BSW: FPGA-Accelerated BLAST-Wrapped Smith-Waterman Aligner

Bryant C. Lam†, Carlo Pascoe†, Scott Schaecher*, Herman Lam†, Alan D. George†

† NSF Center for High-Performance Reconfigurable Computing (CHREC)
Dept. of Electrical and Computer Engineering, University of Florida
Gainesville, FL 32611
{blam, pascoe, hlam, george}@chrec.org

* Monsanto Company
800 North Lindbergh Blvd
St. Louis, MO 63167
scott.r.schaecher@monsanto.com

***Abstract*—The current generation of genome sequencers produces orders of magnitude more sequencing data at a fraction of their former cost, a development that has repositioned the sequencing bottleneck from data acquisition to alignment and analysis. Optimal alignment algorithms, such as Smith-Waterman (SW), provide the most desirable output in terms of sensitivity and accuracy, but are perceived as too computationally demanding for practical use. Practical alignment tools, such as NCBI BLAST, have compensated by using increasingly specialized heuristics to provide faster results at the expense of reduced sensitivity and accuracy. This paper presents an FPGA-based alignment tool that considerably accelerates the SW algorithm with a highly configurable and novel hardware computation engine, while providing a robust software interface that seamlessly integrates into existing BLAST pipelines. This new alignment tool, BLAST-wrapped Smith-Waterman (BSW), improves upon previously published SW-acceleration research by providing a refined tool with a familiar and standard BLAST interface and essential local alignment statistics found in production tools. Experimental evaluation of BSW's performance for several configurations using large, real-world genome datasets shows speedups up to 59 and 7 vs. BLAST (word sizes 7 and 11) and 63 vs. FASTA's SSEARCH on a single Altera Stratix IV E530 FPGA. Scaled speedups exceeding 3300 vs. SSEARCH were observed when executed on up to 64 FPGAs of Novo-G, the reconfigurable supercomputer in the NSF CHREC Center at Florida.**

Keywords*—Smith-Waterman; BLAST; SSEARCH; FPGA; reconfigurable computing; sequence alignment

I. Introduction

In recent years, technological advancement in DNA sequencing has led to an exponential surge in raw sequence data volume and throughput. Current-generation genome sequencers produce orders of magnitude more sequencing data at a fraction of their former cost, a development that has repositioned the sequencing bottleneck from data acquisition to alignment and analysis [1]. With no evidence to show this trend lessening in the near future, solutions aimed at alleviating this bottleneck with emphases on performance, productivity, accuracy, scalability, and/or sustainability become important for the future of personalized genome sequencing and collaborative research efforts. Furthermore, recent solutions [e.g., 2, 3, 4] focus on the development of new, faster, and more specialized alignment algorithms or the acceleration of existing ones, but often overlook the importance of familiarity or ease of integration into higher-order analysis pipelines, which has proven to be a major hurdle for tool adoption. As an attempt to address both issues, this paper presents a new alignment tool, BLAST-wrapped Smith-Waterman (BSW), which features reconfigurable FPGA technology to accelerate the Smith-Waterman (SW) algorithm [5] with a highly parameterized and novel hardware computation engine while also providing a robust software interface that seamlessly integrates into existing processing pipelines that employ NCBI BLAST [6], the most widely used local alignment tool worldwide.

Optimal alignment algorithms, such as SW, provide the most desirable output in terms of sensitivity and accuracy, but are perceived as too computationally demanding for practical use. Practical alignment tools, such as NCBI BLAST, have compensated by using increasingly specialized heuristics to provide faster results at the expense of reduced sensitivity and accuracy. The observed reduction is deemed acceptable in many circumstances, leading to the global adoption of BLAST and solidifying it as the de facto, all-purpose tool for local alignment; however, optimal SW aligners such as FASTA's SSEARCH [7], or the EMBOSS Water tool [8], retain their utility in specialized alignment scenarios when analysis requirements exceed the capabilities of heuristics to fully capture the nuances of underlying data. As costs decrease and genome sequencing becomes more widespread, new tools that attempt to bridge these two competing concerns (performance vs. sensitivity/accuracy) must also address the challenges of productive use, adoption, and integration into existing pipelines and user workflows.

The remainder of the paper is structured as follows. Section II provides a brief overview of the SW and NCBI BLAST algorithms. Section III presents BSW design details focused on its SW hardware core and the robust software architecture supporting it. Compared to previously published SW-acceleration research, BSW provides a refined tool with a familiar and standard BLAST interface and essential local alignment statistics lacking from numerous other accelerators. Compared to BLAST, BSW provides additional functionality such as unrestricted scoring parameters. In Section IV, experimental evaluation of BSW's performance for several configurations on large, real-world genome datasets demonstrates speedups up to 59 and 7 vs. BLAST (word sizes 7 and 11) and 63 vs. SSEARCH on a single Altera Stratix IV E530 FPGA. Additionally in Section IV, the scalability of BSW is showcased with speedups exceeding 3300 vs. SSEARCH on 64 FPGAs when evaluated on Novo-G, the reconfigurable supercomputer in the NSF CHREC Center at Florida [9, 10]. Finally, Section V provides conclusions and directions for future work.

II. Background and Related Work

The foundation of our work relies on previous contributions with the Smith-Waterman algorithm and NCBI BLAST.

