Using Graphic Processor Units for the Study of Electric Propagation in Realistic Heart Models

Andrés Mena[1], Jose F Rodriguez[1,2]

[1]Aragón Institute of Engineering Research, University of Zaragoza, Zaragoza, Spain
[2] Biomedical Research Networking Center in Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN), Spain

Abstract

The multi-scale nature of the electrophysiology problem requires the use of fine temporal and spatial resolutions leading to models with millions of degrees of freedom that need to be solved for a thousand time steps. Solution of this problem requires the use of algorithms with higher level of parallelism in multi-core platforms. The newer programmable graphic processing units (GPU) has become a highly parallel, multithreaded, many-core processor with tremendous computational horsepower. This paper presents results obtained using HESIC, a novel electrophysiology simulation software entirely developed in CUDA. The software implements implicit and explicit solvers for the monodomain model, using operator splitting. The ten Tussher and Panfilov cell model (TP06) has been considered in isolated single cell, and one-, two- and three-dimension anisotropic tissue models discretized with structured meshes with 0.1mm resolution have been used for benchmarking. Results obtained with a NVIDIA C2090 GPU on simulating 100ms of cell activity show that GPU performs non-linearly with the number of degrees of freedom. For small problems (<1000 nodes) the GPU underperforms a single CPU due to GPU's lower clock speed. However, as the problem size increases, the GPU outperforms a single CPU up to 180 fold for the integration of the ionic model, and up to 70 fold for the three-dimensional tissue model. These results points GPU computing as a promising and economic alternative for high performance simulations of heart electrophysiology.

1. Introduction

Over the last years, mathematical modelling and computer simulations have become a useful tool in analysing electrophysiological phenomena. In this particular, one of the major contributions of computer electrophysiology has been in understanding important relations between electrophysiological parameters [1]. In addition, continuous advances on medical imaging techniques have allowed for the development of computational anatomically realistic models of the heart. These anatomically realistic models can then be integrated to multi-scale biophysical model of the heart electrophysiology to obtain reliable quantitative mechanistic models. However, despite the great increase in computer power, execution times remain still prohibitive for these computer models [2].

The multi-scale nature of the electrophysiology problem (time constants for the different kinetics ranging from 0.1 to 500ms) makes difficult its numerical solution, requiring temporal and spatial resolutions of 0.1ms and 0.2mm respectively for accurate simulations, leading to models with millions degrees of freedom that need to be solved for thousand time steps.

Solution of this problem requires the use of algorithms with higher level of parallelism in multi-core platforms. In this regard, the next generation of high-performance computing (HPC) platforms promise to deliver better performance in the PetaFLOPS range. However, achieving high performance on this platforms relies on the fact that strong scalability can be achieved, something challenging due to the performance deterioration caused by the increasing communication cost between processors as the number of cores, n_p, increases. That is, with increasing n_p, the load assigned to each processor decreases, but the communication between different processors associated with the boundaries of a given partitioned domain increases. Therefore, when communication costs domain, no further benefits are obtained from adding additional processors. An alternative to the multi-core platforms is emerging in the newer programmable graphic processing units (GPU) which in recent years has become a highly parallel, multithreaded, many-core processor with tremendous computational horsepower. GPUs outperform multi-core CPUs architectures in terms of memory band width, but underperforms in terms of double precision floating point arithmetic. However, GPUs are built to schedule a large number of threads, thus, reducing latencies in their multi-

core architecture.

This paper presents results obtained using HESIC, a novel electrophysiology simulation software entirely developed in CUDA. The software implements implicit and explicit solvers for the monodomain model, using operator splitting and the Finite Element Method (FEM). Performance results are compared with an explicit multi-CPU based software [3].

2. Methods

2.1. Mathematical model

Cardiac electrophysiology can be described by means of the well established monodomain equation

$$\begin{aligned} \nabla \cdot (\mathbf{D}\nabla V) &= C_m \frac{\partial V}{\partial t} + J_{ion}(V, \mathbf{u}) + J_{stm} \\ \frac{\partial \mathbf{u}}{dt} &= \mathbf{f}(\mathbf{u}, V, t) \end{aligned} \quad (1)$$

where V is the transmembrane potential, $\mathbf{D}$ is the second order anisotropic conductivity tensor, C_m the membrane capacitance, J_{stm} the stimulus current, J_{ion} the ionic current, and $\mathbf{u}$ is a set of state variables associated with the ionic model. The set of equations (1) is subject to the zero flux boundary conditions

