Darwin-WGA: A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup

Yatish Turakhia*
Stanford University
yatisht@stanford.edu

Sneha D. Goenka*
Stanford University
gsneha@stanford.edu

Gill Bejerano
Stanford University
bejerano@stanford.edu

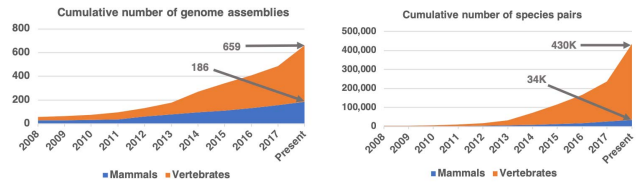
William J. Dally
Stanford University
NVIDIA Research
dally@stanford.edu

***Abstract*—Whole genome alignment (WGA) is an indispensable tool in comparative genomics to study how different life-forms have been shaped by evolution at the molecular level. Existing software whole genome aligners require several CPU weeks to compare a pair of mammalian genomes and still miss several biologically-meaningful, high-scoring alignment regions. These aligners are based on the *seed-filter-and-extend* paradigm with an ungapped filtering stage. Ungapped filtering is responsible for the low sensitivity of these aligners but is used because it is 200× faster than performing gapped alignment, using dynamic programming, in software. In this paper, we show that both performance and sensitivity can be greatly improved by using a hardware accelerator for WGA. Using the genomes of two roundworms (*C. elegans* and *C. Briggsae*) and four fruit flies (*D. melanogaster*, *D. simulans*, *D. yakuba*, and *D. pseudoobscura*), we show that replacing ungapped filtering with gapped filtering increases the number of matching base-pairs in alignments by up to 3×. Our accelerator, Darwin-WGA, is the first hardware accelerator for whole genome alignment and accelerates the gapped filtering stage. Darwin-WGA also employs GACT-X, a novel algorithm used in the extension stage to align arbitrarily long genome sequences using a small on-chip memory, that provides better quality alignments at 2× improvement in memory and speed over the previously published GACT algorithm. Implemented on an FPGA, Darwin-WGA provides up to 24× improvement (performance/$) in WGA over iso-sensitive software. An ASIC implementation of the proposed architecture on TSMC 40nm technology takes around 43W power with 36mm^2 area. It achieves up to 10× performance/watt improvement on whole genome alignments over state-of-the-art software at higher sensitivity, and up to 1,500× performance/watt improvement compared to iso-sensitive software. Darwin-WGA is released under open-source MIT license and is available from https://github.com/gsneha26/Darwin-WGA.**

***Keywords*-Co-processor, Comparative Genomics, Whole Genome Alignment, Gapped Filtering**

I. Introduction

Whole genome alignment (WGA) provides an indispensable tool for comparative genomics that facilitates *genome-wide* (i) identification and prediction of functional elements, such as genes and regulatory sequences [2], (ii) phylogenomics [3], i.e., to infer the evolutionary relationship between species using genomic sequences, and (iii) reconstruction of ancestral genome sequences [4]. Genome databases are rapidly growing (Figure 1a) — sequencing of several thousands of new species is currently underway [5], [6], [7], and more polished assemblies are being produced for existing genomes [8], thus paving an opportunity for millions of pairwise WGA to advance genomic science. Such pairwise alignments of whole genome sequences are costly [9], [10] — a single pairwise WGA of two mammals (human and mouse) takes 36 hours (costing around $57[1]) utilizing all 18 cores (with 36 threads) of a large Amazon EC2 instance (c4.8xlarge). Since the computational load of pairwise WGA grows quadratically with the number of genomes (Figure 1b), a recent study has estimated that computational costs of pairwise WGA would soon be much higher than the sequencing and assembly costs of individual genomes [11]. Unfortunately, the high computational costs of WGAs currently hampers science — of the 56 mammalian (a clade most relevant to human) genomes that we sampled on the UCSC Genome Browser database [12] (one of the largest public databases supporting genome research worldwide), only 264 (8.6%) of the 3080 possible pairwise WGAs were available. Likewise, "reference-free" approaches to multiple sequence alignments (MSAs) [13] and ancestral genome reconstruction [4] rely on a quadratic (to species count) number of WGAs between species pairs but skip most pairs because of the prohibitively high computational cost of WGA.



(a) (b)

Figure 1: Cumulative number of (a) genome assemblies (1 assembly per species) and (b) species pairs (for WGA) available in NCBI genome database [1] by year.

Besides the computational cost, another factor of paramount importance in comparative genomics is the *sensitivity* of the whole genome aligners, i.e., the ability of the underlying algorithm to detect high-scoring align-

*These authors contributed equally to this work.

[1]This paper assumes dollar ($) in the United States currency.

2378-203X/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCA.2019.00050

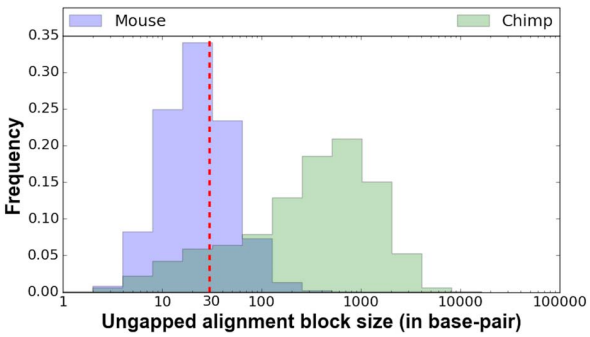


Figure 2: Distribution of ungapped alignment block size from the top-10 highest-scoring chains derived from LASTZ whole genome alignments of mouse and chimp with human. X-axis uses a logarithmic scale. LASTZ requires at least 30bp matches by ungapped extension.

ments [14], [15]. Numerous studies have shown that existing whole genome aligners do not capture several biologically-meaningful, high-scoring alignment regions and increasing sensitivity comes at a high computational cost. For instance, a study [16] found around 20,000 new alignments between human and mouse WGA that were missing from the UCSC Genome Browser by lowering the threshold on the *ungapped extension* stage of LASTZ [17] — a popular whole genome aligner. Another study [18] used a similar strategy and found thousands of new alignments in the *exons* (contiguous segment of gene containing protein-coding information) of orthologous genes (and therefore, of high functional relevance) in vertebrates. Both studies noted that carrying out such sensitive alignments leads to a considerable slowdown in software (up to 100×).

Figure 2 provides a biological insight into the observations of previous studies. It shows the distribution of ungapped alignment blocks from the top-10 highest-scoring *chains* obtained from the LASTZ WGA (see Section II for background on LASTZ and chains) of the human genome with that of chimp and mouse, respectively, before either an *insertion* or *deletion* (indel) is found. The figure highlights that for chimp (more closely related to human), indels are relatively rare - occurring roughly every 641 base-pairs (bp) on average, but for mouse (more distantly related to human), indels occur more frequently, approximately every 31bp on average. In its default setting, LASTZ requires a score equivalent of at least 30 matches (highlighted in figure 2 using a vertical red line) in the ungapped extension stage. All of the alignments to the left of the red line will not be found by ungapped alignment. For distantly-related species, like mouse, this is a substantial fraction of all matching sequences.

This motivates Darwin-WGA — the first hardware-acceleration framework for WGA that not only improves the sensitivity of WGA but also provides a high speedup. This paper makes the following contributions:

1) We introduce a lightweight, and hardware-accelerated implementation of Banded Smith-Waterman (BSW) algorithm [19] that provides 27× performance/$ improvement on FPGA over the software implementation. We show that replacing ungapped filtering with BSW algorithm significantly improves the sensitivity of WGA, and hardware-acceleration does this efficiently.
2) We introduce a novel algorithm, Genome Alignment using Constant memory Traceback with X-drop (GACT-X), and its hardware accelerated design. By combining the recently proposed GACT algorithm [20] with X-drop [21], GACT-X is able to carry out arbitrarily long alignments using 2× less memory and 2× fewer cycles than GACT, while still producing better quality alignments. On area-constrained hardware, like an FPGA, this gives over 4× speedup.
3) Both BSW and GACT-X are based on systolic arrays that have been used for accelerating sequence alignment since 1985 [22]. Our proposed changes changes allow a wide variety of prior systolic array architectures, both academic [20], [23] and commercial [24], [25], to potentially reap the benefits of BSW and GACT-X at a relatively small implementation overhead.
4) We combine D-SOFT seeding [20] with hardware-accelerated BSW and GACT-X algorithm in Darwin-WGA — the first hardware-acceleration framework for whole genome alignments. D-SOFT seeding and BSW algorithm use flexible parameters to tune the sensitivity to various points, including the one which recovers every alignment in LASTZ, and GACT-X extends the alignment in hardware while still providing empirically optimal alignments.
5) We analyze the performance of Darwin-WGA on Amazon EC2 FPGA instance (f1.2xlarge) for four pairwise whole genome alignments at different phylogenetic distances and compare the alignments to LASTZ running on a compute-optimized instance (c4.8xlarge). Darwin-WGA provides up to 24× improvement (performance/$) over the iso-sensitive software. It produces a large number of high-scoring alignments that LASTZ fails to discover and results in alignment *chains* that have up to 5.73% higher average scores, contain up to 3× higher matching base-pairs and align up to 2.7% more exons of protein-coding genes.
6) We also analyze the ASIC implementation of Darwin-WGA hardware using TSMC 40nm CMOS through place-and-route. The ASIC requires 35.92mm^2 area and 43.34W of power and provides an average of 1,500× improvement in performance/watt over the iso-sensitive software.

