Improving the Simulation Performance of Colored Hybrid Petri Nets by the Graphics Processing Units

Ismail ELansary
Department of Mathematics and Computer Science,
Faculty of Science, Port Said University,
42521 - Port Said, Egypt
Email: ielansary@sci.cu.edu.eg

Mostafa Herajy
Department of Mathematics and Computer Science,
Faculty of Science, Port Said University,
42521 - Port Said, Egypt
Email: mherajy@sci.psu.edu.eg

Alaa Hefnawy
Department of Computers and Systems,
Electronics Research Institute
Giza, Egypt
Email: alaa@eri.sci.eg

Nasser Sewilam
Department of Mathematics,
Faculty of Science, Cairo University,
Giza, Egypt
Email: nsweilam@sci.cu.edu.eg

***Abstract*—**

The most substantial aim in systems biology is studying and understanding biological phenomena at the system level. Toward achieving this goal, it is imperative to construct and execute accurate models to predict the behaviour of the underlying system and get more insights about the interactions between the different model components. Coloured Petri nets are a promising tool to model such biological systems, which extend the power of Petri nets by assigned colours to places. However, the sequential implementation of coloured Petri net execution is prohibitively slow. Particularly, when complex models are considered that may contain reactions or species at different scales. In this paper, we focus on a specific class of Petri nets called coloured hybrid Petri nets ($\mathcal{HPN}^{\mathcal{C}}$), which can combine discrete and continuous components at the same model. However, speeding up the simulation of $\mathcal{HPN}^{\mathcal{C}}$ models is of a paramount importance. One direction to release this goal is to resort to parallel processing. In this paper, we use the Graphics Processing Units (GPU) to increase the efficiency of simulating coloured hybrid models, whereby the time-extensive part (the stochastic regime) is simulated on the GPU, while the deterministic simulation is kept running on the CPU. Besides, we compare the performance of the improved approach with the sequential implementation.

***Keywords*—Hybrid Petri Net; Colored Petri Net; Parallel Simulation; GPU; CUDA**

I. Introduction

Due to the current production of a huge amount of biological data in the biological field, we need to resort to formal methods handling the modelling of such systems to understand, analyse, and describe the corresponding biological system. Therefore, many modelling approaches like Petri nets, boolean networks, and ordinary differential equations (ODEs) have already been applied to construct and execute models for a wide range of biological systems [1], [2].

Among these tools, Petri nets [1] provide a graphical and mathematical approach that can describe and analyse biological systems (see e.g., [2], [3], [4], [5], [6], [7], [8], [9]). Moreover, Petri nets are used to describe complex systems and graphically simplify its understanding in a more readable way. Technically, Petri nets consist of two types of nodes: places and transitions. They are connected to each other by arcs (sometimes are also called edges). Places usually represent passive system components like conditions or resources, while transitions represent active system components like events (see [1] for some examples). A place may contain a number of tokens called marking, represented as black dots or as a natural number. Furthermore, there are a wide range of Petri net classes like stochastic Petri nets ($\mathcal{SPN}$) [10], [11], continuous Petri nets ($\mathcal{CPN}$) [12], hybrid Petri nets ($\mathcal{HPN}$) [12], [13] and coloured Petri nets ($\mathcal{PN}^{\mathcal{C}}$) [14], have been developed for modelling and analysing different types of biological systems [3].

Stochastic Petri nets ($\mathcal{SPN}$) are an extension of qualitative Petri nets ($\mathcal{QPN}$) which assign to transitions exponentially distributed waiting times, which are specified as firing rates to transitions. The execution of stochastic Petri nets ($\mathcal{SPN}$) is very slow, particularly for models with huge number of states. Therefore, continuous Petri nets ($\mathcal{CPN}$) are developed to deal with systems having large numbers of molecules. The semantics of $\mathcal{CPN}$ is usually described by a systems of ordinary differential equations (ODEs) that represents the firing of a set of reactions [15].

Moreover, hybrid Petri nets ($\mathcal{HPN}$) combines both stochastic Petri nets and continuous Petri nets. $\mathcal{HPN}$ integrates all functionalities of $\mathcal{SPN}$ and $\mathcal{CPN}$ into one class. Moreover, $\mathcal{HPN}$ deal with biological with biological reactions that take place at different time scales. Besides, $\mathcal{HPN}$ controls the accuracy and the speed of biological models during simulation [2], [15].

