SpectraFlux: Harnessing the Flow of Multi-FPGA in Mass Spectrometry Clustering

Tianqi Zhang
UCSD
La Jolla, CA, US
tiz014@ucsd.edu

Neha Prakriya
UCLA
Los Angeles, CA, US
nehaprakriya@ucla.edu

Sumukh Pinge
UCSD
La Jolla, CA, US
spinge@ucsd.edu

Jason Cong
UCLA
Los Angeles, CA, US
cong@cs.ucla.edu

Tajana Rosing
UCSD
La Jolla, CA, US
tajana@ucsd.edu

ABSTRACT

The identification and quantification of proteins through mass spectrometry (MS) are foundational to proteomics, offering insights into biological systems and disease states. However, current clustering tools struggle to process large-scale datasets. We propose SpectraFlux, a multiple FPGA-based architecture for accelerated mass spectrum clustering that outperforms existing CPU, GPU, and FPGA designs. It employs heterogeneous clustering kernels for adaptive bucket size management and optimizes memory usage by distinguishing between on-chip and high-bandwidth memory (HBM) storage solutions. SpectraFlux is built upon the TAPA-CS framework, which automatically compiles and partitions a large dataflow design across multiple chips with RDMA-based inter-FPGA communication. Our solution shows a 2.7× speed up on a quad-FPGA platform compared to a single FPGA. Additionally, we introduce a refined cost model for frame-based inter-FPGA communication to better accommodate the variable data rates inherent in proteomic data processing. This reduces the inter-FPGA data movement by up to 73%. Finally, SpectraFlux achieves speedups of up to 11× and 17× over SOTA FPGA and GPU accelerators, respectively.

CCS CONCEPTS

• **Hardware** → **Hardware accelerators**; • **Applied computing** → **Metabolomics / metabonomics**; **Bioinformatics**.

KEYWORDS

Multi-FPGA design, Mass Spectrum Clustering, HD Computing, Hierarchical Clustering

ACM Reference Format:

Tianqi Zhang, Neha Prakriya, Sumukh Pinge, Jason Cong, and Tajana Rosing. 2024. SpectraFlux: Harnessing the Flow of Multi-FPGA in Mass Spectrometry Clustering. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3657354

1 INTRODUCTION

In the realm of biological research, alongside genomics, proteomics stands as a crucial field focused on proteins. Central to proteomics is mass spectrometry (MS), essential for analyzing proteins. MS measures the mass-to-charge ratio of ionized protein fragments, producing a large volume of spectra data. This data is key for understanding biological processes and investigating disease mechanisms. MS analysis produces a massive data footprint, stressing the computing and memory demands. Databases like MassIVE [12] are now filled with billions of individual spectra, adding up to hundreds of terabytes in size.

Large-scale proteomics projects frequently generate large quantities of tandem mass spectra [11], many of which are duplicative, resulting in increased data storage burdens and computational loads [13]. To address this, mass spectrum clustering is employed as a data refinement technique, consolidating similar spectra to streamline datasets. This consolidation not only reduces data redundancy but also enhances the clarity of the signal, leading to more precise protein identification and quantification. Therefore, the large-scale MS data places considerable demand on parallelable and scalable mass spectrum clustering acceleration.

Several CPU-based tools [1, 2, 10, 13] have been developed to enhance mass spectrum clustering performance within latency and cost constraints. A primary challenge is the quadratic complexity associated with calculating dense pairwise distances within the data [16]. Spectrum binning divides the large datasets into buckets when preprocessing the spectrum, allowing for a maintainable distance matrix while preserving clustering quality [1]. The adaptive bucket sizes, which are tuned for CPUs [2, 10] and GPUs [16], exhibit a long-tailed distribution, as evidenced in GPU-based strategies. The SOTA FPGA accelerators [6] face limitations due to the restricted capacity of on-chip BRAM, impacting bucket sizes and subsequently the clustering quality for large datasets. Another crucial process is spectra embedding, which involves compressing and extracting features from the mass spectrum vector into a densely informative format. Unlike the lower-dimensional embeddings used in MsCrush [13] and Falcon [1], Hyperdimensional Computing (HDC) proves to be more efficient in data representation due to its utilization of binary hypervectors [16].

