Large Speed Increase Using Novel GPU Based Algorithms to Simulate Cardiac Excitation Waves in 3D Rabbit Ventricles

Jonathan Higham, Oleg Aslanidi, Henggui Zhang

University of Manchester, Manchester, UK

Abstract

Large-scale biophysically detailed computer models of the heart provide a useful tool to understand dynamics of cardiac excitation and mechanisms underlying lethal cardiac arrhythmias. However, high demanding of intensive high performance computing environments limits the practical application of such models.

This paper presents a novel use of a desktop personal computer and the CUDA parallel computing architecture for a highly efficient method of parallel simulation of a 3D ventricular model. We show that substantial speed increases can be obtained using a desktop Graphical Processing Unit (GPU) compared to a single desktop Central Processing Unit (CPU), and that a single GPU can be an effective substitute to large numbers of CPUs.

1. Introduction

Although a vast amount of experimental and clinical data has been collected on the underlying ionic, cellular and tissue substrates, mechanisms of lethal cardiac arrhythmias arising from the functional interactions of these aspects at the whole heart level still remain unclear. Quantitative computer models of the heart integrating multi-scale data provide a powerful tool to understand arrhythmogenic mechanisms. The models allow detailed studies into the complex dynamics of three-dimensional (3D) spatio-temporal cardiac excitation waves in various physiological and pathological conditions, across various multi-physical scales [1-3]. However, the computational costs associated with large-scale cardiac simulations on 3D tissue geometries increases proportionally with increasing resolution (spatial and temporal), and with increasing levels of detail (electrophysiological, anatomical and mechanical) incorporated into the model.

In order to implement these complex cardiac models, several parallel computing paradigms have been developed using cutting edge High-Performance Computing (HPC) facilities [4-6]. However, the reality that a limited number of such HPC facilities exist makes such paradigms both costly and inconvenient.

In this paper, we show that widely available Graphical Processing Units (GPUs) can offer a cheap, convenient alternative to large numbers of Central Processing Units (CPUs), and can allow a substantial increase in the computational speed of large-scale, high-resolution, detailed simulations of cardiac excitation waves, even with the computational environment offered by a single, standard desktop PC.

2. Cardiac simulation

An anatomically accurate and biophysically detailed model of a rabbit ventricle was developed. The geometry of the rabbit ventricles was extracted from a whole heart DT-MRI dataset [7], including anisotropic fiber orientations. Atrial tissue and all visible noise (such as blood filling the right ventricular cavity) were removed. The spatial resolution of the resultant 3D ventricular geometry was 200μm in each direction. The total dimensions were 128 x 120 x 114.

Both ventricles were then segmented, the left ventricle into epicardial (Epi), mid-myocardial (M) and endocardial (Endo) regions, with Endo cells forming a layer around the cavity wall, Epi cells forming the external surface layer, and M cells in between; and the right ventricle into Epi and Endo cells. Previously developed cellular models for the ventricular Endo, M and Epi cells [8] were incorporated into the 3D geometry to simulate the electrical activity of the ventricles.

The mono-domain equation [1-3] was used to describe the electrical excitation dynamics in the tissue:

$$\frac{\partial V}{\partial t} = \nabla(\mathbf{D} \cdot \nabla V) - \frac{I_{ion}}{C_m} \quad (1)$$

Here V is the membrane potential (mV), t is time (s), $\mathbf{D}$ is the tensor of diffusion coefficients (mm^2ms^{-1}) that characterize the electrotonic spread of voltage via the gap junctions, I_{ion} is the total ionic current (pA), and C_m is the membrane capacitance (pF). Anisotropy was achieved by setting different diffusion coefficients in each direction; $D_{\parallel}$, $D_{\perp 1}$, and $D_{\perp 2}$ were set to 0.3, 0.075 and 0.0083 mm^2/ms respectively.

Equation (1) was solved by using the finite difference approach based on the forward Euler's method with a spatial step of 0.2mm and a time step of 0.005ms.

3. Graphical processing units

The GPU is a piece of specialized hardware resident within most personal computers for the purpose of graphical rendering. A Modern GPU is a highly data parallel processor, containing several hundreds of stream processors running in a Single Instruction Multiple Data (SIMD) model. The SIMD model reduces the time required for computation by forcing the condition that the same operation is carried out on each thread. SIMD launches a series of operations called the kernel, which carries out the same operation on thousands of threads with consequently very little memory overhead. This allows near-simultaneous calculation of many independent floating point operations; however it also means that the GPU architecture is suitable only for independent, data-parallel computations.