$$\mathbf{n} \cdot (\mathbf{D}\nabla \mathrm{V}) = 0 \quad (2)$$

2.2. Discretization and numerical algorithm

The set of PDEs defined in equation (1) are spatially discretized using the finite element method, whereas time integration is performed using a Strang based operator-splitting scheme in combination with a generalized trapezoidal family of method [4]. Let t^k be the current time step, where k is the index of the time step. When using the operator-spitting algorithm along with the finite element method for solving the monodomain model, equation (1), the transmembrane potential at time t^{k+1}, $\mathbf{V}^{k+1}$, is obtained is calculated in two steps. First, the electrophysiological model (Eq. (1b))

$$\mathbf{V}^* = \mathbf{V}^k + \Delta t J_{ion}(\mathbf{V}^k, \mathbf{u}) \quad (3)$$

is solved at each mesh point to obtained an intermediate transmembrane potential vector $\mathbf{V}\square$ (Step I). With this result at hand, the transmembrane potential at time step $k+1$ (Step II) is obtained as

$$\mathbf{M}\frac{\mathbf{V}^{k+1}-\mathbf{V}^*}{\Delta t} = -\mathbf{K}[\theta \mathbf{V}^{k+1} + (1-\theta)\mathbf{V}^k] \quad (4)$$

where $\mathbf{M}$ is the mass matrix, or capacity matrix, associated with $C_m \partial V/\partial t$, and $\mathbf{K}$ is the stiffness matrix associated with $\nabla \cdot (\mathbf{D}\nabla V)$, and $\theta \epsilon [0,1]$ is a scalar parameter. Equation (4) can be alternatively written as

$$\hat{\mathbf{K}}\mathbf{V}^{k+1} = \hat{\mathbf{b}} \quad (5)$$

where $\hat{\mathbf{K}}$ is everything that multiplies onto $\mathbf{V}^{k+1}$, and $\hat{\mathbf{b}}$ contains the other terms in Eq. (4).

Hence, the basic algorithm at time t^{k+1} can be summarized, as:

Step I: Use $\mathbf{V}^k$ as initial condition to integrate Eq. (3) to obtain $\mathbf{V}^*$

Step II: Use the result obtained in Step I, $\mathbf{V}^*$, to solve Eq. (5) for $\mathbf{V}^{k+1}$

For different values of the parameter θ, different time integration schemes are obtained for integrating the discretized homogeneous parabolic equation, Eq. (4):

$\theta = 0$ Forward Euler (conditionally stable)
$\theta = 1/2$ Crank-Nicholson (unconditionally stable)
$\theta = 2/3$ Galerkin Scheme (unconditionally stable)
$\theta = 1$ Backward Euler (unconditionally stable)

Step I of the algorithm uses values of the $\mathbf{V}^k$ at each mesh point to integrate the system of ODEs (Eq. 3) corresponding to the ionic model. This step can be performed using either implicit or explicit methods. However, the use of implicit methods requires the solution of a nonlinear system of equations at each node for each time step largely increasing the computational cost. On the contrary, explicit integration, even though computationally cheaper, imposes more stringent conditions on the size of the time step in order to avoid numerical instabilities.

2.3. Parallel implementation

In the discretized scheme, there are two main contributors to the computational cost: solving the system of ODEs at each mesh point, and solving the linear system of equations associated with the parabolic PDE. In order to maximize performance, all vectors and matrices associated with the system of equations reside on the GPU memory with the solution transferred back to the host only when the data has to be saved on disk. In addition, in order to minimize memory storage, all data is stored using sparse matrix structures.

Ionic solver in GPU: The C/C++ code of the ionic model was downloaded from the CellML model repository [5], modified by implementing the Rush-Larsen Technique [6] for integrating the gating variables, and then compiled for GPU. The exact same code was executed in CPU and GPU in order to validate the implementation. In addition, results were compared with the original code provided by tenTusscher and Panfilov

[3]. During execution, the vector of state variables, $\mathbf{u}$, and the current transmembrane potential, $\mathbf{V}^k$, remain in the GPU memory and use to compute $\mathbf{V}^*$ in Eq. (3).

PDE solver in GPU: In order to solve the linear system on GPUs efficiently, we have used CUSP and Thrust libraries [7]. CUSP is implemented for a single GPU and natively supports several sparse matrix formats making it easy to transfer data between the different sparse matrix formats. The library includes highly optimized matrix-vector multiplication algorithms and iterative solvers. In addition it provides a variety of preconditioners based on algebraic multigrid (AMG) and approximate inverse operators. In our implementation, mass and stiffness matrices, $\mathbf{M}$ and $\mathbf{K}$ respectively, are assembled and stored in compressed sparse row (CSR) sparse format and then transformed to an efficient sparse matrix format when transferred in the GPU memory for computations.

