2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops

Process Simulation of Complex Biochemical Pathways in Explicit 3D Space Enabled by Heterogeneous Computing Platform

Jie Li
Dept. of Elec. and Comp. Engg.
Stevens Institute of Technology
Hoboken, NJ.
jli8@stevens.edu

Amin Salighehdar
Dept. of Elec. and Comp. Engg.
Stevens Institute of Technology
Hoboken, NJ.
asalighe@stevens.edu

Narayan Ganesan
Dept. of Elec. and Comp. Engg.
Stevens Institute of Technology
Hoboken, NJ.
nganesan@stevens.edu

***Abstract*—Biological pathways typically consist of dozens of reacting chemical species and hundreds of equations describing reactions within the biological system. Modeling and simulation of such biological pathways in explicit process space is a computationally intensive due to the size of the system complexity and nature of the interactions. Such biological pathways exhibit considerable behavioral complexity in multiple fundamental cellular processes. Hence, there is a strong need for new underlying simulation algorithms as well as need for newer computing platforms, systems and techniques. In this work we present a novel 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 a 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 and obtain better performance over the use of any single platform. The platform achieves 5x total speedup when compared to 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.**

I. Introduction

Biological systems encompass complexity that far surpasses many artificial systems. A biochemical pathway is a cascade of reactions then lead to formation of certain end product(s), in order to carry out specific task(s) within the biological system. 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 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. Modeling via Partial Differential Equations(PDEs) on the other hand, does not capture the discrete particle effects(Eg: too few number of DNA molecules within the nucleus) on the overall behavior of the system but strikes a balance between computational cost and accuracy. On the other hand, reaction modeling in 3D process space is the most computationally intensive but at the same time serves as an accurate virtual computational microscope into biological systems.

In this work, we present an algorithm for simulation study and a novel implementation of the computational technique via a heterogeneous platform. 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. Since, the algorithm, technique and the underlying computational framework, can be applied to a variety of biochemical pathways, we test and validate our system on the realistic JAK-STAT signal transduction pathway as an example application.

The rest of the paper is organized as follows. In the next section we give an overview of traditional stochastic simulation algorithms for simulation of chemical kinetics and then proceed to present our scalable-concurrent algorithm meant to study such complex pathways in reactive process space. We then present the architecture and design of the heterogeneous platform (FPGA + GPU) with the objective of maximizing the operating frequency and the size (in terms of number of particles) that fit on the FPGA while maintaining simulation quality. We apply the framework to study the behavior of a common yet complex JAK/STAT signal transduction pathway. Section III provides an overview of the JAK/STAT pathway. We finally present our results and validation experiments along with performance. We conclude with a discussion of potential future work and implications.

II. Algorithm and Implementation

A. Sequential algorithm

Algorithms such as the Kinetic-Monte Carlo[2], [3] and Gillespi Algorithm[1] 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. Traditionally Gillespie Algorithm[1] and its derivatives such as Kinetic Monte Carlo[2], [3] type algorithms have been used to study the behavior of an ensemble of processes in a shared probability space, meaning a single random variable dictates the next feasible reaction. The Kinetic Monte Carlo based methods broadly follow: **(1)** Enumeration of all feasible reactions between all individual reagents that are at any given conformation. **(2)** Random selection of one of the reactions to carry out, based on the generated set of feasible reactions. The selection of the reaction is based on the relative probability and rate of that reaction. **(3)** Increment the simulation time corresponding to the rate of the reaction just selected. Higher the reaction rate or the number of distinct reactions, smaller the time increment. **(4)** Updating the new set of particles and their coordinates according to the selected reaction and their movements within the 3D space. However, the procedure 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 outlined above in Step (1) overwhelms the computation required to accurately simulate the process behavior. Hence the traditional algorithm above faces fundamental bottleneck from a computational standpoint and is not scalable to simulation study of large biochemical systems within a reactive 3D space. Hence, an efficient simulation suite designed for studying large number of concurrent interactions and processes, and optimized for modern heterogeneous and reconfigurable computational platforms will be imperative in simulating drug treatment and therapy, gene expression, protein-protein interaction and micro-devices that interact with biological systems.

