Instituto Tecnológico de Costa Rica

Vicerrectoría de Investigación y Extensión

Dirección de Proyectos

Informe final de proyecto de investigación

Documento I

Implementación Multi-FPGA de modelos artificiales del cerebro

1360013

Adscrito a:
Escuela de Ingeniería Electrónica

Investigador principal:
Dr. Alfonso Chacón Rodríguez, EIE

Investigador participante:
Ms.C. Carlos Salazar García, IMT

Asistentes de investigación:
Ing. Reinaldo Castro González, EIE

Ing. Kaleb Alfaro Badilla, EIE

Ing. Daniel Zúñiga, estudiante EIE

Ing. Andrés Arroyo Romero, estudiante EIE

Ing. Javier Espinoza González, estudiante CE

Ing. Keilor Mena, estudiante EIE

Ing. Luis León Vega, estudiante EIE

Ing. Ignacio Fernández Garita, estudiante CE

Joseph Blanco Fallas, estudiante EIE

5 de septiembre de 2019

Índice general

1. Datos generales

1.1 Código y título del proyecto

Nombre	Implementación Multi-FPGA de modelos artificiales del cerebro
Código	1360013
Adscrito a	Escuela de Ingeniería Electrónica
Unidades académicas participantes	Área de Ingeniería Mecatrónica

1.2 Autores y direcciones

Investigador principal:
Dr. Ing. Alfonso Chacón Rodríguez, EIE, alchacon@tec.ac.cr

Investigador participante:
MSc. Carlos Salazar García, IMT, csalazar@tec.ac.cr

Asistentes de investigación:
Ing. Reinaldo Castro González, EIE

Ing. Kaleb Alfaro Badilla, estudiante maestría en Electrónica, EIE

Ing. Daniel Zamora Umaña, estudiante EIE

Ing. Andrés Arroyo Romero, estudiante EIE

Ing. Javier Espinoza González, estudiante CE

Ing. Keylor Mena Venegas, estudiante EIE

Ing. Luis León Vega, estudiante EIE

Ing. Ignacio Fernández Garita, estudiante EIE

Ing. Sebastián González Quesada, estudiante CE

Ing. Giovanni Villalobos Quirós, estudiante CE

Ing. Arturo Salas Delgado, estudiante CE

Ing. Francisco Alvarado Ferllini, estudiante CE

Ing. Stiven Sánchez Rodríguez, estudiante CE

Joseph Blanco Fallas, estudiante EIE

Periodo de ejecución:

1 de enero 2017 a 31 de diciembre 2018

2. Resumen

Se ha concluido un proyecto de investigación y desarrollo financiado y apoyado por la Vicerrectoría de Investigación y Extensión, la Vicerrectoría de Docencia y la Escuela de Ingeniería Electrónica del Tecnológico de Costa Rica, el Departamento de Neurociencia del Centro Médico Erasmus en Rotterdam, Países Bajos. Este proyecto pretendía desarrollar una red masiva basada en FPGAs para simulaciones biológicamente significativas de modelos cerebrales. La red debía ser capaz de modelar distintos modelos de redes neuronales biológicamente precisas. Una interfaz Web proveería de acceso a investigadores en el mundo entero que quieran usar la plataforma para sus estudios en neurociencia. El resultado final incluye una plataforma flexible y escalable de varias tarjetas con sistemas FPGA, accedible via una interfaz gráfica, y que trae ya integrados los tres modelos neuronales usados típicamente por los investigadores en neurociencia.

El proyecto ha producido tres publicaciones indexadas Scopus. Los resultados publicados muestras que la plataforma es competitiva cuando se compara con plataformas similares recientemente reportadas en la literatura.

2.1 Abstract

A research and development project has been completed. The project has been financed and supported by Vicerrectoría de Investigación y Extensión, Vicerrectoría de Docencia and Escuela de Ingeniería Electrónica from Tecnológico de Costa Rica, and the Neuroscience Department at Erasmus Medical Center, Rotterdam, the Netherlands

This project intended to develop the base of a massive FPGA-based simulation network for biologically-meaningful brain modeling. The network is able to model different models of biologically-accurate artificial neural networks. An accessible Web-based platform provides access for researchers interested in using the platform for their studies Neuroscience, and related fields. The final result of the project is a flexible, scalable, Multi-FPGA board platform, accessible via a web graphic interface, including three neural models tipically used in neuroscience.

The project has produced three Scopus-indexed publications. Published results show that the platform is competitive against similar platforms recently reported in the literature.

3. Introducción

En la pasada década, se ha invertido una gran cantidad de esfuerzo a nivel mundial para construir y simular poderosos modelos cerebrales que pueden ayudar a desentrañar los misterios del cerebro humano (ver por ejemplo este proyecto bandera de la Unión Europea: www.humanbrainproject.eu).

