Accelerating Action Potential Generation Using GPU Implementation of a Resonant Model of a Cell

Sucheta Sehgal[1], Saif Charania[1], Jonathan Reshef[1], Nitish D Patel[1], Mark L Trew[2]

[1] The University of Auckland, Auckland, New Zealand
[2] Auckland Bioengineering Institute, Auckland, New Zealand

Abstract

Computational modelling of bioelectric phenomena in cardiac tissue has made significant contributions to our understanding of the electrical activity of the human heart. However, extending these simulations over large temporal and spatial ranges is an inherently computationally expensive task. Motivated by the apparent advantages of GPU over CPU in many areas of science, we have developed a new Resonant Model (RM) of an AP of a biological cell amenable to parallel computing. In this study, we have investigated the potential performance benefits of the implementation of the RM on a GPU over the sequential execution on a CPU. In the proposed implementation, all threads in the same GPU warp share data using a warp-shuffle operation, which eliminates the access of global or shared memory. By exploiting the memory model and computational strengths of GPU, the obtained performance of the RM on the GPU is far superior to that of the CPU. On the GPU, the RM demonstrate linear or sub-linear growth in the execution times with the increase in the number of cells. The speedup of the computations achieved on the GPU may, in the near future, facilitate the patient-specific electrophysiological studies and treatment planning.

1. Introduction

Cardiac cell models at different levels of detail for various species have been developed to: understand cardiac bioelectric phenomena (e.g. tissue response to a stimulus or formation of arrhythmias), test hypotheses, plan treatment through patient-specific electrophysiologic studies and validate medical devices [1–3]. A human heart has around 2 x 10^9 cells [4], and detailed model of a cardiac cell can include tens to hundreds of coupled non-linear differential equations with various levels of arithmetic intensity and control complexity. New simulations at the tissue or organ level with detailed cell models are now becoming computationally feasible and tractable only due to the increased availability of computing power. However, this limits the large-scale cardiac simulations to supercomputers or CPU clusters.

Over the last decade, for many areas in system biology, including cardiac electrical dynamics GPU performance has exceeded that of CPU [5]. GPUs have narrowed the gap between the execution of an in-silico model of a biological system and the functioning rate of the actual biological system. However, the notional goal of real-time solution has been elusive. For example, the double-precision 2D GPU simulation of 2^{18} gridpoints of the simplified Karma model (2 variables) of cell electrophysiology [6] was approximately two times slower than real-time. The GPU simulation of detailed model (67 variables) [7] was 1260 times slower than the real-time [8].

Motivated by the computational power of the GPU and to move closer to real-time computations, we have developed a new methodology for modeling an AP of biological cells in our previous work [9]. The cell model developed with this methodology is known as a Resonant model (RM). In the present study, we make the following contributions:

1. We propose a resource-efficient implementation of the sinusoidal terms in the RM.
2. We propose an optimized implementation of the RM on a GPU.
3. We show the acceleration of the RM with GPU implementation as compared to the CPU implementation.

2. Resonant model

The RM describes the electrical behavior of biological cells via simplified in silico representation of their membrane potential ($V(t,\beta)$) by using truncated trigonometric series in Equation (1).

$$V(t,\beta) = a_0(\beta) + \sum_{i=1}^{n} c_i(\beta)\cos(i\omega(\beta)t - \phi_i(\beta)) \quad (1)$$

