ILLINOIS
CSL | Coordinated Science Lab
COLLEGE OF ENGINEERING

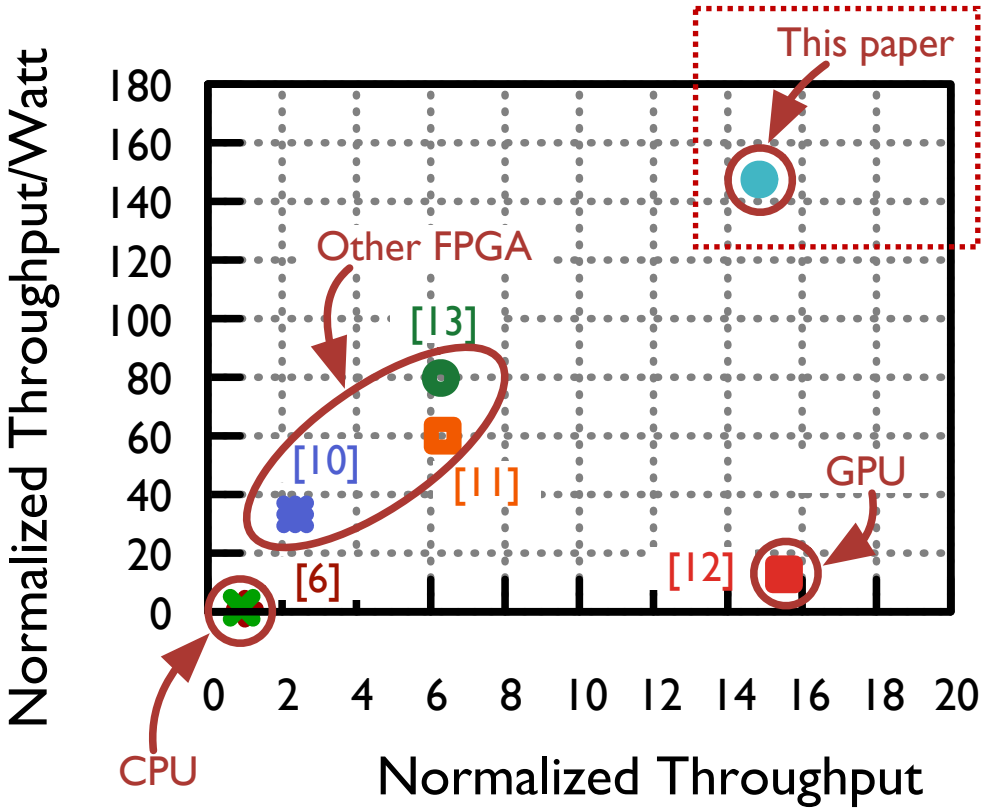
Subho S. Banerjee, Mohamed el-Hadedy, Ching Y. Tan,
Zbigniew T. Kalbarczyk, Steve Lumetta, Ravishankar K. Iyer

On Accelerating Pair-HMM Computations in Programmable Hardware

csl.illinois.edu

Contributions

- Design and implementation for an accelerator to compute the *Forward Algorithm* (FA) on *Pair-Hidden Markov Models* (PHMM) models.
- Demonstrate value of the accelerator supporting computational genomics workflows where PHMM is used to identify mutations in genomes
- Optimize accelerator architecture for both the algorithm and common input data characteristics
- Reduce compute time: **14.85× higher throughput**
- Reduce operational cost (in terms of energy consumption): **147.49× higher throughput per unit energy**



Citations are consistent with those in paper

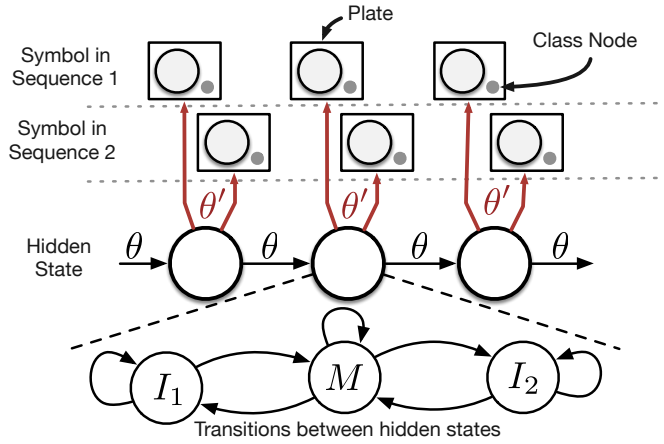
Forward Algorithm on Pair-HMM Models

- PHMM models are Bayesian multinets that allow for a probabilistic interpretation of the alignment problem
  - An alignment models the homology between two sequences via a series of mutations, insertions, and deletions of nucleotides.

