UNIVERSIDAD TECNOLÓGICA DE PEREIRA

Propagación del Potencial Eléctrico en un Tejido 2D de Aurícula Humana utilizando Unidades de Procesamiento Gráfico (GPU)

por

John H. Osorio-Ríos

Proyecto de Grado para optar al título de
de Máster en Ingeniería de Sistemas y Computación

en la fecha
Facultad de Ingenierías
Ingeniería de Sistemas y Computación

9 de diciembre de 2016

Índice general

Capítulo 1

Planteamiento del Problema

Los modelos computacionales de tejido cardíaco han generado gran cantidad de información acerca de la complejidad de las interacciones entre las propiedades de la membrana y la heterogeneidad estructural del corazón. La fortaleza de los modelos computacionales, radica en que proporcionan una plataforma versátil para que los investigadores puedan interpretar datos experimentales y realizar nuevas predicciones sobre hipótesis que no son observables o comprobables usando las técnicas experimentales actuales [1].

En la clínica, hacen incursión los modelos matemáticos ya que permiten el desarrollo de técnicas especializadas y la utilización de robots (cirugía laparoscópica, daVinci Robotic assisted Heart Surgery) en cirugía asistida. El uso de modelos electro-anatómicos ha permitido también el estudio de fármacos (medicamentos) y su efectividad frente a patologías bien identificadas.

Los modelos computacionales de tejido cardiaco están basados en la solución de sistemas de ecuaciones diferenciales parciales y ordinarias no lineales las cuales después de aplicarles algún método de discretización (diferencias finitas, elementos finitos) para su solución resultan en un conjunto de operaciones entre matrices del orden de millones lo cual se traduce en programas con tiempos de ejecución elevados, en los cuales el desempeño de los algoritmos es menor comparado con su versión paralela [2].

Se han establecido diversas técnicas computacionales para disminuir los tiempos de ejecución que van desde el diseño de nuevos algoritmos que disminuyan la complejidad, hasta la utilización de arquitecturas paralelas que permitan dividir el trabajo a realizarse entre distintos procesadores, lo cual se traduce en la disminución de los tiempos de ejecución.

En la década de 1980, aparecen entonces las Unidades de Procesamiento Gráfico **GPU** (Graphics Processing Units) [3], las cuales cuentan con cientos o miles de procesadores

que son aprovechados para disminuir los tiempos de ejecución de los algoritmos. La utilización de estos procesadores (GPU), que se encuentran en las tarjetas gráficas, se debe hacer a través de frameworks especializados que permitan acceder a los recursos computacionales que allí residen.

Desde el año 2007 existen dos opciones, CUDA (Compute Unified Device Architecture) y OpenCL (Open Computing Language), las cuales partiendo del principio de paralelismo de Datos, logran explotar al máximo la capacidad de cómputo existente en estos dispositivos. CUDA como herramienta para realizar paralelización de algoritmos, es altamente utilizada en el campo de la computación científica, [4] [5] ya que nVidia cuenta con una línea de tarjetas gráficas conocidas como Tesla, entre otras orientadas a diversos campos, que son utilizadas en la gran mayoría de centros de supercomputación en el mundo y además cuenta con una gran comunidad de desarrolladores [6] que han hecho de CUDA la herramienta primordial para realizar aceleración de algoritmos.

Se plantea realizar el proceso de implementación del modelo celular para aurícula humana en dos dimensiones planteado por Courtemanche [7] utilizando CUDA como *framework* de trabajo para programar Unidades de Procesamiento Gráfico (GPU) y una estrategia de visualización de datos científicos usando Paraview [8].

Capítulo 2

Justificación

Desde el año 2007 con la invención de CUDA, ha aumentado la utilización de herramientas de cómputo que permitan realizar procesamiento de datos de una forma rápida y precisa. En el campo de la computación científica este es un hecho que se puede evidenciar en los cientos de centros de supercomputación que se encuentran en el listado top500 [5] y green500 [9], también se puede evidenciar una tendencia cada vez mayor a construir procesadores multi-núcleo, lo que permite al usuario final contar con gran capacidad de cómputo en los equipos de escritorio [10] [11] [12].

La predicción del clima, el movimiento de partículas, el análisis de imágenes, la biología computacional, el análisis de mercados, son algunos de los temas que pueden aprovechar el poder computacional de plataformas que cuentan con cientos o miles de procesadores. En este orden de ideas existe un área de creciente interés en el país y en la región, en donde se pueden ver iniciativas como BIOS [13], que plantea hacer frente a los problemas de procesamiento e investigación en estas áreas temáticas, adicionalmente se pueden mencionar los esfuerzos del grupo GITIR de la Universidad de Caldas y del Grupo de Automática de la Universidad Tecnológica de Pereira.

La biología computacional, es la utilización de las ciencias de la computación, estadística y matemáticas para la solución de problemas biológicos [14]. Generalmente los problemas a solucionar en este campo científico necesitan de algoritmos y hardware que permitan su aceleración para obtener resultados en tiempos menores [12] [15]. El modelo cardiaco es un ejemplo de un problema que puede resolverse desde la óptica de la computación científica, como se plantea en Alya Red [16], éste es un campo de aplicación que implica la solución de diversos sistemas de ecuaciones diferenciales, lo que resulta en operaciones matriciales de gran escala [17].

El comportamiento del corazón generalmente es modelado utilizando dos enfoques [17], un modelo eléctrico y un modelo mecánico, el presente proyecto busca la implementación de un modelo netamente eléctrico [18] [7] apoyado en la investigación realizada por [19], donde se estudie la posibilidad de realizar un proceso de optimización de tiempos de ejecución utilizando Unidades de Procesamiento Gráfico (GPU).

La paralelización de este tipo de algoritmos permitirá en primera instancia comprobar que la utilización de procesadores masivamente paralelos es útil en este tipo de investigaciones, también en el campo médico a futuro sería posible realizar modelos computacionales de corazón por individuos, permitiendo así la construcción de corazones *in-silico* que representen a pacientes particulares y evitar así generalizaciones en los tratamientos. Esto sería posible si los tiempos de simulación decrementan, buscando llegar como meta a simulaciones en tiempo real.

Adicionalmente, los resultados obtenidos en estas investigaciones generan gran cantidad de datos que representan la manera en la cuál se distribuye el potencial eléctrico sobre el tejido cardiaco, en este punto es donde se construirá una estrategia de visualización de datos científicos que permita tener mayor claridad de este fenómeno. Ésta estrategia permitirá la visualización de los datos mientras el proceso de cálculo computacional esta siendo llevado a cabo.

Las investigaciones en Colombia en este tipo de proyectos se han dado desde el punto de vista electrofisiológico en la Universidad de Caldas y en la Universidad de Antioquia, sin embargo no hay reportes donde se mencione la utilización de procesadores masivamente paralelos, como las GPU, para atacar el problema de procesamiento, y disminuir tiempos de simulación.

Finalmente la implementación de dicho modelo espera ayudar a los médicos en la experimentación de fármacos y en el diagnóstico de enfermedades que afecten el corazón de los pacientes.

Capítulo 3

Objetivos

3.1. Objetivo General

Implementar algoritmos eficientes para la propagación del potencial eléctrico en un tejido 2D utilizando CUDA como *framework* para programar Unidades de Procesamiento Gráfico (GPU)

3.2. Objetivos Específicos

- Determinar el método de discretización espacial de las ecuaciones diferenciales que rigen la propagación del potencial eléctrico.
- Implementar y simular la propagación del potencial de acción en un arreglo de células cardíacas en 1D y 2D a través de CUDA.
- Implementar un esquema de visualización científica en tiempo de ejecución utilizando Paraview.
- Evaluar el desempeño del modelo construido en C y CUDA.

Capítulo 4

Metodología Seguida en el Proyecto

El desarrollo de la investigación fue realizado desde un punto de vista incremental, en primera instancia un conocimiento básico de conceptos de electrofisiología cardíaca fue necesario, en este paso se hizo un levantamiento de información sobre el modelo eléctrico de aurícula de Courtemanche et al. [7].

Una vez se comprendió de manera básica y apoyados en los expertos en el área, se procedió a la construcción del algoritmo que simula una célula auricular utilizando C como lenguaje de programación, ésta elección se hizo debido a que era necesario pensar en lenguajes compilados que permitieran garantizar tiempos de ejecución bajos. Una vez el modelo programado se finalizó, se determinó con el director del proyecto que fuera evaluado por él mismo y por el Phd. Carlos Alberto Ruiz, quienes dieron fe de que el modelo generaba los potenciales de acción acordes al modelo implementado.

Una vez el modelo fue programado, se procedió a construir un algoritmo que enlazara un conjunto de células cardíacas formando una fibra, esto puede verse en la figura 5.12, lo que permitió evaluar la capacidad de propagación de potencial de acción del modelo de Courtemanche implementado, se utilizó el método de diferencias finitas para solucionar el problema desde el punto de vista espacial y Crank-Nicholson para solucionar el modelo temporal.

Después de la construcción del modelo de propagación en 1D, se construyó el modelo en 2D, lo que permitió la simulación de un tejido de aurícula y evaluar como puede verse en las figuras del capítulo de resultados, que el modelo fue bien implementado.

Al finalizar la construcción del tejido, se procedió a realizar el proceso de implementación del modelo en 2D sobre arquitecturas masivamente paralelas, en este proyecto se

utilizó una GPU. Como puede verse de nuevo en los resultados, las comparativas permitieron definir que la implementación usando CUDA fue sustancialmente superior al comparársele con la versión en CPU. Se utilizó una librería avanzada como ArrayFire [20] para la solución espacial del modelo, de igual manera algunos kernels propios fueron adicionados a la implementación.

Finalmente una estrategia de visualización de datos fue implementada, en este caso se utilizó Paraview Catalyst [8] [21] para realizar ésta tarea. El proceso de visualización en tiempo de ejecución (*in-situ*) fue implementado a través de Paraview Catalyst que permite tener un nodo de cómputo realizando el proceso de renderizado, mientras otro nodo se encarga de realizar el proceso de ejecución del modelo del tejido. Un script en python que usa Paraview-Catalyst fue desarrollado en ésta fase para que ambos nodos pudieran comunicarse.

Capítulo 5

Marco Teórico

Desde el año 2007, cuando nVidia creó CUDA (Compute Unified Device Architecture) [6], se ha iniciado un proceso de estudio e investigación en el área de la computación científica sin precedentes, ya que gracias a la gran penetración de las tarjetas gráficas en el mercado se puede contar con dispositivos de gran capacidad de cómputo en equipos de escritorio.

La evolución de las GPUs ha pasado por la creación de distintas arquitecturas, cada una superando en recursos a su predecesor. A continuación se mencionarán algunas de las arquitecturas existentes.

5.1. Arquitectura G80

Cuando nVidia lanzó al mercado la tarjeta GeForce 8800 se conoció entonces la arquitectura G80, la cual era la primera GPU en el mercado que utilizaba una arquitectura unificada que ejecuta los vértices, la geometría, los píxeles y los programas de computación, además se basa en un modelo de instrucciones conocido como SIMT (*Single Instruction Multiple Threads*) el cual ejecuta múltiples hilos de manera independiente y al mismo tiempo utilizando una sola instrucción. Esta GPU soporta lenguaje C, lo que hace que los programadores no tengan que aprender otro lenguaje y opera de forma integral con una precisión de 32 bits para operaciones de punto flotante ajustándose al estándar IEEE 754 [22].

En la Figura 5.1 se muestra un esquema completo de la arquitectura G80. Ésta arquitectura de nVidia comenzó su desarrollo a mediados de 2002, publicando su versión definitiva en los últimos meses de 2006 conocida como GT200. El objetivo básico de la mejora de las capacidades de procesamiento gráfico llevó a:

- Incremento de forma significativa de las prestaciones con respecto a la última generación de GPUs.
- Aumento de la capacidad de cálculo de punto flotante por parte de la GPU, con la vista puesta en la introducción definitiva de este tipo de procesadores en el ámbito de la computación general.
- Adición de nuevos elementos al *pipeline* clásico, para cumplir las características definidas por Microsoft en DirectX 10.

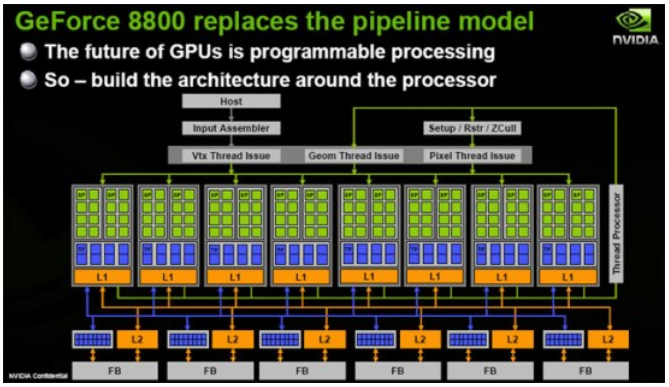

FIGURA 5.1: Arquitectura G80

Una ventaja en esta arquitectura es la posibilidad de equilibrar la carga entre los distintos procesadores, los cuales se pueden asignar a una tarea u otra dependiendo del trabajo a realizar. Como desventaja se puede considerar la mayor complejidad de los procesadores y su no especificidad en un tipo de problemas.

5.2. Architectura Fermi

La arquitectura Fermi ver figura 5.2 dio un salto importante en la arquitectura de las GPU. G80 era una visión general de una arquitectura unificada y paralela. La arquitectura GT200 extendió el rendimiento y la funcionalidad de la arquitectura G80. Esta arquitectura emplea un enfoque completamente nuevo a la hora de diseñar la GPU. Las áreas principales en las que centraron fueron [22]:

- Mejora del rendimiento de precisión doble. El rendimiento en Operaciones de punto flotante aumento 10 veces en comparación con el rendimiento de una CPU.
- Jerarquía de memoria Cache. Cuando un algoritmo paralelo no puede utilizar memoria compartida los usuarios pueden tener acceso a una jerarquía de memoria caché real.

- Más memoria compartida. Lo programadores CUDA tuvieron acceso a más de 16KB de memoria compartida.
- Cambios de Contexto. Se puede realizar cambios de contexto más rápidos entre las aplicaciones, gráficos y cálculos.
- Operaciones más Rápidas. Se obtiene mayor rapidez de lectura y escritura de las operaciones de algoritmos paralelos.

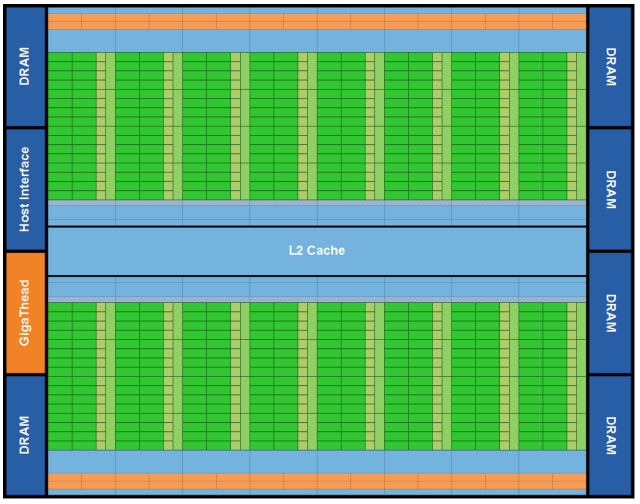

FIGURA 5.2: Arquitectura Fermi

La arquitectura Fermi es modular, cuya base son los SM (Streaming Multiprocessors), los que son arreglos de unidades de cómputo entre las que tenemos: procesadores de *Shaders*, unidades de textura, motores de teselado, entre otras. A su vez, los SM están organizados en arrays de 2 o 4 SMs, los que son denominados GPC (Graphic Processing Cluster), 1 o más GPCs agrupados conforman un GPU basado en la arquitectura Fermi.

5.3. Arquitectura Kepler

A medida que aumenta la demanda del alto rendimiento en la computación paralela desde muchos ámbitos de la ciencia como la medicina, la ingeniería y finanzas; *nVidia* sigue innovando para satisfacer esta demanda con nuevas arquitecturas de GPUs que son extraordinariamente poderosas. Las GPUs de *nVidia* van redefiniendo y acelerando sus capacidades en áreas como las simulaciones bioquímicas, procesamiento de señales, finanzas, ingeniería asistida por computador, dinámica de fluidos computacionales y análisis de datos. La arquitectura Kepler GK110 GPU ayudará a resolver la mayor parte de problemas informáticos al ofrecer mucha más potencia de procesamiento en comparación

a las arquitecturas anteriores proporcionando nuevos métodos para optimizar y aumentar la carga de trabajo de ejecución en paralelo revolucionando la computación de alto rendimiento [23].

