Toward Real-Time Simulation of Cardiac Dynamics

Ezio Bartocci, Elizabeth M. Cherry, James Glimm, Radu Grosu, Scott A. Smolka and Flavio H. Fenton.

In CMSB 2011: Proceedings of the 9th ACM International Conference on Computational Methods in Systems Biology, Paris, France, September 21-23, pages 103-110. ACM, 2011.

Corresponding author: Flavio H. Fenton, fhf3@cornell.edu.

Toward Real-time Simulation of Cardiac Dynamics

E. Bartocci[1,2], E.M. Cherry[3], J. Glimm[2], R. Grosu[1], S.A. Smolka[1], and F.H. Fenton[4]
[1]Department of Computer Science, Stony Brook University, Stony Brook, NY.
[2]Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY.
[3]School of Mathematical Sciences, Rochester Institute of Technology, Rochester, NY.
[4]Department of Biomedical Sciences, Cornell University, Ithaca, NY.

ABSTRACT

We show that through careful and model-specific optimizations of their GPU implementations, simulations of realistic, detailed cardiac-cell models now can be performed in 2D and 3D in times that are close to real time using a desktop computer. Previously, large-scale simulations of detailed mathematical models of cardiac cells were possible only using supercomputers. In our study, we consider five different models of cardiac electrophysiology that span a broad range of computational complexity: the two-variable Karma model, the four-variable Bueno-Orovio-Cherry-Fenton model, the eight-variable Beeler-Reuter model, the 19-variable Ten Tusscher-Panfilov model, and the 67-variable Iyer-Mazhari-Winslow model. For each of these models, we treat both their single- and double-precision versions and demonstrate linear or even sub-linear growth in simulation times with an increase in the size of the grid used to model cardiac tissue. We also show that our GPU implementations of these models can increase simulation speeds to near real-time for simulations of complex spatial patterns indicative of cardiac arrhythmic disorders, including spiral waves and spiral wave breakup. The achievement of real-time applications without the need for supercomputers may, in the near term, facilitate the adoption of modeling-based clinical diagnostics and treatment planning, including patient-specific electrophysiological studies.

Categories and Subject Descriptors

D [**Software**]; F [**Theory of Computation**]; G [**Mathematics and Computing**]; H [**Information Systems**]; I.6 [**Simulation and Modeling**]: *Applications;* J.3 [**Computer Applications**]: *Life and Medical Sciences*

General Terms

Performance; Theory

Keywords

GPU computing, High-performance computational systems biology, cardiac models.

1. INTRODUCTION

Cardiac arrhythmia, such as atrial fibrillation (AF) and ventricular fibrillation (VF), is a disruption of the normal excitation process in cardiac tissue due to faulty processes at the ion-channel and cellular level, at the level of cell-to-cell communication, or at the full organ level. The clinical manifestation is a rhythm with altered frequency (tachycardia or bradycardia, see Fig. 1(b)) or the appearance of multiple frequencies driven by spiral waves of electrical activity (polymorphic tachycardia), with subsequent deterioration to a chaotic signal known as fibrillation (see Fig. 1(c)) [4, 16, 17, 49].

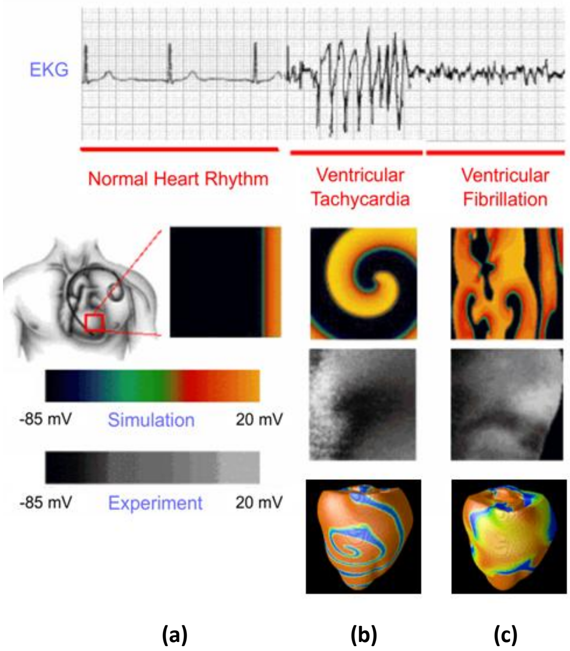


Figure 1. Four views of the transition from (a) normal heart rhythm to (b) ventricular tachycardia and (c) ventricular fibrillation. Top to bottom: electrocardiogram, 2D simulation, imaging-based *in vitro* experiment, and 3D simulation.

An important characteristic of cardiac cells is the *membrane potential*: the difference in voltage between the interior and exterior of a cell. The membrane potential is non-zero because of differences in the concentrations of various ions in the intracellular and extracellular environments along with time-dependent changes in response to the net actions of ion channels and ion pumps (such as the sodium-potassium ATPase) embedded in the membrane, which transport ions across the membrane [10].

Cardiac tissue is typically modeled mathematically as a reaction-diffusion system involving partial differential equations (PDEs) for diffusing species (typically only the transmembrane potential) and a system of nonlinear differential equations for the other state variables, which describe the flux of ions across the cell membrane along with corresponding changes in ion concentrations. Detailed models of cardiac cells can include more than 80 state variables and hundreds of fitted parameters [10].

CMSB '11, September 21-23, 2011 Paris, FR

Because of the computational demands such models place on normal CPU-based and multi-core-based workstations, most studies of the electrical activity of the heart, including real-time interactive Java programs [11], traditionally have focused on using either complex cell models in single-cell formations or simplified cell models in more realistic 3D heart structures. A few studies have used supercomputers to integrate models of intermediate complexity with 3D structures [21, 34, 47].

Real-time simulation refers to the ability to execute a computer model of a physical system at the same rate as the actual physical system. Recently, the advantages of GPU over CPU processing have been established for many areas of science, including systems biology. Using the CUDA (Compute Unified Device Architecture) parallel programming model from NVIDIA, the highly parallel and multi-core capabilities of GPUs can be exploited to achieve extremely fast simulations of complex models in 2D and 3D. If such large-scale realistic models can be simulated in *near* real-time, many more applications, including patient-specific treatment strategies for cardiac-rhythm disorders, become feasible without the need for supercomputers. To date, such efforts have come up short.

In this paper, we show that it is possible to perform simulations of models of cardiac cells ranging in complexity from 2 to 67 variables in *near real time* for realistic problem sizes through careful GPU implementations. To maximize performance gains, model-specific optimization techniques, including partitioning of the model equations among multiple CUDA kernels as appropriate and judicious use of the different types of memory available to GPUs, are incorporated.

The particular models we consider are the 2-variable Karma model, the 4-variable Bueno-Orovio-Cherry-Fenton (BCF) model [3], the 8-variable Beeler-Reuter (BR) model [1], the 19-variable Ten Tusscher-Panfilov (TP) model [46], and the 67-variable Iyer-Mazhari-Winslow (IMW) model [22]. For each of these models, we implement both single- and double-precision versions and demonstrate linear or even sub-linear growth in simulation times with an increase in the size of the two dimensional grid used to model cardiac tissue. We also show that our GPU implementations of these models can increase simulation speeds to near real-time for simulations of complex spatial patterns indicative of cardiac arrhythmic disorders, including spiral waves and spiral wave breakup.