The electrophysiological activity in cardiac tissue can be separated into two components: one the electrical membrane current due to ion channels across each cell wall opening/closing; an independent process relative to other cells and the other is the localised diffusion of membrane potential between cells due to the intracellular coupling; a process dependent only on each cell and its closest neighbours. This independent nature of cardiac modelling makes it an ideal candidate for GPU use.

In order to take advantage of the drastic increase of GPU power in recent years, many new hardware and software architectures for GPU programming have been created [9-10], and used for a variety of scientific applications [11-12]. However, the increase in computational speed for such a highly parallelized processor has a high dependence on software parallelisation, especially under SIMD architecture, where a similar amount of time is required to run an operation on one thread, or on thousands of threads. A variety of speed increases in the range 10x-150x have been reported by the aforementioned scientific applications, but the effects of GPU parallelization on computational tasks within the field of cardiac modelling have not been studied. Therefore, assessing the potential impact of this novel technology on the computational efficiency of cardiac simulation can be important for the future of both cardiac modelling, and the modelling of other biological systems.

4. CUDA

Nvidia has recently released a parallel computing architecture called the Compute Unified Device Architecture, or CUDA. CUDA takes the form of a set of extensions to standard C/C++ to allow interaction between a computer's CPU, and a CUDA enabled GPU. CUDA refers to these two co-processors as the host and the device respectively.

Using CUDA, a user can allocate and access memory local to the device, copy data between the host and device, and offload parallel tasks to the GPU cores as a kernel [10].

The declaration of a CUDA kernel shares the same syntax as the declaration of a C++ function of type 'void'. In addition to the normal user specified input arguments, each kernel contains two additional inputs corresponding to the number of thread blocks (usually given the identifier DimGrid), and threads within each block (DimBlocks) that the kernel acts upon. The value given by DimGrid multiplied by DimBlocks is therefore the total number of threads that each kernel acts upon. Each thread within a kernel initialises two variables at its inception, a thread ID corresponding to the position of the thread within a thread block, and a block ID corresponding to the position of its block within a grid of blocks. This identifier can be used to index array pointers, allowing each thread to act upon, and modify a different memory address.

In order to provide a short comparison on the differences between CPU and GPU style code, a standard piece of CPU code for cardiac simulations is shown below. In this example, serial computation refers to the set of ordinary/partial differential equations used to simulate cell/tissue behavior, discussed earlier in Equation 1, and the Euler method used to solve them.

1) Allocate memory space on the host;
2) Initialize memory on the host;
3) Resolve serial computation on the CPU;
4) Print results.

In this example, steps 1 and 2 are carried out only once at the start of the simulation, and step 4 only once at the end. Step 3 is carried out at every time step for the duration of the simulation. This means that for the purposes of computational efficiency, only step 3 is important.

By contrast, the same code rewritten in CUDA is similar, but with a few added complications.

1) Allocate memory space on the host;
2) Allocate memory space on the device
3) Initialize memory on the host;
4) Copy data from host to device memory;
5) Resolve parallel portions of code on GPU;
6) Copy data from device to host memory;
7) Resolve any serial computations on CPU;
8) Print results.

Similarly to the CPU code, the initialization steps (1,2,3 in this case), and final step (8) are only carried out once, but the computational portions are carried out at every time step of the numerical integration of Equation (1). However, since any serial computation must be

carried out on the CPU and data on each device must be kept up to date after both the serial and parallel components, there are now four steps important for computational efficiency.

The key to achieving a maximal speed increase using CUDA is therefore careful allocation of memory space to minimize memory overhead caused by steps 4 and 6, and especially an emphasis on minimizing serial operations in step 7. In order to minimize the effects of memory latency caused by steps 4 and 6, as little information was retained within the host memory as possible. All differential variables were held within the device memory, and in order to visualize the output, only the membrane potential, *V* was copied to the host memory, and this was only copied every millisecond of simulation time (200 time steps).

In order to obtain the best performance from a CUDA architecture, copying of data between the device and host memories must be kept to a minimum. However, as many common C++ functions (such as file input/output), and command line arguments must be carried out on the host, such functions cannot be avoided completely.

5. Experimental validation

The activation sequence simulated (Fig. 1) was compared with the respective experimental data under similar conditions [13]. The experimental data and the model show both a similar pattern and timing of activation. It should be noted that data from the the CUDA simulations on the CPU, and the C++ simulation on the CPU were identical.

6. Results

The same PC was used to evaluate both the C++ CPU and CUDA GPU based solutions. It was equipped with an Intel core i5 M450 CPU clocked at 2.40 GHz, 4.00GB of host RAM, and an Nvidia GT 330 clocked at 1.2 GHz, with 2.7GB of total graphics memory.