El objetivo de Kepler es ser mucho mejor en rendimiento. GK110 no solo supera a la arquitectura Fermi en potencia sino que también consume menos energía y genera menos calor. Una arquitectura Kepler GK110 incluye 15 unidades y seis controladores de memoria SMX de 64-bits. Las principales características de esta arquitectura son:

- Una nueva arquitectura de procesador SMX.
- Un subsistema de memoria, que ofrece capacidades adicionales de almacenamiento en caché, más ancho banda a nivel de la jerarquía, y una DRAM totalmente rediseñada.
- Soporte de *hardware* en todo el diseño para permitir nuevas capacidades del modelo de programación.

5.4. Arquitectura Maxwell

La nueva arquitectura de nVidia, lanzada en febrero de 2014, conocida como Maxwell, es una arquitectura de GPU que busca una mejora sustancial en el consumo de potencia comparándola directamente con su predecesora Kepler, existen dos tarjetas gráficas de éste tipo en el mercado la GeForce GTX 750 Ti y Geforce GTX 750.

5.4.1. Multiprocesadores más Eficientes

Los arquitectos de Maxwell vieron un gran potencial y una gran capacidad de mejora de la arquitectura Kepler ver Figura 5.3, de tal manera que buscaron dar un gran salto en la construcción de una nueva arquitectura que cuenta con Gestión Mejorada de Instrucciones, Aumento de Ocupación para Código Existente y Latencia de Instrucciones Aritméticas Reducida [24].

5.4.2. Memoria Compartida Dedicada y de Mayor Tamaño

A diferencia de la arquitectura Kepler, Maxwell aprovechará las cachés de nivel uno y las cachés de texturas para construir una sola unidad con mayor capacidad y mayor velocidad de acceso.

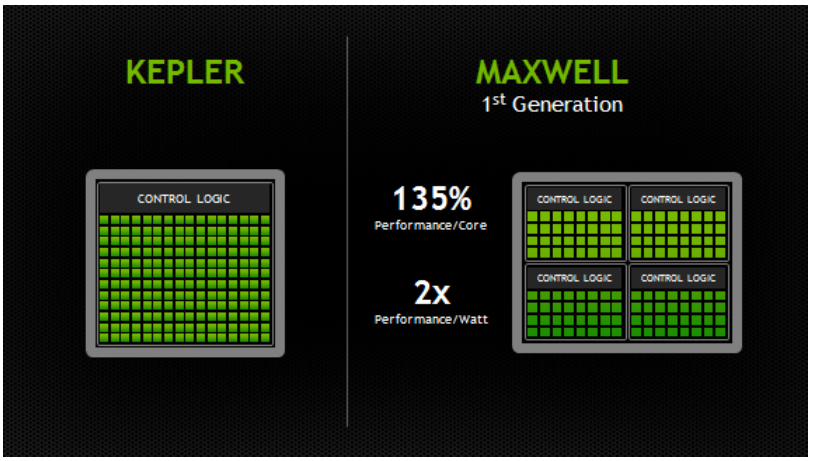

FIGURA 5.3: Mejoras de Eficiencia Energética Arquitectura Maxwell

5.4.3. Operaciones Atómicas en Memoria Compartida más Rápidas

Maxwell posee Operaciones Atómicas de Memoria Compartida de 32 y 64 bits de enteros de manera nativa, y también para realizar operacion CAS (Compare-and-Swap).

5.4.4. Soporte de Paralelismo Dinámico

La arquitectura Maxwell tendrá el soporte de paralelismo dinámico incluso para los chips de las lineas de bajo costo de nVidia.

5.5. Arquitectura Pascal

La arquitectura Pascal es la última de las creaciones de *nVidia*, su construcción está basada en un total de 15.3 billones de transistores, cuenta con un nuevo esquema de interconexión *peer-to-peer* y GPU-to-CPU que buscan disminuir la latencia entre *host* y *device*, nuevas tecnologías para simplificar las tareas de programación y adicionalmente una eficiencia en el consumo de potencia sin precedentes [25].

Las principales características de la arquitectura Pascal P100 son:

5.5.1. Desempeño Extremo Para Computación de Altas Prestaciones y Aprendizaje Profundo

La arquitectura Tesla P100 fue construida para entregar un desempeño excepcional para aplicaciones extremadamente demandantes, ver figura 5.4, en este orden de ideas registra lo siguiente:

- 5.3 TFLOPS de desempeño en operaciones de precisión doble (FP64)
- 10.6 TFLOPS de precisión simple (FP32)
- 21.2 TFLOPS de desempeño *half-precision* (FP16)

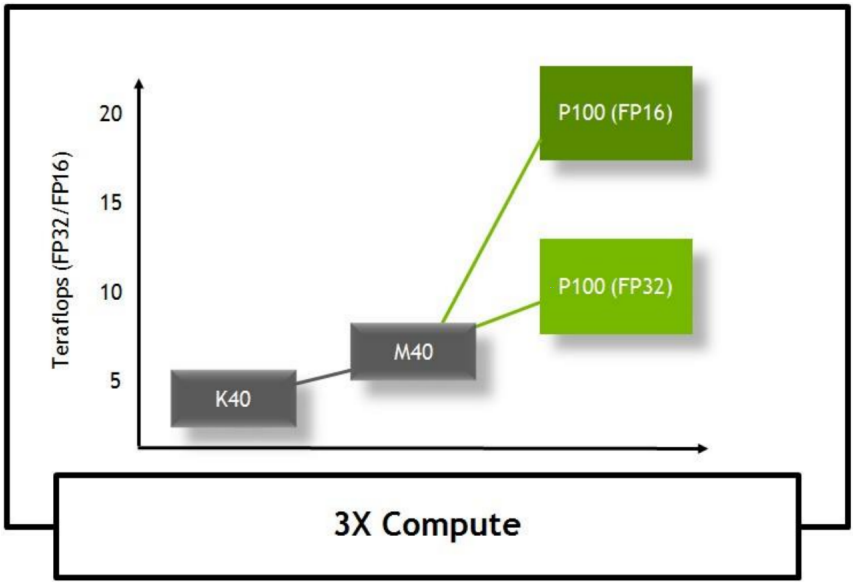

FIGURA 5.4: La arquitectura Tesla P100 supera generaciones anteriores

Adicionalmente a las numerosas áreas en las que la computación de alto desempeño tiene aplicación, *nVidia* ha tenido en cuenta el creciente interés investigativo en Aprendizaje Profundo (*Deep Learning*). Las GPUs de *nVidia* en la actualidad se encuentran en el estado del arte en el entrenamiento de redes neuronales profundas (*Deep Neural Networks*) e Inteligencia Artificial. Se han obtenido mejoras en desempeño de 10X y hasta 20x comparado con CPUs [25].

5.5.2. *NVLink*: Ancho de Banda Extraordinario en conectividad para Multi-GPU y GPU-to-CPU

El crecimiento de la computación científica y su uso a través de aceleradoras como las GPUs, se han construido diversos sistemas multi-GPU, desde estaciones de trabajo hasta servidores e incluso supercomputadores. Muchas configuraciones de sistemas de 4-GPU y 8-GPU están siendo utilizados para resolver problemas cada vez más complicados. Debido a esto los sistemas que utilizan buses PCIe de tercera generación han empezado a convertirse en cuellos de botella.

Este problema se ha solucionado en la presente generación de GPUs de *nVidia* a través de la creación de una nueva interface de interconexión de alta velocidad conocida como

nVLink, esta interface de comunicación mejora a los buses PCIe en un factor de 5x llegando a velocidades de transferencia entre GPUs de hasta 160 Gigabytes/segundo. En la figura 5.5 se puede ver la topología utilizada.

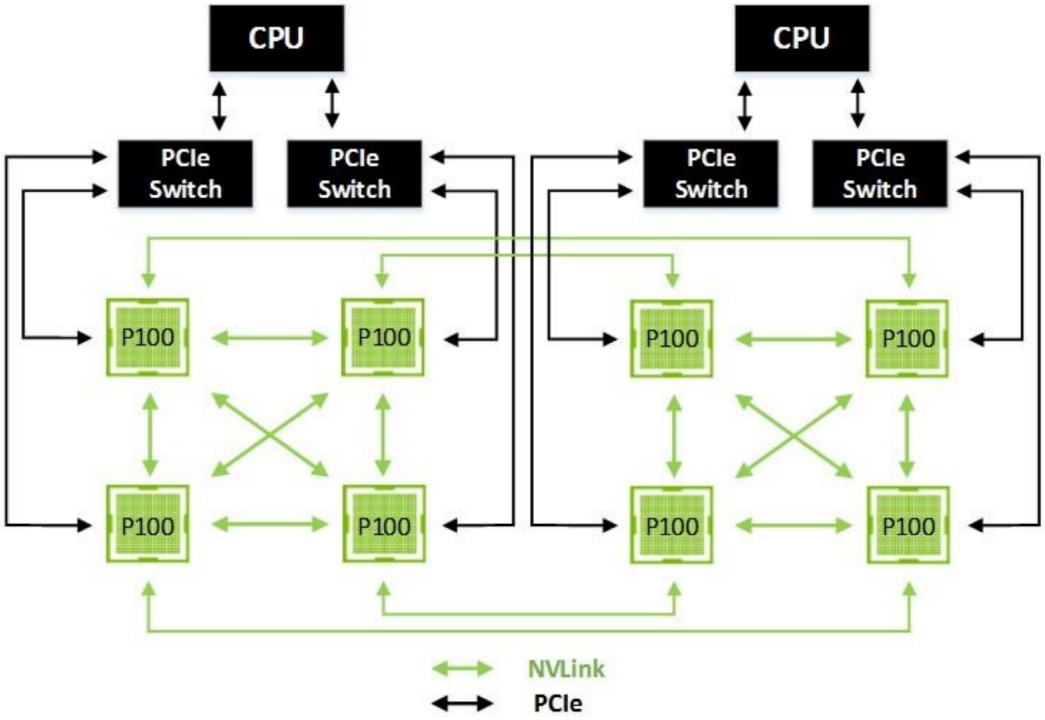

FIGURA 5.5: Bus *nVLink* conectando 8 GPUs

5.5.3. HBM2 Arquitectura de Memoria de GPU de Alta Velocidad

La arquitectura Tesla P100 es la primera arquitectura GPU del mundo que soporta el uso de memoria HBM2. Este tipo de memoria ofrece tres veces (3x) más ancho de banda que la GPU Maxwell GM200. Esto permite a la arquitectura P100 hacer frente a la gran cantidad de conjuntos de datos que se procesan, mejorando la eficiencia y el *throughput*, y a su vez reduciendo la frecuencia de transferencias desde la memoria del sistema.

La memoria HBM2 es memoria apilada y se encuentra localizada sobre el mismo *package* físico que la GPU, esto ocasiona considerables mejoras de espacio al compararse con la memoria GDDR5, esto permite la construcción de servidores GPU mucho más densos.

5.5.4. Modelo de Programación Simplificada para Desarrolladores con Memoria Unificada y *Compute Preemption*

La memoria unificada es un avance para la computación con GPUs de *nVidia* incluida en la arquitectura Pascal GP100. Este avance proveé un único espacio de memoria virtual

para CPU y GPU y reduce la curva de aprendizaje sobre arquitecturas heterogéneas. Los programadores ya no necesitan preocuparse sobre gestionar los datos entre dos sistemas de memoria virtual diferente.

Otra característica importante de Pascal radica en el uso de Compute Preemption, ésta característica permite que las tareas de cómputo sean interrumpidas a nivel de instrucciones y no a nivel de bloque de hilos como en Maxwell y Kepler; de esta manera se garantiza que las aplicaciones no monopolicen el sistema o superen tiempos de ejecución establecidos.

5.6. Bioelectricidad

La bioelectricidad es la disciplina que estudia los fenómenos eléctricos que ocurren en los tejidos biológicos. Su objeto de estudio es:

- El comportamiento del tejido excitable y sus fuentes eléctricas.
- Los efectos de las fuentes bioeléctricas dentro de un volumen conductor.
- Respuesta de las células excitables a los campos eléctricos y magnéticos.
- Las propiedades eléctricas intrínsecas de los tejidos asociadas con la biofísica de membranas y su comportamiento en torno a movilidad de electrolitos.

De este modo se puede plantear a la electro-fisiología cardiaca, como una rama de la bioelectricidad, la cual se encarga de estudiar el comportamiento eléctrico del corazón.

En general, la cantidad de sangre que el corazón bombea por el cuerpo cada minuto, es una función de cuánta sangre el corazón expulsa con cada latido, asociado con su frecuencia respectiva. Muchos factores trabajan en conjunto para regular la frecuencia cardiaca. Una comprensión profunda de los mecanismos normales de activación eléctrica cardiaca y factores modificantes son necesarios para comprender los estados de una enfermedad y su tratamiento apropiado [26].

5.6.1. Membranas, Iones y corrientes

La actividad eléctrica cardiaca es determinada por el potencial, o diferencia de voltaje entre los entornos externos e internos de una célula, ver [27]. Esta diferencia de potencial existe debido a que la membrana celular cardiaca es selectivamente permeable. Esta membrana está compuesta de una bi-capa de lípidos en la cual están situadas proteínas

especializadas que forman canales que permiten el paso de ciertos iones pseudo aleatoriamente o en algunos casos siguiendo distribuciones tipo Poisson o cadenas de Markov entre los espacios intra y extra celulares. Este paso de iones desde y hacia el exterior de la célula son los que permiten la creación del potencial de acción, el cual se puede definir como una función no lineal de voltaje que representa el proceso conjunto de despolarización-repolarización celular cardiaca en la figura 5.6 se puede ver un potencial de acción común y en 5.7 puede verse una descripción un poco más detallada de los potenciales de acción en todo el sistema de conducción y finalmente su integración por peso para conformar el electrocardiograma.

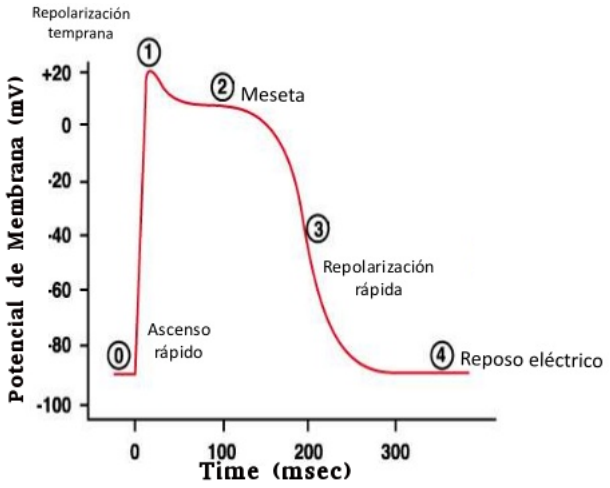

FIGURA 5.6: Potencial de acción de una célula común

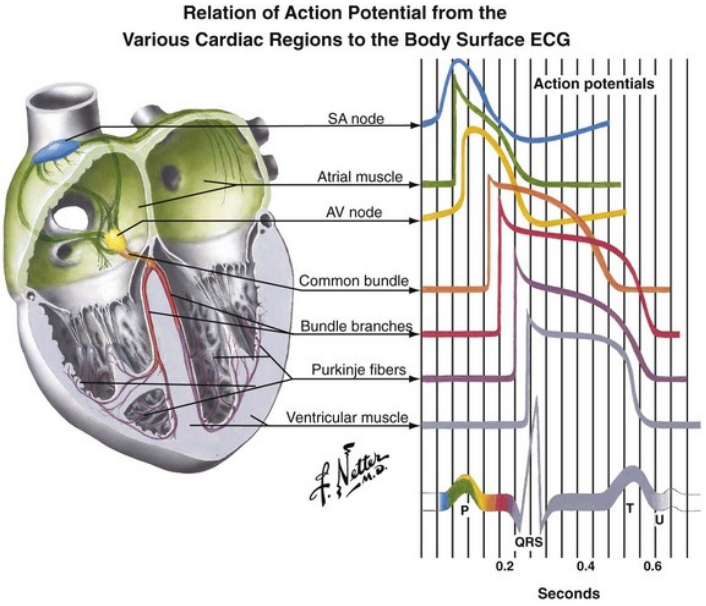

FIGURA 5.7: Relación de Potenciales de Acción a ECG. Imagen tomada de *Netter Images*

5.6.2. Modelo celular

El primer modelo matemático para describir el potencial de acción en la membrana celular fue el desarrollado por Hodking y Huxley(H-H) en 1952 y fue obtenido a partir de un axón de calamar gigante, este modelo es usado como base en la mayoría de los modelos de membrana celular cardíaca. [28] [19]. Su forma general está dada por la ecuación 5.1.

$$I_{ion} = g_{ion} \cdot x \cdot (V_m - E_{ion}) \tag{5.1}$$

Una breve historia de los modelos celulares cardíacos se encuentra en la tesis doctoral de Felipe Alonso Atienza[28]:

> "En 1962, Denis Noble publicó el primer modelo matemático de célula cardíaca,en el que se describía el potencial de acción de las fibras de Purkinje, a partir de la adaptación de las ecuaciones del modelo H-H. Posteriormente, en 1975, McAllister et al. mejoraron el modelo de Noble mediante la incorporación, en forma de nuevas corrientes iónicas, de nuevas evidencias experimentales sobre las propiedades de la membrana celular. Seguidamente, Beeler y Reuter desarrollaron en 1977 el primer modelo de miocito ventricular, reformulado por Luo y Rudy en 1991. Características importantes del tejido cardíaco, como la bombas iónicas y la concentración de cargas, no se tuvieron en cuenta hasta el desarrollo de los modelos de Difrancesco y Noble(para las células de Purkinje), y Luo y Rudy los incorporarían en posteriores versiones revisadas de su modelo de miocito ventricular. Si bien la mayor parte de los modelos celulares cardíacos estaban dirigidos representar el comportamiento de las células ventriculares, unos pocos modelos fueron desarrollados también para las células auriculares. Las células del nodo sinoauricular fueron las primeras células auriculares en ser descritas matemáticamente. En 1998, Nygren et al. publicaron un modelo de miocito auricular, reformulado un año despues por Courtemanche et al."

