# A High-Throughput Pruning-Based Pair-Hidden-Markov-Model Hardware Accelerator for Next-Generation DNA Sequencing

Xiao Wu, *Member, IEEE*, Arun Subramaniyan, *Member, IEEE*, Zhehong Wang, *Member, IEEE*, Satish Narayanasamy, *Fellow, IEEE*, Reetuparna Das, *Fellow, IEEE*, and David Blaauw, *Fellow, IEEE*

***Abstract*—We present the first ASIC accelerator for a pair-hidden-Markov-model (Pair-HMM) in DNA variant calling, which conventionally requires ∼250T FLOPs per sequenced human genome. Using a hardware-algorithm co-design, we opportunistically replace floating point (FP) multiplication with 20-b log-domain addition while employing bound checks to maintain (provable) correct results in downstream processing. FP computation is reduced by 43× on real human genome data. Implemented in a 40-nm CMOS, the 5.67 mm$^2$ accelerator demonstrates 17.3G cell updates per second (CUPS) throughput, marking a 6.6× improvement over our baseline ASIC implementation and 355× GCUPS/mm$^2$ improvement over an FPGA implementation [2].**

***Index Terms*—Algorithm-hardware co-design, floating point (FP) pruning, hardware accelerator, high throughput, pair-hidden-Markov-model (Pair-HMM), variant calling.**

## I. Introduction

Recent advances in next-generation sequencing have enabled fast DNA identification for cancer, genetic disorders, and pathogen detection. As shown in Fig. 1, short DNA fragments are sequenced in a massively parallel fashion, producing billions of DNA reads (strings of ∼100 nucleotides: A, C, G, T) per human genome. Reassembling these DNA fragments to determine differences compared with a common reference genome (referred to as secondary analysis) requires extensive computation. The short reads are aligned to a reference genome in *read alignment* to determine the location of each read's origin. The aligned reads are then processed in the second step: *variant calling*. During variant calling, plausible candidate mutation strings are constructed from aligned reads. Then the likelihood of each candidate mutation strings is evaluated against each read using Pair-HMM forward algorithm (or PFA). Finally, the likelihood of each mutation is marginalized, and the most likely mutation is picked as output. Details of variant calling will be introduced in Section I-A. Variant calling is responsible for identifying disease-related gene mutations and remains extremely time consuming. In particular, PFA for variant calling requires ∼250T FLOPs to infer mutation probabilities and contributes 52% of variant calling computation as shown in Fig. 2(a).

The PFA requires an alignment matrix calculation with a complicated combination of floating point (FP) addition and multiplication to infer the overall similarity of two strings, making it a challenging hardware optimization problem. The PFA has been mapped to a GPU [8] as well as FPGAs using systolic array [3]–[6] and ring-based topologies [2], [7]. However, these methods are constrained by the availability of FP resources. Also, they are direct hardware mappings that do not reoptimize the algorithm to make them hardware friendly. Furthermore, no dedicated ASIC has been demonstrated to date to accelerate the PFA for DNA sequencing.

Manuscript received September 7, 2020; revised November 22, 2020; accepted December 8, 2020. Date of publication December 16, 2020; date of current version February 4, 2021. This paper was approved by Associate Editor Vinayak Honkote. *(Corresponding author: Xiao Wu.)*

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan at Ann Arbor, Ann Arbor, MI 48109 USA (e-mail: lydiaxia@umich.edu).

Digital Object Identifier 10.1109/LSSC.2020.3045148

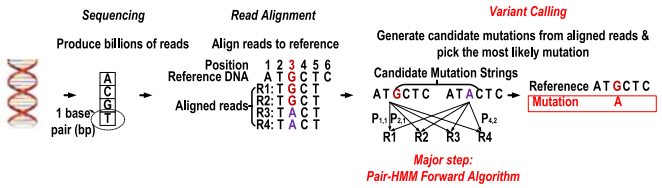


Fig. 1. Overview of secondary analysis in DNA sequencing.

