FPGA Acceleration of Pairwise Distance Calculation for Viral Transmission Clustering

Sahand Salamat, Niema Moshiri, Tajana Rosing
Computer Science and Engineering Department, UC San Diego, La Jolla, CA 92093, USA
{sasalama, niema, tajana}@ucsd.edu

Abstract—**Reconstructing the evolutionary history of closely-related viral strains enables the identification and epidemiological linkage of infected patients in an outbreak, but phylogenetic inference is often too computationally time-consuming to be performed in real-time on large datasets. To combat this, molecular surveillance methods turn to inferring epidemiological linkage from pairwise distances computed between sequences directly. However, during the COVID-19 pandemic, the ultra-large datasets made available from global viral sequencing and data sharing efforts render existing CPU implementations of sequence-based molecular surveillance too slow for real-time analysis. FPGAs are widely used to accelerate bioinformatics applications as they are becoming increasingly available either in local computer systems or in cloud data centers. Providing high parallelism, FPGAs are well-suited for pairwise distance computations. In this work, we introduce, FANTAIL, a highly parallelized FPGA-based Accelerator for computing pairwise distaNce for viral TrAnsmIssion cLustering based on Tamura-Nei 93 (TN93) model. Compared to the state-of-the-art multi-threaded CPU baseline running on an 8-core Intel Core-i7 CPU, FANTAIL provides $56\times$ speedup and $168.1\times$ energy reduction, while providing the same results. Moreover, FANTAIL is able to process a $6.3\times$ larger dataset under 20 minutes compared to the CPU baseline.**

I. Introduction and Related Work

During viral epidemics, understanding how the virus is spreading through a population helps public health officials to control and limit the spread of the pathogen. Answering questions such as "Who infected whom?", "Where do individual outbreaks exist within a given community?", and "How effective are specific treatments or interventions at slowing the spread of the disease?" is crucial to help officials to make critical decisions [1] such as mandating wearing face masks and prioritizing access to vaccines and treatments.

RNA viruses such as Human Immunodeficiency Virus (HIV), Hepatitis C Virus (HCV), influenza, and SARS-CoV-2 (Covid-19) have fast-evolving genomes [2], [3], [4]. The evolutionary history of the virus is constrained by its transmission history [5]. Thus, phylogenetic reconstruction can be utilized to retrace the transmission events of an epidemic [1], [6], [7]. phylogenetic reconstruction is the process of inferring the evolutionary history of a collection of viral genomic sequences. The pairwise distance of genomic sequences increases as the individuals are further apart in the epidemic. Specifically, for individuals who were members of the same outbreak, the viral sequences will have low evolutionary distance. However, for those sequences obtained from individuals from different outbreaks will have high evolutionary distance. Individuals with low distance between their viral sequences are called epidemiologically "linked," and groups of linked individuals can be considered to be "transmission clusters" [1]. Unfortunately, phylogenetic inference is NP-Hard, and even the fastest heuristic tools such as FastTree [8] are still computationally complex such that they are the computational bottlenecks in viral sequence analysis workflows. Therefore, in large-scale pandemics (e.g. COVID-19), real-time surveillance of the outbreaks using the evolutionary trees is not feasible due to the complexity of the problem.

HIV-TRACE is a standard tool for performing real-time molecular epidemiology of HIV and other rapidly-evolving pathogens. HIV-TRACE instead of performing phylogenetic inference, it estimates pairwise evolutionary distances under the Tamura-Nei 93 (TN93) model of nucleotide evolution directly from Hamming distances computed from pairs of sequences [1], [9]. Given a pair of aligned sequences u and v, HIV-TRACE computes the pairwise Hamming distance (i.e., number of mismatches) between u and v. It then estimates pairwise evolutionary distance from the computed Hamming distance under the TN93 model of evolution. The TN93 model is the most general nucleotide substitution model and evolutionary distance under the TN93 model can be estimated directly from the Hamming distance of a single pair of sequences [1]. Pairwise TN93 distances are computed for all pairs of individuals, and a graph is constructed. In the transmission graph, nodes are individuals and edges connect pairs of individuals, the TN93 distance of which is less than a threshold (1.5% is the default selection in HIV-TRACE). Each connected component of the resulting graph is output as a transmission cluster.

Monitoring transmission clusters over time can provide significant insights about changes in properties of an epidemic. Epidemic properties such as population size, geographical structure of the population, impacts of a specific treatment or prevention on the population help public health officials to make critical decisions[10], [7], [11]. Transmission clustering has been widely used in various areas, including criminal investigation [12], [13], [14], [15] and public health investigation [16], [17].

