Genome Sequence Alignment - Design Space Exploration for Optimal Performance and Energy Architectures

Yasir Mahmood Qureshi, Jose Manuel Herruzo,
Marina Zapater, *Member, IEEE,* Katzalin Olcoz, *Member, IEEE,*
Sonia Gonzalez-Navarro, Oscar Plata, and David Atienza, *Fellow, IEEE,*

Abstract—Next generation workloads, such as genome sequencing, have an astounding impact in the healthcare sector. Sequence alignment, the first step in genome sequencing, has experienced recent breakthroughs, which resulted in next generation sequencing (NGS). As NGS applications are memory bounded with random memory access patterns, we propose the use of high bandwidth memories like 3D stacked HBM2, instead of traditional DRAMs like DDR4, along with energy efficient compute cores to improve both performance and energy efficiency. Three state-of-the-art NGS applications, Bowtie2, BWA-MEM and HISAT2, are used as case studies to explore and optimize NGS computing architectures. Then, using the gem5-X architectural simulator, we obtain an overall 68% performance improvement and 71% energy savings using HBM2 instead of DDR4. Furthermore, we propose an architecture based on ARMv8 cores and demonstrate that 16 ARMv8 64-bit OoO cores with HBM2 outperforms 32-cores of Intel Xeon Phi Knights Landing (KNL) processor with 3D stacked memory. Moreover, we show that by using frequency scaling we can achieve up to 59% and 61% energy savings for ARM in-order and OoO cores, respectively. Lastly, we show that many ARMv8 in-order cores at 1.5GHz match the performance of fewer OoO cores at 2GHz, while attaining 4.5x energy savings.

Index Terms—genome sequencing, sequence alignment, NGS, HPC, HBM2, KNL, architecture exploration, many-core.

1 Introduction

THE fast paced digitization of the society has enabled greater efficiency in all phases of our daily lives, but at the price of a significant increase in computational resources. This results in data centers consuming 1% (198TWh) of the global energy demand in 2018 [1]. Data centers host High Performance Computing (HPC) systems used for a wide range of applications, from weather forecasts or particle physics to genomics and precision medicine. In HPC data centers, where performance is the main evaluation metric, x86 based CPUs along with GPUs constitute the backbone and de facto industry standard of the processing infrastructure. Recent studies as in [2], [3] have evaluated using energy-efficient ARM cores in HPC domain. In [4], [5], authors have also looked into ARM based data centers. These works agree that ARM based systems are more energy efficient as compared to x86 based systems, but are not at par in performance with x86, therefore, not suitable for HPC domain. In all these studies, the benchmarks used for evaluation of performance are compute bounded, with x86 outperforming ARM based systems. The works [2], [3] also look into the memory bandwidth (BW) of ARM based systems with traditional DDR as main memory, for memory bounded applications and benchmarks, and conclude that the ARM systems fall behind x86. However, the authors in [3] suggest that ARM based systems might benefit from future 3D high bandwidth memories.

Next generation biomedical applications like genome sequencing, are having an astounding impact in the fields of bioinformatics, cancer research, food microbiology and drug discovery [6], [7]. Genome sequencing is also one of the first steps in understanding a new disease, its effects, development of diagnostic tests and possible cure and vaccines for it. This process should be fast and efficient in case of a new disease outbreak, in order to curtain it, as has been evident during the recent outbreak of the novel coronavirus (COVID-19). SARS CoV-2, the virus that causes COVID-19, was first completely sequenced in China by 11 January 2020 and shared with the world community [8]. The Institut Pasteur in France sequenced the genome for COVID-19 in three days [8], [9]. Since the availability of the genome sequence of COVID-19, scientists and researchers around the world are racing towards the development of vaccine and diagnostic kits, in order to cure and overcome the outbreak.

HPC architectures are usually used for fast NGS due to

- *Y. M. Qureshi, M. Zapater and D. Atienza are with the Embedded Systems Laboratory, Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland. E-mail: {yasir.qureshi, marina.zapater, david.atienza}@epfl.ch*
- *K. Olcoz is with the Department of Computer Architecture, Complutense University of Madrid, 28040 Madrid, Spain. E-mail: katzalin@ucm.es*
- *J. M. Herruzo, S. Gonzalez-Navarro and O. Plata are with the Department of Computer Architecture, Universidad de Malaga. Andalucia Tech, 29071 Malaga, Spain. E-mail: jmherruzo@uma.es, sonia@ac.uma.es, oplata@uma.es*

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 00.0000/TC.0000.0000000

the complexity of the sequence alignment process. These HPC systems are extremely power hungry and performance is usually the only evaluation metric. However, performance alongside with energy should be considered together for enabling scalable HPC systems. HPC systems run both compute-bounded and memory-bounded applications. Genome sequencing [6], [10] is among the latter due to pointer chasing [11]. It is the process of determining the DNA sequence or the order of bases As, Cs, Gs, and Ts making up the organism's genome. Next-generation sequencing (NGS) is a high-throughput genome sequencing method. Before the advent of NGS, Maxam and Gilbert [12] and Sanger along with his colleagues [13] came up with techniques to sequence DNA by fragmentation and chain termination, respectively, in the 1970s. The Sanger sequencing was further commercialized and became the de facto sequencing technique for 30 years. It has the honour of being the sequencing method used for the complete human genome in 2004 [14], through a 13-year effort under Human Genome project with an estimated cost of $2.7 billion [15]. With the advent and rapid developments in NGS, the human genome was again sequenced in 5-months at a cost of $1.5 million [16]. The common feature NGS platforms share is large parallel sequencing of clonally amplified or single DNA molecules separated spatially in a flow cell [15]. This is a departure from Sanger sequencing, which uses separate chain-termination for individual sequencing reactions. NGS is a huge parallel process that generates hundreds of megabases to giga-bases of nucleotide-sequence output in a single instrument run [15]. In sequence alignment, which is one of the first steps in NGS, a sequence read is aligned or checked against a genomic reference for regions of similarity [10]. This process must tolerate differences between the query read and the reference genome, due to errors in the sequencing process and genuine differences between organisms. In addition, the strategies used in sequence alignment have a pointer-chasing nature. They involve a repeated series of irregular memory access patterns through which they determine the memory address of the next (pointer) access, and the previous accessed data is required. Depending on the length of the sequence to be read, this pointer-chasing nature affects the performance of the application. In [17], it is shown that these NGS applications are memory bounded with 40% stalls due to memory, and 80% of these memory stalls is accounted for by long latency DRAM accesses. As these applications are memory bounded, having high performance compute nodes does not help in improving performance, but actually adds to the energy consumption of the system, due to underutilization of the processors. Instead, simpler ARM based architectures along with high bandwidth memories can help in having an energy efficient architecture for NGS with better performance compared to existing solutions, leading to global energy savings for HPC data centers.

NGS applications use full-text indexing strategies, such as the FM-index, based on Burrows-Wheeler transform (BWT) [18], [19] for fast sequence alignments. Additionally, Bowtie2 [20] is a new state-of-the-art NGS application based on FM-index, with efficient multi-threading capabilities. BWA-MEM [21] is a widely used sequence alignment application also based on BWT. Then, HISAT2 [22] is a graph based sequence alignment application and superior to both Bowtie2 and BWA-MEM in performance. HPC class compute resources like the Intel Xeon Phi KNL processors [23], Intel Skylake and Broadwell architectures, which support multiple threads, are used to run these NGS applications and maximize their performance [18], [24]. GPUs are also explored for genome sequencing as in [25]. In [26], authors utilize HPC type many-core x86 clusters for NGS. However, the underlying behaviour in NGS applications, whether it is based on FM-index or graph based search, behaves like pointer chasing, with random accesses to the memory and low cache locality. Thus, HPC nodes like KNL or GPUs are not efficient for these sort of memory bounded applications, as they underutilize and waste the compute resources.

ARM based scale-out processors have been evaluated for memory bounded data center applications, and reported to perform well as in [5], using traditional DDR4 memory. To the best of our knowledge, no studies have analysed ARM based architectures along with 3D stacked memories for memory bounded genome sequence alignment applications. In this paper, we demonstrate that random access memory bounded NGS workload can be executed on energy efficient ARM based platforms, with performance at par or surpassing that of existing x86 or accelerators like KNL, given that there is enough memory bandwidth available, such as the one provided by HBM2. KNL was selected as the x86 based comparison architecture instead of Intel Skylake or Broadwell. The reason is that KNL has a 3D stacked memory, making the comparison fair across the memory sub-system for both our proposed ARM based system as well as x86 based system.

In this paper, we propose and utilize a design space exploration methodology to find performance, energy and area optimized architectures for the NGS application domain. Using this methodology we propose an optimized architecture based on 3D-stacked high bandwidth memory (HBM2) [27] alongside energy efficient ARMv8 64-bit compute cores. We use the gem5-X [28] architectural simulator, an open source, validated and enhanced version of gem5 [29], with HBM2 memory model. Three widely used state-of-the-art NGS applications namely, Bowtie2 [20], BWA-MEM [21] and HISAT2 [22] with different search strategies are used as case study applications for performance and energy optimization. The main contributions of this work are:

- We show by optimizing the memory sub-system using an architectural exploration methodology, HBM2 with no last level cache (LLC) can outperform traditional memory hierarchies with caches and DDR4 for sequence alignment with Bowtie2, with up to 68% in performance improvement and 71% energy savings.
- Combining the memory sub-system with compute core optimization, we explore the architectural design space with different core types, core count, frequency and LLC size with HBM2 as main memory for NGS. We demonstrate that many ARMv8 in-order cores surpass or match the performance of fewer ARMv8 OoO cores saving up to 2x energy when both are operating at same frequency and up to 4.5x energy benefits when in-order cores are operating at lower frequency than OoO cores.
- We explore the use of ARMv8 cores along with HBM2, as a replacement for state-of-the-art Intel Xeon Phi 7210 KNL processor with integrated 3D-stacked MCDRAM memory,

and demonstrate that 16 ARM OoO cores can match the performance of 32 KNL cores. We also show that 28 ARM in-order cores match the performance of 32 KNL cores.
- We perform a frequency sensitivity analysis for ARM-based systems. We show that we can achieve up to 59% and 61% energy benefit, for ARM in-order and OoO system, respectively, when operating more cores at lower frequency, while matching or surpassing the performance, as compared to fewer cores at a higher frequency.

