is proposed to enable memory expansion with unmodified CXL-DIMMs and optimize communication by improving data locality. In addition, algorithm-specific optimizations are proposed to further boost the performance of BEACON.

The main contributions of this paper are listed as follows:

- We propose BEACON, including BEACON-D and BEACON-S, under the scenario of memory dis-aggregation. Located near the memory pool and focusing on genome analysis, BEACON leverages the abundant memory within the memory pool and the high communication bandwidth provided by CXL without making any modification to the cost-sensitive DRAM dies.
- With the proposed architecture design (e.g., processing in the memory pool and in-switch data routing) and memory management framework (e.g., memory allocation and address mapping), BEACON enables efficient on-demand memory expansion with unmodified CXL-DIMMs and eliminates the performance bottleneck of communication.
- In addition, BEACON can be used for multiple applications in genome analysis. For different algorithms, we also adopt algorithm-specific optimizations (multi-chip coalescing for FM-index based DNA seeding and single-pass k-mer counting) to further improve the performance.
- Experiments are performed to demonstrate the performance benefits of BEACON and different optimizations. Overall, compared with state-of-the-art DIMM based NDP accelerators, BEACON-D and BEACON-S improves the performance by 4.70x and 4.13 on average.

II. Background

This section introduces genome analysis, memory dis-aggregation, and the CXL.

Genome Analysis: Genome analysis sets up the foundation of disease understanding, precise medical care, wildlife conservation, and so on [54]. There are a few applications in a genome analysis pipeline. An example of the genome analysis pipeline is shown in Fig. 2. Most previous accelerators for genome analysis focus on one application [28], [29], [30], [36], [52], [67]. We notice that three important applications in this pipeline, i.e., DNA seeding [2], [12], [29], k-mer counting [13], [30], [60], and DNA pre-alignment [5], [34], [38], are all good candidates for NDP due to the following two reasons [2], [29], [60]: First, they only involve very simple arithmetic operations. Second, they are memory-bound and require lots of fine-grained random memory access. According to these observations, as shown in Fig. 2, instead of focusing on one application, BEACON can be used to accelerate three different applications in the pipeline.

Memory Dis-aggregation: Memory dis-aggregation include memory expansion and memory pooling [18], [31], [46]. Memory expansion refers to the process of enlarging the total memory capacity to improve the memory capacity and bandwidth per core [18], [41]. Memory pooling addresses the issue of resource fragmentation, i.e., resource under-utilization due to the mismatched requirements of different workloads [57], by providing a shared memory pool that can be accessed on-demand for different servers. Memory dis-aggregation is becoming a trend [6], [7], [57], because it can improve system performance and resource utilization as well as reduce cost. Many different types of interconnect technologies have been proposed for memory dis-aggregation, including OpenCAPI [17], GenZ [16], FMC Cable [7], Optical memory channel [1], CXL [47], [48], and so on.

Compute Express Link (CXL): As an open industry standard interconnect, CXL offers high-bandwidth and low-latency connectivity between the host processor and devices such as smart I/O devices, accelerators, and memory buffers [66]. CXL enables cache coherence and memory semantics for heterogeneous processing and memory systems based on the PCI Express (PCIe) 5.0 I/O semantics for optimized performance in evolving usage models [47], [48]. CXL supports memory dis-aggregation, including memory expansion and memory pooling [18], [31], [41], [45]. In fact, CXL is one of the most promising technology in enabling memory dis-aggregation. For example, the full-stack memory dis-aggregation design based on CXL has been proposed [45] and the Intel CPU Memory Management Unit (MMU) are going to integrate the CXL Memory Expander in the future [41].

The architecture of the naïve memory pool with the CXL support is: The host is connected to multiple CXL-Switches via the CXL buses and each CXL-Switch is connected with multiple CXL-DIMMs via the CXL buses.

III. Motivations

This section introduces the limitations of the previous work and the reasons for our design choice.

Limitation-1 (Communication): The large gap between the intra-DIMM memory bandwidth and the inter-DIMM communication bandwidth makes inter-DIMM communication the performance bottleneck for the previous DDR-DIMM based accelerators. For example, by leveraging the rank-level memory bandwidth within the DDR-DIMMs, the intra-DIMM memory bandwidth is 12x higher than the inter-DIMM communication bandwidth in MEDAL [29].

According to the experiments, as shown in Fig. 3, with imaginary idealized communication (infinite bandwidth and zero latency, i.e., instant data delivery), on average, performance improvement of 4.36x and energy efficiency improvement of 2.32x can be achieved for the the previous DDR-DIMM based accelerators. The data indicate that the bottleneck of communication has seriously degraded the performance and energy efficiency for the previous work.

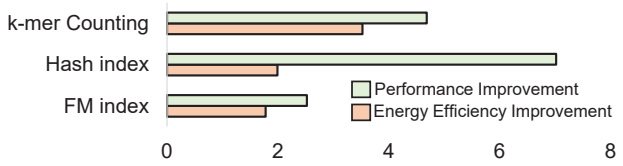

Figure 3. Improvement of performance and energy efficiency for the previous DDR-DIMM based accelerators with imaginary idealized communication, i.e., infinite bandwidth and zero latency. The experimental configuration is in Section. VI-A.

Limitation-2 (Memory Expansion): As we've mentioned in Section I, the ability of memory expansion with unmodified DIMMs is desired for accelerators of genome analysis to deal with different usage scenarios. Unfortunately, the potential of memory expansion in the previous DIMM based accelerators is greatly limited due to three reasons: First, the previous work are based on the DDR-DIMMs, which have poor scalability due to many constraints, e.g., the limited number of memory channels and slots [41], [46]. Second, memory dis-aggregation, which further reduces the amount of local DDR memory, is becoming a trend [7], [57]. Third, since the inter-DIMM communication is the performance bottleneck, the design philosophy of the previous work is to reduce the expensive inter-DIMM communication and leverage the high intra-DIMM bandwidth by intra-DIMM data manipulation enabled with DIMM-customization. Leveraging unmodified DIMMs (i.e., no intra-DIMM data manipulation) as memory expansion means frequently accessing data from the remote unmodified DIMMs (i.e., no intra-DIMM bandwidth) and bring the data back (i.e., frequent inter-DIMM communication), which totally contradicts with their design philosophy mentioned above.

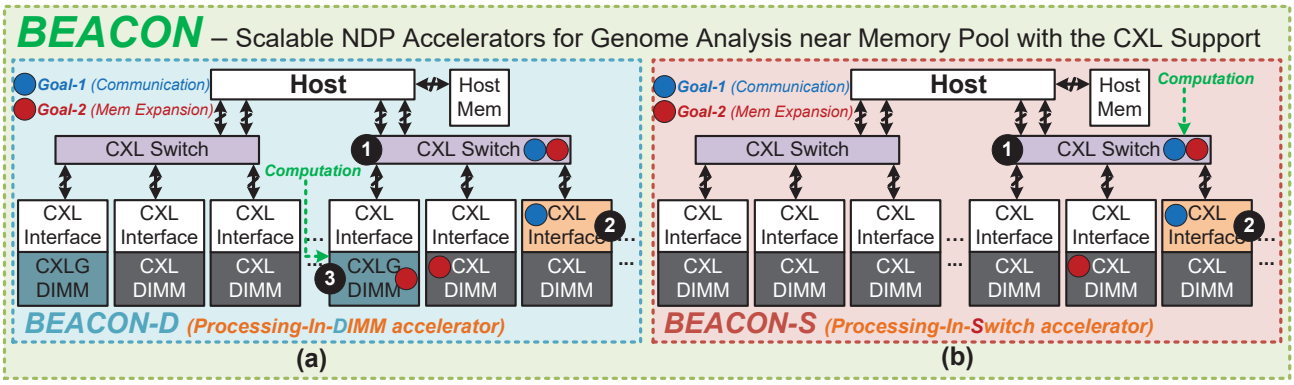

Figure 4. High-level architecture of BEACON. (a) BEACON-D. (b) BEACON-S.

To tackle these two limitations and maintain the non-invasive feature of the previous DIMM based designs, we embrace the trend of memory dis-aggregation to design novel DIMM based accelerators for genome analysis. Located near the memory pool with the CXL support, BEACON fully leverages the abundant memory in the memory pool and the high communication bandwidth of CXL to address the challenges related to memory expansion and communication.

IV. BEACON Architecture

This section presents the BEACON architecture. After an overview of the architecture, we describe the components design, the memory management framework, and the algorithm-specific optimizations in order.

A. Architecture Overview

The goal of BEACON is to build DIMM based accelerators for genome analysis under the scenario of memory dis-aggregation, supporting efficient on-demand memory expansion with unmodified DIMMs and eliminating the performance bottleneck of communication. To this end, BEACON resides in the memory pool with the CXL support to leverage the abundant memory within the memory pool and the high communication bandwidth provided by CXL.

As shown in Fig. 4, BEACON provides two design choices, i.e., BEACON-D and and BEACON-S. The key difference between BEACON-D and BEACON-S is: BEACON-D is a Processing-In-*D*IMM accelerator and performs the computation within the enhanced CXL-DIMMs. As a comparison, BEACON-S is a Processing-In-*S*witch accelerator and performs the computation within the enhanced CXL-Switches. Both BEACON-D and BEACON-S maintain the non-invasive feature and practicality of the previous DIMM based accelerators for genome analysis without making any modification to the cost-sensitive DRAM dies.

BEACON-D have better performance than BEACON-S in some applications with its customized CXL-Genome-DIMM (CXLG-DIMM). The CXLG-DIMMs are computation and fine-grained memory access enabled CXL-DIMMs. The CXLG-DIMMs have the ability to provide fine-grained memory access, which is useful in genome analysis [29], [30]. On the other hand, BEACON-S provides overall comparable performance, i.e., 87.87% performance of BEACON-D, with less hardware modifications, i.e., no modification to any of the CXL-DIMMs.

