Hardware Accelerators for Biocomputing: A Survey

Souradip Sarkar, Turbo Majumder, Ananth Kalyanaraman, Partha Pratim Pande
School Of Electrical Engineering and Computer Science
Washington State University, Pullman, USA
{ssarkar, tmajumde, ananth, pande}@eecs.wsu.edu

***Abstract*— Computing research has become a vital cog in the machinery required to drive biological discovery. Computing has made possible significant achievements over the last decade, especially in the genomics sector. An emerging area is the investigation of hardware accelerators for speeding up the massive scale of computation needed in large-scale biocomputing applications. Various hardware platforms, such as FPGA, Graphics Processing Unit (GPU), the Cell Broadband Engine (CBE) and multi-core processors are being explored. In this paper, we present a survey of hardware accelerators for biocomputing by choosing a representative set of each.**

I. Introduction

The role of computing in molecular biology research has never been more defining. Data processing for biocomputing applications is currently done in software, which often takes a very long time. Aligning even a few hundred sequences using progressive multiple alignment tools consumes several CPU hours on state-of-the-art workstations. Large-scale sequence analysis, often involving up to tens of millions of sequences, has become a mainstay as well as one of the primary bottlenecks in the path to scientific discovery. The molecular biocomputing domain also hosts a set of compute-intensive applications wherein the underlying problems are proven to be computationally intractable (e.g. phylogenetic tree computation, protein folding). These aspects collectively make this an application domain that has the potential to immensely benefit from the incorporation of the latest advancements in circuit design and evolving hardware architectures. Several hardware accelerators have been proposed recently. FPGA-based reconfigurable hardware platforms, GPUs, CBEs, general purpose multi-core processors and Network-on-Chip (NoC) platforms are used as hardware accelerators for biocomputing. A successful solution will adopt and encompass elements from several such approaches. The challenge of designing efficient hardware accelerators for biocomputing research is actively being pursued by a number of researchers worldwide, and from a variety of different perspectives. Figure 1 summarizes the current state of the art. In this paper, we present an overview of the characteristics of hardware accelerators for a representative set of biocomputing applications at the molecular level.

II. Sequence Analysis

Sequence homology detection (or sequence alignment) is a pervasive compute operation carried out in almost all bioinformatics sequence analysis (SA) applications. Due to exponentially growing sequence databases, computing this operation at a large scale is becoming prohibitive. The

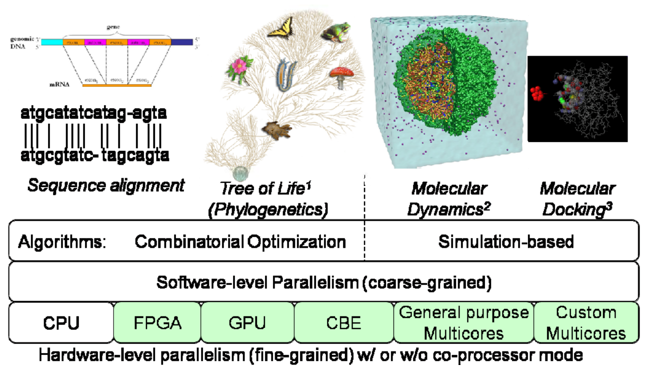
atgcatatcatag-agta
atgcgtatc-tagcagta
Sequence alignment
Tree of Life[1] (Phylogenetics)
Molecular Dynamics[2]
Molecular Docking[3]
Algorithms:
Combinatorial Optimization
Simulation-based
Software-level Parallelism (coarse-grained)
CPU
FPGA
GPU
CBE
General purpose Multicores
Custom Multicores
Hardware-level parallelism (fine-grained) w/ or w/o co-processor mode

Figure 1: Biocomputing applications benefiting from hardware acceleration.

operation can be carried out in three modes: as one-to-one, one-to-many or many-to-many comparisons. The one-to-one alignment operation, called the pairwise sequence alignment (PSA), is used to compute an optimal edit distance between two sequences, after taking into account evolutionary manifestations of mutation such as substitution, insertion and deletion. In the one-to-many comparison model, a query sequence is searched against a database of sequences. The BLAST tool [1] is an example of this class. In the many-to-many comparison model, multiple sequences are analyzed collectively for the purpose of identifying sub-groups of sequences that share a common characteristic such as homology. This operation is often implemented using an all-against-all sequence comparison strategy. Multiple sequence alignment (MSA) is an example of this class. In all of the above, a sequence can be either a DNA or a protein, and the bulk of the computations involve integer arithmetic.

