On-Chip Network-Enabled Many-Core Architectures for Computational Biology Applications

Turbo Majumder
Department of Electrical Engineering
Indian Institute of Technology Delhi
New Delhi, India
turbo@ee.iitd.ac.in

Partha Pratim Pande, Ananth Kalyanaraman
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA, USA
{pande, ananth}@eecs.wsu.edu

Abstract—**Computational molecular biology applications are at the heart of the backend processing in cyber-physical systems when applied to domains such as drug discovery, personalized medicine and genetic disease risk assessment. These applications are characterized by the preponderance of data and computational complexity, and yet require reasonably fast processing in order to have any meaningful impact. As such, hardware acceleration for these applications have generated a lot of research interest. In this paper, we discuss the superiority of Network-on-Chip (NoC)-enabled many-core platforms over other conventional platforms in both the quantum of speedup achieved and the amount of energy consumed. We hence posit that research in NoC-enabled platforms for CPS applications will be a major enabler of future scientific and medical breakthroughs.**

I. Introduction

The role of computing platforms in a biomedical cyber-physical system (CPS) is to model and analyze the dynamics of various biomarkers as well as the interactions between biological entities and therapeutic agents in order to assess the efficacy and success of various medical treatments. In order to accelerate discoveries in biology and bioinformatics, which will provide contextual information in support of more effective data analytics, novel architectures based on Network-on-Chip (NoC) platforms are needed to efficiently exploit massive scales of fine-grain parallelism inherent in these applications. We are going through a time of great churn in genetics that may dramatically impact human biology and medicine. The completion of the human genome project, the development of low-cost, high-throughput parallel sequencing technology, and large-scale studies of genetic variation have contributed to a rich set of techniques and data for the study of genetic disease risk, treatment response, population diversity, and human evolution. In this context, we consider computational molecular biology, which includes important applications such as genome discovery, phylogenetic inference, assessing disease vulnerability of individuals, drug efficacy, etc. There is an ever-increasing throughput demand driven from two angles – volume of data and intensity of computation – with different applications exhibiting one or both of these characteristics in varying degrees. In this paper, we discuss how NoC-based platforms can be architected to meet these requirements of a CPS. Through this, it is our aim to establish on-chip networked many-core architectures as the paradigm of choice for backend data analysis in real-world biomedical CPS applications.

II. Application Perspective and Related Work

Computational biology applications offer a wide range of computational challenges to the designer of the hardware accelerator. While some applications are characterized by the preponderance and volume of data, some others are combinatorial optimization problems over multidimensional real space, yet others involve billions of iterations of complex simulations. Architecting a suitable accelerator for a class of applications needs to take into account these characteristics. As described in the previous section, several hardware platforms have been proposed as enablers for such applications, but NoC offers the greatest degree of customizability and hence energy and throughput performance while accelerating many computational biology applications. In the following we will study two broad classes of such applications – first, genomic sequence analysis, which is characterized by a large volume of input data that need to be processed very fast, and second, phylogeny reconstruction, different flavors of which present NP-complete optimization problems in integer and real number space.

A. Sequence Analysis

Sequence analysis encompasses a wide range of applications, such as pairwise sequence alignment (PSA), multiple sequence alignment (MSA), genome assembly, read mapping, sequence profiling, etc. The primary challenge faced by these applications is the sheer volume of sequence data that may consist of DNA, RNA or protein sequences. They key operation in PSA involves one-to-one comparison of two such sequences, and identifying and aligning subsequences that match in both sequences with the addition of "gaps" if necessary [1]. Note that this process needs to take into account evolutionary mutations like substitution, insertion and deletion. In MSA, the goal is to create (possibly overlapping) classes of homologous sequences through an all-to-all comparison among all sequences. Since these sequences are represented as strings based on a certain alphabet depending on the type of sequence being compared (e.g., {A,C,T,G,-} for DNA sequences), all computations are based on integer arithmetic.

The dynamic programming (DP) algorithm for computing an optimal PSA between two sequences of lengths m and n respectively has a time complexity of $O(mn)$ and space complexity of $O(m+n)$ [1]. This is a two-pass algorithm in

This work was supported in part by the US National Science Foundation (NSF) grants CCF-0845504, CNS-1059289, and CCF-1162202, and Army Research Office grant W911NF-12-1-0373. TM acknowledges support by Govt. of India DST SERB Grant No. SERB/F/1512/2014-15.

