Survey of Hardware Acceleration of Genomic Analysis

Zhuren Liu†, Shouzhe Zhang†, Hui Zhao
Department of Computer Science and Engineering, University of North Texas
{zhurenliu, shouzhezhang}@my.unt.edu, {hui.zhao}@unt.edu
†Both authors contributed equally to this work.

***Abstract*—As the Next-Generation Sequencing (NGS) techniques need to process enormous amounts of data, cost-efficient and high-throughput computational analysis is essential in genomics study. Conventional computing platforms face great challenges to meet these demands due to their limited processing speed and scalability. Hardware accelerators, such as Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs), offer transformative solutions to these computational challenges. This paper provides a state-of-the-art review of the roles of hardware accelerators in genomic analysis. We performed a comprehensive and in-depth analysis of cutting-edge genomics hardware accelerators, such as GPUs, FPGAs, and ASICs, in the context of the specific algorithms they aim to enhance. Besides reviewing opportunities in hardware genome acceleration, we also provide insights into the challenges regarding processing speed, cost efficiency, and scalability.**

***Index Terms*—genome analysis, genome sequencing, hardware accelerator, GPU, FPGA, ASIC, accelerated computing.**

1. Introduction

Genome analysis is the study of an organism's DNA sequence. This technique has numerous vital applications, including tracing pandemic outbreaks, early cancer detection [1], disease cause recognition [2], medication development [3], and protein identification [4]. Next-generation technologies, such as sequencing by synthesis (SBS) [5], Single Molecule Real-Time (SMRT) [6], and nanopore sequencing [7], enable faster and cheaper DNA and RNA sequencing [8]–[10], resulting in an exponential increase in genetic data. Despite these developments, however, genomic data analysis remains difficult; one of the major challenges is the inability to read an organism's entire genome [11]. Therefore, sequencing machines can only extract a small fragment of the organism's DNA sequence, and this is known as short-read alignment [12].

A genome analysis pipeline involves a series of steps that affect the precision, swiftness, and energy efficiency of genome analysis [13]. Such steps include: 1) Basecalling: translating the raw sequencing data produced by HTS into genomic characters such as A, C, G, and T in DNA; 2) Real-time analysis: Known as reads, analyzes the raw sequencing data in real-time when they are being sequenced using specific sequencing technology, such as Nanopore [7] sequencing; 3) Read mapping: identifying similarities and differences between genomic sequences, e.g., between sequenced reads and reference genomes of one or more species. This stage comprises several steps, including indexing, seeding, and alignment; 4) The subsequent steps of further genome analysis.

Figure 1 (a) illustrates the genome analysis pipeline. The pipeline contains two key steps: basecalling and read mapping. Some state-of-the-art basecallers use deep neural networks for translating raw signals into nucleotide bases, e.g., Bonito [14]. The basecaller needs to return a read quality score based on the accuracy of the translation of each nucleotide base. To start with, the basecaller first needs to split the long read from the raw data into smaller fragments before basecalling these fragments. Next, the basecaller needs to reassemble them back into a long read sequence [15].

Read mapping maps the reads from basecalling to a reference genome. Some state-of-the-art read mappers, such as Minimap2 [16], contain at least 3 phases. As shown in Figure 1. ①Indexing allows queries to find matches between a read and reference genome. During indexing, minimizers [17], [18] generated from the reference genome are inserted into a key-value hash table. In this table, minimizers serve as the keys, while their locations in the reference genome contain the values [15]. ②Seeding generates minimizers from a basecalled read, then it searches the generated minimizers in the hash table to find matching regions between the read and reference genome. ③Sequence alignment identifies the similarity between the read and each candidate region in the reference genome. It uses an alignment score to represent the difference between the two sequences. Sequence alignment generally uses a computationally expensive dynamic programming (DP) algorithm [15] to perform approximate string matching (ASM) between two sequences. Figure 1 (b) describes the different approaches to speed up each step of read mapping.

Since software techniques alone are not efficient enough to deal with large amounts of genome data, hardware techniques have been explored to provide solutions for these challenges. Specialized hardware, such as graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs), can dramatically decrease the processing time and improve the efficiency of genomic data analysis. These specialized hardware are well known for their parallel processing capabilities compared with conventional central processing units (CPUs). Their high throughput, multitasking, and parallel processing capabilities work well with the data-intensive and repetitive activities common in genomics, such as variant calling, genome assembly, and sequence alignment. A comprehensive comparison

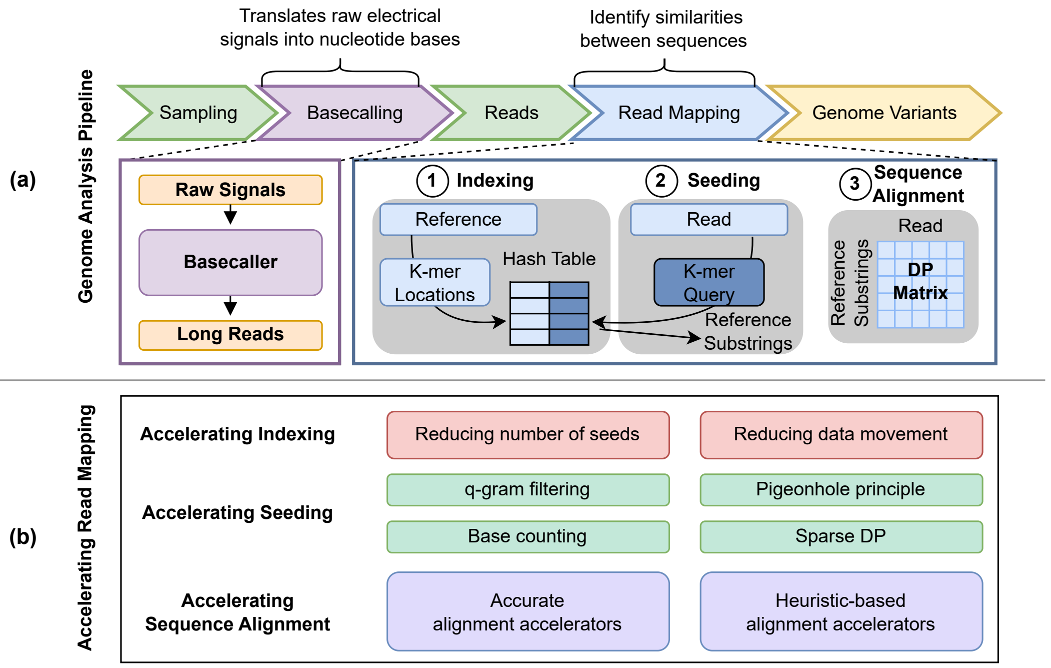

Fig. 1. (a) The typical pipeline of Genome analysis. The basics steps of Basecalling. Three steps of Read Mapping: ① Indexing, ② Seeding, and ③ Alignment. (b) Overview of approaches to accelerate read mapping [11]

between different accelerators, detailing their architectural features, advantages, constraints, processing power (FLOPS), price, and energy, is shown in Table 1 [19]–[21].