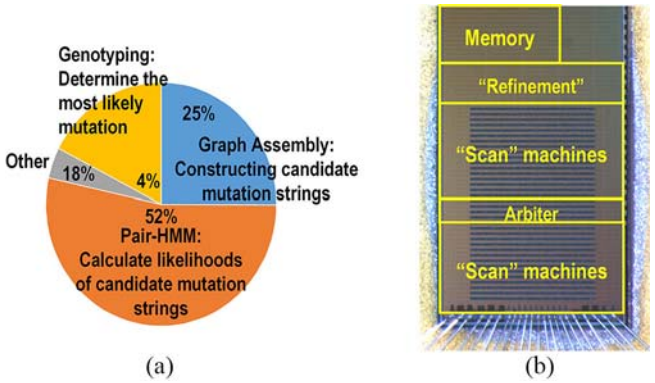


Fig. 2. (a) Runtime breakdown of variant calling. (b) Die photo.

We proposed the first ASIC implementation of the PFA (in prior paper [10]) and demonstrated its operation in a complete DNA sequencing flow. We made the key observation that the output of pair-hidden-Markov-model (Pair-HMM) matrix is mostly contributed by the probabilities of only a few high-quality alignment paths, despite the fact that the algorithm computes all possible alignments between read and candidate mutation string. Intuitively speaking, if one alignment path has only insertions and deletions, its probability score will likely to be orders of magnitude lower than an optimal alignment which mostly consists of matches. In a typical Pair-HMM matrix, most alignment paths are low-quality paths which consist of too many insertions, deletions, or mismatches. Using a new algorithm-hardware co-design inspired by this key observation, we alleviate the throughput bottleneck caused by limited FP resources by devoting FP resources only to estimate high-quality alignment regions (Fig. 4). For nonhigh-quality regions, FP multiplication is replaced with low-accuracy 20b integer *addition* by using log domain number representation, greatly improving the performance. By maintaining error bounds on this low-precision log-domain calculation, we detect and, as needed, rerun with high precision those few cases with possible failure, thereby, obtaining (provable) correctness in the downstream analysis. As a result, FP computation is reduced by 43× when tested on real human data and is replaced with 20 bit log-domain integer processing units (I-PEs) that are 4.6× smaller in area and 1.9× higher in performance than FP-PEs.

### A. Overview of Variant Calling

GATK's HaplotypeCaller is one of the most widely used variant calling tools today [9]. As shown in Fig. 1, the goal of variant

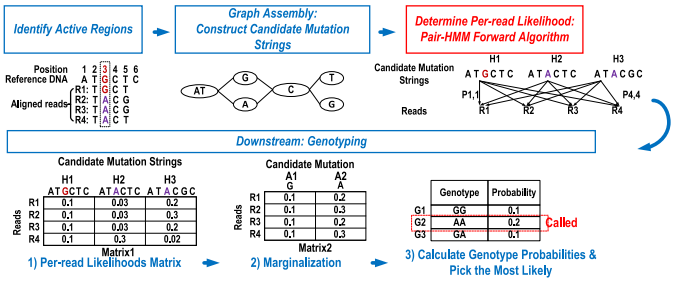


Fig. 3. Overview of variant calling illustrating Pair-HMM and its downstream processing steps.

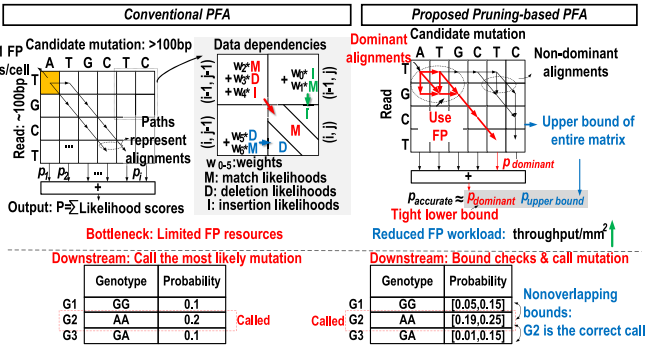


Fig. 4. Concept of the proposed pruning-based PFA.