The rest of the paper is organized as follows. Section II provides relevant background on whole genome alignments and alignment chains, section III presents the system design challenges for WGA and describes the algorithms in Darwin-WGA, section IV presents the architectural details of the proposed hardware accelerator, section V describes the experimental methodology, section VI highlights our results, section VII discusses the significance of our results, section VIII introduces related work and section IX concludes the paper.

II. Background

We assume the reader is familiar with the two classic dynamic programming algorithms for sequence alignment — Needleman-Wunsch [26] and Smith-Waterman [19]. Durbin et al. [27] (Chapter 2) provides a good primer on these algorithms.

Whole Genome Alignment. In comparative genomics, whole genome alignment (WGA) refers to the computational process of aligning entire genome sequences of two or more species in order to study their evolutionary relationship. In particular, WGA helps in identifying set of sequences that are *orthologous* (diverged after a speciation event) or *paralogous* (diverged after a duplication event) [28]. As long as the orthologous and paralogous sequences have not diverged beyond the "twilight zone" [29], [14], [15] during the independent evolution of these species, they can be identified using sequence alignment, such as with the Smith-Waterman algorithm [19] — the foundational algorithm in WGA. Sequences resulting in high-scoring alignments, i.e. they exhibit a higher than expected conservation, are assumed to have a biological significance [30]. The actual biological function of these conserved sequences can be further studied in model organisms using genome editing, such as with CRISPR knock-in/knockout experiments [31], [32]. Therefore, the more the number of orthologous and paralogous bases discovered in WGA, the better for biological discoveries. In this paper, we refer to WGA as the task of aligning a single species pair — a *target* species and a *query* species, which typically serves as the first step for whole genome alignment involving multiple species [13], [33].

All whole genome aligners [34], [35], [9], [36] are heuristic approaches to discover local alignments based on the seed-filter-and-extend paradigm, but in this paper, we focus on LASTZ since it is the aligner of choice in the UCSC Genome Browser - the largest hub of cross-species whole genome alignments.

Chains. Kent et. al [34] devised a new utility, called AXTCHAIN, for post-processing raw whole genome alignments produced by tools such as LASTZ. It produces *chains*, which reduce the ambiguities in annotating orthologous and paralogous sequences in a pair of species and help visualize genomic rearrangements. Chains are constructed as maximally-scoring, ordered sequence of alignments produced by whole genome aligners, which could be separated by large gaps including double-sided gaps, i.e., simultaneous gaps in both species. We use AXTCHAIN in this paper to evaluate and visualize the quality of alignments produced by Darwin-WGA in comparison to LASTZ but refer the interested readers to [34] for more details on WGA chains.

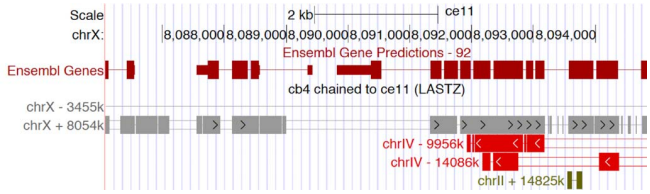


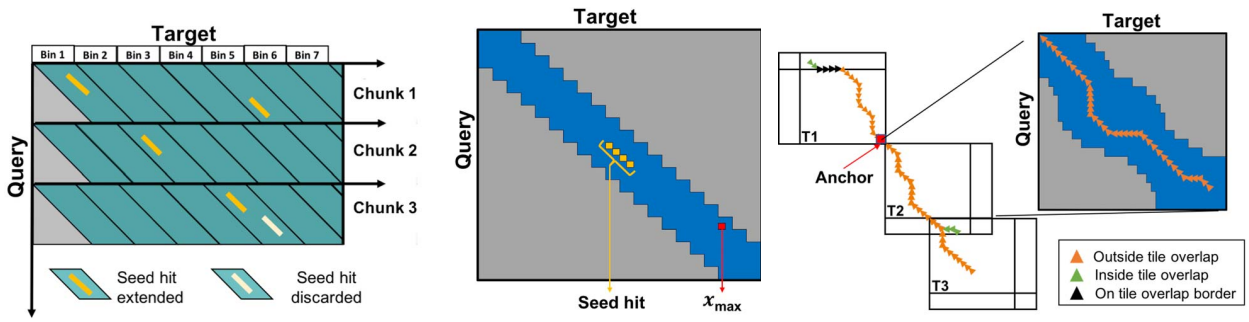
Figure 3: An example UCSC genome browser snapshot of a region (`chrX:8,086,088-8,094,835`) in *C. elegans* genome (ce11) with two annotation tracks - (i) Ensembl gene prediction and (ii) LASTZ chains with *C. briggsae* genome (cb4).

Chains can be visualized using a genome browser. Figure 3 shows a UCSC genome browser [12] snapshot for a region in *C. elegans* genome (ce11 assembly), along with two annotations "tracks" - first for Ensembl gene predictions [37] and second for chains constructed with the *C. briggsae* genome (cb4 assembly) using LASTZ whole genome alignments. Ensembl gene prediction track shows the regions predicted to be genes in the Ensembl 92 gene set - the wide blocks highlighting the *exons* and *untranslated regions* (UTRs) of a gene, the thin lines showing the *introns* and the remaining space is the intergenic region in the genome. The LASTZ chain track shows the five chains aligning to this region in the cb4 genome with a separate color coding scheme for each chromosome in cb4 - thick blocks represent the aligning base-pairs (matching or mismatching), single thin lines showing gaps in cb4, and double lines showing double-sided gaps in both ce11 and cb4.

III. Algorithms in Darwin-WGA

A. System Overview And Challenges

Darwin-WGA accelerates the *seed-filter-and-extend* algorithm used by the popular software whole genome aligners such as LASTZ [17]. It consists of 3 stages – (i) Seeding, (ii) Filtering, and (iii) Extension as depicted in Figure 4 for Darwin-WGA. The *seeding* stage finds small (10-15bp) local matches between the target and the query genomes. Since the matches are short, seeding gives a high false positive rate [38]. The *filtering* stage reduces the false positive rate by aligning longer (hundreds of bp) regions of the two sequences around the "seed hits". Only the hits with a filter score greater than a threshold are aligned in the *extension* stage which may produce alignments that could be millions of bp in length.



(a) D-SOFT filtering (b) Gapped filtering (c) GACT-X extension

Figure 4: Overview of Darwin-WGA (a) Target bins and query chunks constitute a diagonal band, at most 1 seed hit to be extended per diagonal band. (b) Tile for banded Smith-Waterman algorithm, blue represents the band calculated, yellow positions with the seed at its center. (c) Right and left extension from the anchor, tile overlapping and alignment reconstruction from the traceback pointers. The blue band on the right represents the calculated portion of the dynamic programming matrix.

Whole genome alignment present unique algorithm and hardware challenges, particularly in comparison to read assembly that has largely been the focus of prior work in hardware acceleration [20], [39], [24], [25], [40], [41]. *First*, WGA requires finding alignments between genome sequences that have been evolving independently for millions of years and exhibit low sequence similarity (50-100%). In comparison, read assembly aligns sequences that are mostly similar (80-100%) and have a well-characterized error model that depends on the sequencing technique. Aggressive seeding strategies used in read assembly, such as super maximal exact matches (SMEM) [42] or those based on finding multiple collinear seed hits [43], [20], greatly reduce the sensitivity of WGA. *Second*, whole genome aligners rely on more sensitive seeding strategies that initially allow many false positives (to not lose true positives) and let the more sensitive filtering stage discard a majority of those false positives. As a result, by far, the filtering stage dominates the runtime of WGA, which is in contrast to read assembly, where either the extension or the seeding stage dominates the runtime [42], [43], [20], [39]. *Third*, whole genome alignments could also span millions of bp - orders of magnitude longer than typical sequencing reads. Also, they may contain large insertions and deletions (indels) spanning hundreds of bp (corresponding to the large structural changes that occur during the genome evolution [44]) that are not typically encountered in sequencing reads. Long alignments with large indels pose prohibitive on-chip memory requirements on previously proposed hardware architectures [20], [24], [23] (also highlighted later in Figure 10).