Furthermore, although some of the simpler models have been previously studied in detail in tissue, this is not the case for many complex models. The IMW model, for example, has only been simulated in tissue by us in a previous manuscript [3] and using supercomputers. The present manuscript shows, for the first time, that it is possible to simulate a model of that complexity on a desktop computer. Other models of similar complexity that have never been modeled in 2D or 3D (for example, Grandi et al. [13], O'Hara et al. [33]) now can be studied without the need for supercomputers. We therefore believe that by using this approach, anyone with a desktop computer will be able to study a wide range of dynamics for the vast majority of models of cardiac electrophysiology, regardless of the model's complexity.

2. CUDA PROGRAMMING MODEL

CUDA is a general-purpose parallel-computing architecture and programming model that leverages the parallel compute engine in NVIDIA GPUs. Optimal programming of GPUs requires a thorough understanding of the CUDA architecture and the underlying GPU hardware, including the concepts of threads, processors, and kernels, as well as the different levels of memory available. As illustrated in Fig. 2, the GPU architecture is built around a scalable array of multithreaded Streaming Multiprocessors (SMs), made up of 8 or 32 Scalar Processor (SP) cores. SP cores contain a fused add-multiply unit capable of both single- and double-precision arithmetic and share a common local memory.

The CUDA parallel computing model uses tens of thousands of lightweight threads assembled into one- to three-dimensional *thread blocks*. A thread executes a function called the *kernel* that contains the computations to be run in parallel; each thread uses different parameters. Threads located in the same thread block can work together in several ways. They can insert a synchronization point into the kernel, which requires all threads in the block to reach that point before execution can continue. They also can share data during execution. In contrast, threads located in different thread blocks cannot communicate in such ways and essentially operate independently. Although a small number of threads or blocks can be used to execute a kernel, this arrangement would not fully exploit the computing potential of the GPU. To utilize the GPU most effectively, the underlying problem should be divided into a 2D or 3D grid of independent blocks, each of which can be further subdivided into cooperating threads; see Fig. 3. Problems that cannot be implemented in this manner will benefit significantly less from a GPU-based implementation.

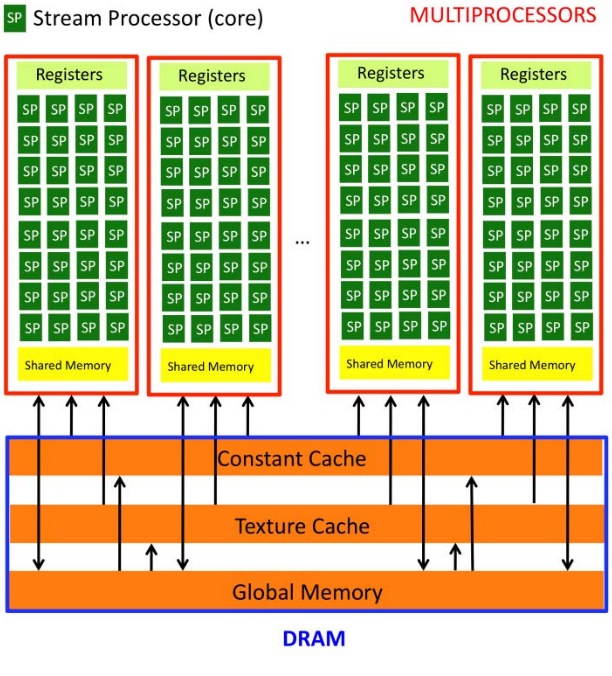


Figure 2. GPU architecture.

Different types of memory are available for use in CUDA, and their judicious use is key to performance. The most general is global memory, to which all threads have read/write access. The generality of global memory makes its performance less optimized overall, so it is important that access to it be coalesced into a single memory transaction of 32, 64, or 128 bytes [32]. Constant memory is a cached, read-only memory intended for storing constant values that are not updated during execution. All instances of a kernel may access these values regardless of location. Texture memory is another cached, read-only memory that is designed to improve access to data with spatial locality in up to three dimensions. For example, texture memory is a natural choice for storing an array that also will require retrieval of neighboring values whenever a single entry is retrieved. Linear interpolation is provided with texture memory. Finally, local memory is invoked when a thread runs out of available registers. CUDA library functions in the host code running on the CPU administer such tasks as kernel execution and memory management.

Additional, significantly faster levels of memory are available within an SM, including 16KB or 32KB of registers partioned among all threads. As such, using a large number of registers within a CUDA kernel will limit the number of threads that can run concurrently. In addition, each SM has a shared memory region (16KB). This level of memory, which can be accessed nearly as quickly as the registers, facilitates communication between threads and also can be used as a memory cache that can be controlled by the individual programmer [25]. Shared memory is divided into 16 banks; for optimal performance, threads executed concurrently should access different banks to prevent bank conflicts [32].

The computing resources of CUDA-capable video cards are characterized by their compute capability. Devices with compute capability 1.0 and 1.1 make up the first generation of CUDA devices, based on the G80 GPU, whereas those with compute capability 1.2 and 1.3 are based on the more advanced GT200 GPU. Only cards with compute capability greater than or equal to 1.3 allow double-precision floating-point operations. Recently NVIDIA introduced a new family of cards called the Fermi-based GF100 GPU with compute capability 2.0 that supports object-oriented programming.

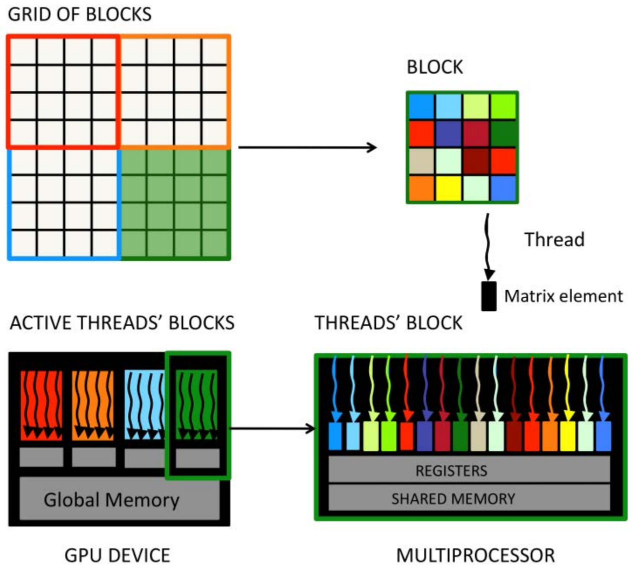


Figure 3. A 2D grid of blocks. The computation of each array element is assigned to a thread. Threads within the same block can communicate through shared memory. Threads in different blocks must communicate through global memory.

Our GPU testbed is an NVIDIA Tesla C2070 processor containing 448 scalar cores organized as 14 SMs with 6GB of DRAM. The processor core clock is 1.15 GHz and the maximum memory-access bandwidth is 144 GB/sec. The C2070 can perform 1030 Gigaflops using single-precision arithmetic or 515 Gigaflops using double-precision arithmetic. To compare our results with other papers, we also have used a GPU Tesla C1060 with 240 SP cores divided equally among 30 SMs, and 4GB of DRAM. The SP core clock is 1.29 GHz and the maximum bandwidth of memory access is 102 GB/sec. The C1060 is able to perform 933 Gigaflops using single-precision arithmetic or 78 Gigaflops using double-precision arithmetic.