SpectraFlux, built using the TAPA-CS framework [7] (an enhancement of TAPA [4]), expands FPGA design partitioning to clusters of FPGAs, harnessing task-level parallelism. It introduces differentiated cost functions for inter-die and inter-board communications. The cost model, previously based on data width, has been adapted in this work for frame-based communication by accounting for the variability in data frames and transmission rates. SpectraFlux stands out as the first mass spectrum clustering accelerator scalable to multiple FPGAs. Its key advancements include:

- Employs Hyperdimensional Computing (HDC) based spectra embedding and hierarchical clustering. Using heterogeneous kernels, it efficiently manages on-chip and HBM storage

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3657354

and narrows the performance gap in clustering quality with existing FPGA [6] and GPU [11, 16] solutions, particularly for long-tailed MS datasets.
- Develops a refined communication cost model for frame-based inter-FPGA data transfers, allowing for more accurate flow prediction and system optimization. This enhanced cost model reduces inter-FPGA communication by up to 73%.
- Excellent scalability, with the 2 FPGA and 4 FPGA setups achieving 1.7× and 2.6× faster performance, respectively, than a single FPGA. SpectraFlux delivers up to 11×, 17×, and 16× speed enhancements over the latest FPGA, GPU, and CPU systems.

2 BACKGROUND AND RELATED WORK

2.1 Mass Spectrometry and Clustering

Proteomic MS analysis is comprised of a data and compute pipeline, shown in Fig. 1, which includes the precise measurement of ionized peptides by mass spectrometers and encompasses preprocessing steps like noise filtering and bucket binning, followed by clustering to group similar spectra for data reduction and downstream analysis like database search. MS clustering has been proven to double the throughput in downstream database searches [13].

The pipeline starts with a mass spectrometer analyzing ionized proteins' mass-to-charge ratios (m/z). In the preprocessing step, low-intensity spectra are filtered and binned into buckets. Near-storage [15] and GPU accelerations [16] have been proposed to optimize bandwidth in this stage. Next, in the clustering stage, pairwise distances are calculated within buckets to avoid the curse of dimensionality in high-dimensional spectra vectors. Various prior tools apply different embedding methods and distance metrics, including similarity scoring functions [10, 11] and neural network-based distance matrices [2]. Hyperdimensional computing (HDC) has also been employed for mass spectra embedding [16] due to its ability to represent data efficiently. Finally, clustering is achieved using methods like DBSCAN [16], Complete-linkage HAC [10, 13, 16], and iterative clustering [13].

Mass spectrum clustering has been accelerated by using GPUs [11, 16] and FPGAs [6]. GPU-based designs can be scaled up either by deploying them on expensive devices equipped with interconnection technologies like NVLink or by significantly increasing overhead via the host. However, GPU-based designs are less energy efficient than FPGA-based accelerators. Despite the increased energy efficiency achievable through FPGA accelerators, prior FPGA-based work [6] faces a loss of clustering quality due to its inability to handle long-tailed real datasets.

FPGA-based clustering algorithms, such as DBSCAN [8], HAC [6], and HDC [17], typically rely on a single FPGA, limiting their scalability. SpectraFlux's application of HDC, which differs from traditional HDC clustering [17], involves utilizing HDC for spectra data representation and HAC for clustering, unlike the iterative comparison of hypervectors to centroids in conventional HDC clustering. Hadoop-based multi-FPGA clustering has been proposed [3], but this approach limits FPGAs to function as slave nodes to different hosts, preventing direct inter-FPGA communication, resulting in large data movement between host and FPGA.

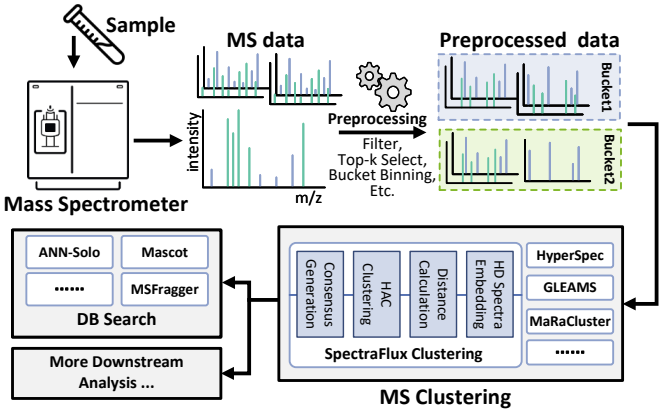

Figure 1: MS data analysis pipeline.

