SpecHD: Hyperdimensional Computing Framework for FPGA-based Mass Spectrometry Clustering

Sumukh Pinge*, Weihong Xu*, Jaeyoung Kang*, Tianqi Zhang*, Niema Moshiri*, Wout Bittremieux†, Tajana Rosing*
*University of California San Diego, La Jolla, CA 92093, USA
{spinge, wexu, j5kang, tiz014, a1moshir, tajana}@ucsd.edu
†University of Antwerp, 2000 Antwerpen, Belgium
wout.bittremieux@uantwerpen.be

***Abstract*—Mass spectrometry-based proteomics is a key enabler for personalized healthcare, providing a deep dive into the complex protein compositions of biological systems. This technology has vast applications in biotechnology and biomedicine but faces significant computational bottlenecks. Current methodologies often require multiple hours or even days to process extensive datasets, particularly in the domain of spectral clustering. To tackle these inefficiencies, we introduce SpecHD, a hyperdimensional computing (HDC) framework supplemented by an FPGA-accelerated architecture with integrated near-storage preprocessing. Utilizing streamlined binary operations in an HDC environment, SpecHD capitalizes on the low-latency and parallel capabilities of FPGAs. This approach markedly improves clustering speed and efficiency, serving as a catalyst for real-time, high-throughput data analysis in future healthcare applications. Our evaluations demonstrate that SpecHD not only maintains but often surpasses existing clustering quality metrics while drastically cutting computational time. Specifically, it can cluster a large-scale human proteome dataset—comprising 25 million MS/MS spectra and 131 GB of MS data—in just 5 minutes. With energy efficiency exceeding 31× and a speedup factor that spans a range of 6× to 54× over existing state-of-the-art solutions, SpecHD emerges as a promising solution for the rapid analysis of mass spectrometry data with great implications for personalized healthcare.**

***Index Terms*—Mass spectrometry, Proteomics, Spectral Clustering, HD computing, FPGA, Personalized Healthcare.**

I. Introduction

Mass spectrometry (MS) is a cornerstone technique in proteomics research, holding a pivotal role in the advancement of personalized medicine. At its essence, MS offers an intricate view into protein compositions, enabling researchers to dissect the protein compositions of various biological samples. Such insights are foundational to tailoring medical treatments to individual patients, harnessing the specificity of their molecular profiles. With the progressive evolution of MS technologies, there has been a significant surge in data production, with monthly datasets reaching terabytes. Repositories like MassIVE, holding over 500TB of data as of September 2023, exemplify this growth and hold the potential to revolutionize personalized medicine through the discovery of patient-specific biomarkers [1]. However, the intricacies of MS data go beyond volume, encompassing a transformation process that converts a biological sample into digital spectral representations (Fig. 1). This data is subsequently structured into digital formats, such as mzML, mgf, etc. In these formats, m/z ratios are paired with ion intensities, turning spectral peaks into vectors that are well-suited for database searching [2], [3], a task similar to pattern matching in personalized drug discovery datasets. This step matches observed spectra with known peptide sequences, identifying proteins in the sample and bridging raw data to biologically relevant insights. In personalized healthcare, this identification aids in pinpointing disease-specific biomarkers and precise treatment interventions.

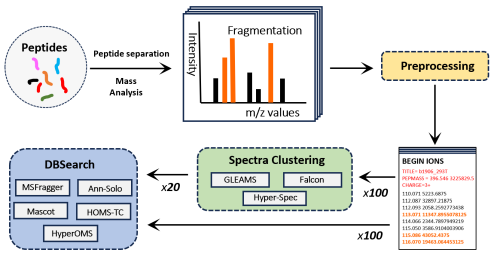


Fig. 1: MS data-analysis pipeline

Despite the prowess of MS, the sheer volume of spectra generated in typical MS experiments poses significant computational challenges, especially in tasks like spectral clustering. In MS/MS spectra, clustering groups alike data into representative consensus spectra. This streamlining not only cuts redundancy and expedites database searching, a major bottleneck in proteomic analysis, but also refines the peptide identification process, potentially halving its runtime [4]. In personalized healthcare settings, the benefits of clustering are evident as expedited data analysis directly impacts the quality and timeliness of patient care; however, despite its advantages, spectral clustering remains underutilized due to its time-consuming nature and limitations of current tools. In response, we present SpecHD, rooted in HDC principles. Designed for FPGA-accelerated MS clustering, SpecHD integrates the HD representation to ensure streamlined binary operations, efficiently harnessing the FPGA's parallel processing, low-latency, and robust computational capabilities. Moreover, it adeptly addresses the often-overlooked computational demands of data preprocessing in MS workflows. The salient features of SpecHD's contributions to MS data processing include:

