Computer Simulation of the Electric Activity of the Heart Using GPU. A Multi-Scale Approach

[1]A. Mena, [3]J.M. Ferrero, [1,2]J.F. Rodríguez

[1]CIBER, Zaragoza, Spain,
[2]Aragon Institute of engineering Research, Universidad de Zaragoza, Zaragoza, Spain
[3] Instituto de Investigación Interuniversitario en Bioingeniería y Tecnología Orientada al Ser Humano (I3BH), Universitat Politecnica de València, Valencia, Valencia, Spain
Email: jfrodrig@unizar.es

Abstract. *The electrophysiology problem posses a big challenge, not only because of the structural complexities inherent to the heart tissue, but also because of the complex electric behaviour of the cardiac cells. Solving the electric activity of the heart requires an enormous effort of data integration for generating a suitable model for simulation. The resulting model will be of multi-scale nature (ranging from the microscopic ionic channels to the surface ECG). The multi-scale nature of the electrophysiology problem 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 newer programmable graphic processing units (GPU) has become a highly parallel, multithreaded, many-core processor with tremendous computational horsepower. This paper presents results obtained with a novel electrophysiology simulation software entirely developed in CUDA. The software implements fully explicit and semi-implicit solvers for the monodomain model, using operator splitting. Performance is compared against classical multi-core MPI based solvers operating on dedicated high-performance computer clusters. Results obtained with the GPU based solver show enormous potential for this technology not only for research but also for clinical application due to its efficiency and lower hardware cost. The versatility of in-silico simulations is demonstrated on simulating the electric activity on realist models of the human heart.*

Keywords: Electrophysiology simulation, GPU simulation, Excitable media

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, increases. That is, with increasing n, 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 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 a multi-CPU based software [3].

2. Methods

The electric activity of the heart can be described by means of the well established monodomain equation [4]

$$\nabla \cdot (\mathbf{D}\nabla V) = C_m \frac{\partial V}{\partial t} + J_{ion}(V, \mathbf{u}) + J_{stm} \tag{1}$$

$$\frac{\partial \mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, V, t) \tag{2}$$

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

$$\mathbf{n} \cdot (\mathbf{D}\nabla \mathrm{V}) = 0 \tag{3}$$

where $\mathbf{n}$ is the outward pointing unit normal to the computational domain. The monodomain model represents an important simplification of the bidomain model with important advantages for mathematical analysis and computation. Despite its simplicity, this model is adequate for studying a number of electrophysiologic problems as ventricular fibrillation or the onset of ischemia in the electric behaviour of the heart [1,5-6].

An efficient form for solving Eq. (1-3) is by applying the Strang based operator-splitting scheme in combination with a generalized trapezoidal family of method for time integrations [7,8], in conjunction with the finite element method for the spatial discretization. The basic algorithm can be summarized the following two steps:

Step 1: Using V(t) as initial condition to integrate

$$\begin{aligned} C_m \frac{\partial V}{\partial t} &= -J_{ion}(V, \mathbf{u}) - J_{stm} \\ \frac{\partial \mathbf{u}}{dt} &= \mathbf{f}(\mathbf{u}, V, t) \end{aligned} \quad \text{for} \quad t \in [t, t + \Delta t] \tag{4}$$

Step 2: Using the result in Step 1 to integrate

$$C_m \frac{\partial V}{\partial t} = \nabla \cdot (\mathbf{D}\nabla V) \quad \text{for} \quad t \in [t, t + \Delta t] \tag{5}$$

When performing Step 2, the computational domain must be discretized in space by a mesh, e.g. finite elements, to approximate the dependent variables of the problem, V and $\mathbf{u}$. Hence, after the spatial discretization, the system of partial differential equations (5) can be written in matrix notation as

$$\mathbf{M}\dot{\mathbf{V}} + \mathbf{K}\mathbf{V} = \mathbf{0} \tag{6}$$

where $\mathbf{M}$ is the mass matrix associated with $C_m \partial V/\partial t$; and $\mathbf{K}$ is the stiffness matrix associated with $\nabla \cdot (\mathbf{D}\nabla V)$. These matrices are obtained by assembling individual element matrices. Equation 7 is called a semi-discrete equation because time is left continuous.