El modelo utilizado en este proyecto es el de Courtemanche-Ramirez-Nattel (CRN98) [7].

5.6.3. Modelo Bidominio

El desarrollo del modelo bidominio [29] y [30] permite hacer una descripción del flujo de corriente eléctrica en el tejido cardiaco. La estructura real de los miocitos interconectados y las fibras musculares pudo abstraerse en dos dominios continuos. Un dominio intracelular que representa a los miocitos y uniones gap y uno extracelular que representa los demás componentes del tejido. Cada uno de los dominios es caracterizado por una conductividad eléctrica que es mayor a lo largo de las fibras y cambia de región a región. Ambos dominios están acoplados eléctricamente únicamente a través de la membrana celular [19].

La teoría de bidominio permitió entonces la aparición de tres consecuencias importantes, primero creo una base teórica que permitión durante décadas representar la actividad cardiaca. Por otra parte, permitió la simulación a través de la aplicación de estímulos internos y externos. Y finalmente, nuevos modelo de corazon fueron construidos [31].

5.6.4. Ecuación del Cable

Las células cardíacas pueden ser consideradas como un cilindro cuyo eje mide aproximadamente 0.1 mm y tienen un radio de 10 a 30 μm. Por otra parte, la fibra de Purkinje es una fibra delgada que se puede ver como un cilindro circular recto y uniforme. Tales estructuras pueden ser modeladas por un cable semi-infinito, con una superficie membranal que tiene propiedades resistivas, capacitivas y una conductividad axial óhmica.

Para entender este modelo se utiliza la ecuación del cable. Para tal efecto, se considera una célula como una pieza cilíndrica larga con una membrana que envuelve el citoplasma. Se supone que el potencial a lo largo de su extensión depende solamente de la variable de longitud, y no de las variables radiales o angulares. Esto permite que el cable pueda ser analizado en una sola dimensión, esta suposición se conoce como hipótesis de núcleo de conducción [19].

Para la formulación del modelo se divide el cable en un número finito de trozos cortos de longitude $\triangle x$, todos de igual potencial. En cada sección del cable se balancean todas las corrientes y sólo aparecen dos tipos de corriente, la corriente axial y la corriente a través de la membrana (corriente trasmembrana). La corriente axial tiene dos componentes, una intracelular y otra extracelular, en ambos casos se consideran óhmicas. Por la ley de Ohm, ver 5.8

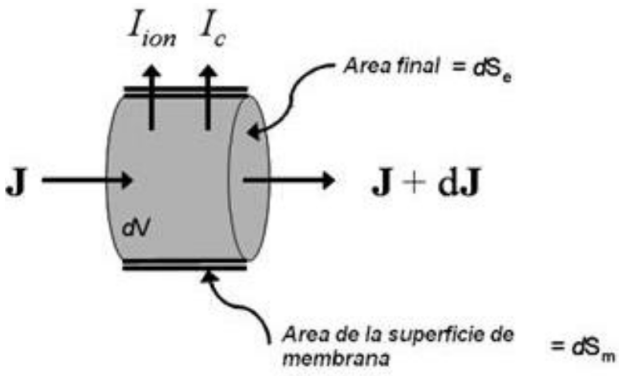

FIGURA 5.8: Corriente en una sección del cable. Fuente [19]

Una vez se define el vector de campo eléctrico como:

$$E = -\nabla\phi_i \tag{5.2}$$

En donde:

$$\nabla = \frac{\delta}{\delta x_1}\hat{i} + \frac{\delta}{\delta x_2}\hat{j} + \frac{\delta}{\delta x_3}\hat{k} \tag{5.3}$$

y ϕ_i es una función escalar para el potencial dentro del cable (o el potencial intracelular, de ahí el subíndice i). Por la ley de Ohm, el vector de flujo $J(\mu A/cm^2)$, el cual representa la densidad de corriente dentro del cable, es proporcional al vector del campo eléctrico que se ve en la figura 5.8

$$J = D \cdot E = -D \cdot \nabla\phi_i \tag{5.4}$$

Y según el procedimiento que se puede ver en [19], se obtiene la ecuación del cable 5.5:

$$\nabla \cdot J = -S_v(C_m\frac{dV_m}{dt} + I_{ion}) \tag{5.5}$$

5.6.5. Ecuaciones Bidominio

El bidominio es una estructura que define un modelo del tejido cardiaco consistiendo de dos dominios que se interpenetran representando las células cardiacas y el espacio que las rodea [19]. Los dos espacios tienen diferentes propiedades eléctricas, por tanto se forma un medio anisótropo.

Aplicando la ley de Ohm a la ecuación 5.4, a estos dos dominios resulta:

$$J_i = D_i \nabla \phi_i \tag{5.6}$$

$$J_e = D_e \nabla \phi_e \tag{5.7}$$

Donde J_i y J_e son las densidades de corriente intra y extra celular, D_i y D_e son tensores de conductividad anisotrópica, partiendo de las ecuaciones 5.6 y 5.7 y siguiendo el procedimiento en [19] se obtiene:

$$\nabla \cdot (D_i \nabla V_m) + \nabla \cdot (D_i \nabla \phi_e) = S_v (C_m \frac{dV_m}{dt} + I_{ion}) \tag{5.8}$$

$$\nabla \cdot ((D_i + D_e) \nabla \phi_e) = -\nabla \cdot (D_i \nabla V_m) \tag{5.9}$$

Las ecuaciones 5.8 y 5.9 son conocidas como las ecuaciones bidominio del tejido auricular. Las variables desconocidas V_m y ϕ_e varían espacialmente en el dominio del problema, es decir toda la aurícula humana y son ecuaciones diferenciales. Por su parte la corriente I_{ion}, es una función de V_m y representa una colección de corrientes detalladas de membrana que para este proyecto corresponden a las planteadas en el modelo de Courtemanche [7].

5.6.6. Modelo auricular de Courtemanche (CRN98)

Los mecanismos iónicos que son fundamentales en muchas propiedades importantes del potencial de acción de la aurícula humana han sido poco estudiados. Usando formulaciones específicas de las corrientes de K^+, Na^+ y Ca^{2+} sobre datos experimentales obtenidos desde miocitos humanos, en conjunto con representaciones de intercambio y otras corrientes, Marc Courtemanche, Rafael J. Ramirez y Stanley Nattel desarrollaron un modelo matemático del potencial de Acción ver figura 5.9.

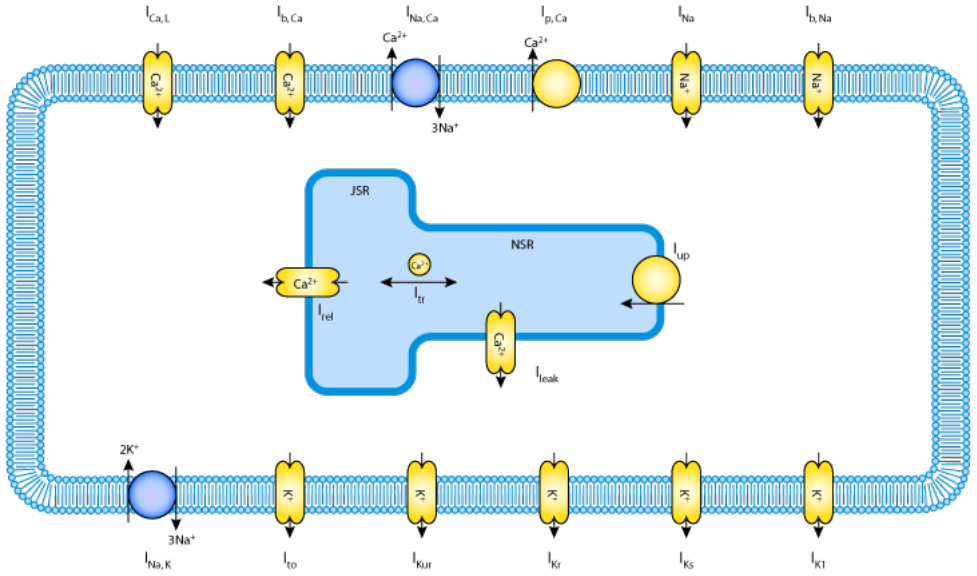

FIGURA 5.9: Representación esquemática del Modelo de Courtemanche

Este modelo matemático fue construido utilizando el formalismo de Hodgkin-Huxley [32] utilizado como base del trabajo clásico de Luo y Rudy de miocitos de cobayas [33] que a su vez estuvo sustentado en la investigación realizada por Beeler-Reuter [34]. Courtemanche efectivamente desarrolló un modelo funcional del potencial de acción cuya meta principal fue que pudiera ser utilizado para comprobar el funcionamiento de las células ante condiciones que previamente no habían sido probadas [35].

El modelo CRN98 de células auriculares humanas fue publicado en 1998 por Marc Courtemanche, Rafael J. Ramírez y Stanley Nattel, quienes tomaron como base el modelo Luo y Rudy de células ventriculares de cobaya(Cavia porcellus) de 1994[28].

La membrana celular se modela como un condensador conectado en paralelo con resistencias variables y fuentes de voltaje, ver figura 5.10).

La variación del potencial de membrana esta dado por la siguiente ecuación:[7]

$$\frac{\partial V}{\partial t} = \frac{-\left(I_{ion} + I_{st}\right)}{C_m} \tag{5.10}$$

donde I_{ion} es la corriente iónica total, I_{st} es la corriente de estímulo y C_m es la capacitancia de la membrana. Éste modelo posee 21 variables, expresiones para 12 corrientes transmembrana y manejo de calcio intracelular, se basa en estudios de corriente rápida de sodio Na^+ (I_{Na}) en humanos, la corriente transitoria de salida K^+ (I_{to}), corriente rectificadora retardada ultra rápida de potasio K^+ (I_{kr}), corriente rectificadora lenta de potasio K^+ (I_{ks}), corriente rectificadora de entrada de K^+ (I_{kl}) y para el resto de las

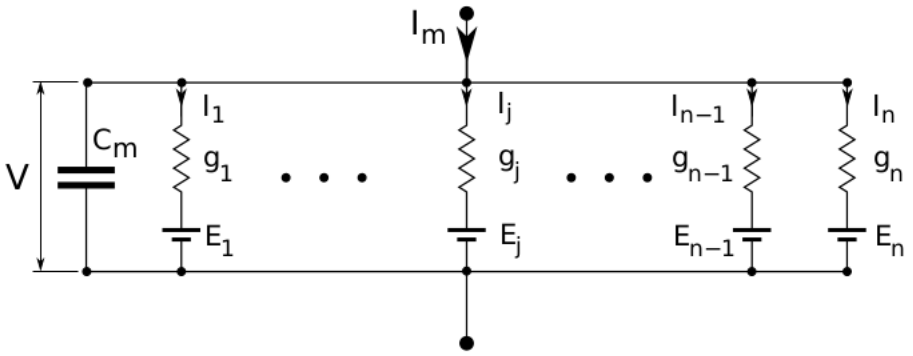

FIGURA 5.10: Circuito equivalente de la membrana celular. Imagen tomada de "Algoritmos para Ecuaciones de Reacción Difusión Aplicados a Electrofisiología" [36]

corrientes y el manejo de calcio intracelular se basa en el estudio previo de Luo y Rudy [33], en el que se describe el comportamiento de las células ventriculares de un corazón de cobaya.

La corriente iónica total (suma de todas las corrientes de Na^+ ,Ca^{2+} y K^+) está dada por:

$$\begin{aligned} I_{ion} = {} & I_{Na} + I_{K1} + I_{to} + I_{Kur} + I_{Kr} + I_{Ks} + I_{Ca,L} + I_{p,Ca} \\ & + I_{Na,K} + I_{NaCa} + I_{b,Na} + I_{b,Ca} \end{aligned} \tag{5.11}$$

La Figura 5.11 muestra la representación esquemática de las corrientes, bombas e intercambiadores incluidos en el modelo CRN98.

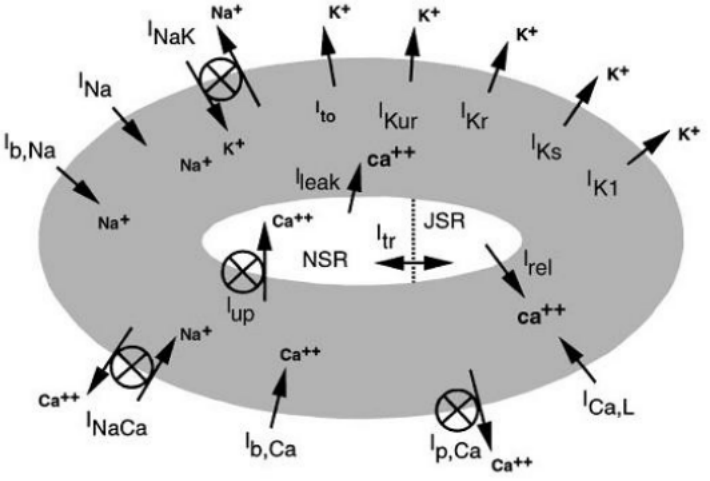

FIGURA 5.11: Representación esquemática. Imagen tomada de "Ionic Mechanisms Underlying Human Atrial Action Potential Properties: Insights from a Mathematical Model" [7]

Las corrientes se definen mediante funciones dependientes del potencial de membrana (V), del potencial de equilibrio para el ión en cuestión (E_i , i = Na^+ , Ca_{2+} , K_+), y del estado de las compuertas de activación e inactivación correspondientes a cada corriente. [19], [37], [38]

5.6.6.1. Propagación del Potencial de Acción

El tejido cardíaco está constituido por fibras musculares, formadas por células de forma aproximadamente cilíndrica de unos 100 um de longitud y de 10 a 15 um de diámetro [39] [40].

Una propiedad importante del tejido cardíaco es la propagación célula a célula del impulso eléctrico. Los medios intracelulares de células vecinas están directamente interconectados a través de *gap junctions*. Cuando se le aplica a una célula un estímulo eléctrico supraumbral, se activa la corriente entrante rápida de sodio ver figura 5.12, paso 1, aumentando su potencial intracelular. La diferencia de potencial resultante entre el medio intracelular de la célula recién excitada y su célula vecina genera una corriente eléctrica que circula a través del *gap junction* ver Figura 5.12, paso 2, elevando consecuentemente el potencial interno de la célula vecina. Cuando el potencial del medio interno supera el potencial umbral, la célula se excita ver Figura 5.12, paso 3 y repite el mismo proceso con sus células más cercanas, propagando así el potencial de acción de una célula a otra ver Figura 5.12, paso 4. [39] [40] [28]

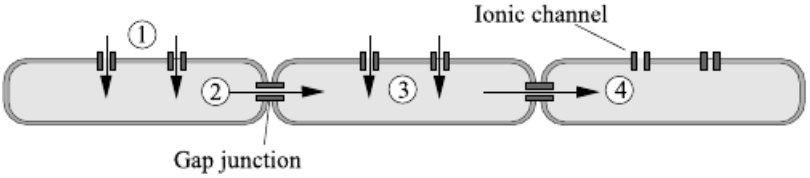

FIGURA 5.12: Propagación de un impulso. Imagen tomada de "A Biophysical Model of Atrial Fibrillation and Electrograms: Formulation, Validation and Applications" [39]

En una fibra solo existe una dirección de propagación, pero en un tejido el estímulo se puede propagar en dirección longitudinal y transversal. La velocidad longitudinal es del orden de 3 a 5 veces más rápida que la velocidad transversal, esta propiedad es conocida como anisotropía [28]. En el presente proyecto se asume que el tejido es isotrópico, por lo tanto ambas velocidades tienen igual valor.