With the increasing accessibility and affordability of high-throughput sequencing, the number of available viral genome sequences is growing rapidly, which in turn creates a need for scalable methods to enable real-time analysis and dissemination. For instance, so far, there are approximately 2 million SARS-CoV-2 genome sequences publicly available, and this number is rapidly increasing every day [18]. Because of the need to compare all pairs of sequences in a given dataset, the complexity of viral transmission clustering increases quadratically with respect to the number of sequences in the database [19]. Therefore, processing real-world size databases of viral sequences requires days or even weeks of processing which makes the real-time surveillance infeasible. The ability to conduct such analyses in a reasonable time is primarily limited by the performance of the existing software tools, optimized for CPUs.

FPGAs have been widely used to accelerate a wide variety of bioinformatics applications [20], [21], [22], [23]. As the size of the generated data has drastically increased, general-purpose processors have had difficulty keeping up. FPGAs have recently become more prevalent and have been widely deployed in data centers to provide easier access for users to implement their accelerators on cloud FPGAs [24], [25], [26], [27], [28], [29]. FPGAs are well-suited for accelerating the pairwise distance calculation of genomics sequences due to their high parallelism. In this paper, we propose an FPGA-based accelerator, FANTAIL, that computes the pairwise distances between the viral sequences for viral transmission clustering. FANTAIL utilized the High Bandwidth Memory (HBM), available on the FPGA board, to reduce the memory bottleneck. HBM, available on Xilinx UltraScale+ devices, provides significantly higher memory bandwidth compared to conventional DRAMs [30], [31]. The summary of our contributions is as follows.

- We proposed a novel architecture to calculate the pairwise distance of viral sequences based on TN93 algorithm [9]. The proposed architecture utilizes the FPGA's available resources to parallelize computations.
- FANTAIL utilizes High Bandwidth Memory to provide faster memory access to maximize the parallelization and consequently significantly accelerate the entire application.
- We evaluated FANTAIL using the available Covid-19 and HIV sequence libraries. FANTAIL is able to process a 6.3× bigger datasets in under 20 minutes as compared to the state-of-the-art multi-threaded CPU baseline. FANTAIL also provides 56× speedup and 168× energy reduction as compared to the CPU baseline.

II. PROPOSED METHOD

Calculating the distance between every pair of viral sequences in the database is the computational bottleneck of transmission clustering. FANTAIL calculates the pairwise distance on FPGA to parallelize these computations. It goes through the entire dataset, calculates the pairwise similarity distances, and transfers the results to the host CPU to perform the transmission clustering. In the following subsections, we first introduce the high-level architecture of FANTAIL. Then we explain the details of each module and the optimizations we have applied to accelerate the entire application.

A. FANTAIL High-Level Architecture

Algorithm 1 is the simplified algorithm of the main pairwise distance calculation. First, two sequences (*Seq1* and *Seq2*) are read from the library. The lengths of both sequences are defined by the length of the viral sequence, and are provided as an input to FANTAIL. After reading the sequences, the tool keeps track of the number of repetitions of corresponding nucleotides in the sequences in *Ncount*. *Ncount* is a two dimensional array where *Ncount*$[N_1][N_2]$, shows the repetition number of N_1 nucleotide in the *Seq1* sequence when the corresponding nucleotide in *Seq2* is N_2. For instance, after processing a pair of sequences, *NCount[A][C]* shows the repetition number of A nucleotides in the first sequence where their corresponding nucleotide in the other sequence is C. To calculate the *NCount* values, for i^{th} nucleotides of both sequences, *Ncount*$[Seq1[i]][Seq2[i]]$ is incremented.

Algorithm 1 Simplified Algorithm of Pairwise Distance Computation

```
1:  procedure PWD($Seq1,\ Seq2,\ AmbMode$)
2:      NCount[4][4]=0
3:      MMC=0                    ▷ $MMC \rightarrow MismatchCounter$
4:      for $i < len(Seq1)$ do    ▷ $len(Seq1) == len(Seq2)$
5:          $NCount[Seq1[i]][Seq2[i]]++$
6:          if $Compare(Seq1[i],\ Seq2[i])$ then
7:              MMC+= $Distance(Seq1[i], Seq2[i], AmbMode)$
8:          end if
9:      end for
10:     CalculateDistance(NCount, MMC)
11: end procedure
12: procedure FANTAIL($SeqLibrary,\ AmbMode$)
13:     D=[]                     ▷ Calculated Pairwise Distances
14:     for $i = 0:\ i < len(SeqLibrary)$ do
15:         Seq1 = read $i^th$ sequence in the library
16:         for $j = i:\ j < len(Seq1)$ do
17:             Seq2 = read $j^th$ sequence in the library
18:             $\{t1, t2\} = PWD(Seq1,\ Seq2,\ AmbMode)$
19:             $D.append(t1)$; $NCount.append(t2)$
20:         end for
21:     end for
22:     return D, NCount
23: end procedure
24: procedure DISTANCECALCULATION($D,\ NCount$)
25:     for e doach element in D
26:         Score = TN93($D,\ NCount$)
27:     end for
28:     return D
29: end procedure
```