BEACON-D: For BEACON-D, as shown in Fig. 4 (a), three components are modified in the naïve memory pool with the CXL support, i.e., the CXL-Switch (❶), the CXL-Interface (❷), and the CXLG-DIMM (❸). Modifications to the CXLG-DIMMs are done to the PCB board of the CXL-DIMMs, leaving the cost-sensitive DRAM dies untouched.

From the architecture perspective, besides performing computation within the CXLG-DIMMs, keep our design goal in mind, as shown in Fig. 4 (a), the components labeled with the blue and red circles aim to achieve the goal of supporting efficient communication and providing efficient memory access, respectively. BEACON-D achieves efficient communication by transferring data via the high-bandwidth CXL buses with the architecture support from the CXL-Switches and the CXL-Interfaces. For memory expansion, as shown in Fig. 4 (a), different CXLG-DIMMs can access data within each other and can also access data in the unmodified CXL-DIMMs. The CXL-Switches help with the memory management and communication coordination.

BEACON-S: For BEACON-S, as shown in Fig. 4 (b), the CXL-Switch (❶) and the CXL-Interface (❷) are modified in the naïve memory pool with the CXL support.

In addition to performing computation within the CXL-Switches, similar to BEACON-D, as shown in Fig. 4 (b), the components labeled with the blue and red circles aim to achieve the goal of supporting efficient communication and providing efficient memory access, respectively. To achieve our design goal, the CXL-Switches and the CXL-Interfaces work together to fully leverage the potential from the high-bandwidth CXL buses to eliminate the performance bottleneck of communication. For memory expansion, the unmodified CXL-DIMMs are regulated and can be accessed by the CXL-Switches efficiently.

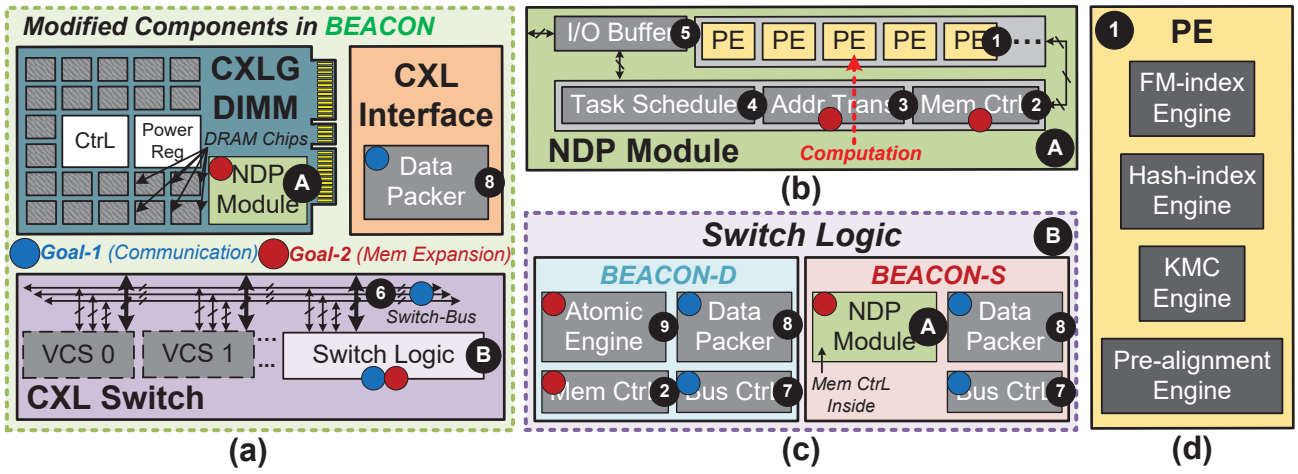

Figure 5. (a) Architecture of the three modified components within BEACON. (b) Architecture of the NDP module. (c) Architecture of the Switch-Logic in BEACON-D and BEACON-S. (d) Architecture of the PE.

B. Components Design

Architectures of the modified components in the CXL based memory pool, i.e., the CXLG-DIMM, the CXL-Interface, and the CXL-Switch, are shown in Fig. 5 (a). Similarly, the components labeled with the blue and red circles aim to achieve the goal of supporting efficient communication and providing efficient memory access. As mentioned before and shown in Fig. 4 (a), the functionality of the CXLG-DIMMs is to perform computation and access memory efficiently. The CXL-Interfaces help to fully leverage the potential of the high-bandwidth communication provided by CXL. As for the CXL-Switches, they are responsible for memory management and communication coordination. In BEACON-S, the CXL-Switches are also responsible for performing computation.

*1) **CXLG-DIMM**:* As shown in Fig. 5 (a), the NDP module (Ⓐ) is added to the CXL-DIMM to build the CXLG-DIMM. In addition, similar to MEDAL [29], in order to improve utilization of memory bandwidth for genome analysis, fine-grained memory access is enabled in the CXLG-DIMMs by providing individual Chip Select (CS) signals to different DRAM chips. To summarize, the CXLG-DIMMs are computation and fine-grained memory access enabled CXL-DIMMs. The CXLG-DIMMs are the accelerator modules in BEACON-D.

NDP Module Design: The design goal of the NDP module is to enable computation for multiple applications in genome analysis and provide efficient memory accessibility to the other CXLG-DIMMs and CXL-DIMMs. Five different components, i.e., the PE, the Memory Controller (MC), the Address Translator, the Task Scheduler, and the I/O Buffer, are added to the NDP module for the design goal.

- ***To Enable Computation:*** To enable computation for multiple applications in genome analysis, multiple PEs are added into the NDP module:

❶ ***PEs:*** As shown in Fig. 5 (b) and (d), the PEs in BEACON are designed to be multi-purpose. They are ASIC logic with pre-determined and fixed functions for our target applications in genome analysis. Support simple integer addition and hash calculation required for the target applications, these PEs contain a FM-index engine, a Hash-index engine, a KMC engine, and a DNA pre-alignment engine.

As for the inputs, tasks, i.e., DNA sequences to be processed with related information, are received from the Task Scheduler. As for the outputs, memory requests are sent to the Address Translator to derive the physical memory addresses. If operands from memory requests are needed for an active task to continue computation, the PE puts that task into the Task Scheduler and switches to process another task in its task queue.

- ***To Provide Efficient Memory Access:*** The Memory Controller (MC) and the Address Translator are added into the NDP module to support the proposed memory management and eliminate the need for help from the host-side memory MC during memory access:

❷ ***Memory Controller (MC):*** The MCs in the NDP module, as shown in Fig. 5 (b), are responsible for the purpose of memory management in BEACON, including memory allocation/de-allocation, data placement, address mapping, memory requests scheduling, and so on. In addition, the MCs in BEACON also enable localized memory access. For example, in BEACON-D, the MCs in the NDP modules on the CXLG-DIMMs enable localized memory access inside the CXLG-DIMMs. Similary, in BEACON-S, the MCs in the NDP modules on the CXL-Switches enable localized memory management without asking for help from the host-side MC.

❸ ***Address Translator:*** The Address Translators, as shown in Fig. 5 (b), receive memory requests from the PEs and translate these memory requests into their physical memory addresses according to the proposed data placement and address mapping schemes in BEACON, which are described

in detail in Section. IV-C. Next, the Address Translators forward these memory requests towards their destinations.

• ***Other Components:*** Besides the components related to computation and memory access, the Task Scheduler and the I/O Buffer are added into the NDP module to support its functionalities and improve the efficiency:

❹ ***Task Scheduler:*** We define a task as a DNA sequence to be processed with related information, e.g, algorithm and current processing status. The Task Schedulers, as shown in Fig. 5 (b), maintain two queues for the inactive tasks, i.e., the incoming task queue and the out-going task queue.

The inputs to the incoming task queue are tasks waiting for operands from memory requests and are sent from the PEs. When all the operands needed for a task within the incoming task queue are ready, this task is pushed to the out-going task queue. The Task Schedulers monitor the statuses of different PEs and the tasks in the out-going task queue are assigned to the PEs that need more tasks to process.

❺ ***I/O Buffer:*** The Input Buffers, as shown in Fig. 5 (b), receive inputs to the NDP modules, including the remote memory requests and the data back with remote/local destinations. Both the remote memory requests and the data back with remote/local destinations are first forwarded to the MCs. The remote memory requests wait at the MCs to be issued out. The data back with remote destinations are forwarded to the Data Packers (introduced later with the CXL-Switch) to be packed together and transferred towards the remote destinations. The data back with local destinations are forwarded to the Task Schedulers, which means the needed operands for the tasks within the incoming task queues in the Task Schedulers have been back.

For BEACON-D, the Output Buffers in the CXLG-DIMMs, as shown in Fig. 5 (b), forward memory requests to the Data Packers within the CXL-Interfaces. For BEACON-S, the Output Buffers in the CXL-Switches forward memory requests to the Data Packers within the CXL-Switches. These memory requests are forwarded towards their destination from the Data Packers.

2) ***CXL-Switch****:* To support the functionalities of the CXL-Switch, as shown in Fig. 5 (a), the Switch-Bus and the Switch-Logic (Ⓑ) are added to the CXL-Switches.

❻ ***Switch-Bus:*** To achieve our design goal of supporting efficient communication, as shown in Fig. 5 (a), the Switch-Bus is added to the CXL-Switches to support efficient in-switch data routing between different components within the same CXL-Switch, e.g, the Virtual CXL Switch (VCS) and the Switch-Logic, eliminating unnecessary data movement between the CXL-Switches and the host.

Switch-Logic Design: For BEACON-D, the design goal of the Switch-Logic is to support efficient communication and provide efficient memory accessibility to the CXL-DIMMs. To achieve the goal, four components, i.e., the Bus Controller (Bus CtrL), the Data Packer, the MC, and the Atomic Engine, are added into the Switch-Logic for BEACON-D.