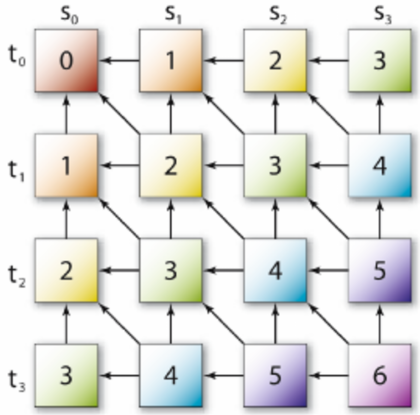
- FA algorithm computes of statistical similarity by considering all alignments between two sequences and computing the overall alignment probability by summing over them

- Can be described by the following equations

$$
\begin{aligned}
f_M(i,j) &= P(a_{mm} f_M(i-1,j-1) + \\
&\quad a_{im} f_I(i-1,j-1) + \\
&\quad a_{dm} f_D(i-1,j-1)) \\
f_I(i,j) &= a_{mi} f_M(i-1,j) + a_{ii} f_I(i-1,j) \\
f_D(i,j) &= a_{md} f_M(i,j-1) + a_{dd} f_D(i,j-1)
\end{aligned}
$$



Equations describe anti-diagonal data-dependecies



PHMM Forward Algorithm in Bioinformatics

- PHMMs form the basis of the variant detection tool `GATK HaplotypeCaller`
- Used to pick n-best haplotypes from by maximizing likelihood of a read originating from the haplotype
  - FA algorithm used
- Constitutes >70% of the runtime of the GATK HaplotypeCaller
- Executes >3E7 times for a standard clinical human dataset

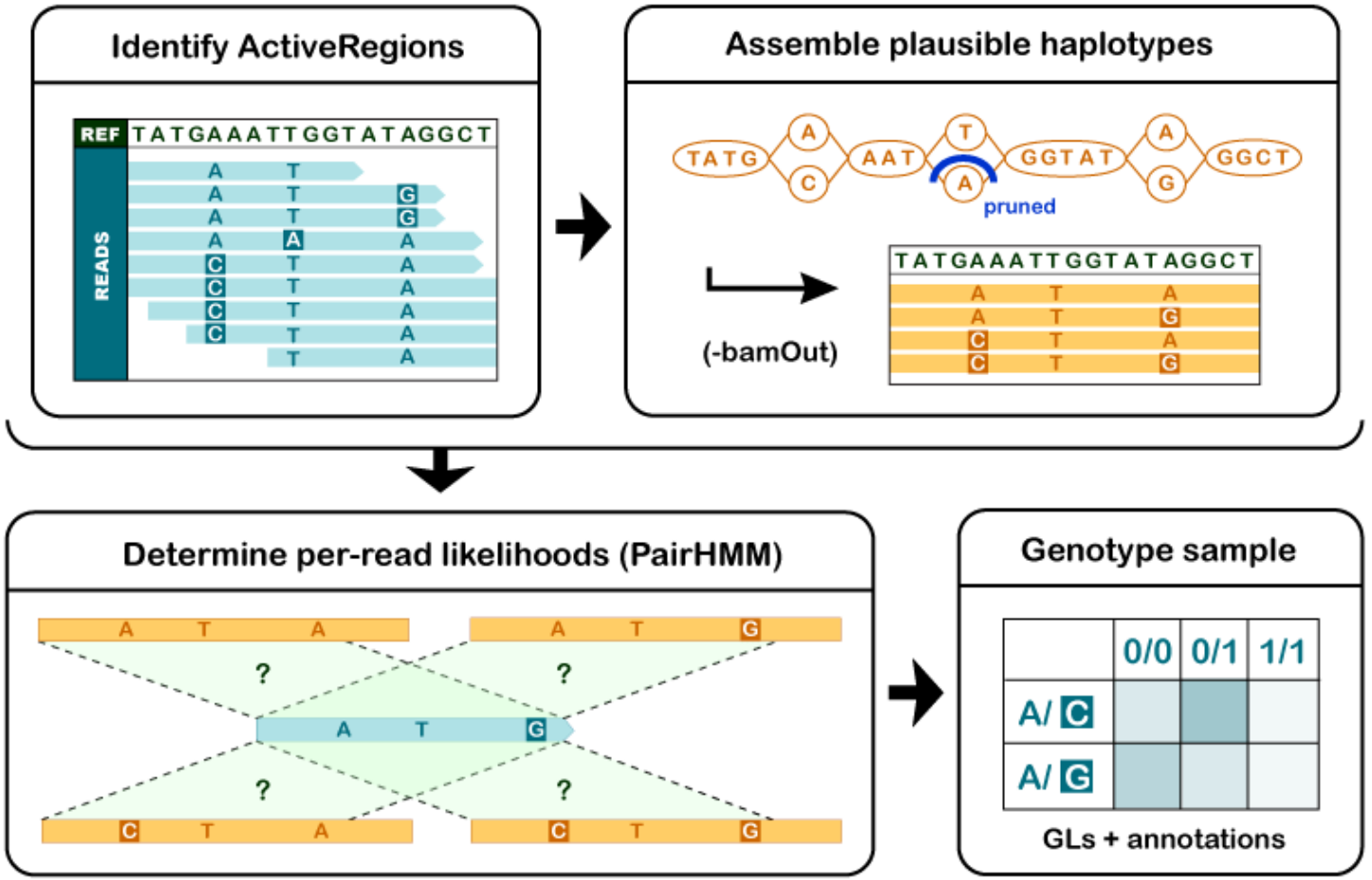
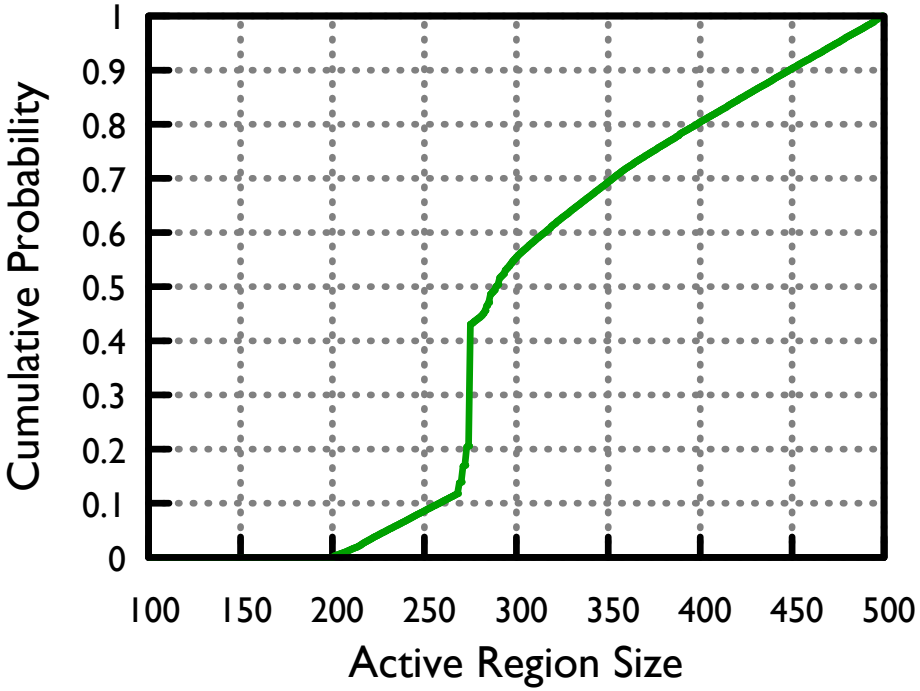