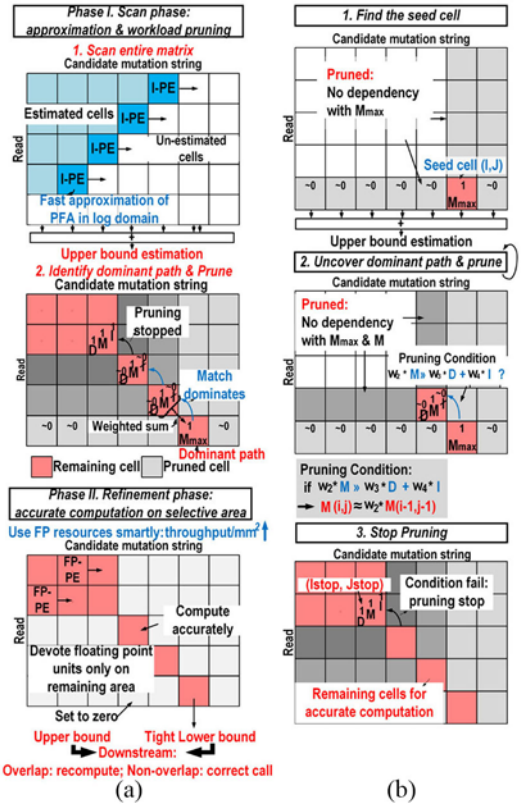


Fig. 5. (a) Two-phase operation of the proposed pruning-based PFA. (b) Detailed pruning methodology of the proposed pruning-based PFA.

calling is to identify mutations (i.e., base pairs in the sample that are different from the reference) using evidence supported by reads. The general flow of variant calling is illustrated in Fig. 3. The tool first identifies active regions (typically hundreds of base pairs) where reads are likely to be different from the reference genome. Second, each of the active regions is reassembled using a De-Bruijn graph to construct plausible candidate mutation strings, H1–H3. Each candidate mutation string is a string (∼hundreds of base pairs) containing identical substrings as well as different mutations. Third, the per-read likelihood of each candidate mutation string is calculated using a Pair-HMM forward algorithm, generating a matrix of likelihoods (Matrix1) for each read (R1–R4) and each candidate mutation string (H1–H3). Pair-HMM is extremely time-consuming and contributes 52% of variant calling computation according to our benchmark of HaploytpeCaller [9] v4.0.11 with chromosome 16, 17, and 18 of HG00419 from 1000 Genomes database (run on Xeon CPU E5 with single thread and AVX support for Pair-HMM acceleration).

In the downstream processing (steps after Pair-HMM), mutations A1, A2 (short substitutions, insertions, and deletions of a few base pairs) are extracted from mutations strings, transforming the string-based likelihoods matrix Matrix1 to mutation-based likelihood matrix Matrix2. During this marginalizing transformation, the highest likelihood of all strings containing a specific mutation is picked as this mutation's likelihood. For example, if both mutation strings H2 and H3 contain mutation A2, the higher per-read likelihood of H2 and H3 in Matrix1 is picked as the per-read likelihood of A2 in Matrix2. Genotypes G1–G3 are generated from mutations A1–A2 based on the ploidy of the sample. The probability of each genotype is derived from Matrix2 using a Bayesian model. Finally, the most likely genotype is called as the variant.

### B. Conventional Pair-HMM Forward Algorithm

A conventional PFA calculates the likelihoods of all alignments between a candidate mutation string and a DNA read using an alignment matrix (Fig. 4, left). Each cell $(i, j)$ indicates how base pair $i$ in the read is aligned to base pair $j$ in the mutation string using one of the three states—insertion, deletion, or match (containing both substitution and real match). Each path in the alignment matrix is a series of state transitions representing one alignment between the read and the mutation string. The PFA aims to infer the sum of probabilities of all alignments.