B. 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 massively parallel processors such as GPUs. The application of GPUs for simulation study of spatial molecular dynamics along with the challenges were also investigated previously[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, 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 an "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. In such a parallel and concurrent framework, each "agent" is capable of reacting with one or finite number of agents in its vicinity. The whole system is then subject to global simulation time that is incremented in fixed units throughout the course of the simulation. 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 with a probability that is set by their velocities which then leads to the products of the specified reaction. The concurrent algorithm (as shown in Figure(1)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. Each reaction ID contains a set of reacting types, the radius of interaction, reaction delay and the products. 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 (Figure(2). 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. The stoichiometric bit-vector is a array of size n of n-bit vectors, where n is the number of chemical species (or types). 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:** The algorithm is depicted in fig(1). The `neighborList` as described above is made computationally efficient via the Verlet Algorithm[7]. In the Verlet method, the `neighborList` need not be build at every step, if the radius is chosen to be larger than the reaction with largest interaction radius r_{max} plus a buffer r_{buf}. In this case, the `neighborList` of all particles stay the same until any one of the particles move a distance greater than $r_{buf}/2$, in which case it needs to be rebuilt. This eliminates the need for building the neighbor list at every time step. The simulation can be also configured to support multiple disparate reaction regions with different particle density.

It is very well possible that each particle is found 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 `ParallelSelect` algorithm proposed here (as shown in Fig(3) is instrumental in consistent particle selection in a parallel and concurrent simulation framework. The algorithm proceeds as follows: 1) Build the `feasibleList` for each particle, which is a subset of `neighborList` and contains the set of particles capable of reacting with the current particle. 2) Sort the `feasibleList` according to the Euclidean metric in order to set the reaction priority. 3) Each particle selects the first available particle in its feasible list for reaction. 4) If the selection is mutual then schedule the corresponding reaction in the reaction pipeline and mark the particle as not available for any more selections, else mark the particle as still available for selection by other particles. Perform steps 3) and 4) until converged or no more available particles in the `feasibleList`.

C. *Heterogeneous computing framework*

The high-throughput and similar nature of 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. The novel application area as well as its implementation on the heterogeneous computing platform has not been studied before. In our previous work[8], we described the FPGA-only architecture for the Feasible list

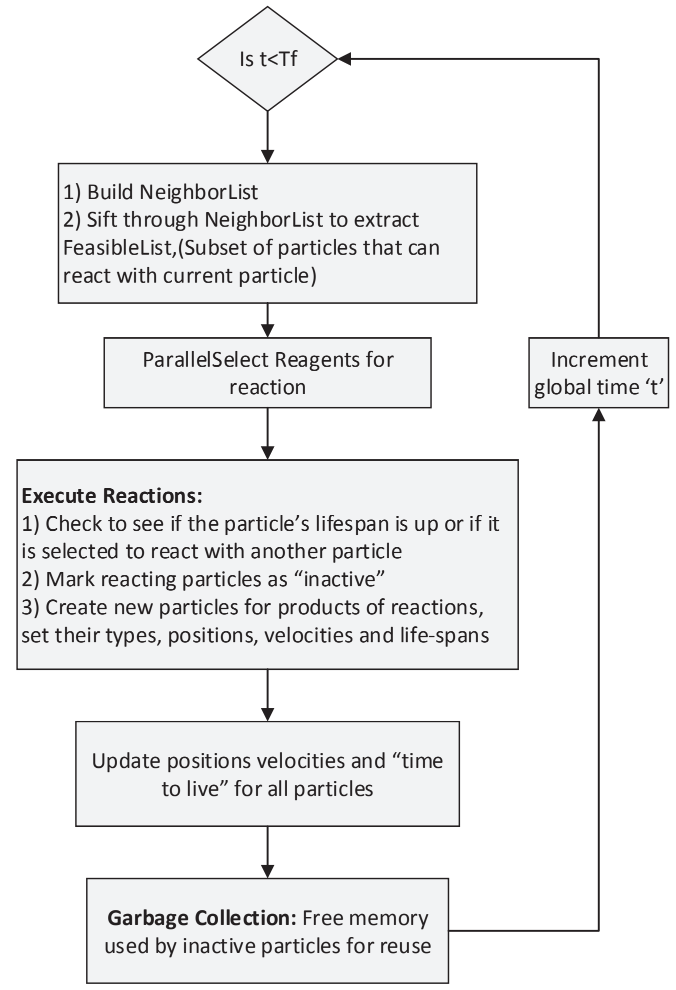
Is t<Tf
1) Build NeighborList
2) Sift through NeighborList to extract FeasibleList,(Subset of particles that can react with current particle)
ParallelSelect Reagents for reaction
Increment global time 't'
Execute Reactions:
1) Check to see if the particle's lifespan is up or if it is selected to react with another particle
2) Mark reacting particles as "inactive"
3) Create new particles for products of reactions, set their types, positions, velocities and life-spans
Update positions velocities and "time to live" for all particles
Garbage Collection: Free memory used by inactive particles for reuse

Figure 1. Tasks in process simulation of chemical kinetics in 3D process space.