The number of mismatches between the corresponding nucleotides of two sequences, *D*, has to also be calculated for finding the genomic distance of two sequences. For each pair of sequences, the *NCount* and *D* are used in the TN93 model (line 26 in Algorithm 1) to calculate the final genomic distance [9]. The TN93 model consists of logarithmic operations which can be executed more efficiently on CPUs. Since the pairwise distance calculations are independent, they can be parallelized and executed in different threads. Thus, FANTAIL calculates *D* and *NCount* on FPGA and sends them to the host CPU, and the CPU calculates the final distance. However, in many cases, *D* and *NCount* can be directly used to infer the transmission clusters without calculating the genomic distances.

Characters A, C, G, and T (U in RNA sequences) are being used to represent the four nucleotides of a DNA/RNA molecule. However, due to the lack of precision, the imperfections in sequencing machines, and the experimental challenges, the sequencing machines cannot precisely detect all nucleotides. Table I shows the ambiguity characters and their corresponding nucleotides. Ambiguity characters W, S, M, K, R, Y are used to represent positions when there is some uncertainty between two nucleotides. For instance,

TABLE I: Ambiguous character table, their corresponding nucleotides and FANTAIL encoding.

Symbol	Description	Encoding	Symbol	Description	Encoding
A	Adenine	0001	M	A or C	0011
C	Cytosine	0010	R	A or G	0101
G	Guanine	0100	W	A or T	1001
T	Thymine	1000	S	C or G	0110
B	C, G, or T	1110	Y	C or T	1010
D	A, G, or T	1101	K	G or T	1010
H	A, C, or T	1011	N	A, C, G, or T	1111
V	A, C, or G	0111	-	Gap	0000

the R nucleotide represents that the nucleotide is either A or G. Characters B, D, H, V are used when there is only confidence that a position is not one of the four nucleotides. For instance, character B presents C, G, and T nucleotides (i.e. the nucleotide is not A). There are 16 primary and ambiguous nucleotides that can be represented by 4 bits. In FANTAIL we encode each nucleotide to a 4-bit binary number to be processed in the FPGA for pairwise distance calculation. Thus, we take into account the ambiguous nucleotides into account during the sequence matching.

To support a wider variety of biological studies, FANTAIL supports different modes, dubbed *AmbMode*, to deal with ambiguous nucleotides in sequence matching. User can select *AmbMode* from the following list {Exact, Resolve, and Average}. In the Exact mode, two nucleotides are matched only if they are represented with the same character ($Distance(A, R, Exact) = 1$). Resolve mode resolves the ambiguous nucleotide to minimize the distance; thus, ambiguous nucleotides are matched with any primary nucleotide in its ambiguity list. For instance, A and R nucleotides are matched in Resolve mode ($Distance(A, R, Resolve) = 0$). In Average mode, the distance of ambiguous nucleotide with its building primary nucleotide is half of that with other nucleotides ($Distance(A, R, Average) = 0.5$, $Distance(C, R, Average) = 1$).

FANTAIL adopts two levels of parallelism, enabled by FPGA abundant resources, to calculate the pairwise distance. It reads multiple sequences and calculates the pairwise distance of one of the sequences against each of the other sequences. To calculate a single pairwise distance, FANTAIL parallelizes the sequence matching and comparing multiple nucleotides at every clock cycle to calculating *MMC*, and *Ncount*. Due to the large size of the viral sequence library, FANTAIL stores the library into the FPGA off-chip memory (DRAM or HBM). Then it loads multiple sequences to the FPGA on-chip memory to provide significantly higher bandwidth to calculate the pairwise distance more efficiently. Reading the sequences from DRAM limits the performance of pairwise calculation due to the long access time to the FPGA DRAM.

FANTAIL utilizes HBM available on the FPGA board to reduce memory access time as HBM provides significantly higher bandwidth than DRAM. The achieved memory bandwidth during the runtime depends on the data access pattern. Random memory accesses take much longer time than the sequential accesses. Although the bandwidth of each HBM channel is sometimes even less than that of DRAM, as HBM has a significantly higher number of channels, it provide significantly higher bandwidth. Fully utilizing the HBM bandwidth requires simultaneous memory access to each HBM channel. FANTAIL takes advantage of independent pairwise calculation of different sequences to fully utilize the available HBM bandwidth.