The likelihoods $M, I$, and $D$ of a particular cell $(i, j)$ is computed from the likelihood of its three neighboring cells: vertical neighbor cell $(i-1, j)$ representing an insert transition, diagonal neighbor $(i-1, j-1)$ for a match transition, and horizontal neighbor $(i, j-1)$ for a delete transition (Fig. 4, left). The computation in each cell involves several additions and multiplications, and single-precision FP is typically required to avoid underflow. The detailed calculation of the PFA is outlined in (1)–(4). The coefficients w0-6 are either fixed parameters or related to base pair quality scores. Nr and Nh are the lengths of the read and mutation strings, respectively

$$M(i,j) = M(i-1,j-1) * W_2 + D(i-1,j-1) * W_3 + I(i-1,j-1) * W_4 \quad (1)$$

$$I(i,j) = I(i-1,j) * W_0 + M(i-1,j) * W_1 \quad (2)$$

$$D(i,j) = D(i,j-1) * W_5 + M(i,j-1) * W_6 \quad (3)$$

$$\text{Result} = M(N_r, N_n) + I(N_r, N_n). \quad (4)$$

## II. Pruning-Based Algorithm-Hardware Co-Design

### A. Proposed Pruning-Based Pair-HMM

We make the key observation that the final score of the matrix is typically dominated by the probabilities of only a few alignment paths, thanks to high quality reads and a small likelihood of genetic mutations. To identify these dominant paths, the proposed pruning-based Pair-HMM algorithm executes in two phases [Fig. 5(a)].

In the *scan* phase, an upper bound likelihood for each cell in the alignment matrix is computed using I-PEs operating in logarithmic

number representation, which replaces multiplication with addition and significantly reduces the hardware complexity. We observed that most dominant paths contain long consecutive diagonal transitions. Therefore, starting from the cell with the maximum matched score in the final row, the algorithm traces back along diagonal cells, pruning horizontally and vertically adjacent cells (i.e., treating their likelihood as zero). It continues this trace as long as the pruned cells, representing delete/insert transitions, do not contribute significantly to the cell score. The result of the first phase is a pruned region as well as an upper bound of the final score of the entire matrix.

In the following *refinement* phase, the (small) un-pruned region is computed using FP-PEs (the likelihood of the pruned region is set to zero), resulting in a (tight) lower bound.

### B. Handle Accuracy Degradation With Error Bound Check

The results from the proposed scan phase and refinement phase suffer from accuracy degradation due to the use of fixed point and pruning. To guarantee a (provable) correct final output after downstream processing, we propose to use the upper bound (result of scan phase) and the lower bound (result of refinement phase) estimation of the exact Pair-HMM result in the downstream analysis. Unlike the conventional method where the output of Pair-HMM is one exact value computed on the entire alignment matrix using FP, the proposed pruning-based Pair-HMM outputs a lower bound and an upper bound of the exact result. Since the main goal of downstream processing is to compare the probabilities of candidate mutations and pick the most likely mutation (Fig. 3, bottom), we substitute this comparison with error bound comparison. As illustrated in Fig. 4, if the lower bound of the selected genotype G2 is higher than the upper bound of all the unselected genotypes, it is guaranteed that the selected genotype has the highest probability. Otherwise, the bounds overlap, and we cannot infer a guaranteed result based on the current error bound estimation of genotypes. In this case, recomputation of the original Pair-HMM matrix is performed. These failing cases (1.5%) are guaranteed to be identified and are recomputed using only FP-PEs, according to our benchmark using chromosome 1 of sample HG00419 from the 1000 Genomes database. Therefore, through a bound comparison in a downstream analysis and selected recomputation, the mutation with the highest likelihood among all candidate mutations is correctly determined as the final variant calling result in all cases.

As briefly described in thesis [11], the upper bound result comes from the scan phase and is an estimation of the entire alignment matrix using fixed points in the log domain. The upper bound is generated by rounding up in each approximation. The lower bound result comes from the refinement phase and is an accurate computation of the unpruned section of the alignment matrix using FP. Since only a subset of the alignments are calculated, the result is naturally a lower bound of the exact result.