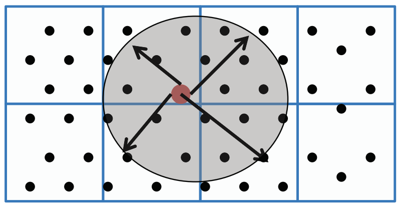

Figure 2. Schematic of `NeighborList` build in 2D

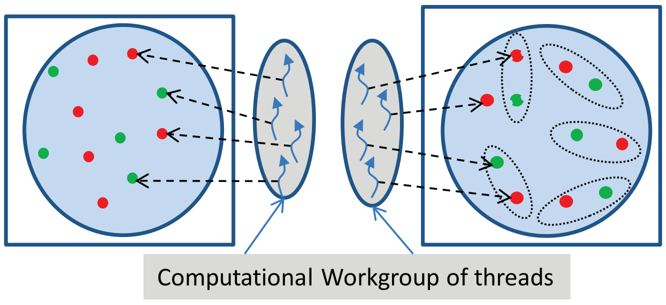

Figure 3. Particles in Red can react with particles in Green. Left: Each particle governed by a thread is in the vicinity of several particles that it can react with. Right: The `ParallelSelect` algorithm leads to consistent selection of reagents on a parallel execution framework.

generation. In the current work we build upon the hardware design and present the integrated (GPU+FPGA) comprehensive framework along with PCI-e bridge and interconnect performance. 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 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 (as shown in Figure(2)).

The above algorithm was implemented to study the behavior of JAK/STAT signal transduction pathway, within intra-cellular 3D space, which is outlined in Section(III). We provide a quick preview of the GPU-kernel times, just to motivate the need for heterogeneous hardware, while a full discussion of performance results follows in Section(III). Table IV shows the average performance of GPU-only implementation and GPU-FPGA heterogeneous implementation of the various kernel functions. Among all kernel functions in the table, `NeighborBuild` function consumes $> 90\%$ 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 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. The on-chip block RAM in an FPGA chip can be configured with more flexibility as needed. However differences in the operating clock speeds between the FPGA and GPUs (typically 200Mhz vs 1200Mhz) along with other on-chip capabilities make GPU and FPGAs suitable for different applications. Reconfigurable hardware are extremely good in handling divergent task pipelines via combinational logic, not subject to software synchronization overheads, do not require similarity of instructions and offer cycle-accurate performance predictability. On the other hand, GPUs have high-instruction throughput and floating point capability which is suitable for data parallel applications. It is possible to leverage the capabilities of each device via a task-level partition of the kernels as shown in Figure (6) and integrate the GPU and FPGA-based platforms.

On a GPU platform, the on-chip shared memory is a critical resource for high-throughput instruction processing, which is allocated per thread block to enable low-latency inter-thread communication. However the shared memory size as well as the bank-conflicts limits speedup in handling and accelerating computation that exhibit data dependency.

D. Hardware Design

Although the presented application is unique, previous work on accelerating molecular dynamics on reconfigurable platform[9], [10], is most related to the current implementation, but from a different application domain. In addition, in this work, we focus on the heterogeneous computing hardware to leverage the capabilities of multiple architectures. In this section we present the hardware design for the task to compute `NeighborList` and a unified heterogeneous computing framework for large scale process simulations in 3D space. This framework also considers other resources such as PCIe communication throughput and the memory. Target hardware: a generic PC and GPU GTX 580 and a PCIe plug-in board ML605 with Xilinx XC6VLX240T. The generic system design is shown in Figure 4.