978-3-9815370-4-8/DATE15/©2015 EDAA

which the forward pass is used to compute a recurrence relation from cell *(0,0)* to cell *(m,n)*. 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, and hence maintain the *O(m+n)* space complexity. 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 concurrently using the *parallel prefix* algorithm developed in [5]. Note that both these techniques lead to a fairly regular partitioning of the original problem.

MSA is inherently a harder problem compared to PSA because the solution for *K* input sequences involves ${}^{K}C_{2}$ PSA steps. In fact, since the problem is NP-Hard, there are approximate algorithms or heuristics that are used to solve it in polynomial time. The run-time of ClustalW [2], which is a popular MSA program, is dominated by all-to-all sequence comparisons that take more than 90% of the total time. This portion has been shown to be accelerated by 50x on an FPGA platform [2]. The above method achieves ~1 GCUPS (billion cell updates per second in the DP matrix). Another sequence search tool BLAST follows a two-stage process – (a) identification of a short-sequence homologous subset, and then (b) comparison of the query sequence with one from every subset. While step (a) is a filtering operation, step (b) involves multiple PSA comparisons. In an FPGA implementation [6] of a flavor of BLAST (known as TREE BLAST) step (a) is implemented using block RAMs (BRAMs) and step (b) is carried out via stream processing. SERVER BLAST, which is another flavor of BLAST, has been implemented using a systolic array holding the query sequence while the filtered database flows through it through FIFOs on the FPGA [6]. In each case, the throughput performance was comparable to a dedicated server at National Center for Biotechnology Information, while achieving a lower power consumption figure. It is interesting to note that this was achieved by customizing the FPGA implementation to fit closely with the particular flavor of BLAST. The importance of customization is further exemplified by the very low speedup (2x compared with a regular PowerPC processor) that is achieved on a Cell Broadband Engine (CBE) [7]. On the other hand, since stream processing forms an integral part of the data flow, GPUs are likely to deliver better acceleration performance. Liu et al. [8] demonstrate this by achieving about 16x speedup on nVidia GeForce 7800 GTX GPU vis-à-vis a unicore run of OSEARCH, an MSA tool. In this case, however, the challenge lies in (a) (relatively difficult) mapping of the problem on to a GPU in terms of computer graphics primitives (using CUDA), which the authors have done for Smith-Waterman algorithm [1], and (b) in being able to utilize the large number of stream processing elements offered by a GPU, which the authors have done by identifying that all elements in the same anti-diagonal of a DP matrix can be computed in parallel. A similar exercise, but for protein sequences, using an enhanced version of Smith Waterman algorithm has been shown to yield up to 186 GCUPS throughput on a dual-GPU GeForce GTX 690 graphics card [9]. While this represents a significant improvement over the FPGA implementation, there are significant development costs involved.

While FPGA provides a high degree of customizability provided the processing resource requirements are modest, GPUs can offer abundant stream (SIMD) processing resources only if the problem can be formulated in a manner that can take advantage of its architecture, which may often pose the greatest challenge. To demonstrate that a tradeoff between these two paradigms may be feasible and desirable, Benkrid et al. [10] present a detailed design and implementation of a generic and highly parameterized FPGA skeleton for PSA using a high-level language Handel-C, and report two orders of magnitude improvement over software for PSA. The proof of efficacy of this approach is further strengthened in [11] where Sarkar et al report 2-3 orders of magnitude better performance compared to other hardware accelerators with a 64-PE on-chip network-enabled platform. We elaborate on this in Section III.

B. Phylogeny Reconstruction

Phylogeny reconstruction, or reconstruction of evolutionary trees for a given set of species is a problem that is relevant to a broad spectrum of biologists – from those interested in identifying evolutionary patterns of fast-mutating pathogens to those working on the ambitious project "Tree of Life" that aims to infer the phylogeny of all known life forms. In all cases, the goal is to build a phylogeny by observing and characterizing variations at the DNA, RNA or protein level. As such, the inputs to phylogenetic inference tools are the products of sequence analysis. Most of the algorithms and methods for inference are characterized by computational intractability and rely on the use of heuristics. This makes the problem data-driven compute-intensive, and one that has the potential to greatly benefit from hardware acceleration.