Recomputation is done iteratively on a read-by-read basis until the bounds do not overlap so that unnecessary recomputation can be minimized. Once a read-haplotype pair is selected for recomputation, the conventional Pair-HMM method with only FP operations is used to obtain the exact result. This methodology can be used in downstream processing steps as long as all operations involved preserve the error bounds.

### C. Details of Cell-Level FP Pruning

Fig. 5(b) shows the pruning method in the scan phase in more detail. The cell $(I, J)$ with the highest match score in the final row indicates the end position of a good alignment and is picked as the seed position for pruning. In the PFA, the match score at the seed $M(I, J)$ is the weighted sum of $M, I, D$ from the diagonally adjacent cell $(I-1, J-1)$. If this match score is significantly larger than the insertion and deletion scores in cell $(I-1, J-1)$, we would arrive at a slightly lower $M(I, J)$ score with insertion and deletion scores set to zero in cell $(I-1, J-1)$. This allows us to prune insertion and deletion scores in $(I-1, J-1)$ and all their adjacent cells to the top and left. The pruning stops at the cell ($I$stop, $J$stop) where the match no longer shows dominance over the insertion and deletion scores. For FP-PE computation, this results in a (small) rectangular region in the top-left of the matrix, followed by a string of consecutive match transitions [Fig. 5(b), shown in red].

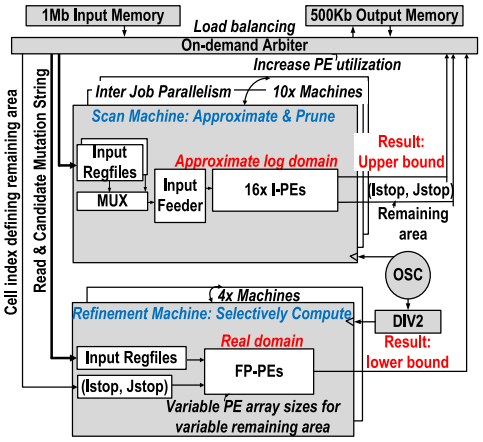


Fig. 6. Top-level architecture of pruning-based PFA accelerator.

### D. Matrix-Level Floating Point Pruning

During marginalization in downstream processing, mutation strings containing the same mutation compete, and the string with the highest likelihood gets selected. Low per-read likelihoods are likely to lose in marginalization and never get used in the final step. Inspired by this implicit filtering, we propose matrix-level FP pruning. If the upper bound result from a scan phase is too small, the whole refinement phase with FP calculation is skipped entirely and the lower bound of this skipped PFA is set to 0. The threshold for matrix-level pruning is set empirically.

### E. Early Stop Detection in Fixed Point Computation

As we are able to substitute the majority (97.7%) of the FP workload with a fixed point approximation, the throughput improvement over conventional PFA becomes bottlenecked by the area and performance gain of fixed point PEs over FP PEs. To further improve the throughput, we aim to reduce the fixed point workload (i.e., the scan phase) by *early stop*. Similar to matrix-level pruning for FP, a fixed point calculation can be terminated early if the final PFA output is predicted to be small. The values in row $i$ of the PFA matrix represent all of the alignments between the entire mutation string and a prefix of the read up to base pair $j$. As integer PEs propagate through the alignment matrix row by row, the likelihoods decrease with each added read base pair in the alignment. Therefore, the maximum value of all of the cells in the current row (Pmax,j) is a loose upper bound estimation of the final result. If Pmax,j is smaller than a threshold fL, the upper bound calculation is terminated and the scan phase is stopped early. The loose upper bound Pmax,j and loose lower bound 0 are passed to downstream processing.