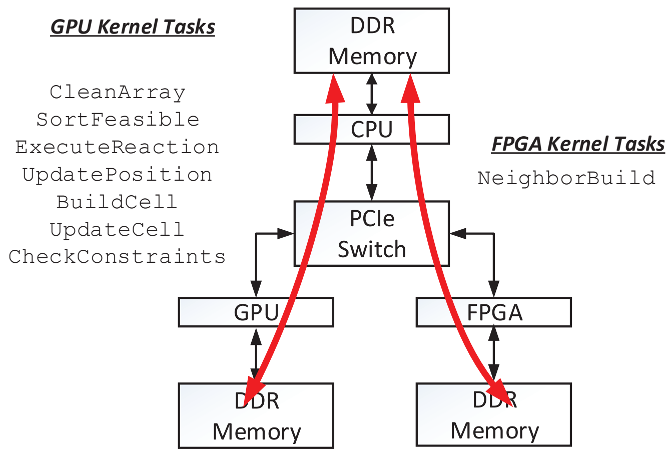

Figure 4. Heterogeneous system framework

The FPGA processes one central cell at a time. Each processing unit 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 one cell.

In order to preserve the accuracy of distance calculation, 32-bit single precision is necessary. Traditionally, since floating point resources on the FPGA are limited to be useful for large scale data processing applications, fixed point precision is implemented with some impact on accuracy. 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. This leads to greater than expected number of Processing Elements to represent particles within each cell.

In our implementation particle coordinates and reagent types are copied from GPU to FPGA as shown in Figure 5. We also maintain a reaction radius lookup table on FPGA, as each reaction may have different reaction effective radius. The total amount of data transferred to the FPGA (assuming an average number of particles to be 1 million) is `coordinates` = 1M x 3 channel x 32 bits = 12MB and `type` = 1M x 32 bit = 4MB. However, the copy-back of the `FeasibleList` can be overlapped with computation.

The FPGA-GPU communication takes place across the PCIe bus via DMA transfers access. The tasks partitioned among the FPGA and GPU such that the `NeighborList` build is performed on the FPGA and the other remaining tasks on the GPU. Once the computation is initiated, data transfer between GPU and FPGA would take place once per iteration. Beside the FPGA processing time, the data transfer which incurs certain overhead also contributes to the total calculation time. The PCIe x4 bus of ML605 board provides 800MBps bandwidth each direction. Normally, PCI transmission efficiency ratio is close to 0.8. Initially, the global list of coordinates of all particles arranged in the order of the cell is copied to the FPGA, in order to calculate the `NeighborList`. In order to decrease the copy-back bandwidth to the GPU, where the reactions are selected and are executed, we only need the list of reacting neighbors `feasiblelist` and not all the neighbors. This decreases the amount of data to be copied back to the GPU. The `feasiblelist` is a size n by m 32 bit array, where n is the maximum number of feasible neighbors and m is the number of total particles. The element $\{i, j\}$ is the global id of feasible neighbor j which particle i will react with. The `feasiblelist` is of size 32(where we assume maximum number of feasible reacting neighbors is 32) x 1M x 32 bit = 128MB which needs around 0.2s to transfer through PCIe.

The critical resources on the FPGA are the hard multipliers, the registers and, in particular, the block RAMs. In table(I) outline the resources on the part Xilinx XC6VLX240T.

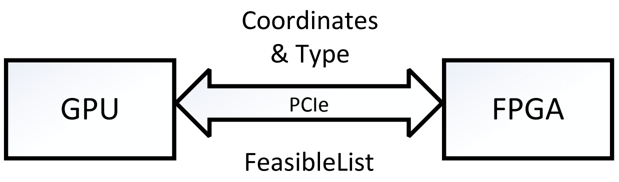

Figure 5. Data is exchanged between GPU and FPGA every time step. Particle coordinates are transferred to the FPGA for the computation of FeasibleList which is copied back to the GPU.

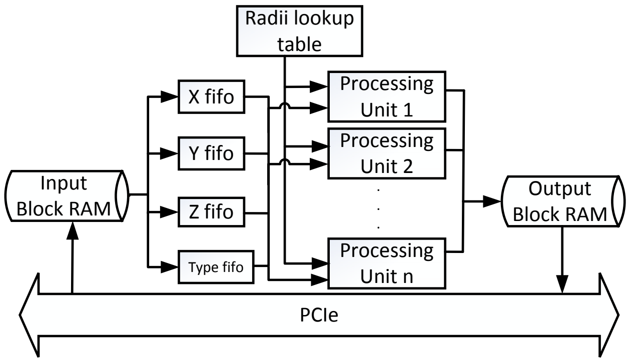

Figure 6. processing unit architecture

Table I
DEVICE CAPABILITY

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

III. JAKSTAT SIGNALING PATHWAY

We apply the current framework to study the behavior of the JAK/STAT signal transduction pathway. The framework is powerful enough to simulate the behavior and interactions of upwards of a million independent agents. The Janus kinase/signal transducers and activators of transcription (JAK/STAT) pathway is one of a handful of pleiotropic cascades used to transduce a multitude of signals for development and homeostasis in animals. In mammals, the JAK/STAT pathway is the principal signaling mechanism for a wide array of cytokines and growth factors[11].