B. D-SOFT filtering

A "seed hit" is a short match between the target and query genome sequences. It is determined by a spaced seed pattern, such as the default seed pattern (*12of19*) used in both Darwin-WGA and LASTZ (Figure 5a).

```
Seed pattern
1110100110010101111

Seed hit
GTAGCGGGCAACTATCCTT
|||X|XX||XX|X|X||||
GTAACTTGCTTAGACCCTT
```

(a) No Transition

```
Seed pattern
11Z0100110010101111

Seed hit
GTAGCGGGCAACTATCCTT
||XX|XX||XX|X|X||||
GTGACTTGCTTAGACCCTT
```

(b) 1 Transition

Figure 5: Default seed pattern in LASTZ and Darwin-WGA with an example of the corresponding seed hit for (a) no transition and (b) 1 transition (in red). In seed pattern, 1 represents an exact match, 0 represents a don't care, and Z represents a transition match.

A more sensitive seed model allows for a subset of substitutions, *transition substitutions* – $A \leftrightarrow G$ and $T \leftrightarrow C$ – in place of a match. Empirically, transitions occur at a higher than random frequency possibly because of the underlying similarity in the shapes of A and G, and T and C as explained in [45]. Figure 5b shows the default seed pattern which allows 1 transition mismatch in place of an earlier match position. When 1 transition mismatch is added to a seed pattern with *m* match positions, it effectively results in $(m + 1)$ more seeds that leads to $(m + 1)$ times more computation.

Darwin-WGA uses a modified D-SOFT algorithm [20] for seeding and preliminary filtering. It divides the query genome into small chunks and searches for seed hits in the target genome for each chunk. Every seed hit falls within a single diagonal band that is associated with a target bin as

shown in figure 4a. Unlike [20], the threshold parameter h is based on the number of seed hits per diagonal band. At most 1 seed hit is extended per diagonal band. This reduces redundant extensions for seed hits within the same diagonal band.

Sensitivity of the alignments depends on the seed hits found and extended. This, in turn, depends on (i) the chunk size, c, (ii) the bin size, b, and (iii) whether or not inclusion transition mismatches are included, tr. Sizes c and b are set large enough to extend only 1 among the closely present seed hits, and avoid unnecessary computation and duplicate alignments. On the other hand, they are low enough to avoid missing out on any seed hit otherwise found by LASTZ. The default settings of LASTZ and Darwin-WGA include transition mismatches in the seed. Both include the support for not including transition mismatch which would result in potentially decreased sensitivity and reduced computation time.

C. Gapped filtering

Seed hits found by D-SOFT are filtered using Banded Smith-Waterman (BSW) [46] for gapped alignment. A tile of size T_b is created with the seed hit at its center, as shown in Figure 4b. The band extends to a length of band size (B) on either side of the diagonal of the tile. With substitution matrix and gap penalties as inputs, dynamic programming along the band using Smith-Waterman equations provides the maximum score, V_{max} and the corresponding position, x_{max} of the tile. If V_{max} is greater than the filtering threshold (H_f), the seed hit is passed to the extension stage using x_{max} as the *anchor*, the starting position for alignment extension.

Unlike Darwin-WGA, LASTZ filters using X-drop ungapped extension [17]. The seed hit is extended along the diagonal from both sides and terminated when the score falls below (maximum score - x-drop value) [21]. If the maximum score is higher than the filtering threshold, an anchor position is found for the alignment extension stage. Ungapped filtering in LASTZ does not allow for any indels. As described in section I, average ungapped block sizes decrease as species are more diverged. Darwin-WGA improves the sensitivity over LASTZ because gapped filtering through BSW allows indels within the band, and therefore, filters out fewer potential true positive seed hits. BSW can also provide increased sensitivity by allowing indels introduced due to sequencing and assembly errors [47] during filtration.

The filtering stage dominates the runtime of both LASTZ and Darwin-WGA owing to the large number of seed hits found in the seeding stage. A vast majority of these are false positives and get filtered out, reducing the workload of the more computationally-expensive gapped extension stage that produces the final set of alignments.

D. GACT-X extension

Darwin-WGA extends the qualifying anchor positions along the upper left (left extension) and lower right (right extension) using GACT-X. It uses a strategy similar to the GACT algorithm [20] for aligning arbitrarily long alignments using constant memory while incorporating the X-drop algorithm [21] (referred to as Y-drop algorithm in [17]) to reduce computation and memory. GACT was developed for long read alignment where gaps are frequent but short. On the other hand, WGA requires aligning over much longer gaps that appear during the evolution of species. We show later in section VI that in order to generate such alignments, GACT tile size would have to be increased with a significant increase in the memory footprint. With the X-drop algorithm, GACT-X can align large sequences with a much smaller on-chip memory requirement and processing time as compared to GACT and still provide near-optimal alignments even for highly divergent species.

GACT-X generates alignments in an overlapping tiled fashion as shown in figure 4c. For each tile, GACT-X requires a substitution matrix, gap penalties, and the Y parameter as input. The cell scores are calculated row-wise and the maximum score, V_{max}, is also tracked. The computation in a row is terminated as soon as the cell score falls below $(V_{max} - Y)$. The next row begins computation from the first column where the score in the cell of the previous row exceeded $(V_{max} - Y)$ value.

GACT-X uses Needleman-Wunsch [26] scoring instead of Smith-Waterman scoring to allow for negative scores which may occur when $(V_{max} - Y)$ is negative.

Once the score calculation completes, the direction pointers are traced from the maximum score position to the beginning of the tile. Figure 4c shows the computed region in blue and the traceback pointers in orange. GACT-X ensures that any gaps at the beginning of the tile are a part of the final alignment. This allows for neighboring tiles to be "stitched" together to produce the final local alignments similar to the ones found by LASTZ.

Similar to GACT, GACT-X forces neighboring tiles to overlap, specified by a parameter O, to reduce the sub-optimal alignments at the tile boundaries. Traceback pointers within the overlap region in a tile are ignored during the extension of that tile. If x_{max} lies outside the overlap region, it is the starting point of the next tile. Otherwise, the starting point is at the intersection of the alignment and overlap boundary. The extension in a direction terminates when the V_{max} of a tile is negative or 0. The final alignment is given as an input if its score is greater than the extension threshold, H_e.

During GACT-X extension, Darwin-WGA implements a hash strategy to remove anchors that would result in duplicate alignments, similar to the anchor absorption strategy in LASTZ. If an unextended anchor is a part of a previous

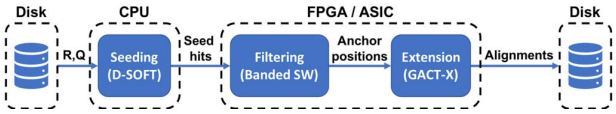
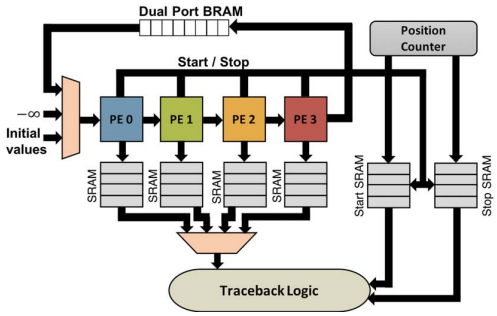
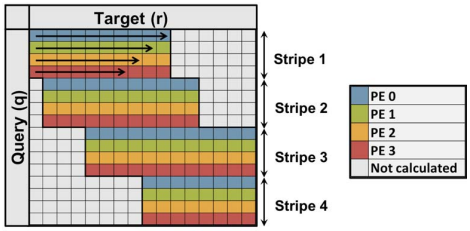


Figure 6: Hardware-software partitioning of the different stages of Darwin-WGA.



(a)



(b)

Figure 7: (a) GACT-X array with 4 PEs. (b) An example of dynamic programming matrix for GACT-X. The arrow heads represent the wavefront.

alignment, it is not extended.

IV. Hardware Architecture

Darwin-WGA implementation is partitioned between hardware and software, as shown in Figure 6. D-SOFT filtering is implemented in software using multiple threads. Linear systolic arrays of N_{pe} processing elements accelerate the Banded Smith-Waterman (BSW) filtering and the GACT-X extension stages. Software configures the arrays, controls the execution of individual tiles, and reconstructs the final alignments from GACT-X array traceback pointers.

Before beginning any computation, the software transfers the target and the query genome sequences (R,Q) onto the DRAM. The software loads the substitution matrix(W), gap extend(e) and gap open(o) penalties, bandwidth (B), and threshold (Y) onto the hardware.