Typical phylogeny reconstruction methods are based on genomic distance (e.g. neighbor joining), combinatorial optimization (e.g. **breakpoint phylogeny**), or statistical methods (e.g. **maximum likelihood** (ML)). Statistical and combinatorial optimization methods are more popular owing to their higher accuracy, but they have a high degree of computational complexity. Statistical methods, typically having super-exponential algorithmic time-complexity, have another advantage that they can also provide a measure of the error in their estimates. While combinatorial optimization typically involves integer computation, statistical methods involve real numbers necessitating floating-point computation.

In the pioneering work done by Blanchette et al. on breakpoint phylogeny [15], they showed that the problem of reconstructing an optimal tree could be reduced to solving numerous instances of the classical Traveling Salesman Problem (TSP) with bounded integer edge-weights. This makes breakpoint phylogeny reconstruction an NP-Hard discrete combinatorial optimization problem that needs to be tackled using a combination of efficient parallelization strategies, run-time heuristics and hardware acceleration. Branch-and-bound method [16] is one the most popular run-time heuristics for solving TSP. Bakos and Elenis [17] accelerated whole-genome phylogeny reconstruction by a factor of 1005x by parallelizing breakpoint median computation and designing a hardware accelerator co-processor. A hardware-software codesign solution, this work constructs the input graph using software and implements the combinatorial search and TSP computation operations on an FPGA-based multicore. The software portion also features elaborate search tree partitioning so as to avoid complex load balancing and inter-core communication issues. However, this work considers only synthetic data sets, while the run-time complexity of the branch-and-bound method (i.e. degree and number of branches getting pruned) is highly dependent on the actual input graph. Majumder et al. [18] proposed a NoC-based custom multicore platform, evaluated the acceleration of the breakpoint-phylogeny application with both synthetic and biological input data, and demonstrated up to 8430x speedup. Their solution features an on-chip network-enabled architecture with up to 64 lightweight cores, each of which supports fine-grained parallel and pipelined matrix computations. No special software-level load-balancing or problem-partitioning techniques were required to achieve the desired speedup, while also making the system very energy-efficient. We discuss more on this in Section III.

Statistical methods of phylogeny reconstruction rely on maximizing the likelihood (ML) or the a posteriori probability (**Bayesian Inference**, BI). Both methods are based on a particular phylogenetic likelihood function (PLF) originally proposed by Felsenstein [19]. In its native form, ML or BI optimization has super-exponential time complexity; hence, several approximate algorithms have been proposed [20]. Most hardware acceleration techniques apply to one or more such approximate algorithms. Since these optimizations are over the real number space, all hardware accelerators must support non-integer arithmetic to a reasonable degree of accuracy. As with breakpoint phylogeny, hardware acceleration systems are based on HW-SW codesign, FPGA-based systems, GPU, CBE, general-purpose and custom multicores.

Mak and Lam [21] proposed a HW-SW codesign approach to accelerating the ML problem. On the software side, they proposed a Genetic Algorithm for Maximum Likelihood (GAML) that runs on a unicore CPU. On the hardware side they used a Xilinx Virtex FPGA to accelerate the ML equation computation. However, the probability model of evolution they considered is the Jukes-Cantor model, which is too simplistic for most real purposes (unlike General Time Reversible, or GTR model). On the FPGA alone, they were able to speed up the ML equation computation by 300x, but the overall time-to-solution reduction was much less. There are two key observations here – first, the degree of targeted acceleration depends on the approximation algorithm used and the customization in the hardware, and second, the target for acceleration needs to cover a significant portion of the original (unicore) run-time in order to translate the speedup to lower time-to-solution.