The most well-known algorithms for integrating (6) in time are members of the generalized trapezoidal family of methods [8]. Let $\mathbf{V}^k$ and $\dot{\mathbf{V}}^{\boldsymbol{k}}$ denote vectors of the transmembrane potential and its time derivative at each nodal point of the discretized domain (mesh) at time t^k, where k is index of the time step, then at time t^{k+1} we can write

$$\mathbf{M}\dot{\mathbf{V}}^{\boldsymbol{k+1}} + \mathbf{K}\mathbf{V}^{\boldsymbol{k+1}} = \mathbf{0} \tag{7}$$

$$\boldsymbol{V}^{\boldsymbol{k+1}} = \mathbf{V}^{\boldsymbol{k}} + \Delta t \dot{\boldsymbol{V}}^{\boldsymbol{k+\theta}} \tag{8}$$

$$\dot{\boldsymbol{V}}^{\boldsymbol{k+\theta}} = (1-\theta)\dot{\mathbf{V}}^{\boldsymbol{k}} + \theta \dot{\boldsymbol{V}}^{\boldsymbol{k+1}} \tag{9}$$

where $\theta \in [0,1]$ is a scalar parameter. When using the operator-splitting, Eq. 1-3 are solved in two steps. First, the electrophysiological model, Eq. 4

$$\boldsymbol{V}^* = \boldsymbol{V}^k + \Delta t(J_{ion}(\boldsymbol{V}^k, \boldsymbol{u}) + J_{stm}) \tag{10}$$

is solved at each mesh point to obtain an intermediate transmembrane potential vector $\mathbf{V}^*$ (Step 1). With this result at hand, using Eq. 8-9 to eliminate $\dot{\mathbf{V}}^{\boldsymbol{k+1}}$ from Eq.7 and $\mathbf{M}\dot{\mathbf{V}}^{\boldsymbol{k}} = -\mathbf{K}\mathbf{V}^{\boldsymbol{k}}$ from the previous converged time increment, the transmembrane potential at time step $k+1$ (Step 2) is obtained as

$$\mathbf{M}\frac{\mathbf{V}^{\boldsymbol{k+1}} - \mathbf{V}^*}{\Delta \boldsymbol{t}} = -\mathbf{K}[\theta \mathbf{V}^{k+1} + (1-\theta)\mathbf{V}^k], \tag{11}$$

or alternatively

$$\hat{\mathbf{K}}\mathbf{V}^{k+1} = \hat{\mathbf{b}} \tag{12}$$

where $\hat{\mathbf{K}}$ is everything that multiplies onto $\mathbf{V}^{k+1}$, and $\hat{\mathbf{b}}$ contains the other terms in Eq. 11.

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. 10 to obtain $\mathbf{V}^*$
Step II: Use the result obtained in Step I, $\mathbf{V}^*$, to solve Eq. 11 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. 5:

θ=0 Forward Euler (conditionally stable)
θ=1/2 Crank-Nicholson (unconditionally stable)
θ=2/3 Galerkin Scheme (unconditionally stable)
θ=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.

Parallel implementation in GPU

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 tenTusscher and Panfilov ionic model (TP06) [9] was downloaded from the CellML model repository [10], modified by implementing the Rush-Larsen integration scheme for the gating variables [11], 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. 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. 10. Adaptive time integration was not incorporated in the GPU implementation in order to reduce latency.

PDE solver in GPU: The linear system given in Eq. 12 was solved on GPUs using the CUSP and Thrust libraries [12] developed by Nvidia. CUSP is implemented for a single GPU and natively supports a number of sparse matrix formats providing specific subroutines for an easy passage between different sparse matrix formats. The library includes highly optimized matrix-vector multiplication algorithms and iterative solvers. In addition, a variety of preconditioners based on algebraic multigrid (AMG) and approximate inverse operators are ready available in the library. In our implementation, mass and stiffness matrices, $\mathbf{M}$ and $\mathbf{K}$ respectively, are assembled in parallel in the GPU and stored in compressed sparse row (CSR) format. These matrices are then transformed to an efficient sparse matrix format when transferred in the GPU memory for computations.

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. The dimension L was varied in order 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. For 2D and 3D problems, triangular and tetrahedral elements were also used to characterize the performance under a variable matrix bandwidth. In addition, since ventricular cardiac muscle is anisotropic, the tissue was modelled as transversally isotropic with the fibre direction oriented along the x-axis. For the benchmarking, cellular action potential was modelled using the TP06 model, which comprises 19 state variables. A propagating wave front was initiated at one of the corners of the model using a stimulation current pulse, J_{stm}, of 50 μA/cm^2 strength and 2 ms duration. The electrical activity was simulated over 100 ms, with a fixed time-step of 20 μs. The parallel performance was evaluated as the speedup, S, i.e., the execution time of the GPU implementation with respect to a single CPU core.