The JAK-STAT pathway is intracellular signaling pathways in all vertebrates. These pathways have involved in

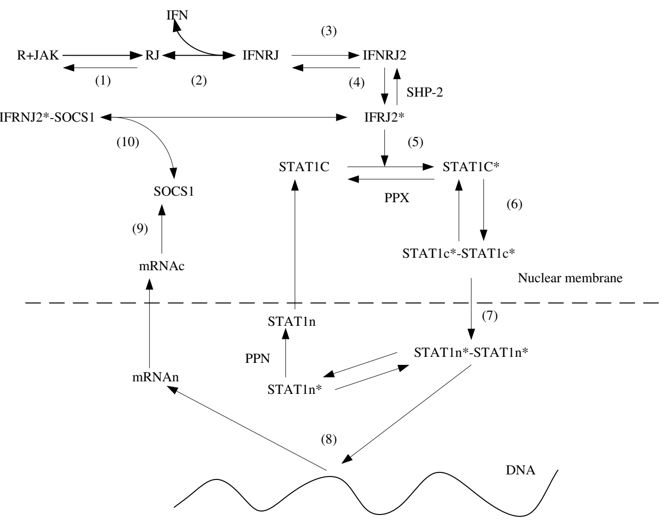

Figure 7. Kinetic model of JAK/STAT signaling pathway

with multiple fundamental cellular processes. Fig(7)shows the dynamic scheme of this pathway and Table (II) shows all the biochemical reactions included in the model along with the rates constants in Table(III). In step 1, JAK binds to the IFN- receptor (IFNR) and forms the IFNR-JAK(RJ) complex. In step 2, IFN- binds to extracellular domain of IFNR-JAK complex and forms the IFN–IFNR-JAK complex(IFNRJ). In next step dimerization of IFNRJ happens and IFRJ2 is formed. The dimerization of the RJ complex leads to the phosphorylation of several tyrosine residues by JAK (Step4) yielding a form as IFRJ2*. The STAT1 binds to IFNRJ2* and is phosphorylated by JAK(step5). The phosphprylated STAT1 forms a homo-dimer (step6). The phosphorylated dimers of STAT1 are translocated to the nucleus (step7) and work as the transcription factors(step8). The SOCS1 is induced by JAK/STAT pathway(step9). The SOCS1 binds to the activated receptor JAK and inhibits its kinase activity(step10).The SHP-2 is known to be a phosphotase for the RJ complex. In addition PPX and PPN are known to be phosphotases for phosphorylated STAT in the cytoplasm and the nucleus respectively[11]. In this kinetic analysis, all reactions are represented by mass-action kinetics. A cell is divided into two compartment,the cytoplasm and the nucleus. The phosphorylated STAT1 dimers(STAT1c*-STAT1c*) are translocated to the nucleus and dephosphorylated STAT1 monomers (STAT1n) are transported from the nucleus to the cytoplasm.