Si bien estos modelos son poderosos y se acercan cada vez más a la funcionalidad de un cerebro real, son típicamente muy intensivos computacionalmente, al punto que plataformas multinúcleo incluso ya no son capaces de alcanzar tiempos razonables de ejecución. Incluso realizando esfuerzos como modificaciones a nivel de software, reduciendo al mínimo el sistema operativo o incluso utilizando sistemas operativos en tiempo real [4], estas técnicas no son suficientes para lograr mejoras en la ejecución de estos modelos.

Los dispositivos FPGA pueden suplir la potencia de cálculo necesaria por su concurrencia inherente. En el DCILab de la Escuela de Ingeniería Electrónica, por ejemplo, se han desarrollado ya aplicaciones que resuelven sobre FPGA operaciones no lineales [6], que se han podido usar exitosamente para el reconocimiento de patrones de disparos de armas de fuego sobre FPGA [20, 21, 10]. Pero incluso las FPGA más modernas no tienen capacidad suficiente para emular simulaciones cerebrales de gran escala (de cientos de miles de neuronas). Es por ello que debe pensarse en extender aún más la capacidad de los sistemas de simulación utilizando redes interconectadas de FPGAs. Esto sin embargo presenta retos a nivel tanto de desarrollo de interconectividad como de síntesis y parelelización de recursos eficiente dentro de los dispositivos para optimizar las velocidades de cálculo, de manera que puedan portarse a la plataforma modelos de redes neurales de tamaño masivo, para llevar a cabo así simulaciones biológicamente sensibles de microsecciones del cerebelo humano.

El Departamento de Neurociencia del Centro Médico Erasmus en Rotterdam, Países Bajos, ya ha propuesto varios modelos computacionales de simulación que han demostrado su fidelidad en emular la funcionalidad del cerebelo humano sobre una sola FPGA (ver [16, 25]). Pero por no estar los sistemas embebidos ni el diseño de sistemas digitales complejos dentro de las especialidades en investigación del centro especialidad del centro, se han topado con dificultades para resolver varios problemas de desarrollo que han motivado la participación de investigadores del Tecnológico en este proyecto. De hecho, ya un estudiante de grado de Ingeniería Electrónica realizó durante el segundo semestre del 2015 una pasantía corta de proyecto de graduación, donde se logró precisamente evaluar un modelo de red olivo-cerebelar inferior sobre una plataforma FPGA. Fueron los resultados positivos de esta corta estancia (por ser publicados) los que demostraron la potencialidad de desarrollar este proyecto en conjunto entre el Centro Médico Erasmus y el Tecnológico de Costa Rica. Resultados que se resumen ya en tres publicaciones además del trabajo doctoral de uno de los investigadores asociado a este proyecto, que incluso realizó una pasantía de casi un año en el 2017-2018 en el mismo Departamento de Neurociencia de Erasmus.

3.1 Objetivos

Se describen aquí los objetivos finales que fueron planteados al iniciar el proyecto, según constan en el comunicado VIE-763-16, que transcribe el acuerdo tomado por el Consejo de Investigación y Extensión de la Vicerrectoría de Investigación y Extensión en su sesión No. 29-2016, artículo 8.14 del 12 de setiembre de 2016. En la sección de análisis de resultados se evaluarán los logros de cada uno de los objetivos específicos, y el grado de cumplimiento del objetivo general.

3.1.1 Objetivo general

El objetivo general propuesto era desarrollar en una plataforma multi-FPGA simulaciones en tiempo real de modelos neuronales biológicamente precisos descargables de una biblioteca de modelos, con una interfaz GUI-Web para entrada de datos, descarga de los modelos neuronales y análisis de resultados.

3.1.2 Objetivos específicos

1. Generar una biblioteca en código HDL de alto nivel portable a FPGA con al menos tres modelos neurales biológicamente realistas.

2. Proponer y evaluar arquitecturas de interconexión de red de datos sobre la que se pueda portar eficientemente un modelo de interconección dinámica de una red neuronal biológicamente precisa.

3. Implementar la arquitectura propuesta en el objetivo anterior con al menos cinco dispositivos FPGA.

4. Desarrollar una interfaz Web-GUI que permita el control, la descarga de los modelos neuronales y el análisis de resultados en tiempo real vía Internet.

4. Marco teórico

Las redes neuronales artificiales (ANN por sus siglas en inglés) son usadas en robótica y aplicaciones en inteligencia artificial en particular por su capacidad de resolver concurrentemente las funciones a través de sus nodos (neuronas), donde la relación entre entradas y salidas de la ANN se determina por la topología de la red y su método de interconexión. Esta topología puede también adaptarse dinámicamente, sujeta a los cálculos neuronales en proceso, imitando de esa manera un comportamiento biológico conocido como plasticidad.