The efficiency of the GPU is further demonstrated on the simulation of a heart beat in a human heart and a human atria. A voxelized human heart with 1289000 elements and 1434129 nodes (DOF) has been considered. Figure 1a shows the voxel mesh (voxel size of

0.4×0.4×0.4 mm^3) along with the fiber orientation of the ventricular tissue. The TP06 model has been used to simulated the action potential model of the ventricular tissue. The efficiency of the code on non-structured meshes was demonstrated on a model of the human atria with 1378054 elements and 266450 nodes (see Figure 1b). The atrial action potential model proposed by Maleckar et al [13] was used for simulationg the electric activity of atrial tissue.

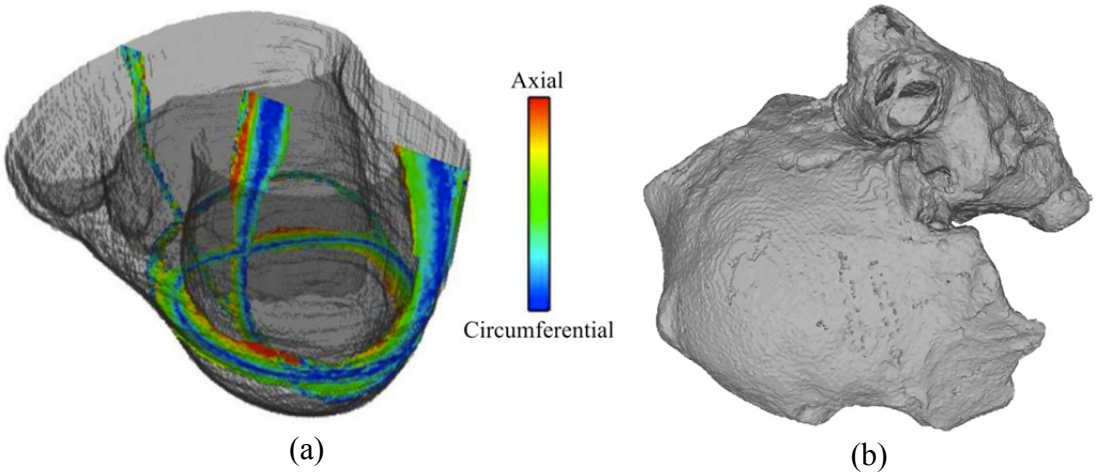


Fig. 1. Three dimensional finite element models. a) Hexahedral mesh of a biventricular human heart depicting the fibre orientation; b) Tetrahedral mesh of a human atria.

GPU simulations were run on a computer node with two Intel-Xeon Quad-Core CPUs E5620 clocked at 2.4GHz 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 single CPU benchmark was run on a single core of a cluster with 8 nodes with two Intel-Xeon Quad-Core E5520 clocked at 2.26GHz and 24GB DDR3 RAM connected by a high speed infiniband network.

3. Results

When solving the monodomain model, there are two critical steps in term of computation time: i) integration of the action potential model (Step I) and ii) the solution of the linear system of equations for the propagation of the electric signal (Step II). The operating splitting algorithm allows for an independent evaluation of the performance on these two steps.

Figure 2 shows the speed-up, *S*, obtained with the GPU in simulating 1 s of cellular activity (Step I) as the number of nodes in the model increases from 1 up to 6 million. The figure shows that core-to-core the GPU is much inferior to a single CPU, with a single CPU thread outperforming a single GPU thread about 480 times. However, the enormous computational horsepower of the GPU leads the GPU to overtake a single CPU as the number of nodes increases as shown in Figure 2a. This figure shows that the speedup increases linearly until reaching an asymptotic speedup of 180× for a model with more than a one million nodes.