A. *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

Table II
CHEMICAL REACTIONS DESCRIBING JAK/STAT SIGNALING PATHWAY

Reactions; Corresponding rate constants
$[R]+[JAK] \leftrightarrow [RJ]$; k_1, k_{-1}
$[IFN]+[RJ] \leftrightarrow [IFNRJ]$; k_2, k_{-2}
$2[IFNRJ] \leftrightarrow [IFNRJ2]$; k_3, k_{-3}
$[IFNRJ2] \rightarrow [IFRNJ2^*]$; k_4
$[IFNRJ2^*]+[STAT1c] \leftrightarrow [IFNRJ2^*-STAT1c]$; k_5, k_{-5}
$[IFNRJ2^*-STAT1c] \rightarrow [IFNRJ2^*]+[STAT1c^*]$; k_6
$[IFNRJ2^*]+[STAT1c^*] \leftrightarrow [IFNRJ2*-STAT1c^*]$; k_7, k_{-7}
$2[STAT1c^*] \leftrightarrow [STAT1c^*-STAT1c^*]$; k_8, k_{-8}
$[IFNRJ2^*]+[SHP-2] \leftrightarrow [IFNRJ2^*-SHP-2]$ k_9, k_{-9};
$[IFNRJ2^*-SHP-2] \rightarrow [IFRNRJ2]+[SHP-2]$; k_{10}
$[PPX]+[STAT1c^*] \leftrightarrow [PPX-STAT1c^*]$; k_{11}, k_{-11}
$[PPX-STAT1c^*] \rightarrow [PPX]+[STAT1c]$; k_{12}
$[PPX]+[STAT1c^*-STAT1c^*] \leftrightarrow [PPX-STAT1c^*-STAT1c^*]$; k_{11}, k_{-11}
$[PPX-STAT1c^*-STAT1c^*] \rightarrow [PPX]+[STAT1c-STAT1c^*]$; k_{12}
$[STAT1c]+[STAT1c^*] \leftrightarrow [STAT1c-STAT1c^*]$; k_{13}, k_{-13}
$[STAT1c^*-STAT1c^*] \rightarrow [STAT1n^*-STAT1n^*]$; k_{14}
$2[STAT1n^*] \leftrightarrow [STAT1n^*-STAT1n^*]$; k_7, k_{-7}
$[PPN]+[STAT1n^*] \leftrightarrow [PPN-STAT1n^*]$; k_{15}, k_{-15}
$[PPN-STAT1n^*] \rightarrow [PPN]+[STAT1n]$; k_{16}
$[PPN]+[STAT1n^*-STAT1n^*] \leftrightarrow [PPN-STAT1n^*-STAT1n^*]$; k_{15}, k_{-15}
$[PPN-STAT1n^*-STAT1n^*] \rightarrow [PPN]+[STAT1n-STAT1n^*]$; k_{16}
$[STAT1n]+[STAT1n^*] \leftrightarrow [STAT1n-STAT1n^*]$; k_{13}, k_{-13}
$[STAT1n] \rightarrow [STAT1c]$; k_{17}
$[STAT1n^*-STAT1n^*]+DNA \rightarrow [mRNAn]$; k_{18}
$[mRNAn] \rightarrow [mRNAc]$; k_{19}
$[mRNAc] \rightarrow [SOCS1]$; k_{20}
$[SOCS1]+[IFNRJ2^*] \leftrightarrow [SOCS1-IFNRJ2^*]$; k_{21}, k_{-21}
$[mRNAc] \rightarrow \phi$; k_{22}
$[SOCS1] \rightarrow \phi$; k_{23}
$[STAT1c]+[SOCS1-IFNRJ2^*] \leftrightarrow [STAT1c-SOCS1-IFNRJ2^*]$; k_5, k_{-5}
$[SHP-2]+[STAT1c-SOCS1-IFNRJ2^*] \leftrightarrow [SHP-2-STAT1c-SOCS1-IFNRJ2^*]$; k_9, k_{-9}
$[SHP-2-STAT1c-SOCS1-IFNRJ2^*] \rightarrow [SOCS1]+[IFNRJ2]+[STAT1c]+[SHP-2]$; k_{10}
$[SHP-2-STAT1c-SOCS1-IFNRJ2^*] \rightarrow [SHP-2-STAT1c-IFNRJ2^*]$; k_{23}

space and time. For performance comparisons, the JAK-STAT signaling pathways was initialized with 1.23 million