2 Related Work

HPC systems, usually based on x86 cores, have traditionally been used for fast NGS and sequence alignment process. These x86 CPU-based architectures are also the first point of execution and testing when a new sequencing algorithm is being developed. The widely used BWA-MEM was tested on Intel Xeon 5420 running at 2.5GHz system, as in [21]. Similarly, Bowtie2 has been reported to run on Intel Xeon X5550 Nehalem running at 2.66 GHz rented from Amazon web services (AWS) system, as in [20]. Previous works in [24] and [30] has been done on improving the scalability of well known and state-of-the-art-aligners like BWA-Mem, Bowtie2 and HISAT2 [22] on Intel based HPC architectures like Intel KNL, Skylake and Broadwell architectures [24].

In addition to x86 CPU based system, GPUs have also been explored for high-throughput sequence alignment task, as in [25] and [31]. The Arioc GPU aligner in [25] achieves up to 10x speedup in comparison to CPU-based system for the seed and extend stage of BWA-MEM. However, the authors have not looked into energy comparisons. Moreover, Intel Xeon X5670 CPUs running at 2.93GHz were used along with the GPUs, which cannot be used as a standalone system. In addition, using GPUs for read alignments is challenging from a software prospective, as one has to manage single-instruction multiple-data (SIMD) threading as well as memory management, including data layout and data transfers between CPU and GPU [32]. Dynamic programming dependencies along with memory intensive tasks in sequence alignment add to the challenges of using parallel threaded GPU implementation of NGS [33]. Therefore, GPUs have not been widely adopted for NGS so far.

Sequence alignment accelerators both on ASICs and FPGAs have been developed previously. The GenAx accelerator [34] provides around 31.7x speedup as compared to 14-core Xeon E5 server. The authors in [35] implement a dataflow architecture on FPGA for Smith-Waterman Matrix-fill and Traceback stages, widely used in sequence aligners like BWA-MEM [21] and Bowtie2 [20]. However, this only accelerates the Smith-Waterman part of the sequence alignment application and the other stages of the application have to be run on the CPU. In [36], the authors implement Smith-Waterman accelerator on FPGA, which is attached as a co-processor to the IBM POWER8 CPU, and achieves 1.6x speed-up compared to CPU-based version. The DRAGEN platform [37] from Illumina is another state-of-the-art FPGA-based sequence aligner which operates in hybrid hardware-software configuration with a dual Intel Xeon processor. The DRAGEN architecture is proprietary and has not been disclosed, but it is reported to achieve 16-18x speed-up compared to BWA-MEM on a software based system [38]. Despite the speed-up that the sequence alignment accelerators can achieve, it takes a significant amount of time and effort for these accelerators to be developed, either on FPGAs or ASICs. On one hand, if they are on FPGAs, which are resource constrained, and hence, the application have to be scaled out on multiple compute nodes [6]. Additionally, application developed for one FPGA might not be compatible across different FPGA generations. On the other hand, in the case of ASICs, if a new sequencing algorithm is developed, it cannot be adopted for the same ASIC platform. Due to these factors, CPU-based sequence alignment is still widespread and is commonly used for research, as it is fast to deploy and the same system can be used with follow-up versions of the application.

On the memory front, NGS applications are memory intensive. High bandwidth memories like 3D stacked HBM2 are well suited for these applications. In [39], authors use HBM2 with the Smith-Waterman algorithm on FPGA-based platform and report 2x speed-up compared to DDR4 memory system. Processing-in-memory (PIM) exploiting HBM2 has been proposed in [40] for the seed location filter, used just before the alignment, and achieve 1.81x-3.65x speed-up compared to a state-of-the-art FastHASH seed location filter. However, in both these works significant software changes were done to use HBM2 with an accelerator on FPGA [39] or as a PIM [40]. Therefore, to the best of our knowledge, this is the first work where it is proposed the use of an ARM many-core compute sub-system along with a high bandwidth HBM2 based memory sub-system for an energy efficient sequence alignment system across three state-of-the-art and widely used NGS applications, namely, Bowtie2 [20], BWA-MEM [21] and HISAT2 [22].

3 Sequence Alignment Applications

Different optimized sequence alignment NGS applications exist, including some of the most widely used, relying upon the FM-index data structures and search algorithms. Among them the applications that we use as our case study. In this section we describe them briefly along with the FM-index.

3.1 FM-index

FM-index is a data structure that allows fast substring searches over large texts [19]. FM-index is based on several data structures and algorithms, such as Suffix Array and Burrows-Wheeler Transform (BWT). Given a pattern or query Q, the FM-index allows to find all occurrences of Q in the text T. The search process takes the following two steps, as shown in Fig.1.a: *count* and *locate*.

3.1.1 *Count*

Count is a backward iterative process which performs two rank queries as highlighted in red in Fig. 1.a and an addition per each character in Q (starting from the end). In FM-index, rank queries are typically performed using memory-consuming data structures, which store previously calculated data, making it a low computing, highly memory bound operation. As shown in Fig. 1.a, in the highlighted yellow part, the count step requires accessing random sections of the memory in each iteration. The result of this step are pointers to the first and last position in the occurrences interval of Q in the sorted list of suffixes from T.

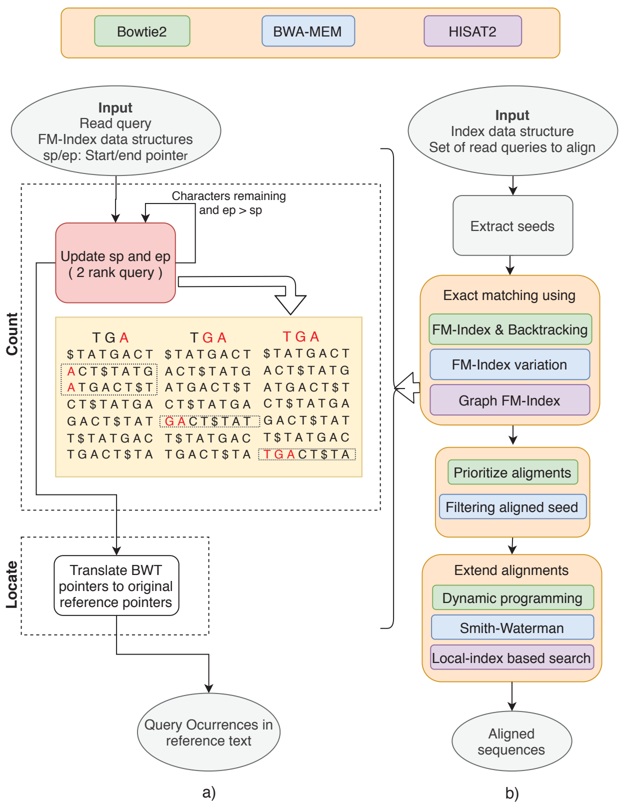

Bowtie2
BWA-MEM
HISAT2
Input
Read query
FM-Index data structures
sp/ep: Start/end pointer
Characters remaining and ep > sp
Update sp and ep (2 rank query)
Count
TGA TGA TGA
$TATGACT $TATGACT $TATGACT
ACT$TATG ACT$TATG ACT$TATG
ATGACT$T ATGACT$T ATGACT$T
CT$TATGA CT$TATGA CT$TATGA
GACT$TAT GACT$TAT GACT$TAT
T$TATGAC T$TATGAC T$TATGAC
TGACT$TA TGACT$TA TGACT$TA
Locate
Translate BWT pointers to original reference pointers
Query Ocurrences in reference text
Input
Index data structure
Set of read queries to align
Extract seeds
Exact matching using
FM-Index & Backtracking
FM-Index variation
Graph FM-Index
Prioritize aligments
Filtering aligned seed
Extend alignments
Dynamic programming
Smith-Waterman
Local-index based search
Aligned sequences
a)
b)

Fig. 1. Phases for (a) FM-index and (b) Genome Sequencing Applications

3.1.2 *Locate*

Locate uses the indexes of the rows to access the suffix array, where it finds the position of every occurrence of Q in the text T. Locate can be performed in one memory access at the cost of a higher memory footprint. With a reduced memory footprint, it also shows a random memory access pattern.

FM-index data structures are used in several well-known sequence alignment applications, such as Bowtie2, BWA and HISAT2 which we will discuss next. All these applications use the *seed and extend* techniques, depicted in Fig. 1.b. The differences between the these alignment applications are depicted through color coded blocks in Fig. 1.b.

3.2 Bowtie2

Bowtie2 [20] is an open-source, ultra-fast and memory-efficient alignment application used for aligning DNA reads to large genomes, able to support gapped alignments. It relies upon the BWT and the FM-index algorithm to quickly find non-exact alignments that satisfy a specified alignment policy. Bowtie2 includes several novelties over the *seed and extend* basis, as the prioritization of seed alignments in order to reduce the computing power used, as shown Fig. 1.b.

Compared to other sequence alignment tools, Bowtie2 is 2.5-3x faster than the Burrows Wheeler Aligner (BWA) when both applications are searching for gapped alignments.

3.3 BWA-MEM

BWA-MEM [21] is another widely used open-source sequence alignment algorithm based on FM-index data structures. It automatically chooses between local and end-to-end alignments, supports paired-end reads and performs chimeric alignment. This algorithm follows *seed-and-extend*, common in other sequence alignment applications as Bowtie2, but including some novelties. BWA-MEM includes an additional step which creates groups of seeds, and filters them in order to reduce unsuccessful seed extension at a later step. The seed extension step also includes a variation focused on reducing the computing time used by sub-optimal seed extension and prioritize end-to-end alignments over local ones.

As most sequence alignment applications, BWA-MEM processes a batch of reads at a time. This algorithm uses this feature to obtain both single-end and paired-end mappings. BWA-MEM also supports multi-threading.

3.4 HISAT2

HISAT2 [22] is also and open-source NGS application based on the *seed-and-extend* and an extension of FM-index for graphs as opposed to the raw FM-index in previous applications, as depicted in Fig. 1.b. Named the Hierarchical Graph FM index (HGFM), it is composed of a global GFM (graph FM) index, representing a population of human genomes and a large set of small GFM indexes, collectively covering the whole genome. These small indexes (named local indexes), combined with several alignment strategies, enable rapid and accurate alignment of sequencing reads.

According to results reported in [22], HISAT2 is faster than any other state-of-the-art sequence alignment algorithms like Bowtie2 and BWA-MEM. Although HISAT2 claims to be scalable and supports multi-threading, we noticed its scalability is limited, both on real hardware as well as in the simulator, as we will discuss in Section 6.3.1.