2.2 TAPA-CS

TAPA-CS [7] is a task-parallel dataflow programming framework designed to partition and compile large designs across multiple FPGAs with minimal user effort, achieving high frequency and throughput. It consists of four major steps: a) task graph construction and parallel synthesis, b) inter-FPGA floorplanning and communication logic insertion, c) intra-FPGA floorplanning and pipeline registers insertion, d) bitstream generation. The first step involves identifying the interconnections between compute modules and synthesizing them to estimate resource utilization. The second step involves mapping computing modules to FPGAs, considering various topologies and protocols, and integrating communication logic between FPGAs. The third step strategically organizes each FPGA's internal layout and pipelines the inter-die logic for high frequency. The final step generates bitstreams. While TAPA-CS is geared towards multi-FPGA design and simplifies interaction with inter-FPGA communication protocols like Ethernet RDMA-based AlveoLink [14] and PCIe-based P2P DMA, its method for estimating inter-FPGA communication costs during task mapping is limited to considering only the streams' data bitwidth. This approach can lead to biased results, particularly for frame-based data transfer. We discuss how we address this shortcoming in Section 3.3.

2.3 MS Clustering Acceleration Challenges

Based on the pipeline shown in Fig. 1, and the shortcomings of prior work, we summarize the challenges as follows:
(1) **Large scale input data:** The preprocessed data lacks information density, necessitating highly efficient embedding methods.
(2) **Computational demand in distance matrix calculation:** The pairwise distance matrix calculation, essential for clustering, faces quadratic computational complexity, making it challenging to store all necessary data on-chip.
(3) **Memory access latency in clustering:** The final clustering requires frequent distance matrix access, demanding an efficient pipeline design to minimize memory access latency.

We use these challenges to guide the design of SpectraFlux and describe our framework in the following Section.

3 SPECTRAFLUX DESIGN

Fig.2 illustrates the architecture of SpectraFlux that is built using the TAPA-CS framework [7]. SpectraFlux features multiple datacenter FPGAs interconnected through a ring bus Ethernet configuration. TAPA-CS facilitates efficient design partitioning with a customized cost function and supports inter-FPGA communication. SpectraFlux presents a range of specialized components: a universal HDC-based spectra embedding kernel, and dual-versioned heterogeneous modules for pairwise distance matrices, Hierarchical Agglomerative Clustering (HAC), and consensus sequence generation, each optimized for handling small and large bucket sizes. The inter-FPGA communication is managed by AlveoLink [14], which is seamlessly integrated into the system by the TAPA-CS framework.

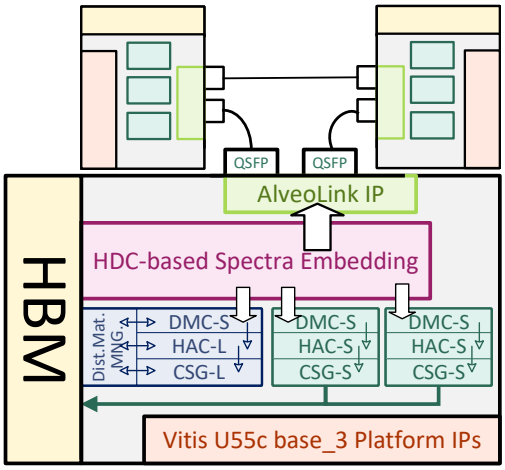

Figure 2: Overall Diagram of SpectraFlux Implemented on Three Ethernet-Connected FPGAs.

3.1 HDC-based spectra embedding

SpectraFlux employs an HDC-based method to embed these significant data pairs from a vector into a D_{hv}-dimensional hypervector (HV), which has been proven to be a highly efficient data format for spectra data concentration [16]. Initially, it quantizes the (m/z, intensity) pair into q_{mz} and q_i bits, respectively. In other words, SpectraFlux represents each datum with $2^{q_{mz}}$ and 2^{q_i} levels. BRAMs store $2^{q_{mz}}$ and 2^{q_i} D_{hv}-dimensional random binary HVs as the base HVs for m/z and intensity, named ID_j, $j \in [0, 2^{qmz} - 1]$ and L_j, $j \in [0, 2^{qi} - 1]$. For each pair (m/z, intensity), bitwise XOR operations are applied to the corresponding vectors from ID and L. The projections of the K most frequent fragments are accumulated, and the majority is selected to obtain the final spectra embedding, represented as

$$spectra = \text{Majority}(\sum^{K} \text{ID}[Q_m(mz)] \oplus \text{L}[Q_i(intensity)]), \quad (1)$$

where $Q_m(\cdot)$, $Q_i(\cdot)$, and Majority$(\cdot)$ are the uniform quantization functions for m/z and intensity, and the function to select the majority, respectively. The summation is done pointwise over the D_{hv} dimension.