3. CARDIAC MODELS

We focus on implementing models of cardiac electrophysiology. At the tissue level, a multicellular preparation usually is approximated using a continuum model, rather than by incorporating discrete cells. The primary variable of interest is the voltage across the cell membrane, called the voltage or membrane potential *V*, along with one or more other variables. In this way, cardiac electrical dynamics can be described using a reaction-diffusion equation of the following form:

$$\frac{\partial V}{\partial t} = \nabla \cdot D \nabla V - \frac{I_{ion}}{C_m},$$

where the first term on the right side represents the diffusion component with diffusion tensor *D* and the second term on the right represents the reaction component with total current across the cell membrane I_{ion} and constant cell membrane capacitatnce $C_{m,}$The diffusion tensor *D* describes how cells are coupled together and is an important determinant of the velocity with which waves propagate in tissue, typically around 70-90 cm/s in human ventricular tissue [30, 45]. In addition, *D* may contain information about tissue architecture that affects wave propagation, including the *x*-, *y*-, and *z*-components of the local muscle fiber orientation. The orientation of fibers is important because diffusion, and hence electrical wave propagation, is fastest in this direction, a property called anisotropy.

The exact form of the reaction term varies depending on the level of detail and complexity of the electrophysiology model. However, it is always nonlinear and is coupled to one or more ordinary differential equations of the following form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{F}(\mathbf{y}, V, t),$$

for additional variables **y** and nonlinear function **F** and for $V = V(t,\mathbf{y})$. Most models of cardiac cell electrophysiology have their origins in the Hodgkin-Huxley model [20] of a neuron along with the Noble model [31], which was the first to apply the same modeling principles to cardiac cells. This type of model describes the reaction term I_{ion} as the sum of currents of different ion species (for cardiac cells, primarily Na^+, K^+, and Ca^{2+}) crossing the cell membrane through ion channels. These currents operate in a specific fashion to generate a cellular action potential, an excursion from a negative resting membrane potential (around -85 mV) to a positive potential (around 20 mV in tissue) and back to the resting membrane potential. The initial increase in potential occurs quickly (on the order of a few ms) via a large inward current carried by Na^+ ions. Over the rest of the action potential, which lasts hundreds of ms in large mammals, the membrane potential is determined primarily by a balance between inward Ca^{2+} currents and outward K^+ currents. The range of voltages and times over which each ion current is active is determined by one or more factors, including the membrane potential, time-dependent gating variables that modulate the permeability of ion channels, and the concentrations of ions inside and outside the cells. Gating variables, ion concentrations, and other terms often are state variables (the vector **y** in the above equation) of a model and evolve according to their own differential equations, possibly according to different time scales.

The precise details of the electrophysiology models can be represented at different levels. Biophysically detailed models take into account a large number of currents and the vector of state variables **y** may have anywhere from a few to dozens of components in addition to the membrane potential. One of the earliest models of cardiac cells is the Beeler-Reuter (BR) model [1], which represents the electrophysiology of ventricular cells (located in the lower chambers of the heart). This model includes a total of eight state variables and has been widely used for several decades; we include it as one of the models we implement in CUDA. More recent models make use of subsequent biophysical discoveries to represent cardiac cell electrophysiology in more detail. We implement two such models, both of human ventricular electrophysiology: the 19-variable Ten Tusscher-Panfilov model (TP) [46] and the 67-variable Iyer-Mazhari-Winslow (IMW) model [22].

Although these models may seem overly complex and overfitted, they are developed to reproduce in detail the dynamics of each specific ion channel that is important in determining overall cardiac cell electrophysiology in normal and pathological conditions. Such information is crucial for understanding how interventions at the ion channel level can affect cell behavior; however, at the tissue level, the wealth of biophysical detail in complex electrophysiology models can obscure the physical phenomena underlying their behavior. Furthermore, as discussed in Ref. [3], the large number of parameters in these models tends to increase the likelihood of converging to a local minimum rather than the global minimum during the parameter fitting process. For this reason, less complicated models also have been developed that can represent the behavior of cardiac cells and tissue in a way that facilitates analysis of their dynamics and of the role of different parameters in determining their behavior. This class of models gives up biophysical detail at the ion channel level in exchange for greater computational tractability One such model is the Karma model [23], which is a simplification of the Noble model. Another such model is the Bueno-Orovio-Cherry-Fenton (BCF) model [3] (also called the minimal model [5, 9]). This model uses four variables and three ion currents that represent summary Na^+, K^+, and Ca^{2+} currents and incorporates the minimal level of complexity necessary to reproduce accurate action potential shapes and rate-dependent properties. We also implement both of these models in CUDA.

In total, we implement 5 different models containing from 2 to 67 variables. By considering models that vary over a broad range of biophysical detail and computational complexity, we are able to identify and address a number of performance issues that arise in the CUDA programming of cardiac models.

4. REACTION TERM

The reaction term in cardiac models consists of anywhere from one to dozens of additional differential equations that provide simple or detailed descriptions of the electrophysiology of cardiac cells. Appropriate implementation of the reaction term is vital to optimizing the performance of cardiac electrophysiology simulations on GPUs.

Performance is increased significantly by tabulating nonlinear functions of one variable in lookup tables that are accessed through the texture memory. This provides two main advantages: it reduces the latency of global memory access, and the hardware provides a built-in linear interpolation capability. To remove singularities that occur in some functions for values that make the denominator of a fraction zero, we calculate the limit of the functions at the relevant values using l'Hopital's rule.

We also improve performance in several other ways. Because division is more computationally expensive than multiplication, all divisions that do not involve variables are replaced with equivalent multiplications. Also, some differential equations are solved using semi-implicit methods that allow the use of larger integration time steps.

In a number of cases, the reaction term in cardiac models uses biological switching functions in the form of a *Heaviside function*, which is a discontinuous function whose value is zero for negative arguments and one for positive arguments. Heaviside functions are usually implemented using an *if* statement, which is penalized by the GPU because it leads to thread divergence during parallel execution. *Thread divergence* refers to threads taking different paths of a conditional branch. Such threads must run serially, which can cause serious performance degradations. In our simulations, we have used an alternative implementation of the Heaviside function in which the *if* statement is replaced by multiplication with a predicate; see Fig. 4.

```
Heaviside(x, th, a, b){
    if (x>th)
        return a;
    else
        return b;
}
```

```
c = b-a;
…
Heaviside(x, th, a, c){
    return a + (x>th)*c;
}
```

Figure 4. Heaviside function implementation by if-statement (left) and by multiplication with a predicate (right).