Figure 1 shows the high-level architecture of FANTAIL optimized for maximizing the memory bandwidth available on FPGA boards. To take advantage of all the memory bandwidths available for the FPGA (both DRAM and HBM), it stores a copy of the viral sequence library on both FPGA DRAM and HBM. Assuming the FPGA board has C_{DRAM} DRAM channels, and C_{HBM} HBM channels, FANTAIL loads a single viral sequence from DRAM and then streams in C_{HBM} different sequences from HBM. FANTAIL consists of C_{HBM} *PWD engines*, each of which calculates the pairwise distance between a sequence, read from HBM, and the sequence read from DRAM. Each PWD engine calculates the pairwise distance independent of the other engines and writes the calculated distance into the output buffer. Then the output buffer module transfers the calculated distances to the FPGA DRAM.

Each PWD engine consists of an AXI interface to stream in a sequence from HBM. Reading a stream requires $\frac{SequenceSize}{AXIDataWidth}$ read transactions. Since these data read requests are sequential, the AXI interface can use burst transactions to minimize the data transfer latency. In FANTAIL the AXI interface reads 512 bits in every transaction. The PWD engine reads 512 bits of a viral sequence from the library every clock cycle (if the data is not available in a clock cycle for any reason, the engine halts) and then it executes the similarity matching between the 512 bits of the sequence and the corresponding nucleotides of the shared sequence. The parallel comparator module performs the element-wise comparison between $\frac{512bit}{4bit} = 128$ nucleotides. The output of each element-wise comparison is the distance of the nucleotide pair. If the nucleotides are matched, the output is 0; if they are partially matched, and the mode is "average" the output is 1; when they are not matched, the output distance is 2.

In addition to outputting the 128 2-bit parallel distances, the parallel comparator module counts the nucleotides and calculates the *NCount* as explained in the algorithm 1. The 128-input tree-adder module consists of 7 pipelined layers of adders. The first layer compromises 64 adders that add 128 2-bit inputs and outputs 64 3-bit results, which are then aggregated in the next layers of adders; eventually, after 7 layers of adders, the output shows the partial distance between 128 nucleotides of the global sequence and the local sequence. To calculate the pairwise distance between the local sequence and the global sequence, the partial distances in each iteration accumulates in an adder. After processing all the nucleotides of the sequences, each PWD engine writes the pairwise distance to the output buffer, which will then be transmitted to the FPGA DRAM.

B. Module Details

The viral sequence library is usually in FASTA format, which is a text file representing nucleotides with characters. Even though text files can be efficiently processed in CPUs, this file format is not optimal for FPGAs . FANTAIL converts the FASTA library file to binary representation suitable for processing in FPGAs. To convert the FASTA file to binary, we use an encoding approach where each nucleotide is mapped to a 4-bit binary number. The encoding process considers

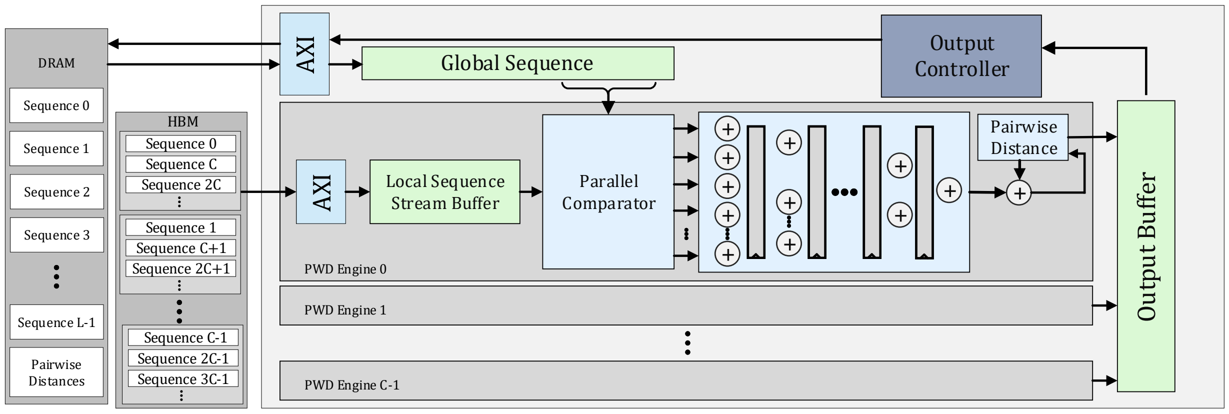

Fig. 1: Overview of the proposed FANTAIL architecture.

the pairwise distance operations to minimize the hardware complexity of the operations. FANTAIL encoding keeps the information of the corresponding nucleotides in the binary representation of the ambiguous nucleotides.