For BEACON-S, since the computation is performed within the CXL-Switches, besides the design goal above, we also need to enable computation within the Switch-Logic. Overall, three components, i.e., the NDP module, the Bus CtrL, and the Data Packer, are added into the Switch-Logic.

• ***To Enable Computation:*** BEACON-S needs the computation capability in its Switch-Logic, we add the same NDP module placed within the CXLG-DIMMs in BEACON-D into the Switch-Logic in BEACON-S to achieve this goal.

• ***To Support Efficient Communication:*** The Bus CtrL and the Data Packer are added into the Switch-Logic to efficiently coordinate in-switch data routing and improve the utilization of communication bandwidth for fine-grained data:

❼ ***Bus CtrL:*** As shown in Fig. 5 (c) and (a), the Bus CtrLs are responsible for the communication coordination and in-switch data routing within the CXL-Switches. The Bus CtrLs manage and control the traffic on the Switch-Bus.

❽ ***Data Packer:*** Applications in genome analysis involve frequent fine-grained random memory access, e.g., 32 Bytes for DNA seeding and 1 bit for k-mer counting [29], [30]. However, the default data transfer granularity in CXL is 64 Bytes. Transferring data with the default data transfer scheme in CXL leads to movement of useless data, wasting communication bandwidth and energy.

The key idea to address this issue in BEACON is to discard the useless data and pack the useful data together before the data transfer. After receiving the data, the packed fine-grained data are unpacked and separated. This approach eliminates the movement of useless data, leading to reduction in communication bandwidth and energy consumption. In Fig. 6 (a) and (b), an example with 16 Bytes data chunk is used to demonstrate the idea of data packing. Before data packing, the data chunk only contains 2 Bytes useful data from 1 memory request, i.e, data 0. After data packing, useful data are packed together and the data chunk contains 16 Bytes useful data from 8 memory requests, i.e., data 0 to data 7, eliminating the movement of useless data.

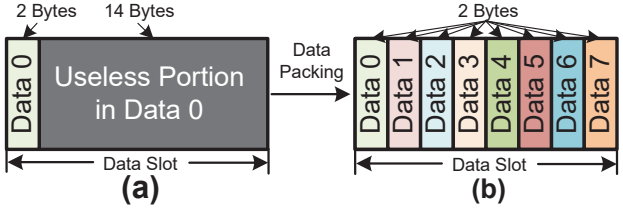

Figure 6. Data chunk to be transferred. (a) Before data packing. (b) After data packing.

The Data Packer is added into the Switch-Logic to enable data packing. The Data Packer, as shown in Fig. 5 (c), packs fine-grained data together according to the format defined by the CXL protocol before the data transfer. It also unpacks and separates the packed fine-grained data it receive.

To Provide Efficient Memory Access: To provide efficient memory access to the unmodified CXL-DIMMs connected with the same CXL-Switch, MCs are needed in the CXL-

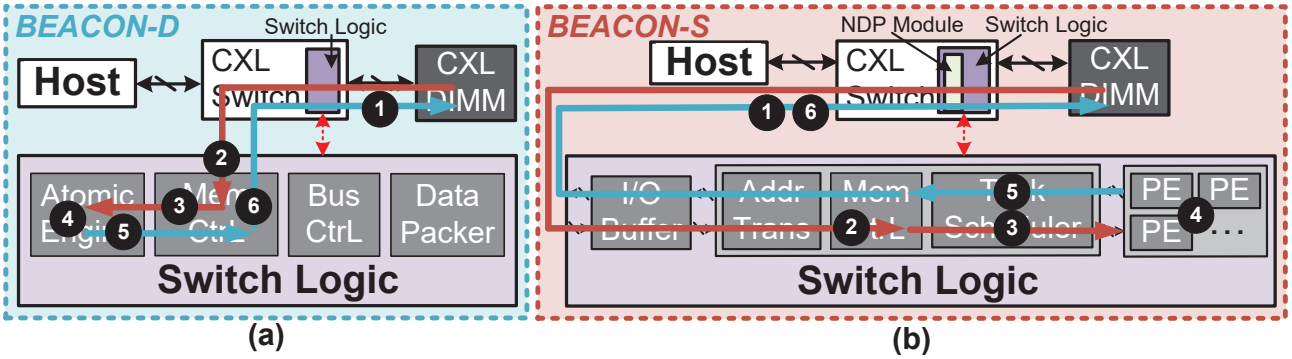

Figure 7. Workflow of performing atomic memory operations. (a) BEACON-D. (b) BEACON-S.

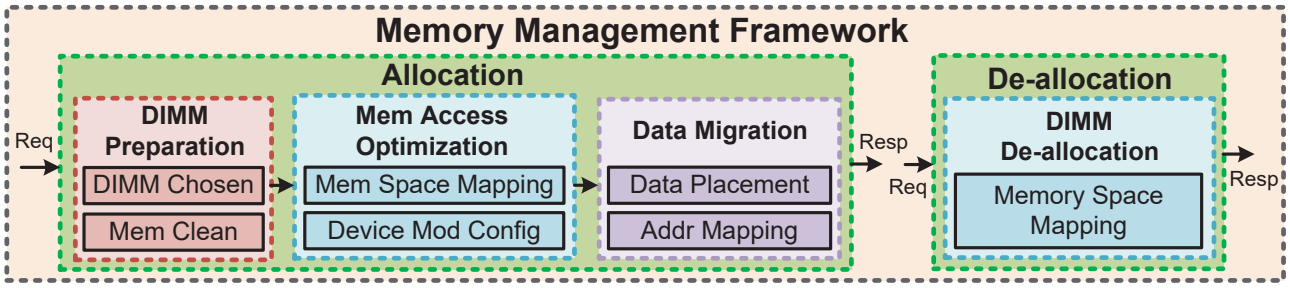

Figure 8. The workflow of the memory management framework in BEACON.

Switches. The MCs are responsible for the customized memory management, including memory allocation/de-allocation, data placement, address mapping, DRAM states maintaining, memory requests scheduling, and so on. For BEACON-D, as shown in Fig. 5 (c), the MCs (❷) are added into the Switch-Logic. For BEACON-S, as shown in Fig. 5 (c), because its NDP modules in the CXL-Switches already contain MCs, we enhance these MCs within its NDP modules and no extra components are needed.

To address the issue of Read-Modify-Write (RMW) data race during memory access, the Atomic Engines are added into the Switch-Logic in BEACON-D. For BEACON-S, because its Switch-Logic already contains many PEs, we reuse these PEs as the Atomic Engines and no extra components are needed.

❾ ***Atomic Engine:*** With parallel processing, Read-Modify-Write (RMW) data race, i.e., simultaneously reading and updating the same memory location, is a challenge. For example, during k-mer counting, multiple tasks may try to read, increase, then write back the same k-mer counter simultaneously. Undetermined order of these operations may lead to incorrect final result. Atomic memory operations can solve the issue of RMW data race as well as reduce bandwidth consumption [43]. In addition, atomic memory operations are useful for many applications [43]. Due to these reasons, we enable atomic memory operations in BEACON to address the issue of RMW data race.

In BEACON-D and BEACON-S, the arithmetic part of the atomic memory operations are performed in the Atomic Engines, shown in Fig. 5 (c), and the PEs within the Switch-Logic, shown in Fig. 5 (b), respectively. The workflows of performing atomic memory operations in BEACON are described below:

For BEACON-D, as shown in Fig. 7 (a), ❶ The MC in the Switch-Logic issues memory request to the target CXL-DIMM. ❷ The data is brought back to the MC. ❸ The data is forwarded to the Atomic Engine. ❹ The arithmetic operations are performed within the Atomic Engine. ❺ The arithmetic result is sent back to the MC. ❻ The MC issues memory request to write back the final result.

The similar workflow for BEACON-S is shown in Fig. 7 (b). The difference between the workflows of BEACON-S and BEACON-D is: For BEACON-D, the arithmetic operations are performed in the Atomic Engines. For BEACON-S, the arithmetic operations are performed in the PEs within the Switch-Logic.

3) ***CXL-Interface:*** To reduce the movement of useless data and improve utilization of the communication bandwidth, as shown in Fig. 5 (c), the Data Packers (❽) described above are also added into the CXL-Interfaces to perform data packing/unpacking before/after the data transfer towards/from the CXL-Switches.

C. Memory Management Framework

The workflow of the proposed memory management framework is shown in Fig. 8. The host coordinates with the CXL-Switches, the unmodified CXL-DIMMs, and the CXLG-DIMMs (if BEACON-D) to perform memory management.

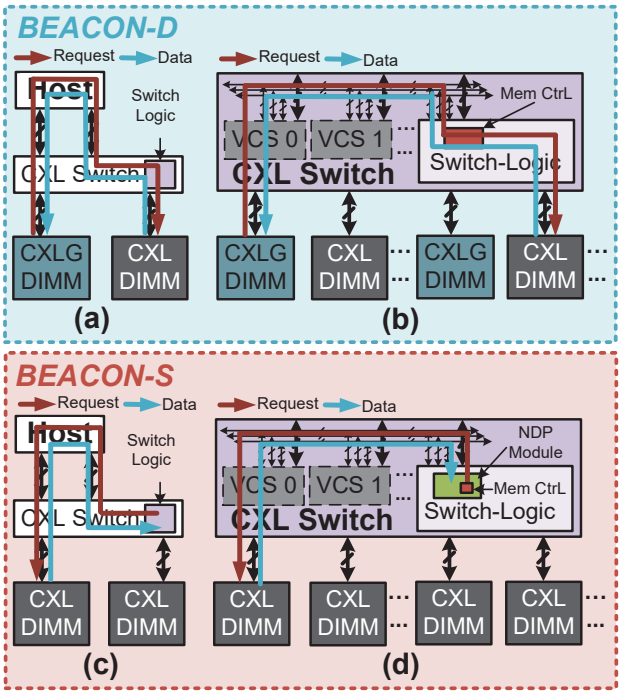

Figure 9. (a), (b) Memory access flows before/after the proposed optimizations for BEACON-D. (c), (d) Memory access flows before/after the proposed optimizations for BEACON-S.