This paper aims to explore the current state, challenges, and future research trends of hardware acceleration in genomic analysis. We will examine several hardware acceleration solutions, evaluating their effectiveness, adaptability, and cost-effectiveness through a thorough review of current state-of-the-art research. This survey seeks to highlight the potential of hardware-accelerated genomics study to help researchers develop a perspective of the area and evaluate future trends. The paper is structured as follows: Sections 2, 3, and 4 review research on accelerating genome analysis with GPUs, FPGAs, and ASICs, respectively. In these sections, we provide reviews of specific hardware accelerators, categorizing them according to the phases of read mapping. Finally, Section 5 presents a discussion about our perspectives. Table 2 lists the hardware-accelerated genome analysis tools covered in this survey.

2. Accelerating Genome Analysis using GPUs

GPUs have been widely used in accelerating parallel applications [22]. GPU-based frameworks can significantly reduce the execution time for genomic analysis [23]. General Purpose Graphics Processing Units (GPGPUs) are a type of general-purpose accelerator that is very efficient in executing SIMD commands. For instance, DeepVariant, a convolutional neural net-based variant caller, has a speedup of over 80x than CPUs when running certain tasks on GPU-accelerated platforms [24]. In this section, we review the prior work of accelerating genomic analysis using GPUs.

2.1 Accelerating Indexing

Zhang et al. [25] evaluated BaseNumber, a GPU-accelerated tool for germline variant calling using whole-genome sequencing (WGS) data. The study showed that BaseNumber can identify genetic variants with increased speed and efficiency. To address the growing demand for high-throughput software when aligning large numbers of short DNA reads, Liu et al. proposed SOAP3 [26], which is the first GPU implementation of compressed index data known as Burrows-Wheeler Transform (BWT). SOAP3 tackles challenges such as reducing the memory accesses to the index from individual threads (cores) and avoiding idle time by controlling the branch and divergence of the threads. Masher [27] is a tool that streamlines read mapping on long genomics, leveraging a hash-based indexing strategy implemented on GPUs. It addresses the computational difficulties that arise from the increasingly complex and lengthy genomic data. Masher uses a hash-based indexing framework. It enables fast look-up and alignment of genomic sequences, significantly speeding up the mapping process for reads longer than 500.

In an effort to optimize the alignment of biological sequences, Franklin et al. [28] optimized the BLAST algorithm. Their algorithm consists of a heuristic seed indexing phase and an extension phase with sequence comparison. They proposed a two-dimensional matrix as a hash table and leveraged GPU's parallel capability to process the hash table. Encarnaccao et al. [29] utilized suffix array-based indexing to accelerate DNA sequence alignment on GPUs. They modified the construction and search algorithms of suffix arrays to enable parallel execution on GPUs. This approach greatly improves the speed and effectiveness of DNA searches.

TABLE 1
COMPARISON OF HIGH-PERFORMANCE COMPUTING ARCHITECTURES

Type	Description	Advantages	Disadvantages	FLOPS	Price	Power
GPU	Graphics Processing Units designed for parallel processing and computational acceleration.	Highly parallel processing capabilities, ideal for machine learning, simulations, and rendering tasks.	High cost and require specialized programming knowledge.	4000+ TFLOPS	\$500 - \$10,000+	100W - 300W
FPGA	Field-Programmable Gate Arrays are reconfigurable integrated circuits that can be programmed for specific tasks.	Reconfigurable, offering flexibility and high performance for specific applications.	Complex to program and reconfigure, and may not match the performance of dedicated hardware for certain tasks.	10+ TFLOPS	\$100 - \$5,000+	10W - 100W
ASICs	Custom-built integrated circuits designed for specific applications or tasks.	Highly efficient for their intended tasks, offering superior performance and lower power consumption.	High development costs and lack of flexibility once manufactured.	1000+ TFLOPS	\$10,000+	10W - 100W

2.2 *Accelerating Sequence Alignment*

Sequence alignment is the process of finding the best alignment of a pair of DNA sequences, RNA sequences, or proteins. It is one of the most fundamental tasks in genomic analysis and one of the most compute-intensive kernels for genomic data analysis, taking up over 90% of the overall time for bioinformatics workloads [30]. In recent years, GPUs have become widely used to speed up sequence alignment, achieving significant performance improvement and energy reduction.

Ren et al. [31] tackle the NGS data problem by presenting a GPU-accelerated version of the GATK HaplotypeCaller (GATK HC), a commonly used DNA variant caller. They also proposed a load-balanced multi-process optimization to overcome the limitations that force sequential execution of the program. Taylor et al. [32] focused on the scalability of genomic analyses using GPUs. They showed that high efficiency at low cost can be achieved using general-purpose libraries like PyTorch and TensorFlow on GPUs, demonstrating significant improvement in runtime and cost-efficiency compared to CPU-based processing.

GASAL2 [33] is a specialized GPU sequence alignment library for aligning DNA and RNA sequences, offering various alignment kernels such as local alignment, global alignment, and semi-global alignment. Genomics-GPU [23] extends these kernels by incorporating 10 widely used genomic analysis applications into one benchmark suite, covering genome comparison, matching, and clustering for both DNAs and RNAs. It also created updated versions of these applications to leverage CUDA Dynamic Parallelism (CDP) to further improve parallel execution throughput.

