Simulation of Complex Biochemical Pathways in 3D Process Space via Heterogeneous Computing Platform: Preliminary Results

Jie Li, Amin Salighehdar, and Narayan Ganesan

Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken NJ 07030, USA
{jli8,asalighe,nganesan}@stevens.edu

Abstract. Biological pathways typically consist of upto hundreds of reacting chemical species and reactions within a biological system. Modeling and simulation of biological pathways in explicit process space is a computationally intensive, both due to the number of interactions and time-scale of processes. Traditional stochastic or ODE based simulation of chemical processes ignore spatial and biological information. Hence there is a need for new underlying simulation algorithms as well as need for newer computing systems, platforms and techniques. Such pathways describe exhibit considerable behavioral complexity in multiple fundamental cellular processes. In this work we present a new heterogeneous computing platform to accelerate the simulation study of such complex biochemical pathways in 3D reaction process space. Several tasks involved in the simulation study has been carefully partitioned to run on a combination of reconfigurable hardware and massively parallel processor such as the GPU. This paper also presents an implementation to accelerate one of the most compute intensive tasks - sifting through the reaction space to determine reacting particles. Finally, we present the new heterogeneous computing framework integrating a FPGA and GPU to accelerate the computation over the use of a any single platform. This framework can achieve 10-times speedup over a single GPU-only platform. Besides, the extensible architecture is general enough to be used to study a variety of biological pathways in order to gain deeper insights into biomolecular systems.

Keywords: GPU+FPGA, Process Simulation in 3D space, Heterogeneous Computing, Complex Biochemical Pathways, Stochastic Simulation.

1 Introduction

Simulation and study of such biochemical pathways will lead to deeper insights and understanding of functions of proteins, kinases and phosphotases that activate and de-activate reagents, sensitivity of various chemical species etc. There are several modeling and simulation tools that are used to study biological pathways, including but not limited to Ordinary Differential Equations(ODEs), graph

D. Goehringer et al. (Eds.): ARC 2014, LNCS 8405, pp. 227–232, 2014.

theoretical analysis of reaction networks, boolean networks and explicit modeling in reactive process space, with each having its own scientific, computational and implementation merits and disadvantages. Although, ODEs are a popular modeling framework and computationally very efficient, they only represent aggregate concentration of the species, and fail to capture many intricacies and local behavior mechanisms within the cell. On the other hand, reaction modeling in 3D process space is the most computationally intensive and serves as a virtual computational microscope into biological systems. Typically, modeling such biological pathways in reaction space requires millions of reagents and beyond and it is imperative to consider all-particle interactions simultaneously within the system. In this paper, we present a new heterogeneous computational framework to study the interactions enabled by the massively parallel processing capability of the GPUs and FPGAs. The computational framework will take the simulation and study of large biological systems to the next level, where in macro-biological systems such as cells, and interaction between multiple cells can be studied to gain valuable insights into real biological processes.

2 Algorithm and Implementation

Sequential Algorithm. Algorithms such as the Kinetic-Monte Carlo[1,2] and Gillespi Algorithm[3] have been used for stochastic simulation of chemical systems, on a sequential execution platform. The algorithm proceeds by listing all possible reactions and choosing to execute one of them based on the stoichiometric rate and the population of reagents. The time counter is then incremented appropriately. However, the procedure (a) doesn't capture spatial and local information and (b) is inherently sequential to be suitable for studying behavior of large number of reagents due to rapid growth of possible interactions between reagents. The number of feasible reactions grows with growing number of species as well as the the number of individual reagents. In general, the growth in the set of all possible interactions grows proportional to $O(N^2)$, for a set of N reagents and $O(M^2)$ for M different chemical species. In the above algorithm, the sequential nature of the enumeration of all possible reactions as required by the algorithm, which overwhelms the computation required to accurately simulate the process behavior. Hence the traditional algorithm above faces fundamental bottlenecks from a computational standpoint and is not scalable to simulation study of large biochemical systems within a reactive 3D space.

Scalable Concurrent Algorithm. In our previous work, we have designed and applied the following algorithm to study the growth of biofilms[4] which was implemented on on massively parallel processors such as GPUs. We have also used GPUs for simulation study of spatial molecular dynamics and their conformation[5]. However, in contrast to purely physical interactions, general chemical interactions will result in creation of new particles and consumption of others in a consistent pattern and in predefined quantities, as described by the chemical equations. Furthermore, in contrast to molecular dynamics problem, where fixed persistent agents interact with all the neighboring agents, chemical