1) To the best of our knowledge, SpecHD is the first to implement the linkage agnostic Nearest-neighbor Chain Hierarchical Agglomerative Clustering (HAC) using FPGAs. This strategic alignment within the HDC framework

979-8-3503-4859-0/DATE24/© 2024 EDAA

boosts MS clustering speed by 6–54× [5], [6], achieving an energy efficiency 31× greater than current benchmarks.
2) With SpecHD, we synergistically blend near-storage MS preprocessing with FPGA capabilities. Guided by design space exploration, this approach advances hardware and energy efficiency while achieving a 3-10× preprocessing speed up compared to [5], enabling seamless data exchanges between the FPGA and NVMe storage.
3) Deviating from conventional tools and leveraging the HD space, SpecHD uses an innovative approach with streamlined preprocessed spectral data, accelerated clustering, and emphasizes superior clustering quality and database search efficiency over other MS solutions.

II. Related Works

MS clustering presents unique challenges compared to traditional methods due to its high-dimensional and noisy spectral data. Each MS spectrum holds hundreds to thousands of intensity values, emphasizing the need for specialized algorithms attuned to such complexities. Both PIM [7] and FPGA-based solutions, as well as HD computing methods, lean towards K-means [8], [9] and DBSCAN [10], with Hierarchical clustering often overlooked despite its potential. In our MS clustering analysis, we observed the shortcomings of K-means and DBSCAN. In contrast, HAC [11] with complete linkage excelled, offering versatility in cluster shapes, cluster counts, and outlier resilience, proving uniquely suited for MS data nuances [12].

MS clustering solutions face the dual challenge of maximizing clustering quality while minimizing computational time. MaRaCluster [12] uses optimized distance metrics for better clustering quality. Falcon [13] employs hashing for dimensionality reduction and leverages approximate nearest neighbor algorithms for faster computations. MsCRUSH [4] utilizes locality-sensitive hashing to minimize pairwise spectra comparisons, while GLEAMS [6] uses a supervised deep neural network to embed spectra for optimized clustering. However, these approaches often make trade-offs between quality and speed. Bridging the gap between previous methods, HyperSpec [5] stands out as the state-of-the-art (SoA) tool, establishing as our primary point of comparison due to its remarkable speed and commendable clustering performance. HyperSpec employs HDC and, by harnessing GPU acceleration, achieves leading results in speed while preserving the clustering quality. However, a significant concern with HyperSpec and other GPU-based solutions is when datasets surpass the GPU's onboard memory capacity. This constraint inhibits efficient data processing and frequently necessitates data transfers between the GPU and the system memory, leading to performance overheads. The increased power consumption of GPUs, particularly at peak operations, also adds to operational costs [14]. Beyond these challenges, HyperSpec is also dependent on general-purpose libraries, offering two flavours of clustering algorithms: DBSCAN via the cuML library and HAC using the fastcluster library, targeting both GPU and CPU platforms, respectively.

Another critical bottleneck in MS clustering tools is the spectra loading and preprocessing step, which notably consumes a significant portion of the total execution time [15]. The escalating growth of MS data poses a substantial challenge to current clustering solutions, methods reliant on CPU and GPU architectures, making repository-scale clustering increasingly impractical. In contrast, FPGA architectures, being well-suited for scalable and power-efficient solutions, can be custom-configured to specific applications, enhancing data handling and minimizing transfer overheads. Given these challenges, there is a recognized need for near-storage (NS) computing in efficient MS clustering, merging high-quality results with computational efficiency.

III. Methodology and Flow

Our end-to-end HD-based framework delineates three critical stages: MS preprocessing, MS encoding, and MS clustering. A specialized NS framework, MSAS [15] has been employed to enhance performance during MS preprocessing. Both the HD encoder and our novel NN-chain HAC accelerator (Fig.5) for MS clustering are seamlessly integrated within the FPGA architecture. Two distinct strategies guide our approach: 1) A comprehensive end-to-end framework, echoing established clustering tools. Within this framework, raw data undergoes preprocessing in the near-storage accelerator, subsequently leveraging direct peer-to-peer (P2P) transfers to the FPGA. The encoded output then utilizes the High Bandwidth Memory (HBM) to harness its vast bandwidth, laying the groundwork for the acceleration of clustering kernels. 2) An efficiency-driven strategy within the HDC framework: raw data is encoded once, and the preprocessed encoded spectras directly interface the clustering kernels (via P2P). This method is chosen over recurrently initiating the computational pipeline, advocating for a one-time preprocessing followed by subsequent updates, effectively bolstering real-time data analysis.