Multiple works have been proposed on the optimization of the Smith-Watersman (SW) [34] algorithm on GPUs. The Smith-Watersman algorithm is known for its precision in sequence alignment, and it is very computationally intensive. This is a dynamic programming algorithm that can provide conserved regions between two sequences and align two partially overlapping sequences [23], [34], [35]. The rapid growth in genomic data makes it necessary to develop efficient computational techniques tailored to the Smith-Watersman algorithm. CUDAlign [36] tackled this problem by focusing on Megabase Genome Sequencing Acceleration using GPUs. It optimizes the Smith-Waterman algorithm to utilize the parallel computing capabilities on GPUs. Sandes et al. [37] then proposed CUDAlign 2.1, which focuses on the Smith-Waterman algorithm for aligning megabase-sized biological sequences using GPU. They observed that they could achieve up to 41.64× speedup when comparing GPU with the Z-align cluster solution [38], a CPU cluster solution for large sequences pairwise alignments. CUDASW++ 3.0 [39] further improved CUDAlign by presenting a hybrid approach to enhance the performance of the Smith-Waterman algorithm by coupling the SIMD (Single Instruction, Multiple Data) instructions on CPUs and GPUs. CUDAlign 4.0 [40] aims to address the exact chromosome-wide alignment, one of the most challenging aspects of genomic sequence analysis, by introducing an incremental speculative traceback mechanism that allows for the handling of large datasets.

Wang et al. [41] improved the transfer and processing of data on the GPU for sequence alignment by identifying the communication bottlenecks between the CPU and GPU and showcases using register shuffle in sequence alignment algorithms, such as SW and Pairwise-Hidden-Markov-Model (PairHMM) [42]. Guo et al. [43] focused on the detection of overlaps between any pair of input reads. They modify Minimap2 [16], [44], the state-of-the-art overlapping tool, to run on both GPUs and FPGAs. To further improve Minimap2 on hardware, Feng et al. [45] proposed manymaps. They redesigned the memory layouts of dynamic programming (DP) matrices to eliminate intra-loop data dependency in the base-level alignment step to accelerate Minimap2 on accelerators.

Striemer et al. [46] explored using GPUs to accelerate the SW algorithm. They observed performance bottlenecks in memory bandwidth, data parallelism, and algorithmic complexity. To remove these bottlenecks, they proposed several GPU hardware design techniques, such as optimizing memory access patterns, employing parallel schedulings, and refining sequence alignment algorithms for better GPU compatibility. MR-CUDASW is another work proposing a multi-round approach utilizing CUDA that focuses on medium-length genomic and metagenomic data to accelerate the SW algorithm [47] on GPUs. It utilizes a GPU's parallel processing capabilities by optimizing memory usage and computational efficiency for medium-length nucleotide reads from modern sequencing machines. There is another highly optimized version of the Smith-Waterman algorithm proposed by Korpar et al. named SW# [48] to enable GPU acceleration of the SW algorithm.

A scalable GPU system developed for computing pairwise Whole Genome Alignments (WGA) is called SegAlign [61].

TABLE 2
GENOME ANALYSIS WITH DIFFERENT TOOLS

Name	CPU	GPU	FPGA	ASICs	Method	Speedup	Reference
H/S Co-design by Lo et al. [49]	-	-	✓	-	Accelerating Base Quality Score Recalibration (BQSR)	40.7×	[49]
Helix	-	-	-	✓	Accelerating nanopore base calling	6×	[50]
PLEDGER	✓	✓	-	-	Whole genome read mapping	11×	[51]
CUDAlign	-	✓	-	-	Accelerating Smith-Waterman algorithm	-	[36]
CUDAlign 2.1	-	✓	-	-	Accelerating Smith-Waterman algorithm	-	[37]
CUDASW++ 3.0	-	✓	-	-	Accelerating Smith-Waterman algorithm	-	[39]
CUDAlign 4.0	-	✓	-	-	Accelerating Smith-Waterman algorithm	-	[40]
BarraCUDA	-	✓	-	-	Based off BWT	6×	[52]
SOAP3-dp	✓	✓	-	-	Based off BWT	10×	[26]
CUSHAWGPU	-	✓	-	-	Based off BWT	-	[53]
GPU based BWT	-	✓	-	-	Based off BWT	12×	[54]
SARUMAN	-	✓	-	-	Based on hash table	5×	[55]
nvBowtie	-	✓	-	-	NVIDIA's CUDA implementation of the Bowtie 2 algorithm	-	[56], [57]
CUDA-BLASTP	-	✓	-	-	Lightweight BLASTP for short queries	10×	[58]
G-BLASTN	-	✓	-	-	GPU-accelerated software tool for BLAST	14.8×	[59]
SW#	-	✓	-	-	Based off Smith-Waterman algorithm	-	[48]
MUMmerGPU 2.0	-	✓	-	-	Based off suffix tree	4×	[60]
MR-CUDASW	-	✓	-	-	Based off Smith-Waterman algorithm	1.17×	[39]
SegAlign	-	✓	-	-	Pairwise Whole Genome Alignments (WGA)	14×	[61]
FPGA design by Tang et al. [62]	✓	-	✓	-	Accelerating short reads mapping	42.9×	[62]
FPGA design by Chen et al. [63]	-	-	✓	-	Accelerating long read	48×	[63]
FHAST	-	-	✓	-	Accelerating Bowtie	70×	[64]
FPGA design by Zhang et al. [65]	-	-	✓	-	Accelerating Smith-Waterman algorithm	250×	[65]
FPGA design by Benkrid et al. [66]	-	-	✓	-	Accelerating Smith-Waterman and Needleman–Wunsch algorithms	-	[66]
FPGA design by Yamaguchi et al. [67]	-	-	✓	-	Accelerating Smith-Waterman algorithm	3×	[67]
FPGA design by Chen et al. [68]	-	-	✓	-	Accelerating Smith-Waterman algorithm	26.4x	[68]
SWIFOLD	-	-	✓	-	Accelerating Smith-Waterman algorithm	58.77×	[69]
FPGASW	-	-	✓	-	Accelerating Smith-Waterman application with backtracking	25.2×	[70]
FPGA design by Di [71]	-	-	✓	-	Accelerating Smith-Waterman algorithm	1.72×	[71]
SeedEx	-	-	✓	-	Read-alignment accelerator focusing on the seed-extension step	6.0×	[72]
GenPiP	-	-	-	✓	In-memory acceleration of basecalling and read mapping	41.6×	[15]
GenStore	-	-	-	✓	In-storage computing system for accelerating read mapping	33.63×	[73]
GRIM-Filter	-	-	-	✓	Accelerating seed location filtering in DNA read mapping	3.65×	[74]
GenASM	-	-	-	✓	Accelerating approximate string matching (ASM)	116×	[75]
ApHMM	-	-	-	✓	Accelerating profile hidden markov models for sequence alignment	260.03×	[76]