reagents interact only with select neighbors while producing new products. In order to leverage the parallel and concurrent framework, each interacting entity or particle is treated as a "autonomous agent" that interacts with other such agents of different type in an independent and autonomous fashion. This helps overcome the sequential limitations imposed by traditional algorithms. The concurrent reactions at each time step is updated to reflect the consumption of old reagent particles and production of newer agents. One of the crucial tasks in transitioning from traditional algorithms to an explicitly defined 3D process space populated by individual particles is conversion of reaction rates to equivalent interaction radii. A pair of particles within the specified interaction radius on a collision course, will react together always or with a probability that is set by their velocities in order to produce the product of the specified reaction. It is very-well possible that each particle is within the interaction radius of several other particles capable of reacting with each other, in such case, efficient parallel techniques to select a set of mutually consistent reactions to carry out, must be formulated. The concurrent algorithm can be stated as follows,
(1) Initialize: The particles positions, drift velocities. **(2) Initialize Reaction Radii:** Enumerate the set of reactions between different types along with the interaction radius of the reaction. For first order reactions of type $A \rightarrow \phi$ or $A \rightarrow B + C$, each particle of type A is assigned a life-span by sampling from an exponential distribution parametrized by its decay rate. For reactions of type $A + B \rightarrow C + D$, the reaction radius is set based on the rate-constant and drift rate of particles[6].**(3) Build Neighbor List:** Divide the simulation volume into disjoint cubic cells of dimensions equal to the largest radius of interaction. In order to identify the neighbors of each particle only the current cell and the 26 adjacent cells in 3D need to be examined. For each particle, build a list of particles of compatible types that could react. This is done efficiently with the help of a stoichiometric bit-vector. In the stoichiometric bit-vector the jth element of ith bit-vector is set to 1 if type i can react with j. A separate lookup table stores the product each corresponding reaction between types i and j. **(4) Start the Simulation:** Sift through the 3D process space of each particle in parallel, scanning for reacting particles and carefully selecting the pairs of particles in a mutually consistent manner for reaction. Increment global time and repeat steps (1) - (4) until simulation time.

2.1 Heterogeneous Computing Framework

The high-throughput and similar nature computation required to process each agent makes any massively parallel processor such as the GPU a good initial choice. However, as we outline below, the reconfigurable hardware co-processor is extremely beneficial in handling tasks that would otherwise strain the memory bandwidth and instruction throughput of the GPU. In this work we demonstrate the power of heterogeneous computational framework in accelerating an application that is not amenable to massively parallel processor alone. In the original GPU implementation of the `NeighborList` build kernel, each thread-block is responsible for building the neighbor list of all the particles within a specific cell

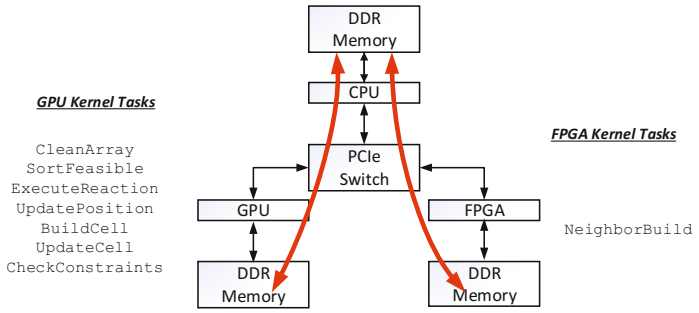

Fig. 1. Heterogeneous system framework

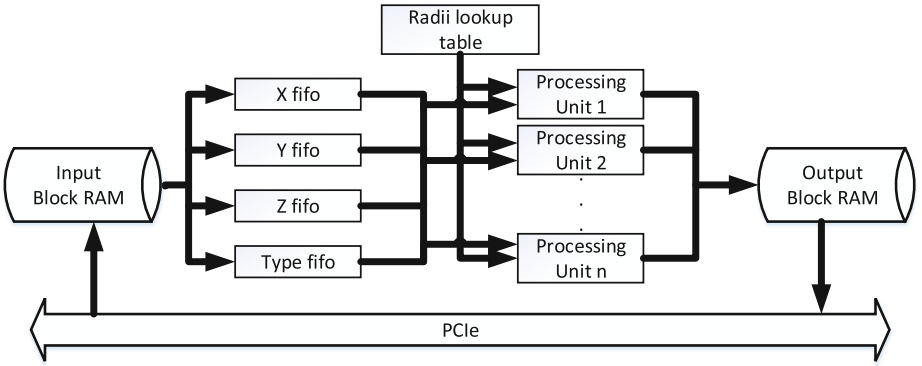

Fig. 2. processing unit architecture

with in the reaction space. To this effect each thread block sifts through the particles in 26-adjacent cells in addition to its own cell to determine the neighbors of each particle within the cell. Among all kernel functions in the table, `NeighborBuild` function consumes 97% of the total execution time. This is due to fact that any parallel implementation that sifts through adjacent cells will require 27x bandwidth to the off-chip global memory, as each cell performs the same task to its neighboring cells. The problem is further amplified by the fact that the `NeighborBuild` kernel is called far more often here than in an application such as molecular dynamics. The faster the movement of particles more often the `NeighborBuild` kernel needs to be called. This places undue strain on the global memory bandwidth even on a high-throughput device and throws off the instruction-to-memory ratio far from the optimal value. In order to overcome this bottleneck, we implement the `NeighborBuild` task on the FPGA and leverage the capability of the heterogeneous computational platform.