Overall, there are similarities as well as some difference between the three genome sequencing case study applications, as has already been shown in Fig. 1.b. We will explore architectures with optimal performance and energy for these NGS domain applications, taking advantage of the similarities of the applications, but flexible enough to cater for the differences between them.

4 Methodology for Architecture Exploration

NGS application domain is usually memory bounded with random memory access pattern. As discussed in Section 3, the three case study applications exhibit a memory bounded behaviour with random access pattern, as in a pointer chasing and graph processing applications. The main compute operation in these applications is the comparison operation, performed when comparing the search sequence to a reference index.

Figure 2 shows the architecture exploration and optimization methodology. The methodology comprises three phases.

4.1 Memory Exploration and Optimization

The first phase we propose is related to memory system exploration and optimization. It requires optimizing the entire memory sub-system comprising both the main memory and the cache. As the application is memory bounded with random access patterns, for the first step we analyse replacing the traditional DDR4 with a higher bandwidth memory like the 3D-stacked HBM2 and look into the performance and energy benefits. If the HBM2 based system with LLC outperforms DDR4 with LLC both in terms of energy and

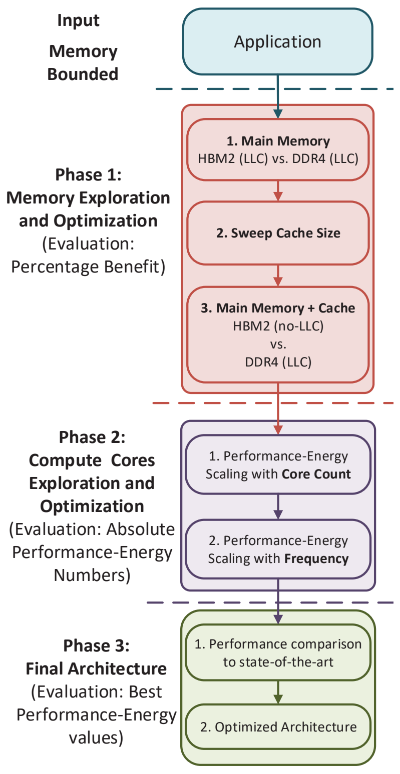

Input
Memory Bounded
Application
Phase 1: Memory Exploration and Optimization (Evaluation: Percentage Benefit)
1. Main Memory HBM2 (LLC) vs. DDR4 (LLC)
2. Sweep Cache Size
3. Main Memory + Cache HBM2 (no-LLC) vs. DDR4 (LLC)
Phase 2: Compute Cores Exploration and Optimization (Evaluation: Absolute Performance-Energy Numbers)
1. Performance-Energy Scaling with Core Count
2. Performance-Energy Scaling with Frequency
Phase 3: Final Architecture (Evaluation: Best Performance-Energy values)
1. Performance comparison to state-of-the-art
2. Optimized Architecture

Fig. 2. Architecture exploration and optimization methodology.

performance, it suggests that HBM2 should replace DDR4 in further optimization steps.

During the second step of the memory exploration phase, we look into sweeping the LLC size while optimizing the cache sub-system. The LLC size providing the best performance is selected as the optimal one.

For the final and third step of the memory exploration phase we optimize the cache sub-system focusing on the LLC. We explore a no-LLC HBM2 system and compare it to LLC with DDR4 both for energy efficiency and performance. If no-LLC HBM2 system outperforms LLC with DDR4 in terms of energy efficiency and performance, we consider it to be a potential candidate for an optimized architecture and use it during subsequent phases of the methodology, otherwise we discard it. If we look into step-1 and step-3, we are effectively comparing three configurations, HBM2 (LLC) vs. DDR4 (LLC) vs. HBM2 (no-LLC). We do not compare to DDR4 (no-LLC), as it has a lower BW compared to HBM2, and without any caching effects of LLC, it has lower performance compared to DDR4 (LLC) and HBM2 (LLC) configurations.

During this first phase in the methodology, we use *percentage* performance/energy benefits instead of absolute values, as we are looking for architectural choices which are better performing and discard the others.

4.2 Compute Core Exploration and Optimization

After the identification of the best memory sub-systems in the first phase, we optimize the compute cores.We use two types of cores: energy efficient in-order cores and high performance OoO cores. In the first stage of this phase, we explore the performance and energy scaling of the region-of-interest (ROI) with the number of compute cores. We look into the optimized core-count and core type based on performance and energy metrics.

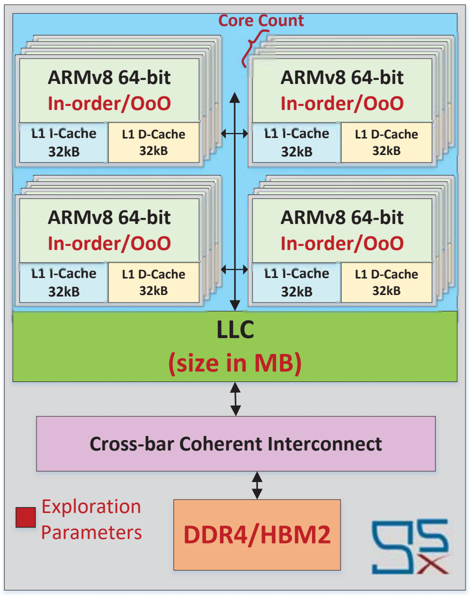


Fig. 3. Architectural block diagram of experimental setup in gem5-X [28].

For second stage of phase 2, we investigate scaling with the core frequency. We look into the core count and core types comparing their performance and energy. We also analyse the area constraints when comparing different core types.

Absolute performance/energy values are used during this phase of the methodology, to have an insight into the best performing system.

4.3 Final Architecture

Finally, we compare the optimized architecture with state-of-the-art systems currently being used for the application under study. If our proposed architecture outperforms the state-of-the-art solutions, we select it as our final optimized architecture.

5 Architectural Exploration and Simulation Framework

Architectural exploration is necessary to find the best architecture for an application, for a given optimization metric. In this paper we consider energy efficiency together with performance as the primary optimization metrics.

5.1 Architectural Simulation Framework

Our simulation framework enables us to perform fast architectural exploration for performance/energy optimized architecture for any given application. For HPC applications, we need a simulation framework capable of running multi-threaded applications on a many-core simulated system.

5.1.1 Experimental Setup

We use gem5-X[1] [28], an open source, validated and extended version of cycle-accurate gem5 [29] architectural simulator, which exhibits a validation error of up to 4% when simulating the ARMv8 64-bit cores of the JUNO platform [41]. ARM full-system (FS) simulation mode is used with an Ubuntu 16.04 OS, as all our case study applications require various OS multi-threading support. As shown in Fig. 3, multiple

1. https://github.com/esl-epfl/gem5-X.git

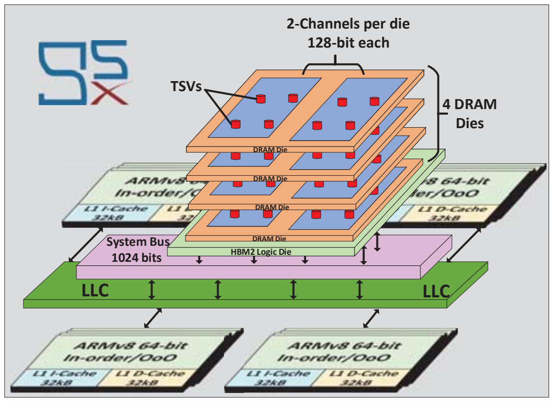


Fig. 4. System level architectural diagram with 3D-stacked HBM2.

ARMv8 64-bit in-order and OoO cores are used for the architectural exploration with L1 instruction (L1-I) and L1 data (L1-D) cache fixed at 32KB using the validated ARM JUNO platform [41] as starting point. All the cores are connected to the LLC, which then connects to the main memory of 4GB via a coherent cross-bar interconnect. Gem5-X statistics for the ROI are used for the performance analysis.

5.1.2 Power Models

For energy evaluation, we use the power model for 28nm CMOS bulk technology node for ARM OoO and in-order cores proposed in [5] and [28]. The power model includes the core active, wait-for-memory (WFM) and static core energy, LLC read/write and static cache energy. For the memory power models, we use the DRAM power values as reported in [42]. Furthermore, counters in gem5-X statistics like active CPU cycles, WFM cycles, cache read and writes hits and main memory accesses are used for power modeling.

5.1.3 High Bandwidth Memories

High bandwidth memories like HBM2 [27] help in alleviating the memory bottleneck of memory bounded applications. We propose to use the 3D-stacked HBM2 memory for such workloads, attaining a bandwidth of 307.2 GB/s. The 3D stacking has been made possible by through-silicon-vias (TSVs), enabling the memory and the logic cores to be placed in the same die resulting in high bandwidth memory accesses.

Gem5-X implements the HBM2 memory model by extending the DRAM controller model in gem5 according to the architectural details of HBM2, as in Table 1. Figure 4 shows the full system architecture with HBM2 in a multi-core system in gem5-X. Four 3D-stacked DRAM dies are connected with TSVs and each die with 2-channels giving a total of 8-channels for HBM2. As each channel is 128-bit wide, it connects to a 1024-bit wide coherent system bus. The system bus connects to the cache hierarchy which subsequently connects to the compute cores.

For the power model of HBM2, the energy values of [43] are used.

TABLE 2
Parameters for architectural exploration and optimization

Parameter	Values
Core Type	ARMv8 in-order, ARMv8 OoO
Core Count	8 to 64 cores
Core Frequency	1GHz, 1.5 GHz, 2 GHz
LLC size	No LLC, 1MB, 1MB/8-cores, 2MB/8 cores as in Table 3
Memory Type	DDR4, HBM2

5.2 Architectural Exploration Parameters

Following the optimization methodology discussed in Section 4, Table 2 summarizes the architectural parameters we sweep to get an optimized architecture, since they have the most impact on system performance and energy.

In our experiments, we pin one software thread to each physical core. For systems with no-LLC, we do not go beyond 28 physical cores, because the simulation turn around time increases drastically with the number of cores and the scaling trend can already be captured with up to 28-core simulations. Similarly, for OoO cores we go to a maximum of 32-cores systems with LLC. Lastly, for in-order cores we additionally simulate 64-core systems, which we will discuss in Section 6.