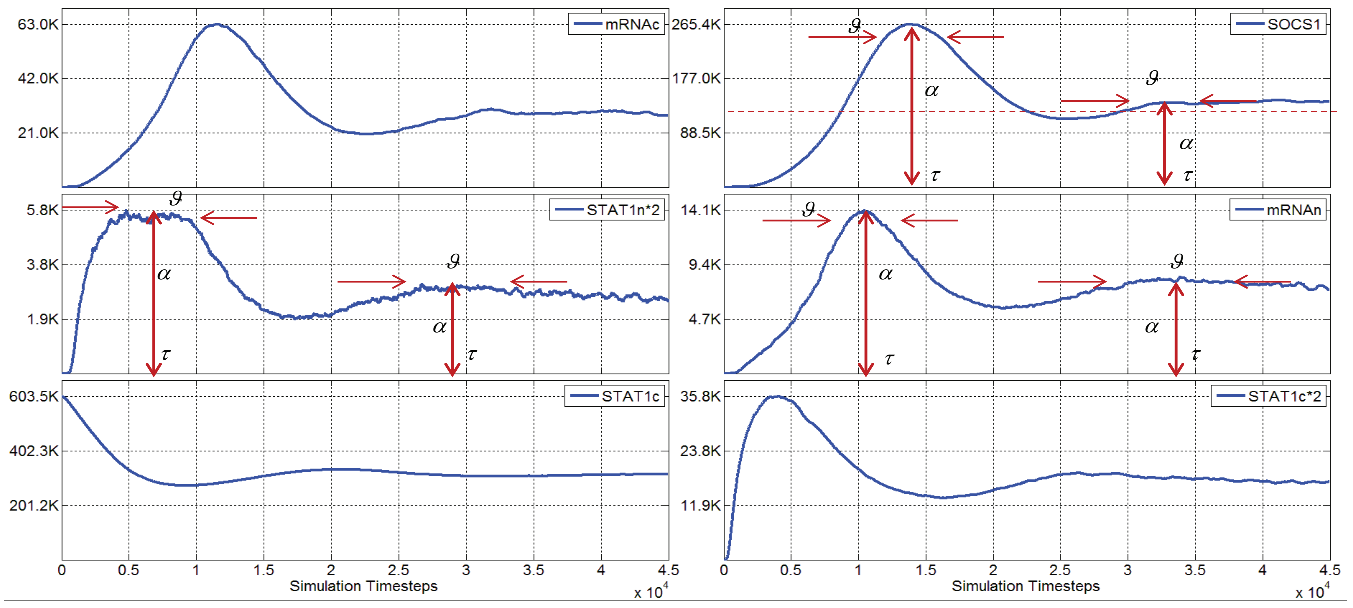

Figure 8. Particle concentration of important bio-molecular types(in thousands of particles) within the JAK-STAT signaling pathway. The annotations refer to biologically relevant quantities such as, time(τ) to peak value(α), and peak duration(ϑ).

Table III
KINETIC PARAMETERS

Parameter Value	Parameter Value
$k_1 = 100, k_{-1} = 0.05$	$k_2 = 20, k_{-2} = 0.02$
$k_3 = 40, k_{-3} = 0.2$	
$k_4 = 0.005$	$k_5 = 8, k_{-5} = 0.8$
$k_6 = 0.4$	$k_7 = 5, k_{-7} = 0.5$
$k_8 = 20, k_{-8} = 0.1$	$k_9 = 1, k_{-9} = 0.2$
$k_{10} = 0.003$	$k_{11} = 1, k_{-11} = 0.2$
$k_{12} = 0.003$	$k_{13} = 0.0002, k_{-13} = 0.2$
$k_{14} = 0.005$	$k_{15} = 1.0, k_{-15} = 0.2$
$k_{16} = 0.005$	
$k_{17} = 0.05$	$k_{18} = 0.01$
$k_{19} = 0.001$	$k_{20} = 0.01$
$k_{21} = 20, k_{-21} = 0.1$	
$k_{22} = 0.0005$	$k_{23} = 0.0005$

Table IV
FUNCTION PERFORMANCE

Kernel function	GPU (ms)	GPU+FPGA(ms)
`CleanArray`	35	35
`SortFeasible`	5	5
`ExecuteReaction`	1	1
`UpdatePosition`	6	6
`BuildCell`	12	12
`UpdateCell`	5	4
`NeighbourBuild`	1423	230
`CheckConstraints`	2	2
Total time	1510	295

particles or independent reacting agents, within a simulation space of 200×200×200 distance units. The system was simulated for a period of 45,000 timesteps while recording the trajectory and concentration of various chemical species. The preliminary results shown in Fig(8) agree qualitatively with those obtained via the ODE based framework([11], [12]). The results show change in particle concentrations of unactivated and activated transcription factor (STAT1c, STAT1c*2), mRNA within the nucleus and cytoplasm and the feedback protein SOCS1, with the total number well over a million particles. The annotations show biologically relevant parameters, such as time to peak, duration of peak and the peak value. For performance comparisons, we set different initial number of particles(independent agents) for this system from 10,000 to 1.35 Million in order to measure the average time-per-step. In Fig(9), we compare the performance of GPU and FPGA implementation of the compute intensive task of calculating the `FeasibleList`. The FPGA achieves approximately 5 x speedup over GPU-only implementation for all system sizes while using the 32 bit floating point to maintain simulation quality.