Hardware Design. Although the presented application is unique and the application domains are different, previous work on accelerating molecular dynamics on reconfigurable platform[7,8], is most related to the current implementation. We present the hardware design for the task to compute `NeighborList` and a

Device	XC6VLX240T
Logic Cells	241,152
Conf.Logic Blocks	37,680
DSP48E1	3,650
Block RAM Blocks	768

(a) Device Capability

Resource	quantities
$SliceRegisters$	54
$Memory$	40
$DSP48Es$	8
$Maximumspeed$	$300M$

(b) Device utilization

unified heterogeneous computing framework for large scale process simulations in 3D space. Target hardware: a generic PC and GPU GTX 580 and a PCIe plug-in board ML605 with Xilinx XC6VLX240T. It is possible to leverage the capabilities of each device via a task-level partition of the kernels as shown in Figure (1). The FPGA processes one central cell at a time. Each processing unit (figure 2)needs to compute the distance between all pairs of particles i and j, where i must be in the central cell but j can be in any of the 26 neighborhood cells or in the central cell. In order to fully parallelize each cell, the system needs as many processing units as the particles in the central cell. One particle in the neighboring cell is processed per time cycle. So, the total execution time of one central cell is 27 x the maximum number of particles in any cell. In order to preserve the accuracy of distance calculation, floating point precision is necessary. Fortunately, modern FPGAs are equipped with ample DSP units that make floating point distance calculation within each processing element possible. With the available resources, it is usually advantageous to use the existing floating point units instead of synthesizing custom fixed precision units. In our implementation particles coordinates and reagent types are copied from GPU to FPGA. We also maintain a reaction radius lookup table on FPGA, as each reaction may have different effective reaction radius. In order to process a million particles, the total amount of data transferred to the FPGA for `coordinates` array is 1Million x 3 channel x 32 bits ≈ 12MB and for particle `types` array is 1Million x 32 bit ≈ 4MB. However, the copy-back of the `FeasibleList` can be overlapped with computation. The tasks partitioned among the FPGA and GPU such that the `NeighborList` build is performed on the FPGA and the other remaining functions on the GPU. Once the computation is initiated, data transfer between GPU and FPGA would take

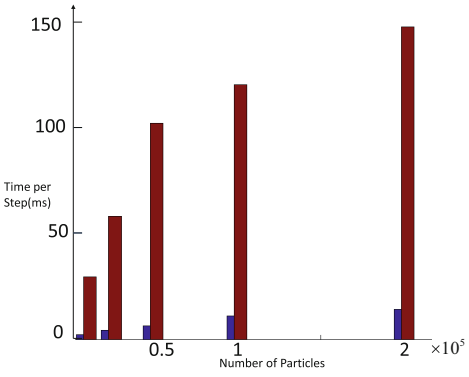

Fig. 3. Performance of the kernel with respect to the total number of particles (independent agents). The FPGA performance is shown in blue bars while the GPU performance is shown in red.

place once per iteration. The critical resources on the FPGA are the hard multipliers, the registers and the block RAMs as shown in table (2).

2.2 Experiments and Performance

The computational framework presented here especially suitable to simulate large and complex biological pathways serving as a macro-molecular visual scope and helps observe key biochemical reactions, as the events unfold in space and time. For performance comparisons, the JAK-STAT signaling pathways was initialized with 1.23 million particles or independent reacting agents, within a simulation space of $200 \times 200 \times 200$ distance units. For performance comparisons, we set different initial number of particles for this system from 10k to 2,000K in order to measure the average time-per-step. In figure (3), we compare the performance of GPU and FPGA implementation of the compute intensive task of calculating the `FeasibleList`. The FPGA achieves approximately 10 x speedup over GPU-only implementation for all system sizes while using the 32 bit floating point to maintain simulation quality.

Acknowledgements. The authors would like to thank the Xilinx University Program(XUP) and the NVIDIA-Professor partnership for their generous support and donation helpful in carrying out the research.

References

1. Cox, D.R., Miller, H.D.: The Theory of Stochastic Processes. Methuen, London (1965)
2. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 184–199. Springer, Heidelberg (2007)
3. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81(25), 2340–2361 (1977)
4. Li, J., Sharma, V., Ganesan, N., Compagnoni, A.: Simulation and study of large-scale bacteria-materials interactions via bioscape enabled GPUs. In: Proceedings of ACM-BCB 2012 (2012)
5. Taufer, M., Ganesan, N., Patel, S.: GPU enabled macromolecular simulations: Challenges and opportunities. IEEE Computing in Science and Engineering 15(1) (January 2012)
6. Erban, R., Chapman, S.J.: Stochastic modelling of reaction-diffusion processes: algorithms for bimolecular reactions. Physical Biology 6(046001) (2009)
7. Chiu, M., Herbordt, M.C.: Molecular dynamics simulations on high-performance reconfigurable computing systems. ACM Trans. Reconfigurable Technol. Syst. 3(4), 23:1–23:37 (2010)
8. Gu, Y., VanCourt, T., Herbordt, M.C.: Explicit design of fpga-based coprocessors for short-range force computations in molecular dynamics simulations. Parallel Computing 34(4-5), 261–277 (2008)