A. Smith-Waterman

Given two character sequences, the SW local alignment algorithm [5] searches for the pair of subsequences, one from each, such that no other pair of subsequences possesses greater similarity under the rules of a selected scoring scheme. Implemented with a dynamic-programming algorithm, the scoring criterion incentivizes correctly matched characters while penalizing character mismatches and the insertion/deletion of unmatched, multi-character strings (gaps). Although the algorithm is effective in identifying significant alignments (including those from distant homologies), its software time complexity is *O(mn)*—where *m* is the database length and *n* is the query length—and is infamously slow when processing large sequences, prompting the creation of faster heuristic aligners or the acceleration of existing ones.

Two notable SW software tools are the EMBOSS Water [8] and FASTA's SSEARCH [7] aligners. Water is one of over 200 tools from the EMBOSS suite maintained by the European Bioinformatics Institute (EMBL-EBI). While Water is able to execute up to twice as fast compared to similar tools in the EMBOSS suite, its space complexity is *O(mn)*, limiting its utility to shorter alignments that fit within memory. SSEARCH is part of the FASTA package maintained by the University of Virginia and is optimized with SSE2 instructions and multithreading support to accelerate alignments. In contrast to a majority of other SW tools, SSEARCH provides local alignment statistics to assess the significance of alignments versus a coincidental hit from random chance [11]. While other software SW aligners such as SWIPE [12] have higher performance vs. SSEARCH, SSEARCH remains ubiquitous due to its maturity and is used as one of our software baselines in later sections.

The SW algorithm is highly amenable for hardware acceleration. While GPU implementations typically use vectorization to achieve performance gains [2, 3], one common method observed in FPGA-accelerated SW designs incorporates a systolic-array architecture to perform score calculations in parallel. This parallelization strategy reduces the time complexity of naive software implementations to *O(m+n)* in hardware. Figure 1a presents a general systolic-array architecture for SW. The processing elements (PEs) are loaded with the query sequence, and each PE individually calculates the scores for its specific query character while the reference database is streamed through. Figure 1b depicts this parallel computation in a wavefront manner. While the classic architecture incorporates centralized control logic to manage the hardware, our SW core (based on previous work in [13]) presents the concept of in-stream control where control logic is minimized and distributed to each PE. Software performs control operations on the hardware by embedding control characters within the query/database stream that passes through each PE.

B. NCBI BLAST

Due to SW's slow performance for large datasets, various heuristic-based tools were created to speed up alignment. One popular heuristic, NCBI's BLAST (Basic Local Alignment Search Tool) is able to very quickly determine alignments with some sacrifice to sensitivity and accuracy [6]. BLAST operates by building a set of words based on a specified word size from

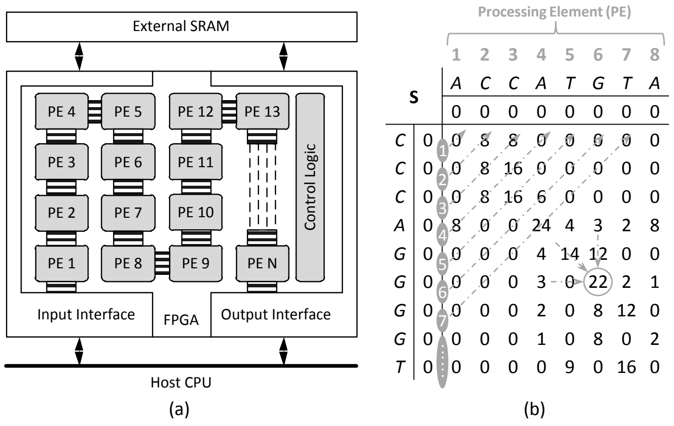

Fig. 1. (a) General SW systolic array, (b) Parallel calculation of scores

the query sequence and quickly searching the reference for exact matches against the set. Exact matches are extended in both directions to determine their candidacy as an alignment. If multiple extended matches are nearby, they may be joined to become a higher scoring alignment. The principle behind this algorithm is that significant alignments between the reference and query tend to have numerous matching characters adjacent to each other; therefore, searching for these strings of matching characters is faster than calculating each character's score. The aforementioned word size determines the minimum string length of matching characters, with the default set at 11. This heuristic strategy improves performance without a substantial hit in sensitivity for sufficiently long references and queries from similar homology, but performance deteriorates rapidly as the heuristics are tuned for higher sensitivity.

The integration of Karlin and Altschul's work on local alignment statistics contributed to the popularity of BLAST. SW reports the optimal alignment's score, but it does not natively provide the means to determine its significance. BLAST determines the significance of an alignment with an E-value based on the lengths of the reference and query, the score of the alignment, and several Karlin-Altschul statistical parameters [11]. The E-value describes the probability that an alignment with similar score will occur by chance in the current reference, decreasing exponentially as the score increases. As E-value approaches zero, the alignment's significance increases. To aid alignment on certain datasets, BLAST can filter results that do not meet an E-value threshold.

III. BLAST-wrapped Smith-Waterman (BSW)

This section introduces the design of our FPGA-accelerated SW local alignment tool, called BSW. The first subsection describes the configurable SW hardware core used for internal query processing. The following subsection describes the supporting software's architecture that provides the BLAST input/output interface and local alignment statistics.