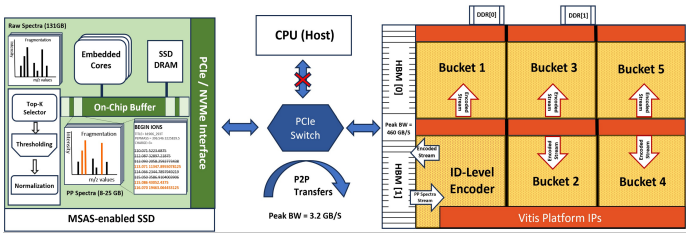


Fig. 2: Top-level dataflow and Kernel organisation

A. Proposed MS Preprocessing Module

In the vast landscape of MS tools, modules such as the Spectra Filter, Top-k Selector, and Scale and Normalization emerge as standard features in MS preprocessing [2], [6] and are integral to MSAS [15]. While the U280 Xilinx Alveo platform prominently supports these functionalities, their synergy with MSAS remains relatively unexplored. The activation of P2P holds a distinct strategic advantage: it enables direct data exchanges between the FPGA and NVMe storage, bypassing the need for intermediary host memory interactions and thereby reducing bandwidth constraints. This feature is also applicable to other Alveo PCIe platforms, contingent upon the host system's ability to support a large physical address space (64GB BAR) [16]. The MSAS accelerator, implemented using CMOS technology, is

integrated into the same die as the SSD's embedded cores. By being on the same die, the MSAS accelerator can directly interface with the global on-chip bus, granting immediate access to data from NAND flashes. This approach empowers it to achieve bandwidths on par with external SSDs, paving the way for faster, more efficient preprocessing of large-scale MS datasets. Within MSAS, the Spectra Filter module plays a pivotal role by filtering out peaks associated with the precursor ion or those with intensities falling below 1% of the base peak, preparing the ground for the Top-k Selector, which employs a streamlined parallel bitonic network. This design minimizes data movement overhead, filtering redundant spectral data early in the preprocessing, thereby promoting efficiency and optimal handling of expansive MS data. Evaluations and advantages of this approach will be elaborated upon in Section IV.

$$\text{bucket}_i = \left\lfloor \frac{(m/z_i - 1.00794) \times C_i}{\text{resolution}} \right\rfloor \quad (1)$$

In the context of large-scale datasets, naive approaches to pairwise spectrum comparisons can rapidly escalate into computational bottlenecks, especially given the constraints of on-chip memory. To mitigate this, we adopt a data organization strategy based on precursor m/z. To manage the computational complexity, we partition the dataset [13] into smaller, discrete 'buckets' calculated as in Eq. 1. Here, C_i represents the charge state of the i^{th} spectrum and 1.00794 is the mass of the charge. The term 'resolution' is used to describe the granularity of the mass spectrometer's measurements, and this value can range from 0.05 to 1. Such an approach proves advantageous for high-throughput MS spectrometers, optimizing the use of computational resources.

B. Proposed MS-spectra Encoder

Each spectrum comprises two vectors corresponding to the m/z ratios and intensities, both sized by `peak_count`. Our aim is to efficiently encode these spectra into a single high-dimensional vector of size D_{hv} using a method that has been proven effective for MS workloads in the HD domain due to its capability to maintain spatial locality [2], [5]. For this purpose, m/z values are quantized to a range represented by `ID[0,f]`, and intensity values to `L[0,q]`. Both vectors, pre-allocated from high-dimensional memory spaces, have a size of D_{hv}. For each pair of m/z and intensity values, bitwise XOR operations are performed on the corresponding vectors from `ID` and `L`. The results are successively accumulated into a single vector until all `peak_count` pairs have been processed. A pointwise majority function is then applied to this aggregated vector, culminating in a refined binarized spectrum hypervector:

$$\text{spectra}_i = \sum_{(i,j)} (I_i \oplus L_j) \quad (2)$$

To accelerate the encoding process, the module employs several hardware-level optimizations designed for FPGA platforms. Specifically, data partitioning directives are applied to the `ID` and `Level` memory arrays via HLS pragmas. This allows for simultaneous multiple accesses to these arrays, thereby facilitating loop unrolling within the `hd_encoding` function. In turn, this minimizes the initiation interval, leading to a significant boost in data throughput. Loop unrolling, enhanced by HLS pragmas, drives parallel processing across `peak_count`. The resultant HD vectors are stored in High Bandwidth Memory (HBM), capitalizing on its massive bandwidth to optimize both memory access and retrieval speeds for the succeeding processing of clustering kernels. Moreover, this optimization compresses our substantial 131GB raw MS dataset down to a streamlined 5GB, as depicted in Fig.(8b). This compact data is efficiently stored in the device's HBM (8GB) for utilization in both previously outlined strategies.