Here, a_0, c_i, ω, and ϕ_i are the dynamic parameters of

 ISSN: 2325-887X DOI: 10.22489/CinC.2019.214

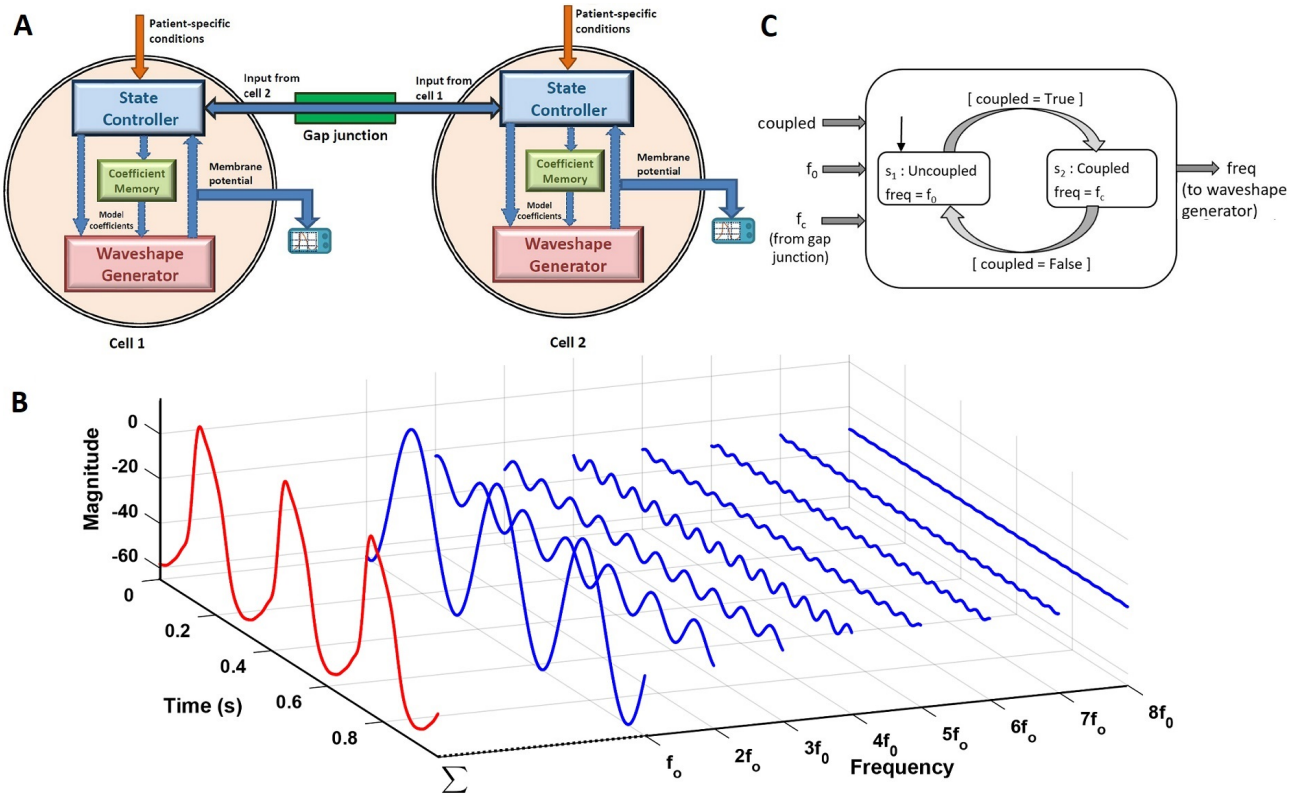

Figure 1. A: Schematic representation of the Resonant model cells (Cell 1 and Cell 2) coupled with a gap junction. B: The contribution of the eight oscillators in the waveshape generator (blue traces) to produce an AP waveshape (red trace) of a sinoatrial node cell. C: Diagram of the state controller of the Resonant model of a sinoatrial node cell. f_0 is the intrinsic beating frequency of a cell and f_c is the entrained frequency.

the RM. The methodology for obtaining the model parameters is detailed in previous work [9]. These parameters change with altered operating condition of a cell (represented as β), such as stimuli from the neighboring cells. The parameters are stored in the coefficient memory of the RM as shown in the Figure 1A (from [9]) and are input to the waveshape generator (WG) to produce AP waveshape. A WG is a set of n sinusoidal oscillators and a constant term $a_0(\beta)$. Each of the oscillators generate $c_i(\beta)\cos(i\omega(\beta)t - \phi_i(\beta))$ and is independent of the oscillators. Figure 1B shows the output of the eight oscillators (blue traces) corresponding to their frequency of oscillation. The superposition of the outputs of all the oscillators and a dc offset results in the AP waveshape of a SAN cell (red trace).

A state controller (SC) makes RM capable of manifesting various dynamic properties as exhibited by the biological cells. Figure 1C shows the SC of a SAN which results in the rhythmic beating (at entrained frequency f_c) of the network of RM SAN cells despite the variations in their intrinsic beating frequencies f_0. When the N RM cells are coupled, the SCs of all the cells make a transition to the coupled state, in which the cells beat at f_c in accordance with the Equation (2). This equation models the coupling between the cells indicated by the gap junction in Figure 1A. $\mathrm{DDT_i}$ and $\mathrm{DDT_c}$ are the diastolic depolarization times of the intrinsic uncoupled beating and coupled synchronized beating of the cells, respectively.

$$\frac{1}{\mathrm{DDT_c}} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{1}{\mathrm{DDT_i}} \tag{2}$$

3. Implementation on CPU and GPU

The sinusoidal terms in Equation (1) are computationally expensive. Therefore, we have proposed an alternative cost effective implementation of an oscillator (a sinusoidal term) as described in Equation (3). $c_i(k)$ is the output of the ith oscillator at kth step. The constant C_i is $i * \omega * dt$ and dt is the time step. Here, the suitable initialization of c_i and s_i generates $c_i \cos(i\omega t - \phi_i)$. The WG comprising of n oscillators implemented with Equation (3) together with the constant term served as a starting point for the GPU implementation and as a reference for performance

comparisons between CPU and GPU.