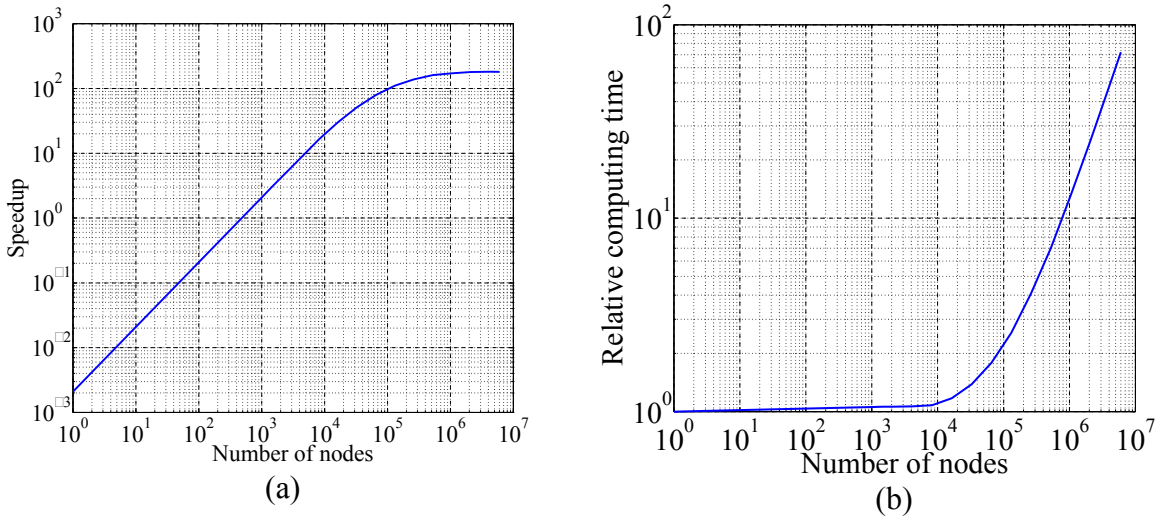


Fig. 2. Results for the integration of the ionic model for a 1s of cell activity. a) Speedup curve; b) Relative computing time for the GPU.

Regarding the relative computing time, Figure 2b shows 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.

The speedup, S, for the monodomain model is shown in Figure 3 for fully explicit ($\theta = 0$) and semi-implicit schemes ($\theta = 1$) for the one-, two-, and three- dimensional problems.

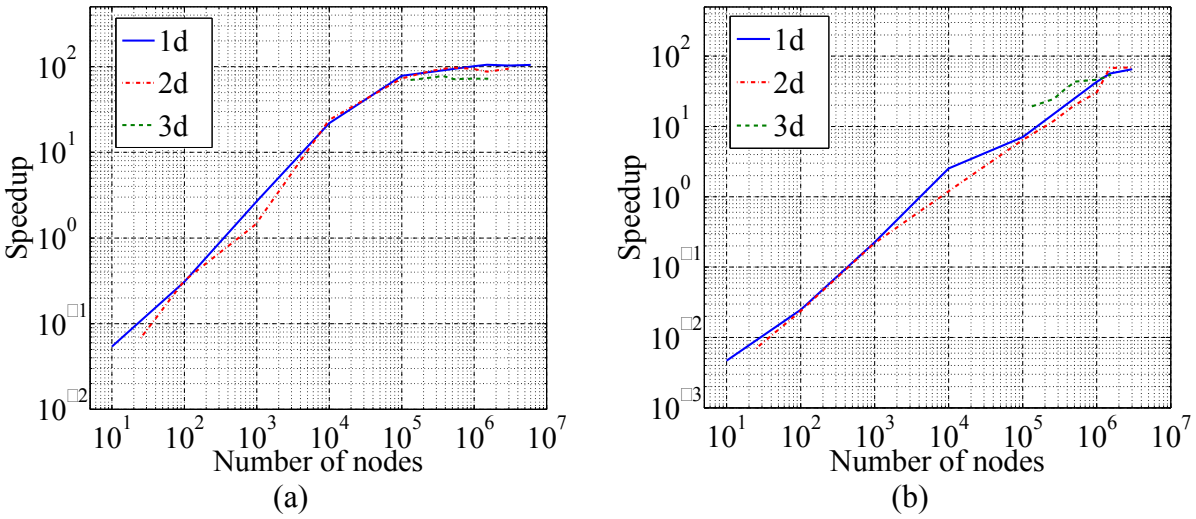
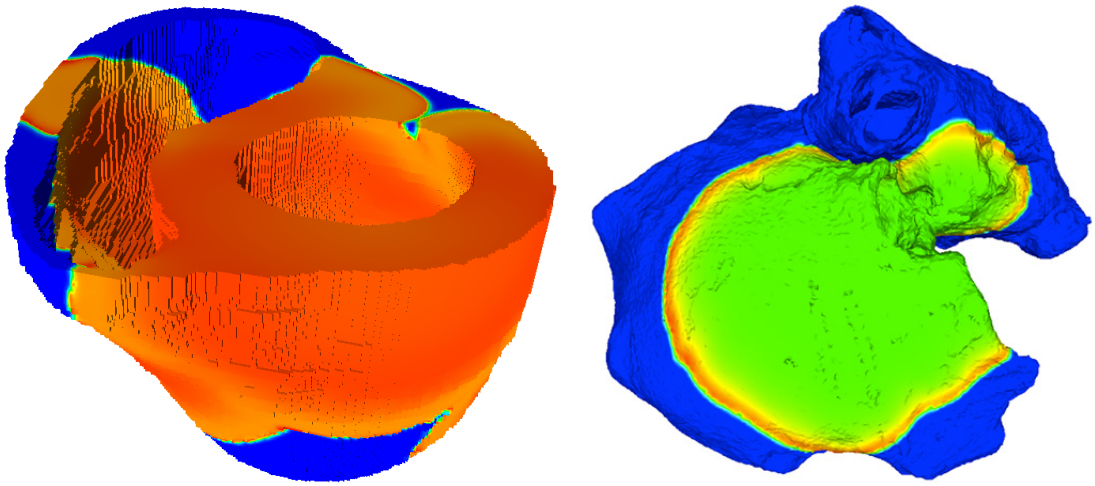