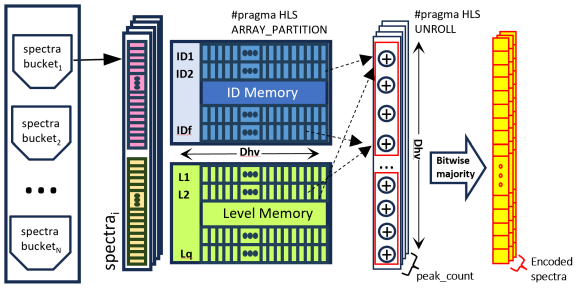


Fig. 3: Proposed HLS optimized spectra encoder

C. Kernel-Level Acceleration for NN-Chain HAC:

HyperSpec [5] represents a significant advancement in MS clustering, yet its dependency on general-purpose libraries and architectures might not optimally tap into the unique strengths of FPGA, especially in terms of parallel processing and real-time capabilities. Classic HAC algorithms face computational bottlenecks due to their $O(n^3)$ time complexity. These algorithms require full matrix updates to calculate pairwise distances between all data points and to identify the minimum distance among all pairs. Addressing these challenges, in a pioneering advancement, SpecHD is among the first to implement the linkage agnostic NN-Chain HAC using FPGAs. Although the foundational concept of the algorithm exists [11], our unique contribution is in its adaptation for FPGA architectures.

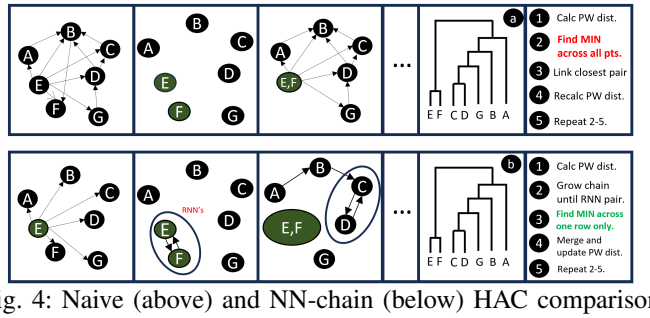


Fig. 4: Naive (above) and NN-chain (below) HAC comparison

The algorithm starts by calculating pairwise distances, akin to traditional methods, but streamlines the following computational steps. The NN-Chain algorithm constructs a local 'chain' of closest points and evaluates this chain to identify Reciprocal Nearest Neighbors (RNN) [11]. Upon identifying an RNN pair, the clusters are merged, and the distance matrix is updated more efficiently, avoiding the need for a full matrix update (Fig. 4).

This targeted approach minimizes redundant calculations and makes NN Chain well-suited for large-scale, data-intensive tasks without compromising clustering quality. Our architecture capitalizes on the robust and adaptable nature of the NN-Chain Algorithm for hierarchical agglomerative clustering (HAC). It is important to note that HAC is inherently sequential with deep dependencies. However, in our approach, we made dedicated efforts to maximize parallelization and enhance optimization. By exploiting the FPGA's intrinsic parallel processing capabilities, our proposed HLS-optimized kernel (Fig. 5) serves as the computational core, parameterized to operate on diverse data structures and design configurations.