Table I shows the unique mapping of nucleotides (primary and ambiguous) to 4-bit binary numbers. The primary nucleotides (A, C, G, and T) are represented using one-hot encoding. The binary representation of each primary nucleotide has only one 1 digit. A, C, G, and T nucleotides are represented as 0001, 0010, 0100, and 1000 binary numbers, respectively. The binary representation of each ambiguous nucleotide is selected by ORing their corresponding primary nucleotides. Therefore, in the binary representation of ambiguous nucleotides, 1 digits show the corresponding primary nucleotides. For instance, nucleotide W corresponds to A and T nucleotides; thus, encoded representation of nucleotide W is achieved by ORing the binary representation of A and T ($W \rightarrow 0001\&1000 = 1001$).

The main advantage of the proposed encoding approach is the simple similarity comparison. If two nucleotides have digit 1 in the same position, they are partially matched, and if they are equal, they are exactly matched. The parallel comparator module takes advantage of simplified nucleotide comparison to provide a fast and energy-efficient pairwise comparison. Figure 2(a) shows the detailed architecture of the parallel comparator module. The parallel comparator module consists of 128 comparator modules, in addition to a series of adders to count the number of nucleotides in parallel. Figure 2(b) shows the architecture of the comparator module. The comparator module gets two nucleotides along with the *AmbMode*, which instructs how to resolve the ambiguous nucleotides and calculates the distance between two nucleotides. The comparator module first ANDs both nucleotides; if the outputs of the AND gates are equal to zero, it means the nucleotides have no primary nucleotide in common. However, if the output of the AND gates is non-zero, the nucleotides are partially/exactly matched. To distinguish if the nucleotides are partially or exactly matched, the comparator module XORs the inputs, and if the outputs of the XORs are zero and the output of ANDs is non-zero, the nucleotides are exactly matched. Then the output of the comparators is connected to a multiplexer that selects the appropriate distance based on AmbMode.

Figure 2(c) shows the architecture of the Nucleotide counter module. The PWD engine streams in 128 nucleotides of the local sequence every cycle. The nucleotides pass through the Nucleotide counter module, where the module counts the number of nucleotides in the local and global sequence as explained in Algorithm 1. The Nucleotide counter module comprises eight series of comparators. All the nucleotides of both sequences are compared to A, C, G, and T nucleotides. Each series of comparator output represents which elements of the input sequence are equal to a specific nucleotide. For instance, the first series of comparators in Figure 2(c) compares 128 elements of the global sequence with the A nucleotide and outputs a 128-bit number where each bit shows whether the corresponding nucleotide in the global sequence is A or not. The outputs of the series of the comparators are then passed through 16 series of AND gates. The outputs of each AND series are then aggregated together and accumulated in adders to compute the number of each nucleotide combination.

III. EXPERIMENTAL RESULTS

A. Experimental Setup

FANTAIL accelerates the pairwise distance calculation for viral transmission clustering using TN93 algorithm [9] on FPGA. FANTAIL is implemented in C++ and synthesized using the Vivado High-Level Synthesis tool (HLS) and integrated with the host code using Xilinx Vitis Accel 2019.2 [33]. FANTAIL runs on U280 FPGA board with 32 GB of DRAM and 8 GB of HBM [34]. The host code is written in OpenCL, which is responsible for transferring the viral library from the storage device to the FPGA off-chip memory (DRAM and HBM). The host code also initiates the kernel and transfers the calculated results back to the host memory to further process the data. To measure the performance of calculating the pairwise distance for the entire viral sequence library, we used OpenCL event profiling.

We report end-to-end execution time of FANTAIL, including reading the data from the FPGA off-chip memory, the computation time, and writing the results to the FPGA off-chip memory, then calculating the actual pairwise distances on the host CPU by using the output of the FPGA. To evaluate the energy efficiency of FANTAIL, we measure the power consumption of the FPGA (including its off-chip memory) by using the provided API by Xilinx Runtime Library (XRT) [35]. We compare the performance and energy efficiency of FANTAIL with the state-of-the-art TN93 tool [32] optimized for multi-thread execution running on Intel i7-8700K CPU