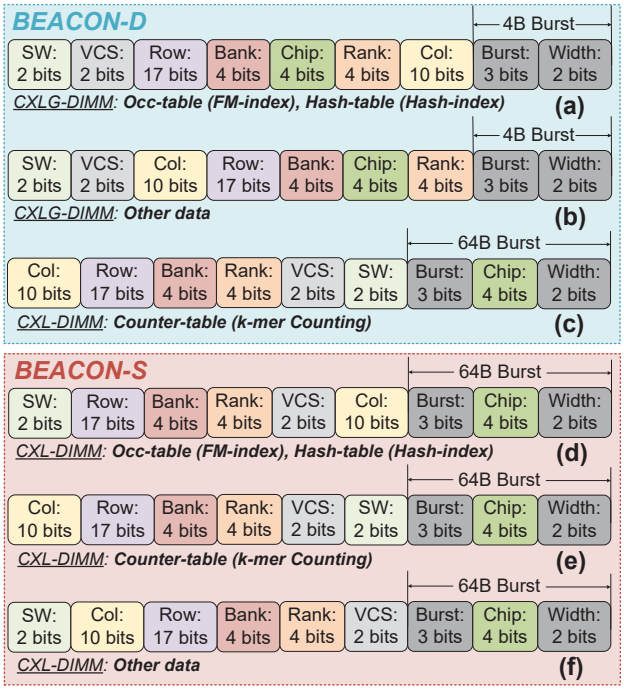

Figure 10. Architecture and data aware address mapping. (a), (b), and (c) are designed for BEACON-D. (d), (e), and (f) are designed for BEACON-S.

Memory Allocation: First, the host sends the memory allocation request with the detailed information, e.g, application, algorithm, dataset, parameters, to the CXL-Switch es via the framework interface. Second, as shown Fig. 8, the CXL-Switches work with the host, the unmodified CXL-DIMMs, and the CXLG-DIMMs (if BEACON-D) to do memory allocation, including DIMM allocation, memory access optimization, and data migration. Third, the CXL-Switches send the response, i.e., successful or failed, back to the host via the framework interface. The details in the second step are described below:

• ***DIMM Allocation:*** The proposed memory management framework manages memory in the granularity of CXL-DIMM. According to the memory allocation request, the memory framework tries to allocate the unmodified CXL-DIMMs in proximity to the NDP modules, e.g., within the same CXL-Switch. After deciding which CXL-DIMMs to allocate, memory clean is performed to migrate the active data for other applications within these CXL-DIMMs to the other available CXL-DIMMs. The host and the CXL-Switches work together to update the related page tables during the memory clean process. After the memory clean, these chosen CXL-DIMMs are used dedicated for BEACON. The memory address within these CXL-DIMMs are marked as non-cacheable for the host.

• ***Memory Access Optimization:*** Because CXL enables cache coherence, with the naïve implementation, as shown in Fig. 9 (a) and (c), the memory requests and data from/to the unmodified CXL-DIMMs need to go through the host for coherence purpose, leading to redundant data movement.

Leveraging the protocol of CXL [48], the memory space in the unmodified CXL-DIMMs is mapped to the device memory space in BEACON and BEACON is set to the device-biased mod. With the software configuration and the architecture support from the MC and the Switch-Bus inside the Switch-Logic, the memory access flows to the unmodified CXL-DIMMs in BEACON-D and BEACON-S are show in Fig. 9 (b) and (d), respectively. The redundant data movement is eliminated.

• ***Data Migration:*** To reduce data movement, the key idea of the proposed data placement scheme is to make full utilization of the data locality. BEACON always tries to put the more frequently accessed data to memory locations in proximity to the NDP modules within BEACON.

Different from the previous work [22], [29], [30], which provides a fixed address mapping scheme, we propose the architecture and data aware address mapping scheme to better leverage features of both the architecture and the specific data. The two architecture and data related principles in the proposed address mapping scheme are:

1. Memory address interleaving is performed according to the architecture of the DIMMs to fully leverage the available memory bandwidth. For the CXLG-DIMMs, the memory address interleaving can be done at the chip-level to leverage its ability to perform fine-grained memory access, while the memory address interleaving can only be done at the rank-level for the unmodified CXL-DIMMs.

2. For data with spacial locality [29], we prioritize to map these data in the DRAM row-by-row. As an example, in Hash-index based DNA seeding, multiple matching locations for a seed are stored continuously within the same DRAM row to fully leverage row-level locality.

Following the above two principles, the detailed address mapping schemes in BEACON are shown in Fig. 10.

After the system initialization, the BEACON framework (including the memory management framework) and the customized MCs are already aware of the architecture information in the system, e.g., the specific types of different DIMMs. The data type information, e.g., whether a data type has spacial locality or not, are also provided to the BEACON framework and the customized MCs.

During the system execution, the related information, e.g., application, algorithm, data type, are included in the memory requests to the customized MCs. The customized MCs schedule the memory accesses based on these information and the proposed address mapping schemes.

Memory De-alocation: For memory de-allocation, first, the host sends the memory de-allocation request with detailed information, e.g, how much memory to de-allocate, to the CXL-Switches via the framework interface. Second, the CXL-Switches work with the CXLG-DIMMs (if BEACON-D) to manage and map the corresponding allocated memory in the CXL-DIMMs to the host memory space. Third, the CXL-Switches send the response, i.e., successful or failed, back to the host via the framework interface.

D. Algorithm-Specific Optimizations

FM-index - Multi-Chip Coalescing: FM-index based DNA seeding requires fine-grained memory access [29]. As shown in Fig.11 (a), fine-grained memory access leads to low memory bandwidth utilization in the conventional DIMM, because all the DRAM chips are accessed in lock-step and only a portion of the data read out from the memory are useful. As shown in Fig.11 (b), the previous work supports fine-grained memory access, i.e., read a single DRAM chip multiple times to get the fine-grained data, to improve the memory bandwidth utilization [29]. Unfortunately, with this method, the amount of memory access to some DRAM chips are greatly increased, while some DRAM chips stay idle. In a word, although better than the conventional approach, the method adopted in the previous work leads to memory bandwidth under-utilization as well.

To address this issue, we propose multi-chip coalescing for data placement and memory access in the CXLG-DIMMs within BEACON. As shown in Fig.11 (c), data in the CXLG-DIMMs are placed and accessed in multi-chip granularity to achieve a sweet point, i.e., no useless data read out and better balanced memory access. The amount of DRAM chips to be coalesced and accessed together is fine-tuned to achieve the best performance.

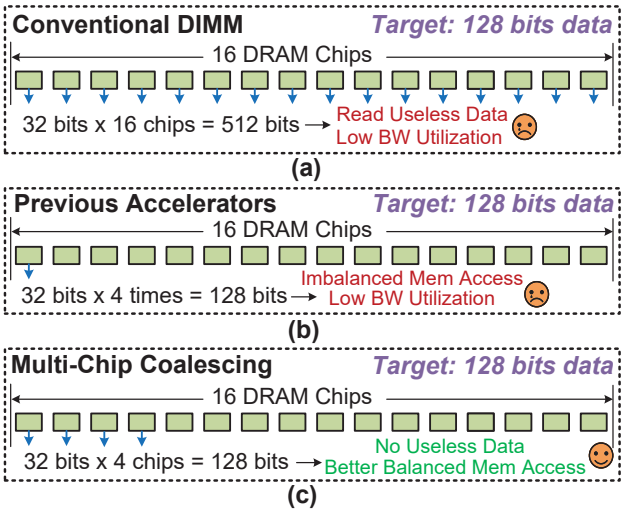

Figure 11. Examples of different approaches for fine-grained memory access. (a) Conventional DIMM. (b) The previous accelerators. (c) Multi-chip coalescing.

k-mer Counting - Single-Pass k-mer Counting: In the previous DIMM based accelerator for k-mer counting, i.e., NEST [30], a multi-pass method is adopted. Specifically, first, each DIMM constructs its own counting Bloom filter locally (processing the entire input data for the first time). Next, the counting Bloom filters from different DIMMs are merged to derive the global Bloom filter and the global Bloom filter is distributed to all the DIMMs. Finally, each DIMM performs k-mer counting independently and access its own copy of the global Bloom filter (processing the entire input data for the second time). This method eliminates the expensive random remote memory access in the first step and the last step at the cost of processing the entire input data twice.

However, BEACON-S performs the computation in the CXL-Switches and the data are distributed in different CXL-DIMMs. With the previous approach, we cannot get the benefits of data localization within a single DIMM, while we still suffer from its overhead of processing the entire input data twice. For this reason, the method of single-pass k-mer counting is adopted in BEACON-S. To be specific, BEACON-S directly deals with the global Bloom filter distributed to different CXL-DIMMs, eliminating the first step and the last step in the previous approach.

V. Discussion

Design Overhead: From the hardware perspective, compared with the previous work, BEACON has smaller or comparable overhead on area and power. Please refer to Section. VI-A for more details. From the software perspective, the necessary software support, e.g., the Memory Management Framework, is integrated in the BEACON framework. Minimal change to the existing software is required in BEACON.