To explore the effects of varying LLC size (L2 in our case), in addition to no-LLC and a fixed LLC of 1MB, we also change the LLC size proportionally to the number of cores, so to have the same LLC size-to-core count ratio, as in Table 3. We use two ratios, 1MB/8-cores and 2MB/8-cores. The ratio is restricted to 2MB/8-cores (at max.), as the LLC size will increase unrealistically large with the number of cores if we increased this ratio further. The ratio scales well with 8, 16 and 32 cores, but for 24 and 28 cores, according to the ratio of 1MB/8-core LLC size should be 3MB and 3.5MB, respectively. Since these sizes are not a power of 2, we scale up the LLC to 4MB for 24 and 28 cores. Similarly, for 2MB/8-cores, we use a size of 8MB for both 24 and 28 cores.

6 Architecture Exploration Results

In this section we will explore and optimize architectures for performance, energy and area for the three genome sequencing applications discussed in Section 3, using the optimization methodology in Fig. 2 presented in Section 4. ROI is the search and alignment phase of each application. Hence, the performance numbers are given in terms of execution time for ROI. The energy consumption corresponds to the energy in the ROI for the complete system including CPU cores, caches and memory.

TABLE 1
HBM2 architecture parameters

Parameter	Value	Parameter	Value
Clock period (tCK)	0.833ns	#Channels	8
Channel width	128 bits	#I/Os	1024 pins
Ranks per channel	1	Bandwidth	2.4Gbps/pin
Banks per rank	16	Burst length	4
Bank groups per rank	4		

TABLE 3
LLC Sizes and scaling with number of cores.

Core Count	Fixed Size LLC	LLC 1MB/8-cores	LLC 2MB/8-cores
8 cores	LLC = 1MB	LLC = 1MB	LLC = 2MB
16 cores	LLC = 1MB	LLC = 2MB	LLC = 4MB
24 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB
28 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB
32 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB

As input data, we used a set of single-end queries generated by the Mason simulation tool [44]. These queries, with 200 symbols in average, have been searched in the Parus major (Great Tit) genome reference Parus_major1.0.3 [45] which is composed of around 1 gigabases.

We first perform an in-depth exploration of Bowtie2, and using the insights we gain during its architectural optimization, we perform a similar exploration for BWA-MEM and HISAT2.

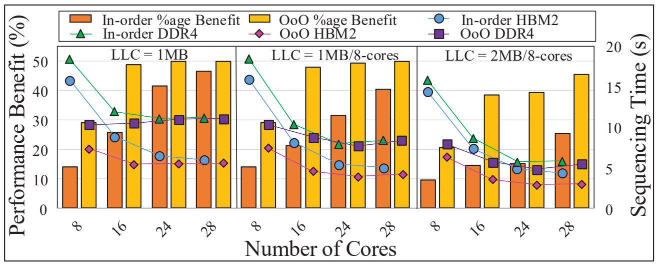
6.1 Bowtie2 Architectural Exploration

For all the performance and energy results, we launched Bowtie2 in gem5-X, and performed 200K read alignments, which is a representative workload for sequence alignment and stresses the system resources. According to phase-1 of the methodology, in this section we first explore the performance and energy benefits of using HBM2, with and without LLC, as compared to DDR4 as in Section 6.1.1 and 6.1.2, respectively. Then, in accordance with phase-2 of the methodology, we discuss how performance-energy scales with core count and frequency, as in Section 6.1.3 and 6.1.4, respectively, in quest for an optimized architecture in phase-3

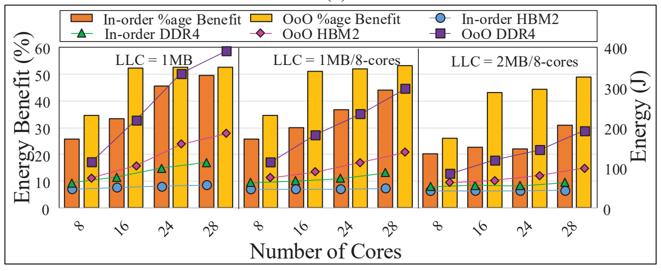
6.1.1 HBM2 vs DDR4

We first look into the performance and energy benefit of using HBM2 instead of DDR4 for Bowtie2 with different core types, core count and LLC size. Figure 5.a shows the performance benefit of using HBM2 instead of DDR4, along with absolute performance in terms of sequencing time. Hence, the baseline architecture for each bar in Fig. 5 is different and is composed of multi-core ARM in-order or OoO cores varying from 8-cores to 28-cores, which operate at 2GHz with DDR4 memory. The LLC for the baselines also varies as fixed 1MB, 1MB/8-core or 2MB/8-cores.

- Architectures with HBM2 always outperform those with DDR4 in terms of performance. The performance benefit is higher for OoO cores compared to in-order cores, as the OoO cores can further exploit the memory BW and, therefore, take more advantage of HBM2. Our results show that OoO cores utilize 2x more memory BW in comparison to in-order cores, due to the speculative multiple issue of instructions in OoO cores.
- The performance benefit of using HBM2 increases with higher core count for in-order cores as the increase in the core count results in a larger stress on memory BW, which HBM2 provides in comparison to DDR4. The BW utilization for HBM2 increases from 7GB/s to 22GB/s when varying the core count from 8 to 28 cores, in comparison to 6GB/s-12GB/s for DDR4.
- For OoO cores, the performance benefit of using HBM2 instead of DDR4 is constant for 16, 24 and 28 cores with fixed LLC size of 1MB and LLC of 1MB/8-core, but it scales with the core count for larger LLC size of 2MB/8-core. The reason being that LLC is a memory BW bottleneck for both HBM2 and DDR4 systems. As OoO cores are stressing the memory to a larger extent, the performance benefits are almost constant for smaller LLC size. However, larger LLC accommodates more search data and helps in alleviating the BW bottleneck problem, as our results show 2x increase in LLC read hits when increasing the LLC size from 1MB to 2MB/8-core. This is also the reason behind the decrease in percentage performance benefit as we increase the LLC size for a given core count. However, lower percentage benefit does not imply lower performance, as we can see in Fig. 5.a that the absolute performance is always better with larger LLC size for a given core count.
- The performance benefit translates linearly into energy benefit, which scales in an identical (but scaled) way as that of performance, with core type, core count and LLC size. Thus, by replacing DDR4 with HBM2 as main memory, we get a performance benefit of up to 50%, as shown in Fig. 5.a, and energy benefit of up to 53% as depicted in 5.b.
- We also ran experiments at 1GHz and 1.5GHz and observed the same scaling trend for both performance and energy that at 2GHz. So, using HBM2 instead of DDR4 bears performance and energy benefits.



(a)



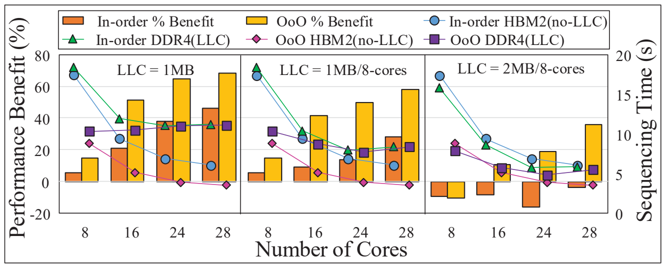
(b)

Fig. 5. Bowtie2 performance and energy benefit of HBM2 vs DDR4 at 2GHz. The bars represent percentage benefits and the points with lines show absolute value. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values.

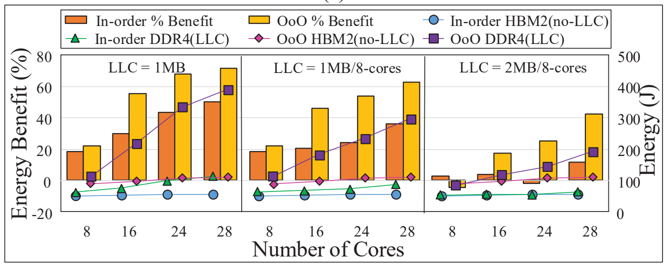
6.1.2 HBM2 (no-LLC) vs DDR4

We explore using HBM2 near to CPU core with no-LLC and compare it to a DDR4 system with LLC for Bowtie2. Figure 6 shows the percentage performance and energy benefit of HBM2 in a system with no-LLC as compared to DDR4 with LLC in the system at 2GHz core frequency. Hence, the baseline system has the same compute cores (in-order or OoO ARM cores) at 2GHz with DDR4 memory and LLC varies from 1MB, 1MB/8-cores to 2MB/8-cores.

- OoO cores with HBM2 and no-LLC are always better than that of DDR4 with LLC both in terms of performance (by up to 68%) and energy (by up to 71.5%), except when the number of cores are 8 and LLC is 2MB. In this configuration, the number of cores is not large enough to fully exploit the memory BW and also the large size of LLC helps in hiding away the latency to the memory, hence, LLC-DDR4 systems performs better. The same trend is true for the energy benefit of OoO cores.
- For in-order cores, the performance and energy benefits increase with the number of cores, except when LLC is larger (i.e. 2MB/8-cores), leading to slightly negative



(a)



(b)

Fig. 6. Bowtie2 performance and energy benefits of near memory compute HBM2 with no-LLC systems in comparison to DDR4 with LLC systems at 2GHz core frequency. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values.

performance benefit. In this case, as in-order cannot stress the memory BW, LLC helps in hiding away the latency to the memory for DDR4 system. However, energy benefits still remain, even if the performance benefits are negative.

- We also ran experiments at 1GHz and 1.5GHz and the scaling trend is the same as that at 2GHz.

As no-LLC HBM2 outperforms DDR4 with LLC, we will further evaluate this memory configuration as well as HBM2 with LLC as it also outperforms DDR4 with LLC, when following further steps of the optimization methodology.

The methodology does not suggest exploring the DDR4 (no-LLC) configuration, as discussed in Section 4.1, since it is the worst performing memory configuration. In any case, as sanity check, we run the experiments for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz. The results show that DDR4 (no-LLC) configuration performs at least 2x slower, as compared to HBM2 (no-LLC) for both in-order and OoO cores. Hence, as suggested by our proposed methodology, we will not be looking into this memory configuration.

6.1.3 Performance-Energy Scaling with Core Count

We explore the scaling of performance and energy with the number of cores, for different core types and cache sizes with HBM2 as main memory. We will not be looking into the scaling results with DDR4 as we already showed in Section 6.1.1 and 6.1.2 that using HBM2 has performance and energy benefits over DDR4. Figure 7 and Fig. 8 show the performance and energy scaling with number of cores at 2GHz, for different configurations.