For each tile, the software provides the target and query beginning positions (r_0,q_0) and the corresponding length(r_{len},q_{len}) to the array as inputs. The arrays read the sequences of lengths r_{len} (or q_{len}) starting from r_0 (or q_0) from the DRAM. The input sequence characters are a part of the extended DNA alphabet A, C, G, T, N. The input is ASCII encoded but they are stored in the BRAMs using 3 bits.

The GACT-X array calculates the scores of the cells in a systolic fashion and exploits *wavefront parallelism* along a stripe of N_{pe} rows. The corresponding N_{pe} cell scores and 4-bit direction pointers are calculated in 1 cycle. Each PE computes the following 3 scores according to the Needleman Wunsch scoring system with affine gap [48] penalty.

Equation 1 calculates the affine gap score for insertion and equation 2 for deletion. Equation 3 calculates the final score which determines the direction pointer. The 4-bit direction pointer consists of – 2 bits to encode whether V was calculated from the horizontal, vertical, diagonal cell or terminating, and 2 bits for representing the affine gap property – whether the insertion or deletion came from opening or extending a gap. Also, V_{max} is calculated in a systolic manner using $V(i,j)$ by each PE.

The first stripe of a tile starts in column 1. Each succeeding stripe starts at the column (j_{start}) where the scores of all the cells exceeds ($V_{max} - Y$). Computation continues along each stripe until the scores of all the cells in a column (j_{stop}) fall below ($V_{max} - Y$) or the computation reaches the final column ($j_{stop} = r_{len}$).

$$I(i,j) = \max \begin{cases} V(i,j-1) - o, \\ I(i,j-1) - e \end{cases} \quad (1)$$

$$D(i,j) = \max \begin{cases} V(i-1,j) - o, \\ D(i-1,j) - e \end{cases} \quad (2)$$

$$V(i,j) = \max \begin{cases} I(i,j) \\ D(i,j) \\ V(i-1,j-1) + W(r_i, q_j) \end{cases} \quad (3)$$

Figure 7 depicts a GACT-X array with $N_{pe} = 4$ with the corresponding dynamic programming matrix in figure 7b. The query characters corresponding to a stripe are loaded to the PEs and the target elements are streamed thought the PEs in a systolic fashion. The V and D values from the last PE of the stripe are stored in the dual port BRAM. Depending on whether a column in the previous stripe was calculated or not, values are either read from the dual port BRAM or initialized to make sure the direction pointers remain within the calculated regions. Also, the initialization scores for the cells in j_{start} guarantee that the direction pointers do not extend beyond the calculated region.

Memory allocated to the direction pointer BRAM is fixed, which is divided into 1 bank per PE. The traceback pointers from the stripes are stored sequentially and the number of columns calculated within each stripe is not fixed. Hence, the start positions and stop positions are stored in separate BRAMs to transition from 1 stripe to another in the traceback logic.

Species	Assembly	Size (Mbp)
C. elegans	ce11	100.0
C. briggsae	cb4	105.0
D. melanogaster	dm6	137.5
D. simulans	droSim1	110.0
D. yakuba	droYak2	120.0
D. pseudoobscura	dp4	127.0

Table I: Species with their assembly name and size.

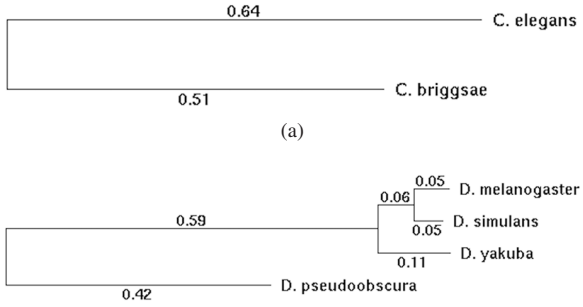


Figure 8: Two phylogenetic trees along with the phylogenetic distances (in substitutions/site) between the species used in the whole genome alignments.

The traceback logic starts at the maximum score position and traverses each step according to the pointers stored in the direction and the position BRAMs. If the traceback reaches the beginning of the target or query, but not the origin, insertions or deletions are added. The 2-bit traceback pointers are sent to software for reconstruction.

The BSW array is a subset of the GACT-X array. It does not require traceback pointers, only the maximum score and positions. Each PE computes the scores according to the Smith-Waterman scoring system with affine gap penalty. Because the band is fixed, the start (j_{start}) and stop (j_{stop}) columns are simple functions of the stripe number n and the bandwidth B, as shown in equations 4 and 5 below.

$$j_{start} = \max\{0, ((n-1)N_{pe} + 1 - B)\} \quad (4)$$

$$j_{stop} = \min\{(r_{len} - 1), (nN_{pe} + B)\} \quad (5)$$

V. Methodology

A. Genome data

To demonstrate the quality and speed of the whole genome alignments, we compared our alignments to LASTZ for alignments in two species groups. First, we chose the nematodes (roundworms), with *Caenorhabditis elegans* — amongst the most extensively used model organism [49] — as the target genome, and align it to the genome of its closest nematode species, *Caenorhabditis briggsae*. Second, we used *Drosophila* (fruit flies), with *Drosophila melanogaster*

Substitution Matrix (W)				
	A	C	G	T
A	91	-90	-25	-100
C	-90	100	-100	-25
G	-25	-100	100	-90
T	-100	-25	-90	91
Gap Penalties				
gap_open			o	-430
gap_extend			e	-30

(a)

Gapped filtering		
Tile Size	T_f	320
Band Size	B	32
Threshold	H_f	3000
GACT-X		
Tile Size	T_e	1920
Overlap	O	128
Y-drop	Y	9430
Threshold	H_e	4000

(b)

Table II: Parameters used for Banded Smith-Waterman and GACT-X arrays in Darwin-WGA.

— another extensively used model organism [50] — as the target genome, and aligned it to the genomes three other species at varying phylogenetic distances from target - (i) *Drosophila simulans*, (ii) *Drosophila yakuba* and (iii) *Drosophila pseudooobscura*. For all genome assemblies, we only use nuclear chromosomes, and remove mitochondrial DNA and unmapped and unlocalized contigs. The assembly details and genome size of the species used in this study are shown in Table I. The improvements of Darwin-WGA depend on the phylogenetic distance of species - for computing the phylogenetic tree and distances between species, we used the PHAST tool [51], and are shown in Figure 8.

B. Setup and Comparison Baseline

Software components of D-SOFT were implemented in C++ and compiled using `g++` (version 5.4.1) at `-O4` optimization level. The parameters used in Darwin-WGA are shown in Table II.

The software baseline was run on a large compute-optimized instance (c4.8xlarge) of Amazon Elastic Cloud Compute (EC2) service. The instance is priced at $1.59/hour at the time of writing. We compared Darwin-WGA alignments and performance to the default settings of LASTZ (version 1.02.00), a state-of-the-art whole genome aligner, built with gcc-4.9.4 compiler. LASTZ default scoring parameters are identical to Darwin-WGA (in table IIa) except the filtration and extension thresholds (equivalent of H_f and H_e) in LASTZ are lower, at 3000. LASTZ is only implemented using a single thread, but for a fair comparison, we created multiple parallel processes of LASTZ (divided by genomic intervals) and used the `parallel` utility in Linux to ensure that all 18 available cores on the instance were fully utilized.

To measure processor and DRAM power on the EC2 instance, we used `cpu-energy-meter` [52], which internally uses Intel RAPL (Runtime Average Power Limiting) [53].

To estimate the speedup resulting from the hardware acceleration of Darwin-WGA, we obtained the runtime of the gapped filtering stage (which would dominate the overall software runtime) using Parasail [54] (version 1.1.12) —

which provides a fast, open-source software implementation of BSW. This runtime is obtained using the number of gapped filtration tiles required in Darwin-WGA and the average tile throughput for the same tile size in Parasail and also corresponds to the runtime of an iso-sensitive software to Darwin-WGA. All cores of the processor were fully utilized while estimating Parasail throughput.

We also compared the alignment quality and throughput of GACT-X with the GACT algorithm [20]. The traceback memory size was varied from 512KB to 2Mb and the tile size was determined accordingly. To estimate the alignment quality, we provided the same starting anchors derived from the Darwin-WGA seeding and filtering stage using chromosome X of ce11 and cb4 genome to both GACT and GACT-X and counted the number of matching base-pairs in the alignments for each algorithm. Throughput was measured in terms of the number of base-pairs aligned per second. Both metrics were normalized to GACT-X with default settings used in Darwin-WGA.