$$c_i(k) = c_i(k-1) - C_i * s_i(k-1)$$
$$s_i(k) = s_i(k-1) - C_i * c_i(k) \quad (3)$$

For GPU programming, CUDA C was used, and the RM CUDA C code was derived from the code written in C for CPU simulations. In the WG, every oscillator is independent of other oscillators, so on the GPU, the AP computation was parallelized at the oscillator level. Thus, in each time step, one CUDA thread performed the calculation of one oscillator, and the coefficients of the oscillator were stored in the local memory of the thread. The outputs from all the oscillators were added using warp-synchronous programming. This allowed effective inter-thread communication by the warp shuffle function without accessing shared memory.

The performance of the RM was evaluated on a PC with an Intel Xeon E7-8867 v4 2.4 GHz CPU with 18 cores, 3 TB RAM, and an NVIDIA Tesla V100 GPU. Simultaneous multithreading (SMT) and Intel Turbo Boost technology were enabled on the CPU. The generated source codes were compiled using gcc compiler with the -o0 and -o3 optimization flags, and icc compiler with the -o3 optimization flag. The presented results were compiled with icc compiler with the -o3 optimization flag as being the fastest. All the simulations were executed in double precision both on the CPU and GPU. In addition, all the simulations were run for 100,000 steps with a time step of 0.1 ms. In all the experiments performed on the GPU, whole simulation data was held in GPU memory and remained on the device for the lifetime of the program. We measured kernel launch and execution time only which excludes the cost of data transfer from the device (GPU) to host (CPU) memory.

4. Results

With the increase in the number of oscillators in the WG, the accuracy of the AP waveshape generated by the RM also increases e.g., the accuracy of the generated SAN AP waveshape with the use of six oscillators [9], when compared with the reference waveshape, was 92.1%. This accuracy was increased to 99.82% with eight oscillators. However, we hypothesized that more oscillators would result in an increase in execution time. In our first designed experiment, we investigated this hypothesis of a trade-off between fidelity and execution time. The results of this experiment are shown in Figure 2. WG 4, WG 5, WG 6, WG 10, and WG 16 are the waveshape generators consisting of 4, 5, 6, 10 and 16 oscillators, respectively. The figure shows that upto 1000 WGs can run in parallel. For more than 1000 WGs, the execution time increased linearly with the number of WGs. The computation of ten million WGs with each WG consisting of six oscillators (WG 6) and ten

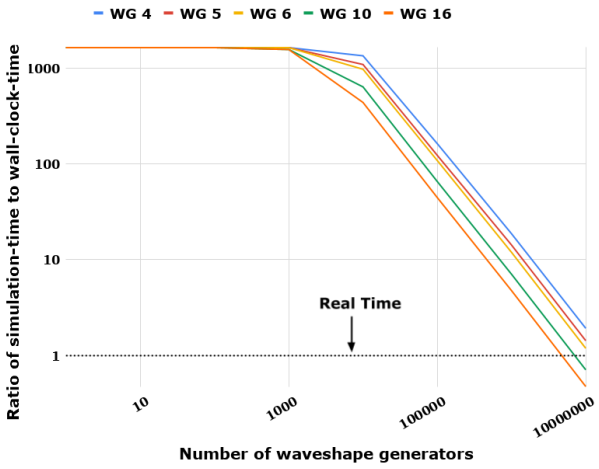

Figure 2. Simulation time on GPU normalized to real-time as a function of a number of waveshape generators.

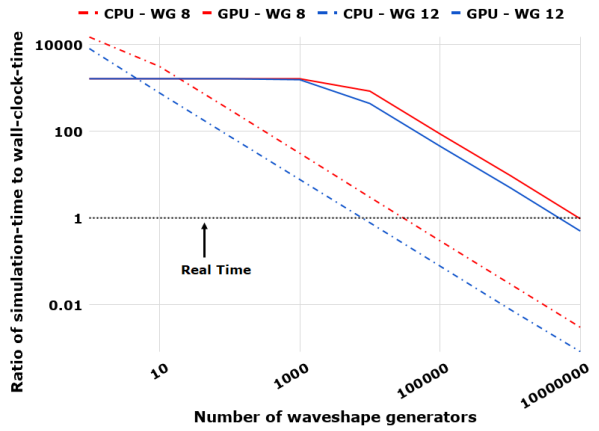

Figure 3. Scaling results for CPU and GPU implementations for a varying number of oscillators. WG 8, WG 12 are the waveshape generators consisting of eight and twelve oscillators, respectively.

oscillators (WG 10) was 19% faster and 30% slower, respectively, than the real-time. Figure 3 shows the comparison between CPU and GPU implementations of the WGs. For more than 10,000 WGs (WG 12), simulations of the RM were 600 times faster on the GPU in comparison to the simulations on the CPU.