Fig. 3 shows the proposed architecture for the embedding process. To optimize memory utilization by burst fetching data from HBM, SpectraFlux separates the data loading module from the embedding module. Since the bucket bin size for spectrum clustering varies, the embedding module incorporates the bucket bin size into the embedding spectra stream to reduce inter-module data width. Furthermore, the spectra embedding process is executed on-the-fly, transmitting packets to downstream processing elements without storing the embedded spectra in external memory.

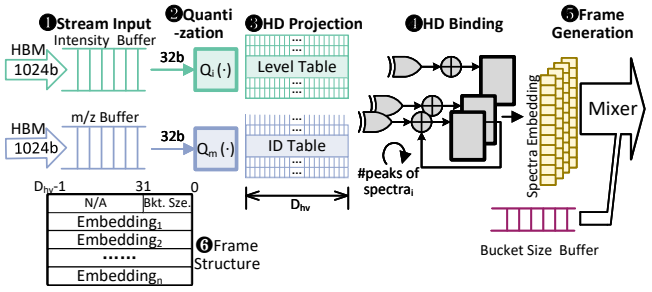

Figure 3: Dataflow of HDC-based Spectra Embedding

3.2 Hierarchical agglomerative clustering

Heterogenous pairwise distance matrix computation (DMC): The HAC algorithm described in the next Section requires a quadratic pairwise distance matrix. Fig.4 presents the cumulative distribution function (CDF) of bucket bin sizes for clustering, utilizing the division criterion of GPU tools[11, 16] for 100GB-scale datasets. This distribution exhibits a long-tailed pattern, where most (>85%) of the bucket sizes are smaller than 320, but there remains a long-tail of larger sizes.

Previous FPGA-based MS clustering like SpecHD [6], stores all data in BRAMs and limits the bucket size to 300. This restriction constrains its applicability to large datasets. In order to efficiently handle both small and long-tailed larger buckets, SpectraFlux introduces two types of distance matrix modules, illustrated in Fig. 5 (a). The first module is tailored for the smaller buckets, using on-chip memory for the distance matrix. The second module, in contrast, leverages HBM memory with an optimized dataflow to manage the long-tailed larger buckets. The cutoff size is 320, as larger on-chip matrices face routing challenges due to HBM congestion.

In the first module, the embedded spectra of the entire bucket are stored in dual-port URAM, enabling single-cycle access to both operands. Since the large dataset size can lead to explosive URAM requirements, SpectraFlux uses a $\log_2 D_{hv}$-bit custom arbitrary precision integer for storing distance values without risk of overflow. For the second module, SpectraFlux enhances external memory access using HLS burst inference. However, HLS typically infers burst access only from single-layer for-loops, while the computation of the pairwise distance matrix requires nested loops with variable inner bounds. To address this, SpectraFlux employs buffers for each operand, decoupling the nested loops and minimizing the frequency of external memory accesses.

Reciprocal Nearest Neighbor pair-based HAC clustering: HAC clustering is a bottom-up algorithm that builds a hierarchy of clusters by iteratively merging the nearest neighbor pairs until the distance between the two closest clusters exceeds a threshold. The concept of Reciprocal Nearest Neighbor (R-NN) pairs, where each

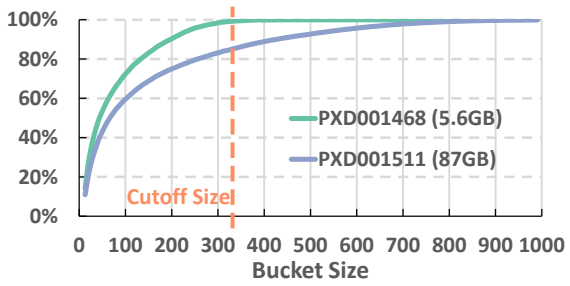

Figure 4: CDF of Bucket Sizes for Clustering

cluster in the pair is the nearest neighbor of the other, allows HAC to commence merging earlier rather than waiting to find the global nearest neighbor pair. However, the R-NN based method introduces uncertainty in the merging order, as R-NN pairs may not be unique like global nearest neighbor pairs. Consequently, it is necessary to identify a consensus sequence to determine a unique cluster. The consensus sequence, akin to the centroid in K-means and used as a characteristic of a specific cluster for downstream analysis [5], is generated in SpectraFlux employing MOST [9] consensus sequences, which selects the spectrum most similar on average to all members of a cluster.