As previously noted, it was found that both the CPU and GPU based solution produced identical ventricular activation sequences, and that these were in agreement with experimental data (Fig. 1). However, the time taken to simulate these sequences varied. On the CPU, the time taken to produce the desired result was measured to be 20375s, and the time taken to achieve the same result on the GPU was 298s, a 68-fold speed increase (Fig. 2).

In addition, the effect of increasing the number of cells within tissue was investigated. In our model, each thread within a kernel corresponds to one cell. By changing the number of cells within the geometry, we can therefore change the number of threads called within each kernel, and assess the effect of degree of parallelization on the speed of the process. As expected from a serial processor,

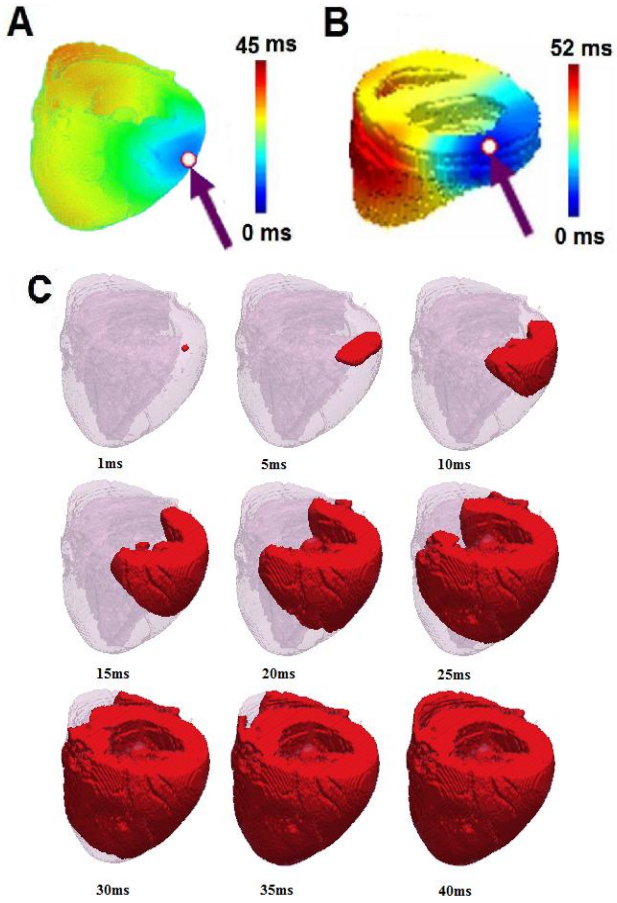

Figure 1. Simulations of excitation wave propagation in the 3D ventricular model solved on a GPU. The simulated ventricular activation sequence is shown (A) along with the respective experimental data (B). In both cases, the tissue activation time is color-coded according to the palette on the right. The initial stimulus application site is shown by an arrow. The panels in (C) show snap shots taken every 5ms of the excitation wave with an iso-surface of $V = 0$ mV.

the CPU speed per cell stays consistent throughout, however, even with increasing numbers of parallel threads there is little effect on the time taken to resolve the computation on the GPU, resulting in a vast increase in speed (Fig. 3).

Figure 3 shows that ~10^3 was the smallest number of cells where the possible speed increases given by a GPU become apparent. It is suggested that this number is an important baseline at which GPU computation becomes an attractive replacement for CPU simulation within the field of biophysically detailed cardiac modelling.

7. Discussion

We have developed a 3D computational model of rabbit ventricles, a numerical solver based on the novel CUDA architecture using GPU technology. We have shown that

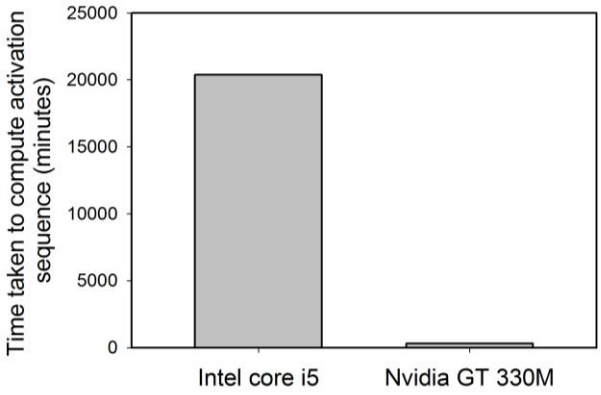

Figure 2. Difference in computational time using a CPU and a GPU. Shown is the time taken to compute a single cycle of wave propagation through the ventricles – a ventricular activation sequence on both a CPU and GPU.