Diagram from GATK Documentation: https://software.broadinstitute.org/gatk/documentation/article.php?id=4148

Shortcomings of Related Work

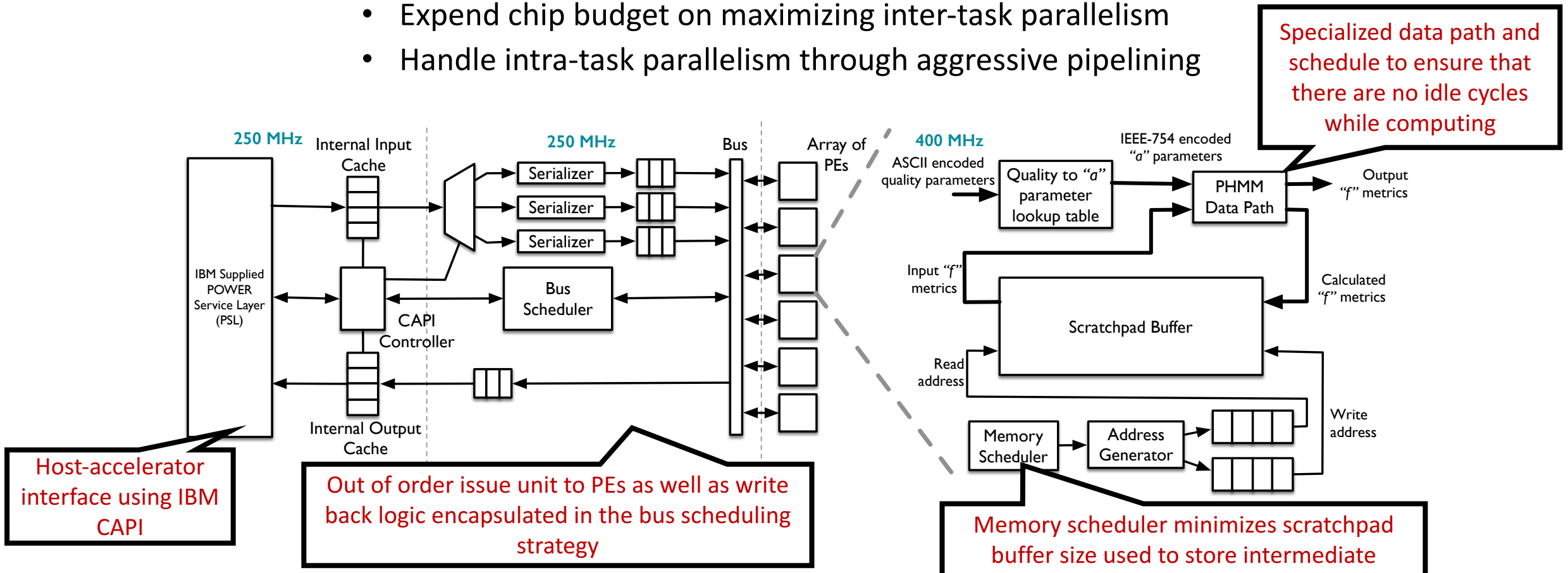
- Past work explores use of FPGAs/ASICs
  - Based on systolic array designs
  - Exploit anti-diagonal parallelism in recurrence pattern
- Common shortcoming is that they are optimized only for the algorithm and not input data characteristics
  - Input size variability can lead to idle cycles for systolic array based designs.