## III. Hardware Implementation

Fig. 6 shows the overall architecture of the proposed pruning-based Pair-HMM accelerator. It consists of 10 scan machines composed of 16 I-PEs each to upper bound and prune matrices and 4 refinement machines composed of FP-PEs to accurately compute the unpruned

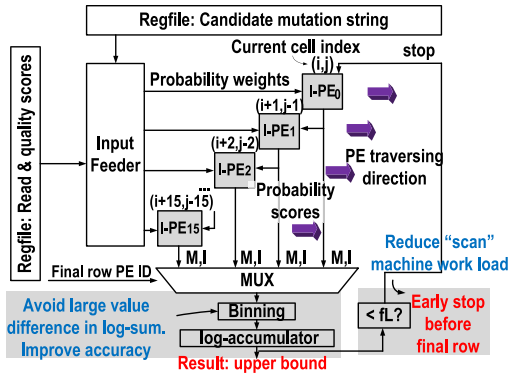


Fig. 7. Architecture of "scan" machines.

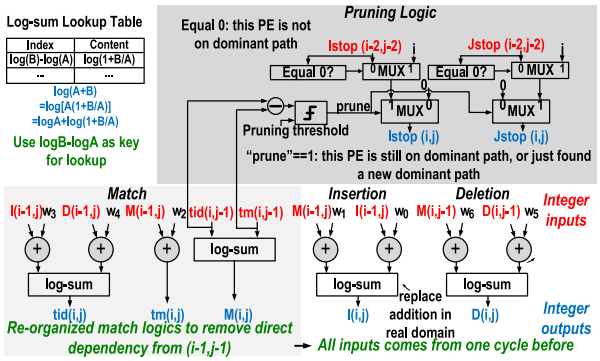


Fig. 8. Circuit implementation of log-domain I-PE.

regions. The refinement machines come in two sizes with 1× and 4× FP-PEs to accommodate the variable size of un-pruned regions. An on-demand arbiter streams in jobs from the input memory, dispatches them to scan and refinement PEs, and streams the results to the output memory. The number of FP-PEs on the prototype accelerator is determined by the pruning ratio, i.e., the average percentage of FP calculation that can be substitute with fixed point calculation. This average pruning ratio (97.7%) gives us an estimated ratio of FP workload and fixed point workload. Due to temporal variation of pruning ratio, FP-PEs can be temporarily underutilized or overutilized, leading to a degraded throughput as reflected in our actual measurement. A tradeoff between cache size (to minimize impact of temporal workload imbalance) and overall throughput can be studied in future works.

Fig. 7 shows the hardware implementation of a scan machine, consisting of 16 PEs (Fig. 8), an input feeder to control PE traversal across the matrix, a binning-based log-sum module to avoid accuracy degradation in the last row, and an early stop detection module. Each PE uses 20 bits fixed point addition and a 15-entry table lookup in the log domain as substitutes for multiplication and addition, respectively, in the real domain. Instead of tracing back to determine the pruned region, the logic in the PEs prune cells as PEs traverse forward across the matrix, avoiding the need to store scores for the entire matrix. The PEs work in parallel when traversing the matrix from left to right. As the PEs traverse, an early detection module opportunistically stops the scan phase once the maximum score in one row is smaller than an established threshold. This optimization takes advantage of downstream processing where extremely low Pair-HMM results are filtered completely, reducing the workload even in the scan phase (by 18%). Because only adjacent PEs communicate with each other, the routing complexity is greatly reduced.

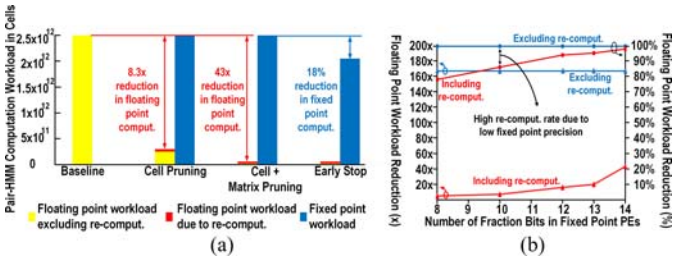


Fig. 9. (a) Pair-HMM workload reduction of the proposed techniques. (b) Trade-off between scan machine precision and overall floating point workload reduction.