A. Internal Smith-Waterman Core Architecture

The SW core within BSW extends our previous work from [13]. This highly customizable core provides for a variety of configurable parameters including scoring values, linear- or affine-gap models, nucleotide or amino-acid alphabets, character-ambiguity mask support, Altera or Xilinx FPGA targets, integer or fixed-point calculations, single or multiple outputs per query, number of query pipelines per FPGA, option to ex-

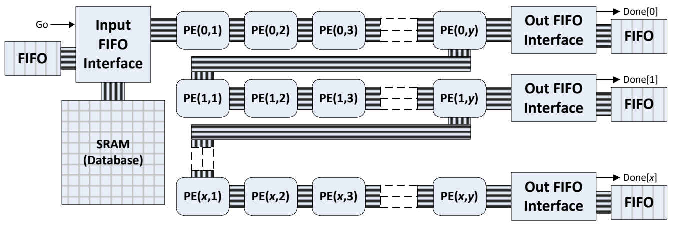

Fig. 2. Smith-Waterman systolic array with query pipelines in BSW

tend queries across multiple FPGAs, and option to record alignment pointers that enable software traceback and alignment statistics. Figure 2 presents the internal systolic-array architecture of BSW.

SW returns the highest similarity score between aligned sequences, but in order to generate the string representation of a calculated alignment, the additional step of tracing back through the dynamic programming matrix from the highest score to a zero termination point is required. This traceback step requires many FPGA resources when naively implemented in hardware, resulting in many previous SW FPGA solutions that completely omit this functionality. BSW takes an alternate approach that minimizes additional hardware requirements without sacrificing functionality. By keeping track of the aligned database subsequence length as well as its termination location in the database, a relatively small, non-minimal bounding box can be placed on the alignment location in the dynamic-programming matrix (a minimal bounding box would require the hardware to keep track of two additional metrics). If the string representation of an alignment is required (determined by the selected output format at runtime), a localized software SW operation with traceback is performed within the bounding box. This software operation requires relatively negligible computation and overlapping traceback with hardware execution yields no additional time penalty. This additional information required to provide traceback support is also used in the calculation of BLAST alignment statistics as well as the logging of multiple outputs per query.

When logging multiple outputs per query, simply reporting the top scoring locations is insufficient, as this would incorrectly identify sub-alignments of the optimal alignment as unique alignments. BSW uses database location and alignment length statistics required in traceback to ensure that only non-overlapping alignments are reported. A limitation to this approach is that only a single subsequence from a query can map to the same subsequence in the database, allowing for the rare loss of sub-optimal alignments only when multiple outputs are enabled.

B. *BSW Software Architecture*

As discussed in the introduction, recently published research solutions on SW acceleration often overlook the importance of familiarity or ease of integration into higher-order analysis pipelines. This oversight has proven to be a major hurdle for tool adoption. The objective of the BSW software modules is to provide a robust software interface that allows BSW to seamlessly integrate into existing BLAST pipelines. The supporting software for BSW consists of three main computational units (shown in Figure 3):

1. *Workers:* synchronize with the FPGA input interface and manage transmission of queries and databases over PCIe.

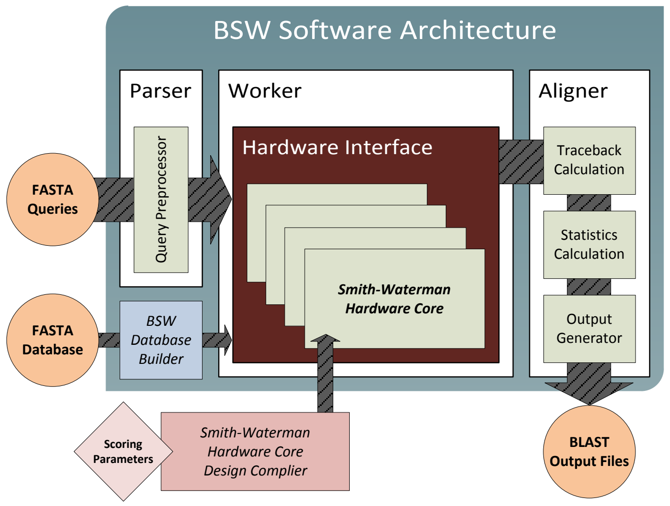

Fig. 3. BSW software architecture with Smith-Waterman core

2. *Parsers:* handle conversion of input sequences into a hardware-compatible format for FPGA processing.
3. *Aligners:* retrieve results from FPGA output interface and perform alignment post-processing such as software traceback, alignment statistics, and output formatting.

A master thread performs initial setup by launching a single worker thread per FPGA, loading the reference database into memory, and loading the specified SW core onto the hardware platform. Each worker thread initializes its own parser and aligner threads to perform input and output processing for its assigned FPGA. Parsers receive a subset of input queries from the main thread and build an input stream for its assigned FPGA. The input stream consists of sequence and pipeline-control characters (21chars/64bits for nucleotide sequences or 12chars/64bits for amino-acid sequences) correctly ordered as to instruct hardware to generate the desired results. Software aligner threads collect and operate on hardware outputs as they become available. Required aligner operations (e.g., additional software traceback, E-value, bit-score, etc.) depend upon the selected BLAST output format. Aligner operations run concurrently with FPGA execution and are hidden behind the typically longer FPGA processing time; however, if the default of one aligner per FPGA cannot process fast enough, the user may specify a runtime option (*-aligners*) to instantiate additional aligners per FPGA. Finally, all aligner outputs are processed through an output generator to create BLAST-compatible output files.