5.6.6.2. Propagación del Potencial de Acción en 2D - Tejido

El tejido cardíaco se modela como una malla de células interconectadas por medio de gap juctions [41]. En la Figura 5.13 R_L y R_T representan los gap junctions.

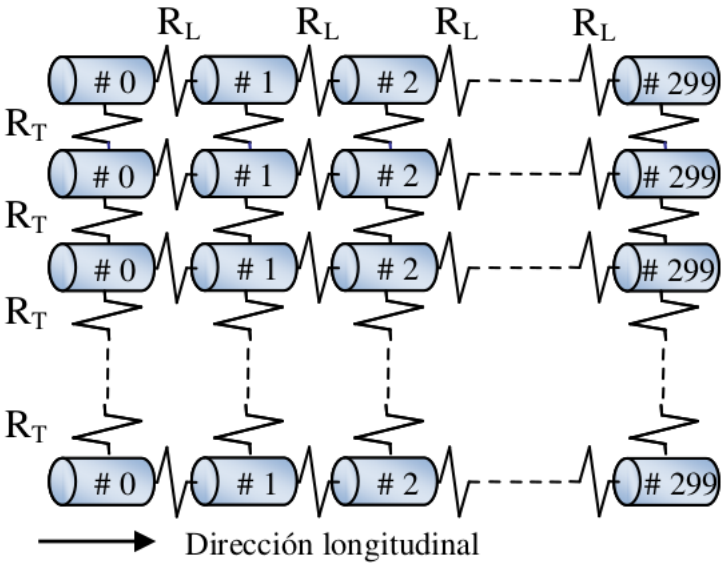

FIGURA 5.13: Malla de células. Imagen tomada de "Modelización y Evaluación de Factores que Favorecen las Arritmias Auriculares y su Tratamiento Mediante Técnicas Quirúrgicas. Estudio de Simulación" [40]

La ecuación que describe la propagación del PA en un tejido bidimensional (2D) es la siguiente [42][41]:

$$\frac{\partial V}{\partial t} = \frac{-(J_{ion})}{C} + Dx\frac{\partial^2 V}{\partial x^2} + Dy\frac{\partial^2 V}{\partial y^2} \tag{5.12}$$

$$J_{ion} = \frac{I_{ion} + I_{st}}{AreaTransversal} \tag{5.13}$$

$$D_i = \frac{1}{\rho_i SvC}, \quad i = x, y \tag{5.14}$$

Donde I_{ion} es la corriente iónica total [pA], I_{st} es la corriente de estimulación [pA], J_{ion} es la densidad de corriente [pA/cm^2], D es el coeficiente de difusión [cm^2/ms], Dx y Dy son los coeficientes de difusión [cm^2/ms] en dirección longitudinal(x) y transversal(y) , Sv es la relación superficie-volumen de un cilindro [cm^{-1}], ρ_x y ρ_y son las resistividades intracelulares [Ωcm] en dirección longitudinal y transversal, y C es la capacitancia por unidad de área [pF/cm^2].

La ecuación 5.12 considera las dos direcciones de propagación: longitudinal (x) y transversal (y). [43]

5.7. *Paraview*

Paraview es una aplicación de código abierto tipo *GNU public* mantenida por Kitware, esta aplicación permite la construcción de visualizaciones científicas de gran calidad utilizando un enfoque interactivo o también a través de tareas batch, está basada en la librería de visualización VTK (Visualization ToolKit) [44]. Paraview fue construída para la visualización de conjuntos de datos extremadamente grandes, de hecho su arquitectura de diseño fue pensada para funcionar en entornos de clusters computacionales [8]. En la Figura 5.14 se puede ver un ejemplo de visualización científica usando Paraview. Y en la figura 5.15 se puede ver la arquitectura conceptual que usa Paraview para realizar el proceso de renderizado de escenas.

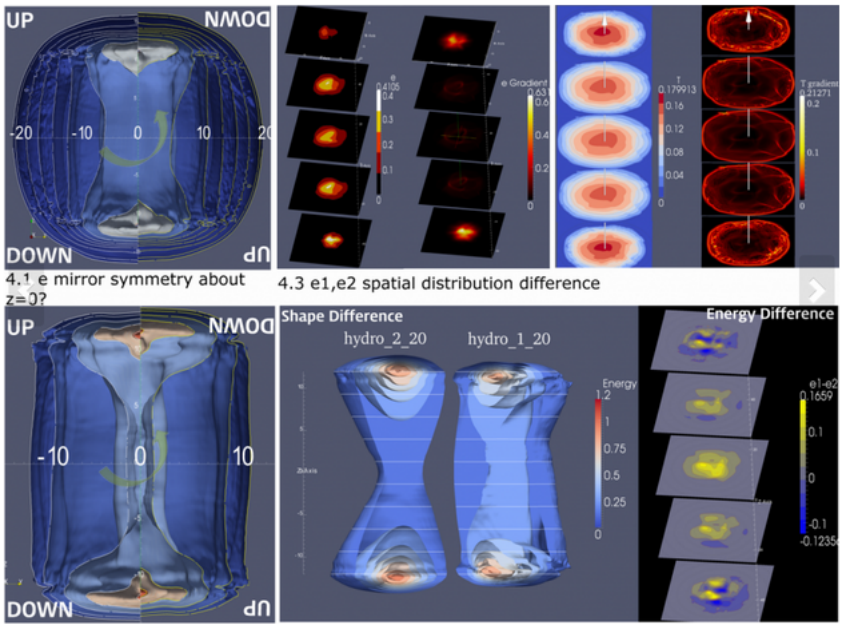

FIGURA 5.14: Ejemplo de visualización científica usando Paraview

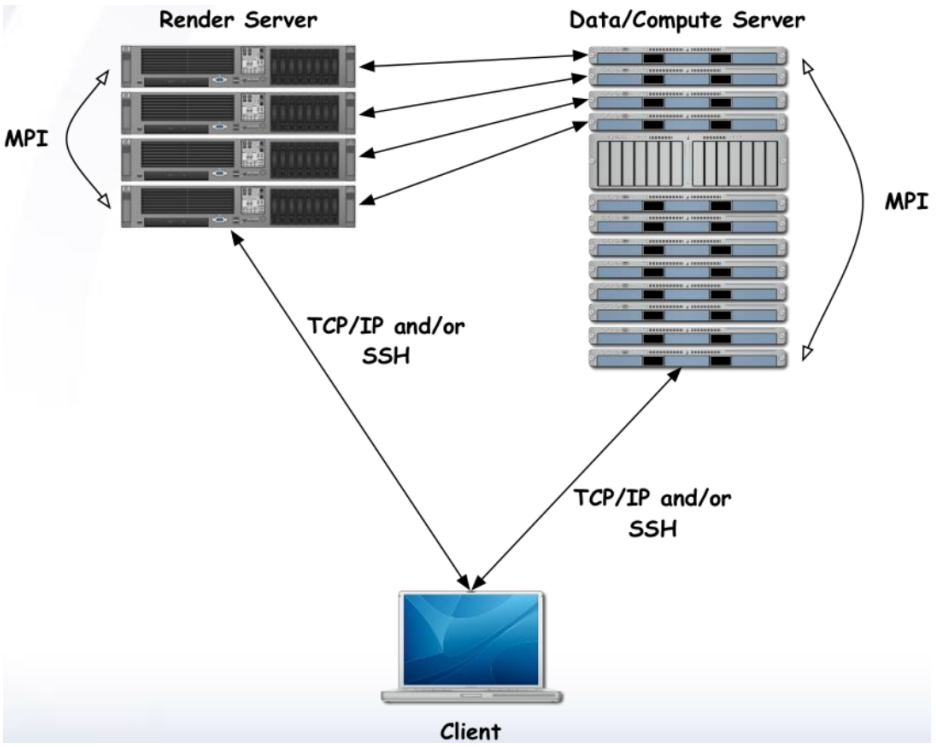

FIGURA 5.15: Arquitectura Conceptual Paraview

5.7.1. *Paraview Catalyst*

Debido a las capacidades de cómputo actuales, al crecimiento de los supercomputadores y a la utilización de arquitecturas masivamente paralelas, como las GPU, *Kitware* a través de *Paraview* ha creado un módulo, llamado *Catalyst*, que realiza la visualización de datos in-situ, lo que permite que los usuarios e investigadores vayan visualizando los resultados de su simulación en tiempo de ejecución. Esta herramienta permite que las visualizaciones sean realizadas una vez los datos están listos para ser presentados a los científicos [21]. El modelo de trabajo de *Catalyst* puede verse en la Figura 5.16.

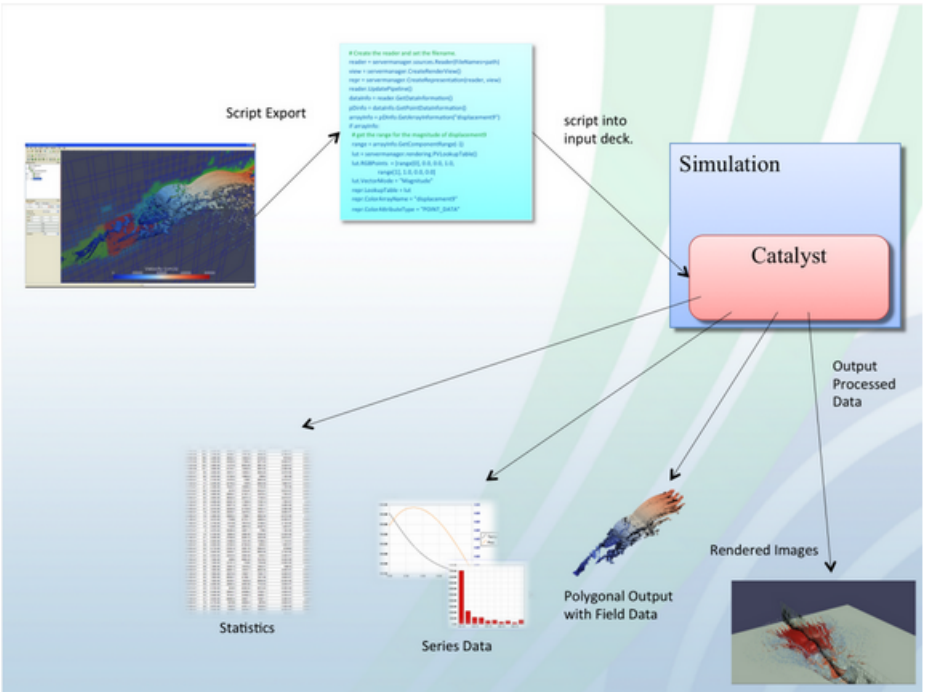

FIGURA 5.16: Visualización de datos usando *Paraview* desde el punto de vista del Usuario

Capítulo 6

Estado del Arte

Desde el 2007 se han realizado diversas investigaciones en el campo de simulación cardiaca, apareciendo un creciente interés en su implementación utilizando sistemas heterogéneos. En la actualidad y debido al gran impacto generado por la computación científica usando GPUs, se ha iniciado un campo de gran interés en cuanto a la aceleración de algoritmos que permitan comprender el comportamiento del corazón.

6.1. Simulación de Modelos Cardíacos sobre Sistemas de Cómputo Heterogéneos

La complejidad de los sistemas de ecuaciones diferenciales que envuelven la solución del modelo eléctrico cardiaco, ha ocasionado la aparición de diversos esfuerzos para disminuir la carga computacional que este tipo de problemas contiene, en el año 2008 Orovio [45] plantea un modelo de ventrículo conocido como *Minimal Ventricular* a través del cuál se propone modelar el potencial de acción del ventrículo utilizando un total de 12 ecuaciones diferenciales, en la misma línea existen diversos métodos numéricos que intentan disminuir la complejidad computacional para acelerar el proceso de simulación desde el punto de vista de la complejidad computacional, ya en 1978 Rush y Larsen lanzan un método numérico para atacar este problema como puede verse en [46] y en el año 2009 Sundnes et al. construyen una extensión de segundo orden del método de Rush y Larsen para resolver las ecuaciones dinámicas de membrana como se ve en [47] y también como lo mencionan Perego y Veneziani en [48], estos métodos generalmente funcionan siempre y cuando se cumplan un conjunto de condiciones en cuanto a la actualización de las variables, y a la modificación dinámica del paso de tiempo y del mallado espacial realizado.

En el año 2012 se presenta el que sería el primer trabajo que simula ondas espirales utilizando un modelo cardiaco basado en microestructura del tejido y un modelo complejo de miocito, todo lo anterior ejecutándose sobre unidades de procesamiento gráfico (*GPU*) según [49], éste trabajo es ejecutado sobre un Clúster de 8 nodos cada uno con dos GPUs y simulando $1cm^2$ de tejido cardíaco por un período de 10ms, el tiempo tomado para la simulación fue de 1302 segundos. Durante el mismo año Nimmagadda et al. en su artículo [50] establece una simulación de una malla de $256X256X256$ células por 350ms utilizando el modelo de Ten Tusscher [51] y sobre un cluster con un total de 4 GPU en un tiempo de cómputo de 664 segundos.

Durante el 2013 Marcotte et al. [52] plantean una implementación del modelo de Orovio [45], en el cual establecen la simulación de ventrículo en 2D utilizando OpenCL. Ya en el 2014 Garcia-Molla et al. [53] construyen una implementación del modelo de Courtemanche con un total de 163000 células utilizando un tamaño de paso temporal adaptativo para el cual una simulación de 300ms se toma alrededor de 53.6 segundos en completarse.

En el 2015 Xia et al. [54] plantean la simulación en 3D de un modelo de aurícula de Oveja utilizando sistemas con GPU Tesla K40, en este caso una aceleración de 200X fue reportada. En este mismo año Chunxi [55] implementa un modelo celular de aurícula con un total de 500 células sobre una GPU GTX550i de *nVidia* y reporta un tiempo de ejecución de 20 segundos.

Es clara la tendencia a la utilización de sistemas de cómputo heterogéneos, esto con el fin de disminuir los tiempos de ejecución, y que las simulaciones empiecen a ser útiles en el campo clínico.

En Colombia, se han realizado esfuerzos en cuanto al modelado cardiaco, sin embargo no se han encontrado reportes sobre la utilización de Unidades de Procesamiento Gráfico para acelerar los tiempos de ejecución de este tipo de simulaciones, razón por la cuál la investigación aquí expuesta representa un nuevo campo de estudio nacional que permite la articulación de grupos de investigación como Sirius (Universidad Tecnológica de Pereira) y GITIR (Universidad de Caldas), los cuales aúnan esfuerzos en pro de la consolidación de redes de investigación que articulen el conocimiento en computación de alto desempeño y modelado cardíaco.

6.2. Visualización *in-situ*

Las tendencias actuales de procesamiento sobre GPUs, al igual que la necesidad de tener adecuadas visualizaciones de los procesos de simulación han permitido que se inicien

investigaciones en el área de visualización *in-situ*.

En 2009 Reumann et al. [56] plantean la realización de un proceso de visualización en sistemas de supercomputación, muestran la posibilidad de aprovechar todos los nodos en el sistema para ayudarse en el proceso de generación de imágenes a partir de un modelo (*renderizado*), crean una herramienta conocida como *SPVN (Scalable Parallel Visualization Networking)*.

En el año 2010 Kanthasamy en su tesis de maestría [57] muestra un enfoque de visualización paralela offline de un conjunto de datos resultado de un proceso de simulación cardíaca. En 2011 Rivi et al. [58] en su artículo de revisión muestra dos herramientas para realizar visualización *in-situ*, una a través de un plugin utilizando Paraview en el campo de astrofísica y otra utilizando la herramienta Visit para una simulación cerebral.