There has been a significant amount of software development on statistical methods of phylogeny (e.g., see [22]) but as we have explained, these may be inadequate in order to process large amounts of data being generated from sequence analysis at a desired throughput. These programs need to be efficiently parallelized to take advantage of hardware accelerators. Also, these accelerators need to be designed with a focus on the primary computation modules (e.g. PLF). An ML program that has been extensively experimented with owing to a high degree of parallelizability is Randomized Axelerated Maximum Likelihood version VI for High Performance Computing (RAxML-VI-HPC). Unlike GAML [21], it can incorporate various models of evolution and variable rate parameters across sites. Blagojevic et al. exploited the various levels of parallelization offered by this program, custom-mapped different modules and created explicit synchronization schedules on a CBE [23]. Although they reported speedup over general-purpose multicores, such as Intel Xeon and IBM Power5, the sheer complexity of porting the application (software-level customizations and optimizations required for CBE) entails a significant software redevelopment effort that makes the entire combination unappealing. A more focused FPGA-based hardware implementation to speed up PLF computation is presented in [24]. Here, Alachiotis et al propose a Xilinx Virtex 5 based platform that uses a large number of DSP slices to implement double precision floating-point arithmetic for millions of computations of the likelihood equation. Given the scale of parallelization, the number of DSP slices poses a limitation and requires very stringent resource allocation using multiplexers. Although speedup is modest (<10x), which is attributable to the resource constraint and the clock frequency constraint on the FPGA, this work highlights the importance of accelerating the PLF computation and resource allocation to multiple PLF computation kernels to achieve overall speedup. Similar to ML (i.e. based on PLF), a popular BI tool, MrBayes, has been accelerated on different hardware platforms such as GPU, CBE and general-purpose multiprocessors. Pratas et al [25] evaluate performance, scalability and programmability of MrBayes on these platforms.

Given the importance of being able to handle a massive scale of parallel real-number computation kernels featuring a particular set of operations (i.e. PLF computation), it would be intuitive to (a) design custom lightweight cores that can carry out these computations more efficiently than a general-purpose floating-point processor, (b) design a framework that overcomes the communication bottleneck, and (c) design a simple, efficient, hardware-managed resource allocation system that optimizes usage of processing resources on the accelerator. These approaches are followed in [26] to accelerate RAxML-VI-HPC. The accelerator followed a co-processor model where computationally heavy tasks contributing to a significant majority of the unicore run-time were offloaded to it via PCIe. Application speedup of up to ~1061x was achieved using 3-D NoC. We refer the reader to Section III for further details on this design.

It is clear from the discussion in this section that NoC-enabled platforms are better poised to deliver higher throughput compared to other hardware accelerators. In the next section, we provide more details on these architectures. .

III. NoC-Based Accelerators for Bioinformatics

A. Sequence Analysis

As mentioned in the previous section, the NoC-based platform for accelerating PSA [11] delivers orders of magnitude improvement in speedup over previously considered hardware accelerators. This work designs NoC architectures and evaluates their performance for the two commonest techniques for evaluating sequence alignments in parallel – the parallel prefix (PP) method and the anti-diagonal (AD) methods for solving the PSA DP table. For two sequences *S1* and *S2* with lengths *m* and *n* respectively, each of *p* PEs gets *n/p* characters from *S2*, and every character from *S1* one at a time in the PP method. The communication pattern among PEs is point-to-point as shown in Fig. 1; hence, the authors have used circuit-switched routers for inter-PE communication instead of full-blown wormhole switching. The network latency is a determining factor of system performance, and is minimized if the NoC topology is a *log_2p*-dimensional hypercube. However, since such a topology is infeasible for implementation, this work embeds this hypercube on to a 2-D mesh. In order to maintain one-hop links between PEs that were originally neighbors on the hypercube (but may be several hops away on the 2-D mesh), this work proposes and implements a special NoC switch that can provide a bypass path between next-to-neighbor PEs on a mesh in both the directions.

In the AD method, both sequences *S1* and *S2* are available to all PEs. At each step of the forward pass, the PEs compute an anti-diagonal of the DP table. This operation is easily parallelized among the available PEs because each cell of the anti-diagonal can be computed independently. The allocation of cells to PEs does not impact the overall computation complexity, but it is shown that allocating cells to PEs in a cyclical manner renders the traffic pattern generated from the forward pass of AD to be entirely neighborhood communication. In the backward pass, processors are grouped depending on the number of cells in each partition of the anti-diagonal following the technique proposed by Hirschberg [29] and modified by Huang [3]. This requires a broadcast-like communication, and the bypass logic in the NoC switches developed for PP method can be effective in this case too.