A further extension of Petri nets are Coloured Petri nets which are suitable for studying bigger and complex biological

models as they simplify the modelling process by representing each group of repeated components with one component but with different colours. In other words, coloured Petri nets are more appropriate for modelling model components which having same structure but with different marking [13], [16].

Similarly, studying complex and larger biological systems need to employ discrete and continuous variables as well as continuous and stochastic processes in the same model [2], [15]. Therefore, hybrid modelling and simulation is ideal to deal with these challenges. However, the current implementation of hybrid simulation is is not very fast for big models especially when a stochastic part is present [17]. In this scenario, processing cycles are mainly consumed by the repeatative firing of stochastic reactions.

In this paper we propose the use of the graphical processing units to boost the performance of simulation of coloured hybrid Petri nets for employed for biological applications. We design an efficient parallel algorithm that offload the execution of the stochastic simulation from the main CPU while taking advantage of the computational power of the GPU. In our proposal, we kept the deterministic part executing on the CPU as it does not represent a big challenge to the overall hybrid simulation.

This paper is organised as follows: first we briefly summarise the steps of the hybrid simulation of biological systems. Next, we introduce a parallel algorithm to simulate coloured hybrid Petri net model using a hybrid CPU/GPU technique. Afterwards, we discuss the performance of our proposed approach. Finally, we conclude with final remarks and future work.

II. Background

In this section, we briefly discuss related work to the hybrid simulation as well as parallel processing.

A. The Hybrid Simulation Algorithm

Consider a well stirred system of N molecular species $\{S_1, S_2, ..., S_N\}$ which interacting through M chemical reaction channels $\{R_1, R_2, ..., R_M\}$. The system in a container of fixed volume V and in thermal balance. The system's state at any time t is described by an N-vector $X(t) = \{X_1(t), X_2(t), ..., X_N(t)\}$ where $X_i(t)$ denotes to the number of molecules of species S_i at time t [18], [19].

Beginning from the initial state $X(t_0)$ the aim is to find an estimated evolution of the vector X during the time t. Generally, there are two broad formal approaches to mathematically find the change in the state vector: the deterministic approach and the stochastic one. If the system contains species with large numbers of molecules then a set of ODEs can represent the evolution of the system with respect to time and is given as follows:

$$\frac{d[S_i]}{d_t} = f_i([S_1], ..., [S_N]) \tag{1}$$

where $[S_i]$ refer to the concentration of species S_i and $f_i([S_1], ..., [S_N])$ is the rate of change of S_i [2], [15].

However, if the system contains species with low numbers of molecules, then the deterministic simulation will not result in accurate model behaviour [15]. Therefore, we can simulate the model at the molecular level by using stochastic simulation, which takes into consideration the stochastic nature of chemical reactions. Gillespie [18] introduces two simulation algorithms which is derived from the Monte Carlo framework to simulate Markov systems. One of them is the first reaction method, which begins by generating M random numbers $r_1, ..., r_M$ from the unit-interval uniform distribution $(0, 1)$, and computing

$$\tau_{j'} = \frac{1}{a_{j'}(x)} \ln\left(\frac{1}{r_{j'}}\right) \quad (j' = 1, ...M) \tag{2}$$

then it takes

$$\left.\begin{array}{ll} \tau = & the\, smallest\, of\, the\, \tau_{j'} \\ j = & the\, index\, of\, the\, smallest\, \tau_{j'} \end{array}\right\}$$

where $\tau_1, ..., \tau_M$ are putative times to the next firings of the respective reaction channels and $a_{j'}(x)$ is the propensity of reaction $R_{j'}$ at a state $X(t) = x$. Afterwards, the chosen reaction is fired and the system state is updated. This steps are repeated till the simulation end time is reached [18], [19].

However, both stochastic and deterministic methods are not suitable to simulate stiff and complex models due to the mentioned reasons. In this case we require a simulation method that combines the merits of the deterministic and stochastic simulation techniques. The ideal choice in such cases is a hybrid simulation [15].