Los avances en la neurociencia y un mayor entendimiento del cerebro han guiado paulatinamente a la creación de modelos matemáticos de neuronas (y de redes completas) que no solo imitan el comportamiento biológico sino que pueden simularlo con apreciable detalle. Un ejemplo es la red neuronal de espiga (Spiking Neural Network o SNN) [1,2]. Aquí, no solo se transfiere información con la tasa de disparo de cada neurona, sino por la transferencia de espigas. Dada la habilidad de las SNN para modelar características neuronales y adaptarlas de acuerdo con la amplitud del tren de impulsos, su fercuecnia y tiempos precisos de arribo. Así una SNN tiene un poder predictivo y computacional superior a una SNN.

Por otra parte, la implementación de redes neuronales más grandes gracias a los avances en la integración de dispositivos microelectrónicos, permite ahora simular de manera biológicamente precisa el funcionamiento del cerebro. Este tipo de simulaciones son muy relevantes, tanto que la Academina Nacional de Ingenieros en Estados Unidos la han incluido dentro de sus grandes retos en ingeniería [3], especialmente por tres razones: (1) Aceleración de la investigación del cerebro: dependiendo de la complejidad de los modelos biológicamente correctos, estas simulaciones pueden proveer de conocimientos que van desde el comportamiento de simples células hasta la dinámica de red de regiones enteras del cerebro, sin necesidad de experimentos in-vivo. (2) Rescate del cerebro: si las funciones cerebrales pueden simularse precisamente y en tiempo real, esto puede llevar a dispositivos para rescatar funciones cerebrales, prótesis robóticas e implantes que restablezcan pérdidas de funcionalidad cerebral. (3) I. A. avanzada: el entender mejor las dinámicas computacionales de los sistemas biológicos puede llevar a modelos de IA bioinspirados más avanzados que los alcanzados actualmente por ANNs para aplicaciones robóticas y autónomas (4). Nuevos paradigmas de arquitectura de computadoras: alternativas al modelo de Von-Neumann serían muy útiles para aplicaciones paralelas masivas y pueden potencialmente proveer de sistemas tolerantes a defectos que emulen la adaptabilidad del cerebro.

Ahora, el reto principal para construir estas SNNs complejas y biológicamente precisas yace sobre todo en la carga computacional y de comunicaciones necesaria para sus simulaciones. Ello sin mencionar el masivo paralelismo implícito en estas redes, con el que las arquitecturas tradicionales de procesamiento por CPUs no manejan bien. Aunque el usar GPUs puede aprovechar su paralelismo de ejecución para optimizar el rendimiento de modelos neuronales, no poseen la capacidad de interconexión capaz de manejar el gran

intercambio de datos necesario en redes neuronales masivas, y por lo tanto se ven limitadas para realizar simulaciones en tiempo real [22]. Además, los GPU son ineficientes en términos de energía. El uso de supercomputadores es por otro lado limitado debido a su costo, dificultad de implementación y costo energético. Uno podría incluso utilizar un sistema operativo en tiempo real, para garantizar una mejora en el cumplimiento de los plazos de cada tarea crítica [4], pero aun así el rendimiento no estaría ni cerca del óptimo requerido. Una alternativa interesante es el uso de ASICs, ya sea de señal mixta o digital. En el primer caso, se pueden aprovechar los modelos mismos de los dispositivos que modelen analógicamente estas neuronas, lo que provee de gran velocidad. Pero la interconectividad de estos circuitos es limitada, sin hablar de su dificultad de implementación y escalamiento, y los problemas asociados con la falta de precisión debida a desapareamiento de transistores, corrientes de fuga, diafonía, etc. [22].

Un ASIC digital por otra parte puede equiparar eficientemente el paralelismo de modelos biológicos en tiempo real, y puede ser muy útil para aplicaciones robóticas y prostéticas, o aplicaciones numéricas sea extremadamente complejas y rápidas, donde más bien sea necesario disminuir en la medida de lo posible el consumo de potencia (ver por ejemplo los desarrollos realizados ya por investigadores del DCILab, en particular [20, 21, 9, 7, 8, 17, 18, 11]. Pero son caros de implementar y, sobre todo, inflexible (la implantación de un nuevo modelo neuronal requeriría un nuevo ciclo de fabricación). Aquí es donde el uso de FPGAs ofrece una alternativa: proveen de suficiente capacidad de desempeño para simulaciones en tiempo real, requieren no mucho más energía que un ASIC digital, y son sobre todo reconfigurables fácilmente, lo que permite modificar modelos de cerebro bajo demanda de una manera rápida, sobre todo si se usan HDLs y síntesis de alto nivel. Esta misma flexibilidad además permite emular la plasticidad de las redes neuronales biológicas en formas que otras soluciones no lo permiten.