In order to support ease of deployment into existing pipelines, the hardware core is interfaced with a BLAST wrapper with additional hardware-specific runtime options. The majority of runtime options available in BLAST adjust its various heuristics. BSW, however, does not require the majority of these heuristics due to its optimal SW core and safely ignores the appropriate options. Several new runtime options are added to allow users to specify the hardware core to load (raw binary file, *-rbf*), its operating frequency (*-clk*), and system configuration (*-fpgas* and *-boards*).

Included in the BSW package is a database-builder companion application that transforms input FASTA-formatted database files into a custom BSW-compatible database format. All software is implemented with C++11 and leverages several of its new features such as improved threading support.

IV. Experimental Evaluation on Novo-G

In this section, we experimentally evaluate the performance of BSW for single- and multi-FPGA configurations when compared to software tools NCBI BLAST and FASTA's SSEARCH. Alignment accuracy (for optimal alignments) is compared only with SSEARCH. Single-FPGA experiments were performed on a single Altera Stratix IV E530 FPGA on a GiDEL PROCStar IV (PS4) board along with a quad-core Intel Xeon E5620 CPU for host support. Multi-FPGA experiments were performed targeting multiple E530s, in multiple PS4 boards, in multiple PS4 compute nodes of Novo-G, the reconfigurable supercomputer detailed in [9, 10]. Each PS4 node is equipped with four PS4 boards, two E5620 CPUs with 8 MB cache operating at 2.40 GHz, and 32 GB of DDR3 memory. BLAST and SSEARCH baseline performance are measured on these PS4 nodes as well, of course without the aid of FPGAs.

A. Experimental Setup

BSW is best utilized in alignment scenarios where high result sensitivity and accuracy is required on datasets consisting of medium to large databases (e.g., chromosomes) and massive amounts of short- to medium-length queries. Dataset composition of this type maximizes computational parallelization while ensuring DMA transfer overheads and software-based calculations are efficiently hidden behind FPGA processing. To conduct our experiments, datasets and runtime parameters were chosen to illustrate such real-world alignment scenarios, specifically the local alignment of 5' untranslated region (UTR) sequences to whole genome reference assemblies.

Datasets for Experimentation: 5' UTRs are regions of DNA that reside immediately upstream of a protein coding region and are known to play critical roles in post-translational regulation of gene expression due to the presence of numerous regulatory elements [14]. The average length of 5' UTR sequences is relatively consistent across phylogenetically diverse taxonomic classes, with a median length of approximately 150 nucleotides in eukaryotes; however, they can range up several thousand nucleotides in length. In this experiment, we choose two datasets centered on eukaryotic 5' UTR sequences. The first set from *The Arabidopsis Information Resource* (TAIR) consists of the *Arabidopsis thaliana* reference genome and its 5' UTRs [15]. The second dataset consists of the soybean genome assembly *Glycine max* from *Phytozome* [16]. The genome of *G. max*, consisting of 975 Mbp in 20 chromosomes with additional repetitive sequence information in unmapped scaffolds, was selected as a case study for performance with larger databases. Similar to *A. thaliana*, the selected sequences for *G. max* are a subset of the 5' UTR sequences from its genome assembly.

From each of the two chosen datasets for experimentation, a random sampling of 5' UTRs between 128 and 256 bp from all available UTRs is performed to build our query sets. This filtering is done to reduce variance in runtime performance. For both query sets, shorter sequences tend to appear in larger quantities, enabling and encouraging a BSW load-balancing strategy elaborated on in BSW's multi-FPGA analysis. In order to evaluate for potentially real-world evolutionary variance, randomly distributed fixed mutation rates of 0%, 1%, and 10% were applied to the 5' UTR sequences in different experiments.

Parameter Selection: Runtime parameters for BLAST require modification from the defaults in order to suit the shorter length queries in our datasets. Scoring parameters for affine-gap runs are set as *-reward* 1, *-penalty* −3, *-gapopen* 5, and *-gapextend* 2. Linear-gap experiments modify the gap penalties to −6, −6. Soft masking (*-soft_masking*) and filtering (*-dust*) are disabled for these datasets. To account for the possibility of short alignments with high E-values simply due to the increased probability of the alignment occurring randomly, as well as higher mutation rates in the 10% mutation set, the E-value threshold (*-evalue*) is set to 100. UTR region similarity search in a comparative genomics setting would expect higher levels of dissimilarity such that a high E-value is acceptable. Experiments are performed for word sizes (WS) 7 and 11. While WS 7 is considered appropriate for these relatively short sequence datasets, we provide performance results for WS 11 in order to illustrate the significant effect that WS can have on BLAST performance.

For SSEARCH threading is set to 1 to determine baseline characteristics. Trials with SSEARCH at the default of 16 threads shows a sub-linear improvement between 4.6 and 4.9 when compared to single-thread execution (refer to timing justification in appendix).

B. Experimental Evaluation

Tables I and II (appendix) present execution times (in [h:]mm:ss format) for the previously discussed experiments using affine- and linear-gap models for NCBI BLASTn version 2.2.24+, FASTA SSEARCH version 36.3.5a, and BSW on Novo-G. BSW results are presented in several configurations: single Altera Stratix IV E530 FPGA; single PS4 board consisting of 4 FPGAs; single Novo-G PS4 node consisting of 16 FPGAs; and finally, four PS4 nodes with 64 total FPGAs.