In SpectraFlux, similar to the distance matrix computation module, there are two optimized versions for HAC clustering. HAC for small buckets directly reads and updates the distance matrix stored in URAM, while one for large buckets includes a cache to buffer parts of the distance matrix, thereby decoupling external memory read/write operations. The HAC module initially finds the smallest value in an arbitrary row, indicating the nearest point to the selected node, and places it in a stack. It then iteratively finds the nearest point to the stack's top and continues this process until an R-NN is located. During this process, for both kernels, the optimization of the dataflow is specifically designed to hide the memory access latency. However, this latency hiding is effectively achieved only during the operation of finding the minimum value in a single row, an operation that may occur across multiple rows. Following this, the algorithm performs node/cluster merging and updates the distance matrix. To track the clustering hierarchy, previous FPGA designs [6] used two separate sets: one for merging and another for mergings with inter-distances below the threshold, running until all nodes merge into a single cluster. In contrast, SpectraFlux tracks only the relevant merging and terminates early if the nearest pair is farther than the threshold by monitoring the count of distances below the threshold.

Unlike algorithms developed for CPUs/GPUs, the allocated BRAM size in FPGA designs must handle the worst-case scenario as it cannot be expanded during runtime. While the previous FPGA work [6] allocated $O(n^2)$ size of BRAM to record all leaf nodes of the subtree rooted at each node, SpectraFlux employs a linked list for tracing, reducing the memory requirement from $O(n^2)$ to $O(n)$ and only tracking the latest clustering hierarchy. The data structure comparison is shown in Fig. 5 (e).

Consensus sequences generation (CSG): The final stage of the process involves generating consensus sequences for the clusters, as per the MOST criterion [9]. This criterion identifies the spectrum that best represents all members of a cluster, determined by the

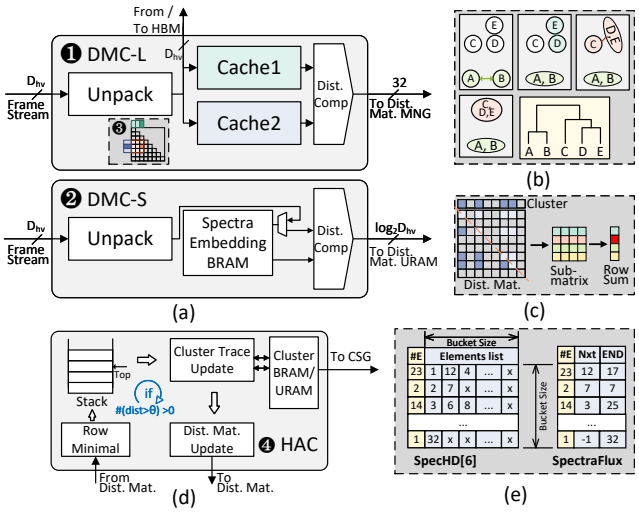

Figure 5: (a) Pairwise Distance Matrix Computing Architectures for Large and Small Buckets, with Cached DMC Example. (b) Hierarchical Agglomerative Clustering Workflow. (c) Consensus Sequence Generation Workflow. (d) Consensus Architecture of HAC. (e) Data Structures Comparison: Cluster Trace in SpecHD [6] and SpectraFlux.

average similarity. As shown in Fig.5 (c), this involves extracting a submatrix from the overall distance matrix, which includes only the distances among members of the same cluster. For each point 'i' in this submatrix, we calculate the cumulative distance from 'i' to all other points within the cluster. The point with the lowest total distance is then designated as the consensus point for each cluster. This point is considered the spectrum that most closely resembles, on average, all other spectra in the cluster.

3.3 Frame-size-aware Task Mapping

TAPA-CS [7] is able to map tasks (or modules) within multiple multi-die FPGA environments. It is important to optimize the communication flow, both within a single FPGA (across different dies) and across multiple FPGAs due to the frequency degradation observed through inter-die and inter-chip communication. The cost of this communication is encapsulated in the following equation:

$$\sum_{e_{i,j} \in E} e_{ij}.width \times \text{dist}(P_i, P_j) \times \lambda \tag{2}$$

In this equation, $e_{ij}.width$ denotes the bit width of the FIFO that connects tasks v_i and v_j. The scaling factor λ adjusts for the type of communication protocol used, such as Ethernet or PCIe for inter-FPGA communication, and is set to 1 for intra-FPGA (inter-die) communication. The function $\text{dist}(P_i, P_j)$ computes the physical distance between devices, considering the L1 distance for inter-die communication within an FPGA, and topology-based calculations for inter-FPGA communication, for example, using minimal hop counts for daisy chain and bidirectional ring topology [7]. Note that TAPA-CS's task mapping approach is hierarchical, assigning different cost values for inter-FPGA and intra-FPGA communications due to their different physical constraints.