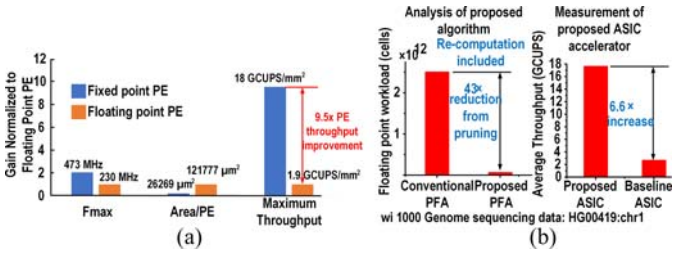


Fig. 10. (a) Performance comparison of FP PE versus fixed point PE. (b) Analysis of the proposed algorithm and measurement result of the proposed ASIC.

TABLE I
PERFORMANCE COMPARISON

| | [2] | [4] | [8] | [8] | This work |
|---|---|---|---|---|---|
| Platform | FPGA Arria 10 | FPGA Virtex 7 | CPU POWER8 (20 cores) | GPU Nvidia Tesla K40 | ASIC |
| Technology | 20nm | 28nm | 22nm | 28nm | 40nm |
| Chip area (mm²) | 858* | 858* | 1298 | 561 | 5.67 |
| Topology | PE ring | Systolic array | N.A. | N.A. | Pruning-based |
| Frequency (MHz) | 230.73 | 166.7 | 3420 | 745 | 120 |
| Power (W) | N.A. | N.A. | N.A. | N.A. | 0.756 |
| Peak floating point performance (GFLOPS) | 1366 | 2000 | 1094.4 | 4290 | 6.9 |
| Throughput normalized to floating point perfoamance (CUPS/FLOPS) | 0.0216 | 0.0113 | 0.0024 | 0.0006 | 2.5072 |
| Throughput normalized to area (GCUPS/mm²) | 0.0086** | 0.0129** | 0.0006** | 0.0023** | 3.0511 |
| Power efficiency (GCUPS/W) | N.A. | N.A. | N.A. | N.A. | 22.8836 |

* Only reticle size was available
** Normalized to technology node

## IV. MEASUREMENTS AND PERFORMANCE ANALYSIS

Fig. 9(a) summarizes the main techniques employed in the proposed pruning-based Pair-HMM and the respective computation reduction. This benchmark is carried out using chromosome 1 of sample HG00419 from the 1000 Genomes database. Compared to the baseline algorithm, cell pruning alone achieves an 8.3× FP workload reduction. By implementing matrix pruning, we achieve 43× FP reduction. By using the early stop technique in the scan phase, a 18% fixed point calculation can further be saved, leading to an extra 1.22× increase in throughput during scan phase.

Fig. 9(b) shows the tradeoff between fixed point precision and total FP workload reduction. As fixed point precision increases, the recomputation rate is reduced thanks to a tighter upper bound estimation. As a result, the total FP workload reduction (including recomputation) improves at the cost of higher fixed point PE complexity.

Fig. 10(a) shows the performance and area gain (from Automatic Place and Route) of fixed point PE over FP PE. Fixed point PEs are 4.6× smaller in area and 1.9× higher in performance than FP PEs, leading to a 9.5× improvement at maximum if all FP workload is replaced with fixed point.

Fabricated in 40-nm CMOS with 5.67 $mm^2$ die area, the accelerator reaches 120 MHz with 756 mW. Fig. 10(b) shows that the number of cells requiring FP calculation is reduced 43× (including recomputation due to bound check failure). The proposed accelerator was

verified with real sequencing data: chr22:16040000-16540000 of sample HG00419 from the 1000 Genomes database. It shows 17.3 GCUPS average throughput (including recomputation) which is a 6.6× improvement over an FP-only baseline ASIC implementation (normalized to the same area). We also compared the throughput of the proposed work with prior works. The common way to benchmark the Pair-HMM is to use a synthetic data "10s" [12]. However, this test dataset was designed specifically to the conventional method where Pair-HMM is a self-contained step that outputs a single result. Our proposed method outputs an upper bound and a lower bound that needs to be checked in downstream processing for error bound overlapping and, therefore, cannot be evaluated with the standard "10s" dataset. In addition, the overall throughput of the proposed method is affected by recomputation rate, which depends on the actual quality of the input reads produced by sequencing machine. Therefore, it is important to use a real sequencing dataset so that the recomputation rate can be correctly evaluated and included in throughput comparison.