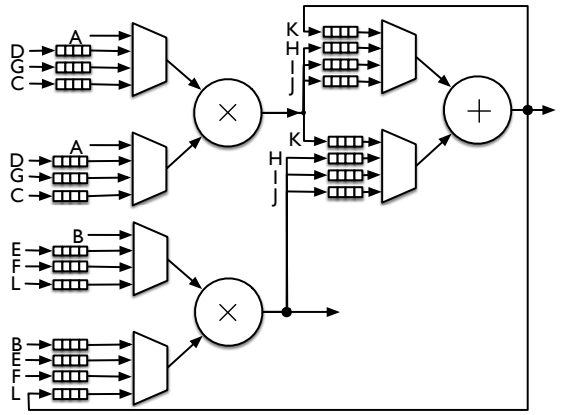
CDF shows nearly uniform distribution of input sizes for small (<250) and large (>350) input string size for computation on NA12878 sample

Our Design

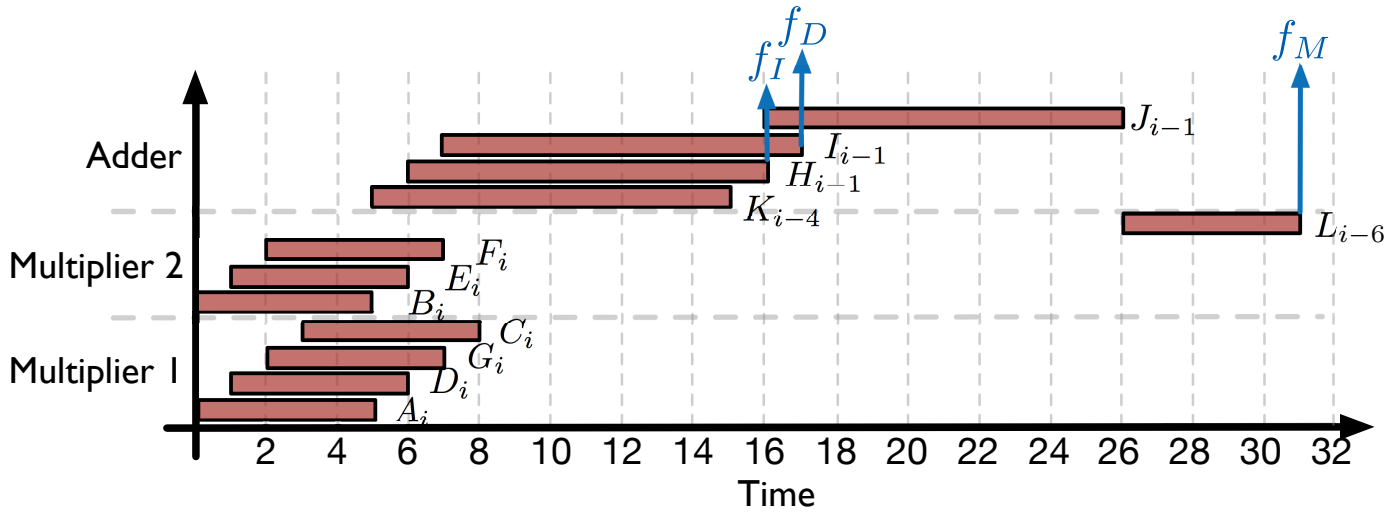
- **Design Goal:** Optimize design to execute different input sizes in parallel
  - Expend chip budget on maximizing inter-task parallelism
  - Handle intra-task parallelism through aggressive pipelining