A central concern in the implementation of the reaction term is the number of registers used per thread. The total number of threads per block and the number of registers per thread should be chosen to best utilize the available computing resources. The relation among these quantities, as given by the CUDA Programming Guide [32], is

$$\frac{R}{B \times ceil(T,32)}$$

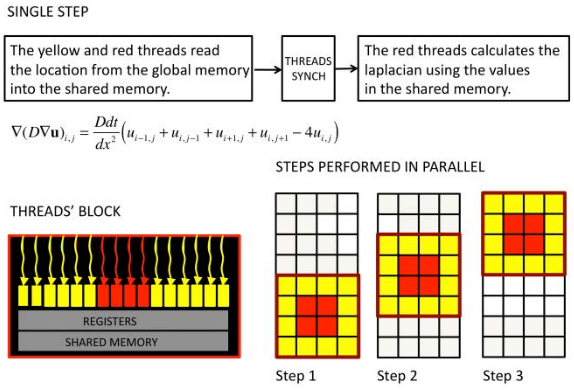
where R is the total number of registers per multiprocessor (a device-specific quantity), B is the number of active blocks per multiprocessor, T is the number of threads per block, and ceil(T,32) is T rounded up to the closest multiple of 32. Having multiple active blocks for each multiprocessor ensures that the multiprocessor will not be idle during thread synchronization or device memory access. By overlapping execution of blocks that wait and blocks that can run, the multiprocessor is able to hide the communication latency better.

In simple cardiac models with only a small number of variables (two or four), it is possible and in fact is advisable to implement the solution of the reaction term as a single kernel. In this case, the number of registers used per thread is usually less than 32, so that 2 or more active thread blocks of 256 threads can be executed by the same multiprocessor (with a device equipped with 16KB of registers for each multiprocessor). In more complex cardiac models having more than four variables, use of a single kernel to solve the reaction term is not recommended and often is not possible because the number of registers available per thread is insufficient. In this case, the solution of the reaction term is implemented as a sequence of multiple kernel invocations, with each kernel devoted to solving a group of related variables. Because a kernel invocation may modify the input of the following kernel, it is necessary to resolve these dependencies by buffering the variables that are common input among the kernels. Every kernel invocation introduces an overhead. To optimize the performance of our implementation with multiple kernels, we used the visual profiler provided by the recent CUDA SDK to find the best trade-off between kernel splitting, resource utilization, and the kernel invocation overhead.

5. DIFFUSION TERM

Cardiac models also include a diffusion term that spatially couples the main variable (membrane potential). Solving the diffusion term essentially consists of calculating the Laplacian operator for all of the grid points. This operation requires frequent access to values of neighboring cells. The use of the global memory is not desirable for this operation, because the specific memory-access pattern that the threads should follow in order to read from the neighbor cells is not coalesced [32], which reduces performance considerably. To solve this problem more efficiently, we consider two solutions, one using shared memory and the other using texture memory [28, 37].

In the shared-memory approach, the grid points can be subdivided easily into smaller overlapping parts (see Fig. 5), which then can be assigned to the threads' blocks. The values at neighboring elements are then read using shared memory within a block. This operation is performed by all the threads of the block, which control both the yellow and the red elements shown in Fig. 5. After synchronizing among the threads belonging to the same block, the threads controlling the red cells read the neighborhood collected in the shared memory and write in their cell the updated value of the Laplacian. This solution can be used with both single- and double-precision implementations, but the drawback is that it needs to use more threads than the number of matrix elements.



The yellow and red threads read the location from the global memory into the shared memory.

The red threads calculates the laplacian using the values in the shared memory.

$$\nabla(D\nabla\mathbf{u})_{i,j} = \frac{Ddt}{dx^2}\left(u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1} - 4u_{i,j}\right)$$

Figure 5. Calculating the diffusion term using shared memory.

An alternative approach that we have considered is to use the texture memory, which provides a cache that is optimized for 1D, 2D or 3D spatial locality, so that threads that read texture addresses that are close together will achieve the best performance. Currently, it is not possible to bind the texture to double-precision data, so the use of the texture memory for implementing the diffusion term is restricted only to single-precision implementations.

6. SIMULATION RESULTS

In this section, we present 2D simulations of the five models of interest and analyze their performance. Four square grids of increasing size are used to assess how the performance scales with the number of nodes. Although the larger grid sizes are physiologically unrealistic for 2D human cardiac surfaces, the numbers of nodes they contain are similar to what would be required for some 3D implementations. Note that because of the necessity of representing information from neighboring thread blocks in shared-memory implementations, our 16 x 16 thread blocks are effectively 14 x 14 for the shared-memory implementations. Therefore, the grid sizes in the shared-memory implementations (512, 1024, 1536, and 2048) are slightly different than those in the texture-memory implementations (520, 1038, 1556, and 2074) in all cases.

6.1 Karma Model (2 Variables)

The Karma model is a simplified model of cardiac electrophysiology that reproduces some basic features of cardiac dynamics, including wavelength oscillations, which can be seen in Fig. 6. To quantify GPU performance, we initiated a spiral wave [14] using the Karma model in square grids with each side consisting of 512, 1024, 1536, or 2048 elements (corresponding to 2^{18}, 2^{20}, $1.125\text{x}2^{21}$, and 2^{22} grid points, respectively), as shown in Fig. 6. Note that the wavelength of a spiral wave in this model is smaller than that of the human ventricular models (compare Figs. 9 and 12). We used three different implementations: double precision, single precision using shared memory to calculate the diffusion term, and single precision with the texture memory used for the diffusion term. The double-precision simulation required just over twice as much time as the corresponding single-precision simulation. For the single-precision simulations, use of the texture memory for the diffusion term improved performance. For the smallest grid size, which was similar in size to the surface area (epicardium) of a human ventricle, the simulation times were almost real time for the shared memory implementations, and the simulation was faster than real-time for single precision using the texture memory.

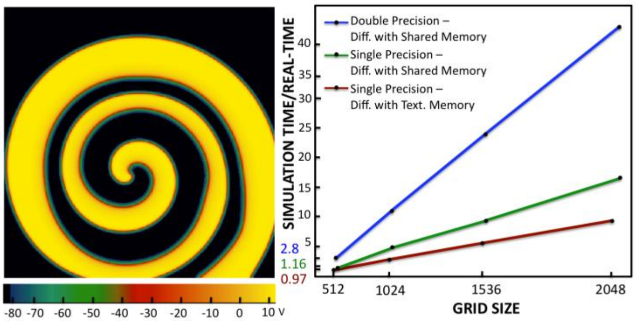


Figure 6. Left: Spiral wave using the Karma model in a 512 x 512 tissue (12.8 cm x 12.8 cm; resolution 0.0262 cm). Right: Simulation time normalized to real time for computing 1 s using different grid sizes.

6.2 BCF Model (4 Variables)

The BCF model is a minimal model of cardiac electrophysiology that reproduces many properties of cardiac tissue and can be parameterized [3, 5] in many cases to reproduce the dynamics of more complex models as well as experimental data. As with the Karma model, we initiated a spiral wave using the BCF model for square grids with each side consisting of 512, 1024, 1536, or 2048 elements and used the same three implementations: double precision, single precision with shared memory, and single precision with texture memory, as shown in Fig. 7. For the BCF model, the double precision simulation required almost three times as much time as the corresponding single-precision simulation. For the single-precision simulations, use of the texture memory for the diffusion term improved performance, but not by as large a factor as for the Karma model. For the smallest grid size, the simulation times were between a factor of 2 and 5 times greater than real time for all three implementations.