Due to the incorporation of both fast and slow reactions in the hybrid simulation approach, the propensities of the stochastic reactions rely on the change of state of continuous simulated reactions. Gillespie [18], [19] derived the reaction probability function as

$$p(\tau, j|x, t) = a_j(x)\, exp(-a_0(x)\tau) \tag{3}$$

Therefore a hybrid simulator can switches between stochastic and continuous submodules by using (4). In other words, it integrates a system of ODEs representing the deterministic reactions along with the summation of the propensity, $a_0^s(x)$

$$g(x) = \int_t^{t+\tau} a_0^s(x) d_t - \xi = 0 \tag{4}$$

where ξ is a random number from $U(0, 1)$, and $a_0^s(x)$ is the summation propensity of stochastic reactions. The stochastic event takes place when (4) is satisfied.

B. Hybrid Petri Nets

$\mathcal{HPN}$ play an important role in constructing and executing hybrid models, principally large and complex systems, where systems can contain discrete as well as continuous components. Moreover, it is an appropriate graphical and mathematical tool to simulate and model multi-time scale biological systems. $\mathcal{HPN}$ can combine stochastic and deterministic processes in the same model, where the stochastic simulation can be used to take care of the fluctuation and discreteness features and simultaneously, the deterministic parts can be employed to speed up the simulation by constructing and numerically solving a system of ODEs [2], [15].

$\mathcal{HPN}$ contain many types of elements: places (continuous and discrete), transitions (stochastic, continuous, scheduled,

immediate and deterministic) and arcs which connect between places and transitions [15].

The number of molecules of a given species represented by non-negative integer numbers in the discrete places. Otherwise, continuous places carry non-negative real numbers which represent the concentration of the corresponding species. Moreover, stochastic transitions, which represented by a single line square, fire randomly with exponentially distributed random waiting time. Immediate transitions, which represented by a black bar, fire with no delay. Deterministically delayed transitions, which represented by black squares, fire after a certain constant time. Finally, continuous transitions represented by shaded line square and fire continuously. Their semantics are administered by a set of ODEs which acquaint the changes in the transition's pre- and post-places. [2], [15].

A set of different arcs are used to connect the two types of nodes (places and transitions). There are six types of arcs inhibitor, read, standard, reset, equal, and modifier arcs.

C. Colored Hybrid Petri Nets

Coloured Petri nets ($\mathcal{PN}^{\mathcal{C}}$), are a coloured and sturdy extension of normal Petri nets, where one component represents a group of similar components, each component is distinguishable with distinctive colour. Coloured Petri nets introduce a simple representations of complex biological systems. Moreover, Colored Petri nets enable us to increase the size of a model consisting of many similar subnets only by adding new colours [3].

Notwithstanding, biological systems models which have to be studying and analysing, rapidly change in its size and types. Therefore, coloured hybrid Petri nets ($\mathcal{HPN}^{\mathcal{C}}$) are a Petri net class that integrates all the features of hybrid Petri nets and coloured Petri nets into one class. Therefore, colored hybrid Petri nets are useful for modelling complex and larger biological systems [2], [3].

D. Parallel Processing

Parallel processing is an efficient technique playing a leading role in reducing computational time. It attempts to enhance the performance of sequential algorithms by executing multiple processes simultaneously and dividing tasks among multiple processors. The created processes executes these subtasks simultaneously. Therefore, it confirmed to be an effective tactic which can be employed to reduce the overall computational time and minimise the required size of memory space [20].

Nevertheless, there exist various architecture and different programming models that can be adapted various implementations. Such models can be classified into four categories as follow [20], [21]:

- Single Instruction, Single Data stream (SISD) refer to one processor that execute one instruction stream at the time.
- Single Instruction, Multiple Data streams (SIMD) have a single control unit (CU) and multiple processing units (PU) which execute a single instruction stream over many PUs.
- Multiple Instruction, Single Data stream (MISD) where numerous functional units execute various processes by performing diverse directives on the same data.
- Multiple Instruction, Multiple Data streams (MIMD) it is like computers with multiprocessors, with each processor can do asynchronously group of instructions.

In the following, we discuss in more details a few specialised models of parallel processing related to our paper topic.

1) Graphics Processing Units: Recently, one of the most important hardware components of computer architecture is the Graphics Processing Units (GPUs), which were at first designed for accelerating the performance of 3-D images for the real-time show. Later on, GPUs are used as parallel processing tools because it can contain hundreds of cores. Moreover, GPU is multithread, highly parallel, multiple core processors and has a high memory bandwidth to use to solve the computational problems. Over the recent few decades, the GPUs computational performance has rapidly increased more than that of common CPUs [20].