Algorithmically, computing an optimal PSA between two sequences of lengths m and n respectively, can be achieved using dynamic programming (DP) in $O(mn)$ time and $O(m+n)$ space [2]. The algorithm computes an $(m+1)*(n+1)$ table in two passes. In the forward pass, the table is computed from cell $(0, 0)$ to cell (m, n), wherein a recurrence is applied at every cell (i,j) based on the values at cells $(i-1,j)$, $(i-1,j-1)$ and $(i,j-1)$. The main challenge in the backward pass is to be able to retrace without storing the DP table that was computed during the forward pass. Several coarse-grain parallel algorithms have been developed. Huang [3] uses a wavefront technique whereby the cells along each anti-diagonal are computed in each parallel time step. Aluru et al. [4] introduced another technique in which cells along each row can be computed in each time step using the parallel prefix technique.

Hardware accelerators provide fine-grain parallelism. FPGA-, GPU- and CBE-based implementations primarily

This work is partially supported by NSF grant (IIS-0916463)

[1] http://www.tolweb.org/tree/
[2] http://www.ks.uiuc.edu/Research/STMV/
[3] http://wwwcs.uni-paderborn.de/~lst/HotDock/

rely on software and use pre-existing hardware platforms to map algorithms. Such generic hardware-based systems can be used for multiple applications with only software modifications.

Hardware accelerators using FPGA have been developed for implementing ClustalW [5], which is a popular MSA program. Since the underlying problem is NP-Hard, ClustalW approximates a solution in polynomial time. A *K*-sequence MSA problem involves computing ${}^{K}C_{2}$ PSA comparisons. This all-against-all sequence comparison is the dominant phase within ClustalW, taking more than 90% of the total time. The FPGA implementation uses Xilinx Virtex II XC2V6000, platform accommodating 92 processing elements (PEs) with a maximum clock speed of 34 MHz. This gives a speedup of around 10 for the overall MSA and about 50 for PSA. It achieves a sustained performance (including all data transfer) of ~1 GCUPS (billion cell updates per second in the DP matrix).

The sequence search tool BLAST proceeds by first identifying a subset of database sequences that have short matching segments with the query sequence and then performing a more thorough evaluation of the query against each short-listed candidate. The filtering step is implemented using a lookup table data structure, and a subsequent evaluation as a unit PSA. Sachdeva et al. [6] implemented BLAST on CBE, consisting of a 64 bit Power Processor Element (PPE) and eight Synergistic Processing Elements (SPEs). It achieves a speedup of 2 compared to that implemented on a single Power PC processor. The FPGA BLAST [7] is implemented on Annapolis Microsystems WildstarII-Pro board with two Xilinx Virtex-II FPGAs. The authors have implemented two FPGA BLAST algorithms, namely the TREE BLAST and the SERVER BLAST. The notion behind the former algorithm is that, it can be executed with iterative merging using a tree structure. The FPGA is initialized with the query sequence and the scoring matrix. The indexing of the scoring arrays is done using the block RAMs (BRAMs). The database is streamed from the memory through the FPGA. The main component of SERVER BLAST is a systolic array that holds a query string while the database flows through it. This is implemented using a FIFO buffer in FPGA. The performance reported was comparable to that of a dedicated server at National Center for Biotechnology Information. Liu et al. [8] demonstrates about 16 fold speedup over OSEARCH, which is an MSA tool using nVidia GeForce 7800 GTX GPU. Mapping of the algorithm onto GPU is done exploiting the fact that all elements in the same anti-diagonal of the DP matrix can be computed independent of each other in parallel. Fragment programs are used to implement the arithmetic operations specified by the recurrence relation. They have reformulated the Smith-Waterman algorithm [2] in terms of computer graphics primitives, in an attempt to exploit the GPU platform for optimum performance.

A simple search operation against a database of known sequences involves computing an unprecedented number of sequence alignment operations. An effective way to address this would be to integrate huge number of PEs on a single chip for exploiting the massive scale of fine-grain parallelism inherent in bioinformatics applications. Systems based on multiple cores on a single chip are emerging as a viable alternative for accelerating various scientific applications. Network-on-Chip (NoC) is viewed as an enabling methodology to obtain such high degree of integration in a single chip. An NoC-based implementation of PSA [9] reports significant improvement over other hardware accelerators because of its custom made architecture and interconnection topology. The result of using 64 PEs in such a system achieved two to three orders of magnitude better performance compared to other existing hardware accelerators. NoCs also provide the freedom to design and experiment with different network topologies and their suitability to different algorithmic settings. Table 1 summarizes the performances of various hardware accelerators designed for SA.