Table I
EXPERIMENTAL CONFIGURATION

Configuration of the CPU basline	
CPU Model/CPU Frequency(GHz)	Intel Xeon E5-2680 v3/2.50
Memory(GB)/L1(KB)/L2(KB)/L3(MB) Cache	384/64/256/32
Configuration of MEDAL and NEST	
Memory(GB)/DDR Channels	512/4
Customized DIMM per Channel	2
Configuration of BEACON	
Memory(GB)/Switch/VCS per Switch	512/2/2
CXLG/CXL-DIMM per VCS (BEACON-D)	1/1
CXL-DIMM per VCS (BEACON-S)	2
Configurations of DIMM	
DIMM Capacity(GB)/DRAM Chip	64/8Gb ×4
Ranks per DIMM/Chips per Rank	4/16
BankGroups per Chip/Banks per BankGroup	4/4
Clock Frequency(MHz)/tRCD-tCAS-tRP	1,600/22-22-22

Programming Burden: One of the design target for BEACON is to relieve the programming burden for the end-users, e.g., biology experts. To achieve this goal, considering that we only need to deal with a few computationally simple applications, we choose to integrate PEs with pre-determined and fixed functions into BEACON. With this design, the end-users only need to provide the related information, e.g., application, algorithm, dataset size, input task number, and task parameters, to the User-Interface (UI) of the BEACON framework. No coding and no programming are required for the end-users.

Non-Uniform Memory Access (NUMA): NUMA is an important issue for architectures with non-uniform bandwidth and latencies. With all the proposed optimizations, on average, BEACON achieves 96% performance of its corresponding design with imaginary idealized communication, indicating that NUMA isn't a problem for our target applications. Please refer to Table. II for more experimental details.

Extension to Other Applications: BEACON can be easily extended as a practical, cost-effective, and scalable accelerator for other memory-bound applications, such as image processing [26], graph processing [3], and database searching [40], by replacing the PEs within the NDP module BEACON can also be extended as general purpose NDP accelerator by replacing the PEs with light-weight, low-power, general-purpose processing units [62]. After the PE replacement, the data placement method and the address mapping scheme need to be changed accordingly to fully leverage the potential of the BEACON architecture.

Extension to Other Architectures: Besides CXL, there are other emerging communication specifications [57], such as Gen-Z [16], OpenCAPI [17], and CCIX [15], the designs and optimizations in BEACON could also be migrated to and be used in the NDP architectures with these emerging communication specifications.

VI. EXPERIMENTAL RESULTS

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

Table II
HARDWARE OVERHEAD OF PE IN DIFFERENT ARCHITECTURES

Architecture	Area (um^2)	Dynamic Power (mW)	Leakage Power (uW)
MEDAL [29]	8941.39	10.57	36.16
NEST [30]	16721.12	8.12	24.83
BEACON	14090.23	9.48	18.97

A. Experimental Setup

Configuration of the Baselines: For FM-index and Hash-index based DNA seeding, the software we use for the CPU baselines are BWA-MEM [44] and SMALT [58], respectively. The hardware baseline is the previous DIMM based accelerator for DNA seeding, i.e., MEDAL [29]. For k-mer counting, the software we use for the CPU baseline is BFCounter [53]. The hardware baseline is the previous DIMM based accelerator for k-mer counting, i.e., NEST [30]. For DNA pre-alignment, the software we use for the CPU baseline is Shouji [5]. For the hardware baselines mentioned above [29], [30], based on Ramulator [39], we implement cycle-accurate simulators for them. We have verified the simulation results against the previous work to ensure the correctness of our simulation.

The detailed configurations of these baselines are shown in Table. I. The DIMMs used in the experiments have the same parameters. The same PEs described below are used in the NDP baselines and BEACON. We've ensured that BEACON and the NDP baselines have **the same area overhead**.

We need to mention that the experiments are **in favor of the NDP baselines**. As mentioned in Section. III, the baselines cannot use unmodified DIMMs for memory expansion. In the experiments, as shown in Table. I, all the DIMMs in the NDP baselines are customized DIMMs according to their designs. These customized DIMMs provide better performance and higher energy efficiency for genome analysis. In BEACON-D, only a portion of the DIMMs are customized. In BEACON-S, none of the DIMMs are customized. Due to the hardware configuration in favor of the baselines, the previous DIMM based accelerators have slightly higher energy efficiency than BEACON. BEACON can also achieve higher energy efficiency with more customized CXLG-DIMMs, but that is not our design goal. Because it's obviously more convenient, scalable, and cost-effective to scale-out by leveraging existing unmodified CXL-DIMMs in the memory pool, instead of keep inserting more highly customized DIMMs into the computing system.

Configuration of BEACON: Ramulator [39] is modified to build a cycle-accurate simulator for BEACON. The configuration of BEACON is shown in Table. I. We use pre-layout Design Compiler [63] with 28 nm technology [64] to synthesize the timing, energy, and area of the PEs. As shown in Table. II, we have compared the area, the dynamic power, and the leakage power of the PEs in BEACON against the corresponding parameters in MEDAL and NEST [30]. The data in Table. II indicates that, compared with the previous work, the PEs in BEACON have smaller or comparable

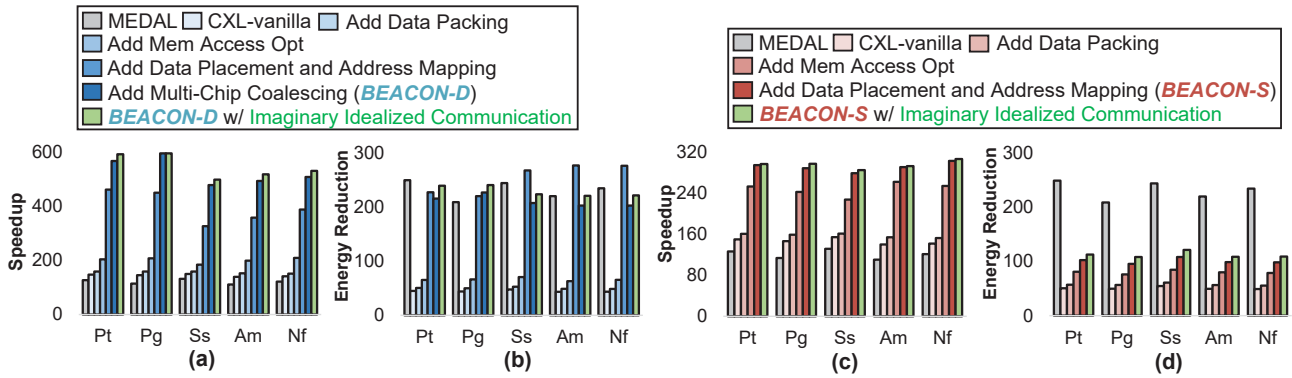

Figure 12. Experimental results for FM-index based DNA seeding. (a), (b) Performance improvement and energy reduction of BEACON-D. (c), (d) Performance improvement and energy reduction of BEACON-S.

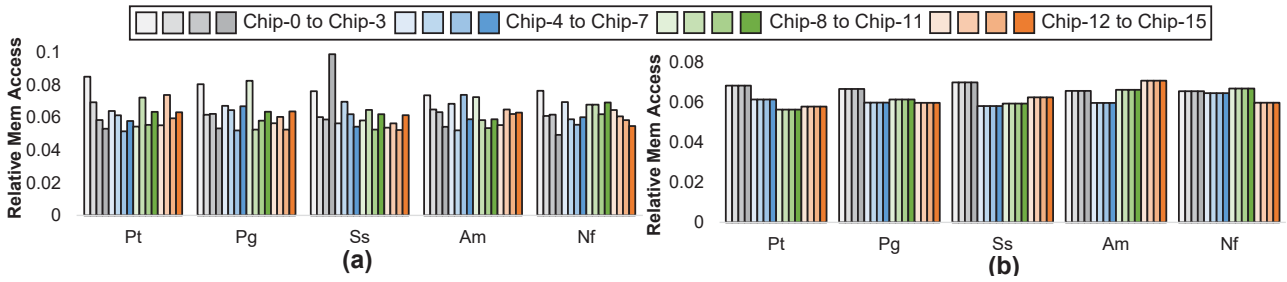

Figure 13. Normalized memory access to different DRAM chips for FM-index based DNA seeding. (a), (b) Without and with multi-chip coalescing.

overhead on area and power. Considering that the previous work focuses on a single application and BEACON deals with multiple applications, we conclude that the PEs in BEACON have reasonably low hardware overhead. The computational latencies for FM-index based DNA seeding, Hash-index based DNA seeding, k-mer counting, and DNA pre-alignment are equal to 16, 10, 59, and 82 DRAM cycles.

For BEACON-D, there are 128 PEs within each CXLG-DIMM. For BEACON-S, there are 256 PEs within each CXL-Switch. The configuration of the DIMM is shown in Table. I. The energy consumption of DRAM is calculated with DRAMPower [11]. The energy consumption for the communication come from [33], [37].

Datasets: For FM-index based DNA seeding, Hash-index based DNA seeding, and DNA pre-alignment, five different genomes, i.e., Pinus taeda (Pt), Picea glauca (Pg), Sequoia sempervirens (Ss), Ambystoma mexicanum (Am), and Neoceratodus forsteri (Nf), from NCBI [56] are used. For k-mer counting, human genome with coverage ratio of 50x is used.

Next, for different applications, the performance improvement and energy reduction of BEACON are presented. All the data are normalized to the corresponding data of the 48-thread CPU baselines.

B. FM-index based DNA Seeding

BEACON-D: For BEACON-D, the performance improvement of designs with step-by-step optimizations on FM-index based DNA seeding are shown in Fig. 12 (a). On average, CXL-vanilla, i.e., the naïve NDP accelerator near the memory pool with the CXL support, improves the performance of the 48-thread CPU baseline and MEDAL by 144.18x and 1.20x, respectively. As for the optimizations, data packing improves the performance by 1.08x. Performance improvement of 1.29x is achieved with memory access optimization. Moreover, the proposed data placement and address mapping gain performance improvement of 1.96x. In addition, multi-chip coalescing brings performance improvement of 1.34x. Overall, BEACON-D outperforms the 48-thread CPU and MEDAL by 525.73x and 4.36x in performance, respectively. BEACON-D achieves 96.52% performance of the corresponding design with idealized communication (infinite bandwidth and zero latency), indicating that communication is no longer an issue.