Subsets of the appendix table data are plotted in Figures 4 (BSW performance vs. BLAST) and 5 (vs. SSEARCH). The dashed line on each graph at 16 FPGAs denotes the speedup of a *single cluster node* vs. the software baselines, also on one node. Points above 16 FPGAs are shown to illustrate the scalability of BSW, and thus the 64-FPGA configuration is executed on four cluster nodes while the software baselines are still executed (normalized) on only one node.

Mutation Effect on Performance: Execution times from Tables I and II for BLAST show several trends. Most significantly, as the mutation rate increases, run time decreases. This decrease is attributed to the number of alignments identified and reported. With 0% mutation, all UTRs will match identically to the genome assembly they were originally extracted from, but may align to several other locations triggering additional computation. When applying a 1% mutation rate, several of these extraneous alignments begin to differentiate enough such that they are no longer considered significant. With an applied 10% rate, significantly sufficient, randomly distributed, genetic variation reduces the number of relevant alignments and decreases execution time.

In all tested scenarios, the execution times of SSEARCH remain fairly constant with increasing mutation rate. It is only observed with 10,000 queries that higher mutation rates cause a slight performance increase due to slightly fewer detected alignments; however, as SSEARCH still performs a SW alignment between the subject and query, runtimes are roughly estimable. With each magnitude increase in the number of queries, all SSEARCH runs predictably increase by approximately the same magnitude.

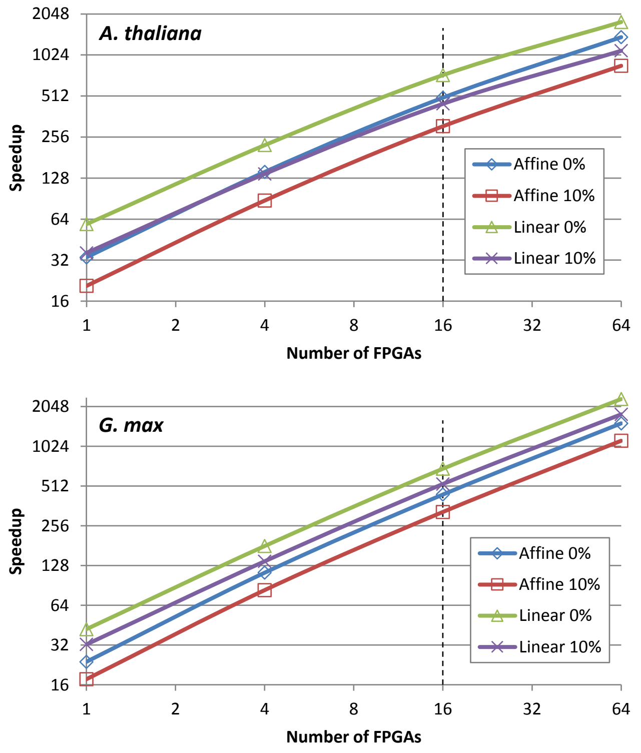

Fig. 4. Speedup of BSW vs. NCBI BLAST (WS 7)

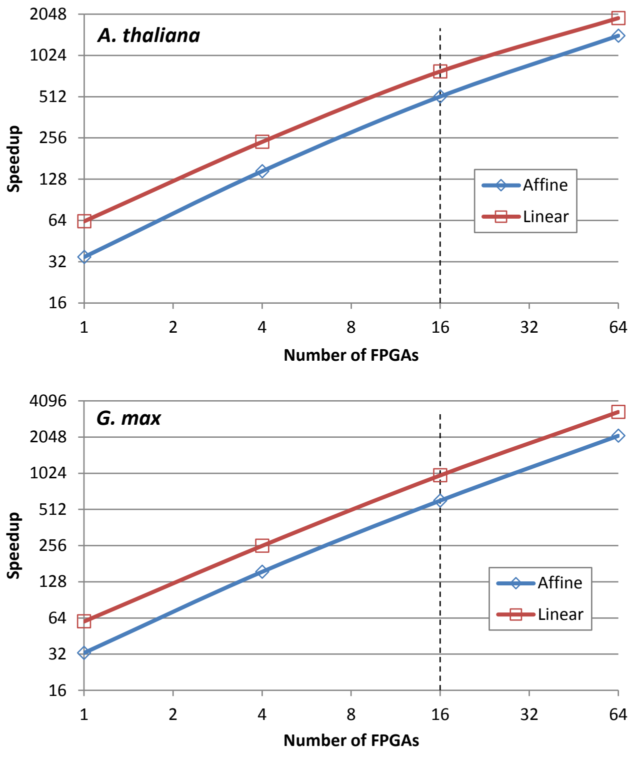

Fig. 5. Speedup of BSW vs. SSEARCH

Single-FPGA Performance: With a single FPGA, BSW performs favorably vs. BLAST and SSEARCH for both genomes.

- For affine-gap experiments, speedup vs. BLAST is 34 and 21 for mutation rates of 0% and 10% with *A. thaliana*, and speedup is 24 and 18 with *G. max*, as shown in Figure 4.
- BSW performance gains vs. SSEARCH are 35 and 33 for *A. thaliana* and *G. max*, as shown in Figure 5.

The SW hardware core in BSW can be configured with either affine-gap or linear-gap hardware. The linear-gap hardware, however, is more area efficient and should be used when datasets are not expected to have significant insertions/deletions such as with these UTRs. The core, when configured for linear-gap, can pack more processing elements onto the device due to the area reduction and can be clocked at a higher frequency for improved performance.