Energy and timing (latency) characteristics of the NoC-based platforms were evaluated for both approaches. For PP method, computation energy decreased with increasing *p*, as expected, but the total energy dissipation was dominated by the communication energy, which increased with increase in *p*. For small *p*, the total communication time is much smaller than the computation time. With increasing *p*, the computation time falls significantly but the communication time increases linearly, and eventually dominates the total time. The optimum value of *p* was shown to be 16 using the energy-latency product. In the AD method, both computation and communication latencies decreased with increasing *p*, but communication energy shows a noticeable increase beyond *p*=64. From the energy-latency product analysis, *p*=128 was found to be the optimum value. The PP implementation achieved nearly 23000x speedup over software and at least two order of magnitude over the nearest hardware accelerator, while the AD implementation sped up the computation by ~9500x over software and over 60x with respect to the nearest hardware accelerator (CBE in both cases [7]). A detailed performance comparison is given in Table 1.

This work demonstrates that a NoC-based hardware accelerator offers significant improvement over other hardware accelerators owing to the custom made PE architecture and interconnection topology. Even with a standard multi-hop mesh-based topology, it is possible to achieve orders of magnitude performance improvement compared to other existing hardware accelerators. The quantum of acceleration clearly demonstrates that in addition to efficient, custom PEs, the network-on-chip (NoC) acts as a critical enabler to fast, low-latency data exchange between different PEs. Since the particular topology of NoC is also a design parameter, the architect gets the additional flexibility to choose the appropriate architecture for the PE *and* NoC to deliver enhanced performance, which makes NoC a platform of choice for accelerating other similar applications.

B. Phylogeny Reconstruction

There are two broad classes of applications in this domain – one that involves combinatorial optimization over integer number space (such as solving multiple instances of bounded edge-weight TSP) and the other that involves use of statistical methods (based on both likelihood and a posteriori probability) and different models of evolution with variation in rate

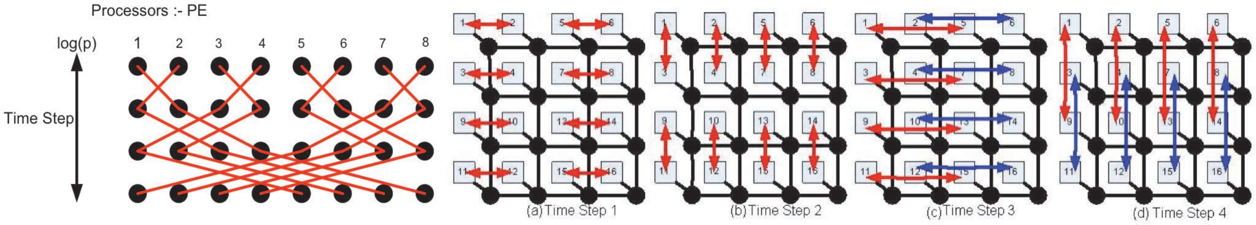

Figure 1. Inter-core communication pattern in the parallel prefix method

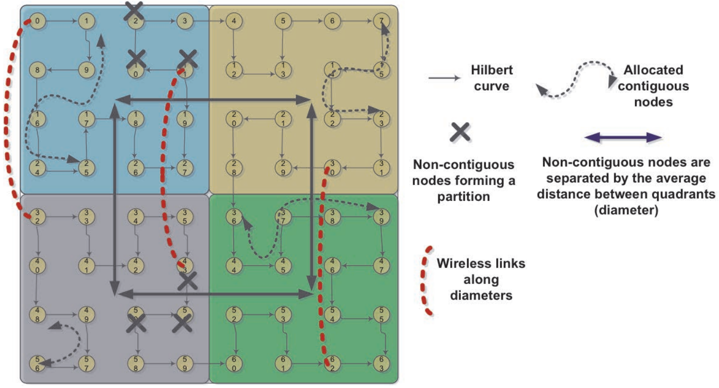
Hilbert curve
Allocated contiguous nodes
Non-contiguous nodes forming a partition
Non-contiguous nodes are separated by the average distance between quadrants (diameter)
Wireless links along diameters

Figure 2. A Hilbert curve embedded on an 8x8 mesh NoC. Also shown are non-contiguous nodes allocated to one task kernel, and wireless shortcuts.

parameters across sites. Breakpoint phylogeny is an example of the former class and ML phylogeny an example of the latter.