La limitante principal para usar FPGAs es que, aunque poseen capacidades de implementar redes neuronales biológicamente precisas de algunos cientos de neuronas [22], aún no pueden contener en una sola de ellas los cientos de miles de neuronas y los millones de interconexiones necesarios para simular una región cerebral. Es aquí donde debe buscarse el crear una red escalable de FPGAs que trabajen en paralelo bajo algún esquema de orquestado de red, que permita distribuir el cómputo paralelo de manera eficiente entre las diferentes FPGA, mientras los datos de estímulo son alimentados y los resultados leídos en tiempo real a través de una interfaz sencilla de alto nivel.

Entre las avenidas a explorar, se halla el uso de lenguajes de alto nivel para acelerar el proceso de construcción del hardware, mediante lo que se conoce como síntesis de alto nivel [26, 1]. La empresa Xilinx ofrece una variedad de este tipo de herramientas dentro de su suite Vivado. Por otro lado, es posible aprovechar la integración existente en los nuevos modelos de sistemas en chip (SoC por sus siglas en inglés), que incorporan aparte de la tela programable de la FPGA, un procesador completo con sus periféricos, lo que permite realizar de manera práctica co-diseños hardware-software, donde una parte del algoritmo que se ejecuta en los CPUs, puede acelerarse mediante unidades de procesamiento personalizadas en hardware (algo que comúnmente recibe el nombre de computación he-

terogénea). Xilix ya provee de dispositivos de ese estilo: la famila Zynq, que incorpora FPGAs de la famila Zrtyx, Kintex y Ultra, con procesadores M9 de ARM, con multiplicidad de recursos de Entrada-salida y almacenamiento. Además, el ambiente de desarrollo de Vivado ofrece también la posibilidad de integrar los diseños en hardware (conocidos como IP) como periféricos accesibles para el procesador via GPIO o buses estándar AXI (incluyendo canales DMA). El ambiende de desarrollo de software (SDK) de estas mismas herramientas, aydan con la interfaz a nivel C o C++, mediante la creación de drivers para aplicaciones tanto bare-metal como sobre sistemas operativos Linux (para más detalles, ver [27, 13, 28]).

5. Metodología

En este proyecto se ha aplicado una metodología basada en el método de ingeniería, en el cual es necesario primero identificar el estado actual del arte fortalecer los conceptos y metodologías de diseño asociadas, proponer arquitecturas o soluciones evaluadas preliminarmente para reducir el ámbito de selección, e implementar aquellas escogidas con su consecutiva evaluación. Este proceso se resumen en las actividades que fue necesario realizar para cumplir con los objetivos planteados en el proyecto.

Así, para cumplir con el objetivo 1, se inició el proceso con una búsqueda exhaustiva de documentación bibliográfica para que, en conjunto con los investigadores de Erasmus, se pudiera realizar a partir de la misma la selección de los modelos neuronales a implementar. Definidos dichos modelos, sería posible una implementación de al menos tres de ellos dentro de una biblioteca en código HLS portable a FPGA. Una vez finalizada la implementación y antes de portear el código a FPGA, prosiguiría una verificación avanzada a nivel de código de los modelos. Finalmente, debería existir un ambiente de verificación dentro de la FPGA para verificar en hardware el funcionamiento de los modelos.

Consecuentemente, para cumplir con el objetivo 2, se estudiaron también estrategias de implementación de buses avanzadas para interconexión a nivel de FPGA y se evaluaron aquellas más adecuadas para seleccionar la que proveyese mejores resultados en cuanto a rendimiento. Definida la estrategia de interconexión se debía realizar la interconexión en tiempo real paralelo entre cinco dispositivos FPGA, corriendo un modelo biológicamente preciso de una microsección biológicamente realista. Esta actividad cumpliría con el objetivo 3

Finalmente, había de concluir con el desarrollo de un servidor web que cargara una página web de manera que se pudiera de manera remota controlar el modelo cerebral que se desea reproducir (ver objetivo 4). Había también que generar el sistema que permitiera cargar de manera remota los archivos de estímulo para la simulación y ejecución sobre toda la red de FPGAs.

En general, los resultados obtenidos y la consolidación de la metodología guiarían la realización de un proyecto a futuro donde se optimizarán las implementaciones propuestas en este proyecto. Vale hacer ver que el producto principal de este proyecto ha sido el simulador llamado TEC-Brain, pero que a estos resultados que se discutirán más adelante, es necesario añadir todo el procedimiento metodológico anterior que incluye herramientas, código y conocimientos que podrán utilizarse en las proyectos que se han generado a partir de los resultados de esta investigación.