The performance benefits of multi-chip coalescing comes from balanced memory access to different DRAM chips and better utilization of memory bandwidth. As shown in Fig. 13 (a), without multi-chip coalescing, the memory access to different DRAM chips are unevenly distributed, leading to memory bandwidth under-utilization. As shown in Fig. 13 (b), with multi-chip coalescing, the memory access to different DRAM chips are well-balanced, i.e., with less variations, and memory bandwidth is better utilized.

For energy efficiency, as shown in Fig. 12 (b), most optimizations improve the energy efficiency of BEACON-D, except for the multi-chip coalescing. Without the multi-chip coalescing, we have more fine-grained control on data accessing, i.e., less useless data are read out, but the

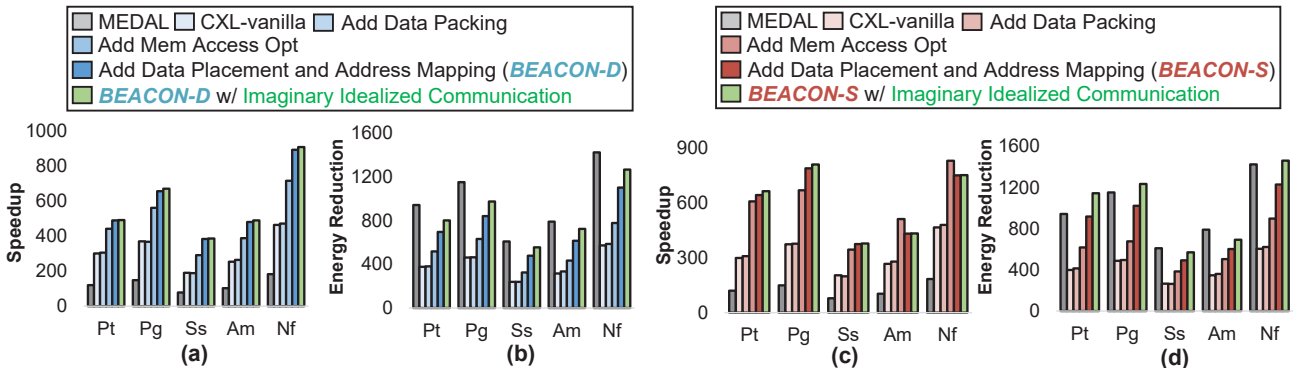

Figure 14. Experimental results for Hash-index based DNA seeding. (a), (b) Performance improvement and energy reduction of BEACON-D. (c), (d) Performance improvement and energy reduction of BEACON-S.

unbalanced memory access leads to performance degradation. With the multi-chip coalescing, well-balanced memory access improves performance. However, accessing data in relative coarse-granularity means more useless data are read out. To summarize, the multi-chip coalescing sacrifices a little bit energy efficiency to boost the performance. Overall, BEACON-D reduces energy consumption of the CPU baseline by 210.59x and achieves 91.26% energy efficiency of MEDAL. Compared with the corresponding design with idealized communication, BEACON-D achieves 92.09% of its energy efficiency.

BEACON-S: For BEACON-S, the performance improvement of designs with step-by-step optimizations on FM-index based DNA seeding are shown in Fig. 12 (c). On average, CXL-vanilla improves the performance of the 48-thread CPU baseline and MEDAL by 146.64x and 1.22x, respectively. As for the optimizations, data packing improves the performance by 1.08x. Performance improvement of 1.57x is achieved with memory access optimization. Moreover, the proposed data placement and address mapping gain performance improvement of 1.18x. Overall, BEACON-S outperforms the 48-thread CPU and MEDAL by 291.62x and 2.42x in performance, respectively. BEACON-S achieves 98.48% performance of the corresponding design with idealized communication, eliminating the issue of communication.

Energy wise, as shown in Fig. 12 (d), BEACON-S reduces energy consumption of the CPU baseline by 100.60x and achieves 43.59% energy efficiency of MEDAL. Compared with the corresponding design with idealized communication, BEACON-S achieves 82.57% of its energy efficiency. Compare Fig. 12 (b) and (d), the energy efficiency improvement of the data placement and address mapping in BEACON-D is larger than that in BEACON-S, because BEACON-D is more ”de-centralized and asymmetrical” than BEACON-S. Generally speaking, the more ”de-centralized and asymmetrical” an architecture is, the more important the data placement and address mapping are.

C. Hash-index based DNA Seeding

BEACON-D: For BEACON-D, the performance improvement of designs with step-by-step optimizations on Hash-index based DNA seeding is shown in Fig. 14 (a). On average, CXL-vanilla improves the performance of the 48-thread CPU baseline and MEDAL by 309.13x and 2.54x, respectively. As for the optimizations, performance improvement of 1.81x is achieved with memory access optimization. Data packing, data placement, and address mapping don’t bring much performance benefits, because of two reasons: First, Hash-index based DNA seeding doesn’t require many fine-grained memory access. Second, the communication is not a bottleneck in BEACON-D for this application even only with the proposed architecture support. Overall, BEACON-D outperforms the 48-thread CPU and MEDAL by 572.17x and 4.70x, respectively. BEACON-D achieves 98.59% performance of the corresponding design with idealized communication, indicating that communication is no longer an issue.

As for the energy efficiency, as shown in Fig. 14 (b), BEACON-D reduces energy consumption of the CPU baseline by 960.72x and achieves 85.58% energy efficiency of MEDAL. Compared with the corresponding design with idealized communication, BEACON-D achieves 84.05% of its energy efficiency.

BEACON-S: For BEACON-S, the performance improvement of designs with step-by-step optimizations on Hash-index based DNA seeding is shown in Fig. 14 (c). On average, CXL-vanilla improves the performance of the 48-thread CPU baseline and MEDAL by 302.48x and 2.48x, respectively. As for the optimizations, performance improvement of 1.50x is achieved with memory access optimization. Moreover, the proposed data placement and address mapping gain performance improvement of 1.21x. Data packing doesn’t bring much benefits, because the amount of fine-grained memory access in Hash-index based DNA seeding is limited. Overall, BEACON-S outperforms the 48-thread CPU and MEDAL by 556.66x and 4.57x, respectively. BEACON-S achieves 98.64% performance of the corresponding design

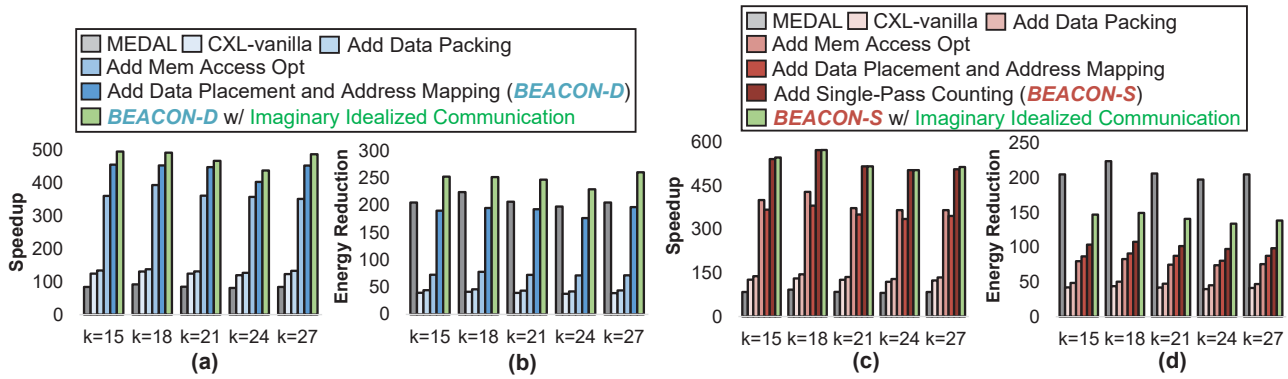

Figure 15. Experimental results for k-mer counting. (a), (b) Performance improvement and energy reduction of BEACON-D. (c), (d) Performance improvement and energy reduction of BEACON-S.

with idealized communication, eliminating the performance bottleneck of communication.

Energy wise, as shown in Fig. 14 (d), BEACON-S reduces energy consumption of the CPU baseline by 832.32x and achieves 76.11% energy efficiency of MEDAL. Compared with the corresponding design with idealized communication, BEACON-S achieves 86.27% of its energy efficiency.

D. k-mer Counting

BEACON-D: For BEACON-D, the performance improvement of designs with step-by-step optimizations on k-mer counting are shown in Fig. 15 (a). On average, CXL-vanilla improves the performance of the 48-thread CPU baseline and NEST by 124.88x and 1.46x, respectively. As for the optimizations, data packing improves the performance by 1.07x. Performance improvement of 2.75x is achieved with memory access optimization. Moreover, the proposed data placement and address mapping gain performance improvement of 1.21x. With all proposed optimizations, BEACON-D outperforms the 48-thread CPU and NEST by 443.08x and 5.19x, respectively. BEACON-D achieves 92.98% performance of the corresponding design with idealized communication, indicating that communication is no longer an issue.

For energy efficiency, as shown in Fig. 15 (b), BEACON-D reduces energy consumption of the CPU baseline by 190.23x and achieves 91.57% energy efficiency of NEST. Compared with the corresponding design with idealized communication, BEACON-D achieves 76.71% of its energy efficiency.