TABLE 1: OVERVIEW OF SPEEDUPS ACHIEVED FOR SEQUENCE ANALYSIS UNDER DIFFERENT HARDWARE ACCELERATOR MODELS ([5][6][8][9])

Type of SA	**Speedup over Serial Implementation**			
	FPGA	*GPU*	*CBE*	*Multi-core (NoC)*
PSA	100	70	6	22,000
MSA	13	7	2	N.A.
BLAST	N.A.	16	2	N.A.

III. PHYLOGENETICS

While SA represents a data-intensive application class in biocomputing, phylogenetics represents a compute-intensive class of applications. In phylogenetics research, the primary goal is to reconstruct evolutionary trees that best describe the evolutionary relationship among different species, by observing and characterizing variations at the DNA and protein level. The "Tree of Life" is an example of an ambitious project for inferring the phylogeny linking all known life forms. Typical probability models of evolution used for this purpose are Jukes-Cantor (JC) and General Time Reversible (GTR). Unlike SA, the computational intractability of the problem is the primary stumbling block to advance the state of research in phylogenetic inference, as the underlying problems have been proven to be NP-Hard under various formulations [10]. The choice amongst three main strategies - neighbor-joining, maximum parsimony (MP) and maximum likelihood (ML) - often depends on the nature of the problem at hand. The following discussion covers different hardware accelerators for MP and ML in the order of increasing problem complexity. Most of the work addresses ML, which is statistically the most accurate and computationally the most intense of the strategies, involving numerous floating point computations for evaluating the phylogenetic likelihood function (PLF).

Mak and Lam [11] proposed a hybrid hardware/software system for solving the phylogenetic tree reconstruction using the Genetic Algorithm for Maximum Likelihood (GAML) approach. The genetic algorithm is implemented in software and the computationally intensive ML equation is implemented in hardware. This work uses a Xilinx Virtex XCV800 FPGA as the hardware accelerator and a Pentium 4 PC with 1 GB RAM for running the software. The likelihood function is evaluated in parallel in the dedicated FPGA. Their results while reconstructing a 4-taxa phylogenetic tree under the JC Model demonstrate an overall speedup of 30 over software and an ML speedup of over 300, despite the communication overhead of the hybrid system. This work however does not explicitly state how the acceleration scales for larger taxa or more realistic complex models like GTR.

Alachiotis et al. explored the use of FPGA for accelerating the computation of PLF in [12]. A Xilinx Virtex 5 SX240T with 1056 DSP48E slices has been used. The DSP slices have been used to implement double-precision floating point multipliers and adders. Due to the limited amount of DSP48E slices on the FPGA, several multiplexer units are deployed to optimally exploit the available computational resources. A Sun x4600 system equipped with 8 dual-core AMD Opteron processors running at 2.6 GHz with 64 GB of main memory was used as the baseline. An average speedup of 8.3 over a single core has been demonstrated for trees comprising of 4 to 512 sequences on FPGA. The FPGA implementation also outperforms OpenMP-based parallel implementation on 16 cores in most cases, achieving speedups from 0.96 to 7.46. The projected computational time for a full tree traversal using Felsenstein's pruning algorithm for 512 taxa is less than 1 ms, based on reported clock speed of 284.152 MHz.

Bakos and Elenis [13] proposed a co-processor design for whole-genome phylogenetic reconstruction using a parallelized version of breakpoint median computation, which is an expensive component of the MP phylogenetic tree inference. The co-processor uses an FPGA-based multi-core implementation of the combinatorial search portion of the Travelling Salesman Problem (TSP) algorithm while the TSP graph construction is performed in software. The search tree partitioning is carried out in such a manner that each core explores the tree in a different order. This is done to avoid complex load-balancing and inter-core communication issues that occur if disjoint subtrees are assigned to different cores, because any of them might be subject to pruning. Their test system consists of 3.06-GHz Intel Pentium Xeon processor and a single XilinxVirtex-2 Pro 100 FPGA connected to the host using a PCI-X interconnect. The best average speedup of 1,005 over software is observed using 3 cores. The best overall reduction in execution time is by a factor of 417. All these observations are for synthetic data and hence difficult to correlate with real-life examples.

Randomized Axelerated Maximum Likelihood version VI for High Performance Computing (RAxML-VI-HPC) is an efficient parallel algorithm based on ML for phylogenetic tree inference. Blagojevic et al. have explored the porting, optimization and evaluation of RAxML-VI-HPC on CBE [14]. They carry out a detailed empirical optimization of RAxML on CBE, with additional support from the runtime environment. Different layers of parallelism have been used – task-level parallelism across SPEs, task vectorization within SPEs and/or loop-level parallelization across SPEs. It is shown that CBE outperforms both Intel Xeon and IBM Power5 and is more cost-effective and power-efficient than either architecture. However, the sheer complexity of porting the algorithm and the various optimizations required for CBE collectively pose a significant roadblock.