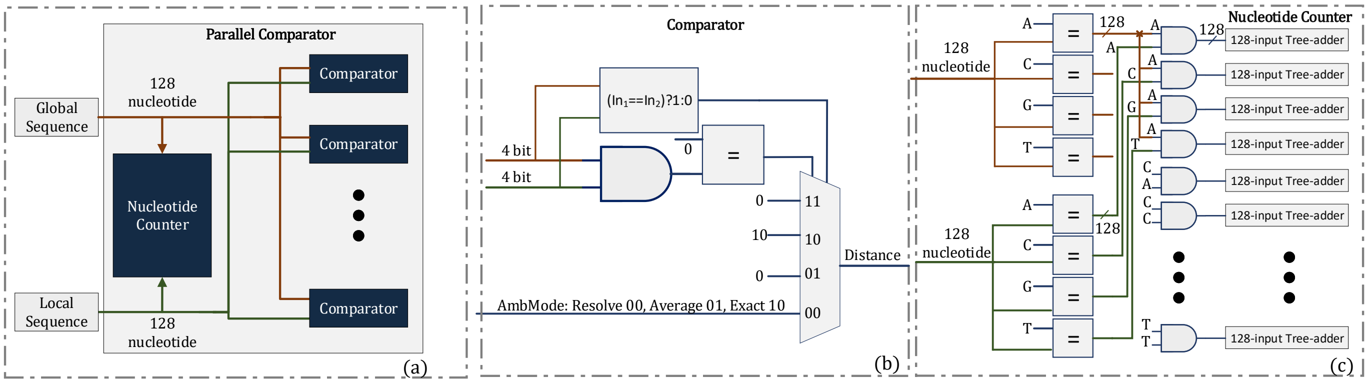

Fig. 2: (a) The architecture of the parallel comparator, (b) the architecture of a single comparator, (c) The architecture of the nucleotide counter.

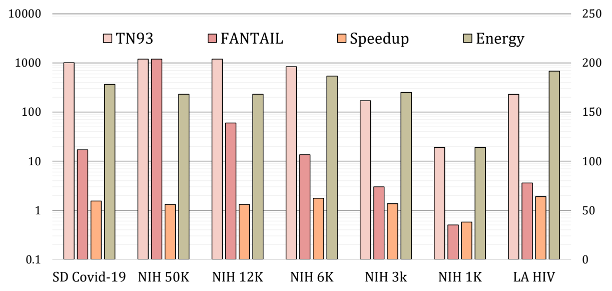

Fig. 3: Performance of FANTAIL compared to the state-of-the-art TN93 implementation [32].

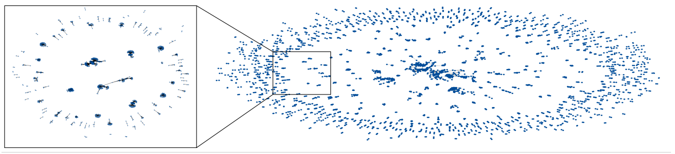

Fig. 4: Covid-19 transmission clusters of San Diego county sequence library.

with 16GB memory. FANTAIL provides the same results as the CPU baseline. For our evaluation, we used the publicly available datasets for HIV [36] and Covid-19 sequences from NIH [18] and City of San Diego [37].

B. FANTAIL *Results*

Figure 3 shows the execution time of FANTAIL and the multi-threaded CPU-based TN93 tool [32] for different viral sequence libraries. The figure also shows the performance improvement and energy reduction of FANTAIL compared to the TN93 baseline. The left Y-axis, which is in logarithmic scale, shows the execution time of the CPU baseline and FANTAIL for different viral sequence libraries. The right Y-axis shows the speedup and energy reduction of FANTAIL compared to the CPU baseline. We tested the performance of FANTAIL for the San Diego Covid-19 library (SD Covid-19) with 7813 sequences as well as randomly selecting 1000 (1K), 3K, 6K, 12K, and 50K sequences from the NIH Covid-19 library. We also performed the pairwise distance calculation for the Los Alamos National Laboratory HIV library (LA HIV) with 4362 sequences. For Covid-19 sequence libraries, the TN93 tool [9] is unable to calculate all the pairwise distances in 20 minutes. Therefore, the execution time for NIH 50K and NIH 12K is not available. The biggest dataset that the TN93 tool [9] can process is the SD Covid-19 library, which took 16 minutes and 50 seconds. However, FANTAIL is able to process 50k Covid-19 sequences in less than 20 minutes.

FANTAIL shows 56× speedup and 168.1× energy reduction, on average for all the libraries, compared to the CPU-based TN93 [9] baseline; note that for NIH 12k and NIH 50K we used the average speedup and energy reduction since the execution time of TN93 was not available (N/A). FANTAIL uses 54% of the FPGA LUTs and 36% of FPGA on-chip memories (BRAMs and URAMs). Thus, FANTAIL performance is limited by the FPGA off-chip memory bandwidth. Although FANTAIL is compatible with FPGA boards without HBM, it provides significantly higher performance ($\frac{HBMBW}{DRAMBW}\times$) on a system with HBM compared to a system with only DRAMs.

For the smaller number of sequences, 1k, FANTAIL shows less performance improvement compared to larger libraries (38× compared to 59.5× for SD Covid-19), since TN93 is more efficient when the size of the library is smaller. However, FANTAIL has a regular data access pattern that is independent of the database size as long as it fits into the FPGA off-chip memory (8 GB). For larger viral libraries, data should be divided into smaller batches so that the pairwise distances can be calculated over multiple iterations. In pairwise calculation of the HIV dataset (LA HIV), FANTAIL shows 63.8× speedup and 191.7× energy reduction compared to the CPU baseline. For calculating the pairwise distance of HIV sequence, FANTAIL uses the same hardware as the one used for Covid-19 sequences, as the length of the viral sequences can be set during the runtime.