It accelerates the seeding and filtering stages of LASTZ by efficiently parallelizing the data-dependent instructions for the SIMT (single instruction, multiple threads) on GPUs. SegAlign also scales efficiently over multiple GPU nodes and can achieve good accuracy. Liu et al. developed CUSHAW [53], a high-performance GPU-optimized short-read aligner using the Burrows-Wheeler transform (BWT) and the Ferragina–Manzini index to reduce the search space. Similar to CUSHAW [53], SOAP3-dp [77] also uses the Burrows-Wheeler transform. SOAP3-dp is an extension of SOPA3 [26] that can improve the speed and sensitivity of short-read alignment by allowing multiple mismatches and gaps. Torres et al. [54] demonstrated that taking advantage of the GPU parallelism can significantly accelerate the exact alignment of short-read genetic sequences using the Burrows-Wheeler Transform. Klus et al. developed BarraCUDA [52] that can also leverage a GPU's massive parallelism to accelerate short-read alignment through the Burrows-Wheeler transform. In comparison, LOGAN is a GPU framework focusing on long-read alignment [30]. It optimized the popular X-drop alignment algorithm for GPUs to achieve high performance by efficiently handling the complexities and variations inherent to long sequencing reads. Genesis [78] demonstrated how to perform genomic data analysis operations using a series of SQL-style queries and construct hardware pipelines with Genesis hardware library modules. Genesis utilizes parallelism and data reuse by employing a dataflow architecture and on-chip scratchpads. It uses non-blocking APIs to manage the accelerators for concurrent execution of the accelerator and the host.

To achieve an exact and complete alignment of short-read sequences to microbial genomes using GPU, Blom et al. [55] proposed SARUMAN [79]. This framework can significantly speed up the short-read alignment using GPUs. BLAST (Basic Local Alignment Search Tool) [80] is a widely used algorithm for genome analysis. It is a program that finds regions of local similarity between sequences. Liu et al. developed CUDA-BLASTP [58] that demonstrates GPUs can significantly improve the performance of protein sequence alignment tools like BLASTP (Basic Local Alignment Search Tool for Proteins). Zhao et al. developed G-BLASTN [59] and demonstrated the potential of using GPUs to accelerate nucleotide alignment tools compared to the sequential NCBI-BLAST.

3. ACCELERATING GENOME ANALYSIS USING FPGAS

Recently, FPGAs (Field-Programmable Gate Arrays) have taken an important role in accelerating genomic analysis. FPGAs provide a versatile, high-performance computing solution that can be customized for specific genomic computations. This allows researchers to process large datasets much more efficiently. In this section, we focus on FPGAs as accelerators.

3.1 Accelerating Indexing

Tang et al. [62] tackled the growth of NGS data in a different way by introducing a CPU-FPGA heterogeneous architecture to accelerate a short-read mapping algorithm. The FPGA handles computationally intensive tasks such as exact match search and alignment, while the CPU manages tasks that involve more complex logic and decision-making. The system uses a custom-designed pipeline to maximize data throughput and minimize latency. Chen et al. [63] proposed an FPGA-

TABLE 3
BILLIONS OF CELL UPDATES PER SECOND (GCUPS), ENERGY, AREA, DATASET LENGTH, ACCURACY AND EXPERIMENT SYSTEM FOR DIFFERENT GENOME ANALYSIS TOOLS

Name	GCUPS	Energy	Area	Dataset length	Accuracy	System	Reference
H/S Co-design by Lo et al. [49]	-	-	-	57,962,777 #reads	98.9%	Xeon E5-2680 v4 CPU Xilinx Virtex UltraScale+ VCU1525	[49]
Helix	-	11×	43.83 mm^2	6,154 bases	-	NVIDIA Tesla T4 GPU Intel Xeon E5-4655 v4 CPU	[50]
PLEDGER	-	5.9×	-	150 bases	99%	Intel i7-8750H CPU & Nvidia GTX 1050Ti Odroid N2 + Cortex A53 & Mali-G52 GPU	[51]
FPGA design by Tang et al. [62]	-	-	-	100 bases	-	Xilinx Virtex5 LX330 FPGA	[62]
FPGA design by Chen et al. [63]	-	-	6.71×	4,096 bases	-	Intel Xeon W3505 CPU Xilinx XUPV5-LX110T	[63]
FHAST	-	-	-	62,851,893 #reads	99.9%	Intel Xeon 2.13 GHz Xilinx Virtex 5 FPGA	[64]
FPGA design by Zhang et al. [65]	25.6	-	-	65,536 bases	-	XD1000 platform	[65]
FPGA design by Yamaguchi et al. [67]	16	32×	-	-	-	ADM-XRC-5T2 with XC5VLX330T	[67]
FPGA design by Chen et al. [68]	-	6×	-	-	-	Xilinx VC707 FPGA	[68]
SWIFOLD	132.43	-	-	65M bases	-	Intel Xeon E5-2670 & E5-2695 v3 Intel Arria 10 GX & Xeon Phi 3120P, NVIDIA GTX 980	[69]
FPGASW	105.9	3.84×	-	-	-	Xilinx Virtex7 XC7VX485T FPGA	[70]
FPGA design by Di et al. [71]	42.47	8.49×	-	256 bases	-	Xilinx Virtex-7 FPGA Xilinx Kintex Ultrascale FPGA	[71]
SeedEx	-	11×	28.76 mm^2	787,265,109 #reads	98.19%	Intel Xeon E5-2686 v4 Xilinx Ultrascale+ VU9P FPGA	[72]
GenPiP	-	32.8×	163.8 mm^2	2,577,692,011 #reads	-	ReRAM	[15]
GenStore	-	27.17×	0.20 mm^2	-	-	In-storage processing system	[73]
GRIM-Filter	-	-	-	4,389,429 #reads	-	Intel(R) Core i7-2600 CPU	[74]
GenASM	-	37×	10.69 mm^2	10M #reads	-	Intel Xeon Gold 6126 CPU Nvidia Titan V GPU	[75]
ApHMM	-	115.46×	6.536 mm^2	163,482 #reads	-	AMD EPYC 7742 processor NVIDIA A100 & Titan V GPUs	[76]