- There are a number of configurations in Fig. 7 that either match or outperform the performance of state-of-the-art 32 KNL cores operating in turbo boost at 1.5GHz (maximum KNL frequency) with 3D stacked memory, with 1 thread per core. E.g., we can observe that 16 ARM OoO cores at 2GHz surpass the performance of 32 KNL cores at 1.5GHz. We can also observe that 32 ARM in-order cores at

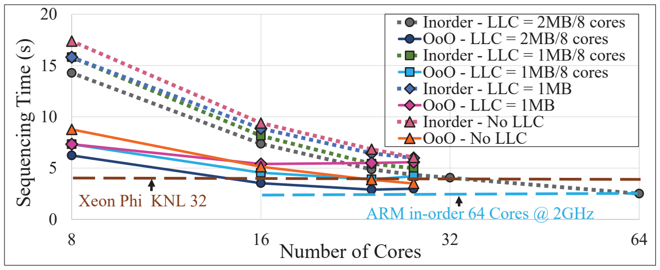


Fig. 7. Bowtie2 performance scaling with number of cores at 2GHz using HBM2 as main memory

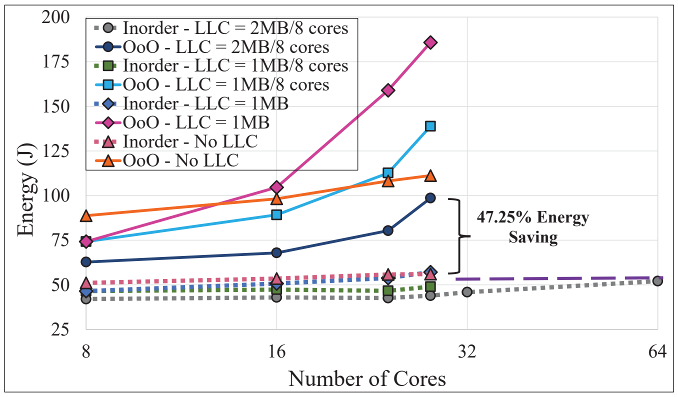


Fig. 8. Bowtie2 energy scaling with number of cores at 2GHz using HBM2 as main memory

2GHz match the performance of 32 KNL cores at 1.5GHz. As we will show in Section 6.1.4, different ARM cores at 1.5GHz also match and outperform 32 KNL cores (operating at 1.5GHz). We choose a comparison point of 32 KNL cores, as opposed to maximum 72 KNL cores, since we are able to capture the performance and energy scaling trends in comparison to 32 ARM OoO cores and 64 ARM in-order cores. In fact, this analysis already shows that this configuration surpasses or matches at least the performance of 32 KNL cores.

- Performance improves when increasing the number of cores, except for OoO cores with LLC. This is because larger core count implies more memory requests through LLC, which gets bottle-necked, leading to performance stagnation. However, we see that systems with HBM2 and no-LLC do not have this bottleneck and therefore, the performance improves with increasing core count. Systems with in-order cores do not exhibit this effect as in-order cores do not generate a lot of memory requests, thus, LLC does not get bottle-necked either.
- Many-core in-order system can match or outperform fewer OoO cores performance, with much lower energy. To illustrate this, we additionally run a simulation of 64 in-order ARM cores, as shown in Fig. 7 and Fig. 8. We can see that 64 ARM in-order cores with LLC of 2MB/8-cores, outperform 28 ARM OoO cores at 2GHz by 16.38%and 32 KNL cores (at 1.5GHz) by 37.5%. This leads to an energy benefit of 47.25% over 28 ARM OoO cores.
- If we consider area, the area of a single OoO core for 28nm CMOS bulk (2.05mm^2) is almost 3 times that of a single in-order core (0.7mm^2) as reported in [28]. So, 64 ARM in-order cores take 23.8% less area when compared to 28 ARM OoO cores. Consequently, 64 ARM in-order cores

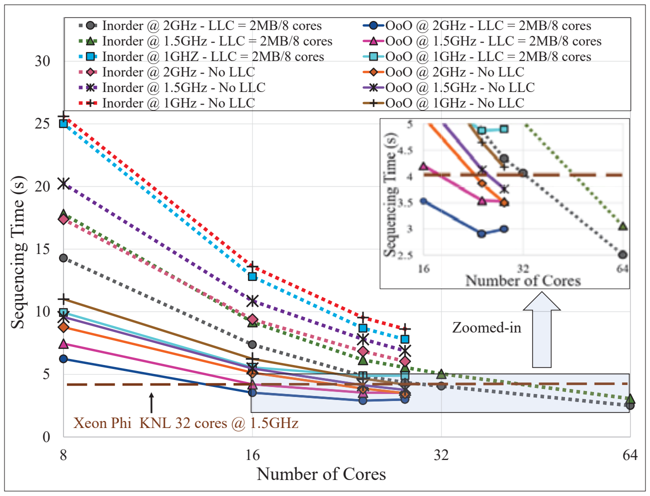


Fig. 9. Bowtie2 performance scaling with frequency for different core types and LLC size

with LLC of 2MB/8-cores is the most efficient architecture in terms of performance, energy and core area.

- In all in-order cores systems (with or without LLC) or OoO systems without LLC, as the performance scales with core count, the increase in the energy consumption is not with the same slope, but with a lower slope, as shown Fig. 8. This implies, we have higher performance gain, with slight increase or almost constant energy.

Many ARMv8 in-order core system with LLC and HBM2 not only outperforms KNL but also fewer ARM OoO core system, in terms of performance, energy and area.

6.1.4 Performance-Energy Scaling with Frequency

We explore the performance and energy scaling with compute core frequencies varying from 1GHz to 2GHz, for different core types and LLC size. For the sizes of LLC, we will consider the extremes, i.e., no-LLC and LLC size of of 2MB/8-cores. The trends for both fixed LLC size of 1MB and LLC of 1MB/8-cores are encapsulated between these two extremes, as at 2GHz in Section 6.1.3, hence, they will not be considered in this section for frequency scaling. Figure 9 shows the performance scaling of different architectures with HBM2, with different frequencies and number of cores. The results are discussed next.

- **Outperforming KNL:** We compare the performance of all the architectures against 32-KNL cores (@ 1.5GHz), and demonstrate that there are many-core ARM 64-bit architectures with HBM2 (for different core types, core count, frequency and LLC size), that can either match or surpass the performance line of 32-KNL cores, as shown in Fig. 9. We compare the performance and energy of all the ARM architectures outperforming KNL in Fig. 10. Firstly, we see that our proposed architectures can match the performance of 32-KNL cores at 1.5GHz with as little as 24 OoO cores at 1.5GHz or 16 OoO cores at 2GHz with LLC of 2MB/8-cores. We also see that 64 in-order ARM cores at 2GHz with LLC of 2MB/8-cores is the best configuration in terms of performance. However, in terms of energy efficiency, 64 in-order ARM cores at 1.5GHz with LLC of 2MB/8-cores is the best with 2.38x less energy when compared to operating at 2GHz.

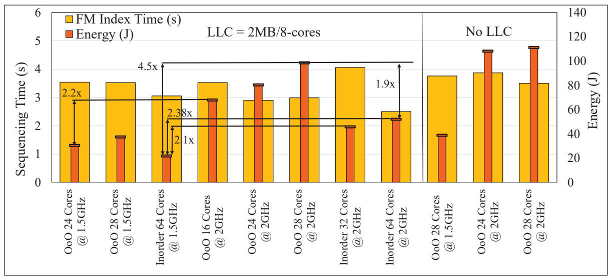


Fig. 10. Bowtie2 performance and energy of all configurations surpassing 32 KNL cores performance

- **Fewer OoO vs. many in-order cores:** We observe that many in-order cores operating at lower frequency can match the performance of fewer OoO cores at higher frequency. E.g., in Fig. 9 28 OoO cores with LLC of 2MB/8-cores 2GHz is outperformed by 16.5% in performance by 64 in-order cores at 2GHz, with energy savings of 1.9x as shown in Fig. 10. Furthermore, if we reduce the operating frequency of 64 in-order cores to 1.5Ghz, it matches the performance of 28 OoO cores at 2GHz, but with energy savings of 4.5x as in Fig. 10. This also results in an area benefit of 23.8%.
- **Performance Stagnation vs No LLC System:** If we look at the zoomed-in section of Fig. 9, we see that systems with OoO cores and LLC have performance stagnated, due LLC not being able to provide enough BW as required by high number of OoO cores, even if HBM2 is the main memory. However, if we remove LLC, and look at the corresponding no-LLC system, the performance always scales with the increase in core count, and no performance stagnation. Therefore, a no-LLC system, is beneficial from scaling perspective for OoO cores as this translates into area savings. But Fig. 10 shows that no-LLC has slightly higher energy demand than corresponding LLC systems. So even though stagnation exist for OoO cores, a system with LLC still performs better in terms of performance and energy when compared to no-LLC system. This stagnation effect is not evident in 64-core in-order system with LLC. Thus, a many-core in-order system with LLC is best in terms of performance, energy, area and scaling.
- **Frequency Scaling based Energy Savings:** Figure 9 and Fig. 10 show that 64 ARM in-order cores at 1.5GHz with LLC can surpass the performance of 32 in-order cores at 2GHz, resulting in an energy saving of 2.1x. Similarly, 24 OoO cores at 1.5 GHz match the performance of 16 OoO cores at 2GHz with energy savings of 2.2x.

Many-core (64-cores) ARMv8 in-order system with LLC and HBM2 is the best performing when compared to KNL and fewer ARMv8 OoO cores. It is also more energy efficient and takes less area in comparison to fewer OoO cores. When targeting energy efficiency it operates best at 1.5GHz. However, if we are targeting performance, the same systems performs best at 2GHz.

6.2 BWA-MEM Architectural Exploration

Following the exploration methodology in Fig. 2 for Bowtie2, we achieved a performance/energy optimized architecture. Similarly, following the methodology, BWA-MEM is the second genome sequencing application that we will use to explore optimal NGS architectures in this section. We launch BWA-MEM in gem5-X FS mode and performed 100K read alignments. We reduced the read alignments in BWA-MEM as it is almost twice as slow as Bowtie2, so 200K read for BWA-MEM was unfeasible in terms of simulation turn around time. Similar to Bowtie2, using the exploration and optimization methodology, we first explore the performance and energy benefits of using HBM2, with and without LLC, as compared to DDR4. Then, we discuss how performance-energy scale with core count and frequency so to have an optimized architecture.