BEACON-S: For BEACON-S, the performance improvement of designs with step-by-step optimizations on k-mer counting are shown in Fig. 15 (c). On average, CXL-vanilla improves the performance of the 48-thread CPU baseline and NEST by 125.57x and 1.47x, respectively. As for the optimizations, data packing improves the performance by 1.09x. Performance improvement of 2.83x is achieved with memory access optimization. The proposed data placement and address mapping achieves 92.16% performance of the previous case, because communication is not a issue for k-mer counting in BEACON-S and data localization undermines utilization of the available memory bandwidth. The proposed data placement and address mapping scheme reduce data movement and lead to 1.12x improvement in energy efficiency. In addition, single-pass k-mer counting brings performance improvement of 1.48x. Overall, BEACON-S outperforms the 48-thread CPU and NEST by 527.99x and 6.19x, respectively. BEACON-S achieves 99.48% performance of the corresponding design with idealized communication, eliminating the performance bottleneck of communication.

Energy wise, as shown in Fig. 15 (d), BEACON-S reduces energy consumption of the CPU baseline by 102.05x and achieves 49.12% energy efficiency of NEST. Compared with the corresponding design with idealized communication, BEACON-S achieves 71.82% of its energy efficiency.

E. DNA Pre-alignment

For DNA pre-alignment, as shown in Fig. 16, BEACON-D and BEACON-S improve performance of the 48-thread CPU baseline by 362.04x and 359.36x, respectively. They also reduce the energy consumption of the 48-thread CPU baseline by 387.05x and 382.80x, respectively.

F. Energy breakdown

The energy breakdown for BEACON is shown in Fig. 17. The energy consumption ratio of computation is less than 1% for both BEACON-D and BEACON-S, indicating that the computation is highly efficient and almost free in BEACON.

For BEACON-D, as shown in Fig. 17 (a), in CXL-vanilla, communication dominates the energy consumption and consumes 60.68% energy on average. Data packing, memory access optimization, and the proposed data placement and address mapping reduce the energy consumption ratio of communication to 57.49%, 27.15%, and 14.40%. In addition, the multi-chip coalescing slightly reduces the energy consumption ratio of communication to 14.01%.

For BEACON-S, as shown in Fig. 17(b), in CXL-vanilla, communication also dominates the energy consumption and consumes 52.35% energy on average. Data packing, memory access optimization, and the proposed data placement and

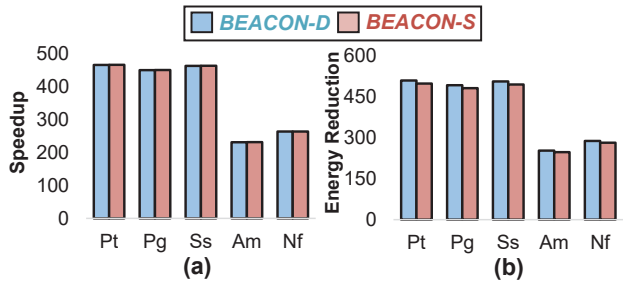

Figure 16. Experimental results for DNA pre-alignment. (a) Performance improvement. (b) Energy reduction.

address mapping reduce the energy consumption ratio of communication to 48.65%, 25.05%, and 10.29%. Although the single-pass k-mer counting slightly increases the energy consumption ratio of communication to 13.17%, it decreases the total energy consumption in BEACON-S.

To summarize, communication dominates the energy consumption in CXL-vanilla. For both BEACON-D and BEACON-S, the proposed optimizations greatly reduces the energy consumption from communication.

G. Improvements from the Optimizations

To demonstrate the effectiveness of the proposed optimizations for the BEACON architecture, we summarize the improvements from the optimizations as follows:

For BEACON-D, on average, the proposed optimizations improves the performance and energy efficiency by 2.21x and 3.70x, respectively. The proposed optimizations also reduces the energy consumption ratio of communication from 60.68% to 14.01%. For BEACON-S, on average, the proposed optimizations improves the performance and energy efficiency by 1.99x and 2.04x, respectively. The proposed optimizations also reduces the energy consumption ratio of communication from 52.35% to 13.17%.

VII. Related Work

Accelerator for Genome Analysis: Many previous researches propose accelerators for applications in genome analysis based on multi-core CPU [13], [19], FPGA [8], [9], [12], [23], [24], [25], GPU [9], [21], [49], [49], and ASIC [65]. However, as we have mentioned, many applications in genome analysis are memory-bound. These designs mostly focus on the computation part and cannot address the memory related challenges in genome analysis effectively.

PIM and NDP Accelerator for Genome Analysis: Many memory technologies, such as 3D-stacking DRAM [32], [52], [67] and ReRAM [28], [35], [69], have been used to build PIM and NDP accelerators for genome analysis. Unfortunately, these work have their own limitations such as constrained memory capacity [59], not cost-effective [29], immature technology [30], and so on.

DIMM based Accelerator for Genome Analysis: Among the existing PIM and NDP work for genome analysis, the

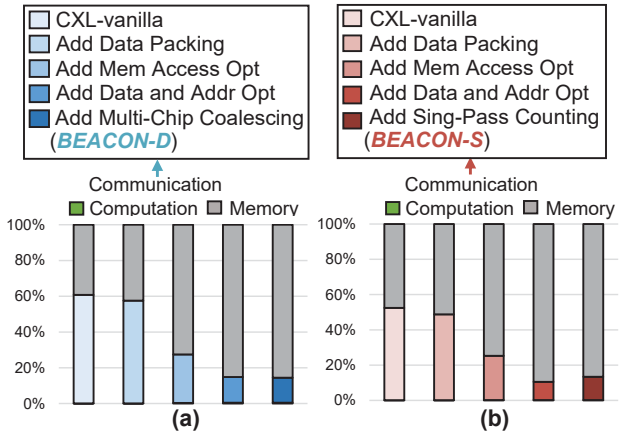

Figure 17. Energy breakdown. (a) BEACON-D. (b) BEACON-S.

DIMM based approaches [14], [29], [30] seem to be very promising due to the non-invasive designs to the cost-sensitve DRAM dies, which are practical, cost-effective, and scalable. However, these DIMM based designs also have their limitations in communication and memory expansion. BEACON addresses the limitations of these previous DIMM based designs.

Emerging Communication Specifications: Aim to build dis-aggregate I/O fabric as well as enable cache coherence between the host and accelerators, many emerging communication technologies have been proposed [57], such as CXL [48], OpenCAPI [17], CCIX [16], and Gen-Z [15]. As we have mentioned in Section. V, the designs and optimizations in BEACON could also be migrated to and be used in the NDP architectures with other emerging communication specifications.

VIII. Conclusion

To address the limitations of the previous DIMM based accelerators for genome analysis, i.e., performance bottlenecked by communication and limited potential for memory expansion under the scenario of memory dis-aggregation, this paper proposes BEACON, including BEACON-D and BEACON-S. As DIMM based NDP accelerator located near the dis-aggregated memory pool with the CXL support, BEACON enables efficient on-demand memory expansion with unmodified CXL-DIMMs and eliminates the performance bottleneck of communication without making any modification to the cost-sensitive DRAM dies. In addition, BEACON can be used for multiple applications in genome analysis. For different algorithms, we also adopt algorithm-specific optimizations to further improve the performance. Experimental results demonstrate that BEACON-D and BEACON-S improve performance of state-of-the-art DIMM based NDP accelerators by 4.70x and 4.13x, respectively.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their feedback. This work was supported in part by a gift from Samsung.

REFERENCES

[1] B. Abali, R. J. Eickemeyer, H. Franke, C.-S. Li, and M. A. Taubenblatt, "Disaggregated and optically interconnected memory: when will it be cost effective?" *arXiv preprint arXiv:1503.01416*, 2015.

[2] N. Ahmed, V.-M. Sima, E. Houtgast, K. Bertels, and Z. Al-Ars, "Heterogeneous hardware/software acceleration of the bwa-mem DNA alignment algorithm," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2015, pp. 240–246.

[3] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3, pp. 105–117, 2016.

[4] M. Alser, Z. Bingöl, 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, 2020.

[5] M. Alser, H. Hassan, A. Kumar, O. Mutlu, and C. Alkan, "Shouji: a fast and efficient pre-alignment filter for sequence alignment," *Bioinformatics*, vol. 35, no. 21, pp. 4255–4263, 2019.

[6] B. Benton, "Ccix, gen-z, opencapi: Overview & comparison," in *OpenFabrics Workshop*, 2017.

[7] M. Bielski, C. Pinto, D. Raho, and R. Pacalet, "Survey on memory and devices disaggregation solutions for hpc systems," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*. IEEE, 2016, pp. 197–204.

[8] S. M. Bose, V. S. Lalapura, S. Saravanan, and M. Purnaprajna, "k-core: Hardware accelerator for k-mer generation and counting used in computational genomics," in *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*. IEEE, 2019, pp. 347–352.

[9] N. Cadenelli, Z. Jaksić, J. Polo, and D. Carrera, "Considerations in using opencl on gpus and fpgas for throughput-oriented genomics workloads," *Future Generation Computer Systems*, vol. 94, pp. 148–159, 2019.

[10] G. S. L. Center, "Your doctorÂ¡Â¯s new genetic tools," http://learn.genetics.utah.edu/content/precision/example/, 2017.

[11] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens, "DRAMPower: Open-source DRAM power & energy estimation tool," *URL: http://www. drampower. info*, vol. 22, 2012.

[12] M.-C. F. Chang, Y.-T. Chen, J. Cong, P.-T. Huang, C.-L. Kuo, and C. H. Yu, "The smem seeding acceleration for DNA sequence alignment," in *Field-Programmable Custom Computing Machines (FCCM), 2016 IEEE 24th Annual International Symposium on*. IEEE, 2016, pp. 32–39.

[13] R. Chikhi and G. Rizk, "Space-efficient and exact de bruijn graph representation based on a bloom filter," *Algorithms for Molecular Biology*, vol. 8, no. 1, p. 22, 2013.