based system specifically optimized for long DNA reads. An FPGA architecture was developed focusing on maximizing data throughput and parallel processing capabilities, which are optimized for the specific requirements of long-read mapping algorithms, such as SW and BLAST. FHAST [64] is an FPGA-based system designed to accelerate Bowtie, a widely used software tool for aligning short DNA sequences to large genomes. FHAST significantly improves the alignment speed and efficiency by offloading critical parts of the Bowtie algorithm to an FPGA.

3.2 Accelerating Sequence Alignment

For sequence alignment on FPGAs, Zhang et al. [65] proposed an innovative reconfigurable supercomputing platform, XD1000. They leveraged the flexibility and computational power of FPGAs by using a multistage PE design, a pipelined control mechanism, and a compressed substitution matrix storage structure. It implemented the SW algorithm for DNA and protein sequences. Benkrid et al. [66] introduced a flexible and highly parallel FPGA-based framework for pairwise sequence alignment. This framework takes advantage of an FPGA's flexibility and parallel processing capability and supports both SW and Needleman-Wunsch (NW) algorithms.

Yamaguchi et al. [67] first introduced two methods for organizing parallelism in the SW algorithm and compared their performance to peak performance at varying levels of parallelism. Then, they proposed a design to leverage the inherent parallelism of FPGAs to enhance the performance and efficiency of sequence alignment tasks. This design combined a pipelined architecture and a hierarchical memory system, resulting in significant speedups and optimized resource utilization. The FPGA design demonstrated significantly higher energy efficiency compared to GPU. Chen et al. [68] addressed the challenge of handling different input sizes and extensive pruning techniques used in modern aligners. They proposed an architecture designed to effectively manage varied input sizes and utilize software pruning strategies to speed up the SW algorithm. SWIFOLD [69] also addressed the computational demands of the Smith-Waterman algorithm by implementing it on an FPGA using OpenCL. SWIFOLD demonstrates excellent performance for small and medium datasets compared with other state-of-the-art methods.

FPGASW [70] is an FPGA-based implementation of the Smith-Waterman sequence alignment algorithm designed to address the backtracking stage bottlenecks. It utilizes a linear systolic array and optimized hardware modules for backtracking. Di et al. [71] proposed an implementation of the Smith-Waterman algorithm without heuristics to ensure result accuracy. They included a series of architectural optimizations to maximize performance for FPGAs using OpenCL and uses the Berkeley roofline model for performance tracking and optimization guidance. The design includes systolic arrays, data compression features, shift registers, and a custom port mapping strategy for scalability.

Lo et al. [49] focused on FPGA-based acceleration on the base quality score re-calibration (BQSR) step in GATK. Their work concentrated on adjusting the BQSR algorithm to deal with random memory access conflicts with FPGA-specific features such as ultraRAM. SeedEx [72] is a powerful and memory-efficient solution for genome sequencing alignment tasks. It is a read-alignment accelerator that focuses on the seed-extension step in genome sequencing. SeedEx uses a narrow-band seed-extension approach to efficiently handle most reads, requiring only a small edit distance for alignment.

4. Accelerating genome analysis using ASICs

Application-specific integrated circuits (ASICs) provide customized hardware solutions optimized for handling the vast amount of data and complex algorithms involved in genomics research. ASICs are designed and optimized for specific tasks that offer unparalleled performance for designated genomics applications. In this section, we focus on ASICs in genomics research. We explore the potential of ASICs for accelerating indexing, seeding, and sequence alignment.

4.1 Accelerating Indexing

Lou et al. [50] used a specialized ASIC architecture named HELIX to tackle the specific challenges of nanopore base-calling, which not only provides significant speed and accuracy improvement over existing base-calling solutions but also marks a first-of-its-kind innovation in the use of an ASIC for real-time genomic analysis. Maheshwari et al. [51] implemented PLEDGER, a software-hardware co-design with memory-aware implementation, to improve the read-mapping process. PLEDGER optimizes algorithmic flows and hardware utilization to overcome the memory bottleneck commonly seen in read mapping. They demonstrated that hardware-software co-design can achieve high-performance genomic data processing, even within the limitations of embedded systems. GenPiP [15] is an in-memory genome analysis accelerator that combines basecalling and read mapping processes to address the data movement and redundant computations bottleneck in existing genome analysis pipelines. GenPiP proposes a chunk-based pipeline for parallel processing and early rejection of low-quality or unmapped reads to enhance throughput with minimal accuracy loss.

4.2 Accelerating Seeding

GRIM-Filter [74] is an ASIC system that aims to reduce data movement and computational overheads in genome sequence analysis with in-storage processing. It proposes accelerating read mapping using processing-in-memory (PIM) for fast seed location filtering by using a seed location filtering algorithm optimized for 3D-stacked memory systems. GRIM-Filter introduced a new representation of coarse-grained segments of the reference genome and employs parallel in-memory operations to identify reads within each segment. GenStore [73] is an in-storage processing system designed to reduce data movement and computational overheads in genome sequence analysis. It addresses the bottlenecks of traditional methods by processing data where it is stored. GenStore uses a hardware/software co-design to take advantage of in-storage filters (Exactly-matching and Non-matching). It reduces data movement and improves processing speeds by integrating computational capabilities directly into storage devices.

4.3 Accelerating Sequence Alignment

GenASM [75] is designed to address the bottleneck in ASM. It is the first framework to accelerate ASM in genome sequence analysis. GenASM focuses on bitvectorbased ASM and has modified the Bitap algorithm to increase parallelism and reduce memory usage. It includes a hardware accelerator with specialized systolic-array-based compute units and on-chip SRAMs. ApHMM [76] addresses the computational and energy inefficiencies in the Baum-Welch algorithm for profile hidden Markov models (pHMMs) by using a flexible acceleration framework. It utilizes a hardware-software co-design, featuring flexible hardware, exploiting data dependency patterns, and using on-chip memory with memorization. It implements a hardware-based filter for efficient computation, minimizing redundant calculations.