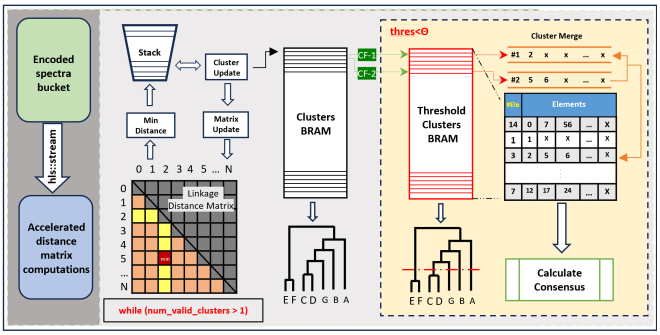


Fig. 5: Proposed FPGA-accelerated NN-Chain HAC architecture

Optimized Distance Matrix Computation: The architecture incorporates specialized modules, including a fast unrolled XOR and an efficient population count (*popcount*) module, both parameterized for D_{hv} bits of dimensionality. A dataflow approach is employed, facilitating task-level parallelism by enabling concurrent execution of both reading the encoded spectra and calculating distances, thereby boosting the efficiency of spectra processing and accelerating the computation of the distance matrix. Due to the inherently $O(n^2)$ nature of the distance matrix, which demands significant storage, only the lower triangular part of the distance matrix is retained, capitalizing on its symmetry. Furthermore, the use of 16-bit fixed-point arithmetic results in a significant reduction in memory footprint while maintaining computational accuracy.

Proposed NN-Chain HAC Architecture (Fig. 5): The process begins by selecting an arbitrary point and calculating its minimum distance to all other points based on a linkage distance matrix. Both the selected point and its nearest neighbor are initially added to a stack. As the algorithm iterates, new elements are added to this stack based on the smallest distance criterion until a reciprocal nearest neighbor is identified. Specifically, if the last index in the stack matches the index of the current minimum distance, the algorithm proceeds to cluster merging and updates the distance matrix.

The algorithm manages two separate sets of clusters. One set, stored in Cluster BRAM, is subject to the exhaustive tree traversal, as local chains cannot always be guaranteed to fall under a predefined distance threshold. Other set only merges if the inter-cluster distance is below this threshold. Each cluster is comprised of three components: element count, the elements, and a correction factor (CF) for adaptive adjustments. Upon identifying an RNN, clusters merge based on their indices by folding the second cluster into the first, updating both their elements and total count. A deleted cluster is effectively removed from future traversals, and its position is replaced by the next cluster in the array. CF's are used to dynamically synchronize these cluster updates. Following each cluster merge, the distance matrix is updated based on the chosen linkage criteria. Our flexible architecture, supporting various linkage criteria like Ward, single, and complete linkage, has demonstrated that complete linkage yields the most reliable results in our implementation. In the concluding steps, the algorithm calculates a consensus cluster by evaluating the lowest average minimum distance to all other spectra within that cluster, based on the original distance matrix. Various optimization techniques, such as memory partitioning and pipelining, are deployed to maximize computational efficiency and throughput. These features make our NN-Chain architecture a robust and adaptable solution for FPGA platforms.

IV. Results

In our experimental setup, we utilized the Xilinx Alveo U280 Data Center Accelerator Card, featuring an HBM2 total capacity of 8GB and a bandwidth of 460GB/s. For our SoA benchmarking, our setup includes a server with a 12-core CPU, 128GB DDR4 memory, and a 2TB NVMe solid-state drive. An NVIDIA GeForce RTX 3090 GPU with 24GB RAM was chosen for a comparative evaluation of SoA GPU tools. We conducted extensive design space exploration for the SSD-level MSAS accelerator during our preprocessing phase, specifically targeting both speed and energy optimization. In terms of software tools, `HyperSpec` [5] served as the SoA GPU tool, `msCRUSH` [4] represented the SoA for CPU solutions, `falcon` [13] was acknowledged for SoA in cluster sizes, and `GLEAMS` [6] was considered for its leading clustering quality. To evaluate the robustness of our approach, we selected datasets intended to highlight the effects of varying sizes and intrinsic characteristics. As seen in Table I, PXD001468 and PXD001197 have comparable spectra counts, but their size disparity underscores the variations within datasets. Such variations can arise from disparities in the number of data points within individual spectra or distinct noise characteristics. Evaluating our preprocessing and clustering modules across these differences offers deeper insight into its overall performance.

A. Clustering quality

The largest dataset used for clustering quality evaluation was the Human Proteome Draft dataset, with corresponding spectrum identifications obtained from the MassIVE reanalysis RMSV000000091.3 using MS-GF+ [17] and matched against the UniProtKB/Swiss-Prot human reference proteome.