We next investigated the effect of adding a SC on the RM cell computations. Four RM SAN cells consisting of eight oscillators in the WG were coupled (as described in the previous section) to form a cluster. Each cluster consisted of four RM SAN cells and was independent of every other cluster. Figure 4 shows the obtained results. RM execution for less than eight clusters on the GPU was slower than the CPU as the GPU was underutilized with the small

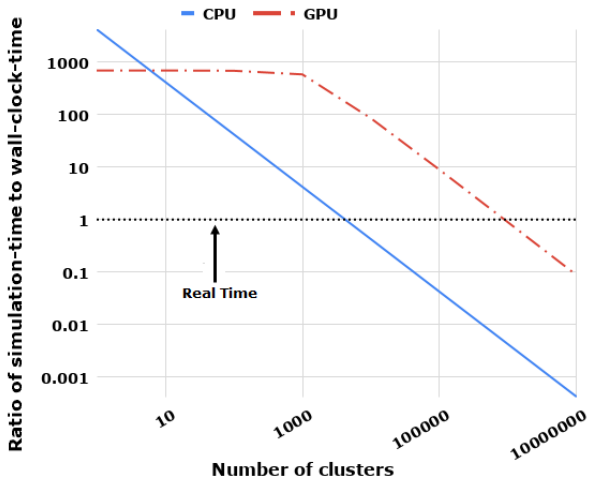

Figure 4. Speed comparison of the SAN resonant model clusters on the GPU and CPU. Each cluster consists of four resonant model SAN cells.

number of RM clusters. In addition, the kernel launch and other overheads were amortized over the number of clusters. One thousand clusters (4000 RM cells) were executed in parallel at no additional running time cost. Simulating 400K RM SAN cells was nine times faster than the real-time. However, on the CPU, less than 20K RM cells could be simulated in real-time. For 40K and more RM cells, the GPU implementation executed 200 times faster than the CPU.

5. Conclusion and Future work

In summary, we have shown that the RM is amenable to parallelism. The significant performance gains obtained from the GPU implementation of the cardiac cell RM may enable closer to real-time performance of heart simulations, minimizing the need for dedicated supercomputers. This facilitates the adoption of modelling-based precision health in the near future. To achieve the maximum gains in computational efficiency, it is necessary to consider cell-specific RM aspects of the implementation, including appropriate division of the computational workload among multiple kernels, and the use of new computational efficient schemes. Future work could involve simulating the performance of the RM of other biological cells and cardiac dynamics in tissues of realistic sizes by using GPU architectures. We also hope to compare the performance of the RM with other reduced and general models of biological cells on GPU.

References

[1] Ai W, Patel ND, Roop P, Malik A, Trew ML. Cardiac electrical modeling for closed-loop validation of implantable devices. IEEE Transactions on Biomedical Engineering 2019; 1–1.

[2] Fenton FH, Cherry EM. Models of cardiac cell. Scholarpedia 2008;3:1868.

[3] Ai W, Patel ND, Roop PS, Malik A, Andalam S, Yip E, Allen N, Trew ML. A parametric computational model of the action potential of pacemaker cells. IEEE Transactions on Biomedical Engineering 2017;65(1):123–130.

[4] Adler C, Costabel U. Cell number in human heart in atrophy, hypertrophy, and under the influence of cytostatics. Recent advances in studies on cardiac structure and metabolism 1975;6:343–355.

[5] Okuyama T, Okita M, Abe T, Asai Y, Kitano H, Nomura T, Hagihara K. Accelerating ode-based simulation of general and heterogeneous biophysical models using a gpu. IEEE Transactions on Parallel and Distributed Systems 2013; 25:1966–1975.

[6] Karma A. Electrical alternans and spiral wave breakup in cardiac tissue. Chaos An Interdisciplinary Journal of Nonlinear Science 1994;4:461–472.

[7] Iyer V, Mazhari R, Winslow RL. A computational model of the human left-ventricular epicardial myocyte. Biophysical journal 2004;87:1507–1525.

[8] Bartocci E, Cherry EM, Glimm J, Grosu R, Smolka SA, Fenton FH. Toward real-time simulation of cardiac dynamics. In Proceedings of the 9th International Conference on Computational Methods in Systems Biology. ACM, 2011; 103–112.

[9] Sehgal S, Patel ND, Malik A, Roop PS, Trew ML. Resonant model—a new paradigm for modeling an action potential of biological cells. PLOS ONE 2019;14:1–25.

Address for correspondence:

Nitish D. Patel
20 Symonds Street, Faculty of Engineering, Building 401
The University of Auckland, Auckland 1010, New Zealand
nd.patel@auckland.ac.nz