Fig. 3. Speedup curve for the solution of the monodomain model with constant time step for one-, two, and three-dimensional problems. a) Fully explicit scheme ($\theta = 0$); b) Semi-implicit scheme ($\theta = 1$).

Figure 3a shows the results for the fully explicit scheme. The same trend as in Figure 2a for the integration of the ionic model can be observed. As for the case of the integration of the ionic model, for small problems the GPU underperforms the single CPU execution due to the lower clock-speed of the single GPU with respect to the single CPU. However, as the number of degrees of freedom increases (the threshold was found close to 800 nodes) the GPU outperforms the CPU until reaching an asymptotic behaviour with a speedup of: 100× for 1d-problems, 90× for the 2d-problem, and 70× for the 3d-problem. The reduction in acceleration

is due to the additional cost implied by the matrix multiplication that takes place in Step II. For the semi-implicit scheme a similar trend as for the fully explicit scheme is observed. However, for the semi-implicit scheme the GPU does not reaches a saturation point due to the higher computational cost required for solving the system of equations (Eq. 12) on each time step. This increased computational cost is also noted by the reduction in the speed-up (65× for 1d-problem, 60× for 2d-problems, and 50× for 3d-problems), and the increased threshold up to approximately 8000 nodes as the minimum problem size for which the GPU starts to overtake the CPU.



(a) (b)

Fig. 4. Depolarization fronts at 20 ms after stimulation. a) Biventricular human heart; b) Human atria.

Figure 4 shows the depolarization front for the biventricular human heart and the human atria after 20 ms of stimulation. The speed-up obtained in these cases was 50× for the human heart and 40× for the human atria with respect to a single CPU core. In addition, the GPU still performed 1.8× faster than 42 cores of the cluster described in the Methods section, demonstrating the enormous potentiality of this technology.

4. Discussion

The potential of GPU implementation for solving the monodomain equations of cardiac electrophysiology has been investigated. Scalability tests were performed for fully explicit and semi-implicit numeric schemes in 1d-, 2d-, and 3d-model problems with structured and unstructured meshes using novel software entirely developed in C++/CUDA. The performance of the method was ulteriorly demonstrated on realistic models of a biventricular human heart and human atria.

Previous studies [14] have reported speedups of 32× for the monodomain model using an explicit finite difference scheme along with the phase I Luo-Rudy ionic model [15], a much less complex model as the TP06 model used in this study (7 state variables instead the 19 state variables of the TP06 or the 28 state variables of the Maleckar et al. action potential model of atria). In their study Sato et al [14] have established the solution of the PDE equation as the bottle neck of the computation with GPU, in part because they were forced to used single precision for their computations requiring a more stringent step size when solving the Step II of the algorithm than the required for integrating the action potential model (Step I). In addition, Sato et al. were using an adaptive time stepping when integrating the system of ODEs, which introduces latencies in the computations. In our implementation, we integrate 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, similarly as has been proposed in order

studies [2]. 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 [14] with an explicit scheme.