Processing Element (PE) Design



Circuit representation of the computation datapath

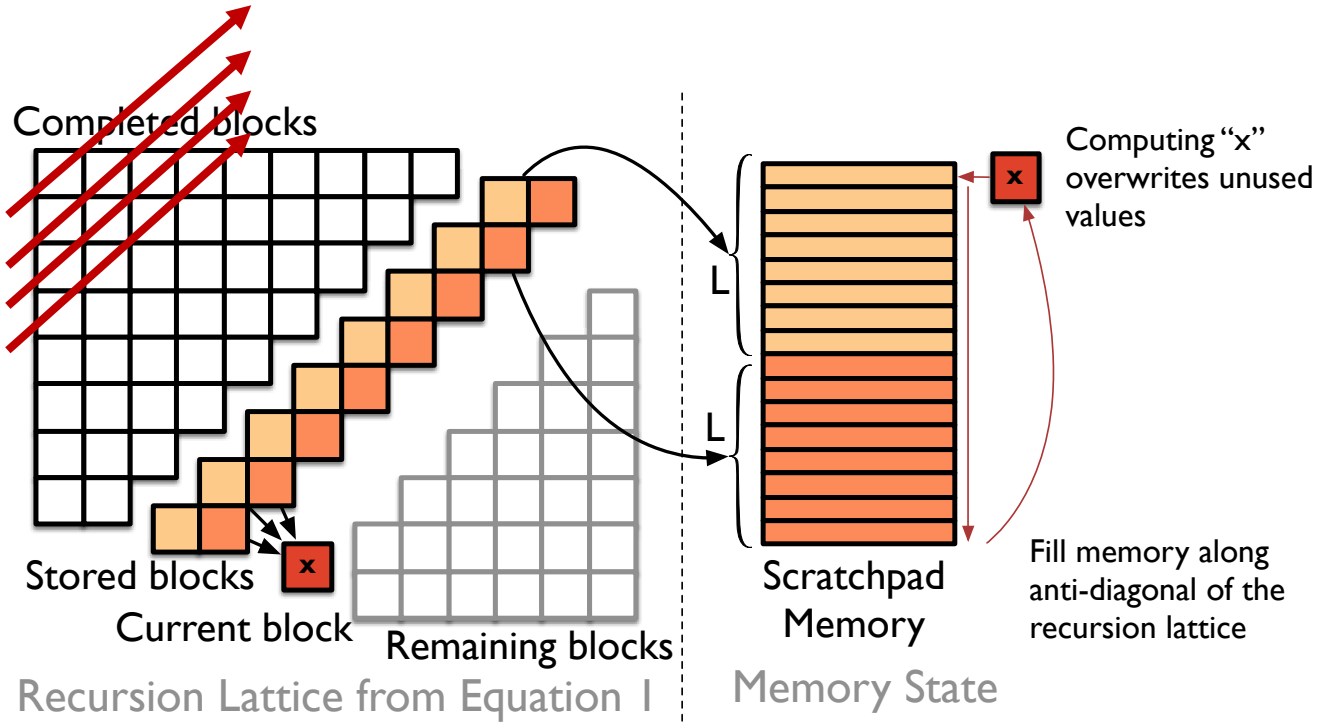


Gantt chart corresponding to schedule of operations

- **Goal:** Schedule operations to minimize idle cycles
  - Schedule presented above has no idle cycles
  - Schedule temporally multiplexes the adders and multipliers
  - Entire pipeline is 8-deep (8 Operations in flight at a time)