6.2.1 HBM2 vs DDR4

We investigate the performance-energy benefits of using HBM2 instead of DDR4 for BWA-MEM with different core types, core count and frequency as shown in Fig. 11. Hence, the only difference between the baseline and the explored architecture is the use of HBM2 instead of DDR4. The LLC size is fixed at 2MB/8-cores, as we discussed for Bowtie2, that this LLC size has the best performance. As BWA-MEM also uses a similar FM-Index strategy as Bowtie2 with some variation, we use the same LLC size.

- We see that systems with HBM2 achieve up to 6.5% and 10.5% performance benefit for in-order and OoO cores, respectively, when compared to DDR4. The performance benefit scales and increases with the number of cores, as well as with the core frequency.
- The performance benefit translates into energy savings of up to 16% and 18% for in-order and OoO cores, respectively, when using HBM2 instead of DDR4.

Thus, using HBM2 instead of DDR4 is beneficial both in terms of performance and energy for BWA-MEM.

6.2.2 HBM2 (no-LLC) vs DDR4

We explore using HBM2 with no-LLC and compare the energy and performance to a DDR4 system with LLC for BWA-MEM, in accordance with second step of phase-1 in the methodology in Section 4. For BWA-MEM, a no-LLC HBM2 system, as compared to DDR4 with LLC, is inferior in performance and energy efficiency for both in-order and OoO cores, in contrast to Bowtie2.

Therefore, we discard no-LLC HBM2 memory configuration for further phases of the methodology.

We run the experiments for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz, as a sanity check. The results show that DDR4 (no-LLC) configuration performs 28% and 38% slower, as compared to HBM2 (no-LLC) for in-order and OoO cores, respectively. Thus, in accordance with the methodology, DDR4 (no-LLC) configuration will not be explored.

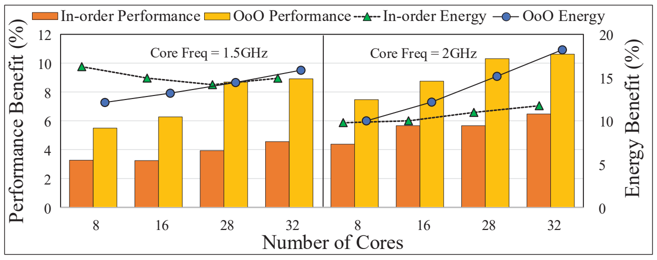


Fig. 11. BWA-MEM performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 2MB/8-cores

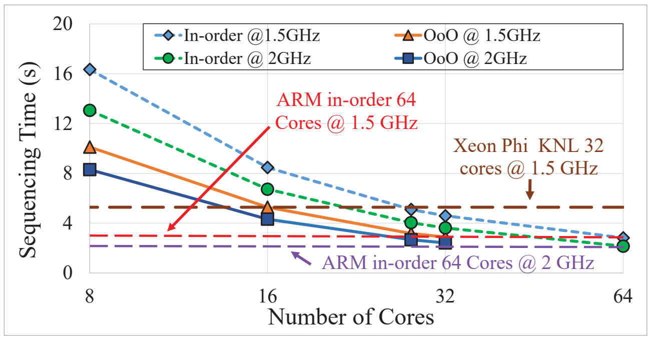


Fig. 12. BWA-MEM performance scaling with core-count and frequency. LLC is fixed at 2MB/8-cores.

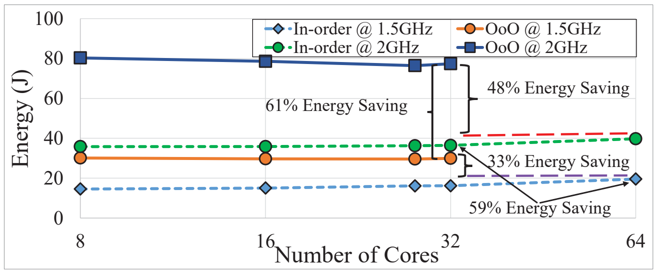


Fig. 13. BWA-MEM energy scaling with core-count and frequency. LLC is fixed at 2MB/8-cores.

6.2.3 Performance-Energy Scaling with Core-Count and Frequency

We look into the performance and energy scaling of BWA-MEM with the number of cores as well as with the core frequency for both in-order and OoO cores in accordance with phase-2 of the methodology. We will not explore the architectures of HBM2 with no-LLC, as we discussed in Section 6.2.2, that these systems are inferior in performance to DDR4 system with LLC. We will also not be looking into DDR4 system with LLC, as we also concluded in Section 6.2.1, that HBM2 system with LLC outperforms DDR4 system with LLC, both in terms of performance and energy. Figure 12 and Fig. 13 show the performance and energy scaling of BWA-MEM, respectively, for different core-types, core frequency and core count with HBM2, as a main memory with LLC of 2MB/8-cores.

- As shown in Fig. 12, the performance of BWA-MEM scales with number of cores as well as with core frequency. We can also see that ARM OoO cores outperform in-order cores for a particular core count. However, many-core in-order system can match or outperform fewer OoO cores system. E.g, 64 in-order cores match the performance of 32 OoO cores, with both operating at the same frequency (1.5GHz/2 GHz). This leads to energy savings, which we will discuss next.
- The energy scaling of BWA-MEM with both frequency and core-count is shown in Fig. 13. As the performance scales with core count, the energy curves are almost flat, indicating a very small increase in energy with the core count. So, 64 in-order cores, outperform 32 OoO cores at 2GHz with energy savings of 48%. For the same comparison at 1.5GHz, the energy saving is 33%. For 28nm

technology node, 64 in-order cores occupy 32% less area when compared to 32 OoO cores.

- Figure 12 also shows that 16 OoO ARM cores at 1.5 GHz match the performance of 32 Intel KNL also at 1.5GHz. If we compare with ARM in-order cores, which are three times smaller in area than ARM OoO, 28 in-order cores match the performance of 32 KNL cores for BWA-MEM.
- Figure 12 shows that 32 OoO cores at 1.5GHz match the performance of 32 OoO cores at 2GHz, giving an energy benefit of 61% as shown in Fig. 13. Also, we see that 64 in-order cores at 1.5GHz surpass the performance of 32 in-order cores at 2GHz resulting in energy savings of 59%.

Overall, for BWA-MEM, 64 in-order ARM cores with HBM2 and LLC is the best architecture in terms of performance, energy efficiency and area, consequently, achieving the same optimized architecture than for Bowtie2.

6.3 HISAT2 Architectural Exploration

Graph based genome sequencing application HISAT2, as discussed in Section 3.4, is the third and final application we studied in the context of energy and performance efficient architecture for genome sequencing applications. HISAT2 was ported to compile for ARMv8 cores. We performed 500K read alignments for HISAT2, in-contrast to 100K for BWA-MEM and 200K for Bowtie2, as HISAT2 is faster than both Bowtie2 and BWA-MEM, giving us around the same simulation turn-around time as for Bowtie2 and BWA-MEM.

6.3.1 HISAT2 Scaling Problem

When HISAT2 is launched either on a simulation platform, like gem5-X, or natively on a server, it scales in multi-threading mode on a multi-core system (1-thread/core), up to a certain thread count, i.e., around 8 ARM cores in gem5-X and around 16 cores on an Intel Xeon server. After that, performance saturates and does not scale with number of cores. This is due to the lack of further parallelization support in HISAT2. To overcome it, we launched multiple instance of HISAT2, each instance using 4 threads (on 4 cores). Therefore, to stress 16 cores, we would launch 4 instances of HISAT2, each using 4 cores, with the number of read alignments equally distributed among them. This does not affect the memory footprint for HISAT2, as it supports using memory-mapped I/O for reference index which many HISAT2 instances can share.

6.3.2 Clustered Architecture

As discussed in the previous section, as HISAT2 scales to a limited number of threads, multiple instances of HISAT2 are launched, each using 4-cores. Hence, we propose a clustered architecture with 4-core clusters, each with its own LLC to prevent cache thrashing. This clustered architecture is shown in Fig. 14. Each cluster connects to a coherent system bus, which ultimately connects to the main memory, which is HBM2 in the figure, but can be DDR4 as well.

The LLC is set to 512KB for each cluster. This gives us 1MB/8-cores, in contrast to 2MB/8-cores for Bowtie2 and BWA-MEM. The reason being that the dataset in HISAT2 is smaller and can fit in a smaller cache. We ran experiments in gem5-X with both 2MB/8-cores and 1MB/8-cores, and observed no difference in performance. Thus, the choice for a smaller cache leads to higher energy and area savings. If

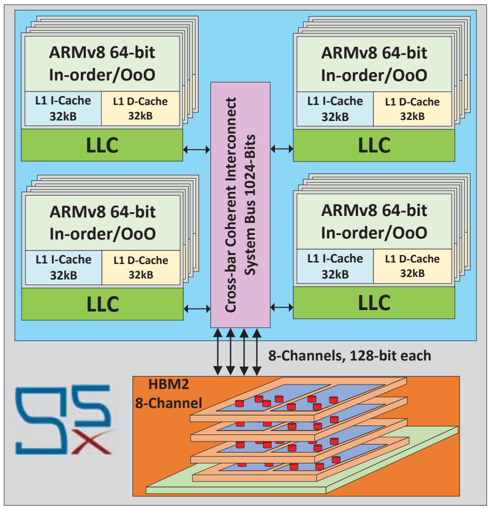

ARMv8 64-bit In-order/OoO
L1 I-Cache 32kB
L1 D-Cache 32kB
LLC
Cross-bar Coherent Interconnect System Bus 1024-Bits
8-Channels, 128-bit each
HBM2 8-Channel

Fig. 14. Clustered architecture with 4-cores per cluster for HISAT2.

we further reduce the cache size, the performance starts to deteriorate.

We will now describe the optimization methodology for HISAT2, as we did earlier for Bowtie2 and BWA-MEM.

6.3.3 HBM2 vs DDR4

With the clustered architecture for HISAT2, we look into the performance and energy benefits of using HBM2 in comparison to DDR4 (both with LLC) for different core types, core count and frequency as shown in Fig. 15. The LLC size is fixed at 512KB/4-cores (1MB/8-cores). Therefore, the baseline multi-core architecture differs from the proposed architectures only at the memory level with DDR4 being used for the baseline.

- For in-order cores, a HBM2 based systems performs up to 12% better when compared to DDR4 based system, and up to 14% better for OoO cores. The performance benefit increases with the number of cores as well as with the core frequency, as shown by the bars in Fig. 15.
- The energy benefit of using HBM2 instead of DDR4 for both in-order and OoO cores is up to 26%.