For data structures small enough for single-cycle transmission (as *packets*), this cost estimation works well. However, large data structures challenge HLS's 2048 maximum data width limit, impacting routability. Thus, data is assembled into larger *frames*, as depicted in Fig.3 ❻ .

The cost function 2 only considers the packet size, not the frame size. This cost function is feasible for inter-die communication because the number of Super Long Lines (SSL, cross-die lines) is limited. For intra-FPGA transfers, the frame data can utilize a ping-pong buffer (shared BRAM). However, this function does not adequately model inter-FPGA communication, as it is predominantly managed through message passing. In other words, the physical constraints for inter-FPGA and inter-die communications are different: bandwidth is the primary concern for the former, while bitwidth is for the latter. To better model inter-FPGA communication, SpectraFlux modifies the cost function as follows:

$$\sum_{e_{i,j} \in E} e_{ij}.width \times e_{ij}.depth \times e_{ij}.rate \times \text{dist}(P_i, P_j) \times \lambda \quad (3)$$

Here, $e_{ij}.depth$ represents the depth of the stream, indicating the maximum frame size, and $e_{ij}.rate$ refers to the data rate of the frame. The value of $e_{ij}.rate$ depends on the input dataset. We get an appropriate value estimate by running software simulation when implementing SpectraFlux.

4 EVALUATION AND RESULTS

4.1 Experiment setup

We implemented SpectraFlux employing HLS with the TAPA-CS framework [7], based on Vitis v2023.1. In our experimental setup, we utilized up to four Xilinx Alveo U55c Data Center Accelerator Cards, interconnected via QSFP28 ports using active cables in a bidirectional ring topology. Additionally, these cards can communicate with each other via PCIe Peer-to-Peer (P2P). The benchmarks for CPU and GPU baselines were conducted on a server equipped with a 12-core CPU, 128GB DDR4 memory, and an NVIDIA GeForce RTX 3090 GPU with 24GB RAM.

The datasets used in our experiments are summarized in Table 1 and are publicly accessible from the PRIDE repository. For ease of reference in this paper, they are categorized as (T)iny, (S)mall, (M)edium, and (L)arge based on their size. Notably, the medium dataset has the largest average bucket size. The small dataset is approximately 4.5 times larger than the tiny dataset, yet they contain a similar number of spectra. This disparity arises from variations in spectra length and density. All datasets were preprocessed using HyperSpec [16]. As preprocessing falls outside the scope of this work, we excluded the preprocessing time from the CPU/GPU baselines to ensure a fair comparison.

Table 1: MS Spectra Datasets for Evaluation

Name	Type	PRIDE ID	# Spectra	Bucket Size	Size
PXD-T	Kidney	PXD001468	1.1M	32	5.6GB
PXD-S	Kidney	PXD001197	1.1M	39	25GB
PXD-M	HeLa	PXD003258	4.1M	117	54GB
PXD-L	HEK293	PXD001511	4.2M	77	87GB

4.2 Performance Evaluation

Throughput of Each Component in SpectraFlux: We evaluated the throughput of each component within SpectraFlux for different cluster bucket sizes. For this, 100 buckets were randomly selected at each bucket size from PXD-L (skipping sizes with insufficient buckets). The average runtime for a single bucket is shown in Fig. 6. The throughput of the HD embedding kernel scales almost linearly, being up to 5.6× faster than the clustering kernel for small buckets at its maximum capacity (320). The overhead of clustering with external memory access increases with bucket size, becoming 4.1× slower than on-chip data storage at half the maximum bucket size (160) and rising to 8.9× at the maximum.

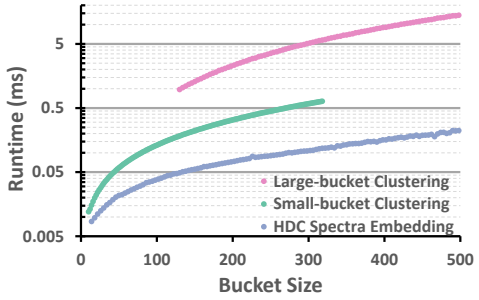

Figure 6: Runtime of components in SpectraFlux when scaling to different clustering bucket bins size

Data Movement Compression with Different Cost Functions: As outlined in Section 3.1, data transmitted from the HD embedding to clustering kernels shares a consistent frame structure and width, with only the number of packets per frame varying. The original cost function led to arbitrary placement decisions for DMC-S or DMC-L in relation to the HD embedding kernel across FPGAs. Figure 7 demonstrates that our proposed cost function (Eq. 2) resulted in a 38% to 73% reduction in inter-FPGA data movement.