For breakpoint phylogeny, the challenge is to solve multiple TSP kernels in a very time-efficient manner as part of the branch-and-bound run-time heuristic [16]. This method requires the solution of a matrix reduction problem to determine a lower-bound cost at every node in the solution space tree. One of the key innovations in [18] is a pipelined architecture that also exploits the fine-grained parallel parallelism in this computation to deliver high-throughput linear-time matrix reduction [27]. Another challenge is to store on the chip the solution space tree (adjacency matrix format) that changes at every node (as a result of matrix reduction). The memory requirement has also been reduced by storing only changes in the adjacency matrix while traversing down the branch, and efficiently reconstructing the matrix while coming up (towards the root); genome lengths of up to 1024 can be handled. Different subtrees are assigned to different PEs and the lower-bound costs need to be exchanged among them to determine when some branches can be pruned. A novel limited broadcast mechanism that reduces unnecessary flooding of the NoC works well on mesh and quad-tree topologies. The quad-tree topology is better for higher system size, and provides greater energy efficiency. Speedups of over 8400x have been obtained with 64-PE systems, while consuming a maximum of 28 W of power. Performance evaluation was carried out with real genomes, and greater speedup was obtained in cases where larger number of TSP kernels needed to be evaluated, thereby demonstrating the efficacy of the solution.

In the case of ML phylogeny reconstruction, the challenge is to traverse a large number of bootstrap trees and find out the most likely tree. This entails optimization in a real continuous space of numerous tree topologies with varying branch lengths, and numerous computations of PLF [19]. As mentioned in Section II-B, there are three primary challenges involving (a) efficient real number arithmetic, (b) low-latency data exchange among computing kernels and (c) seamless allocation of numerous (billions of) task kernels. For (a), the authors in [26] followed a Fixed-Point Hybrid Number System, where real numbers are represented in log-domain as fixed-point numbers, which enables fast multiplication, addition, log and antilog computations. For (b), they explored different 2-D and 3-D NoC frameworks and their effect on speedup and energy consumption. 3-D NoC outperformed 2-D NoC by delivering up to 62% more speedup while consuming up to 38% less energy. The resource allocation problem (c) is particularly interesting in this case. The application spawns a large number of task kernels (that can be accelerated) of varying computation footprints. Each kind of kernel needs to be allocated a different number of PEs, multiple kernels execute concurrently, and they have varying execution times. Additionally, PEs allocated to a particular kernel need to communicate with one another, hence they need to be co-localized. Finally, all of these decisions must be made with the minimum possible time overhead. The authors in [26] simplified the resource allocation problem (on 2-D/3-D NoC) by translating it to a single dimension with a help of space-filling Hilbert curves logically interposed on the physical NoC. An example is shown in Fig. 2 for an 8x8 NoC. The accelerator followed a co-processor model where computationally heavy tasks contributing to a significant majority of the unicore run-time were offloaded to it via PCIe. Application speedup of up to ~1061x was achieved using 3-D NoC (considering all overheads). A comparison of application speedups for ML phylogeny is given in Table 2.

C. Beyond Wireline NoCs

NoC-based platforms have delivered tremendous gains in performance over other conventional hardware accelerators, both in terms of run-time and total energy consumption. However, some of these gains come from NoC topologies that face implementation challenges in the current technology scenario, such 3-D NoC. Ordinary mesh-like (mesh, torus, folded torus) topologies lead to a high network diameter, a problem that is only partially solved by hierarchical topologies, such as quad-tree [27]. The maximum number of hops required for any two communicating PEs is still a function of the system size. In this context, there is a need to introduce efficient "long-range" links, which can achieve single-cycle communication, essentially creating shortcuts. This will help to reduce both the energy and the latency of communication. We have already demonstrated that a small-world NoC architecture where short-range local interconnects are implemented through normal metal wires and long-range shortcuts are implemented through wireless interconnects can achieve higher bandwidth and lower energy dissipation compared through the traditional wireline mesh architectures [31][32]. In particular, it has been shown in [28] that introduction of a few wireless shortcuts (see Fig. 2) on a folded torus NoC leads to a throughput of over 10^{11} vector operations per second, where each operation consumes ~0.5 nJ and produces a favorable thermal profile. With respect to the ML phylogeny application, nearly twice the speedup of a 3-D NoC is obtained on a 2-D NoC with a few wireless shortcuts [30], while the allocation methods introduced in [26] still hold good for this wireless NoC (WiNoC) topology, even with varying computational footprints of task kernels. Hence, the choice of small-world based WiNoC over conventional mesh-based NoC is clear if we want to achieve high throughput with lower energy dissipation. This is true for all kinds of applications – SA, phylogeny reconstruction, etc. – because all of these involve some kind of long-range communication.