Moreland et al. en 2015 y 2016 [59] [60] empiezan una nueva tendencia en los procesos de visualización, en este caso la idea es aprovechar que los procesos de simulación son realizados en sistemas HPC totalmente Heterogéneos y en gran medida utilizando co-procesadores; de éste modo se muestra un enfoque a través del cual se intenta realizar la visualización aprovechando las GPUs sobre las cuales en algunos casos también es realizado el procesamiento de la simulación.

6.3. Alya Red

Desde el año 2012 el Centro de Supercomputación de Barcelona ha estado trabajando de manera conjunta con diversas instituciones a nivel mundial para la construcción de un modelo computacional cardiaco desde el punto de vista físico, mecánico y biológico.

Su objetivo es desarrollar un modelo computacional para simular el funcionamiento del corazón humano y está siendo desarrollado por un equipo multidisciplinar que implica a médicos, bioingenieros e investigadores en supercomputación y en imagen médica. Actualmente, gracias a Alya Red, los científicos que forman parte del proyecto pueden simular modelos ventriculares procedentes de geometrías reales.

Otro objetivo es crear una nueva herramienta para ayudar a comprender mejor el funcionamiento del sistema cardiovascular a médicos de investigación clínica y farmacéutica. Esta herramienta será una infraestructura tecnológica de simulación vinculada a la computación de altas prestaciones (HPC, por las siglas en inglés de High Performance Computing) [17].

Capítulo 7

Implementación Modelo Eléctrico de Courtemanche

A las ecuaciones 5.13 en 1D y 5.12 en 2D, se les aplica el método de diferencias finitas para obtener un sistema de ecuaciones de la forma AX = B, posteriormente este sistema de ecuaciones es resuelto con Armadillo en el algoritmo secuencial y ArrayFire [20] en la versión paralela. El coeficiente de difusión D es necesario para construir la matriz A y el vector B.

El código fuente de la implementación se encuentra disponible en github[1].

7.1. D - Coeficiente de difusión

$$D = \frac{1}{\rho S v C} \tag{7.1}$$

Donde ρ es la resistividad intracelular $[\Omega cm]$, Sv es la relación superficie volumen $[1/cm]$ y C es la capacitancia por unidad de area $[pF/cm^2]$.

Los valores que fueron usados para calcular D están basados en [7] y [40] según muestra la tabla 7.1.

[1] Enlace del repositorio con el código fuente *https://github.com/kala855/paravis*

r	radio de célula auricular	0.0008 cm
h	longitud de célula auricular	0.01 cm
ρ	resistividad intracelular	200 Ωcm
Cm	Capacitancia	100 uF

TABLA 7.1: Valores necesarios para calcular D

Para calcular **la relación superficie-volumen** se considera que una célula tiene forma cilíndrica, por lo tanto su área y volumen son los siguientes.

$$Area = 2\pi rh, \quad Volumen = \pi r^2 h \tag{7.2}$$

De 7.2 se obtiene:

$$Sv = \frac{Area}{Volumen} = \frac{2\pi rh}{\pi r^2 h} = \frac{2}{r} \tag{7.3}$$

El área transversal de la célula es :

$$areaTransversal = \pi r^2 = \pi * (0{,}0008cm)^2 = 0{,}000002cm^2 \tag{7.4}$$

La **capacitancia por unidad de área** es la siguiente:

$$C = Cm/areaTransversal = 100pF/0{,}000002cm^2 = 49735803{,}412pF/cm^2 \tag{7.5}$$

Con $C = 49735803{,}412pF/cm^2$ y $Sv = \frac{2}{0{,}0008cm}$, la ecuación 7.1 tiene los siguientes valores:

$$D = \frac{1}{\rho SvC} = \frac{1}{\rho \frac{2}{r} C} = \frac{r}{2\rho C} = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412pF/cm^2)} \tag{7.6}$$

$$D = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412x10^{-12}F)} = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412x10^{-12}F/cm^2} \tag{7.7}$$

Las unidades del coeficiente de difusión D son $[cm^2/ms]$, para llegar a éstas es necesario convertir Faradios(F) a segundos(s)/ohms(Ω). Para lograr esto se parte de la siguiente ecuación de corriente en un condensador:

$$I = C\frac{\partial V}{\partial t} \tag{7.8}$$

Donde de I es corriente en amperios (A), C es capacitancia en faradios (F), V es el voltaje en voltios (V) y t tiempo en segundos (s).

$$[A] = [F]\frac{[V]}{[s]} \tag{7.9}$$

De la ecuación 7.9

$$[s] = [F]\frac{[V]}{[A]} \tag{7.10}$$

Por ley de Ohm se tiene que $R = \frac{V}{I}, [\Omega] = \frac{[V]}{[A]}$

$$[s] = [F][\Omega] \tag{7.11}$$

Por lo tanto un faradio se puede expresar como:

$$[F] = \frac{[s]}{[\Omega]} \tag{7.12}$$

Teniendo en cuenta la ecuación 7.12, la ecuación 7.7 se puede expresar como:

$$D = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412x10^{-12}F)} = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412x10^{-12}\frac{s}{\Omega})} \tag{7.13}$$

$$D = \frac{0{,}0008cm}{(2)(200\Omega cm)(49735803{,}412x10^{-9}\frac{ms}{\Omega})} = 0{,}00004\frac{cm^2}{ms} \tag{7.14}$$

7.2. Representación matemática de la propagación del PA en un tejido

La ecuación diferencial que modela la propagación del potencial de acción en un tejido es la siguiente:

$$\frac{\partial V}{\partial t} - D_x\frac{\partial^2 V}{\partial x^2} - D_y\frac{\partial^2 V}{\partial y^2} = -\frac{Jion}{C} \tag{7.15}$$

Con las siguientes condiciones [2]:

$$V_{x,y}^{0} = -81{,}2 \qquad \text{Condición inicial}$$
$$\frac{\partial V}{\partial x} = 0 \qquad \text{Condición de borde Neumann}$$

El método de solución a esta ecuación de derivadas parciales mediante diferencias finitas es muy similar al abordado en una dimensión en [61], de donde también se toma el proceso de representación matemática en un tejido, sólo se diferencia en algunos aspectos relacionados a los bordes y a la representación de la discretización del dominio espacial.

7.2.1. Selección de la malla del dominio

En el dominio del tiempo se selecciona una malla igual a como se realizó en el caso de una fibra, en cambio para la malla del dominio espacial, se discretiza en intervalos iguales tanto en x como en y.

En la figura 7.1 se resalta la importancia de los puntos azules ya que estos son los bordes que delimitan el problema, adicionalmente se toman como parámetros de simulación $\triangle x = 0{,}02cm$, $\triangle y = 0{,}02cm$ y un tamaño de malla por defecto de $165 * 165$ células, los parámetros definidos para la simulación fueron tomados de investigaciones realizada en [19], [52], [50].

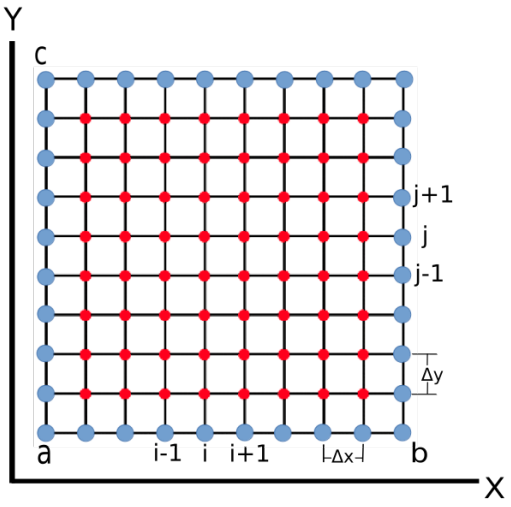

FIGURA 7.1: Discretización espacial del tejido. Fuente [61]

[2] Se usará la siguiente notación $V_{x,y}^{t}$ para indicar en la parte inferior el espacio y en la parte superior el tiempo.

7.2.2. Sustitución de las derivadas parciales

Luego de tener definida la malla de los dominios, se sustituyen las derivadas parciales, al igual que en una fibra se sustituye la derivada parcial de la función respecto al tiempo con diferencias progresivas y las derivadas parciales respecto al espacio se aproximan siguiendo el esquema algorítmico de Crank-Nicholson, quedando la ecuación de la siguiente manera:

$$\frac{V_{i,j}^{n+1}-V_{i,j}^{n}}{\triangle t}-D_x\left(\frac{V_{i+1,j}^{n}-2V_{i,j}^{n}+V_{i-1,j}^{n}}{2\triangle x^2}+\frac{V_{i+1,j}^{n+1}-2V_{i,j}^{n+1}+V_{i-1,j}^{n+1}}{2\triangle x^2}\right)- D_y\left(\frac{V_{i,j-1}^{n}-2V_{i,j}^{n}+V_{i,j+1}^{n}}{2\triangle y^2}+\frac{V_{i,j-1}^{n+1}-2V_{i,j}^{n+1}+V_{i,j+1}^{n+1}}{2\triangle y^2}\right)=-\frac{Jion}{C}$$

Factorizando, se obtiene:

$$\frac{V_{i,j}^{n+1}-V_{i,j}^{n}}{\triangle t}-\frac{D_x}{2\triangle x^2}\left(V_{i+1,j}^{n}-2V_{i,j}^{n}+V_{i-1,j}^{n}+V_{i+1,j}^{n+1}-2V_{i,j}^{n+1}+V_{i-1,j}^{n+1}\right)- \frac{D_y}{2\triangle y^2}\left(V_{i,j-1}^{n}-2V_{i,j}^{n}+V_{i,j+1}^{n}+V_{i,j-1}^{n+1}-2V_{i,j}^{n+1}+V_{i,j+1}^{n+1}\right)=-\frac{Jion}{C}$$

Se multiplica cada miembro de la ecuación por $\triangle t$ y se realiza el siguiente cambio de variable $S_x=\frac{\triangle tDx}{2\triangle x^2}$ y $S_y=\frac{\triangle tD_y}{2\triangle x^2}$

$$V_{i,j}^{n+1}-V_{i,j}^{n}-Sx\left(V_{i+1,j}^{n}-2V_{i,j}^{n}+V_{i-1,j}^{n}+V_{i+1,j}^{n+1}-2V_{i,j}^{n+1}+V_{i-1,j}^{n+1}\right)- Sy\left(V_{i,j-1}^{n}-2V_{i,j}^{n}+V_{i,j+1}^{n}+V_{i,j-1}^{n+1}-2V_{i,j}^{n+1}+V_{i,j+1}^{n+1}\right)=-\frac{Jion}{C}\triangle t$$

Se expande la ecuación, se agrupan los términos semejantes y se dejan al lado derecho todos los términos conocidos y al lado izquierdo todos los términos desconocidos:

$$\begin{aligned}&-S_yV_{i,j-1}^{n+1}-S_xV_{i-1,j}^{n+1}+(1+2S_x+2S_y)V_{i,j}^{n+1}-S_xV_{i+1,j}^{n+1}-S_yV_{i,j+1}^{n+1}=\\&S_xV_{i-1,j}^{n}+(1-2S_x-2Sy)V_{i,j}^{n}+S_xV_{i+1,j}^{n}+S_yV_{i,j-1}^{n}+S_yV_{i,j+1}^{n}-\frac{Jion}{C}\triangle t\end{aligned}\tag{7.16}$$

Esta ecuación se escribe mejor en forma matricial creando una matriz A con los coeficientes de los términos del lado izquierdo, un vector columna con los términos desconocidos

X y un vector columna con los términos conocidos que representan el lado derecho B y así obtener un sistema de la forma $AX = B$ pero antes es necesario representar la matriz de puntos en x y en y como un vector esto se logra aplanando la matriz estableciendo las filas de forma consecutiva como puede verse en la figura 7.2.

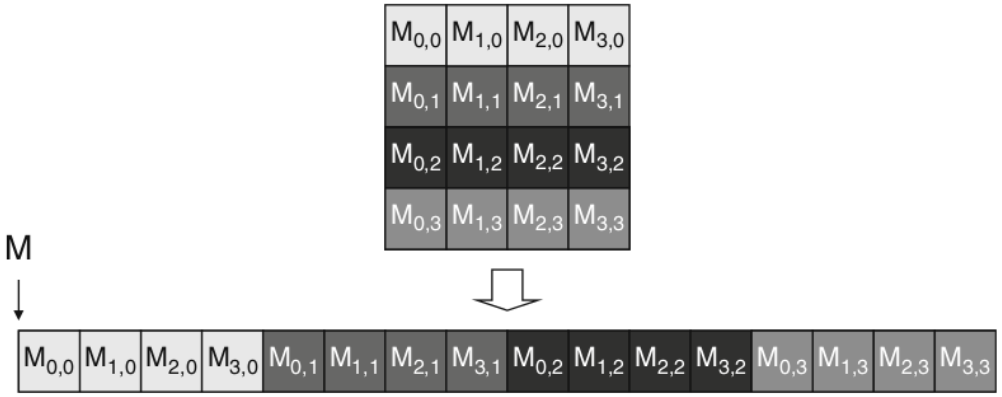

FIGURA 7.2: Representacion de una Matriz en 2D como un Vector. Imagen Tomada del Libro "Programming Massively Parallel Processors"[62]

La construcción de estas matrices son algo mas complicadas que en 1D debido al tratamiento de los bordes y a la representación de la matriz espacial como un vector.

Para la Matriz A este sería el esquema:

$$A= \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & & & & & & & & & & & \\ 0 & 0 & -Sy & \cdots & -Sx & (1+2Sx+2Sy) & -Sx & \cdots & -Sy & 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & -Sy & \cdots & -Sx & (1+2Sx+2Sy) & -Sx & \cdots & -Sy & 0 & & 0 & 0 \\ \vdots & \vdots & & & & & & & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Nótese que en este caso se tiene una matriz con cinco diagonales a diferencia de la matriz A en una dimensión que tiene tres, otro aspecto característico es que sus diagonales no son completamente consecutivas.

El vector solución X está dado por la forma:

$$X = \begin{bmatrix} V_{1,1}^{n+1} \\ V_{1,2}^{n+1} \\ \vdots \\ V_{i,j}^{n+1} \\ V_{i,j+1}^{n+1} \\ \vdots \\ V_{c,b-1}^{n+1} \\ V_{c,b}^{n+1} \end{bmatrix}$$

El vector B está constituido por los valores del lado derecho de la ecuación los cuales son valores conocidos, su esquema puede verse en la siguiente ecuación.

$$B = \begin{bmatrix} 2rV_0^n + (1-2r)V_1^n + rV_2^n - f(x,t)\triangle t \\ rV_1^n + (1-2r)V_2^n + rV_3^n - f(x,t)\triangle t \\ rV_2^n + (1-2r)V_3^n + rV_4^n - f(x,t)\triangle t \\ \vdots \\ rV_{N-1}^n + (1-2r)V_N^n + 2rV_{N+1}^n - f(x,t)\triangle t \end{bmatrix}$$

Capítulo 8

Implementación del Modelo Usando CUDA

Para la implementación de Modelo computacional se construyeron dos algoritmos, el primero fue desarrollado utilizando la librería Armadillo de C++ [63], todo el código es accesible a través de github[1], cabe aclarar que la implementación en CPU aprovecha todos los núcleos presentes en el procesador.

Por otro lado se realizó la implementación del Modelo de Courtemanche [7] sobre GPU, en este caso se utilizó C++, CUDA [6] y Arrayfire [20]. Se aprovecha la capacidad de cómputo de la GPU para acelerar la versión construída sobre CPU.

8.1. Funciones Kernel Construidas en CUDA

La implementación realizada en CUDA fue construida a partir de la versión secuencial, normalmente este tipo de algoritmos de solución de sistemas de ecuaciones diferenciales tienen un ciclo sobre el cual se evoluciona el modelo temporalmente, ese ciclo no fue paralelizado, sin embargo la solución espacial del problema si es altamente paralelizable, en este caso se construyeron las siguientes funciones kernel:

- Se implementó una función kernel en CUDA que aprovecha la totalidad de procesadores presentes en la GPU. Esta función llamada *d_update_B* permite realizar el cálculo del vector resultado B en cada paso de tiempo.

[1] Enlace del repositorio con el código fuente *https://github.com/kala855/paravis*

- Una función *d_copy_voltage* fue implementada para actualizar los valores de voltaje sobre cada una de las células presentes en la malla de simulación. Esta actualización es realizada de manera paralela.
- Funciones de inicialización de datos *init_d_prevv* y *init_d_B* fueron construidas para asignar valores iniciales a los voltajes y a las soluciones de cada una de las células simuladas.