5. Discussion and Perspectives

The application of GPUs in genomics research has revolutionized the field, offering significant improvements in speed and efficiency. However, despite their advantages, integrating GPUs into genomics research workflows presents several challenges. The architectural differences between CPUs and GPUs require modifying existing genome algorithms to fully leverage the GPU. With NGS generating massive amounts of data, data movement is still a bottleneck for GPUs due to limited network bandwidth [81], [82]. Because GPUs are general-purpose hardware, developing tailored GPU architectures to support certain algorithms is costly [83], [84]. Instead, most GPU-related genome frameworks focus on algorithm optimization, trying to adapt the software to match the underlying hardware. In addition, many GPU implementations did not utilize multi-threading or vector instructions on the CPU side. It has been shown that optimizing CPU-executed instructions in addition to GPU acceleration can achieve significant speed-ups [85]. Advanced GPU implementations, such as those leveraging shared memory and asynchronous data transfers, highlight the complexity of effective CUDA [86] programming. We expect the ongoing developments in GPU technology to further increase the performance gap with CPUs, leading to more complex and large-scale simulations in life sciences [87].

FPGAs provide a combination of flexibility, improved processing speeds, parallel processing capability, and energy efficiency. However, there are also several challenges. First, FPGAs require more knowledge of hardware programming from users. Additionally, FPGAs also face the challenges of high development costs, limited flexibility (compared to software solutions), and difficulty in integration with existing data pipelines [63], [71]. Some specific expertise is needed to fully take advantage of the potential of specialized hardware like the FPGAs in the bioinformatics field. In the future, using FPGAs as versatile hardware accelerators can benefit from the development of more user-friendly FPGA programming frameworks and developing hybrid systems that combine FPGAs with other accelerators.

ASICs are specifically designed hardware to execute certain algorithms and can usually achieve better performance among the three types of accelerators [73], [75]. However, their specific design limits their applicability to a wide range of genomic algorithms. For example, GenStore [73] significantly reduces data movement and computational overhead. However, it's limited to a number of genome algorithms and filter techniques. GenAsm's [75] low-power ASM framework offers significant improvements in performance and energy, though it may encounter challenges in handling highly divergent sequences. GenPiP [15] presents an in-memory acceleration technique that integrates basecalling and read mapping. However, it struggles with the complexity of integration and hardware compatibility. The cost of designing and manufacturing ASICs could be higher than that of GPUs and FPGAs due to their limited applications. They are also costly and time-consuming. Additionally, they lack the flexibility of reconfigurable hardware like FPGAs. Once designed, an ASIC cannot be easily modified to accommodate new algorithms or changes in existing ones. A balanced tradeoff between performance and costs needs to be well explored when considering ASICs as a genomics accelerator.

An effective way to speed up sequence alignment for NGS is to integrate a large number of processing elements onto a single chip, fully utilizing the fine-grain parallelism present in many genome workloads. Network-on-chip (NoC) is a highly efficient method for achieving this kind of large-scale integration. Much work has already been done on NoC architecture, such as NoC for GPUs [88] and heterogeneous chiplets [89]. However, specific NoC architecture for genome workloads has not been fully explored.

As the data size keeps growing in genomics analysis, memory bandwidth becomes a most severe limitation to performance. Memory bandwidth includes off-chip bandwidth and on-chip bandwidth. There have been many works developing process-in-memory architectures to decrease genomic data to be fetched from off-chip memory. However, there is little work trying to remove the on-chip bandwidth bottleneck. Specialized NoC design for genome workloads is an under-explored area that may further improve performance. Another promising opportunity for future research is through hardware-software co-design. Such an approach can achieve optimal speed-ups and efficiency, and yet it still faces several challenges, such as the design complexity of custom hardware, the need for specialized bioinformatics knowledge, the ever-growing genomics technologies, and the fast-evolving genomic algorithms. From our perspective, the ultimate solution lies in the combination of expertise in both hardware and software design to coordinatively optimize the overall system.

ACKNOWLEDGEMENTS

This work was supported in part by NSF grants 2008911, 2046186, and 2051062.

REFERENCES

[1] Jacob J Chabon et al. Integrating genomic features for non-invasive early lung cancer detection. *Nature*, 580(7802):245–251, 2020.

[2] Yan Nie et al. The non-homologous end-joining activity is required for fanconi anemia fetal hsc maintenance. *Stem Cell Res Ther*, 10:1–10, 2019.

[3] Rama R Gullapalli et al. Next generation sequencing in clinical medicine: Challenges and lessons for pathology and biomedical informatics. *Journal of pathology informatics*, 3(1):40, 2012.

[4] Shichao Feng et al. Metalp: An integrative linear programming method for protein inference in metaproteomics. *PLOS Computational Biology*, 18(10):e1010603, 2022.

[5] David R Bentley et al. Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, 456(7218):53–59, 2008.

[6] John Eid et al. Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[7] Daniel Branton et al. The potential and challenges of nanopore sequencing. *Nature biotechnology*, 26(10):1146–1153, 2008.

[8] Yongqi Zhu, Nan Liu, and Qing Yang. A new approximation algorithm for genomic scaffold filling based on contig. In *2023 IEEE Healthcom*, pages 72–77. IEEE, 2023.

[9] Jay Shendure et al. Dna sequencing at 40: past, present and future. *Nature*, 550(7676):345–353, 2017.

[10] Fatih Ozsolak and Patrice M Milos. Rna sequencing: advances, challenges and opportunities. *Nature reviews genetics*, 12(2):87–98, 2011.

[11] Mohammed Alser et al. Accelerating genome analysis: A primer on an ongoing journey. *IEEE/ACM Micro*, 2020.

[12] Tony Robinson, Jim Harkin, and Priyank Shukla. Hardware acceleration of genomics data analysis: challenges and opportunities. *Bioinformatics*, 37(13):1785–1795, 2021.

[13] Onur Mutlu and Can Firtina. Accelerating genome analysis via algorithm-architecture co-design. In *ACM/IEEE DAC*, 2023.