Table 1. Speedup of PSA on various hardware accelerators

Platform	**FPGA**	**GPU**	**CBE**	**NoC**
Speedup	~100	70	6	22000

Table 2. Speedup of breakpoint and ML phylogeny on various hardware accelerators

Platform	**FPGA**	**GPU**	**CBE**	**General-purpose multicore**	**Wired NoC-based multicore**	**WiNoC-based multicore**
Breakpoint phylogeny	1005	-	-	-	8430	-
Maximum likelihood	381	8.5	12	12	1061 (3-D), 908 (2-D)	2050

IV. Future Trend

Many-core platforms of today have largely embraced the NoC paradigm, both in academia and in industry, as exemplified by Xeon Phi (Intel), TILE-Gx72 (Tilera), etc. However, these platforms typically have mesh-like wired NoCs (obvious choice due to ease of VLSI implementation) and high power budgets. We show here that the exact communication pattern is determined by the particular nature of the application (or class of applications). We also bring out the importance of designing hardware accelerators tailored to the application domain, and the benefits of designing WiNoC platforms. Although there is a certain cost associated with this effort, many computational biology applications have a scientific and commercial impact that is large enough to absorb it. In this context, we see immense scope for accelerated biological discovery from cyber-physical systems that can benefit from energy-efficient WiNoC-based many-core platforms.

References

[1] T.F. Smith, M.S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 25 March 1981.

[2] T. Oliver, B. Schmidt, D. Nathan, R. Clemens, D. Maskell, "Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW," *Bioinformatics*, vol. 21, no. 6, pp. 3431-3432, 2005.

[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, N. Futamura, and K. Mehrotra, "Parallel biological sequence comparison using prefix computations", *J. Parallel and Distributed Computing*, Volume 63, Issue 3, March 2003, pp.264–272.

[5] R. E. Ladner and M. J. Fischer. 1980. Parallel Prefix Computation. *J. ACM* 27, 4 (October 1980), 831-838.

[6] M.C. Herbordt, J. Model, G. Yongfeng, B. Sukhwani, T. VanCourt, "Single Pass, BLAST-Like, Approximate String Matching on FPGAs," in *Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006 (FCCM '06)* , pp.217-226, 24-26 April 2006.

[7] V. Sachdeva, M. Kistler, E. Speight, T.-H.K. Tzeng, "Exploring the Viability of the Cell Broadband Engine for Bioinformatics Applications," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2007 (IPDPS 2007)*, pp.1-8, 26-30 March 2007.

[8] W. Liu, B. Schmidt, G. Voss, W. Muller-Wittig, "Streaming Algorithms for Biological Sequence Alignment on GPUs," *IEEE Trans. Parallel and Distributed Systems,* vol.18, no.9, pp.1270-1281, Sept. 2007.

[9] Y. Liu, A. Wirawan, B. Schmidt, "CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions," *BMC Bioinformatics* **14**:117, 2013.

[10] K. Benkrid, Y. Liu, A. Benkrid, "A Highly Parameterized and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol.17, no.4, pp.561-570, April 2009.

[11] S. Sarkar, G.R. Kulkarni, P.P. Pande, A. Kalyanaraman, "Network-on-Chip Hardware Accelerators for Biological Sequence Alignment," *IEEE Transactions on Computers,* vol.59, no.1, pp.29-41, Jan. 2010.

[12] C.B. Olson, et al, "Hardware Acceleration of Short Read Mapping," in *Proceedings of the IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM 2012)*, pp.161-168, 29 April – 1 May 2012.

[13] T. Martínek, M. Lexa, "Hardware Acceleration of Approximate Tandem Repeat Detection," in *Proceedings of the IEEE 18th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM 2010)*, pp.79-86, 2-4 May 2010.