2.4. Benchmarking

Benchmarking was conducted for 1D, 2D and 3D problems. The model geometry was defined as a cable of length L, a rectangle with dimensions $L\times L$, and a cuboid with dimensions of $L\times 7\times 20$ mm^3, with L a positive integer set to achieve a given number of degrees of freedom in the problem. The different geometries where meshed with linear, quadrilateral and hexahedral elements of size 0.1 mm, 0.1×0.1 mm^2 and $0.1\times 0.1\times 0.1$ mm^3 respectively. Since ventricular cardiac muscle is anisotropic, the tissue was modeled as transversally isotropic with the fiber direction defined to be oriented along the x-axis.

Cellular dynamics was modeled using the human ventricular action potential model proposed by tenTusscher and Panfilov [8] (TP06) comprising 19 state variables. A propagating wave front was initiated at one of the corners of the model using a transmembrane current pulse of 50 μA/cm^2 strength for 2 ms, and the electrical activity was simulated over 100 ms. The monodomain model was integrated with a fixed time-step of 20 μs.

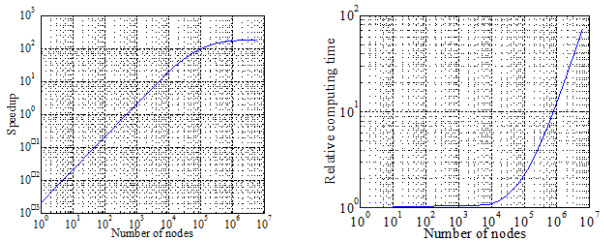
The parallel performance was evaluated as the speedup, S, i.e., the reduction in execution time of the GPU implementation with respect to a single CPU core.

GPU simulations were run on a computer node with two Intel-Xeon Quad-Core CPUs E5620 clocked at 2.4MHz and 48GB DDR3 RAM. The node is equipped with four Nvidia Tesla M2090 GPUs, each with 6GB DDR5 RAM for a total of 24GB DDR5 RAM. All simulations were run in a single GPU.

The CPU benchmark was run on a two Intel-Xeon Quad-Core E5520 clocked at 2.26GHz and 24GB DDR3 RAM.

3. Results

Before conducting the benchmarking, we first evaluated the speedup obtained in evaluating the ionic model (Eq. 3) for 1000 ms in many cells without considering the propagation of the signal. Figure 1a shows the speedup, S, up to six million cells. As can be seen in the Figure, a single GPU thread is about 478 slower than a single CPU core. However, as the number of nodes in the model increases, the GPU overtakes the CPU until reaching an asymptotic speedup of 180× for a model with more than a one million nodes. It is interesting to note that the C2090 GPU is able to evaluate the cell electrophysiology model in 8000 nodes simultaneously with almost no degradation in the computing time (see Figure 1b). However, for models with more than one million nodes, the computing time increases almost linearly.



(a) (b)

Figure 1. Results for the integration of the ionic model for a 1s of cell activity. a) Speedup curve; b) Relative computing time for the GPU.

The speedup, S, for the monodomain model is shown in Figure 2 for the 1D, 2D, and 3D problems.

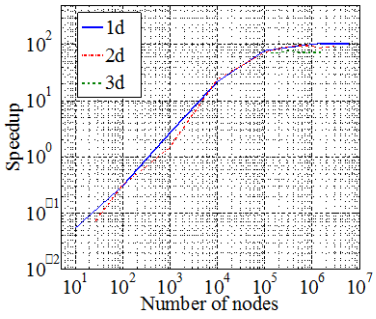


Figure 2. Speedup curve for the solution of the monodomain model with constant time step for one-, two, and three-dimensional problems.

Figure 2 shows the same trend as Figure 1a for the integration of the ionic model. For small problems the GPU underperforms the single CPU execution. However, as the number of degrees of freedom increases the GPU starts to outperform the CPU until reaching a rather asymptotic behaviour with a speedup of: 100× for 1d-problems, 90× for the 2d-problem, and 70× for the 3d-problem.

4. Discussion

In this study, the potential of GPU implementation for solving the monodomain equations of cardiac electrophysiology has been investigated. Scalability tests were carried out for 1d-, 2d-, and 3d-model problems with structured meshes using a novel software entirely developed in C++/CUDA.