Therefore, using HBM2 instead of DDR4 is beneficial both in terms of performance and energy for HISAT2, in a system with LLC.

6.3.4 HBM2 (no-LLC) vs DDR4

For HISAT2, HBM2 with no-LLC does not perform better than a DDR4 system with LLC in terms of performance as well as energy efficiency for both in-order and OoO cores, as was the case with BWA-MEM. Thus, no-LLC HBM2 configuration for HISAT2 will be not be further investigated in the optimization methodology.

As a sanity check of the methodology regarding the DDR4 (no-LLC) configuration, we run the experiments for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz. The results show that DDR4 (no-LLC) configuration performs 35% and 38% slower, as compared to HBM2 (no-LLC) for in-order and OoO cores, respectively. Therefore, in accordance with the methodology, DDR4 (no-LLC) configuration will not be explored.

6.3.5 *Performance-Energy Scaling with Core-Count and Frequency*

Figure 16 and Fig. 17 show the performance and energy scaling, respectively, for different core-types, core frequency and core count with HBM2 as a main memory with LLC of 512KB/4-cores. We will not explore the architectures of HBM2 with no-LLC, as we discussed in Section 6.3.4, that these systems are inferior in performance to DDR4 system with LLC. We will also not be looking into DDR4 system with LLC, as we also concluded in Section 6.3.3, that HBM2 system with LLC outperforms DDR4 system with LLC, both in terms of performance and energy.

- Figure 16 shows the performance scaling of HISAT2 with number of cores and core frequency. We can also see that for a given core-count, ARM OoO cores outperform in-order cores. However, many-core in-order system can match the performance of fewer OoO cores system. E.g, 64 in-order cores match the performance of 32 OoO cores, with both operating at the same frequency.
- Figure 17 shows the energy scaling of HISAT2 with both frequency and core-count. As the performance scales with core count, the energy curves are almost flat, indicating a very small increase in energy with the core count. As discussed above, 64 in-order cores match the performance of 32 OoO cores at 2GHz and giving energy savings of 2x. Similarly at 1.5GHz, 64 in-order cores consume 39% less energy than 32 OoO cores, and at the same time matching performance. For 28nm technology node, 64 in-order cores occupy 32% less area when compared to 32 OoO cores.
- Furthermore, we compare the performance of ARM based architectures with Intel KNL. Figure 16 shows that 16 OoO ARM cores at 1.5GHz match the performance of 32 Intel KNL also at 1.5GHz. If we compare with ARM in-order cores, which are three times smaller in area than ARM OoO, 28 in-order cores at 1.5 GHz match the performance of 32 KNL cores for HISAT2.
- Finally, we compare the performance of the same system configurations at different frequencies. Figure 16 shows that 64 in-order cores at 1.5GHz match the performance of 32 in-order cores at 2GHz resulting in energy savings of 51%, as shown in Fig. 17. We also see that 32 OoO cores at 1.5GHz surpass the performance of 16 OoO at 2GHz cores giving us an energy benefit of 61%.

Overall, in the case of HISAT2, a system with 64 in-order ARM cores with HBM2 and LLC is the best architecture in terms of performance, energy efficiency and area.

6.4 Discussion

After following the optimization methodology for three case study applications representing the NGS genome sequencing applications, we can draw the following conclusions;

1) In all the three applications, systems with HBM2 as main memory outperformed their counterparts with DDR4, both in terms of performance and energy efficiency. This leads to a performance benefit of 50%, 10.5% and 14% for Bowtie2, BWA-MEM and HISAT2, respectively, with corresponding energy savings of up to 53%, 18% and 26%, respectively. This is due to the fact that the three NGS applications are memory bounded and, therefore, benefit from high memory BW of up to 307.2 GB/s [27], which is provided by HBM2. We also observe that the performance and energy benefits of using HBM2 are more for OoO cores compared to in-order, as OoO cores can exploit more BW and therefore, take more advantage of HBM2.
2) For no-LLC HBM2 systems compared to LLC with HBM2 or DDR4, both BWA-MEM and HISAT2 showed that LLC with HBM2 or DDR4 performed better than no-LLC HBM2 in terms of performance and energy. As discussed previously, HBM2 with LLC was better than DDR4 with LLC. Hence, HBM2 with LLC was selected as the optimized memory sub-system for both BWA-MEM and HISAT2. In case of Bowtie2, no-LLC HBM2 showed performance and energy benefits of up to 68% and 71.5%, respectively, as compared to DDR4 with LLC. Moreover, no-LLC HBM2 outperformed LLC with HBM2, when the LLC is small (1MB/8-core). However, if the LLC is large (2MB/8-cores), LLC with HBM2 was the best performing architecture in terms of performance and energy for Bowtie2.
3) Both ARM OoO and in-order cores outperformed HPC class Intel KNL with similar 3D-stacked memory as HBM2

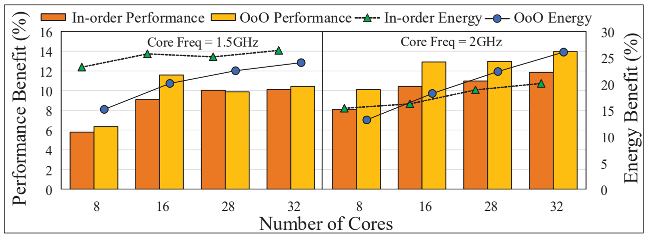


Fig. 15. HISAT2 performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 512KB/4-cores

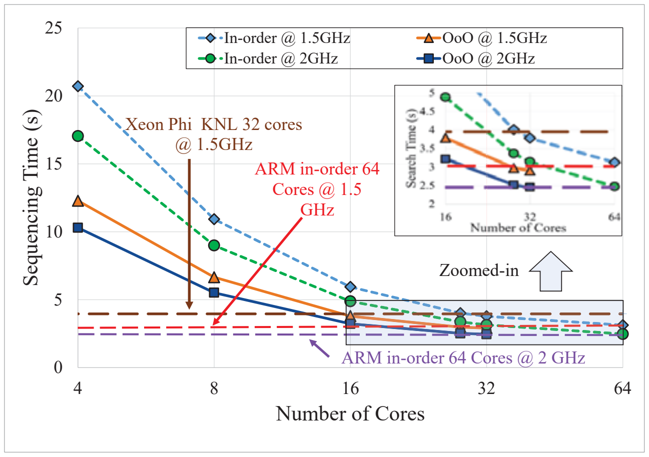


Fig. 16. HISAT2 performance scaling with core-count and frequency. LLC is fixed at 512KB/4-cores.

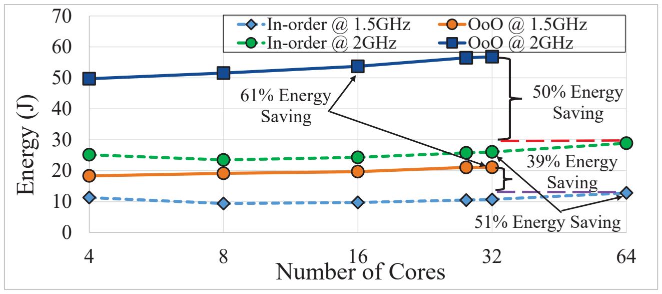


Fig. 17. HISAT2 Energy scaling with core-count and frequency. LLC is fixed at 512KB/4-cores.

(used in proposed ARM systems), with fewer or similar core counts at the same core frequency of 1.5 GHz, for all the three applications. This is due to the fact that as the applications are memory bounded, high performing Intel KNL cores do not help in improving the performance. Hence, energy efficient ARM cores coupled with HBM2 surpass the performance of Intel KNL.

4) For a given system, based on either ARM in-order or OoO cores, using frequency and core count scaling, we showed that a system operating at a lower operating frequency can either match or surpass the performance of that at higher frequencies with energy savings of up to 55% (2.2x) for Bowtie2 and 61% for both BWA-MEM and HISAT2, when using OoO cores. Similarly, for in-order cores, we got an energy benefit of up to 52% (2.1x), 59% and 51% for Bowtie2, BWA-MEM and HISAT2, respectively.
5) Many energy-efficient ARM in-order cores either matched or outperformed fewer (almost half) high performance ARM OoO cores at the same operating frequency with energy savings of up to 47.5%(1.9x), 48% and 50% for Bowtie2, BWA-MEM and HISAT2, respectively. This is due to the memory bounded nature of the applications, which do not benefit from high performance cores. As discussed, since in-order are almost 3x smaller in area than OoO, it also gave us core-area savings of 23.8% for Bowtie and 32% for both BWA-MEM and HISAT2. Furthermore, in case of Bowtie2, not only in-order cores surpass the performance of OoO cores, but that also at a lower operating frequency, resulting in 4.5x energy savings.
6) HISAT2 application showed performance scaling up to 8 cores in the system. To alleviate this problem, we proposed a clustered system with an instance of HISAT2 on each cluster (4-core cluster with 512KB LLC), and the read alignments distributed among all clusters.

Therefore, many-core (64 cores in our case) ARM in-order system with HBM2 and LLC was the overall best performing system for all three applications in terms of performance, energy efficiency and area.

7 Conclusion

In this paper, we have proposed an architectural exploration and optimization methodology to optimize heterogeneous computing architectures targeting genome sequencing alignment. Using the gem5-X architectural simulator and the proposed optimization methodology, we explored architectures to optimize the performance and energy of the system for three genome sequence alignment applications i.e., Bowtie2, BWA-MEM, and HISAT2. All these applications are memory bounded with random memory access. Our analysis in this work has shown such memory bounded workloads do not require power hungry HPC compute nodes like Intel Xeon Phi KNL, but instead require improvements in memory BW to enhance the overall performance and energy. Furthermore, we have proven that by using high BW memories like HBM2 coupled with energy efficient compute cores, NGS applications achieve up to 68% performance and 71% energy benefit compared to a traditional system with DDR4. We also demonstrated that a variety of architectures based on ARMv8 in-order and OoO cores with LLC and HBM2 outperform 32-core Intel KNL processor. Moreover, we have shown that by using frequency scaling we can achieve up to 59% energy savings for ARM in-order cores and up to 61% for OoO cores. Lastly, we have highlighted that up to 4.5x energy savings can be achieved using many simple in-order cores, instead of fewer complex OoO cores. Thus, we do not require power hungry HPC class resources like KNL for efficiently executing NGS applications, but rather simple many-core ARM in-order architectures with HBM2. These newly proposed ARM-based many-core architectures perform much better in terms of both performance and energy efficiency.