- For linear-gap experiments, single-FPGA speedup vs. BLAST is 59 and 36 for mutation rates of 0% and 10% with *A. thaliana*. Speedup is 42 and 32 with *G. max*.
- Performance improvement of BSW over SSEARCH for linear-gap calculations also experiences a substantial increase to 63 for *A. thaliana* and 60 for *G. max*.

Multi-FPGA Performance: For single-FPGA executions, BSW uses a hardware core with pipeline lengths equal to the longest query length in order to support the entire set of queries. When moving to multiple FPGAs, the same hardware core may be used for all FPGAs, but a different load-balancing strategy is beneficial due to varying query length. A configuration with shorter pipelines limits the max query length a particular core can process, but allows for additional pipelines to be instantiated. With multiple FPGAs, short-length queries can be mapped to FPGAs loaded with short-length pipelines while longer queries can be mapped to FPGAs with longer pipelines. This strategy is used in BSW for all multi-FPGA experiments. With four FPGAs, the query-length range of 128–256 is divided equally into four intervals of 128–160, 161–192, 193–224, and 225–256, with each FPGA assigned an interval. FPGAs with smaller pipelines can replicate more of them and, in the case of affine-gap experiments, the hardware core with 160-length pipelines is successfully replicated 10 times to maximize area utilization whereas the 256-length pipelines can only be replicated 6 times. Maximum clock frequency for these affine-gap cores remains fairly consistent around 180 MHz. Since linear-gap calculations require less hardware, pipelines can be replicated further and clocked higher. The 160-length pipelines are replicated 13 times and the 256-length pipelines are replicated 9 times, averaging a new maximum operating frequency of 220 MHz. By distributing the queries via length onto the appropriate FPGAs, this load-balancing strategy allows us to better utilize the FPGA resources and attain super-linear speedup when going from one to four FPGAs for all 10,000-query experiments compared to a straightforward linear speedup methodology whereby the longest pipeline length is used for all four FPGAs. Scalability beyond four FPGAs is achieved by replicating this configuration to larger granularities: one-node experiments consist of four boards each processing a specific range of lengths and four-node experiments repeat the same with each node processing their range of queries.

- For *A. thaliana*, BSW speedup vs. BLAST at one node consisting of 16 FPGAs is at least 498 for affine-gap 0% mutation and 729 for linear-gap 0% mutation.
- Speedup vs. SSEARCH averages 513 for affine-gap and 782 for linear-gap with *A. thaliana*.
- BSW speedup vs. BLAST for 10,000 queries on the larger *G. max* for four nodes (64 FPGAs) is at least 1530 (affine gap, 0%) and 2337 (linear gap, 0%).

- Speedup vs. SSEARCH averages 2096 and 3308 for affine and linear-gap experiments with *G. max*.

BSW Timing Analysis: Note that BSW execution times are constant regardless of mutation rate and scale linearly with the number of queries divided by the number of pipelines instantiated on the FPGAs. Thus, an equation can be used to estimate the runtime of BSW for large databases:

$$time_{BSW} \approx \left\lceil \frac{queries}{pipelines} \right\rceil \times \frac{size_{db}}{freq_{clk}} + time_{load}$$

In this equation, the sequential load time is a parameter that significantly influences the execution of smaller query sets. BSW consists of two load-time components: database load and FPGA bitfile load. On program startup, the subject database is loaded into memory and is heavily influenced by the size of the reference (e.g., 0.4s for *A. thaliana* vs. 3s for *G. max).* The FPGA bitfile load time is platform dependent (e.g., between 2.3 and 2.7 seconds for PROCStar IVs on Novo-G) and contributes significantly to the increase of BSW's execution times for all *A. thaliana* experiments at 100 queries and below when transitioning from one board to one node (one-node experiments must perform this bitfile load four times sequentially). Even at 10,000 queries, the sequential load time influences up to 50% of *A. thaliana* runtimes.

BSW Dataset Amenability: The experiments conducted illustrate that there are real-world scenarios for sequence alignment that prove unfavorable for heuristics-based tools such as BLAST. Other datasets, however, may be more amenable for these tools than BSW. BSW derives the majority of its performance gains from the parallelization of sequence alignment with multiple queries. The number of query-processing pipelines that can be replicated, however, is directly limited by the area constraints of the FPGAs and the design parameters used for alignment (e.g., scoring criteria, gap model). With BSW's current SW hardware design, pipeline length equates to the maximum sequence length that can be processed; however, a longer pipeline length reduces the design's routability onto the FPGA and decreases the maximum number of PEs that can be routed. As a result, BSW's performance can be increased as pipeline lengths are reduced, allowing more of these shorter pipelines to be successfully replicated. The consequence of this is that BSW is best used to align relatively short-length queries, but its performance with long queries is decreased due to a reduction in the number of parallel processing pipelines.

V. Conclusions

In this paper, we presented BSW, our optimal local alignment tool featuring a novel and efficient SW hardware-accelerated core and robust software architecture with BLAST input/output interfaces and essential alignment statistics. Future work for BSW includes rigorous verification and debugging, additional BLAST option support, and official release.

BSW vs. BLAST: Our experiments show that, for specialized alignment scenarios where high sensitivity is required of BLAST, BSW offers at least comparable if not better performance. And, because of the BLAST-compatible input/output interfaces and alignment statistics, the optimal alignment output of BSW is easily integrated with existing BLAST analysis pipelines. Thus, when sensitivity requirements exceed the capabilities of BLAST to fully capture the nuances of underlying data (i.e., when requirements force the use of tools like SSEARCH over BLAST), the argument for BSW is even stronger; for example, when a desired scoring parameter sensitivity exceeds that which BLAST allows as valid input (e.g., +1/-3 with gap penalty of 12/1).