C. *Covid-19 Transmission Clustering using* FANTAIL

Figure 4 shows the transmission graph of the San Diego Covid-19 sequence library. FANTAIL calculates the pairwise distance, and any pair of sequences with a distance less than a threshold are connected to each other. In Figure 4 we used 0.0001 as the threshold. The right figure shows the entire transmission clusters, where there are a few huge clusters where each person infects many people; additionally, there are many small clusters in which each person infects only a few people. Therefore, understanding the demographics of people in the huge clusters where they infect many people helps health officials to control the pandemics.

IV. Conclusion

Clustering the transmission history of pathogens helps health officials to make critical decisions to limit the spread of diseases. By pairwise comparison of viral sequences, we can reconstruct the transmission events, as members of the same local outbreak will have low evolutionary distance. However, as the number of viral genome sequences grows rapidly, it is impossible to perform real time inference on large datasets. Our FANTAIL is a high-performance and energy-efficient FPGA-based accelerator for pairwise distance calculation for viral transmission clustering. FANTAIL provides $56\times$ speedup and $168\times$ energy reduction compared to the state-of-the-art multi-threaded CPU baseline running on an 8-core Intel Core-i7 CPU. In other words, FANTAIL is able to process a $6.3\times$ larger dataset than the CPU baseline with the same execution time, while providing equally accurate results.

Acknowledgements

This work was supported in part by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, in part by SRC Global Research Collaboration (GRC) grant, DARPA HyDDENN grant, and also NSF grants #2028040, #1730158, #1911095, #2003279 and #1826967.

References

[1] S. L. Kosakovsky Pond, S. Weaver, A. J. Leigh Brown, and J. O. Wertheim, "Hiv-trace (transmission cluster engine): a tool for large scale molecular epidemiology of hiv-1 and other rapidly evolving pathogens," *Molecular biology and evolution*, vol. 35, no. 7, pp. 1812–1819, 2018.

[2] M. Bonsignori, H.-X. Liao, F. Gao, W. B. Williams, S. M. Alam, D. C. Montefiori, and B. F. Haynes, "Antibody-virus co-evolution in hiv infection: paths for hiv vaccine development," *Immunological reviews*, vol. 275, no. 1, pp. 145–160, 2017.

[3] S. D. Frost and E. M. Volz, "Modelling tree shape and structure in viral phylodynamics," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 368, no. 1614, p. 20120208, 2013.

[4] D. Corcoran, M. C. Urban, J. Wegrzyn, and C. Merow, "Virus evolution affected early covid-19 spread," *medRxiv*, 2020.

[5] N. Moshiri, M. Ragonnet-Cronin, J. O. Wertheim, and S. Mirarab, "FAVITES: simultaneous simulation of transmission networks, phylogenetic trees, and sequences," *Bioinformatics*, p. bty921, 2018.

[6] M. K. Grabowski and A. D. Redd, "Molecular tools for studying hiv transmission in sexual networks," *Current Opinion in HIV and AIDS*, vol. 9, no. 2, p. 126, 2014.

[7] J. O. Wertheim, A. J. Leigh Brown, N. L. Hepler, S. R. Mehta, D. D. Richman, D. M. Smith, and S. L. Kosakovsky Pond, "The global transmission network of hiv-1," *The Journal of infectious diseases*, vol. 209, no. 2, pp. 304–313, 2014.

[8] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree 2 - Approximately maximum-likelihood trees for large alignments," *PLoS ONE*, vol. 5, no. 3, 2010.

[9] K. Tamura and M. Nei, "Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees.," *Molecular Biology and Evolution*, vol. 10, no. 3, pp. 512–526, 1993.

[10] S. J. Little, S. L. K. Pond, C. M. Anderson, J. A. Young, J. O. Wertheim, S. R. Mehta, S. May, and D. M. Smith, "Using hiv networks to inform real time prevention interventions," *PloS one*, vol. 9, no. 6, p. e98443, 2014.

[11] J. O. Wertheim, A. M. Oster, J. A. Johnson, W. M. Switzer, N. Saduvala, A. L. Hernandez, H. I. Hall, and W. Heneine, "Transmission fitness of drug-resistant hiv revealed in a surveillance system transmission network," *Virus evolution*, vol. 3, no. 1, p. vex008, 2017.

[12] A. Blanchard, S. Ferris, S. Chamaret, D. Guétard, and L. Montagnier, "Molecular evidence for nosocomial transmission of human immunodeficiency virus from a surgeon to one of his patients," *Journal of Virology*, vol. 72, no. 5, pp. 4537–4540, 1998.