Acknowledgments

This work has been partially supported by the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657), the EC H2020 WiPLASH (GA No. 863337), the EC H2020 RECIPE (GA No. 801137), the EU FEDER and the Spanish MINECO (GA No. RTI2018-093684-B-I00), the Spanish CM (S2018/TCS-4423), Spanish MINECO TIN2016-80920-R, JA2012 P12-TIC-1470 and UMA18-FEDERJA-197 projects.

References

[1] Data centres and data transmission networks tracking clean energy progress. [Online]. Available: https://www.iea.org/tcep/buildings/datacentres/
[2] M. Radulovic *et al.*, "Mainstream vs. emerging HPC: Metrics, trade-offs and lessons learned," in *SBAC-PAD*, 2018, pp. 250–257.
[3] N. Rajovic *et al.*, "Supercomputing with commodity cpus: Are mobile socs ready for HPC?" in *SC*, 2013, pp. 1–12.
[4] Z. Ou *et al.*, "Energy- and cost-efficiency analysis of ARM-based clusters," in *CCGRID*, 2012, pp. 115–123.
[5] A. Pahlevan *et al.*, "Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or not?" in *DATE*, 2018, pp. 147–152.
[6] B. Schmidt and A. Hildebrandt, "Next-generation sequencing: big data meets high performance computing," *Drug Discovery Today*, pp. 712 – 717, 2017.
[7] J. S. Black *et al.*, "The impact of next generation sequencing technologies on haematological research – a review," *Pathogenesis*, pp. 9 – 16, 2015.
[8] "Whole genome of the Wuhan coronavirus, 2019-nCoV, sequenced," 2020. [Online]. Available: www.sciencedaily.com/releases/2020/01/200131114748.htm
[9] "Institut Pasteur isolates strains of coronavirus 2019-nCoV detected in France," 2020. [Online]. Available: https://pasteur.fr/en/press-area/press-documents/institut-pasteur-isolates-strains-coronavirus-2019-ncov-detected-france
[10] H. Li *et al.*, "A survey of sequence alignment algorithms for next-generation sequencing," *Brief Bioinform.*, pp. 473–483, May 2010.
[11] K. Hsieh *et al.*, "Accelerating pointer chasing in 3D-stacked memory: Challenges, mechanisms, evaluation," in *ICCD*, 2016, pp. 25–32.
[12] A. M. Maxam *et al.*, "A new method for sequencing dna," *Proceedings of the National Academy of Sciences*, pp. 560–564, 1977.
[13] F. Sanger *et al.*, "Dna sequencing with chain-terminating inhibitors," *Proceedings of the National Academy of Sciences*, pp. 5463–5467, 1977.
[14] "Finishing the euchromatic sequence of the human genome," *Nature*, vol. 431, no. 7011, pp. 931–945, Oct. 2004.
[15] K. V. Voelkerding *et al.*, "Next-Generation Sequencing: From Basic Research to Diagnostics," *Clinical Chemistry*, pp. 641–658, 04 2009.
[16] D. A. Wheeler *et al.*, "The complete genome of an individual by massively parallel DNA sequencing," *Nature*, pp. 872–876, 04 2008.
[17] R. Appuswamy *et al.*, "Sequence alignment through the looking glass," Jan. 2018.
[18] J. M. Herruzo *et al.*, "Accelerating sequence alignments based on fm-index using the intel KNL processor," *TCBB*, pp. 1–1, 2018.
[19] P. Ferragina *et al.*, "Opportunistic data structures with applications," in *FOSC*, 2000, pp. 390–398.
[20] B. Langmead *et al.*, "Fast gapped-read alignment with bowtie 2," *Nature Methods*, pp. 357–359, Mar 2012.
[21] H. Li *et al.*, "Fast and accurate long-read alignment with burrows–wheeler transform," *Bioinformatics*, pp. 589–595, Jan. 2010.

[22] D. Kim *et al.*, "Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype," *Nature Biotechnology*, vol. 37, no. 8, pp. 907–915, Aug. 2019.

[23] A. Sodani *et al.*, "Knights landing: Second-generation intel xeon phi product," *IEEE Micro*, pp. 34–46, Mar 2016.

[24] X. Li *et al.*, "Improving the thread scalability and parallelism of bwa-mem on intel hpc platforms," in *HPCC*, 2019, pp. 1858–1865.

[25] R. Wilton *et al.*, "Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space," *PeerJ*, p. e808, Mar. 2015.

[26] J. González-Domínguez *et al.*, "Parallel and scalable short-read alignment on multi-core clusters using UPC++," *PLOS ONE*, p. e0145490, Jan. 2016.

[27] K. Sohn *et al.*, "A 1.2 v 20 nm 307 GB/s HBM DRAM with at-speed wafer-level io test scheme and adaptive refresh considering temperature distribution," *JSSC*, pp. 250–260, Jan 2017.

[28] Y. M. Qureshi *et al.*, "Gem5-X: A gem5-based system level simulation framework to optimize many-core platforms," in *2019 SpringSim*, April 2019, pp. 1–12.

[29] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, pp. 1–7, Aug. 2011.

[30] B. Langmead *et al.*, "Scaling read aligners to hundreds of threads on general-purpose processors," *bioRxiv*, 2018.

[31] R. Luo *et al.*, "SOAP3-dp: Fast, accurate and sensitive GPU-based short read aligner," *PLoS ONE*, vol. 8, no. 5, p. e65632, May 2013.

[32] M. Anderson *et al.*, "Considerations when evaluating microprocessor platforms," in *USENIX HotPar'11*, USA, 2011, p. 1.

[33] A. Khajeh-Saeed *et al.*, "Acceleration of the smith–waterman algorithm using single and multiple graphics processors," *Journal of Computational Physics*, vol. 229, no. 11, pp. 4247–4258, Jun. 2010.

[34] D. Fujiki *et al.*, "Genax: A genome sequencing accelerator," in *ISCA*, 2018, pp. 69–82.

[35] K. Koliogeorgi *et al.*, "Dataflow acceleration of smith-waterman with traceback for high throughput next generation sequencing," in *FPL*, 2019, pp. 74–80.

[36] M. Jaspers, "Acceleration of read alignment with coherent attached fpga coprocessors," *Master Thesis, TU Delft*, 2015.

[37] N. A. Miller *et al.*, "A 26-hour system of highly sensitive whole genome sequencing for emergency management of genetic diseases," *Genome Medicine*, vol. 7, no. 1, Sep. 2015.

[38] Illumina, "Accuracy improvements in germline small variant calling with the dragentm platform." [Online]. Available: http://www.albiogen.by/upload/documents/DRAGEN%20Accuracy%20Application%20Note.pdf

[39] R. Ben Abdelhamid *et al.*, "A block-based systolic array on an hbm2 fpga for dna sequence alignment," in *ARC*, 2020, pp. 298–313.

[40] J. S. Kim *et al.*, "GRIM-filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies," *BMC Genomics*, vol. 19, no. S2, May 2018.

[41] ARM, "ARM vexpress juno r2 development platform," 2015.

[42] S. Lee *et al.*, "Leveraging power-performance relationship of energy-efficient modern DRAM devices," *IEEE Access*, pp. 31 387–31 398, 2018.

[43] M. O'Connor *et al.*, "Fine-grained dram: Energy-efficient DRAM for extreme bandwidth systems," in *MICRO*, 2017, pp. 41–54.

[44] M. Holtgrewe, "Mason - A read simulator for second generation sequencing data," Freie Universitaet Berlin, Tech. Rep. 962, 2010.

[45] "Parus_major1.0.3 - Genome - Assembly - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/assembly/GCF_001522545.1/

Yasir Mahmood Qureshi received his Master degree in Embedded Computing Systems from University of Southampton, UK and NTNU, Trondheim, Norway in 2013. He is currently a Ph.D. student at the Electrical Engineering Doctoral program, in Embedded Systems Laboratory, EPFL. His research interests are energy efficient computing, heterogeneous compute and hybrid memory architectures.

Jose M. Herruzo received his B.S. and M.S. degrees in Computer Engineering from University of Cordoba in 2014 and from the University of Malaga in 2015, respectively. He is currently working toward the PhD degree at the University of Malaga. His current research interests include processing in memory, near-data processing, stacked memory architectures, high-performance computing and high-throughput sequence alignment.

Marina Zapater is Associate Professor at the University of Applied Sciences Western Switzerland (HES-SO) since 2020, and external collaborator at ESL in EPFL, Switzerland, since 2016. She received her Ph.D. degree in Electronic Engineering from Universidad Politécnica de Madrid, Spain, in 2015. Her research interests include power and performance design and optimization, from edge to high-performance computing architectures.

Katzalin Olcoz received a Ph.D. degree in Physics in 1997 from the Complutense University (UCM) of Madrid. She is Associate Professor in the Department of Computer Architecture and System Engineering of the Complutense University. She was a visiting professor at EPFL (Lausanne, Switzerland) from April to June, 2018. Her research interests include high performance computing, energy efficiency and virtualization.

Sonia González-Navarro received the BS and MS degrees in mathematics, in 2000 and the PhD degree in computer science in 2006, both from the Univeristy of Málaga, Spain. She is currently an associate professor in the Computer Architecture Department at the University of Málaga. Her research interests include computer arithmetic, floating point number computation, high-performance architectures for data-intensive applications, near-data processing.

Oscar Plata received a Ph.D. in Physics in 1989 from the University of Santiago de Compostela, Spain. He started as Assistant Professor in the same University where he became Associated Professor in 1990. He moved to the University of Malaga in 1995, where he became Full Professor in the Computer Architecture Dept. in 2002. His research interests are related to high performance computing and parallel architectures. He was Associated Editor of IEEE Transactions of Computers from 2015 to 2019.

David Atienza (M'05-SM'13-F'16) is an associate professor of electrical and computer engineering, and head of the Embedded Systems Laboratory (ESL) at EPFL, Switzerland. He received his Ph.D. in computer science and engineering from UCM, Spain, and IMEC, Belgium, in 2005. His research interests include system-level design methodologies for multi-processor system-on-chip (MPSoC) servers and edge AI architectures. He has co-authored more than 350 papers, one book, and 12 patents. Dr. Atienza has received among other recognitions the ICCAD 10-Year Retrospective Most Influential Paper Award in 2020, the Most Influential DAC Under-40 Innovators Award in 2018, and an ERC Consolidator Grant in 2016. He is an IEEE Fellow and ACM Distinguished Member.