Results from Figure 3 also indicate that a minimum problem size is required in order to maximize GPU performance. This is due to the slower clock speed of the GPU cores and limited double precision floating point arithmetic as compare to CPU cores. This lower core performance is, however, compensated by the capability of the GPU to schedule a vast numbers of threads and efficiently reducing latency in this many core architecture. In this regard, for the C2090 our results indicate that for the ODE solver, the GPU is able to accommodate up to 8000 nodes without degradation of computer performance. For the monodomain solver, this threshold was found to be in 800 nodes for the fully explicit scheme and in 8000 nodes for the semi-implicit scheme. After this threshold is surpassed the scalability of the GPU with the number of degrees of freedom approaches to linearity. Even though these results are dependent on the GPU card used, this threshold appeared to be independent of the problem dimensionality for the cases considered. We must remark that the bandwidth of the linear system did not change much between the structured and unstructured meshes, being 27 for the 3d-structured mesh and 31 for the 3d-unstructured mesh. This explains the similarity between the results obtained for both benchmarks, as well as the speedup found for the ventricle and atria simulations.

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 [12] can reduce the computation time for 3-d problems up to a 10%.

5. Conclusions

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 core with respect to a single CPU core, a single GPU card offered excellent performance with speedups of 70× for three-dimensional problems solved explicitly and near 50× for three-dimensional problems solved with a semi-implicit scheme when compared with a single CPU. These results demonstrate that personal workstation is able to perform a simulation of the electric activity of a whole heart in reasonable times that enable researchers to interact more easily with their simulations.

Working with GPUs poses additional programming challenges over traditional parallel CPU implementations. However, like parallel CPU implementations, an efficient management of threads and memory are required if maximum performance of the GPU is to be achieved.

Acknowledgments

This project was partially supported by the “VI Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica” from the Ministerio de Economía y Competitividad of Spain (grant numbers TIN2012-37546-C03-01 and TIN2012-37546-C03-03) and the European Commission (European Regional Development Funds – ERDF - FEDER).

References

[1] Rodríguez B, Trayanova N, Noble D, Modeling Cardiac Ischemia. *Annals of the New York Academy of Science*, 1080: 395-414, 2006.

[2] Neic A, Liebmann M, Hoetzl E, Mitchell L, Vigmond E, Haase G, Plank G. Accelerating cardiac bidomain simulations using grpahic processing units. *IEEE Transactions on Biomedical Engineering*, 59(8): 2281-90, 2012.

[3] Heidenriech E, Ferrero(Jr) JM, Doblare M, Rodriguez JF. Adaptive Macro Finite Elements for the Numerical Solution of Monodomain Equations in Cardiac Electrophysiology. Annals of Biomedical Engineering, 38(7): 2231-45, 2010.

[4] Geselowitz DB, Miller III, WT. A bidomain model for anisotropic cardiac muscle. *Annals of Biomedical Engineering*, 11: 315– 334, 1983.

[5] Trayanova N, Eason J, Aguel F. Computer simulations of cardiac defibrillation: a look inside the heart. *Computing and Visualization in Science*, 4: 259–270, 2002

[6] Ferrero JM, Trenor B, Rodriguez B, Saiz J. Electrical activity and reentry during acute regional myocardial ischemia: insights from simulations. *International Journal of Bifurcation and Chaos*, 13: 3703–3715, 2003.

[7] Sundnes J, Nielsen BF, Mardal KA, Cai X, Lines GT, Tveito A. On the computational complexity of the bidomain and the monodomain models of electrophysiology. *Annals of Biomedical Engineering*, 34: 1088–1097, 2006.

[8] Hughes TJR. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Prentice Hall Inc., Englewwog Cliffs, NJ, 1987.

[9] ten Tusscher KHWJ, Panfilov AV. Alternants and spiral breakup in a human ventricular tissue model. *American Journal of Physiology Heart and Circulation Physiology*, 291: H1088–H1100, 2006.

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

[11] Rush S, Larsen H. A practical algorithm for solving dynamic membrane equations. *IEEE Transactions on Biomedical Engineering*, 25(4): 389-92, 1978.

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

[13] Maleckar MM, Greenstein JL, Giles WR, Trayanova N A. K+ current changes account for the rate dependence of the action potential in the human atrial myocyte. *American Journal of Physiology - Heart and Circulatory Physiology, 297*, H1398-H1410, 2009

[14] Sato D, Xie Y, Weiss JN, Qu Z, Garfinkel A, Sanderson AR. Acceleration of cardiac tissue simulation with graphic processing units. *Medical and Biological Engineering and Computing*, 47(9): 1011-15, 2009.

[15] Luo CH, Rudy Y. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interactions. *Circulation Research*, 68(6): 1501-26, 1991.