[13] C. P. Goujon, V. M. Schneider, J. Grofti, J. Montigny, V. Jeantils, P. Astagneau, W. Rozenbaum, F. Lot, C. Frocrain-Herchkovitch, N. Delphin, *et al.*, "Phylogenetic analyses indicate an atypical nurse-to-patient transmission of human immunodeficiency virus type 1," *Journal of virology*, vol. 74, no. 6, pp. 2525–2532, 2000.

[14] E. O. Romero-Severson, I. Bulla, and T. Leitner, "Phylogenetically resolving epidemiologic linkage," *Proceedings of the National Academy of Sciences*, vol. 113, no. 10, pp. 2690–2695, 2016.

[15] M. Kaye, D. Chibo, and C. Birch, "Comparison of bayesian and maximum-likelihood phylogenetic approaches in two legal cases involving accusations of transmission of hiv," *AIDS research and human retroviruses*, vol. 25, no. 8, pp. 741–748, 2009.

[16] F. Lewis, G. J. Hughes, A. Rambaut, A. Pozniak, and A. J. L. Brown, "Episodic sexual transmission of hiv revealed by molecular phylodynamics," *PLoS Med*, vol. 5, no. 3, p. e50, 2008.

[17] E. M. Volz, E. Ionides, E. O. Romero-Severson, M.-G. Brandt, E. Mokotoff, and J. S. Koopman, "Hiv-1 transmission during early infection in men who have sex with men: a phylodynamic analysis," *PLoS Med*, vol. 10, no. 12, p. e1001568, 2013.

[18] "Covid rna sequence library." https://www.ncbi.nlm.nih.gov/sars-cov-2/. Accessed: 2021-05-27.

[19] K. Tamura and M. Nei, "Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees.," *Molecular biology and evolution*, vol. 10, no. 3, pp. 512–526, 1993.

[20] S. Salamat and T. Rosing, "Fpga acceleration of sequence alignment: A survey," *arXiv preprint arXiv:2002.02394*, 2020.

[21] H.-C. Ng, S. Liu, and W. Luk, "Reconfigurable acceleration of genetic sequence alignment: A survey of two decades of efforts," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–8, IEEE, 2017.

[22] S. Sarkar, T. Majumder, A. Kalyanaraman, and P. P. Pande, "Hardware accelerators for biocomputing: A survey," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3789–3792, IEEE, 2010.

[23] S. Salamat, J. Kang, Y. Kim, M. Imani, N. Moshiri, and T. Rosing, "Fpga acceleration of protein back-translation and alignment," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 822–827, IEEE, 2021.

[24] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. A. M. Elfadel, "Fpgaaas: A survey of infrastructures and systems," *IEEE Transactions on Services Computing*, 2020.

[25] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on fpga virtualization," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 131–1317, IEEE, 2018.

[26] S. Salamat, A. Haj Aboutalebi, B. Khaleghi, J. H. Lee, Y. S. Ki, and T. Rosing, "Nascent: Near-storage acceleration of database sort on smartssd," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 262–272, 2021.

[27] S. Salamat, B. Khaleghi, M. Imani, and T. Rosing, "Workload-aware opportunistic energy efficiency in multi-fpga platforms," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2019.

[28] A. Jahanshahi, M. K. Taram, and N. Eskandari, "Blokus duo game on fpga," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, pp. 149–152, IEEE, 2013.

[29] S. Salamat, S. Shubhi, B. Khaleghi, and T. Rosing, "Residue-net: Multiplication-free neural network by in-situ no-loss migration to residue number systems," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 222–228, IEEE, 2021.

[30] Z. Wang, H. Huang, J. Zhang, and G. Alonso, "Shuhai: Benchmarking high bandwidth memory on fpgas," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 111–119, IEEE, 2020.

[31] K. Kara, C. Hagleitner, D. Diamantopoulos, D. Syrivelis, and G. Alonso, "High bandwidth memory on fpgas: A data analytics perspective," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 1–8, IEEE, 2020.

[32] "Tn93 tool repository." https://github.com/veg/TN93. Accessed: 2021-05-27.

[33] V. Kathail, "Xilinx vitis unified software platform," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 173–174, 2020.

[34] "Alveo u280 data center accelerator card." https://www.xilinx.com/support/documentation/data_sheets/ds963-u280.pdf. Accessed: 2021-05-27.

[35] "Xilinx runtime library (xrt)." https://www.xilinx.com/products/design-tools/vitis/xrt.html. Accessed: 2021-05-27.

[36] "Los alamos national laboratory hiv library." https://www.hiv.lanl.gov/content/sequence/NEWALIGN/align.html#web. Accessed: 2021-05-27.

[37] "San diego county coivd-19 library." https://searchcovid.info/. Accessed: 2021-05-27.