[14] Chris Seymour, Oxford Nanopore Technologies Ltd. Bonito: A pytorch basecaller for oxford nanopore reads, 2019. Oxford Nanopore Technologies, Ltd. Public License, v. 1.0.

[15] Haiyu Mao et al. Genpip: In-memory acceleration of genome analysis via tight integration of basecalling and read mapping. In *IEEE/ACM MICRO*, 2022.

[16] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[17] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *ACM MOD*, 2003.

[18] Michael Roberts et al. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.

[19] Nvidia hopper architecture in-depth — nvidia technical blog. https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/.

[20] Intel challenges nvidia with gaudi 3 ai accelerator. https://www.forbes.com/sites/stevemcdowell/2024/04/09/intels-challenges-nvidia-with-gaudi-3-ai-accelerator/.

[21] Intel® stratix® 10 fpgas overview - high performance intel® fpga. https://www.intel.com/content/www/us/en/products/details/fpga/stratix/10/item.html. (Accessed on 07/29/2024).

[22] Xianwei Cheng et al. Amoeba: A coarse grained reconfigurable architecture for dynamic gpu scaling. In *ICS*, 2020.

[23] Zhuren Liu et al. Genomics-gpu: A benchmark suite for gpu-accelerated genome analysis. In *IEEE ISPASS*, 2023.

[24] Accelerate genomic analysis for any sequencer with nvidia parabricks v4.2 — nvidia technical blog. https://developer.nvidia.com/blog/accelerate-genomic-analysis-for-any-sequencer-with-parabricks-v4-2/.

[25] Qian Zhang, Hao Liu, and Fengxiao Bu. High performance of a gpu-accelerated variant calling tool in genome data analysis. *bioRxiv*, pages 2021–12, 2021.

[26] Chi-Ming Liu et al. Soap3: Gpu-based compressed indexing and ultra-fast parallel alignment of short reads. In *ALGO*. Citeseer, 2011.

[27] Anas Abu-Doleh et al. Masher: mapping long (er) reads with hash-based genome indexing on gpus. In *ACM BCB*, 2013.

[28] Franklin LA Cruz-Gamero and Juan Carlos Gutiérrez Cáceres. Optimization of blast seed indexing in the alignment of dna sequences with gpu using cuda. In *IEEE CLEI*, 2018.

[29] Gustavo Encarnação, Nuno Sebastião, and Nuno Roma. Advantages and gpu implementation of high-performance indexed dna search based on suffix arrays. In *IEEE HPCS*, 2011.

[30] Alberto Zeni et al. Logan: High-performance gpu-based x-drop long-read alignment. In *IEEE IPDPS*, 2020.

[31] Shanshan Ren, Koen Bertels, and Zaid Al-Ars. Gpu-accelerated gatk haplotypecaller with load-balanced multi-process optimization. In *IEEE BIBE*, 2017.

[32] Amaro Taylor-Weiner et al. Scaling computational genomics to millions of individuals with gpus. *Genome biology*, 20:1–5, 2019.

[33] Nauman Ahmed et al. Gasal2: a gpu accelerated sequence alignment library for high-throughput ngs data. *BMC bioinformatics*, 20:1–20, 2019.