*1) **Clustered Spectra vs. Incorrect Clustering Ratio**:* Fig. 6 illustrates the balance `SpecHD` achieves between clustered spectra ratio and incorrect clustering ratio (ICR). The efficacy of a clustering algorithm can be gauged by its ability to achieve a high clustered spectra ratio while maintaining a low ICR ($\sim$1-2%). A robust algorithm ensures high fidelity in results without being overly aggressive, as aggressive clustering may compromise the quality of analyses. `SpecHD`'s performance

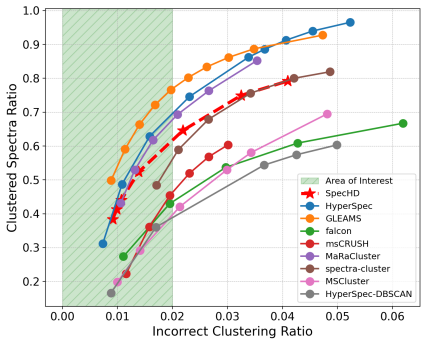


Fig. 6: Clustered spectra ratio vs incorrect clustering ratio

was benchmarked against 9 other tools. With an ICR target of 1%, crucial for maintaining the integrity of downstream analyses, SpecHD achieved an impressive clustered spectra ratio of 45%. This result surpassed tools like msCRUSH [4] and falcon [13], and stood firm against HyperSpec's 48% [5] and MaRaCluster's 44% ICR [12]. Although GLEAMS [6] offered superior ratios, it required considerably more computational effort. In comparison, SpecHD boasted a 54× speedup over GLEAMS. This highlights SpecHD's ability to balance speed and accuracy, evidenced by its 1.5-2× speedup in spectra searching by eliminating redundant searches (ICR = 1-2%).

*2) **Peptide Identification Overlap**:* Clustered consensus spectra, which are vital for unique peptide sequence identification in database searches, can be effectively visualized using Venn diagrams. Benchmarking against PXD000561 dataset, SpecHD displayed promising results: a narrow gap of 1.38% behind GLEAMS [6] for peptides with a 2+ charge, while outshining HyperSpec [5] by 7.33%. For 3+ charge, SpecHD lagged GLEAMS by 3.24% but surpassed HyperSpec by 5.10%. This indicates SpecHD's prowess in balancing both performance and clustering quality. Further insights into SpecHD's metrics revealed its completeness metric at 0.82, a touch beneath the usual 0.85 seen in other tools. This tradeoff allows SpecHD to identify a broader range of unique peptides, reinforcing its potential for comprehensive human proteome studies.

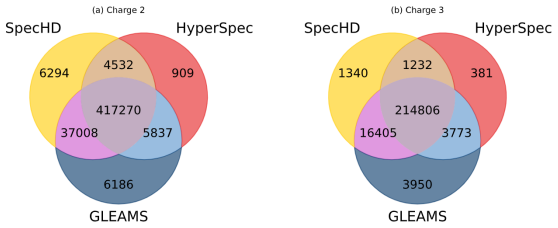
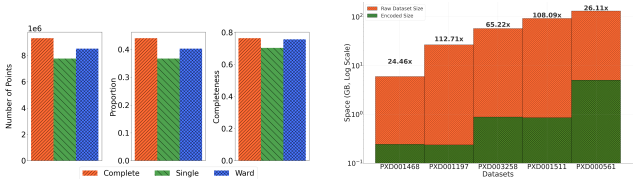


Fig. 7: Overlap of identified unique peptides

B. Data Compression & Design Configurations

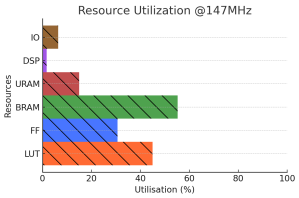
Preliminary tests assessed the best linkage criteria for HAC within our SpecHD NN-Chain FPGA algorithm. Complete linkage was dominant with a 44% clustering ratio and a 0.764 completeness score, followed closely by Ward linkage at 40% and 0.756, whereas single linkage trailed behind. For data compression, $D_{\text{hv}} = 2048$ played a crucial role to maintain a balance of accuracy and utilization, resulting in compression



(a) Linkage Comparison (b) Compression factor

Fig. 8: Linkage Efficacy and Compression-SpecHD

rates ranging from 24× to 108× across datasets. While we expedited certain aspects of the HAC, achieving significant speedup necessitated deploying multiple kernels, helped by the lack of intra-kernel dependencies. The integration of these additional clustering kernels allowed a linear speedup. However, adding more kernels also brought about routing challenges, underscoring the balance between parallelism, speed, and FPGA resource constraints. Presented in Fig.2, our setup, which comprises one encoder and five clustering kernels, registered a LUT usage of 44.97% and a BRAM utilization of 55.26%.