C. FPGA Design, Synthesis And Power

We implemented Darwin-WGA on an FPGA instance (f1.2xlarge) on the Amazon EC2 service. This instance contains one Xilinx Virtex UltraScale Plus FPGA connected to a 64GB DDR4 memory and is priced at $1.65/hour at the time of writing — comparable to the software baseline instance. Our hardware design was written in Verilog and compiled using the Xilinx OpenCL Compiler (`xocc`) utility in the SDAccel Environment [55]. We were able to map 50 BSW arrays and 2 GACT-X arrays, each with $N_{pe} = 32$ PEs, on the FPGA and operate the clock at 150MHz. `xocc` was also used to provide the chip power estimate of a fully routed design in vectorless mode. DRAM power was separately estimated using [56].

D. ASIC Synthesis, Layout And Power

For ASIC analysis, We also synthesized single BSW and GACT-X arrays, each with N_{pe}=64 PEs, to a TSMC 40nm CMOS process using the tcbn40lpbwp standard cell library at the worst-case process-voltage-temperature (PVT) corner. Synthesis was performed using Synopsys Design Compiler (DC) [57] in topographical mode for pre-layout frequency and power estimates. Synopsys IC compiler (ICC) [58] performed place and route to estimate the ASIC area. SRAM memory was excluded during DC synthesis and Cacti [59] was used to estimate the area and power of the SRAM memory. FPGA cycle counts were used to estimate ASIC throughput by scaling to ASIC frequency. We used Ramulator [60] to estimate the peak memory bandwidth for the ASIC memory trace with four DDR4-2400R x8 memory channels. We provisioned the number of BSW and GACT-X arrays on the ASIC to make DRAM bandwidth the bottleneck and scaled the area and power estimates accordingly. DRAMPower [56] was used to estimate the DRAM power.

E. Sensitivity and Noise Analysis

Both LASTZ and Darwin-WGA generate output alignments in MAF (Multiple Alignment Format) format [61] which is post-processed to generate chains using the AXTCHAIN utility [34] with flag `-linearGap=loose`. These chains are uploaded to the UCSC genome browser [62] for visualization.

In absence of ground-truth, measuring the sensitivity in the final set of whole genome alignments is a challenge. We measure sensitivity using 3 different metrics: (i) the score comparison of the top-10 chains — which serve as a good proxy for estimating the number of orthologous base-pairs, (ii) the number of matching base-pairs in all chains — which helps in estimating the sensitivity on orthologous as well as paralogous base-pairs and, (iii) the number of "orthologous" exons of protein-coding genes found in chains — which serves as a proxy for the sensitivity on regions of the genome with high functional significance. In order to find the orthologous exons, we use all exons in the protein-coding gene set in the Ensembl 92 gene set for the two target genomes (ce11 and dm6) and use TBLASTX [63] - a sensitive alignment tool specialized for discovering protein-coding alignments - to align with the query genomes with `-evalue 1e6 -max_target_seqs 1` setting for maintaining only high-confidence alignments.

Improvements in alignment sensitivity often come at the cost of higher noise (more false positive alignments). In order to estimate the false positive rate (FPR) of Darwin-WGA, we first create a "random" target genome by shuffling the 2-mer sequences of ce11 using the `fasta-shuffle-letters` utility [64]. This shuffle preserves the 2-base statistics which are pronounced in genomes [65] while still producing a null model sequence that is not derived from evolution. We produce a whole genome alignment to this sequence with cb4 query and consider every obtained alignment as a false positive. The experiment is repeated 3 times (with differently shuffled ce11 genome) in order to get statistically significant results. A similar approach is used for estimating the FPR of LASTZ as a baseline.

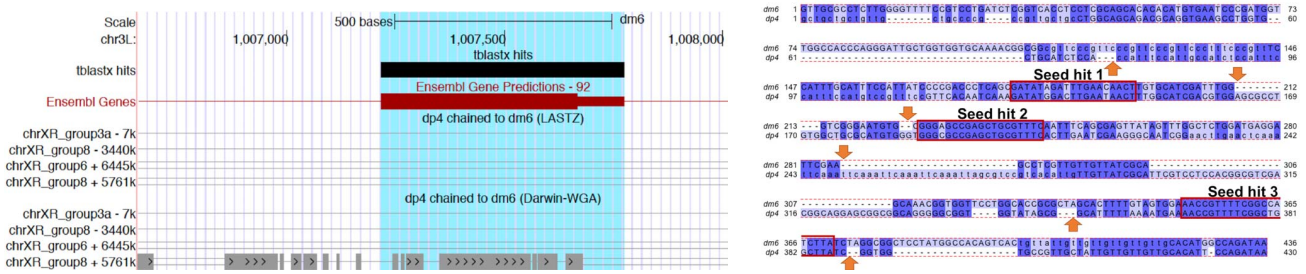
VI. Results

A. ASIC Synthesis, Layout, and Frequency

The power and area analysis of the various components of Darwin-WGA ASIC are summarized in Table IV. The critical path of the design has a delay of 1ns resulting in 1GHz operational frequency. At 40nm technology, the ASIC has an area of nearly 36mm^2 with a peak power of 43.34W. BSW arrays dominate the logic area of the ASIC and consume almost 60% of the chip power. GACT-X uses the majority of SRAM on the chip for storing the direction pointers which take up nearly half of the chip area.

Species Pair	Top 10 chain scores	Matched Base-Pairs Counts		Exon Counts		
		LASTZ	Darwin-WGA	Total (TBLASTX)	LASTZ	Darwin-WGA
ce11-cb4	(+5.73%)	57,935,120	180,768,878 (3.12×)	95,430	90,623	93,072 (+2.70%)
dm6-dp4	(+1.86%)	70,428,788	100,116,352 (1.42×)	58,898	58,111	58,347 (+0.41%)
dm6-droYak2	(+0.05%)	127,821,760	180,606,123 (1.41×)	75,836	75,634	75,700 (+0.09%)
dm6-droSim1	(+0.03%)	117,963,461	147,482,665 (1.25×)	71,673	70,991	71,134 (+0.20%)

Table III: Sensitivity comparison of Darwin-WGA and LASTZ for three different metrics. Darwin-WGA improvements are shown in parenthesis.



(a) (b)

Figure 9: (a) UCSC genome browser shot of a region in the dm6 genome consisting of a single exon gene to which an alignment was found using TBLASTX to dp4 genome. LASTZ and Darwin-WGA chain tracks show that the alignment was discovered by Darwin-WGA but missed by LASTZ. (b) Alignment of dm6 and dp4 sequences for the highlighted region in (a). Red boxes show that three seed hits are contained in the alignment. Upward and downward pointing arrows show the beginning of gaps around seed hits.

	Component	Configuration	Area (mm^2)	Power (W)
BSW	Logic	64 × (64PE array)	16.6	25.6
GACT-X	Logic	12 × (64PE array)	4.2	6.72
	Traceback SRAM	12 × (64PE × 16KB/PE)	15.12	7.92
DRAM	DDR4-2400R	4 × 32GB	-	3.10
Total			**35.92**	**43.34**

Table IV: Area and Power breakdown of individual components in Darwin-WGA ASIC operating at 1.0GHz.

The performance of this chip is limited by the available memory bandwidth which can be improved through memory link compression or the use of higher-bandwidth memories (GDDR or HBM) and scaling up the number of arrays accordingly.

B. Sensitivity and Noise Analysis

Table III summarizes the performance of Darwin-WGA and LASTZ in their respective default modes in terms of the sensitivity metrics described in section V-E. Darwin-WGA has longer and better scoring top chains as compared to LASTZ for all pairs of species considered. The top scoring chains are used as a metric to quantify the number of orthologous base-pairs aligned between the species. The average improvement in the scores of the top-10 chains is more significant for more distantly related species – ce11-cb4 and dm6-dp4 (Figure 8) than closely related pairs – dm6-droSim1 and dm6-droYak2. Therefore, by allowing gaps in the filtering stage, Darwin-WGA generates much better alignments than LASTZ for distantly related species.

Table III also indicates that the matching base-pairs in Darwin-WGA chains are significantly higher than those in LASTZ chains. It follows the expected trend of higher improvement with greater phylogenetic distance between the species pair. The matching base-pairs represent sequence conservation and help in identifying the orthologous as well as paralogous sequences between species. In general, paralogs are more numerous and faster evolving than orthologs making them harder to discover with WGA [66]. Darwin-WGA helps identify these paralogs with much higher sensitivity. We believe that Darwin-WGA alignments can aid in better understanding of the genomic functionality and several other biologically significant insights.