In [15], MrBayes, a program for Bayesian inference of phylogenetic trees has been used on three different architectures to evaluate performance, scalability and programmability. General purpose multi-core (dual-core and quad-core Intel and AMD) processors and CBE support the Multiple Program Multiple Data (MPMD) model while GPUs support Single Program Multiple Data (SPMD) model. The PLF in MrBayes is parallelized using OpenMP directives for the general-purpose multiprocessors, POSIX threads for the CBE systems and Compute Unified Device Architecture (CUDA) for the GPU systems. For hardware-managed caches, the sharing of a cache level within the chip by all cores is a determining factor for efficient synchronization and hence scalability. Systems with software-managed caches like CBE compensate the user effort by efficient synchronization mechanisms. On the other hand, there are fewer data transfers between the device memory and CPU because GPU has sufficient memory to handle input data. CUDA automatically handles data transfer synchronization, thus relieving the user of the responsibility of providing any explicit synchronization mechanism. PLF computation speedup is penalized by computation intensity and communication overhead inside the multi-cores. Quad-core AMD Opteron, where four cores are on a single die and share the same L2 cache, scales better compared to quad-core Intel Xeon, which has two L2 caches each shared by a pair of cores. For CBE, speedup values are close to ideal for small data sets and performance is stable across different computation intensities. Even though SPEs do not share a common cache, CBE is more tolerant to synchronization, primarily because it relies on user-generated software for this. However, speedup values for large data sets and computation intensities are almost equal for general-purpose multi-cores and CBEs. GPUs display an increase in speedup as the computation intensity increases because they are designed to perform efficient execution of small parallel threads in a scenario where the computation-to-data ratio is high. In terms of total frequency-normalized execution times, the general-purpose multi-core still achieves the best performance. This is based on the sum of the time spent in executing the parallel portion of the code (PLF) and that for the rest of the code. The degradation in total execution time for CBE is due to the fact that the PPE that handles the serial portion of the code is a rather simple core with a small cache, in-order execution capability and is burdened with the additional responsibility of synchronizing among SPEs. Table 2

TABLE 2: PERFORMANCE COMPARISON FOR PHYLOGENETIC INFERENCE ([11][12][13][15])

Phylogenetic tree reconstruction strategies	**FPGA**		**GPU**		**Cell Broadband Engine**		**General Purpose Multi-core**	
	Application speedup	*Total speedup*	*Application speedup*	*Total speedup*	*Application speedup*	*Total speedup*	*Application speedup*	*Total speedup*
Maximum parsimony (MP)	1005	417	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Maximum likelihood (ML)	381	32, 13.68	8.5	1.9	12	1.5	12	10

summarizes the speedups achieved by different hardware accelerators for the MP/ML (application speedup) computation and the overall algorithm (Total speedup).

IV. Simulation-based Applications

Molecular Dynamics (MD) is a widely used technique for studying the structural and functional characteristics of biomolecules. It is a particular case of an N-body problem where hundreds of thousands of atoms of a molecular complex such as a protein are allowed to interact over a fixed spatial dimension, for over millions of discrete time steps. As such, the problem is heavily compute cycle bound. In [16], the use of CUDA-based GPU implementation for MD applications is reported to have achieved speedups of around 20 compared to the regular CPU cores. Chiu et al [17] report 80-fold speedup over the previous GPU based design, using two Altera Stratix-III FPGAs. Design of a massively parallel machine, ANTON [18] for MD computation is particularly notable. This design consists of 512 identical MD-specific ASICs that interact in a high-speed NoC. It is expected to achieve a speedup of about 100 over the IBM Blue Gene supercomputer. Molecular Docking is another simulation class application that is heavily used to study the binding orientation of small molecule drug candidates to their protein drug targets. Due to its high computational complexity, dedicated hardware platforms have been considered for docking. The FPGA [19] and GPU [20] based prototype implementations provide significant speed up over standard serial implementation.

V. Comparative Discussion

GPU and CBE provide multi-core platforms where applications with sufficient scope for parallelization can be ported. From the programmability perspective, GPUs are easier for implementation than CBEs, which require the user to write elaborate synchronization software. Furthermore, due to the inherent architectural properties of a GPU, it achieves comparable or higher speedups than a CBE-based platform for compute-intensive applications. On the flip side, the GPU-based solution spends a significant fraction of its time in CPU-GPU communication through PCI-X. General purpose multi-core platforms provide the best overall speedup and also provide the maximum ease of porting of code, while the effort required for CBE is maximum. Due to their reconfigurable nature, FPGA platforms provide the scope for implementing parallel architectures specifically optimized to solve SA or phylogenetic inference. The advantage of NoC-based multi-core architectures is that both computation and intra-chip communication can be tailor-made for the application, providing orders of magnitude performance improvement.