6. Análisis de resultados

Este proyecto ha significado el desarrollo de varios productos relacionados con cada uno de los objetivos específicos propuestos, para así culminar en una primera propuesta de un módulo multiFPGA de simulación cerebral realista acelerada, bautizado como TECBrain, cuya imagen final puede verse en la figura 6.1. Es por ello que se desglosan de dicha manera a continuación los resultados particulares del desarrollo de cada objetivo, con su discusión y análisis por separado, para luego ofrecer una discusión conjunta de los mismos y las conclusiones relevantes en la sección 7.

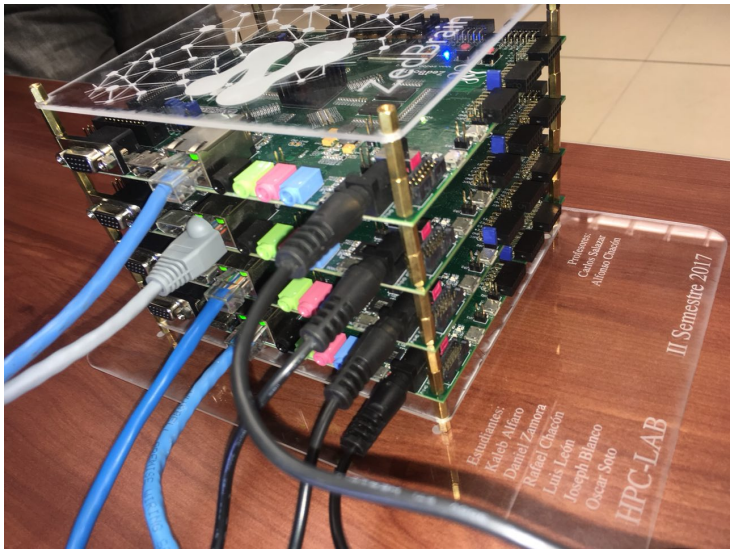

Figura 6.1: Imagen final del TECBrain completamente conectado (no incluida la plataforma de visualización, mostrada por aparte más adelante). TECBrain es sistema multi-FPGA totalmente reconfigurable para simulación cerebral usando una aproximación de codiseño co-design hardware/software.

6.1 Desarrollo de modelos neuronales biológicamente realistas en HDL, HLS, portables a FPGA

. Se han construido tres modelos neuronales biológicamente realistas. Los mismos responden a las necesidades de precisión biológica versus su complejidad de simulación. Los tres modelos en particular fueron:

a. Modelo de integración y disparo Como base de los modelos, se generó un sencillo modelo de integración y disparo. Este modelo sirve únicamente de fundamento, pues no se ajusta específicamente para la aplicación buscada: es demasiado sencillo y no provee de una alta precisión biológica. No obstante, el poseer ya su modelo permite en el futuro hacer pruebas para simulaciones de redes neuronales masivas debido a su bajo costo computacional.

b. Modelo de Izhikevich Pese a la sencillez de este modelo (solo dos ecuaciones diferenciales no lineales acopladas), se ha convertido en un patrón de referencia, pues emula la gran mayoría de los patrones de comportamiento existente en las SNN. Además, se ha implementado en varios sistemas de aceleración de simuladores SNN basados en FPGA, por lo que se convirtió en un candidato natural para el proyecto. Se ha probado el mismo a nivel de código en un microprocesador ARM. No se ha completado su migración a RTL-HLS, la cual se encuentra en proceso en este momento. No obstante, su implementación en ARM ya permite su portabilidad a la Zedboard.

c. Modelo extendido de Hodgking-Huxley del núcleo olivar inferior en el cerebelo. El modelo eHH es un modelo basado en conductancias, y como tal es de los más detallados en su especificidad biológica. Su contracara es el excesivo costo computacional del mismo. De este modelo se implementó tanto una versión pura en software para un clúster Xeon de 64 núcleos, como otra para un ARM dual-core implantado en una tarjeta Zedboard Zynq-7000. Estas dos implementaciones sirvieron de base para el análisis preeliminar de costos de procesamiento, y de paso se conviertieron en los modelos de referencia a vencer una vez que se implementaron las versiones HLS portadas a FPGA (más detalles de estos resultados se pueden obtener de [5, 23, 14, 19]

6.2 Definición de estrategia de interconexión dinámica de la red neuronal

Se arrancó con una propuesta de interconexión vía Ethernet, por su alta flexibilidad y fácil integración. La misma se implementó a través de dos modalidades: con una biblioteca ligera de TCP/IP en un modelo bare-metal de la unidad de procesamiento y luego con un stack de Linux, que permitió la integración del protocolo TCP-IP (ver subsección 6.3 para detalles). No obstante, mientras se avanzaba en el desarrollo ya de la plataforma MultiFPGA, se notó que este esquema de interconexión es insuficiente para soportar la necesaria escalabilidad.