Se aprovechó la utilización de clases sobre la GPU, concepto que viene utilizándose actualmente en arquitecturas masivamente paralelas de este tipo y que permite extender el poder de lenguajes como C++ a entornos de super-computación. De esta forma se tienen instanciados objetos de tipo célula sobre la GPU y allí se construyeron también todos los métodos que permiten la actualización de las diferentes corrientes presentes en el modelo de Courtemanche [7]. Para lograr esto se construyó un conjunto de funciones kernel en un archivo de cabecera diferente con su respectivo codigo fuente *cell.cu y cell.cuh*, esta implementación es una propuesta propia del proyecto desarrollado.

Todas las funciones pueden ser consultadas en *repositorio*[2]

8.2. Solución del sistema de Ecuaciones AX=B

Como bien es conocido, la solución de un sistema de ecuaciones de la forma $AX = B$ ha sido un problema altamente estudiado, diferentes métodos han sido creados para obtener la solución del vector X, en este desarrollo este vector es el potencial de acción de cada una de las células presentes en la malla de simulación.

Dado que el tamaño de la matriz A es gigante, para una simulación de 165X165 células ésta matriz es de $27225 * 27225$, se encuentra que gran parte del tiempo de procesamiento es consumido por la solución de este sistema.

Se plantea entonces que este proceso sea solucionado por algún solver existente que permita facilitar el cálculo y que además aproveche la capacidad de cómputo presente en las GPU, en el presente proyecto fue utilizado ArrayFire [20].

Adicionalmente el algoritmo realiza llamados a la solución del sistema de ecuaciones en cada paso de tiempo, esto hizo que se realizara una factorización LU sobre la matriz A, cuyos valores no cambian durante todo el proceso de simulación, y con esta factorización se trabajará dentro de la evolución temporal de la solución. De esta forma se garantiza

[2]Repositorio *https://github.com/kala855/paravis/blob/master/src/atriumModel/visual/simulation2D-CN.cu*

una disminución de la complejidad computacional en el solver de ArrayFire ya que la Factorización LU sólo es realizada una vez y posteriormente se sigue trabajando con esta descomposición en el resto de ejecución del algoritmo, ver figura 8.1.

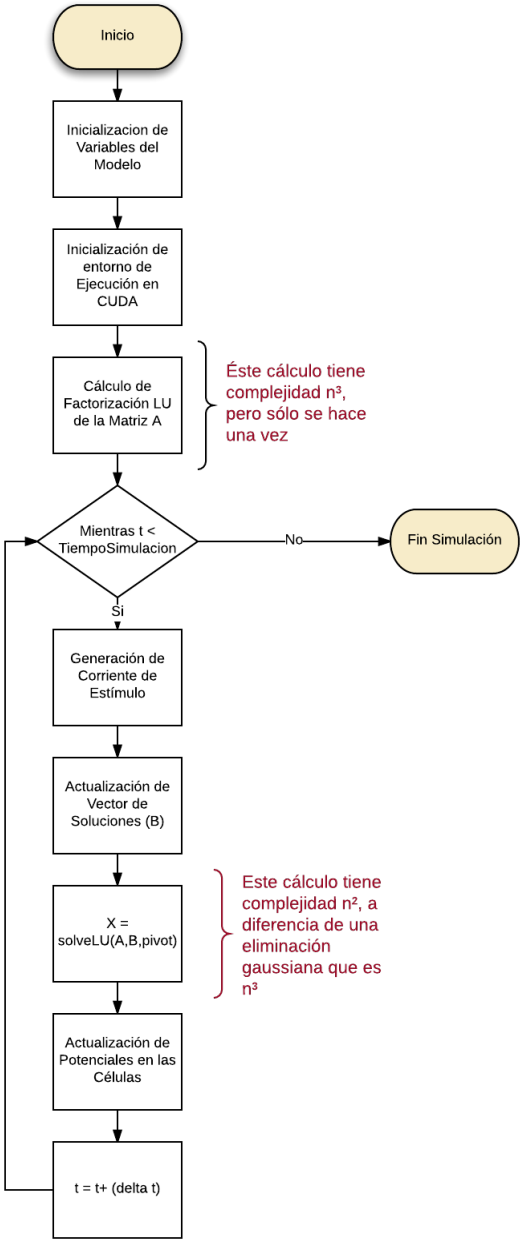

FIGURA 8.1: Diagrama de Flujo Implementación.

Capítulo 9

Arquitectura de visualización

El proceso de visualización de los resultados fue construido a través de la herramienta Paraview [8], [21]. La idea que se llevó a cabo permite que mientras el algoritmo se encuentra en ejecución en un sistema de cómputo, en otro pueda realizarse el proceso de renderizado lo que garantiza un tiempo de simulación menor y que las gráficas que se visualicen sean de mayor calidad.

A continuación puede verse el diseño conceptual de la arquitectura utilizada para realizar el proceso de visualización en tiempo de ejecución:

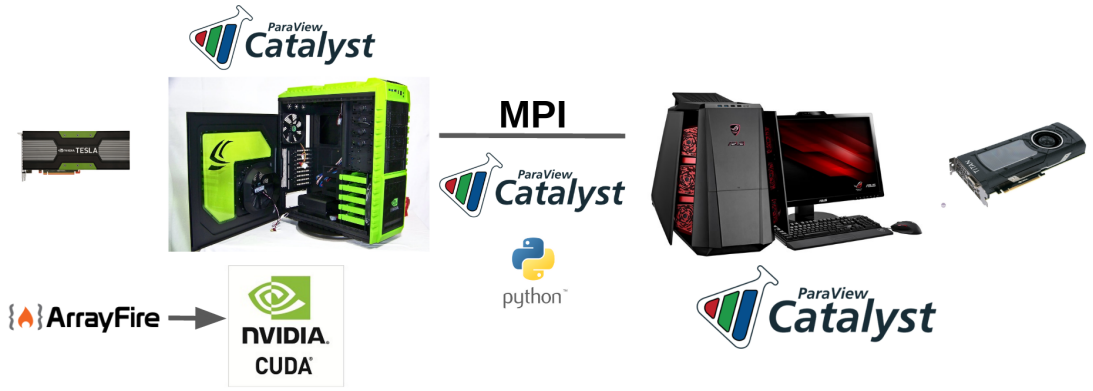

FIGURA 9.1: Arquitectura de Visualización. Fuente el autor

La figura 9.1 muestra los dos nodos de cómputo, uno de los cuales posee una tarjeta *Tesla K40C* que es utilizada para el proceso netamente de cálculo, y otro nodo que posee una tarjeta *Titan X* el cual es utilizado en el proceso de renderizado y visualización de los datos, ambos nodos deben tener instalado *Paraview-Catalyst*, el cual permite realizar el proceso de visualización in-situ. Llama la atención la utilización de *Python*, esto es porque *Paraview-Catalyst* permite invocar un script de comunicación y creación de objetos VTK (*Visualization ToolKit*) que posteriormente serán visualizados sobre el nodo elegido para tal fin [44].

La arquitectura de conexión es cliente-servidor, en este caso el nodo servidor es donde se realiza el proceso de cálculo y el nodo cliente es el equipo donde se realiza el proceso de visualización. El esquema de conexión es sencillo, lo único que debe hacerse en el cliente es ejecutar *Paraview* con soporte de *Catalyst* e indicarle que se va a conectar a una fuente de datos que sería el nodo de cálculo, por su parte el script en Python define el cliente o conjunto de clientes que realizarán el proceso de renderizado a través de la herramienta de Paraview, el código puede verificarse en el *repositorio*[1] del proyecto .

[1]Repositorio *https://github.com/kala855/paravis/blob/master/src/atriumModel/pythonScripts/nuevoTest.py*

Capítulo 10

Resultados Obtenidos

A continuación se presentarán un conjunto de pruebas del modelo, la idea es ver los tiempos de ejecución obtenidos, la aceleración conseguida y finalmente el comportamiento del modelo implementado con diferentes tipos de estímulos.

Como primera parte, el potencial de acción obtenido con el modelo de Courtemanche implementado, para una célula individual y para una célula extraída de la malla, se puede ver en las figuras 10.1 y 10.2, ambas implementaciones tienen un paso de tiempo de 0.025ms, y la graficación de los datos se realiza cada 10 pasos de tiempo:

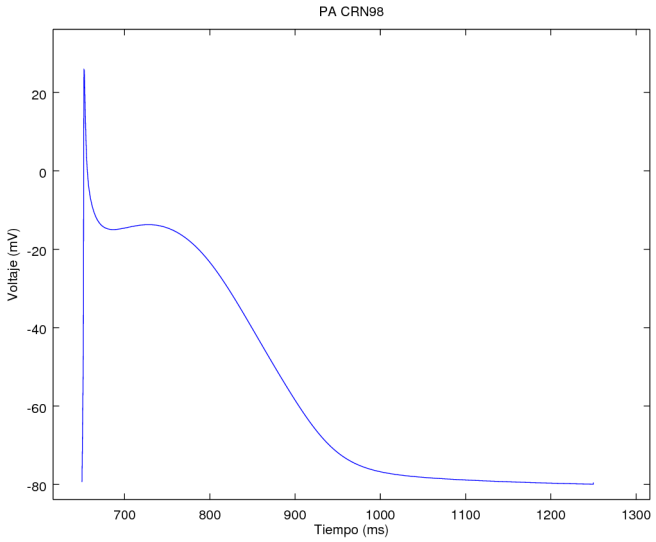

FIGURA 10.1: Potencial de Acción de una célula de la Malla con un BCL=1200ms y una duración del estímulo de 2ms

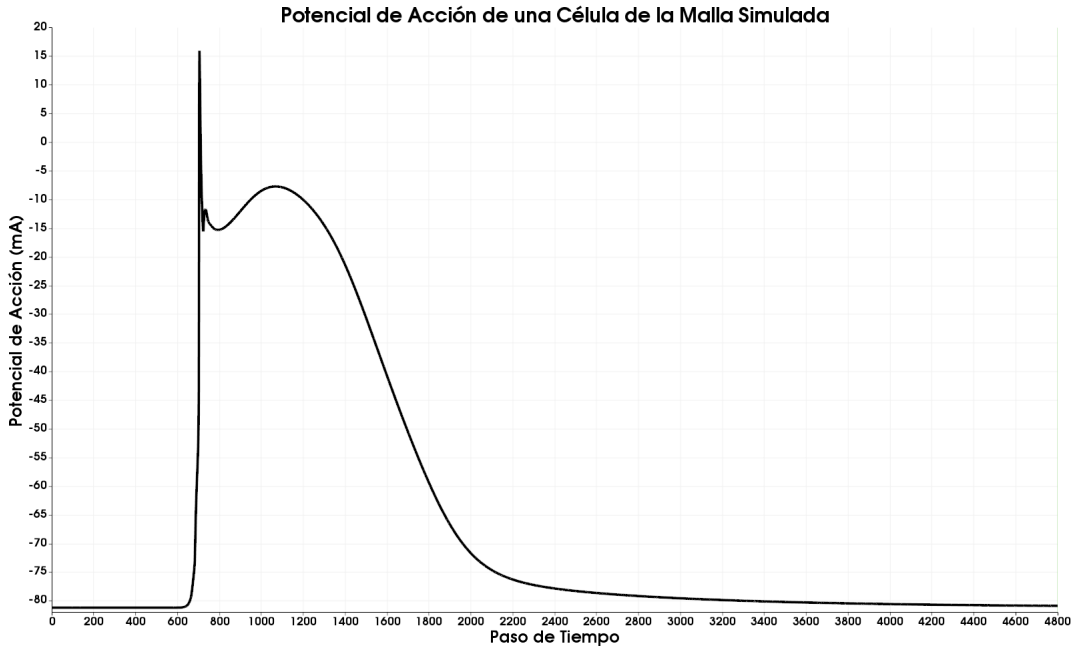

FIGURA 10.2: Potencial de Acción de una célula de la Malla con un BCL=1200ms y una duración de estímulo de 2ms

Como puede verse en la figura 10.1 y 10.2, el modelo para una célula es fiel a lo expuesto por Courtemanche-Ramirez-Nattel en [7], tanto para el modelo individual como para el modelo a nivel de la malla, de igual manera al visualizar un conjunto de 3 células diferentes se encuentra la figura 10.3 en este caso para el modelo en una sola dimensión, adicionalmente si se compara con la figura 10.4 que simula 5 células de la malla se puede ver que el modelo continúa manteniendo la forma de Potencial y los valores de voltaje en fase de despolarización definidos por Courtemanche.

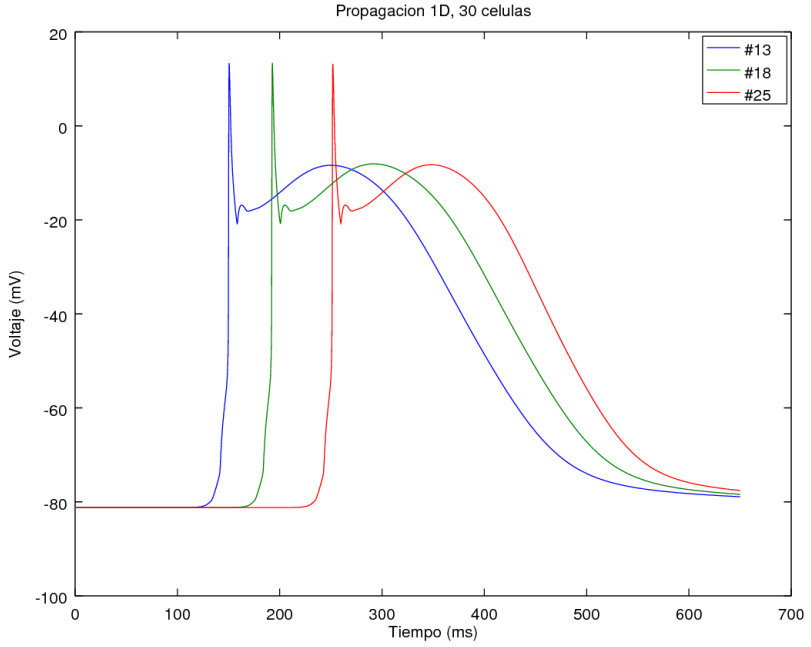

FIGURA 10.3: Potencial de Acción de 3 células de una Fibra con un BCL=1200ms y una duración del estímulo de 2ms

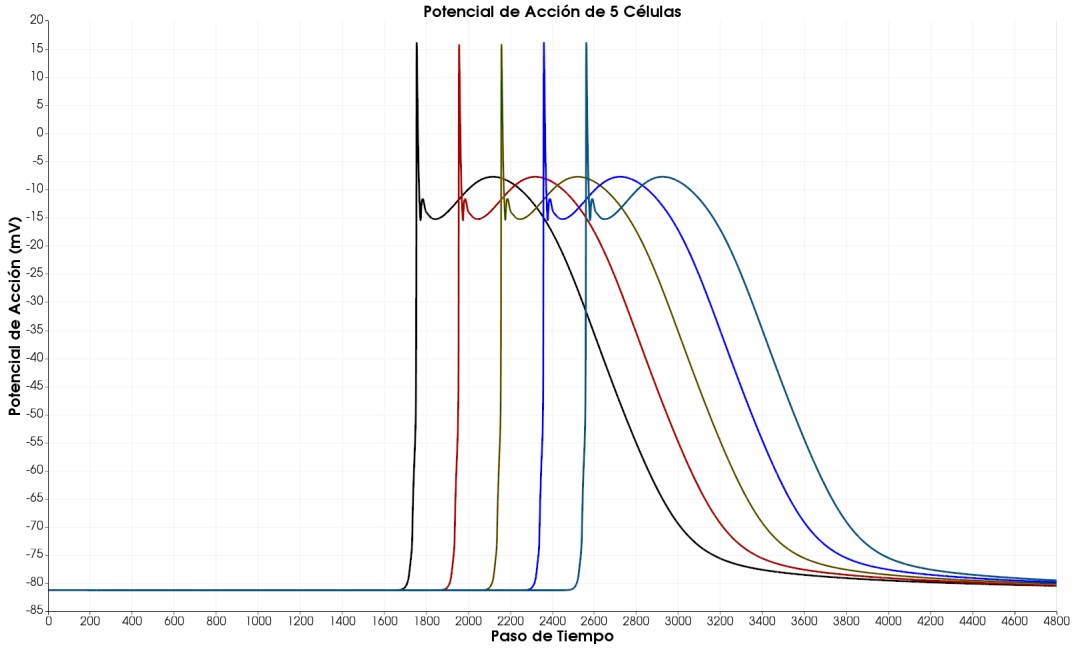

FIGURA 10.4: Potencial de Acción de 5 células de la Malla con un BCL=1200ms y una duración del estímulo de 2ms

A continuación son presentados los tiempos de ejecución del algoritmo paralelizado en CPU utilizando 8 hilos de procesador ver figura 10.5 y el tiempo de ejecución del algoritmo implementado sobre la GPU ver figura 10.6.