The field of bioinformatics and computational biology is host to applications that combine a number of desirable attributes – emerging importance, high computational complexity, inherent data-parallelism, and well-defined communication and computation patterns – soliciting the design, development and application of novel hardware accelerators. While all the hardware models studied so far have shown significant levels of improvement over single CPU implementations, we posit that multi-core platforms, both general purpose and NoC-based, hold the most promise for practical adoption owing to their ease of implementation, projected scalability and/or provision for higher degrees of design flexibility.

VI. References

[1] S. Altschul et. al., "Basic Local Alignment Search Tool", *Journal of Molecular Biology*, 1990, 215, pp. 403-410.

[2] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences," *J. Molecular Biology*, 1981, vol. 147, pp. 195-197.

[3] X. Huang, "A space-efficient parallel sequence comparison algorithm for a message-passing multiprocessor", *International Journal of Parallel Programming*, 1989, 18(3): pp. 223–239.

[4] S. Aluru et al., "Parallel biological sequence comparison using prefix computations", *.l Parallel and Distributed Computing*, 2003, 63: pp.264–272.

[5] T. Oliver et. al., "Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW", *Bioinformatics*, 2005, vol. 21(16): pp. 3431-3432.

[6] V. Sachdeva et. al., "Exploring the Viability of the Cell Broadband Engine for Bioinformatics Applications", *Parallel and Distributed Processing Symposium*, 2007, pp. 1-8.

[7] M. Herbordt et. al., "Single Pass, BLAST-Like, Approximate String Matching on FPGAs", *Proc. 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2006, pp. 217-226.

[8] W. Liu et al, "Streaming Algorithms for Biological Sequence Alignment on GPUs", *IEEE Trans. on Parallel and Distributed Systems*, 2007, vol. 18, No. 9, pp. 1270-1281.

[9] S. Sarkar, et. al, "Network-on-Chip Hardware Accelerators for Biological Sequence Alignment" *IEEE Trans. Comp.* (Accepted April 2009).

[10] B. Chor and T. Tuller, "Maximum Likelihood of Evolutionary Trees: Hardness and Approximation," *Bioinformatics*, 2005, vol. 21, (1), pp. 97-106.

[11] T. S. T. Mak and K. P. Lam, "High Speed GAML-based Phylogenetic Tree Reconstruction Using HW/SW Codesign," *Proc. Computational Systems Bioinformatics*, 2003, pp. 470.

[12] N. Alachiotis et. al., "Exploring FPGAs for Accelerating the Phylogenetic Likelihood Function," Proc. IEEE International Symposium on Parallel and Distributed Processing 2009, pp 1-8.

[13] J. D. Bakos and P. E. Elenis, "A Special-Purpose Architecture for Solving the Breakpoint Median Problem," *IEEE Trans. VLSI Systems*, Vol. 16, No. 12, Dec 2008, pp. 1666-1676.

[14] F. Blagojevic et. al., "RAxML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine," *Proc. IEEE International Symposium on Parallel and Distributed Processing* 2007, pp. 1-10.

[15] F. Patas et. al., "Fine-grain Parallelism using Multi-core, Cell/BE, and GPU Systems: Accelerating the Phylogenetic Likelihood Function," *Int. Conf. Parallel Processing,* 2009, pp. 9-17.

[16] J. Stone et. al., "Accelerating molecular modeling applications with graphics processors". *J. Computational Chemistry*, 2007, vol. 28, pp. 2618–2640.

[17] M. Chiu and M. C. Herbordt, "Molecular Dynamics Simulations on High Performance Reconfigurable Computing Systems", *ACM Trans. on Reconfigurable Technology and Systems*, 2010 (Accepted for publication).

[18] D. E. Shaw et. al., "Anton: A Special-Purpose Machine for Molecular Dynamics Simulation," *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*, San Diego, California, June 9–13, 2007, pp.1-12.

[19] B. Sukhwani and M.C. Herbordt, "FPGA Acceleration of Rigid-Molecule Docking Codes", IET Computers & Digital Techniques, 2009 (In Press).

[20] B. Sukhwani and M.C. Herbordt, "GPU Acceleration of a Production Molecular Docking Code"*, Proc. of Workshop on GPGPU*, 2009, pp. 19-27.