Table III shows the total number of the exon counts from the TBLASTX analysis of the target genomes in the Ensembl 92 gene set and the corresponding query genomes. Here again, Darwin-WGA outperforms LASTZ in every case and the percentage of improvement depends on both the phylogenetic distance between the species and the number of exons not covered by LASTZ. WGA of ce11-cb4 shows the highest improvement of 2.7%, since LASTZ covers 95%

Species pair	LASTZ runtime (sec)	Darwin-WGA Workload			Iso-sensitive s/w runtime (sec)	Darwin-WGA runtime (sec)		Darwin-WGA Improvement	
		Seeds (10^6)	Filter tiles (10^6)	Extension tiles (10^6)		FPGA	ASIC	FPGA (Perf/$)	ASIC (Perf/W)
ce11-cb4	481	1,362	14,585	4.4	64,960	3,823	219	19.1×	1,478.3×
dm6-dp4	643	2,972	32,070	2.9	142,627	5,936	461	23.2×	1,547.2×
dm6-droYak2	654	2,835	32,476	3.7	144,454	6,001	469	23.2×	1,539.8×
dm6-droSim1	557	2,478	28,267	2.1	125,700	4,987	404	24.3×	1,553.2×

Table V: Comparison of the runtimes of LASTZ, iso-sensitive software and Darwin-WGA (FPGA and ASIC). Darwin-WGA workload has also been shown for the three stages. Performance/$ is used as a metric for comparing the improvement of the FPGA implementation and Performance/Watt for the ASIC implementation of Darwin-WGA, versus the iso-sensitive software.

	CPU (c4.8xlarge)	FPGA (Virtex UltraScale+)	ASIC (TSMC 40nm)
Power (W)	215	65	43

Table VI: Power estimations (including DRAM) of the three computing platforms.

of the exons found in TBLASTX, leaving a relatively large room for up to 5% improvement. For dm6-droYak2, LASTZ covers as high as 99.73% of the exons. Hence, a 0.09% improvement by Darwin-WGA is still significant. A higher exon coverage in Darwin-WGA implies that there are higher number of accurate alignments as compared to LASTZ.

The UCSC browser shot in Figure 9 shows an example of a biologically significant region marked by an exon where Darwin-WGA finds an alignment but LASTZ does not. The browser shot alignment chains for the dp4 genome generated using Darwin-WGA and LASTZ. The highlighted portion covers a single-exon Ensembl gene in dm6 to which an alignment hit is found by TBLASTX with the dp4 genome. Figure 9b zooms into the base-pair level view of the dm6-dp4 alignment in the highlighted region. The alignment is not only long (over 400bp) but also has high conservation (over 58%) and should ideally be detected by a whole genome aligner. The particular alignment contains three seed hits (highlighted in red boxes) found by both, Darwin-WGA and LASTZ. However, close to each seed hit, there are gaps (indels) on either side of the hit (shown in upward and downward pointing arrows) which explains why the region gets dropped in the ungapped extension stage in LASTZ. On the other hand, Darwin-WGA accounts for gaps through BSW, and extends these particular seed hits to an alignment.

Darwin-WGA outperforms LASTZ on all the metrics of sensitivity and the added sensitivity can be completely attributed to gapped filtering stage. In order to makes sure that the additional sensitivity does not come with a high FPR, we did the noise analysis described in section V-E. When cb4 genome is chained with ce11 using Darwin-WGA, 180,768,878 matching base-pairs were found in the chains. When cb4 genome is aligned to the shuffled ce11 sequence (preserving 2-base statistics), only 1,334 matching base-pairs were found in the chains on an average over the 3 experiments, resulting in an FPR of 0.0007%. For the same set of experiments using LASTZ, the FPR was found to be 0.0002%, with 57,935,120 and 103 average matching base-pairs using ce11 and shuffled ce11 genome, respectively. FPR is mostly affected by the choice of H_f in Darwin-WGA. If we reduced H_f in Darwin-WGA to LASTZ's default H_f of 3000, it would result in a significantly high FPR of 1.48% since more gaps are tolerated in the filtering stage. Hence, we use $H_f = 4000$ for the default settings in Darwin-WGA for a significant improvement in sensitivity with an acceptable false positive rate.

C. Performance Analysis

Table V provides the runtimes of LASTZ, iso-sensitive software, as well as the FPGA and ASIC implementations of Darwin-WGA. FPGA improvement over the iso-sensitive software on CPU is calculated using the performance/$ metric. Since it is difficult to estimate the cost to provide a service using the Darwin-WGA ASIC on the cloud, we use performance/watt as a proxy to measure the ASIC improvement over the iso-sensitive CPU baseline. Power requirements for CPU, FPGA and ASIC are summarized in Table VI. LASTZ runtime has been recorded using all 36 cores on the c4.8xlarge Amazon EC2 instance. With a similar resource setup for Parasail, we recorded a throughput of 225K tiles/sec, which is used for estimating the runtime for software iso-sensitive to Darwin-WGA. The total workload for Darwin-WGA is also shown in Table V in terms of the number of seeds for seeding stage and the number of tiles for filtering and extension stages. Table V shows that in going from LASTZ to iso-sensitive software, there is a considerable slowdown in software — 200× on average. This shows that the added sensitivity in Darwin-WGA (Table III) comes at a high computational cost, which is consistent with the observations in [16], [18].

Hardware-acceleration of Darwin-WGA on FPGA as well as ASIC shows great improvements. On the Xilinx Virtex UltraScale Plus FPGA, we fit a total of 50 BSW arrays and 2 GACT-X arrays, each with 32 PEs. At a frequency of 150 MHz, FPGA gives a throughput (bandwidth) of 6.25M tiles/sec (2.1 GB/s) for BSW and 4.6K tiles/sec (14.6

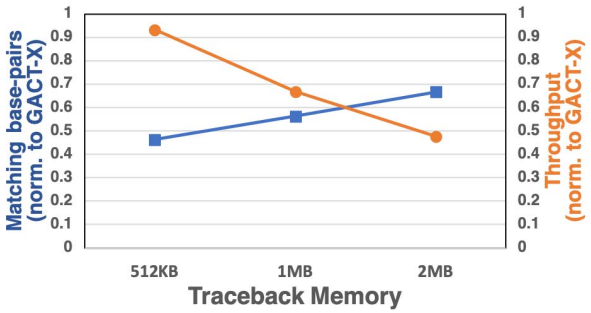


Figure 10: Alignment quality (in matching base-pair count) and throughput (in base-pairs aligned per second) of GACT for different traceback memory sizes normalized to the default configuration of GACT-X.

MB/s) for GACT-X. FPGA shows 19-24× improvement in performance/$ compared to iso-sensitive software. Most improvements on FPGA result from the acceleration of the gapped filtering stage, which has a 27× performance/$ improvement, but accelerating GACT-X on FPGA is also necessary to maintain high performance.

For the Darwin-WGA ASIC described in section VI-A, we get a throughput of 70M tiles/sec (44.8GB/s) for 64 BSW arrays and 300K tiles/sec (1.15 GB/s) for 12 GACT-X arrays. On average, Darwin-WGA ASIC gives $1,500\times$ performance/watt improvement over the iso-sensitive software. Interestingly, at 5× lower power, Darwin-WGA ASIC is 1.3-2× faster than LASTZ, implying that the ASIC provides both higher speed and sensitivity over the state-of-the-art software.

D. GACT-X: Comparison with GACT

Figure 10 shows the number of matching base-pairs and the throughput (in number of base-pairs aligned per second) of GACT for three different traceback memory sizes, normalized to GACT-X. The number of base-pairs aligned by GACT increases as the traceback memory size increases, indicating that GACT is able to discover longer alignments, but at lower throughput. At 1MB traceback memory (same as GACT-X), GACT does much worse in both throughput, at 0.66×, and matching base-pairs, at 0.56×, compared to GACT-X. In fact, GACT does worse even at twice the traceback memory of GACT-X. At 2MB, GACT uses a tile size of 2048bp, compared to 1920bp in GACT-X, and even though GACT computes more cells than GACT-X, its alignments are worse overall. This is because, unlike GACT-X, which roughly constraints the alignment close to the main diagonal of the tile, GACT allows the alignment within a tile to deviate much farther from the diagonal in presence of large gaps, which tends to terminate alignments early even when multiple tiles are used. GACT is well-suited for long read alignments, where are gaps are short but frequent, whereas GACT-X is better suited for cross-species alignments, where gaps are fewer but tend to be long.

VII. Discussion

Large genome projects, such as Genome 10K [5] and B10K [6], lead multi-million dollar efforts to sequence and assemble more than 10,000 vertebrate genomes by 2021. The Earth BioGenome project [67], which was established in early 2018, aims at sequencing the genomes of all species on the planet at a multi-billion dollar budget. Pairwise whole genome alignments are imperative for gaining new biological insights from the generated genome assemblies.

At 10,000 vertebrate genome assemblies, whole genome alignments even for a small fraction of the possible 100 million species pairs would incur several millions of dollars of computational cost. The cost would further increase by 200× for whole genome alignments as sensitive as Darwin-WGA, making it prohibitively expensive to perform such sensitive alignments using software. Darwin-WGA ASIC would require a fraction of the sequencing and assembly costs of 10,000 vertebrate genomes to design and manufacture, and with 305× speedup at 5× less power, it could provide an economically-meaningful solution to carry out sensitive whole genome alignments at scale. Currently, the FPGA implementation of Darwin-WGA already provides a compelling solution for performing sensitive whole genome alignments, saving up to 24× in dollar cost compared to iso-sensitive software.