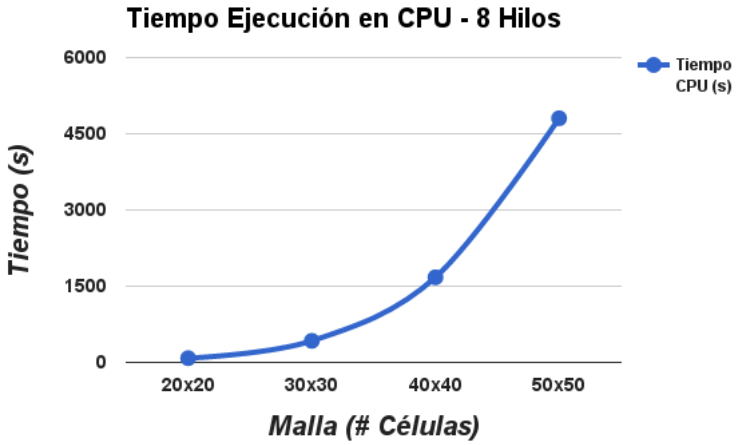

FIGURA 10.5: Tiempo de Ejecución algoritmo CPU

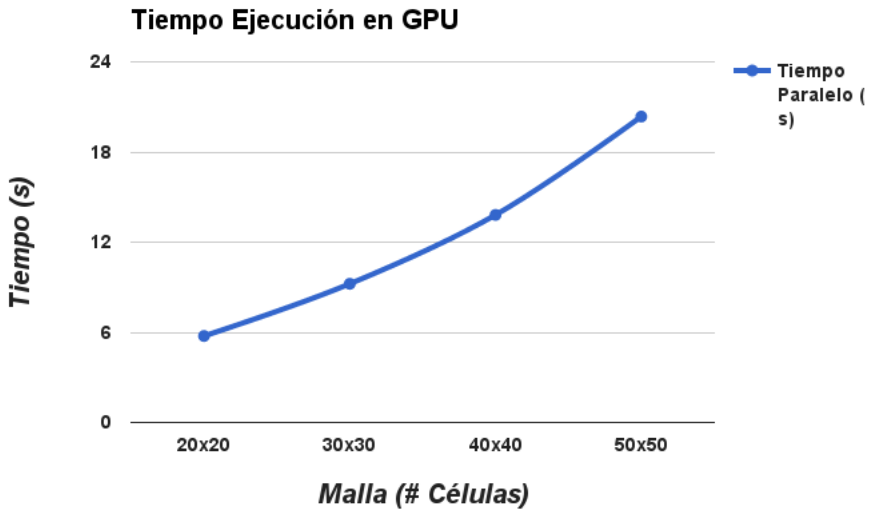

FIGURA 10.6: Tiempo de Ejecución algoritmo GPU

Como puede verse en las figuras 10.5 y 10.6 los tiempos de ejecución difieren bastante, siendo mucho menores los tiempos tomados por el algoritmo implementado en GPU para solucionar el modelo. Se toma como máximo tamaño de malla el de 50x50 porque daba un resultado en tiempos razonables para la implementación en CPU.

En la figura 10.7 se ve la gráfica de aceleración del modelo implementado, es importante aclarar que se están simulando 300ms de BCL de tejido aurícular, con una discretización espacial igual a 0.02cm y una discretización temporal de 0.025s.

De la gráfica puede obtenerse que en efecto la máxima aceleración del algoritmo que pudo registrarse fue de 240X, para una malla de 50X50 células, circunstancia que teóricamente

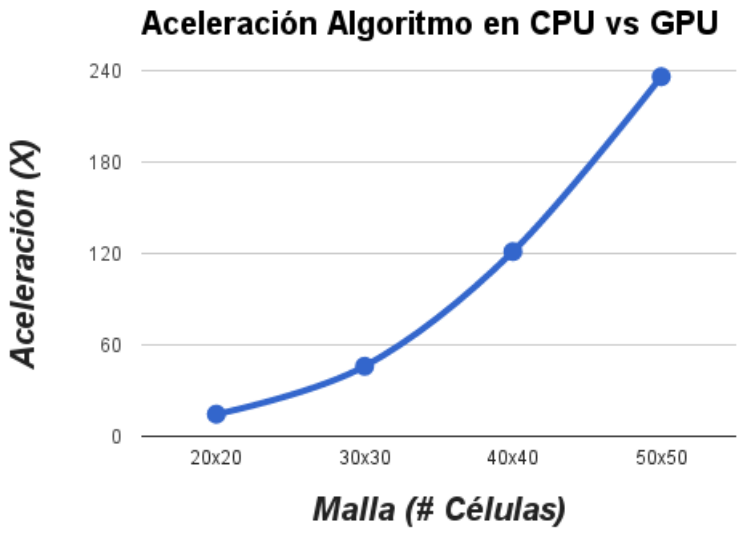

FIGURA 10.7: Aceleración obtenida por el Algoritmo Implementado

se esperaba si se tiene en cuenta la cantidad de núcleos prosentes en una tarjeta como la Tesla K40C, y que la solución del sistema de ecuaciones $AX = B$ es altamente paralelizable.

En la figura 10.8 se ve la propagación del potencial de acción sobre una malla de 165x165 células cuando es excitada la tercera fila de células del tejido, se está simulando un $BCL = 1200ms$, se pueden observar cuatro frentes de onda que muestran su comportamiento en cuatro diferentes pasos de tiempo.

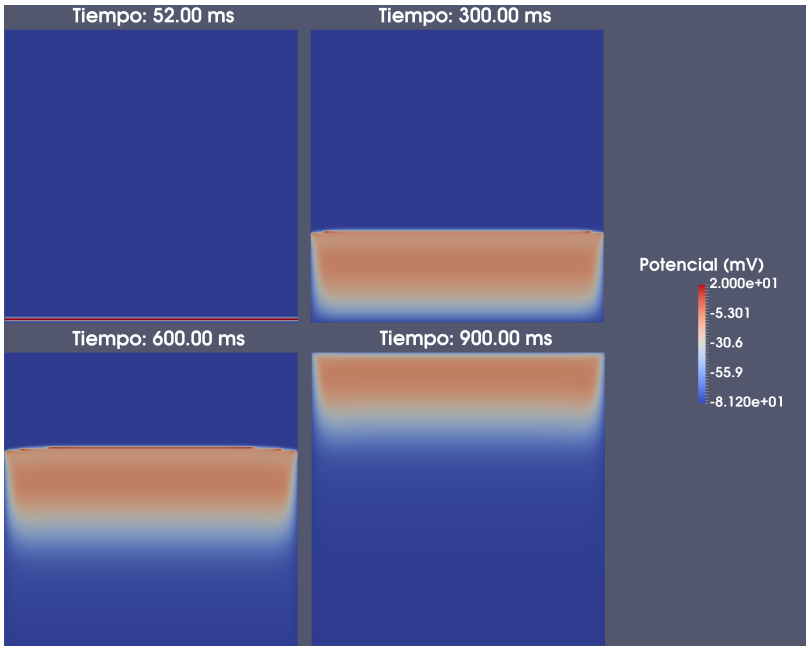

FIGURA 10.8: Simulación de una malla de 165x165, cuando se excita una fila de células del tejido

La figura 10.9 muestra el proceso de simulación cuando se estimula inicialmente la columna de células central del modelo, también son cuatro frentes de onda en diferentes pasos de tiempo.

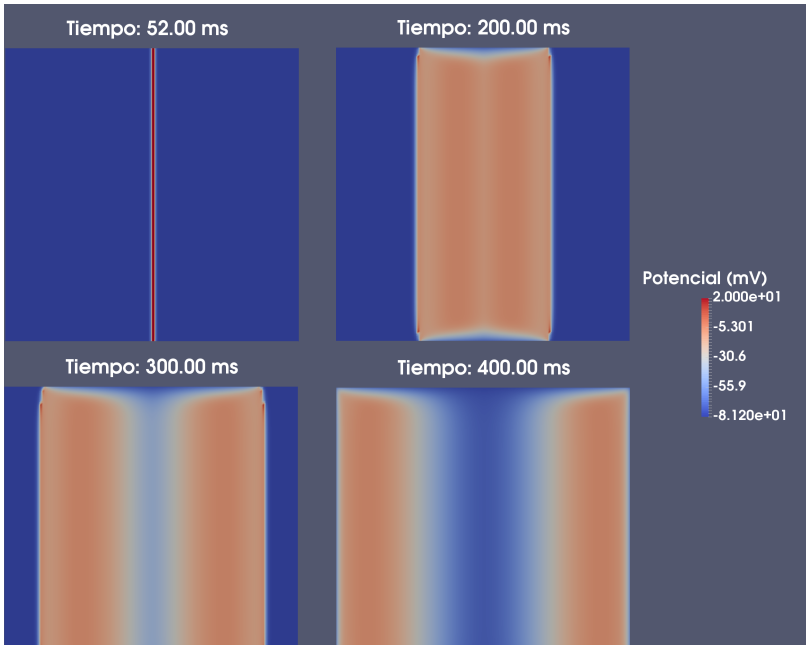

FIGURA 10.9: Simulación de una malla de 130x130 células, cuando se excita una columna

Finalmente en la generación de estímulos se prueba como se ve en la figura 10.10, una estimulación de un rectángulo de células, en 4 diferentes pasos de tiempo.

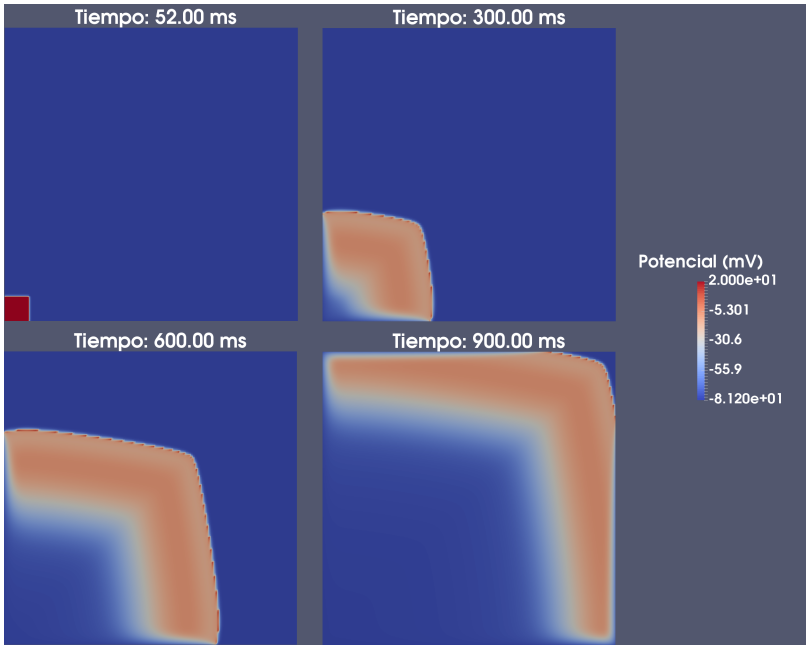

FIGURA 10.10: Simulación de una malla de 165x165 células, cuando se excita un recuadro de células

También fue realizado un proceso de comparativa entre los tiempos de ejecución del algoritmo paralelizado en CUDA y el mismo algoritmo con el proceso de visualización activado, ésta información puede consultarse en la figura 10.11.

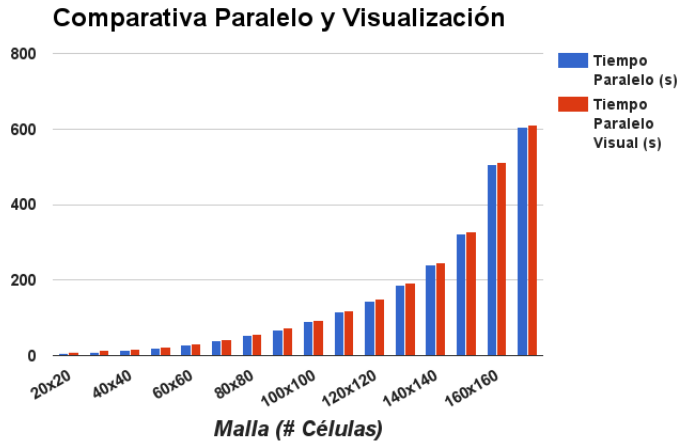

FIGURA 10.11: Comparativa de tiempos entre la versión en CUDA sin Visualización y con Visualización

Ésta última gráfica permite establecer que el proceso de visualización no impacta significativamente el desempeño del algoritmo, la afectación se da por un valor constante promedio de 4.2 segundos que se le suma a cada uno de los tiempos de la versión en CUDA sin visualización.

Finalmente también se puede observar una gráfica de utilización de memoria de la GPU, ver figura 10.12.

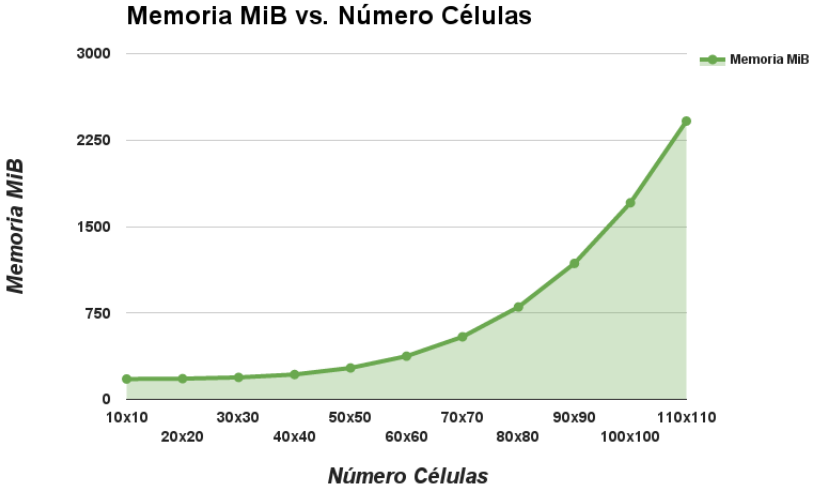

FIGURA 10.12: Consumo de memoria de la GPU de acuerdo al tamaño de Malla

Como resultados adicionales del proceso de investigación se obtuvieron los siguientes productos:

- Tesis de Pregrado sobre Simulación Eléctrica cardiaca de los Estudiantes Ivan Darío Valencia y Daniel Felipe Marín.
- Póster sobre simulación de una fibra auricular en el XXVI Congreso Colombiano de Cardiología y Cirugía Cardiovascular .
- Póster Internacional sobre Aceleración de un Modelo de Simulación Eléctrica Cardiaca utilizando GPUs en el GPU Technology Conference en San Diego-California.
- Ponencia en el 11 Congreso Colombiano de Computación sobre implementación del modelo de Courtemanche sobre unidades de Procesamiento Gráfico.
- Repositorio abierto en GitHub que contiene el código fuente en su totalidad y que puede ser utilizado para futuras investigaciones

Capítulo 11

Conclusiones y Trabajos Futuros

El proceso de investigación permitió en primera instancia la construcción de un modelo eléctrico celular cardiaco en dos dimensiones que será utilizado como herramienta de simulación de patologías médicas como se menciona en [64], esto gracias al contacto que se obtuvo con el médico cardiólogo internista Eduardo Ramirez a través del doctor Oscar Henao, con quien se está implementando un modelo de Nodo Sinusal para evaluar las consecuencias que tiene el marcapasos natural del corazón sobre tejido auricular.

El algoritmo en CPU del modelo de Courtemanche fue construído utilizando armadillo como librería matemática y de paralelización en CPU, dicho modelo fue comparado en desempeño con la implementación en GPU construida utilizando CUDA y ArrayFire como solver del sistema de ecuaciones lineales, ambas implementaciones pueden ser consultadas en el repositorio abierto en GitHub del proyecto. La aceleración obtenida como pudo verse en la figura 10.7 muestra un excelente rendimiento comparado con la versión paralelizada sobre una sola CPU utilizando todos sus cores. Es importante anotar que a futuro la idea es que el proyecto pueda ejecutarse sobre diferentes nodos de cómputo cada uno de los cuales podría contar con co-procesadores masivamente paralelos.