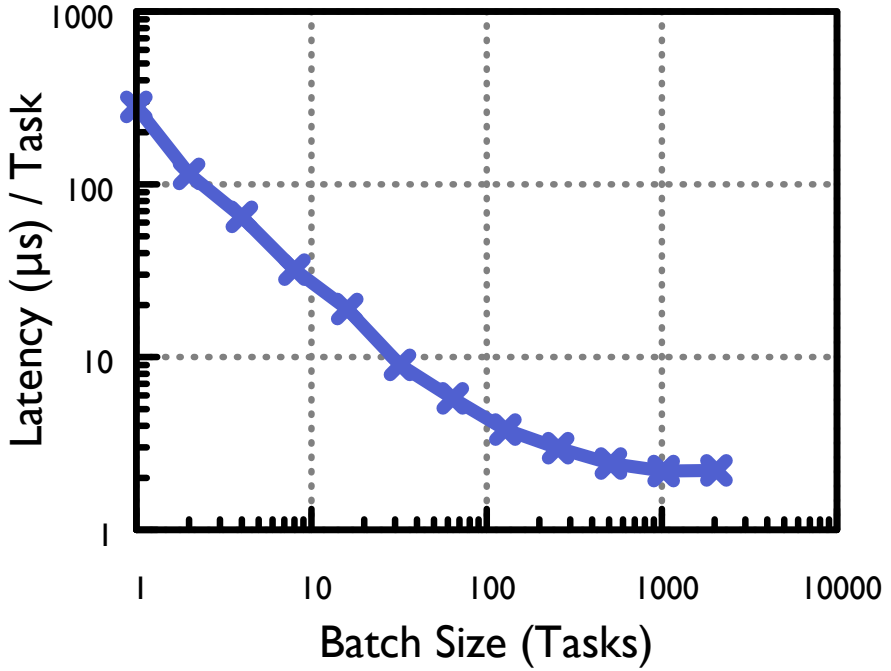
Minimize Storage Requirements

- Temporary scratchpad space is required to store intermediate outputs produced from the FA algorithm
- We minimize this space by following the anti-diagonal recursion pattern of the FA algorithm
- As a result, we need only *O(L)* space instead of $O(L^2)$ space to store entire matrix.



Dealing with Accelerator Invocation Overheads

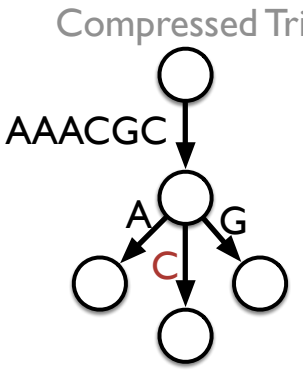
- Accelerator invocation overhead significantly reduces performance because of OS overhead of initializing accelerator
- Solution: Amortize cost of accelerator invocation by batching multiple invocations
  - OS sends batch of tasks to acc. Hardware dist across PEs
- Demonstrate several approaches to select task batches
  - Simple task batching
  - Common prefix memoization
  - FA on partially ordered strings

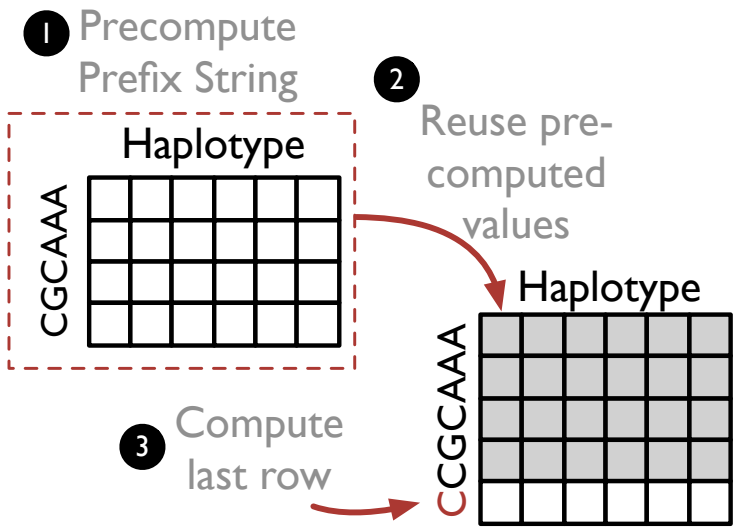


Task batching: Significant drop in mean latency of a PHMM task when OS overhead is amortized over large batches

Common Prefix Memoization

- Similar inputs to PHMM) have common prefixes
- Naïve algorithm recomputes PHMM for all pairs of strings
- Our solution:
  - Construct a prefix trie to find the longest common prefix in an input task batch
  - Compute PHMM FA for prefix only once
  - Saves compute time and host-accelerator bandwidth

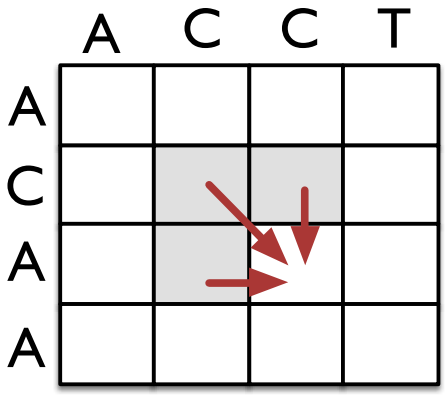




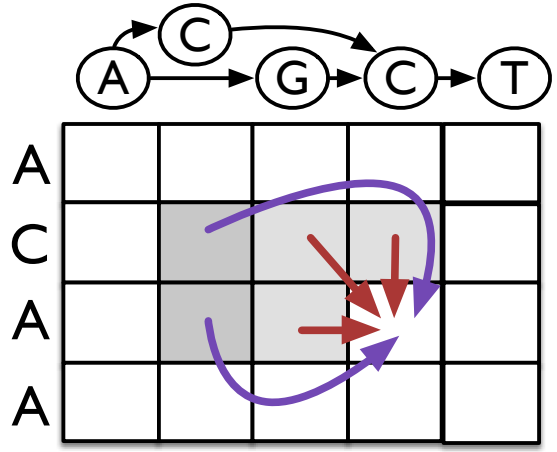
- Example
  - (AAACGCA, AAACCGG); (AAACGCC, AAACCGG); (AAACGCG, AAACCGG)
  - Read (Input 1) has common prefix for a single haplotype (Input 2)
  - Construct TRIE for Input 1
  - Precompute matrix for prefix on accelerator
  - Compute last row and column on host CPU