[14] J. Cong, Z. Fang, M. Gill, F. Javadi, and G. Reinman, "AIM: accelerating computational genomics through scalable and noninvasive accelerator-interposed memory," in *Proceedings of the International Symposium on Memory Systems*. ACM, 2017, pp. 3–14.

[15] C. Consortium, "Ccix specification," https://www.ccixconsortium.com/, Nov. 2021.

[16] G.-Z. Consortium, "Gen-z specifications," https://genzconsortium.org/specifications/, Nov. 2021.

[17] O. Consortium, "Opencapi specification," https://opencapi.org/, Nov. 2021.

[18] T. Coughlin, "Digital storage and memory," *Computer*, vol. 55, no. 1, pp. 20–29, 2022.

[19] S. Deorowicz, M. Kokot, S. Grabowski, and A. Debudaj-Grabysz, "Kmc 2: fast and resource-frugal k-mer counting," *Bioinformatics*, vol. 31, no. 10, pp. 1569–1576, 2015.

[20] H. Ellegren, "Genome sequencing and population genomics in non-model organisms," *Trends in ecology & evolution*, vol. 29, no. 1, pp. 51–63, 2014.

[21] M. Erbert, S. Rechner, and M. Müller-Hannemann, "Gerbil: a fast and memory-efficient k-mer counter with gpu-support," *Algorithms for Molecular Biology*, vol. 12, no. 1, p. 9, 2017.

[22] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, "Nda: Near-dram acceleration architecture leveraging commodity dram devices and standard memory modules," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 283–295.

[23] E. Fernandez, W. Najjar, and S. Lonardi, "String matching in hardware using the FM-index," in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*. IEEE, 2011, pp. 218–225.

[24] E. B. Fernandez, W. A. Najjar, S. Lonardi, and J. Villarreal, "Multithreaded fpga acceleration of dna sequence mapping," in *2012 IEEE Conference on High Performance Extreme Computing*. IEEE, 2012, pp. 1–6.

[25] E. B. Fernandez, J. Villarreal, S. Lonardi, and W. A. Najjar, "FHAST: FPGA-based acceleration of Bowtie in hardware," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 12, no. 5, pp. 973–981, 2015.

[26] P. Gu, X. Xie, Y. Ding, G. Chen, W. Zhang, D. Niu, and Y. Xie, "ipim: Programmable in-memory image processing accelerator using near-bank architecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 804–817.

[27] W.-j. Guan, Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui *et al.*, "Clinical characteristics of coronavirus disease 2019 in china," *New England Journal of Medicine*, 2020.

[28] W. Huangfu, S. Li, X. Hu, and Y. Xie, "RADAR: a 3d-ReRAM based DNA alignment accelerator architecture," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[29] W. Huangfu, X. Li, S. Li, X. Hu, P. Gu, and Y. Xie, "Medal: Scalable dimm based near data processing accelerator for dna seeding algorithm," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 587–599.

[30] W. Huangfu, K. T. Malladi, S. Li, P. Gu, and Y. Xie, "Nest: Dimm based near-data-processing accelerator for k-mer counting," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[31] R. Iyer, V. De, R. Illikkal, D. Koufaty, B. Chitlur, A. Herdrich, M. Khellah, F. Hamzaoglu, and E. Karl, "Advances in microprocessor cache architectures over the last 25 years," *IEEE Micro*, vol. 41, no. 6, pp. 78–88, 2021.

[32] B. K. Joardar, P. Ghosh, P. P. Pande, A. Kalyanaraman, and S. Krishnamoorthy, "Noc-enabled software/hardware co-design framework for accelerating k-mer counting," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–8.

[33] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, "CACTI-IO: CACTI with off-chip power-area-timing models," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 7, pp. 1254–1267, 2015.

[34] R. Kaplan, L. Yavits, and R. Ginosar, "Rassa: Resistive prealignment accelerator for approximate dna long read mapping," *IEEE Micro*, vol. 39, no. 4, pp. 44–54, 2018.

[35] R. Kaplan, L. Yavits, R. Ginosar, and U. Weiser, "A resistive CAM Processing-in-Storage architecture for DNA sequence alignment," *arXiv preprint arXiv:1701.04723*, 2017.

[36] R. Kaplan, L. Yavits, and R. Ginosasr, "Bioseal: In-memory biological sequence alignment accelerator for large-scale genomic data," in *Proceedings of the 13th ACM International Systems and Storage Conference*, 2020, pp. 36–48.

[37] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "Gpus and the future of parallel computing," *IEEE micro*, vol. 31, no. 5, pp. 7–17, 2011.

[38] J. S. Kim, D. S. Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, and O. Mutlu, "Grim-filter: Fast seed location filtering in dna read mapping using processing-in-memory technologies," *BMC genomics*, vol. 19, no. 2, pp. 23–40, 2018.

[39] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible DRAM simulator." *Computer Architecture Letters*, vol. 15, no. 1, pp. 45–49, 2016.

[40] O. Kocberber, B. Grot, J. Picorel, B. Falsafi, K. Lim, and P. Ranganathan, "Meet the walkers: Accelerating index traversals for in-memory databases," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2013, pp. 468–479.

[41] W. Kwon, "Etri extended memory pool system architecture using gen-z protocol," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 123–125.

[42] W. Kwon, C. Park, and M. Oh, "Gen-z memory pool system architecture," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2020, pp. 1356–1360.

[43] J. D. Leidel and Y. Chen, "Hmc-sim-2.0: A simulation platform for exploring custom memory cube operations," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2016, pp. 621–630.

[44] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997*, 2013.

[45] H. Li, D. S. Berger, S. Novakovic, L. Hsu, D. Ernst, P. Zardoshti, M. Shah, I. Agarwal, M. Hill, M. Fontoura *et al.*, "First-generation memory disaggregation for cloud platforms," *arXiv preprint arXiv:2203.00241*, 2022.

[46] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," *ACM SIGARCH computer architecture news*, vol. 37, no. 3, pp. 267–278, 2009.

[47] C. E. Link, "Compute express link 2.0 white paper," https://www.computeexpresslink.org/, Nov. 2020.

[48] C. E. Link, "Compute express link specifications - revision 2.0," https://www.computeexpresslink.org/download-the-specification, Nov. 2020.

[49] Y. Liu and B. Schmidt, "Evaluation of GPU-based seed generation for computational genomics using burrows-wheeler transform," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 684–690.

[50] R. Lu, X. Zhao, J. Li, P. Niu, B. Yang, H. Wu, W. Wang, H. Song, B. Huang, N. Zhu *et al.*, "Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding," *The Lancet*, 2020.

[51] X. Ma, M. Mau, and T. F. Sharbel, "Genome editing for global food security," *Trends in biotechnology*, vol. 36, no. 2, pp. 123–127, 2018.

[52] N. Mcvicar, C.-C. Lin, and S. Hauck, "K-mer counting using bloom filters with an fpga-attached hmc," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2017, pp. 203–210.

[53] P. Melsted and J. K. Pritchard, "Efficient counting of k-mers in dna sequences using a bloom filter," *BMC bioinformatics*, vol. 12, no. 1, p. 333, 2011.

[54] J. D. Merker, A. M. Wenger, T. Sneddon, M. Grove, Z. Zappala, L. Fresard, D. Waggott, S. Utiramerur, Y. Hou, K. S. Smith *et al.*, "Long-read genome sequencing identifies causal structural variation in a mendelian disease," *Genetics in Medicine*, vol. 20, no. 1, pp. 159–163, 2018.

[55] A. Nag, C. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "Gencache: Leveraging in-cache operators for efficient sequence alignment," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 334–346.

[56] NCBI, "Genome database," https://www.ncbi.nlm.nih.gov/genome, 2018.

[57] C. Pinto, D. Syrivelis, M. Gazzetti, P. Koutsovasilis, A. Reale, K. Katrinis, and H. P. Hofstee, "Thymesisflow: A software-defined, hw/sw co-designed interconnect stack for rack-scale memory disaggregation," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 868–880.

[58] D. H. Ponstingl, "SMALT," http://www.sanger.ac.uk/science/tools/smalt-0, 2016.

[59] S. H. Pugsley, J. Jestes, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li, "Comparing implementations of near-data computing with in-memory map reduce workloads," *IEEE Micro*, vol. 34, no. 4, pp. 44–52, 2014.

[60] A. Ramachandran, Y. Heo, W.-m. Hwu, J. Ma, and D. Chen, "Fpga accelerated dna error correction," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1371–1376.

[66] S. Van Doren, "Hoti 2019: Compute express link," in *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE Computer Society, 2019, pp. 18–18.

[61] J. Shendure and H. Ji, "Next-generation DNA sequencing," *Nature biotechnology*, vol. 26, no. 10, pp. 1135–1145, 2008.

[62] G. Singh, J. Gómez-Luna, G. Mariani, G. F. Oliveira, S. Corda, S. Stuijk, O. Mutlu, and H. Corporaal, "Napel: Near-memory computing application performance prediction via ensemble learning," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[63] Synopsys, "Design complier," https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html, 2018.

[64] F. Technology, "28nm technology libraries," https://www.faraday-tech.com/cn/category/BrowseByTechnology?method=browserCategory&tech=28nm&master.ipSearchForm.technology=28nm.

[65] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly," in *ACM SIGPLAN Notices*, vol. 53, no. 2. ACM, 2018, pp. 199–213.

[67] T. Vijayaraghavan, A. Rajesh, and K. Sankaralingam, "MPU-BWM: Accelerating sequence alignment," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 179–182, 2018.

[68] Y. Wang, X. Li, D. Zang, G. Tan, and N. Sun, "Accelerating fm-index search for genomic data processing," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–12.

[69] F. Zokaee, M. Zhang, and L. Jiang, "Finder: Accelerating fm-index-based exact pattern matching in genomic sequences through reram technology," in *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 2019, pp. 284–295.