Previous studies [9] have reported speedups by a factor of 32 for the monodomain model using an explicit finite difference scheme along with the phase I Luo-Rudy ionic model [10] that has fewer state variables than the TP06 model used in this study. In their study Sato et al [9] have established the solution of the PDE equation as the bottle neck of the computation with GPU since they have allow an adaptive time step when integrating the ionic model. In out implementation, we have proceeded by integrating with a constant time step size that guarantees both, the stability of the ionic model (most restrictive step size) and the stability of the PDE. With this strategy we have been able to make full use of the GPU parallel potential by reducing latency during GPU execution leading to acceleration of up to 90× for 2d-problems similar to those considered in [9]. However, it is worth mention that when adaptive time stepping is allowed in CPU execution, GPU acceleration is reduced to 20×.

Our results also indicate that a minimum problem size is required in order to get the maximum of the GPU implementation. This is due to the slower speed of the GPU cores and limited double precision floating point arithmetic as compare with the CPU core. This deficiency is, however, compensated by the capability of the GPUs to schedule a vast numbers of threads and efficiently reducing latency in this many core architecture. In fact, out results indicate that for problems with up to 8000 degrees of freedom, the GPU needed the same computing time. After this threshold is surpassed the scalability of the GPU with the number of degrees of freedom approaches to linearity. Interestingly, this threshold appeared to be independent of the problem dimensionality. However, our benchmark was only limited to structured meshes.

Another aspect worth mentioned from our studies was the influence of the sparse matrix format for storing the finite element matrices. In this regard, using the efficient Hybrid sparse matrix format offered in the CUSP-library [7] can reduce the computation time for 3-d problems in about a 10%.

5. Conclusion

This study demonstrates that significant reduction on computing time can be achieved for solving the cardiac monodomain equations. Despite the significant lower performance observed on a single GPU thread, a single GPU card offered excellent performance with speedups of 70× for three dimensional problems solved explicitly when compared with a single CPU. Preliminary results from our group also show excellent performance for the case of implicit schemes. In this case, acceleration of 54× are obtained for a 3D problem with 1.5 million degrees of freedom showing the great potential of GPU computing for cardiac electrophysiology. Current work is being conducted on extending the software to multi-gpu platforms in order to solve larger problems.

Acknowledgements

The authors would like to acknowledge research funding from the University of Zaragoza through the grant 171-242.

References

[1] Rodríguez B, Trayanova N, Noble D, Modeling Cardiac Ischemia. Ann. N.Y. Acad. Sci. 2006; 1080: 395-414.

[2] Neic A, Liebmann M, Hoetzl E, Mitchell L, Vigmond E, Haase G, Plank G. Accelerating cardiac bidomain simulations using grpahic processing units. IEEE J Trans Biomed Eng 2012;59(8):2281-90.

[3] Heidenriech E, Ferrero(Jr) JM, Doblare M, Rodriguez JF. Ann Biomed Eng 2010; 38(7): 2231-45.

[4] Hughes TJR. The finite element method: linear static and dynamic finite element analysis, 672pp. Prentice Hall Inc, Englewwog Cliffs, New Jersey 1987.

[5] CellML model repository. Available: http://models.cellml.org/cellml

[6] Rush S, Larsen H. A practical algorithm for solving dynamic membrane equations. IEEE J Trans Biomed Eng 1978; 25(4): 389-92.

[7] Bell N, Garland M. (2012). CUSP-library: Generic Parallel Algorithms for Sparse Matrix and Graph Computations. Availablecode. google.com/p/cusp-library/

[8] ten Tusscher KHWJ, Panfilov AV. Alternants and spiral breakup in a human ventricular tissue model. Am J Physiol Heart Circ Physiol 2006;291:H1088–H1100.

[9] Sato D, Xie Y, Weiss JN, Qu Z, Garfinkel A, Sanderson AR. Acceleration of cardiac tissue simulation with graphic processing units. Med Biol Eng Comput 2009; 47(9): 1011-15.

[10]Luo CH, Rudy Y. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interactions. Circ Res 1991; 68(6): 1501-26.

Address for correspondence.

Jose F Rodriguez.
Aragon Institute of Engineering Research.
Mechanical Engineering Department
Edf. Betancourt, María de Luna S/N,
Zaragoza 50018, Spain
jfrodrig@unizar.es, andres@menasl.com.