Se procedió entonces a abrir una línea paralela de desarrollo de una arquitectura de capas de comuncación personalizada para TECBrain, que se encuentra en desarrollo en este preciso momento, como parte de la tesis doctoral del investigador Ing. Carlos Salazar García, basándose en un marco generador de interconexiones que desarrolló el Ing. Ronny Ramírez García. El objetivo acá es que la correcta implementación de esta arquitectura permitirá luego intercambiar fácilmente el esquema inicial usando Ethernet, de una manera transparente. Esto no implica, no obstante, que se descartará del todo la conectividad Ethernet, que siguió siendo el medio de acceso a TECBrain para su control, configuración y despliegue de datos (cuyos resultados son explicadon en las siguientes subsecciones). Así fue posible seguir avanzando en la construcción de la plataforma de cómputo necesaria para TECBrain.

6.3 Implementación MultiFPGA del modelo eHH

6.3.1 Primera aproximación: modelos *bare-metal* y basado en MPI (todo el procesamiento en el PL de la Zynq 7020)

Una vez definido el protocolo Ethernet como primera vía de intercomunicación, se procedió a la implementación del modelo eHH sobre FPGA. El primer paso fue un traslado del código C++ provisto por los autores de [22], a la plataforma escogida. Este primer paso se reporta ampliamente en [23, 5]. Se realizó una implementación *bare-metal* (sin sistema operativo) que utilizaba el ARM simplemente como una unidad de gestión de interconexión de datos entre la unidad de simulación del modelo eHH y una PC conectada por red (ver figura 6.2). Se usó una biblioteca que provee de servicios básicos TCP/IP para aplicaciones a nivel bare-metal. La interconexión entre el ARM y la unidad eHH se implementó via un esquema de AXI4-DMA. Los resultados iniciales (ver figura 6.3) muestran como la interconexión multiFPGA va disminuyendo el tiempo de procesamiento para un máximo de 6000 neuronas, pero la latencia de Ethernet es de una a tres milisegundos, dos órdenes de magnitud por encima de lo deseado para una simulación en tiempo real del modelo eHH.

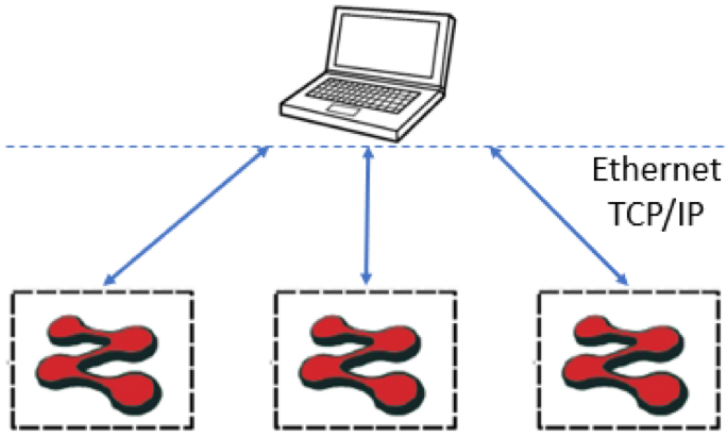

Figura 6.2: Arquitectura general propuesta inicialmente para la interconexión de la red MultiFPGA. Nodos corriendo en *bare-metal*, bajo control vía Ethernet desde una PC.

Ante la situación presentada, se montó una versión mejorada, que distribuyera las tareas usando MPI sobre una plataforma Linaro, pero siempre con todo el peso del cómputo directamente sobre el PL de la Zedboard, según la topología mostrada en la figura 6.4,. No obstante, la figura 6.5, indica que no hay ganancia en esta implementación y más bien aumenta levemente la latencia enormemente (hasta unos 10ms para una red de una decena de nodos, por los costos añadidos al usar MPI). No obstante, se ha ganado flexibilidad en la configuración del sistema. Por ello decidió seguirse por esta línea para la siguiente aproximación.