BSW vs. SSEARCH: In all tested situations, BSW performance exceeds that of SSEARCH. BSW excels at datasets requiring increased sensitivity/accuracy of results and delivers impressive performance gains in all scenarios where a user would require an optimal SW tool such as SSEARCH.

References

[1] M. Gross, "Riding the wave of biological data," in *Current Biology*, Mar 2011, vol. 21, no. 6, pp. R204–R206. doi:10.1016/j.cub.2011.03.009

[2] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions," in *BMC Bioinformatics*, 2013, vol 14, no. 117. doi:10.1186/1471-2105-14-117

[3] S. Lee, C. Lin, and C. L. Hung, "GPU-based cloud service for Smith-Waterman algorithm using frequency distance filtration scheme," in *BioMed Research International*, vol. 2013, Article ID 721738, 8 pages, 2013. doi:10.1155/2013/721738

[4] Y. Chen, "A hybrid short read mapping accelerator," in *BMC Bioinformatics*, 2013, vol. 14, no. 67. doi:10.1186/1471-2105-14-67

[5] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," in *J. Mol. Biol.*, 1981, vol. 147, pp. 195–197. doi:10.1016/0022-2836(81)90087-5

[6] S. F. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," in *Nucl. Acids Res.*, 1997, vol. 25, no. 17, pp. 3389–3402. doi:10.1093/nar/25.17.3389

[7] M. Farrar, "Striped Smith–Waterman speeds database searches six times over other SIMD implementations," in *Bioinformatics*, 2007, vol. 23, no. 2, pp. 156–161. doi:10.1093/bioinformatics/btl582

[8] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: The European Molecular Biology Open Software Suite," in *Trends in Genetics*, 2000 vol. 16, no. 6, pp. 276–277.

[9] A. George, H. Lam, A. Lawande, C. Pascoe, and G. Stitt, "Novo-G: a view at the HPC crossroads for scientific computing," in *Proc. of the Int. Conf. on Eng. of Reconf. Sys. and Algs. (ERSA)*, 2010.

[10] A. George, H. Lam, and G. Stitt, "Novo-G: at the forefront of scalable reconfigurable computing," in *IEEE Computing in Sci. & Eng. (CiSE)*, Jan/Feb 2011, vol. 13, no. 1, pp. 82–86. doi:10.1109/MCSE.2011.11

[11] S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," in *Proc. Natl. Acad. Sci.*, 1990, vol. 87, no. 6, pp. 2264–2268.

[12] T. Rognes, "Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation," in *BMC Bioinformatics*, 2011, vol. 12, no. 221. doi:10.1186/1471-2105-12-221

[13] C. Pascoe, A. Lawande, H. Lam, A. George, Y. Sun, W. Farmerie, and M. Herbordt, "Reconfigurable supercomputing with scalable systolic arrays and in-stream control for wavefront genomics processing," in *Proc. of Symposium on Application Accelerators in High-Performance Computing (SAAHPC)*, Jul 2010.

[14] F. Mignone, C. Gissi, S. Liuni, and G. Pesole, "Untranslated regions of mRNAs," in *Genome Biology*, 2002, vol. 3, no. 3:reviews0004.1.

[15] Carnegie Institution and National Center for Genome Resources, "The Arabidopsis Information Resource (TAIR): a model organism database providing a centralized, curated gateway to Arabidopsis biology, research materials and community," in *Nucl. Acids Res.*, 2003, vol. 31, no. 1, pp. 224–228. doi:10.1093/nar/gkg076

[16] Joint Genome Institute and The Center for Integrative Genomics, "Phytozome: a comparative platform for green plant genomics," in *Nucl. Acids Res.*, 2012, vol. 40, no. D1, pp. D1178–D1186. doi:10.1093/nar/gkr944

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant Nos. EEC-0642422 and IIP-1161022.

APPENDIX

TABLE I. PERFORMANCE OF LOCAL ALIGNMENT FOR 5' UTR SEQUENCES WITH AFFINE GAP PENALTIES OF −5, −2

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (0% mutation)							***Glycine max vs. G. max five prime UTRs (0% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:28	00:36	00:37	00:04	00:04	00:11	00:11	00:19	01:10	04:53	00:15	00:15	00:22	00:21
100	01:27	02:45	06:26	00:15	00:05	00:11	00:11	04:13	13:43	50:06	01:42	00:35	00:22	00:22
1,000	14:02	31:20	1:04:26	01:54	00:28	00:16	00:12	1:54:37	3:49:56	8:13:01	15:13	03:24	01:06	00:23
10,000	2:13:47*	10:22:24*	10:41:24	18:32	04:23	01:15	00:27	9:05:39*	59:54:29*	82:06:15	2:30:25	31:48	08:08	02:21

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (1% mutation)							***Glycine max vs. G. max five prime UTRs (1% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:27	00:35	00:37	00:04	00:04	00:11	00:11	00:19	01:09	04:53	00:15	00:15	00:22	00:21
100	01:22	02:39	06:26	00:15	00:05	00:11	00:11	04:05	13:47	50:06	01:42	00:36	00:22	00:22
1,000	13:05	29:49	1:04:26	01:54	00:28	00:16	00:12	1:43:02	3:34:04	8:12:56	15:13	03:24	01:06	00:23
10,000	2:03:03*	9:54:06*	10:41:24	18:32	04:23	01:15	00:27	8:35:14*	58:33:43*	82:05:58	2:30:25	31:48	08:08	02:22