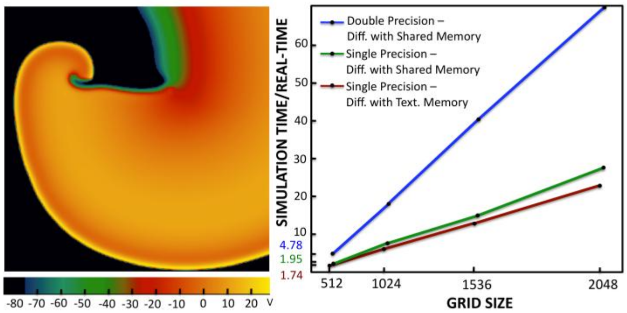


Figure 7. Left: Spiral wave using the BCF model in a 512 x 512 tissue (12.8 cm x 12.8 cm; resolution 0.025 cm). Right: Simulation time normalized to real time for computing 1 s using different grid sizes.

6.3 BR Model (8 Variables)

The BR model is an 8-variable model of cardiac electrophysiology that was the first detailed model of mammalian ventricular cell electrophysiology. As with the Karma and BCF models, a spiral wave was initiated using the BR model for square grids with each side consisting of 512, 1024, 1536, or 2048 elements and the performance of the same three implementations (double precision, single precision with shared memory, and single precision with texture memory) was quantified, as shown in Fig. 8. For the BR model, the double-precision simulation was about two times slower than the corresponding single-precision simulation. As with the Karma model, use of the texture memory for calculation of the diffusion term improved performance significantly for the single precision case. For the smallest grid size, the simulation times for the three implementations were between a factor of 10 and 25 times longer than real time.

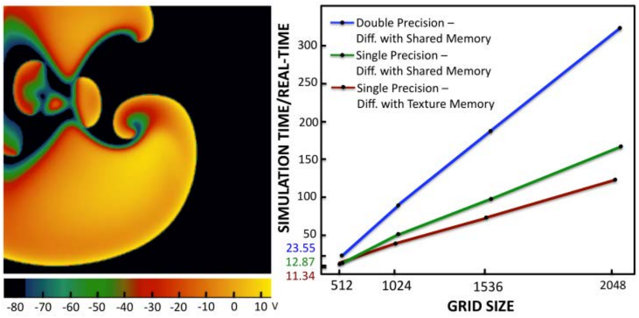


Figure 8. Left: Spiral wave using the BR model in a 512 x 512 tissue (10.24 cm x 10.24 cm; resolution 0.02 cm). Right: Simulation time normalized to real time for computing 1 s using different grid sizes.

6.4 TP Model (19 Variables)

The TP model is a 19-variable model that describes the electrophysiology of human ventricular cells. As with the previous models, a spiral wave was initiated using the TP model for square grids with each side consisting of 512, 1024, 1536, or 2048 elements and the performance of the same three implementations was quantified, as shown in Fig. 9. For the TP model, the double-precision simulation was about two to three times slower than the corresponding single-precision simulation. Use of the texture memory for calculation of the diffusion term resulted in a substantial performance improvement: for the largest grid size, the texture-memory simulation required only half as much time as the corresponding shared-memory simulation. At the smallest grid size, the simulation times for the three implementations were between a factor of 35 and 70 times longer than real time.

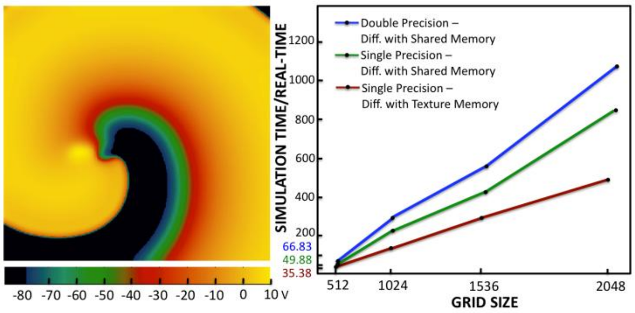


Figure 9. Left: Spiral wave using the TP model in a 512 x 512 tissue (10.24 cm x 10.24 cm; resolution 0.02 cm). Right: Simulation time normalized to real time for computing 1 s using different grid sizes.

For the models discussed so far, no significant differences were observed between the single- and double-precision simulations. We know, however, that for some more biophysically detailed models, including the TP model, single precision is not sufficient to represent small but important changes in the intracellular K^+ and intracellular Na^+ concentrations over the course of each action potential. Thus, in the single-precision simulations of the TP model, very small changes in concentration were represented as zeros, which produced non-smooth time traces of these concentrations within a single action potential. Although the concentration differences between single and double precision over one action potential were slight, the difference accumulated over time and changed not only the value of the concentration but also the trend of the concentration over time, especially for the K^+ concentration, as shown in Fig. 10. The differences in concentrations affected the time progression of spiral waves generated using single and double precision. Fig. 11 shows snapshots of spiral waves obtained after 600 s (10 min) of simulation time and indicates that the waves are at different points in their rotation paths.

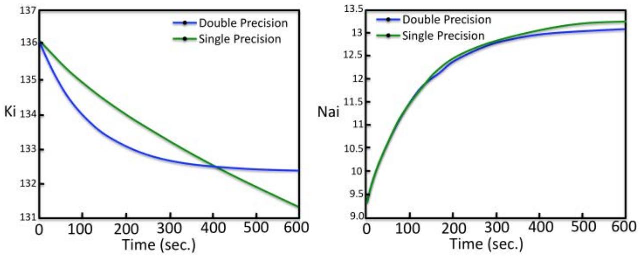


Figure 10. Time evolution of the intracellular K^+ (left) and Na^+ (right) concentrations observed at a representative grid point for the TP model with single and double precision.

As Figs. 10 and 11 show, although single precision is faster, it may introduce errors. Therefore, it is necessary to determine if the tradeoff between speed and accuracy is acceptable when making predictions using these models. For some very specific cases, such as when simply reproducing activation maps that show how a wave propagates through the atria or ventricle, single precision may be acceptable. For most other types of studies, such as those involving fibrillation, use of double precision probably is necessary.

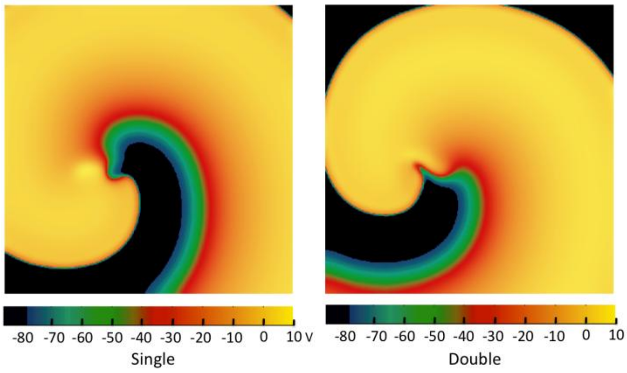


Figure 11. Spiral waves generated with the TP model using single precision (left) and double precision (right) after 10 mins.

6.5 IMW Model (67 Variables)

The IMW model is a 67-variable model that describes the electrophysiology of human ventricular cells in more detail than the TP model. As with the previous models, a spiral wave was initiated for square grids with each side consisting of 512, 1024, 1536, or 2048 elements and the performance of the same three implementations was quantified, as shown in Fig. 12. For the IMW model, the double-precision simulation was about twice as slow as the corresponding single-precision simulation. As with the Karma, BR, and TP models, use of the texture memory for calculation of the diffusion term improved performance significantly. At the smallest grid size, the simulation times for the three implementations ranged from 680 to 1300 times longer than real time. As with the TP model, double precision is necessary for adequate representation of ion concentrations.