[14] M.N.M. Isa, K. Benkrid, T. Clayton, "A novel efficient FPGA architecture for HMMER acceleration," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2012*, pp.1-6, 5-7 Dec. 2012.

[15] M. Blanchette, G. Bourque, D. Sankoff, "Breakpoint phylogenies," *Genome Informatics Workshop*, Tokyo: University Academy Press, pp. 25-34, 1997.

[16] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Operations Research* 196311:6 , 972-989.

[17] J.D. Bakos, P.E. Elenis, "A Special-Purpose Architecture for Solving the Breakpoint Median Problem," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol.16, no.12, pp.1666-1676, Dec. 2008.

[18] T. Majumder, S. Sarkar, P.P. Pande, A. Kalyanaraman, "NoC-Based Hardware Accelerator for Breakpoint Phylogeny," *IEEE Transactions on Computers,* vol.61, no.6, pp.857-869, June 2012.

[19] J. Felsenstein, "Evolutionary trees from DNA sequences: A maximum likelihood approach," *J. Mol. Evol.* Vol. 17, No. 6, pp. 368-376, 1981.

[20] Z. Yang, "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods," *J. Mol. Evol.* Vol. 39, No. 3, pp. 306-314, Sep. 1994.

[21] T.S.T. Mak, K.P. Lam, "High speed GAML-based phylogenetic tree reconstruction using HW/SW codesign," in *Proc. of the 2003 IEEE Bioinformatics Conference, (CSB 2003)*, pp.470-473, 11-14 Aug. 2003.

[22] Phylogeny Programs (http://evolution.genetics.washington.edu/phylip/software.html) Last accessed 21 November 2014.

[23] F. Blagojevic, A. Stamatakis, C.D. Antonopoulos, D.S. Nikolopoulos, "RAxML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2007 (IPDPS 2007)*, pp.1-10, 26-30 March 2007.

[24] N. Alachiotis, E. Sotiriades, A. Dollas, A. Stamatakis, "Exploring FPGAs for accelerating the phylogenetic likelihood function," in *Proceedings of IEEE International Symposium on Parallel & Distributed Processing, 2009 (IPDPS 2009)*, pp.1-8, 23-29 May 2009.

[25] F. Pratas, P. Trancoso, A. Stamatakis, L. Sousa, "Fine-grain Parallelism Using Multi-core, Cell/BE, and GPU Systems: Accelerating the Phylogenetic Likelihood Function," in *Proceedings of the International Conference on Parallel Processing, 2009 (ICPP '09)*, pp.9-17, 22-25 Sept. 2009.

[26] T. Majumder, M.E. Borgens, P.P. Pande, A. Kalyanaraman, "On-Chip Network-Enabled Multicore Platforms Targeting Maximum Likelihood Phylogeny Reconstruction," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol.31, no.7, pp.1061-1073, July 2012.

[27] T. Majumder, S. Sarkar, P. Pande, A. Kalyanaraman, "An optimized NoC architecture for accelerating TSP kernels in breakpoint median problem," *Application-specific Systems Architectures and Processors (ASAP), 2010 21st IEEE International Conference on* , vol., no., pp.89,96, 7-9 July 2010.

[28] T. Majumder, P. P. Pande, A. Kalyanaraman, "High-Throughput, Energy-Efficient Network-on-Chip-Based Hardware Accelerators", *Sustainable Computing: Informatics and Systems (Elsevier)*, vol. 3, no. 1, March 2013, pp. 36-46.

[29] D.S. Hirschberg, "A Linear Space Algorithm for Computing Maximal Common Subsequences," *Comm. ACM,* vol. 18, no. 6, pp. 341-343, 1975.

[30] T. Majumder, P.P. Pande, A. Kalyanaraman, "Wireless NoC Platforms With Dynamic Task Allocation for Maximum Likelihood Phylogeny Reconstruction," *Design & Test, IEEE* , vol.31, no.3, pp.54,64, June 2014.

[31] S. Deb, et al., "Design of an energy efficient CMOS compatible NoC architecture with millimeter-wave wireless interconnects," *IEEE Trans. Comput.*, 2013, pp. 2382-2396.

[32] A. Ganguly, K. Chang, S. Deb, P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multi-core systems," *IEEE Trans. Comput.*, vol. 60, no. 10, 2010, pp. 1485-1502.