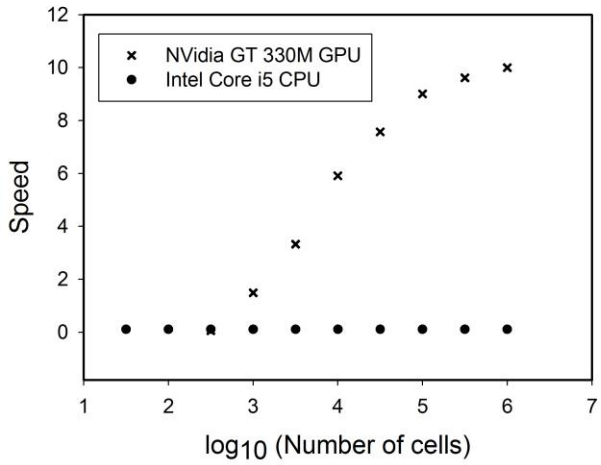

Figure 3. Comparison in speed between the CPU and GPU while running simulations of a 3D cardiac with a varying number of cells. Speed is given as 1/time to compute 10^4 kernel executions.

a current desktop PC with a commercial GPU provides an efficient environment for the large scale, complex biological simulations that until now, remain the domain of expensive HPC environments. We have shown that 10^3 threads is a minimum after which GPU computation is an effective alternative. With a sufficiently large number of threads, a 68-fold speedup can be achieved.

The speed up observed was towards the higher end of the range expected by Nvidia, and of the same order as the 10x speed increase reported in solving Maxwell's equations [11] and the 50x speed increases reported in the work on neural networks [12]. Such a large disparity in observed GPU speedups across different disciplines is due to a key limitation of GPU parallelization; the possible speed up is heavily dependent on the type of problem solved. However, as we have shown in the present study, biophysically detailed cardiac modelling is an ideal candidate for GPU computing.

References

[1] Rudy Y. From genome to physiome: integrative models of cardiac excitation. Ann. Biomed. Eng., 2000;28:945-950.

[2] Noble D. Modelling the heart – from genes to cells to the whole organ. Science 2002;295:1678-1682.

[3] Boyett MR, Li J, Inada S, Dobrzynski H, Schneider JE, Holden AV, Zhang H. Imaging the heart: computer 3-dimensional anatomic models of the heart. J. Electrocardiol 2005;38:113-120.

[4] Reumann M, Fitch BG, Rayshubskiy A, Keller DU, Weiss DL, Seemann G, Dossel O, Pitman MC, Rice JJ. Strong scaling and speedup on up to 16,384 processors in cardiac electro-mechanical simulations. Annual International Conf IEEE Eng. Med. Biol. Soc. 2009;2795-2798.

[5] Pitt-Francis J, Gamy A, Gavaghan D. Enabling computer models of the heart for high-performance computers and the grid. Phil. Trans. R, Soc. A, 2006;364:1501-1526.

[6] Bordas R, Carpentieri B, Fotia G, Maggio F, Nobes R, Pitt-Francis J, Southern J. Simulation of cardiac electrophysiology on next generation high performance computers. Phil. Trans. A 2009;367:1951-1969.

[7] Benson AP, Gilbert SH, Aslanidi OV, Zhang H, Boyett MR, Dobrzynski H, Holden AV 0.2 mm cubic voxel reconstruction of a rabbit heart geometry and architecture using diffusion tensor magnetic resonance imaging. Proc. Physiol. Soc., 2008;10:PC11.

[8] Aslanidi OV, Sleiman RN, Boyett MR, Hancox JC, Zhang H. Ionic Mechanisms for electrical heterogeneity between rabbit Purkinje fiber and ventricular cells. Biophys J. 1998;98:2420-2431.

[9] Buck I, Foley T, Horn D, Sugarman J, Fatahalian K, Houston M, Hanrahan P. Brook for GPUs: Stream computing on graphics hardware. ACM Transactions on graphics, 2004;23:777-786.

[10] Nvidia. CUDA Programming Guide. *http://developer.nvidia.com/objectcuda_1_0.html 2007*

[11] Dosopoulos S, Jin-Fa L. Discontinuous Galerkin Time Domain for Maxwell's equations on GPUs. 2010 URSI International Symposium on Electromagnetic Theory 2010:989-999.

[12] Guzhya A, Dolenko S, Persiantsev I. Multifold Acceleration of Neural Network Computations using GPU. Artificial Neural Networks – ICANN 2009;5768:380-373.

[13] Han C, Liu Z, Zhang X, Pogwizd S, He B. Noninvasive three-dimensional cardiac activation imaging from body surface potential maps: a computational and experimental study on a rabbit model. IEEE Trans. Med. Imaging 2008;27:622-1630.

Address for correspondence

Henggui Zhang.
Biological Physics Group, School of Physics and Astronomy, University of Manchester, Schuster Building, Brunswick Street, Manchester, M13 9PL, UK.
E-mail: *henggui.zhang@manchester.ac.uk*.