2) CUDA: In 2006, NVIDIA has presented its own massively parallel architecture called compute unified device architecture (CUDA) which made a revolution in GPU programming approach. CUDA is an open-source parallel processing architecture and uses NVIDIA GPU parallel compute engine to solve massively computational problems. Moreover, CUDA is based on the C programming language [20].

The CUDA programming interface is a group of additions to the C language, which allows targeting kernel code, and a runtime library. The CUDA N diverse threads is implemented in a SIMD manner when a kernel is launched. Indeed, CUDA kernels carry out as a net of thread blocks. Threads in a block can communicate and share data with each other through shared memory and can coordinate memory accesses by synchronising their execution. Moreover, CUDA enables the developers to assign the number of threads for each block and the number of blocks as arguments to the execution configuration [20].

3) Unified Memory: Beginning from CUDA 6, Nvidia presented one of the most significant parallel programming architecture refinements in CUDA platform, Unified Memory. Recently, in cluster or nodes, the memories of GPU and CPU are detached by the PCI-Express bus. On the other hand, Unified Memory establishes a rally of managed memory which can be accessed from both CPU and GPU. Moreover, it appears like CPU memory to code running on the CPU and GPU memory to code running on the GPU. Figure 1 presents a schematic digram of the Unified Memory model.

III. Materials and Methods

A. Parallel Simulation Algorithm

In this section, we introduce the proposed method of parallelising the hybrid simulation by the help of the GPU. As we mentioned earlier, the hybrid model combines fast and slow reactions. The stochastic simulation is usually used to execute slow part. Therefore, in our parallel model, the stochastic simulation is selected to be executed on the GPU

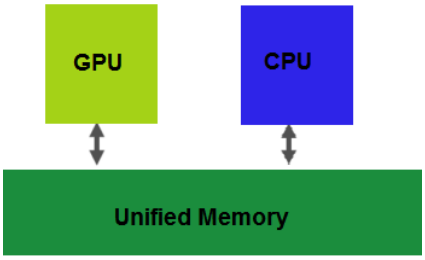

Fig. 1. CUDA Unified Memory accessed from both CPU and GPU

and the deterministic simulation is kept running on the CPU. Moreover, We use the CUDA Unified Memory concept to reduce the data transfer between the CPU and the GPU. Due to the memory resources limit, the number of threads per block is restricted. Thus, the number of threads per block should be multiple of 32 threads (e.g., 64,128,256) to optimise the GPU memory usage.

In the following, we illustrate how such hybrid models can be simulated via our hybrid architecture.

Algorithm 1 presents the steps used to simulate the hybrid model on the GPU. Beginning from an initial marking which corresponds to the initial state of a biological system, the algorithm calculates the change in state vector over the time, representing as the current marking m(CurrentTime).

At the beginning, the current marking is set to the initial marking. Moreover, the net reactions are split among the CUDA kernel blocks and the propensities $a(t_j)$ are computed for the stochastic and deterministic transition (lines 3-4), the integration of stochastic total propensity is calculated in line 5. All data which calculated is stored in CUDA Unified Memory to enable access for GPU and CPU.

If there is non stochastic transition to fire the algorithm will integrate the ODEs until the event E occur. Where, the integration will stop if the event E occur. We employ the SUNDIALS CVODE library to perform the ODEs integration [15], [6].

Line 11 updates all the propensities of the transitions which share a pre-place with a deterministic transition.

In lines 12-17 a stochastic transition is chosen to fire when the event occurs. The process of selecting a reaction and update the net run on the GPU, where we split transitions between blocks and each block is assigned a task.

Each block computes the local minimum τ and its corresponding reaction index. Line 18 fire the selected reaction and update its propensity and the propensities of any other transitions that are affected.

The simulation terminates when the current simulation time overrides the simulations end time, which is given by the user.