FA on Partially Ordered Strings

- Inputs to the PHMM accelerator in GATK is computed from DeBruijn graphs
- **Core Idea:**
  - Do not dispatch multiple paths from DeBruijn graphs as separate tasks
  - Dispatch entire graph at same time
- Present an extension of the POA algorithm [1] for computing FA between single read and entire DeBruijn graph



Traditional PHMM Dependency Lattice

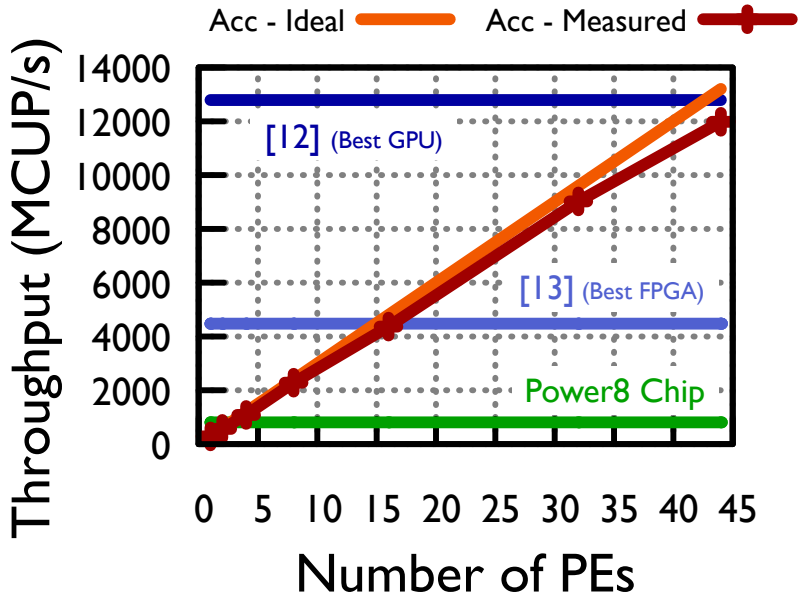


POA based PHMM Dependency Lattice

[1] C. Lee, C. Grasso, and M. F. Sharlow, "Multiple sequence alignment using partial order graphs," *Bioinformatics*, vol. 18, no. 3, pp. 452–464, Mar 2002.

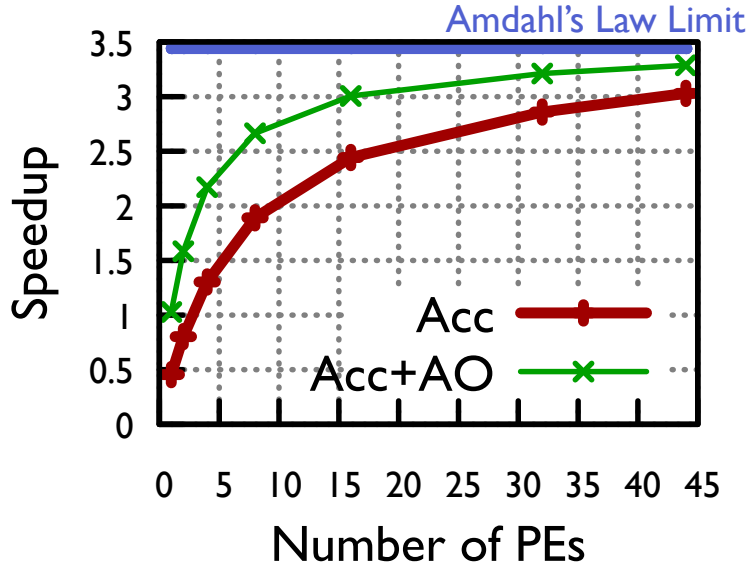
Results: Performance Benchmarking

Performance of the accelerator in a PHMM micro-benchmark



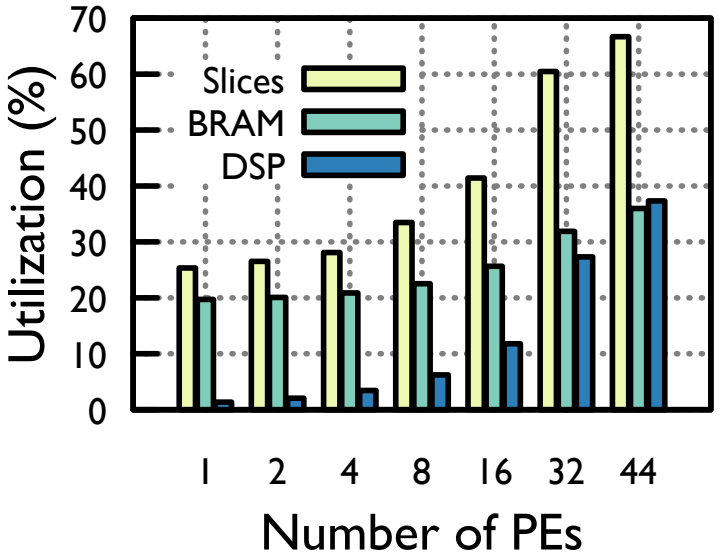
- 14.85× higher throughput than an 8-core CPU baseline (that uses SIMD and multi-threading)
- 147.49× improvement in throughput per unit of energy expended