To establish a conservative throughput comparison between the proposed work and prior works, we quote the maximum theoretical throughput (assuming 100% PE utilization at all time) for prior works while using the actual measured throughput for our work using sample HG00419. The maximum theoretical throughput is the product of peak frequency and total number of PEs. Therefore, the throughput numbers for prior works (Table I) are an optimistic estimation of their actual throughput.

We obtain speedups of 355× and 1344× in CUPS/mm$^2$ compared to FPGA [2] and NVidia K40 GPU [8] implementations, respectively. Fig. 2(b) shows the die photograph.

## V. Conclusion

In summary, we proposed a pruning-based Pair-HMM forward algorithm and, to our knowledge, its first ASIC accelerator to speed up secondary analysis in whole-genome sequencing. This algorithm-hardware co-design reduces FP calculations by 43×. As summarized in Table I, the implemented ASIC accelerator achieves a speedup of 6.6× over an FP only, baseline ASIC implementation. It achieves 355× and 1344× increase in CUPS/mm$^2$ compared to FPGA [2] and GPU [8] implementations, respectively.

## References


[1] M. Carneiro, "Acceleration variant calling," in *Proc. Intel Genomic Sequencing Pipeline Workshop*, 2013.

[2] S. Huang *et al.*, "Hardware acceleration of the pair-HMM algorithm for DNA variant calling," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, Feb. 2017, pp. 275–284.

[3] C. Rauer *et al.*, "Accelerating genomics research with OpenCL and FPGAs," Santa Clara, CA, USA, Intel, White Paper, 2017.

[4] J. Peltenburg *et al.*, "Maximizing systolic array efficiency to accelerate the PairHMM forward algorithm," in *Proc. IEEE Int. Conf. Bioinformat. Biomed.*, Dec. 2016, pp. 758–762.

[5] S. Ren *et al.*, "FPGA acceleration of the pair-HMMs forward algorithm for DNA sequence analysis," in *Proc. IEEE Int. Conf. Bioinformat. Biomed.*, Nov. 2015, pp. 1465–1470.

[6] M. Ito and M. Ohara, "A power-efficient FPGA accelerator: Systolic array with cache-coherent interface for pair-HMM algorithm," in *Proc. IEEE Symp. Low Power High Speed Chips*, Apr. 2011, pp. 1–3.

[7] G. J. Manikandan *et al.*, "Acceleration of the pair-HMM algorithm for DNA variant calling," in *Proc. IEEE 24th Annu. Int. Symp. Field Program. Custom Comput. Mach.*, May 2016, p. 137.

[8] S. Ren *et al.*, "Efficient acceleration of the pair-HMMs forward algorithm for GATK haplotypecaller on graphics processing units," *Evol. Bioinform.*, vol. 14, Mar. 2018.

[9] GATK Team, *HaplotypeCaller in a Nutshell*, Broad Inst., Cambridge, MA, USA, 2020. Accessed: Aug. 2020. [Online]. Available: https://gatk.broadinstitute.org/hc/en-us/articles/360035531412-HaplotypeCaller-in-a-nutshell

[10] X. Wu *et al.*, "17.3 GCUPS pruning-based pair-hidden-Markov-model accelerator for next-generation DNA sequencing," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2020, pp. 1–2.

[11] X. Wu, "Energy efficient circuits and system for Internet of Things and hardware accelerator design for genome sequencing," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. Michigan, Ann Arbor, MI, USA, 2019.

[12] *Pair-HMM Test Data*. Accessed: Oct. 31, 2020. [Online]. Available: https://github.com/MauricioCarneiro/PairHMM/tree/master/test data