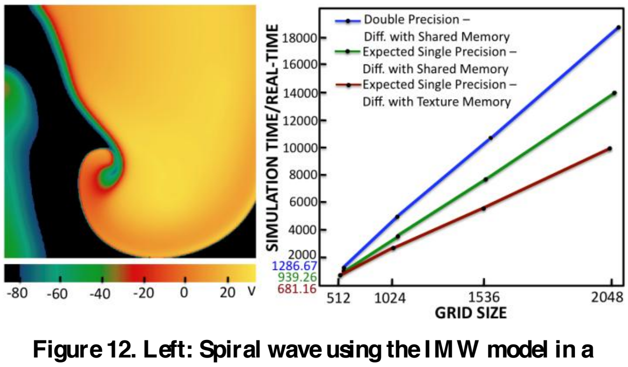


Figure 12. Left: Spiral wave using the IMW model in a 512 x 512 tissue (10.24 cm x 10.24 cm; resolution 0.02 cm). Right: Simulation time normalized to real time for computing 1 s using different grid sizes.

7. PERFORMANCE

Fig. 13 shows the performance for the different grid sizes as a function of the number of model variables. For the models with 4, 8, and 19 variables, the simulation time scales linearly with the number of variables. For the IMW model (67 variables), the departure from linear scaling can be explained by several factors. First, it is necessary to split the solutions of the ordinary different equations into 21 kernel calls. As a result, for any variable needed by more than one kernel, it is necessary to duplicate calculation of that variable within each such kernel to avoid communication between kernels. This duplication results in increased overhead for every integration step computed. In addition, the IMW code was not optimized as fully as the codes for the other models were (in terms of lookup tables, division eliminations, etc.).

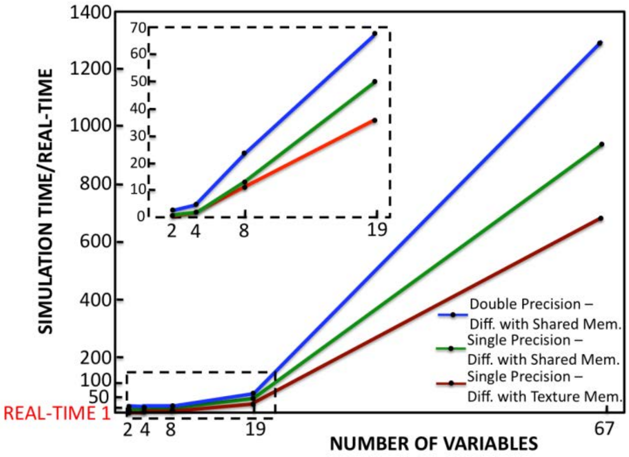


Figure 13. Simulation time normalized to real time as a function of the number of model variables.

We expect to obtain better performance for all the models by using other integration methods for the diffusion term, such as the alternating direction implicit scheme, which allows the use of larger integration time steps [8].

We also intend to extend our results to 3D and realistic cardiac anatomies. We note that we do not expect a significant change in performance in 3D. The number of grid points in the largest grid sizes analyzed here (more than 4 million nodes) is similar to the number of grid points that would be required for some 3D implementations. In considering potential clinical applications, we note that there are two categories to consider, simulatiosn of atrial arrhythmias and simulations of ventricular arrhythmias. Because of a lack of effective treatment options for some atrial arrhythmias, we expect that real-time simulations would be more immediately applicable to atrial arrhythmias. Anatomically, the atria are significantly smaller and thinner than the ventricles, so that the number of grid points required to implement realistic simulations of atrial arrhythmias should be on the order of the number of grid points in our largest 2D simulations here. Extending the code to 3D is expected to be relatively straightforward, with the possible exception of the diffusion term, which will incorporate information from a greater number of grid points. However, we note that only one variable (the membrane potential *V*) diffuses. In addition, as the number of model variables increases, the diffusion term occupies a decreasing fraction of the total computational effort. For this reason, although more complex models may require more computational resources overall, they are less affected by the expense of calculating the diffusion term. Thus, we expect that it will be computationally feasible to extend our implementations to perform near real-time simulations in realistic 3D atrial anatomies involving simpler electrophysiology models with more grid points or more complex electrophysiology models with fewer grid points.

8. RELATED WORK

Much effort has been devoted to accelerating applications of computational systems biology [18] and molecular biology [41] using large clusters of CPUs or application-specific integrated circuits (ASICs). These solutions are usually very expensive and may not be readily available to a broad group of users. This paper focuses on the use of GPUs as solution that can be implemented easily within a workstation.

Over the last five years, GPU performance has exceeded that of CPUs. As this trend continues [28], many branches of science requiring large-scale simulation, such as systems biology [7, 44], have turned to GPU implementations, and cardiac electrical dynamics is no exception [6]. Realistic human heart geometries at currently feasible spatial resolutions require that the tissue structures are discretized in meshes containing between 2^{24} and 2^{27} grid points [21, 34]. Each cell, in turn, involves a separate implementation of the mathematical equations describing its electrophysiology as explained in Section 3; these descriptions can be as simple as two [23] or as complicated as 67 [22] or 87 [13] ordinary differential equations. Even with a simple cell model, 0.6 seconds of simulation requires about 2 days using 32 CPUs [34]; for a more complex model, the same simulation time uses about 10 hours with 6144 CPUs [21].

GPUs have been shown to be superior to CPUs for reaction-diffusion equations similar to those used to describe cardiac dynamics in both 2 and 3 dimensions, with typical acceleration values between 5 and 40 depending on the algorithms used [28, 38] and to study turbulent dynamics generated by spiral wave dynamics [2, 53]. In addition, GPUs have been used for intracellular calcium dynamics within a single cell using Monte Carlo simulations, with a 15,000-fold reduction in time compared to previous studies [19]. GPUs have also been used to accelerate heart manipulations to enhance intervention simulations such as catheter positioning [51], surgical deformation [29], simple contractions [50, 52], and ECG generation [42, 43].

Although most simulations of cardiac electrical dynamics at the level described in this paper currently use CPUs, the use of GPUs is becoming increasingly common. The first simulation of cardiac arrhythmias using GPUs actually was performed on an Xbox 360 [40] using the BCF model [3] (Fig. 7). The first GPU-based simulation study of cardiac dynamics in 3D [39] involved the eight-variable Luo-Rudy I (LRI) model [26] and a realistic rabbit ventricular structure, with 1 s of simulation taking 72 minutes on a single GPU, compared to 45 minutes using a cluster of 32 CPUs. Since then, other studies have compared the speeds between CPUs and GPUs for different cardiac cell models. The 27-variable Mahajan et al. model [27] was reported to run 9 to 17 times faster (depending on tissue size) using 4 GPUs than 4 CPUs [48].

More recently, Rocha et al. [36] reported a gain of up to 20 times for a single GPU implementation compared to a parallel CPU implementation running with 4 threads on a quad–core machine, with parts of the code accelerated by a factor of 180 for the 8-variable LRI model [26] and the 19-variable TP model [46]. Lionetti et al. [24, 25] showed how different optimizations are needed for different cell models (two-variable FitzHugh-Nagumo [12], eight-variable BR [1], 18-variable Puglisi-Bers (PB) [35], 42-variable Grandi et al. [15], and 87-variable Flaim et al. [13]). In particular, they obtained a speedup of 6.7 for the 87-variable model.