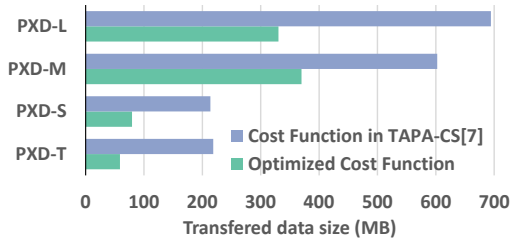

Figure 7: Reduction in Inter-FPGA Data Movement Using Proposed Cost Function

Comparison of Communication Protocols and FPGA Quantity: We evaluated SpectraFlux using both Ethernet and PCIe-based inter-FPGA communications. PCIe P2P, requiring host synchronization and initiation, contrasts with Ethernet's host-independent but logic-intensive setup. We test the following configuration: a single FPGA design with one HD embedding kernel, one small bucket clustering kernel, and three large bucket kernels. The 2-FPGAs-PCIe setup has one HD embedding kernel, one small-bucket clustering, and seven large-bucket clustering kernels. For the Ethernet version, the two and four FPGA setups have one HD embedding kernel, one small bucket, and five and eleven large bucket kernels respectively.

Performance, shown in Fig. 8, revealed that the 2-FPGA setup is only 1.2× faster than a single FPGA for smaller datasets. Ethernet marginally outperformed PCIe for tiny datasets but was up to 22% slower for larger ones due to fewer clustering kernels. The 4-FPGA setup, however, achieved a 2.6× speed increase for larger datasets.

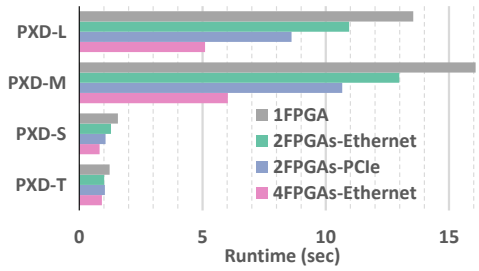

Figure 8: Performance Comparison of SpectraFlux Across Different FPGA Quantities and Communication Protocols.

MS Clustering Speedup: Figure 9 compares SpectraFlux's performance with 4 FPGAs against state-of-the-art FPGA (Spec-HD [6]), GPU (HyperSpec [16]), and CPU (GLEAMS [2]) tools. The results show a 10× to 16× speedup over GLEAMS [2], the fastest known CPU tool for mass spectrum clustering [6]. Additionally, SpectraFlux is 5.8 × −11× faster than the leading FPGA accelerator [6] and 10 × −17× faster than the GPU [16].

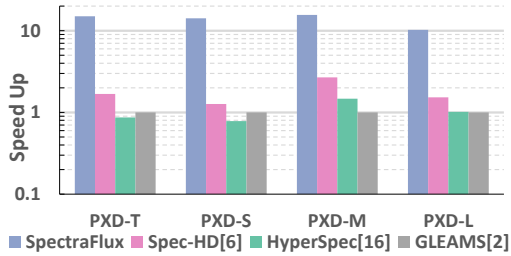

Figure 9: Speedup of SpectraFlux

4.3 Hardware Utilization and Overhead

The resource utilization breakdown for the Ethernet-connected SpectraFlux is shown in Fig.10. FPGA0 is configured with one HD embedding kernel, one small-bucket clustering kernel, and two large-bucket clustering kernels. The remaining three FPGAs are homogenous, each equipped with three large-bucket clustering kernels. The overhead imposed by AlveoLink[14] is approximately 2%-3% for LUTs, registers, and BRAM, along with the usage of one additional HBM channel. Despite the overall FPGA utilization being only around 55%, we encountered congestion near the HBM, which limited our ability to add more kernels. The negligible utilization of DSPs aligns with our expectations, as SpectraFlux primarily performs bitwise operations. The binary nature of the HVs simplifies distance computation to a Hamming distance calculation.

5 CONCLUSION

We introduce SpectraFlux, an FPGA-based framework for accelerating mass spectrum clustering, leveraging the TAPA-CS framework to scale across multiple data center FPGAs. SpectraFlux efficiently handles diverse dataset segments, showing up to a 1.7× speed improvement with 2 FPGAs and 2.6× increase with 4 FPGAs.