Se construyó un esquema de visualización en tiempo de ejecución del modelo de Courtemanche utilizando *Paraview-Catalyst* como herramienta de visualización *in-situ*, éste esquema podrá ser utilizado para el proceso de visualización de simulaciones más complejas aprovechando ya no solo 2 nodos sino pudiendo escalar a un clúster con mayor cantidad de elementos de procesamiento. Adicionalmente y de acuerdo a la investigación desarrollada en [60] y [59] los procesos de visualización podrán ser implementados de mejor manera a partir de la utilización de una librería que se encuentra aún en desarrollo como lo es VTK-M, o aprovechando directamente las GPUs para realizar el proceso de renderizado mientras el cálculo es realizado.

La figura 10.12 muestra un problema de complejidad espacial debido al tamaño de la matriz A, este problema será atacado a futuro a traves de la utilización y construcción de algoritmos que permitan trabajar con matrices dispersas.

La implementación de un modelo auricular y ventricular en 3D queda como objetivo a desarrollarse en procesos investigativos futuros, al igual que sus procesos de visualización y respectiva aceleración utilizando arquitecturas masivamente paralelas, el algoritmo debe implementarse con mallas no estructuradas y con un metodo numérico diferente como Elementos Finitos (FEM).

Constrastando con los objetivos especificos planteados para el desarrollo del proyecto, puede encontrarse entonces que cada uno de ellos fue alcanzado, los productos resultado de cada uno puede verificarse en el repositorio del proyecto, y adicionalmente como parte del contenido del presente documento.

- Estudiar los conceptos básicos de bioelectricidad relacionados con la formación de potenciales eléctricos cardiacos. El resultado de este objetivo puede verse durante la definición del marco teórico y estado del arte del presente documento.
- Estudiar el modelo eléctrico de Courtemanche y la visualización en tiempo de ejecución. Este objetivo se hace tangible en la medida en que su comprensión se ve a través de la documentación del estado del arte en el tema de visualización realizado.
- Implementar en lenguaje C (secuencial) el modelo eléctrico básico de Courtemanche de una célula cardíaca. Este modelo se encuentra desarrollado en el repositorio de Github.
- Construir un arreglo de células cardíacas en 2D para simular la propagación del potencial de acción utilizando el modelo del cable o el monodominio en lenguaje C (Secuencial), aplicando el método de diferencias finitas. Como es claro en el documento se realiza el proceso de desarrollo matemático de este proceso, adicionalmente el repositorio en Github del proyecto contiene el código fuente que muestra el desarrollo de éste objetivo.
- Implementar y simular la propagación del potencial de acción en un arreglo de células cardíacas en 2D aplicando el método de diferencias finitas a través de CUDA. Se construye para este objetivo un programa de cómputo que muestra la utilización de CUDA y Arrayfire para la paralelización del modelo auricular de Courtemanche. El código fuente se encuentra en Github.
- Implementar un esquema de visualización científica en tiempo de ejecución utilizando Paraview. El esquema de visualización en tiempo de ejecución es desarrollado a

través de la utilización de Paraview, Python, C y CUDA. El código de la aplicación puede verse en el repositorio de Github.

- Evaluar el desempeño del modelo construido en C y CUDA. Se construyen gráficas de tiempos y de aceleración comparativos entre ambas implementaciones, éstas pueden verse en el desarrollo del trabajo.

Bibliografía

[1] Andrés P. Castaño and Carlos A. Ruiz. Propagación de onda en un tejido cardiaco 3d usando dos modelos auriculares. *Revista Colombiana de Cardiología*, 20(4):201–207, Febrero 2013. URL `http://www.revcolcard.org`.

[2] EM Photonics. Em photonics cula tools, June 2009. URL `https://developer.nvidia.com/em-photonics-cula-tools`.

[3] Richter Jake, Eckert Jeff, Nelson Jean, and Welch Jill. S3 video boards. *Infoworld*, may 1992.

[4] Trader T. Nvidia kepler parts top green500, Nov 2013. URL `http://www.hpcwire.com/2013/11/22/nvidia-kepler-parts-top-green500/`.

[5] TOP 500 List. Top 500 tree maps, nov 2015. URL `http://top500.org/statistics/treemaps/`.

[6] nVidia. About cuda, nov 2015. URL `https://developer.nvidia.com/about-cuda`.

[7] Courtemanche M., Rafael R., and Stanley N. Ionic mechanisms underlying human atrial action potential properties: Insights from a mathematical model. *The American Physiological Society*, pages 301–321, 1998.

[8] Kitware. Paraview website, Agosto 2015. URL `http://www.paraview.org/`.

[9] Green 500 List. Green 500, nov 2015. URL `http://www.green500.org/`.

[10] Intel. Intel web site, nov 2015. URL `http://www.intel.la/content/www/xl/es/homepage.html`.

[11] amd. amd web site, sep 2015. URL `http://www.amd.com/en-us`.

[12] nVidia. nvidia web site, Feb 2016. URL `http://www.nvidia.com/page/home.html`.

[13] Centro De Bio-Informática BIOS. Bios web site, Jan 2016. URL `http://bios.co`.

[14] Department of Biological Statistics and Computational Biology at Cornell University. Computational biology, nov 2015. URL http://bscb.cornell.edu/about/computational-biology.

[15] EM Photonics. Cula tools web site, Jun 2015. URL http://www.culatools.com/.

[16] M. Vázquez and G. Houzeaux. Alya red - hpc-based computational biomechanics for supercomputers, nov 2015. URL http://www.bsc.es/computer-applications/alya-red-ccm.

[17] M. Vázquez and G. Houzeaux. Cardiac computational modeling ccm, nov 2015. URL http://www.bsc.es/computer-applications/alya-red-hpc-based-computational-biomechanics/cardiac-computational-modeling.

[18] Nygren A, Fiset C., Firek L., Clark JW, and Clark RB. Mathematical model of an adult human atrial cell: the role of k+ currents in repolarization. *National Center for Biotechnology Information*, pages 63–81, 1998.

[19] Andres Paolo Castano. *Caracterización de Arritmias Cardiacas a través de un Estudio de Simulación*. PhD thesis, Universidad Pontificia de Salamanca, 2015.

[20] Pavan Yalamanchili, Umar Arshad, Zakiuddin Mohammed, Pradeep Garigipati, Peter Entschev, Brian Kloppenborg, James Malcolm, and John Melonakos. ArrayFire - A high performance software library for parallel computing with an easy-to-use API, 2015. URL https://github.com/arrayfire/arrayfire.

[21] kitware. Paraview catalyst, Aug 2016. URL http://www.paraview.org/Wiki/ParaView/Catalyst/Overview.

[22] nVidia. Fermi architecture, nov 2015. URL http://www.nvidia.es/object/fermi_architecture_es.html.

[23] nVidia. Kepler: La arquitectura de cómputo más rápida del mundo, April 2012. URL http://www.nvidia.es/object/nvidia-kepler-es.html.

[24] M. Harris. 5 things you should know about the new maxwell gpu architecture, Feb 2014. URL https://devblogs.nvidia.com/parallelforall/5-things-you-should-know-about-new-maxwell-gpu-architecture/.

[25] nVidia. Nvidia tesla p100. Technical report, nVidia, Abril 2016. URL http://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf.

[26] Alfred E. Buxton, Kristin E. Ellison, and Gregory F. Michaud Malcolm M. Kirk. Introduction to cardiac electrophysiology, the electrocardiogram, and cardiac arrythmias. Technical report, Brown University, Feb 2010.

[27] Howard D. Simms and David B. Geselowitz. Computation of Heart Surface Potentials Using the Surface Source Model. *Journal of Cardiovascular Electrophysiology*, 6(7):522–531, jul 1995. ISSN 1045-3873. doi: 10.1111/j.1540-8167.1995.tb00425.x. URL `http://doi.wiley.com/10.1111/j.1540-8167.1995.tb00425.x`.

[28] Felipe Alonso Atienza. Estudio de los mecanismos de las arritmias cardíacas mediante modelado y procesado robusto digital de señal, 2008.

[29] C.S Henriquez. Simulating the Electrical Behavior of Cardiac Tissue Using the Bidomain Model. *Crit Rev Biomed Eng*, 1993.

[30] L. A Segel and L Edelstein-Keshet. A Primer on Mathematical Models in Biology. *Society for Industrial and Applied Mathematics (SIAM)*, 2014.

[31] M Potse. Mathematical Modelling and Simulation of Ventricular Activation Sequences: Implications for Cardiac Resynchronization Therapy. *Cardiovasc Transl Res*, 2012.

[32] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1-2):500–544, 1952. ISSN 00928240. doi: 10.1007/BF02459568. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/pdf/jphysiol01442-0106.pdf`.

[33] Luo C. and Rudy Y. A dynamic model of the cardiac ventricular action potencial. *American Heart Association*, pages 1071–1096, 1994.

[34] G W Beeler and H Reuter. Reconstruction of the action potential of ventricular myocardial fibres. *The Journal of physiology*, 268(1): 177–210, 1977. ISSN 0022-3751. doi: 10.1113/jphysiol.1977.sp011853. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1283659{&}tool=pmcentrez{&}rendertype=abstract`.

[35] Auckland Bioengineering Institute. Cellml project, Dec 2014. URL `http://www.cellml.org`.

[36] Elvio A. Heidenreich. Algoritmos para ecuaciones de reacción difusión aplicados a electrofisiología, 2009.

[37] Carlos Ruiz. *Estudio de la vulnerabilidad a reentradas a través de modelos matemáticos y simulación de la aurícula humana.* PhD thesis, Universidad Politécnica de Valencia, 2011. URL `http://riunet.upv.es/handle/10251/9104?locale-attribute=ca`.

[38] Oscar Henao. *Vulnerabilidad a arritmias por reentrada en pared ventricular heterogénea en presencia de isquemia regional: estudio teórico de simulación.* Servicio de Publicaciones, Valencia (Spain), 1 edition, 2008. ISBN 9788469115428.

[39] Vincent Jacquemet. A biophysical model of atrial fibrillation and electrograms: formulation, validation and applications, 2004.

[40] Catalina Tobón Zuluaga. Modelización y evaluación de factores que favorecen las arritmias auriculares y su tratamiento mediante técnicas quirúrgicas. estudio de simulación, 2010.

[41] Boris Ja. Kogan. *Introduction to Computational Cardiology.* Springer Science+Business Media, 1 edition, 2010.

[42] Karen Cardona Urrego. Modelización matemática y simulaciónde los efectos de la lidocaína sobre la actividad eléctrica cardíaca en tejido ventricular, 2008.

[43] de la Loma José María, Cardona Karen, and Saiz Javier. Modelado y simulación de la actividad eléctrica de células ventriculares. *Rev. Fac. Ing. Univ. Antioquia Antioquia*, 80-89(46):31–34, 2008.

[44] kitware. Visualization toolkit (vtk), Jun 2016. URL `http://www.vtk.org/`.

[45] Alfonso Bueno-Orovio, Elizabeth M. Cherry, and Flavio H. Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544–560, 2008. ISSN 00225193. doi: 10.1016/j.jtbi.2008.03.029.

[46] Stanley Rush and Hugh Larsen. A Practical Algorithm for Solving Dynamic Membrane Equations. *IEEE Transactions on Biomedical Engineering*, BME-25 (4):389–392, jul 1978. ISSN 0018-9294. doi: 10.1109/TBME.1978.326270. URL `http://ieeexplore.ieee.org/document/4122859/`.

[47] J. Sundnes, R. Artebrant, O. Skavhaug, and A. Tveito. A Second-Order Algorithm for Solving Dynamic Cell Membrane Equations. *IEEE Transactions on Biomedical Engineering*, 56(10):2546–2548, oct 2009. ISSN 0018-9294. doi: 10.1109/TBME.2009.2014739. URL `http://ieeexplore.ieee.org/document/4785511/`.

[48] Mauro Perego and Alessandro Veneziani. An efficient generalization of the rush-larsen method for solving electro-physiology membrane equations. *Electronic Transactions on Numerical Analysis*, 35:234–256, 2009. ISSN 10689613.

[49] Bruno Gouvêa de Barros, Rafael Sachetto Oliveira, Wagner Meira, Marcelo Lobosco, and Rodrigo Weber dos Santos. Simulations of Complex and Microscopic Models of Cardiac Electrophysiology Powered by Multi-GPU Platforms. *Computational and Mathematical Methods in Medicine*, 2012:1–13, 2012. ISSN 1748-670X. doi: 10.1155/2012/824569. URL `http://www.hindawi.com/journals/cmmm/2012/824569/`.

[50] Venkata Krishna Nimmagadda, Ali Akoglu, Salim Hariri, and Talal Moukabary. Cardiac simulation on multi-GPU platform. *The Journal of Supercomputing*, 59 (3):1360–1378, mar 2012. ISSN 0920-8542. doi: 10.1007/s11227-010-0540-x. URL `http://link.springer.com/10.1007/s11227-010-0540-x`.

[51] K H W J Tusscher, D Noble, P J Noble, and A V Panfilov. A model for human ventricular tissue. *American Journal of Phisiology*, pages 1573–1589, 2004.

[52] Christopher D. Marcotte and Roman O. Grigoriev. Implementation of PDE models of cardiac dynamics on GPUs using OpenCL. *Journal of Computational Physics*, sep 2013. URL `http://arxiv.org/abs/1309.1720`.

[53] V.M. Garcia-Molla, A. Liberos, A. Vidal, M.S. Guillem, J. Millet, A. Gonzalez, F.J. Martinez-Zaldivar, and A.M. Climent. Adaptive step ODE algorithms for the 3D simulation of electric heart activity with graphics processing units. *Computers in Biology and Medicine*, 44:15–26, 2014. ISSN 00104825. doi: 10.1016/j.compbiomed.2013.10.023.

[54] Yong Xia, Kuanquan Wang, and Henggui Zhang. Parallel Optimization of 3D Cardiac Electrophysiological Model Using GPU. *Computational and Mathematical Methods in Medicine*, 2015:1–10, 2015. ISSN 1748-670X. doi: 10.1155/2015/862735. URL `http://www.hindawi.com/journals/cmmm/2015/862735/`.

[55] Chunxi Zhao. Computer simulation implementations and optimization of the right atrium of the heart based on GPU. *International Conference on Materials Engineering and Information Technology Applications*, page 5, 2015. URL `http://www.atlantis-press.com/php/download{_}paper.php?id=25838607`.

[56] M. Reumann, C. Morris, D. U. J. Keller, G. Seemann, O. Dössel, G. D. Abram, and J. J. Rice. In *Towards Run Time Visualization in Cardiac Modeling*, pages 999–1002. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-03882-2_266. URL `http://link.springer.com/10.1007/978-3-642-03882-2{_}266`.

[57] Kanthasamy Kalpana. *Parallel Visualization of a 3D Heart Model in an Heterogeneous Computing Environment*. PhD thesis, Malaya University, 2010. URL `http://repository.um.edu.my/481/1/Dissertation-Kalpana.pdf`.

[58] Marzia Rivi, Luigi Calori, Giuseppa Muscianisi, and Vladimir Slavnic. In-situ visualization: State-of-the-art and some use cases. *PRACE White Paper*, pages 1–18, 2012. URL `http://www.prace-ri.eu/IMG/pdf/In-situ{_}Visualization{_}State-of-the-art{_}and{_}Some{_}Use{_}Cases-2.pdf`.

[59] Kenneth Moreland, Matthew Larsen, and Hank Childs. Visualization for Exascale: Portable Performance is Critical. *Supercomputing Frontiers and Innovations*, 2(3): 67–75, jul 2015. ISSN 23138734. doi: 10.14529/jsfi150306. URL `http://superfri.org/superfri/article/view/77`.

[60] Kenneth Moreland, Christopher Sewell, William Usher, Li-ta Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, Matthew Larsen, Chun-Ming Chen, Robert Maynard, and Berk Geveci. VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications*, 36(3):48–58, may 2016. ISSN 0272-1716. doi: 10.1109/MCG.2016.48. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7466740`.

[61] Daniel Marín and Iván Valencia. *Implementacion de Diferencias Finitas en CUDA Aplicado a un Modelo 2D del Comportamiento Electrico Auricular*. Pública, Universidad Tecnológica de Pereira, 2015.

[62] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[63] Sanderson Conrad. Armadillo c++ library, 2015. URL `http://arma.sourceforge.net/`.

[64] John Roy M and Kumar Saurabh. Sinus Node and Atrial Arrhythmias. *Circulation - American Heart Association*, page 10, 2016. doi: 10.1161/CIRCULATIONAHA.116.018011.