In the remainder of this section, we relate our findings to prior efforts using GPUs to accelerate cardiac electrophysiology simulations. Sato et al. report 1 s of simulation in the 8-variable LR1 model in an 800x800 domain taking 283 s; in contrast, our simulations in the 8-variable-BR model (the two models are mathematically almost equivalent and share more than 90% of the same equations) take 11.34 s on a 512 x 512 domain and 39.2 s on a 1024 x 1024 domain (rescaling our times to the 800 x 800 domain results in a comparable speedup of a factor of 11). Vigmond et al. [48] report that 1 s of simulation time on 5 million nodes using the 27-variable Mahajan et al. model takes about 16 ksec (~4.5 h), whereas our 19-variable TP implementation in a 2048 x 2048 domain (close to 5 million nodes) takes about 8.2 min. However, a direct comparison is difficult to make as there is not only a difference of eight ODEs, but their simulations utilize a computationally more expensive bidomain approximation (used during simulations of defibrillation, where a Poisson equation needs to be solved at each time step).

Lionetti et al [24, 25] performed 300 ms of a heart beat simulation on a domain that consisted of 42,240 grid points to represent a ventricular section. Their main interest was to optimize the ODE portion of the reaction-diffusion system, so no spatial integration was performed and all the cells were decoupled. Therefore, their integration times did not include the spatial integration. They also used different optimization techniques for the different cell models considered. For the two-variable FHN model, 300 ms of simulation required 5.91 s; for the eight-variable BR model, 22.64 s; for the 18-variable PB model, 49.87 s; and for the 87-variable Flaim et al. model, 119.29 s. To compare with our simulations, in which the smallest domain consisted of 512 x 512 grid points (a domain about 6.2 times larger), and for 1 s of simulation time, we need to multiply their timing results by 20.5. Therefore, 1 s of simulation of the two-variable Karma model (with the same complexity as the FHN model) took 0.97 s vs. 121 s, the eight-variable BR model took 11.34 s vs. 464 s, the 19-variable TP model took 35.4 s vs. the 18-variable PB model 1022 s, and the 67-variable IMW 681 s vs. the 87-variable Flaim et al. model 2445 s. It is important to recall that the simulations by Lionetti et al. do not include the spatial integration component, making our timing results even more impressive in comparison.

Rocha et al. report simulations of the eight-variable LR1 and the 19-variable TP models for 500 ms for different 2D grid sizes (the largest of which was 640 x 640) using a higher spatial resolution of 0.01 cm. To compare with their results, we performed 500 ms simulations using the same domain size and spatial resolution. They report a simulation time of 11.4 minutes and 2.8 hours for the LR1 and the TP models, whereas we obtain for the BR and TP models 23.05 s and 285.56 sec on a C1060 card similar to theirs and 13.9 s and 105.4 s on a C2070 (Fermi-based) card. It is important to note that the times reported by Rocha et al. include outputting data at unspecified intervals; for comparison, our times include outputting a byte representation of the voltage at all nodes every 1 ms.

9. CONCLUSIONS

In summary, we have shown that we can achieve near real-time performance of simulated cardiac dynamics in tissues of realistic sizes by using GPU architectures. To achieve the maximum gains in computational efficiency, it is necessary to consider model-specific aspects of the implementation, including appropriate division of the model among multiple kernels and careful use of the available levels of memory. The significant performance gains should facilitate implementation of novel applications of simulation, including possible use in diagnosing cardiac disease or developing patient-specific treatment strategies.

10. ACKNOWLEDGMENTS

We would like to thank Riccardo Piergallini and Paolo Monteverde for granting us access to the Tesla C2070 with the Fermi chipset. We also would like to thank the anonymous reviewers for their valuable comments. The revised version of this paper is much improved thanks to their feedback. This work was supported in part by the National Science Foundation through grants No. CCF-0926190, CCF-1018459, and CDI-1028261; by the National Institutes of Health through grant No. R01HL089271-01A2; and by the Air Force Office of Scientific Research through grant No. FA0550-09-1-0481.

11. REFERENCES

[1] Beeler, G.W. and Reuter, H. 1977. Reconstruction of the action potential of ventricular myocardial fibres. *The Journal of Physiology*. 268, 1 (Jun. 1977), 177-210.

[2] Berg, S. et al. 2011. Synchronization based system identification of an extended excitable system. *Chaos*. in press, (2011).

[3] Bueno-Orovio, A. et al. 2008. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*. 253, 3 (Aug. 2008), 544-60.

[4] Cherry, E.M. and Fenton, F.H. 2008. Visualization of spiral and scroll waves in simulated and experimental cardiac tissue. *New Journal of Physics*. 10, 12 (2008), 125016.

[5] Cherry, E.M. et al. 2007. Pulmonary vein reentry--properties and size matter: insights from a computational analysis. *Heart Rhythm: The Official Journal of the Heart Rhythm Society*. 4, 12 (Dec. 2007), 1553-62.

[6] Clayton, R.H. et al. 2011. Models of cardiac tissue electrophysiology: progress, challenges and open questions. *Progress in Biophysics and Molecular Biology*. 104, 1-3 (2011), 22-48.

[7] Dematté, L. and Prandi, D. 2010. GPU computing for systems biology. *Briefings in Bioinformatics*. 11, 3 (May. 2010), 323-333.

[8] Fenton, F. and Karma, A. 1998. Vortex dynamics in three-dimensional continuous myocardium with fiber rotation: Filament instability and fibrillation. *Chaos*. 8, (1998), 20-47.

[9] Fenton, F.H. 1999. *Theoretical investigation of spiral and scroll wave instabilities underlying cardiac fibrillation*. Doctoral Thesis. Northeastern University.

[10] Fenton, F.H. and Cherry, E.M. 2008. Models of cardiac cell. *Scholarpedia*. 3, 8 (2008), 1868.

[11] Fenton, F.H. et al. 2002. Real-time computer simulations of excitable media: JAVA as a scientific language and as a wrapper for C and FORTRAN programs. *Bio Systems*. 64, 1-3 (Jan. 2002), 73-96.

[12] Fitzhugh, R. 1961. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*. 1, 6 (Jul. 1961), 445-466.

[13] Flaim, S.N. et al. 2006. Contributions of sustained INa and IKv43 to transmural heterogeneity of early repolarization and arrhythmogenesis in canine left ventricular myocytes. *American Journal of Physiology. Heart and Circulatory Physiology*. 291, 6 (Dec. 2006), H2617-2629.

[14] Frazier, D.W. et al. 1989. Stimulus-induced critical point. Mechanism for electrical initiation of reentry in normal canine myocardium. *The Journal of Clinical Investigation*. 83, 3 (Mar. 1989), 1039-52.

[15] Grandi, E. et al. 2010. A novel computational model of the human ventricular action potential and Ca transient. *Journal of Molecular and Cellular Cardiology*. 48, 1 (2010), 112-121.

[16] Gray, R.A. et al. 1995. Mechanisms of cardiac fibrillation. *Science (New York, N.Y.)*. 270, 5239 (Nov. 1995), 1222-3; author reply 1224-5.