(a) Resource Utilization

Configurations	Parameter
linkage_criteria	complete
θ	0.27
D_{hv}	2048
Q	16
F	4096
max_peak_count	50
max_cluster_size	300

(b) Design Setup

Fig. 9: Design parameters and insights

C. SpecHD vs State-of-the-art solutions

*1) **Pre-processing Results**:* To quantify the efficiency of our preprocessing module, we evaluated it across five different datasets, as shown in Table I. The hardware setup and configurations are based on an Intel SSD DC P4500, and we have emulated the setup as described in [15]. Energy estimates are derived from combining SSD simulation data with the SSD power model referenced [18]. For MS clustering, HyperSpec [5] establishes the preprocessing benchmark by employing multiprocessing to concurrently process files across k distinct CPU cores. We observe a speedup ranging from 3.4-10× over five diverse data-sets when compared to [5], while also achieving a considerable reduction in energy consumption—implications of which will be detailed in subsequent sections.

TABLE I: Preprocessing Performance Metrics

Sample	PRIDE ID	Spectra	Size	PP time(s)	Speedup [5]	Energy(J)
Kidney	PXD001468	1.1M	5.6 GB	1.79	10.0×	17.38
Kidney	PXD001197	1.1M	25 GB	8.22	4.2×	77.27
HeLa	PXD003258	4.1M	54 GB	18.44	4.3×	166.53
HEK293	PXD001511	4.2M	87 GB	28.53	3.4×	268.22
Human	PXD000561	21.1M	131 GB	43.38	8.9×	382.62

*2) **Speedup comparisons**:* In the field of MS-based proteomics, the efficiency of spectral clustering tools is critically measured by runtime, which becomes increasingly important as MS repositories continue to expand. With this in mind, we delve into an end-to-end runtime comparison between SpecHD and several other comparative tools. Across five datasets (Table

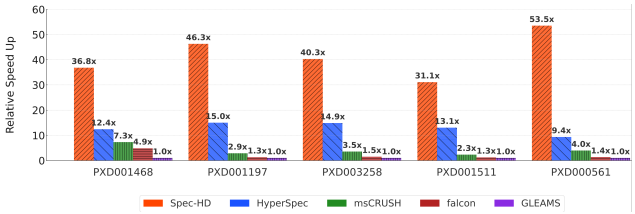


Fig. 10: End-to-end runtime speedup

I), SpecHD achieves remarkable speed-ups, ranging from 31× over GLEAMS for dataset PXD001511 to an impressive 54× for PXD000561. Against HyperSpec, we note a 6× speed-up, solidifying SpecHD's efficiency. Our analysis shows that FPGA systems like SpecHD offer superior speed with a single encoder module, outweighing the flexibility of GPU-accelerated encoding, which allows for on-the-fly reconfigurations.

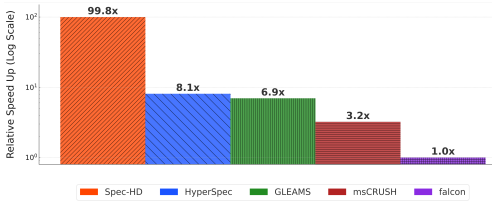
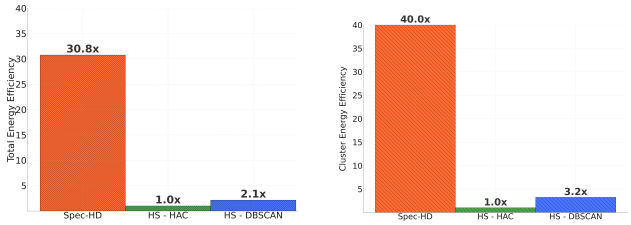


Fig. 11: Standalone clustering speedup for PXD000561

Within the HDC framework, restarting the computational pipeline for every new analysis is inefficient. Instead, a strategy of one-time preprocessing followed by subsequent updates enhances real-time data analysis. Leveraging the impressive data compression metrics, our approach underscores that one-time preprocessing significantly boosts efficiency. When concentrating exclusively on standalone clustering of pre-encoded vectors, the runtime gains are remarkable. SpecHD processed PXD000561 in just 80 seconds, marking a 12.3× speed-up over HyperSpec's 993 seconds and a 14.3× advantage over GLEAMS.