VIII. Related Work

Hardware architectures. Prior work in hardware acceleration of genome sequence alignment has largely been focused on the assembly of sequencing reads [39], [24], [20], [41]. GenAx [39], Edico Genome [24] and MPU-BWM [41] accelerate the assembly of low-error short reads, while Darwin [20] proposes an accelerator for the assembly of noisy long reads. As mentioned in section III-A, read assembly primarily requires the acceleration of the seeding and extension stages, unlike WGA, where filtering is the computationally dominant stage. Moreover, WGA requires a more sensitive filtering stage compared to read assembly and requires the extension stage to model the large structural changes happening during the evolution of genome accurately.

Previous work [25], [23] has also designed hardware accelerators for BLAST [68] — the most widely used algorithm for cross-species homology search and a precursor to LASTZ [17]. However, BLAST is designed for searching homologous sequences using short query sequences in a large database and does not handle whole genome alignments. Architectures in [25], [23] are also limited in the lengths of sequences they align (few kilo base-pairs). Of prior work, Darwin [20] is the only hardware-accelerator capable of aligning arbitrarily long sequences through GACT

algorithm, but as we show in our results, WGA imposes prohibitive on-chip memory requirements for GACT and does not perform as well as GACT-X. To that end, Darwin-WGA is the first hardware accelerator designed primarily for WGA.

Algorithms. Many algorithms for WGA were proposed in the previous decade. These include AVID [69], MUMMER [36], BLASTZ [9], and LASTZ [17]. Over the years, LASTZ has become the de facto standard for whole genome alignments for its high sensitivity and speed. It is the primary algorithm in many multiple sequence alignment tools, including MULTIZ [33] and Cactus [13]. LASTZ is also the default WGA tool used in the UCSC Genome Browser [12]. Recent work [16], [18] has shown that lowering the threshold of ungapped filtering improves the sensitivity of LASTZ significantly, but, at a high computational cost. Cactus [13] also lowers the same threshold in LASTZ for higher sensitivity. Darwin-WGA is the first whole genome aligner that replaces the ungapped filtering stage of LASTZ with a gapped filtering stage and improves WGA sensitivity substantially.

In recent years, much work [40], [70], [71], [43] has focused on fast seeding and filtering approaches for read assembly. Aggressive seeding and filtering strategies work well in read assembly, but for WGA of distantly related species, these stages are required to be far more sensitive. By using a hardware-accelerator for the gapped filtering stage, Darwin-WGA provides a highly sensitive filtering stage for WGA at practical speed.

IX. Conclusion And Future Work

Remarkable advances have been made in sequencing and editing of genomes in recent years, but we are still far from decoding the function of the genome. Whole genome alignments help in identifying conserved sequences between species. Using these conserved sequences, techniques such as genome editing help in decoding the function of the genome. In this paper, we show that by replacing ungapped filtering used in every software whole genome aligner, by gapped filtering, provides improved sensitivity with up to $3\times$ more matching base-pairs in the alignment chains. Increased sensitivity with gapped filtering comes at a high slowdown by over 2 orders of magnitude in software. In this paper, we present Darwin-WGA, a hardware acceleration framework for whole genome alignment with gapped filtering. It provides up to $24\times$ dollar cost improvement using FPGA and $1,500\times$ performance/watt improvement using ASIC. We also propose a novel algorithm and accelerator for GACT-X which allows arbitrarily long whole genome alignments to be carried out with constant hardware memory and is at least $2\times$ better in memory requirement and speed compared to previously published method. We believe that Darwin-WGA is a practical whole genome aligner that allows for sensitive whole genome alignments to be carried out at scale and could potentially lead to more biological discoveries. A future version of Darwin-WGA will also allow for TBLASTX-like search in the amino acid space for protein-coding genes in addition to DNA alignments for rest of the genome.

Acknowledgment

We thank the anonymous reviewers for their helpful feedback. We thank Heidi Chen for reviewing the manuscript and her valuable comments. We thank Prof. Boris Murmann and Kevin Zheng, who provided the access to ASIC CAD tools. We also thank "AWS Cloud Credits for Research" program and Xilinx for providing University licenses of development tools.

References

[1] NCBI, "Ncbi genome database." http://www.ncbi.nlm.nih.gov/genome/, 2018.

[2] K. Lindblad-Toh, M. Garber, O. Zuk, M. F. Lin, B. J. Parker, S. Washietl, P. Kheradpour, J. Ernst, G. Jordan, E. Mauceli, *et al.*, "A high-resolution map of human evolutionary constraint using 29 mammals," *Nature*, vol. 478, no. 7370, p. 476, 2011.

[3] F. Delsuc, H. Brinkmann, and H. Philippe, "Phylogenomics and the reconstruction of the tree of life," *Nature Reviews Genetics*, vol. 6, no. 5, p. 361, 2005.

[4] R. E. Green, E. L. Braun, J. Armstrong, D. Earl, N. Nguyen, G. Hickey, M. W. Vandewege, J. A. S. John, S. Capella-Gutiérrez, T. A. Castoe, *et al.*, "Three crocodilian genomes reveal ancestral patterns of evolution among archosaurs," *Science*, vol. 346, no. 6215, p. 1254449, 2014.

[5] G. K. C. of Scientists, "Genome 10k: a proposal to obtain whole-genome sequence for 10 000 vertebrate species," *Journal of Heredity*, vol. 100, no. 6, pp. 659–674, 2009.

[6] G. Zhang, "Genomics: Bird sequencing project takes off," *Nature*, vol. 522, no. 7554, pp. 34–34, 2015.

[7] H. A. Lewin, G. E. Robinson, W. J. Kress, W. J. Baker, J. Coddington, K. A. Crandall, R. Durbin, S. V. Edwards, F. Forest, M. T. P. Gilbert, *et al.*, "Earth biogenome project: Sequencing life for the future of life," *Proceedings of the National Academy of Sciences*, vol. 115, no. 17, pp. 4325–4333, 2018.

[8] Z. N. Kronenberg, I. T. Fiddes, D. Gordon, S. Murali, S. Cantsilieris, O. S. Meyerson, J. G. Underwood, B. J. Nelson, M. J. Chaisson, M. L. Dougherty, *et al.*, "High-resolution comparative analysis of great ape genomes," *Science*, vol. 360, no. 6393, p. eaar6343, 2018.

[9] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, "Human–mouse alignments with blastz," *Genome research*, vol. 13, no. 1, pp. 103–107, 2003.

[10] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg, "Versatile and open software for comparing large genomes," *Genome biology*, vol. 5, no. 2, p. R12, 2004.

[11] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?," *PLoS biology*, vol. 13, no. 7, p. e1002195, 2015.

[12] D. Karolchik, R. Baertsch, M. Diekhans, T. S. Furey, A. Hinrichs, Y. Lu, K. M. Roskin, M. Schwartz, C. W. Sugnet, D. J. Thomas, *et al.*, "The ucsc genome browser database," *Nucleic acids research*, vol. 31, no. 1, pp. 51–54, 2003.

[13] B. Paten, D. Earl, N. Nguyen, M. Diekhans, D. Zerbino, and D. Haussler, "Cactus: Algorithms for genome multiple sequence alignment," *Genome research*, 2011.

[14] S. L. Clarke, J. E. VanderMeer, A. M. Wenger, B. T. Schaar, N. Ahituv, and G. Bejerano, "Human developmental enhancers conserved between deuterostomes and protostomes," *PLoS genetics*, vol. 8, no. 8, p. e1002852, 2012.

[15] M. Hiller, S. Agarwal, J. H. Notwell, R. Parikh, H. Guturu, A. M. Wenger, and G. Bejerano, "Computational methods to detect conserved non-genic elements in phylogenetically isolated genomes: application to zebrafish," *Nucleic acids research*, vol. 41, no. 15, pp. e151–e151, 2013.

[16] M. C. Frith and L. Noé, "Improved search heuristics find 20 000 new alignments between human and mouse genomes," *Nucleic acids research*, vol. 42, no. 7, pp. e59–e59, 2014.

[17] R. S. Harris, *Improved pairwise alignment of genomic DNA*. PhD Thesis, The Pennsylvania State University, 2007.

[18] V. Sharma and M. Hiller, "Increased alignment sensitivity improves the usage of genome alignments for comparative gene annotation," *Nucleic acids research*, vol. 45, no. 14, pp. 8369–8377, 2017.

[19] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in applied mathematics*, vol. 2, no. 4, pp. 482–489, 1981.