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

[35] Friman Sánchez et al. Performance analysis of sequence alignment applications. In *IEEE IISWC*, 2006.
[36] Edans Flavius O Sandes and Alba Cristina MA de Melo. Cudalign: using gpu to accelerate the comparison of megabase genomic sequences. In *ACM PPoPP*, 2010.
[37] Edans Flavius de O Sandes and Alba Cristina MA de Melo. Retrieving smith-waterman alignments with optimizations for megabase biological sequences using gpu. *IEEE TPDS*, 24(5):1009–1021, 2012.
[38] Azzedine Boukerche, Rodolfo Bezerra Batista, and Alba Cristina Magalhaes Alves De Melo. Exact pairwise alignment of megabase genome biological sequences using a novel z-align parallel strategy. In *2009 IEEE IPDPS*, pages 1–8, 2009.
[39] Yongchao Liu, Adrianto Wirawan, and Bertil Schmidt. Cudasw++ 3.0: accelerating smith-waterman protein database search by coupling cpu and gpu simd instructions. *BMC bioinformatics*, 14:1–10, 2013.
[40] Edans Flavius de Oliveira Sandes et al. Cudalign 4.0: Incremental speculative traceback for exact chromosome-wide alignment in gpu clusters. *IEEE TPDS*, 27(10):2838–2850, 2016.
[41] Jie Wang, Xinfeng Xie, and Jason Cong. Communication optimization on gpu: A case study of sequence alignment algorithms. In *IEEE IPDPS*, 2017.
[42] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.
[43] Licheng Guo et al. Hardware acceleration of long read pairwise overlapping in genome sequencing: A race between fpga and gpu. In *IEEE FCCM*, 2019.
[44] Heng Li. New strategies to improve minimap2 alignment accuracy. *Bioinformatics*, 37(23):4572–4574, 2021.
[45] Zonghao Feng et al. Accelerating long read alignment on three processors. In *ACM ICPP*, 2019.
[46] Gregory M Striemer and Ali Akoglu. Sequence alignment with gpu: Performance and design challenges. In *IEEE ISPA*, 2009.
[47] Amir Muhammadzadeh. *MR-CUDASW-GPU accelerated Smith-Waterman algorithm for medium-length (meta) genomic data*. PhD thesis, University of Saskatchewan, 2014.
[48] Matija Korpar and Mile Šikić. Sw#–gpu-enabled exact alignments on genome scale. *Bioinformatics*, 29(19):2494–2495, 2013.
[49] Michael Lo et al. Algorithm-hardware co-design for bqsr acceleration in genome analysis toolkit. In *IEEE FCCM*, 2020.
[50] Qian Lou, Sarath Chandra Janga, and Lei Jiang. Helix: Algorithm/architecture co-design for accelerating nanopore genome basecalling. In *ACM PACT*, 2020.
[51] Sidharth Maheshwari et al. Pledger: Embedded whole genome read mapping using algorithm-hw co-design and memory-aware implementation. In *IEEE DATE*, 2021.
[52] Petr Klus et al. Barracuda-a fast short read sequence aligner using graphics processing units. 5:1–7, 2012.
[53] Yongchao Liu, Bertil Schmidt, and Douglas L Maskell. Cushaw: a cuda compatible short read aligner to large genomes based on the burrows–wheeler transform. *Bioinformatics*, 28(14):1830–1837, 2012.
[54] José Salavert Torres et al. Using gpus for the exact alignment of short-read genetic sequences by means of the burrows-wheeler transform. *IEEE/ACM TCBB*, 9(4):1245–1256, 2012.
[55] Jochen Blom et al. Exact and complete short-read alignment to microbial genomes using graphics processing unit programming. *Bioinformatics*, 27(10):1351–1358, 2011.
[56] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.
[57] Nuno Subtil Jacopo Pantaleoni. Nvbio: Nvbio. https://nvlabs.github.io/nvbio/.
[58] Weiguo Liu, Bertil Schmidt, and Wolfgang Muller-Wittig. Cuda-blastp: accelerating blastp on cuda-enabled graphics hardware. *IEEE/ACM TCBB*, 8(6):1678–1684, 2011.
[59] Kaiyong Zhao and Xiaowen Chu. G-blastn: accelerating nucleotide alignment by graphics processors. *Bioinformatics*, 30(10):1384–1391, 2014.
[60] Cole Trapnell and Michael C Schatz. Optimizing data intensive gpgpu computations for dna sequence alignment. *Parallel computing*, 35(8-9):429–440, 2009.
[61] Sneha D Goenka et al. Segalign: A scalable gpu-based whole genome aligner. In *IEEE SC*, 2020.
[62] Wen Tang et al. Accelerating millions of short reads mapping on a heterogeneous architecture with fpga accelerator. In *IEEE FCCM*, 2012.
[63] Peng Chen et al. Accelerating the next generation long read mapping with the fpga-based system. *IEEE/ACM TCBB*, 11(5):840–852, 2014.
[64] Edward B Fernandez et al. Fhast: Fpga-based acceleration of bowtie in hardware. *IEEE/ACM TCBB*, 12(5):973–981, 2015.
[65] Peiheng Zhang, Guangming Tan, and Guang R Gao. Implementation of the smith-waterman algorithm on a reconfigurable supercomputing platform. In *HPRCTA conjunction with SC07*, pages 39–48, 2007.
[66] Khaled Benkrid, Ying Liu, and AbdSamad Benkrid. A highly parameterized and efficient fpga-based skeleton for pairwise biological sequence alignment. *IEEE VLSI*, 17(4):561–570, 2009.
[67] Yoshiki Yamaguchi, Hung Kuen Tsoi, and Wayne Luk. Fpga-based smith-waterman algorithm: Analysis and novel design. In *ARC*. Springer, 2011.
[68] Yu-Ting Chen et al. A novel high-throughput acceleration engine for read alignment. In *IEEE FCCM*, 2015.
[69] Enzo Rucci et al. Swifold: Smith-waterman implementation on fpga with opencl for long dna sequences. *BMC systems biology*, 12:43–53, 2018.
[70] Xia Fei et al. Fpgasw: accelerating large-scale smith–waterman sequence alignment application with backtracking on fpga linear systolic array. *Interdisciplinary Sciences: Computational Life Sciences*, 10:176–188, 2018.
[71] Lorenzo Di Tucci et al. Architectural optimizations for high performance and energy efficient smith-waterman implementation on fpgas using opencl. In *IEEE DATE*, 2017.
[72] Daichi Fujiki et al. Seedex: A genome sequencing accelerator for optimal alignments in subminimal space. In *IEEE/ACM MICRO*, 2020.
[73] Nika Mansouri Ghiasi et al. Genstore: a high-performance in-storage processing system for genome sequence analysis. In *ACM ASPLOS*, 2022.
[74] Jeremie S Kim et al. Grim-filter: Fast seed location filtering in dna read mapping using processing-in-memory technologies. *BMC genomics*, 19:23–40, 2018.
[75] Damla Senol Cali et al. Genasm: A high-performance, low-power approximate string matching acceleration framework for genome sequence analysis. In *IEEE/ACM MICRO*, 2020.
[76] Can Firtina et al. Aphmm: Accelerating profile hidden markov models for fast and energy-efficient genome analysis. 21(1):1–29, 2024.
[77] Ruibang Luo et al. Soap3-dp: fast, accurate and sensitive gpu-based short read aligner. *PloS one*, 8(5):e65632, 2013.
[78] Tae Jun Ham et al. Genesis: A hardware acceleration framework for genomic data analysis. In *2020 ACM/IEEE ISCA*, pages 254–267. IEEE, 2020.
[79] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.
[80] Johanna McEntyre and Jim Ostell. The ncbi handbook. *Bethesda (MD): National Center for Biotechnology Information (US)*, 2002.
[81] Xianwei Cheng et al. Packet pump: overcoming network bottleneck in on-chip interconnects for gpgpus. In *DAC*, 2018.
[82] Hui Zhao et al. Designing scalable hybrid wireless noc for gpgpus. In *ISVLSI*, 2018.
[83] Khoa Ho et al. Improving gpu throughput through parallel execution using tensor cores and cuda cores. In *ISVLSI*, 2022.
[84] Zhuren Liu et al. Predicting gpu performance and system parameter configuration using machine learning. In *2022 IEEE ISVLSI*, pages 253–258. IEEE, 2022.
[85] Oliver Korb, Thomas Stutzle, and Thomas E Exner. Accelerating molecular docking calculations using graphics processing units. *Journal of chemical information and modeling*, 51(4):865–876, 2011.
[86] NVIDIA Corporation. *CUDA C Programming Guide*, 2023. Accessed: 2024-06-16.
[87] Marco S Nobile et al. Graphics processing units in bioinformatics, computational biology and systems biology. *Briefings in bioinformatics*, 18(5):870–885, 2017.
[88] Siamak Biglari Ardabili and Gholamreza Zare Fatin. Icla unit: Intra-cluster locality-aware unit to reduce l 2 access and noc pressure in gpgpus. *Journal of Circuits, Systems and Computers*, 31(01):2250015, 2022.
[89] Siamak Biglari et al. Designing reconfigurable interconnection network of heterogeneous chiplets using kalman filter. In *GLSVLSI*, 2024.