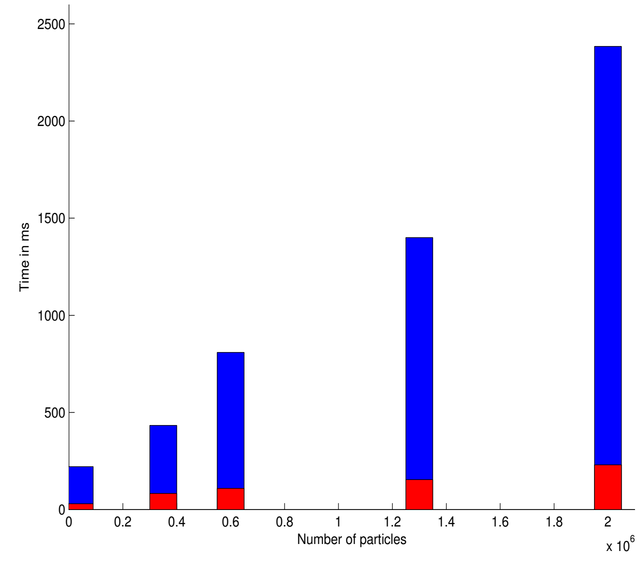

Figure 9. Performance of the kernel with respect to the total number of particles (40k, 350k, 600k, 1.3M and 2M independent agents). The GPU-FPGA performance is shown in red bars while the GPU only performance is shown in blue.

IV. Discussion

In this paper we present a novel heterogeneous computational framework to study a large and complex biological pathways in 3D process space. The explicit simulation of the pathways in 3D space offers deeper understanding than simple ODE or PDE based models. The ODE based model assume the reagents to be well mixed while ignoring boundary effects, while the PDE based model completely ignore the discrete effects. For example within the nucleus there are only a few discrete copies of the DNA which cannot be captured by PDEs as they assume continuum of concentration values. Furthermore, DNA transcription is a highly discrete process as it is first unfolded and undergoes chromatin dynamics. Such effects can be accurately captured by process simulation and the above framework aims to pave the way for routine simulation of large and complex pathways. To that end we present a computational framework capable of handling large and complex pathways via a heterogeneous platform. The GPU-FPGA implementation shows five times speedup over our original single GPU-only platform. Besides, the extensible architecture is general enough to be used to study a variety of computationally intensive systems. The future work will include a variety of optimization strategies for both GPU and FPGA and application to other biomolecular systems.

V. 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 this research.

References

[1] D. T. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, Dec. 1977.

[2] D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*. Methuen, London, 1965.

[3] A. Phillips and L. Cardelli, “Efficient, correct simulation of biological processes in the stochastic pi-calculus,” in *Computational Methods in Systems Biology*, ser. LNCS, vol. 4695. Springer, September 2007, pp. 184–199.

[4] J. Li, V. Sharma, N. Ganesan, and A. Compagnoni, “Simulation and study of large-scale bacteria-materials interactions via bioscape enabled GPUs,” in *Proceedings of ACM-BCB 2012*, 2012.

[5] M. Taufer, N. Ganesan, and S. Patel, “GPU enabled macromolecular simulations: Challenges and opportunities,” *IEEE Computing in Science and Engineering*, vol. 15, no. 1, Jan 2012.

[6] R. Erban and S. J. Chapman, “Stochastic modelling of reaction-diffusion processes: algorithms for bimolecular reactions,” *Physical Biology*, vol. 6, no. 046001, 2009.

[7] L. Verlet, “Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules,” *Physical Review*, vol. 159, no. 1, pp. 98–103, 1967.

[8] J. Li, A. Salighehdar, and N. Ganesan, “Simulation of complex biochemical pathways in 3D process space via heterogeneous computing platform: Preliminary results,” in *Intl. Symp. on Applied Reconfigurable Computing*, April 2014.

[9] M. Chiu and M. C. Herbordt, “Molecular dynamics simulations on high-performance reconfigurable computing systems,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 3, no. 4, pp. 23:1–23:37, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1862648.1862653

[10] Y. Gu, T. VanCourt, and M. C. Herbordt, “Explicit design of fpga-based coprocessors for short-range force computations in molecular dynamics simulations,” *Parallel Computing*, vol. 34, no. 45, pp. 261 – 277, 2008, reconfigurable Systems Summer Institute 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819108000215

[11] S. Yamada, S. Shiono, A. Joo, and A. Yoshimura, “Control mechanism of JAK/STAT signal transduction pathway,” *FEBS Letters*, vol. 534, pp. 190–196, 2003.

[12] A. Gambin, A. Charzyska, A. Ellert-Miklaszewska, and M. Rybiski, “Computational models of jak1/2-stat1 signaling,” *JAK-STAT*, vol. 2, 2013.