[17] Gray, R.A. et al. 1998. Spatial and temporal organization during cardiac fibrillation. *Nature*. 392, 6671 (Mar. 1998), 75-8.

[18] Hagiescu, A. et al. Submitted. A platform-aware GPU realization of a systems biology application. *The 41st International Conference on Parallel Processing* (Taipei, Taiwan, Submitted).

[19] Hoang-Trong, T.M. et al. 2011. GPU-enabled stochastic spatiotemporal model of rat ventricular myocyte calcium dynamics. *Biophysical Journal*. 100, (Feb. 2011), 557.

[20] Hodgkin, L. and Huxley, A.F. 1952. A quantitative description of membrane currents and its application to conduction and excitation in nerve. *Journal of Physiology*. 117, (1952), 500-544.

[21] Hosoi, A. et al. 2010. A multi-scale heart simulation on massively parallel computers. *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (Washington, DC, USA, 2010), 1–11.

[22] Iyer, V. et al. 2004. A computational model of the human left-ventricular epicardial myocyte. *Biophysical Journal*. 87, 3 (Sep. 2004), 1507-1525.

[23] Karma, A. 1994. Electrical alternans and spiral wave breakup in cardiac tissue. *Chaos*. 4, 3 (Sep. 1994), 461-472.

[24] Lionetti, F.V. 2010. *GPU Accelerated Cardiac Electrophysiology*. Master's Thesis. University of California, San Diego.

[25] Lionetti, F.V. et al. 2010. Source-to-Source Optimization of CUDA C for GPU Accelerated Cardiac Cell Modeling. *Euro-Par 2010 - Parallel Processing*. P. D'Ambra et al., eds. Springer Berlin Heidelberg. 38-49.

[26] Luo, C.H. and Rudy, Y. 1991. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction. *Circulation Research*. 68, 6 (Jun. 1991), 1501-1526.

[27] Mahajan, A. et al. 2008. A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates. *Biophysical Journal*. 94, 2 (Jan. 2008), 392-410.

[28] Molnár Jr., F. et al. Simulation of reaction-diffusion processes in three dimensions using CUDA. *Chemometrics and Intelligent Laboratory Systems*. In Press, Corrected Proof.

[29] Mosegaard, J. et al. 2005. A GPU accelerated spring mass system for surgical simulation. *Medicine Meets Virtual Reality 13: The Magical Next Becomes the Medical Now*. IOS Press. 342-348.

[30] Nanthakumar, K. et al. 2007. Optical mapping of Langendorff-perfused human hearts: establishing a model for the study of ventricular fibrillation in humans. *American Journal of*

Physiology. Heart and Circulatory Physiology. 293, 1 (Jul. 2007), H875-880.

[31] Noble, D. 1962. A modification of the Hodgkin--Huxley equations applicable to Purkinje fibre action and pace-maker potentials. *The Journal of Physiology*. 160, (Feb. 1962), 317-352.

[32] NVIDIA CUDA Programming Guide v. 3.0: *http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf*.

[33] O'Hara, T. et al. 2011. Simulation of the undiseased human cardiac ventricular action potential: Model formulation and experimental validation. *PLoS Comput Biol*. 7, 5 (May. 2011), e1002061.

[34] Potse, M. et al. 2006. A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart. *IEEE Transactions on Bio-Medical Engineering*. 53, 12 Pt 1 (Dec. 2006), 2425-2435.

[35] Puglisi, J.L. and Bers, D.M. 2001. LabHEART: an interactive computer model of rabbit ventricular myocyte ion channels and Ca transport. *American Journal of Physiology. Cell Physiology*. 281, 6 (Dec. 2001), C2049-2060.

[36] Rocha, B.M. et al. 2011. Accelerating cardiac excitation spread simulations using graphics processing units. *Concurrency and Computation: Practice and Experience*. 23, 7 (May. 2011), 708-720.

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

[38] Sanderson, A.R. et al. 2008. A framework for exploring numerical solutions of advection–reaction–diffusion equations using a GPU-based approach. *Computing and Visualization in Science*. 12, 4 (Mar. 2008), 155-170.

[39] Sato, D. et al. 2009. Acceleration of cardiac tissue simulation with graphic processing units. *Medical & Biological Engineering & Computing*. 47, 9 (Sep. 2009), 1011-1015.

[40] Scarle, S. 2009. Implications of the Turing completeness of reaction-diffusion models, informed by GPGPU simulations on an XBox 360: cardiac arrhythmias, re-entry and the Halting problem. *Computational Biology and Chemistry*. 33, 4 (Aug. 2009), 253-260.

[41] Shaw, D.E. et al. 2007. Anton, a special-purpose machine for molecular dynamics simulation. *Proceedings of the 34th annual international symposium on computer architecture*. 35, 2 (Jun. 2007), 1–12.

[42] Shen, W. et al. 2009. GPU-based parallelization for computer simulation of electrocardiogram. *Computer and Information Technology, International Conference on* (Los Alamitos, CA, USA, 2009), 280-284.

[43] Shen, W. et al. 2010. Parallelized computation for computer simulation of electrocardiograms using personal computers with multi-core CPU and general-purpose GPU. *Computer Methods and Programs in Biomedicine*. 100, 1 (Oct. 2010), 87-96.

[44] Szafaryn, L.G. et al. 2009. Experiences accelerating MATLAB systems biology applications. *Proceedings of the Workshop on Biomedicine in Computing: Systems, Architectures, and Circuits*. (Jun. 2009), 1-4.

[45] Taggart, P. et al. 2000. Inhomogeneous transmural conduction during early ischaemia in patients with coronary artery disease. *Journal of Molecular and Cellular Cardiology*. 32, 4 (Apr. 2000), 621-630.

[46] Ten Tusscher, K.H.W.J. and Panfilov, A.V. 2006. Alternans and spiral breakup in a human ventricular tissue model. *American Journal of Physiology. Heart and Circulatory Physiology*. 291, 3 (Sep. 2006), H1088-1100.

[47] Trayanova, N.A. 2011. Whole-heart modeling: applications to cardiac electrophysiology and electromechanics. *Circulation Research*. 108, 1 (Jan. 2011), 113-128.

[48] Vigmond, E.J. et al. 2009. Near-real-time simulations of bioelectric activity in small mammalian hearts using graphical processing units. *Conference Proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. (2009), 3290-3293.

[49] Witkowski, F.X. et al. 1998. Spatiotemporal evolution of ventricular fibrillation. *Nature*. 392, 6671 (Mar. 1998), 78-82.

[50] Yu, R. et al. 2009. A framework for GPU-accelerated virtual cardiac intervention. *The International Journal of Virtual Reality*. 8, 1 (2009), 37-41.

[51] Yu, R. et al. 2010. Real-time and realistic simulation for cardiac intervention with GPU. 3, (Jan. 2010), 68-72.

[52] Yu, R. et al. 2010. GPU accelerated simulation of cardiac activities. *Journal of Computers*. 5, 11 (Nov. 2010).

[53] Zudrop, J. 2011. *Simulation of weak turbulent Rayleigh-Bénard convection on a GPU*. Master's Thesis. Max-Planck-Institute for Dynamics and Self-organization.