[20] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 199–213, ACM, 2018.

[21] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning dna sequences," *Journal of Computational biology*, vol. 7, no. 1-2, pp. 203–214, 2000.

[22] R. J. Lipton and D. Lopresti, "A systolic array for rapid string comparison," in *Proceedings of the Chapel Hill Conference on VLSI*, pp. 363–376, Chapel Hill NC, 1985.

[23] J. D. Buhler, J. M. Lancaster, A. C. Jacob, R. D. Chamberlain, *et al.*, "Mercury blastn: Faster dna sequence comparison using a streaming hardware architecture," *Proc. of Reconfigurable Systems Summer Institute*, 2007.

[24] R. McMillen and M. Ruehle, "Bioinformatics systems, apparatuses, and methods executed on an integrated circuit processing platform," Apr. 21 2015. US Patent 9,014,989.

[25] T. B. Solisions, "Timelogic decypher blast engine introduction," 2010.

[26] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[27] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[28] C. N. Dewey and L. Pachter, "Evolution at the nucleotide level: the problem of multiple whole-genome alignment," *Human Molecular Genetics*, vol. 15, no. suppl_1, pp. R51–R56, 2006.

[29] B. Rost, "Twilight zone of protein sequence alignments," *Protein engineering*, vol. 12, no. 2, pp. 85–94, 1999.

[30] G. Bejerano, A. C. Siepel, W. J. Kent, and D. Haussler, "Computational screening of conserved genomic dna in search of functional noncoding elements," *Nature Methods*, vol. 2, no. 7, p. 535, 2005.

[31] T. S. Klann, J. B. Black, M. Chellappan, A. Safi, L. Song, I. B. Hilton, G. E. Crawford, T. E. Reddy, and C. A. Gersbach, "Crispr–cas9 epigenome editing enables high-throughput screening for functional regulatory elements in the human genome," *Nature biotechnology*, vol. 35, no. 6, p. 561, 2017.

[32] J. P. Shen, D. Zhao, R. Sasik, J. Luebeck, A. Birmingham, A. Bojorquez-Gomez, K. Licon, K. Klepper, D. Pekin, A. N. Beckett, *et al.*, "Combinatorial crispr–cas9 screens for de novo mapping of genetic interactions," *Nature methods*, vol. 14, no. 6, p. 573, 2017.

[33] M. Blanchette, W. J. Kent, C. Riemer, L. Elnitski, A. F. Smit, K. M. Roskin, R. Baertsch, K. Rosenbloom, H. Clawson, E. D. Green, *et al.*, "Aligning multiple genomic sequences with the threaded blockset aligner," *Genome research*, vol. 14, no. 4, pp. 708–715, 2004.

[34] W. J. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler, "Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes," *Proceedings of the National Academy of Sciences*, vol. 100, no. 20, pp. 11484–11489, 2003.

[35] W. J. Kent, "Blat – the blast-like alignment tool," *Genome research*, vol. 12, no. 4, pp. 656–664, 2002.

[36] A. L. Delcher, S. L. Salzberg, and A. M. Phillippy, "Using mummer to identify similar regions in large sequence sets," *Current protocols in bioinformatics*, no. 1, pp. 10–3, 2003.

[37] B. L. Aken, S. Ayling, D. Barrell, L. Clarke, V. Curwen, S. Fairley, J. Fernandez Banet, K. Billis, C. García Girón, T. Hourlier, *et al.*, "The ensembl gene annotation system," *Database*, vol. 2016, 2016.

[38] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[39] D. Fujiki, A. Subramaniyan, T. Zhang, Y. Zeng, R. Das, D. Blaauw, and S. Narayanasamy, "Genax: a genome sequencing accelerator," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pp. 69–82, IEEE Press, 2018.

[40] M.-C. F. Chang, Y.-T. Chen, J. Cong, P.-T. Huang, C.-L. Kuo, and C. H. Yu, "The smem seeding acceleration for dna sequence alignment," in *Field-Programmable Custom Computing Machines (FCCM), 2016 IEEE 24th Annual International Symposium on*, pp. 32–39, IEEE, 2016.

[41] T. Vijayaraghavan, A. Rajesh, and K. Sankaralingam, "Mpubwm: Accelerating sequence alignment," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 179–182, 2018.

[42] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem," *arXiv preprint arXiv:1303.3997*, 2013.

[43] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, 2018.

[44] T. L. Newman, E. Tuzun, V. A. Morrison, K. E. Hayden, M. Ventura, S. D. McGrath, M. Rocchi, and E. E. Eichler, "A genome-wide survey of structural variation between human and chimpanzee," *Genome research*, vol. 15, no. 10, pp. 1344–1356, 2005.

[45] W. M. Brown, E. M. Prager, A. Wang, and A. C. Wilson, "Mitochondrial dna sequences of primates: tempo and mode of evolution," *Journal of molecular evolution*, vol. 18, no. 4, pp. 225–239, 1982.

[46] K.-M. Chao, W. R. Pearson, and W. Miller, "Aligning two sequences within a specified diagonal band," *Bioinformatics*, vol. 8, no. 5, pp. 481–487, 1992.

[47] S. Meader, L. W. Hillier, D. Locke, C. P. Ponting, and G. Lunter, "Genome assembly quality: assessment and improvement using the neutral indel model," *Genome research*, 2010.

[48] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of molecular biology*, vol. 162, no. 3, pp. 705–708, 1982.

[49] S. Brenner, "The genetics of caenorhabditis elegans," *Genetics*, vol. 77, no. 1, pp. 71–94, 1974.

[50] M. D. Adams, S. E. Celniker, R. A. Holt, C. A. Evans, J. D. Gocayne, P. G. Amanatides, S. E. Scherer, P. W. Li, R. A. Hoskins, R. F. Galle, *et al.*, "The genome sequence of drosophila melanogaster," *Science*, vol. 287, no. 5461, pp. 2185–2195, 2000.

[51] M. J. Hubisz, K. S. Pollard, and A. Siepel, "Phast and rphast: phylogenetic analysis with space/time models," *Briefings in bioinformatics*, vol. 12, no. 1, pp. 41–51, 2010.

[52] SoSy-Lab, "Cpu-energy-meter." https://github.com/sosy-lab/cpu-energy-meter.

[53] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using rapl," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 13–17, 2012.

[54] J. Daily, "Parasail: Simd c library for global, semi-global, and local pairwise sequence alignments," *BMC bioinformatics*, vol. 17, no. 1, p. 81, 2016.

[55] L. Wirbel, "Xilinx sdaccel whitepaper," 2014.

[56] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, "Drampower: Open-source dram power & energy estimation tool," *URL: http://www.drampower.info*, vol. 22, 2012.

[57] Synopsys Inc., "Compiler, design and user, rtl and guide, modeling." http://www.synopsys.com, 2001.

[58] Synopsys IC, "Compiler user guide." http://www.synopsys.com, 2013.

[59] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," tech. rep., Technical Report 2001/2, Compaq Computer Corporation, 2001.

[60] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible dram simulator," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 45–49, 2016.

[61] "Maf format." https://genome.ucsc.edu/FAQ/FAQformat.html.

[62] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler, "The human genome browser at ucsc," *Genome research*, vol. 12, no. 6, pp. 996–1006, 2002.

[63] S. McGinnis and T. L. Madden, "Blast: at the core of a powerful and diverse set of sequence analysis tools," *Nucleic acids research*, vol. 32, no. suppl_2, pp. W20–W25, 2004.

[64] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble, "Meme suite: tools for motif discovery and searching," *Nucleic acids research*, vol. 37, no. suppl_2, pp. W202–W208, 2009.

[65] K. Jabbari and G. Bernardi, "Cytosine methylation and cpg, tpg (cpa) and tpa frequencies," *Gene*, vol. 333, pp. 143–149, 2004.

[66] F. A. Kondrashov, I. B. Rogozin, Y. I. Wolf, and E. V. Koonin, "Selection in the evolution of gene duplications," *Genome biology*, vol. 3, no. 2, pp. research0008–1, 2002.

[67] E. Pennisi, "Sequencing all life captivates biologists," *Science*, vol. 355, no. 6328, pp. 894–895, 2017.

[68] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[69] N. Bray, I. Dubchak, and L. Pachter, "Avid: A global alignment program," *Genome research*, vol. 13, no. 1, pp. 97–102, 2003.

[70] J. S. Kim, D. S. Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, and O. Mutlu, "Grim-filter: Fast seed location filtering in dna read mapping using processing-in-memory technologies," *BMC genomics*, vol. 19, no. 2, p. 89, 2018.

[71] M. Alser, O. Mutlu, and C. Alkan, "Magnet: understanding and improving the accuracy of genome pre-alignment filtering," *arXiv preprint arXiv:1707.01631*, 2017.