Finalmente, se construyó una plataforma de visualización en un servidor Web, que permite no solo graficar resultados, sino configurar las simulaciones en términos de tamaño de

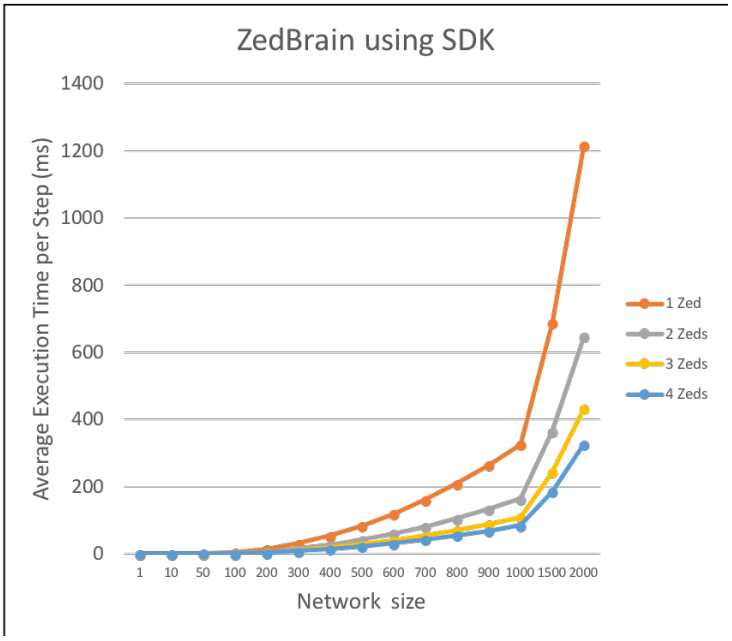

Figura 6.3: Resultados en tiempo bajo la primera aproximación. Nótese el escalamiento conforme más FPGAs se suman al TECBrain. Pero la latencia es muy alta (uno a dos milisegundos para redes de menos de cincuenta nodos, inaceptable para el modelo eHH en tiempo real).

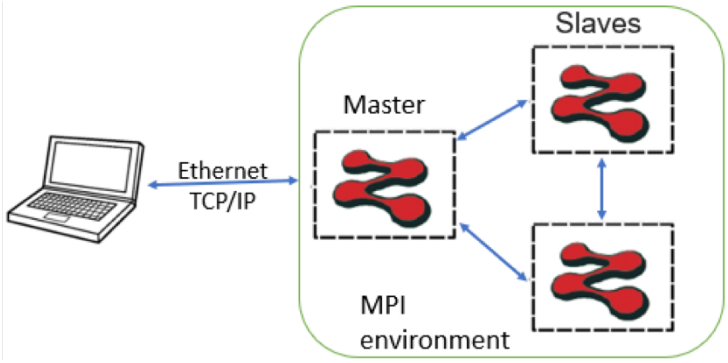

Figura 6.4: Arquitectura general basada en MPI para la interconexión de la red MultiFPGA. Nodos corriendo un sistema operativo Linaro, bajo control vía MPI de una de las Zedboard, que distribuye las tareas.

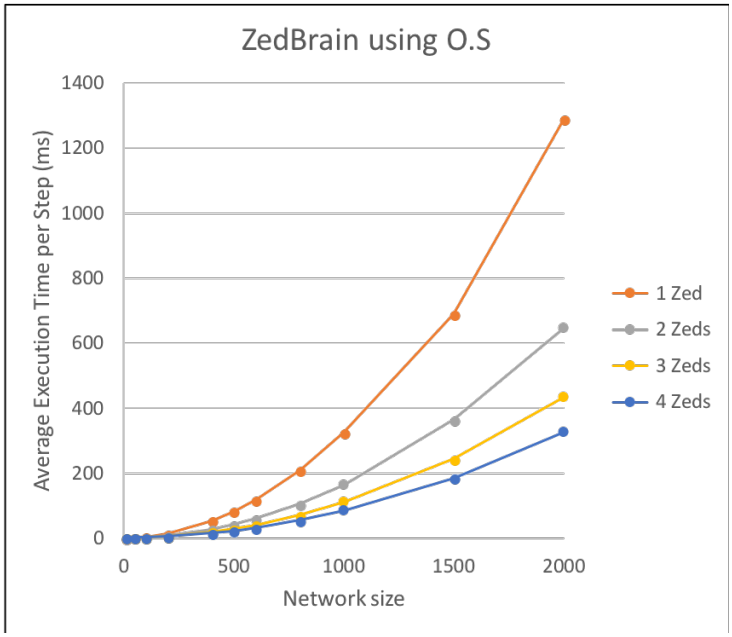

Figura 6.5: Resultados en tiempo bajo la segunda aproximación. Nótese prácticamente no hay ganancia con la implementación anterior. Además, la latencia aumenta a unos seis milisegundos, por los costos añadidos al usar MPI.

la red neuronal y cantidad de pasos a emular. Los resultados de este primer paso, si bien funcionales pues se pudieron simular hasta dos mil neuronas distribuidas entre cuatro FPGAs, distaron mucho de alcanzar un rendimiento satisfactorio. Para más detalles, acceder a [23, 5]. Una demostración de video de esta primera aproximación (llamada ZedBrain entonces), puede mirarse en https://www.youtube.com/watch?v=KgMpX-eb0Cc.