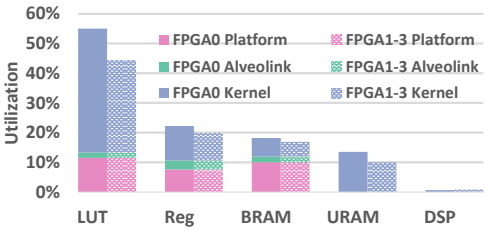

Figure 10: Resource Utilization of SpectraFlux.

We refined the cost function for frame-based inter-FPGA communication, reducing data movement by 73%. It achieves speedups of up to 11×, 17×, and 16× over SOTA FPGA, GPU, and CPU systems. Future work will focus on overcoming limitations like HBM congestion and enhancing performance with multi-channel memory.

ACKNOWLEDGMENTS

This work was supported in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA, PRISM project award 000705769 , and NSF grants 2003279, 1826967, 1911095, 2052809, 2112665, 2112167, and 2100237.

REFERENCES

[1] Wout Bittremieux et al. 2021. Large-scale tandem mass spectrum clustering using fast nearest neighbor searching. *Rapid Communications in Mass Spectrometry* (2021), e9153.
[2] Wout Bittremieux et al. 2022. A learned embedding for efficient joint analysis of millions of mass spectra. *Nature methods* 19, 6 (2022), 675–678.
[3] Ching-Che Chung and Yu-Hsin Wang. 2017. Hadoop cluster with FPGA-based hardware accelerators for K-means clustering algorithm. In *2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. IEEE, 143–144.
[4] Licheng Guo et al. 2023. TAPA: A Scalable Task-Parallel Dataflow Programming Framework for Modern FPGAs with Co-Optimization of HLS and Physical Design. *ACM Trans. Reconfigurable Technol. Syst.* (sep 2023).
[5] Xiyang Luo et al. 2022. A comprehensive evaluation of consensus spectrum generation methods in proteomics. *Journal of proteome research* 21, 6 (2022), 1566–1574.
[6] Sumukh Pinge et al. 2024. SpecHD: Hyperdimensional Computing Framework for FPGA-based Mass Spectrometry Clustering. In *2024 Design Automation and Test in Europe Conference (DATE)*. Valencia, Spain.
[7] Neha Prakriya et al. April 27-May 1, 2024. TAPA-CS: Enabling Scalable Accelerator Design on Distributed HBM-FPGAs. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1* (La Jolla, CA, US,) *(ASPLOS 2024)*. Association for Computing Machinery, New York, NY, USA, 15 pages. https://doi.org/10.1145/3620666.3651347
[8] Neil Scicluna et al. 2015. ARC 2014: A Multidimensional FPGA-Based Parallel DBSCAN Architecture. *ACM Trans. Reconfigurable Technol. Syst.* 9, 1, Article 2 (nov 2015), 15 pages. https://doi.org/10.1145/2724722
[9] David Tabb et al. 2003. Similarity among tandem mass spectra from proteomic experiments: detection, significance, and utility. *Analytical chemistry* 75, 10 (2003), 2470–2477.
[10] Matthew The and Lukas Kall. 2016. MaRaCluster: A fragment rarity metric for clustering fragment spectra in shotgun proteomics. *Journal of proteome research* 15, 3 (2016), 713–720.
[11] Paul Ka Po To et al. 2021. ClusterSheep: A Graphics Processing Unit-Accelerated Software Tool for Large-Scale Clustering of Tandem Mass Spectra from Shotgun Proteomics. *Journal of Proteome Research* 20, 12 (2021), 5359–5367.
[12] UCSD. 2023. MassIVE: Mass Spectrometry Interactive Virtual Environment. https://massive.ucsd.edu.
[13] Lei Wang et al. 2018. msCRUSH: fast tandem mass spectral clustering using locality sensitive hashing. *Journal of proteome research* 18, 1 (2018), 147–158.
[14] Xilinx. 2023. AlveoLink. https://github.com/Xilinx/AlveoLink.
[15] Weihong Xu et al. 2022. A near-storage framework for boosted data preprocessing of mass spectrum clustering. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 313–318.
[16] Weihong Xu et al. 2023. HyperSpec: Ultrafast Mass Spectra Clustering in Hyperdimensional Space. *Journal of Proteome Research* (2023).
[17] Tinaqi Zhang et al. 2023. HD2FPGA: Automated Framework for Accelerating Hyperdimensional Computing on FPGAs. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–9.