B. Case Study

In this section, we assess the performance of the proposed algorithm by comparing it with the sequential one. We select a case study to evaluate the algorithm. We use coloured hybrid Petri net ($\mathcal{HPN}^{\mathcal{C}}$) model to permit the fast change of the net size and illustrate the performance difference when we increase

Algorithm 1 Parallel Simulation of hybrid Petri net

1: Current time$\leftarrow$ 0;
2: $\xi \leftarrow exp(1)$generate random number U(0,1) using the cuRAND CUDA library;
3: $m(Current\ time) \leftarrow m(0)$;
4: Split the reaction among the CUDA kernels blocks and calculate the individual propensities $a(t_j)$;
5: calculate the cumulative propensity a_0 Using the GPU;
6: **while** while CurrentTime $<$ EndTime **do**
7: **if** There are non-stochastic transitions **then**
8: CurrentTime = the current integrator time;
9: Initialise the ODE solver by m(CurrentTime);
10: Integrate the system of ODEs generated using (1) and g(m(CurrentTime)) until an event E occurs using the CPU;
11: Update $(a(t_i), a_0)$;
12: **if** Event E occured **then**
13: $m(Current\ time) \leftarrow m(0)$;
14: exp(1);
15: Split the reactions between CUDA kernels blocks;
16: Each block computes its local minimum τ and its corresponding reaction index;
17: Find the global minimum τ and reaction index;
18: Fire the chosen transition;
19: Update $a(t_i)$;
20: **else if** CurrentTime $>=$ EndTime **then**
21: terminate;
22: **end if**
23: **end if**
24: **end while**

the model size. The $\mathcal{HPN}^{\mathcal{C}}$ model represents spatial calcium dynamics [6]. The model tracks changes in cytoplasmic and ER intracellular calcium concentrations. We can changed the size of the model easily by changing the colour set and unfolded the coloured model to a low level version. Therefore, we will have different net size depending on the selected coloured set. We changed the colour definition ten times and unfolded the colour model to have different net size. The model is adopted from [6].

IV. Results and Discussion

The evaluation of parallel simulation algorithm have been conducted with HP Pavilion Sleekbook 15-b004sg Ultrabook with Intel's Core i5-3317U dual-core processor and NVIDIA GeForce GT 640M GPU.

Table I lists the different experiment outcomes. We utilise models of different sizes ranging from 10 to 200000 places. For small models the speedup is not significant. As the model size increased, the speed up ratio between the pure sequential implementation and the one implemented by GPUs also starts to increase. This data is also visualised in Figure 2.

V. Conclusions and Future Work

In this paper, we have introduced an efficient algorithm to improve the hybrid simulation by using the potential of parallel simulation. The presented algorithm use the Unified Memory concept to reduce the data transfer and communication between the host (CPU) and the device(GPU). Moreover,

TABLE I. COMPARISON OF EXCUTION TIME (IN SECONDS) AND SPEEDUPS OF SEQUENTIAL VS PARALLEL VERSION OF HYBRID SIMULATION ALGORITHM CONSIDERING DIFFERENT SIZES OF THE MODEL.

Net size	Serial time(s)	parallel time(s)	Speedup
10	0.8	0.6	1.33
100	3.8	0.84	4.5
500	7.45	0.9	8.2
1000	13.88	1.23	11.3
5000	29.3	2.1	14
10000	41.15	2.92	14.1
50000	50.2	3.23	15.5
100000	65.15	4.15	15.7
150000	69.61	4.23	16.5
200000	73.2	4.32	17

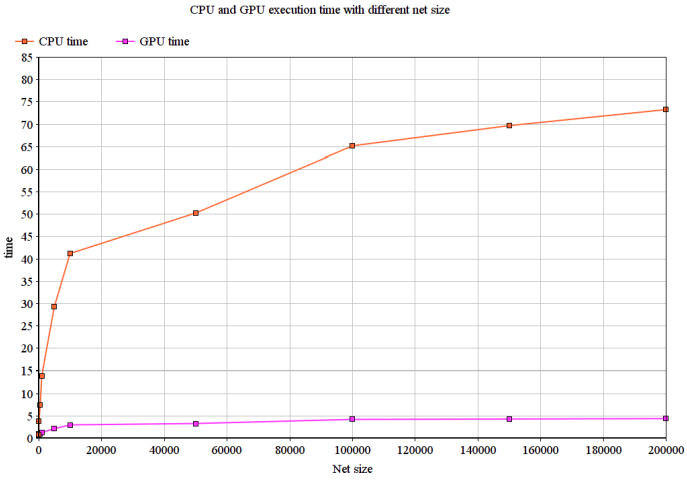

Fig. 2. A relationship between the sequential and the parallel performance when the net size is increased.

we have implemented the proposed algorithms, using CUDA and the standard C++ language.