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (10% mutation)							***Glycine max vs. G. max five prime UTRs (10% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:13	00:20	00:37	00:04	00:04	00:11	00:11	00:13	00:57	04:52	00:15	00:15	00:22	00:21
100	00:45	01:50	06:26	00:15	00:05	00:11	00:11	01:53	10:03	49:58	01:42	00:36	00:22	00:22
1,000	06:32	20:23	1:04:24	01:54	00:28	00:16	00:12	46:20	2:28:57	8:12:33	15:13	03:24	01:07	00:23
10,000	1:11:28	6:23:57	10:40:58	18:32	04:23	01:15	00:27	4:26:34*	44:13:32*	82:03:53	2:30:25	31:48	08:08	02:22

TABLE II. PERFORMANCE OF LOCAL ALIGNMENT FOR 5' UTR SEQUENCES WITH LINEAR GAP PENALTIES OF −6, −6

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (0% mutation)							***Glycine max vs. G. max five prime UTRs (0% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:14	00:21	00:37	00:04	00:04	00:11	00:11	00:18	01:09	04:51	00:13	00:13	00:20	00:21
100	00:43	02:01	06:25	00:09	00:05	00:11	00:11	03:27	13:05	49:48	00:57	00:22	00:21	00:22
1,000	06:46	24:14	1:04:10	01:04	00:18	00:14	00:11	1:06:34	3:07:45	8:10:59	08:20	01:59	00:50	00:23
10,000	47:49*	9:55:33*	10:38:55	10:08	02:40	00:49	00:20	3:14:57*	57:46:50*	81:47:47	1:22:04	19:16	05:00	01:29

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (1% mutation)							***Glycine max vs. G. max five prime UTRs (1% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:14	00:21	00:37	00:04	00:04	00:11	00:11	00:17	01:08	04:51	00:13	00:13	00:21	00:21
100	00:40	01:57	06:25	00:09	00:05	00:11	00:11	03:20	12:55	49:48	00:57	00:22	00:21	00:22
1,000	06:15	23:04	1:04:11	01:04	00:18	00:14	00:11	1:01:10	2:57:11	8:10:59	08:20	01:59	00:49	00:22
10,000	44:33*	9:31:11*	10:38:54	10:08	02:40	00:49	00:20	3:03:56*	56:33:26*	81:47:40	1:22:04	19:15	05:00	01:29

	Arabidopsis thaliana vs. A. thaliana five prime UTRs (10% mutation)							***Glycine max vs. G. max five prime UTRs (10% mutation)***						
No. of Queries	BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration				BLAST word size 11	BLAST word size 7	SSEARCH	BSW FPGA Configuration			
				1	4	4×4	4×4×4				1	4	4×4	4×4×4
10	00:06	00:13	00:37	00:04	00:04	00:11	00:11	00:12	00:57	04:51	00:13	00:13	00:21	00:21
100	00:22	01:28	06:24	00:09	00:05	00:11	00:11	01:42	10:03	49:46	00:57	00:22	00:21	00:22
1,000	03:10	17:23	1:04:12	01:04	00:18	00:14	00:11	30:32	2:19:00	8:10:47	08:20	01:59	00:50	00:23
10,000	35:15	6:06:23	10:38:47	10:08	02:40	00:49	00:20	1:37:27*	44:15:11*	81:45:54	1:22:05	19:15	05:00	01:29

Scoring parameters: +1, −3 with gap penalties of −5, −2 for affine-gap experiments and −6, −6 for linear-gap experiments
BSW FPGA configurations: FPGA (1), board (4), node (4×4), four nodes (4×4×4)
** Program execution killed by operating system due to excessive memory use (>32GB).*
BLAST execution times are heavily affected by heuristic parameter settings. The cumulative effect on execution time of deviating from each parameter's default setting is presented here. Baseline execution time is 16:12 (A. thaliana, 1000 queries, 0% mutation, word size 7). Increasing E-value threshold to 100 increases runtime penalty by 2 (32:01). Disabling masking and filtering increases runtime penalty by 3 (1:27:45). Total performance penalty of these parameters is approximately 6 from the defaults.
For A. thaliana UTRs (0% mutation, affine gap), SSEARCH with its default of 16 threads finished execution in 00:08, 01:18, and 13:00 for 10, 100, and 1000 queries, a sub-linear improvement between 4.6 and 4.9 when compared to single-thread execution. Presented results can be scaled appropriately to account for multi-threading in SSEARCH.
Example runtime commands for affine-gap experiments with Arabidopsis thaliana:

```
bsw -rbf 2x6x256_180MHz.rbf -clk 180 -db Arabidopsis.bswdb -query TAIR10_five_prime_UTRs.fasta
blastn -evalue 100 -word_size 7 -reward 1 -penalty -3 -gapopen 5 -gapextend 2 -soft_masking false -dust no -db Arabidopsis -query TAIR10_five_prime_UTRs.fasta
ssearch36 -E 100 -3 -b 1 -d 1 -r +1/-3 -f -3 -g -2 -T 1 TAIR10_five_prime_UTRs.fasta Arabidopsis.fa
```