(a) End-to-end clustering (b) Standalone clustering

Fig. 12: Energy efficiency

*3) **Energy efficiency**:* In our assessment, SpecHD's energy efficiency was benchmarked against HyperSpec, utilizing measurement tools such as Intel RAPL for CPU, Nvidia SMI for GPU, and Xilinx XRT for FPGA. In our evaluation of benchmarks, we prioritized runtime efficiency as our analysis indicated that other tools with substantially extended runtimes inherently consumed more energy, making them less comparable in this context. Notably, the DBSCAN variant showcases a runtime three times faster than HAC. However, there's a trade-off in clustering quality, as shown in Fig. 6. In the end-to-end energy efficiency, SpecHD exhibited enhancements of 14× over HyperSpec-DBSCAN and 31× over HyperSpec-HAC, while in the clustering phase, the gains were 12× and 40×, respectively. SpecHD's performance can largely be attributed to FPGA's inherent prowess and its emphasis on NS processing, which counters data transfer limitations prevalent in conventional systems. As a result, SpecHD serves as an excellent choice for applications where energy efficiency is of importance.

V. Conclusion

In this paper, we introduced SpecHD, a novel framework that integrates the strengths of HD computing with FPGA-accelerated architecture, targeting the current inefficiencies in mass spectrometry-based proteomics. In-depth evaluations reveal that SpecHD can efficiently process a vast human proteome dataset in a mere 5 minutes. It outperforms other tools, achieving speedups between 6× and 54×, and showcases an impressive energy efficiency of over 31×, all while retaining the potential for repository scale clustering. Moving forward, it becomes imperative to consider further refinements for SpecHD and contemplate its integration right at the point of data capture within established proteomics processing workflows.

Acknowledgment

This work was supported in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA (SRC grant number - 2023-JU-3135). This work was also supported by NSF grants #2003279, #1911095, #1826967, #2100237, #2112167, #2052809, #2112665.

References

[1] C. A. Ciocan-Cartita *et al.*, "The relevance of mass spectrometry analysis for personalized medicine through its successful application in cancer 'omics'," *Int J Mol Sci*, vol. 20, no. 10, May 2019.
[2] J. Kang, W. Xu *et al.*, "Massively parallel open modification spectral library searching with hdc," 11 2022.
[3] J. Kang *et al.*, "Accelerating open modification spectral library searching on tensor core in high-dimensional space," *Bioinformatics*, Jun. 2023.
[4] L. Wang *et al.*, "mscrush: Fast tandem mass spectral clustering using locality sensitive hashing," *J. Proteome Res.*, vol. 18, pp. 147–158, 2019.
[5] W. Xu *et al.*, "Hyperspec: Ultrafast mass spectra clustering in hyperdimensional space," *J. Proteome Res.*, vol. 22, no. 6, Jun 2023.
[6] W. Bittremieux *et al.*, "A learned embedding for efficient joint analysis of millions of mass spectra," *Nat Methods*, vol. 19, pp. 675–678, Jun 2022.
[7] M. Imani *et al.*, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *MICRO*, 2020.
[8] M. Imani, S. Salamat *et al.*, "Fach: Fpga-based acceleration of hyperdimensional computing by reducing computational complexity," 2019.
[9] S. Salamat, M. Imani, and T. Rosing, "Accelerating hyperdimensional computing on fpgas by exploiting computational reuse," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1159–1171, 2020.
[10] N. Scicluna *et al.*, "Arc 2014: A multidimensional fpga-based parallel dbscan architecture," *ACM Trans. Reconfigurable Technol. Syst.*, nov 2015.
[11] F. Murtagh and P. Contreras, "Methods of hierarchical clustering," 2011.
[12] M. The and L. Käll, "Maracluster: A fragment rarity metric for clustering fragment spectra in shotgun proteomics," *JPR.*, Mar 2016.
[13] W. Bittremieux *et al.*, "Large-scale tandem mass spectrum clustering using fast nearest neighbor searching," *Rapid Commun. Mass Spectrom.*, 2021.
[14] A. Pattnaik *et al.*, "Scheduling techniques for gpu architectures with processing-in-memory capabilities," in *PACT*, 2016.
[15] W. Xu, J. Kang, and T. Rosing, "A near-storage framework for boosted data preprocessing of mass spectrum clustering," in *DAC*, NY, USA, 2022.
[16] Xilinx, "Pcie peer-to-peer communication," 2022, xRT Documentation.
[17] S. Kim and P. A. Pevzner, "Ms-gf+ makes progress towards a universal database search tool for proteomics," *Nat Commun*, vol. 5, Oct 2014.
[18] M. Jung *et al.*, "Nandflashsim: High-fidelity, micro-architecture-aware nand flash memory simulation," *ACM Transactions on Storage*, Jan 2015.