There is a possible enhancement to the presented algorithm, whereby slow and fast reaction can also be simulated on the GPU.

REFERENCES

[1] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[2] M. Herajy, M. Schwarick, and M. Heiner, *Hybrid Petri Nets for Modelling the Eukaryotic Cell Cycle*. Springer Berlin Heidelberg, 2013, ch. ToPNoC VIII, pp. 123–141.

[3] M. Herajy, F. Liu, and C. Rohr, "Coloured hybrid Petri nets for systems biology," in *Proc. of the 5th International Workshop on Biological Processes & Petri Nets (BioPPN)*, vol. 1159, 2014, pp. 60–76.

[4] M. Herajy and M. Heiner, "Adaptive and bio-semantics of continuous Petri nets: Choosing the appropriate interpretation," *Fundamenta Informaticae*, vol. 160, pp. 53–80, 2018.

[5] M. Herajy, F. Liu, and M. Heiner, "Efficient modelling of yeast cell cycles based on multisite phosphorylation using coloured hybrid Petri nets with marking-dependent arc weights," *Nonlinear Analysis: Hybrid Systems*, vol. 27, pp. 191 – 212, 2018.

[6] M. Herajy, F. Liu, C. Rohr, and M. Heiner, "Snoopy's hybrid simulator: a tool to construct and simulate hybrid biological models," *BMC Systems Biology*, vol. 11, no. 1, p. 71, Jul 2017. [Online]. Available: https://doi.org/10.1186/s12918-017-0449-6

[7] M. Heiner, I. Koch, and K. Voss, "Analysis and simulation of steady states in metabolic pathways with Petri nets," in *Proc. of the 3rd Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*. University of Aarhus, 2001, pp. 15–34.

[8] M. Matsui, S. Fujita, S. Suzuki, H. Matsuno, and S. Miyano, "Simulated cell division processes of the xenopus cell cycle pathway by genomic object net," *Journal of Integrative Bioinformatics*, p. 0001, 2004.

[9] S. Fujita, M. Matsui, H. Matsuno, and S. Miyano, "Modeling and simulation of fission yeast cell cycle on hybrid functional Petri net," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E87-A, no. 11, pp. 2919–2927, 2004.

[10] M. Ajmone, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, John Wiley and Sons, 1995.

[11] M. Heiner, S. Lehrack, D. Gilbert, and W. Marwan, "Extended stochastic Petri nets for model-based design of wetlab experiments," in *Transactions on Computational Systems Biology XI*. Berlin, Heidelberg: Springer, 2009, pp. 138–163.

[12] H. Alla and R. David, "Continuous and hybrid Petri nets," *J. Circ. Syst. Comp.*, vol. 8, no. 1, pp. 159 –188, 1998.

[13] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2010.

[14] K. Jensen, "Coloured Petri nets and the invariant-method," *Theoretical Computer Science*, vol. 14, no. 3, pp. 317–336, 1981.

[15] M. Herajy and M. Heiner, "Hybrid representation and simulation of stiff biochemical networks," *Nonlinear Analysis: Hybrid Systems*, vol. 4, pp. 942–959, 6 2012.

[16] A. Ismail, M. Herajy, and M. Heiner, *A Graphical Approach for the Hybrid Modelling of Intracellular Calcium Dynamics Based on Coloured Hybrid Petri Nets*. Springer, 2018, to appear.

[17] M. Herajy and M. Heiner, "Accelerated simulation of hybrid biological models with quasi-disjoint deterministic and stochastic subnets," in *Hybrid Systems Biology: 5th International Workshop, HSB 2016, Grenoble, France, October 20-21, 2016, Proceedings*, E. Cinquemani and A. Donzé, Eds. Cham: Springer International Publishing, 2016, pp. 20–38.

[18] D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, no. 25, pp. 2340–2361, 1977.

[19] ——, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, 2007.

[20] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Upper Saddle River, NJ: Addison-Wesley, 2010.

[21] *Professional CUDA C Programming*, 1st ed. Birmingham, UK, UK: Wrox Press Ltd., 2014.