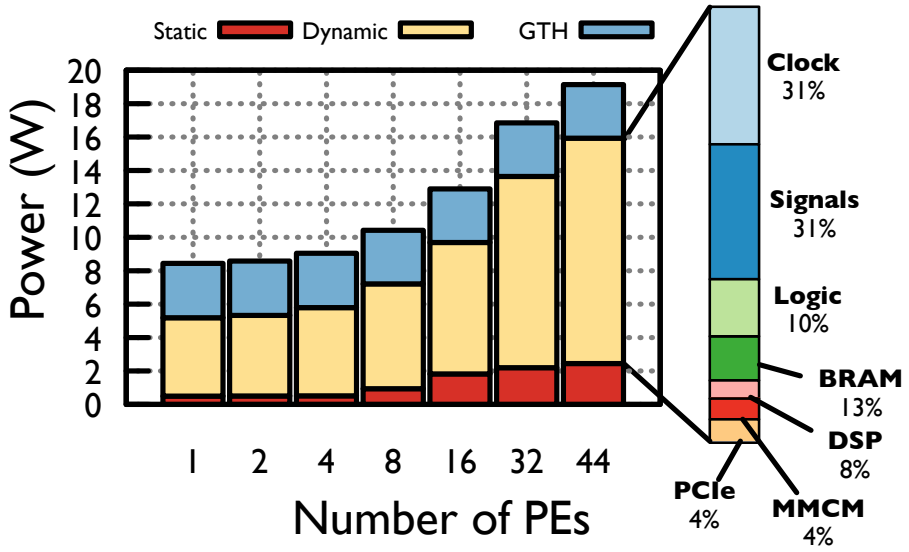
Performance of the end-to-end GATK HaplotypeCaller application



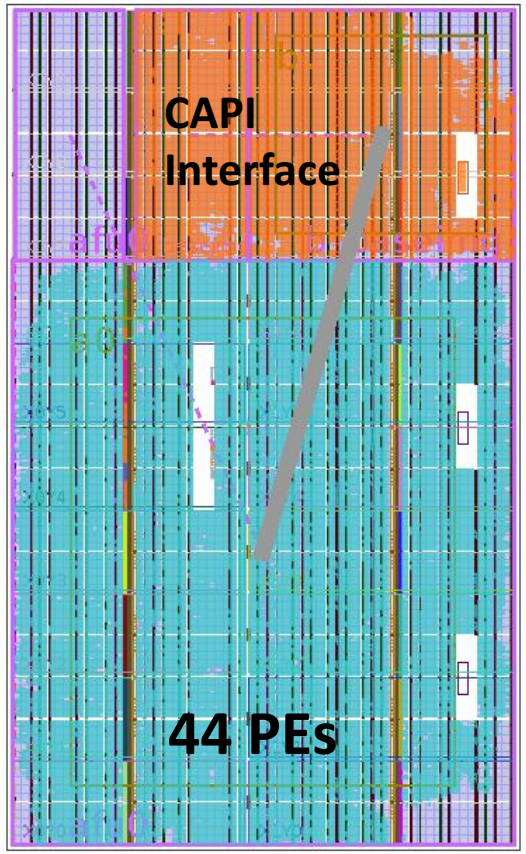
- 3.287× speedup over CPU-only baseline
- 3.48× is maximum attainable speedup accroding to Amdahl's Law

Results: On-Chip Resource Utilization





Physical Layout on a Xilinx XC7VX6905T



- The use of logic slices is the limiting factor
- Potential for larger gains in micro-benchmark performance for larger FPGAs
  - Memory bandwidth becomes a bottleneck [Simulation results in paper]
- Negligible gains to be had in terms of end-to-end application performance
  - Already close to Amdahl's law limit

Conclusions

- We demonstrate an FPGA based accelerator for the PHMM FA algorithm that achieves
  - 14.85× higher throughput than CPU baseline
  - 147.49× higher throughput per unit energy expended
- Immediate application in variant discovery and genotyping workloads
- **Takeaway:** Design methodology of using input data characteristics in addition to algorithmic characteristics to specialize accelerator design can be more generally

Questions?

- Code available at https://github.com/CSLDepend/PairHMM
- Email authors at ssbaner2@illinois.edu