6.3.2 Segunda aproximación: modelo eHH sobre el procesador, con unidad de cálculo de corrientes de unión de brecha en el PL de la Zynq 7020

Para mejorar el rendimiento y las prestaciones de distribución de tareas, se procedió a una segunda propuesta: trasladar el modelo eHH al ARM sobre un sistema operativo Linaro. Esto, sumado a uso de una biblioteca TCP/IP más poderosa para asegurar la confiabilidad de la distribución de datos ya usado en el punto anterior, otorgó flexibilidad para el almacenamiento de datos y control de los dispositivos, en especial de los canales DMA. Así se pudo aprovechar mejor las capacidades en DRAM disponibles en la Zedboard (pues ya no sería necesario almacenar todos los datos necesarios para la simulación dentro de los BRAM de las Zynq 7020) y así aumentar el número de células a simular. Por otra parte, como fue visto en la primera aproximación, fue claro que el uso de MPI desde una tarjeta Zedboard maestra, ayudaba a flexibilizar y balancear la distribución de la red completa sobre las distintas placas Zedboard. Así, se procedió a la migración hacia una arquitectura más hardware-software, con una parte de la emulación (los compartimentos principales de Soma, Axon y Dendrita) trasladados directamente al ARM, donde se podrían ejecutar de manera más eficiente gracias a la capacidad multihilo del procesador, y la existencia de una

unidad de coma flotante integrada. Así, en el PL del Zynq 7020 solo fue necesario ahora implementar la unidad de cálculo de uniones de brecha (Gap Junction Unit), que debido a la alta interconectividad del modelo eHH del núcleo olivar inferior, es típicamente el que más recursos de computación absorbe. Un estudio detallado al respecto, y los resultados parciales de esta segunda iteración pueden hallarse en [14]. En la figura 6.6 se muestra el diagrama funcional del hardware tal cual se implementó sobre la Zynq.

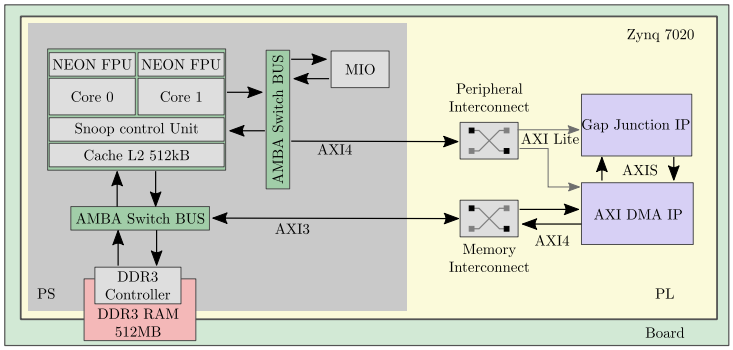

Figura 6.6: Arquitectura general del sistema. La GJU se controla por un hilo de ARM A9, via AXI4, mientras que el otro hilo resuelve los otros compartimentos y las operaciones de entrada y salida de la Zedboard. Imagen tomada de [3].

Además, su posterior mejora fue presentada en un paper en la conferencia LASCAS 2019 [3], donde ya fue posible superar, en una sola placa Zeboard de bajo costo, la velocidad de simulación de un procesador i7 Intel con cuatro núcleos de 64 bits a 3.9GHz. Un ejemplo de los tiempos logrados se muestra en la figura 6.7.

6.3.3 Tercera aproximación: aplicación de un modelo de flujo de datos optimizado para HLS

Con las lecciones aprendidas en los pasos anteriores, y teniendo claro ahora que las barreras principales que evitan el máximo aprovechamiento de la FPGA son el ancho de banda de la memoria dispobible en la Zedboard, y la latencia asociada con el protocolo TCP/IP, se determinó que era necesario atacar dos frentes. El primero, fue optimizar aún más el procesamiento a nivel de GJU, aplicando técnicas de optimización HLS que dirigen la síntesis en función del hardware que es posible desarrollar. Este tipo de técnicas implica un conocimiento maduro sobre el diseño modular digital, de manera que se codifique el C++ para que el HLS interprete de manera más eficiente las estructuras más acordes con el flujo de datos deseado. Esta metodología es tomada de [12], y fue aplicada exitosamente en [19]. Además, fue la base de un paper aceptado en la conferencia CARLA-2019 [2]. Aquí se demuestra ya que el diseño fundamental de TECBrain sobre la Zedboard es competitiva contra el mostrado en [22], con una tarjeta de muchas más altas prestaciones. Nótese en particular en la tabla 6.1, que ha sido tomada de[2], que se ha mejorado sustancialmente el uso de recursos (ir a la referencia para más detalles). La figura 6.8 muestra un detalle de la modularización del diseño, descrita en C++ para guiar a la herramienta HLS para

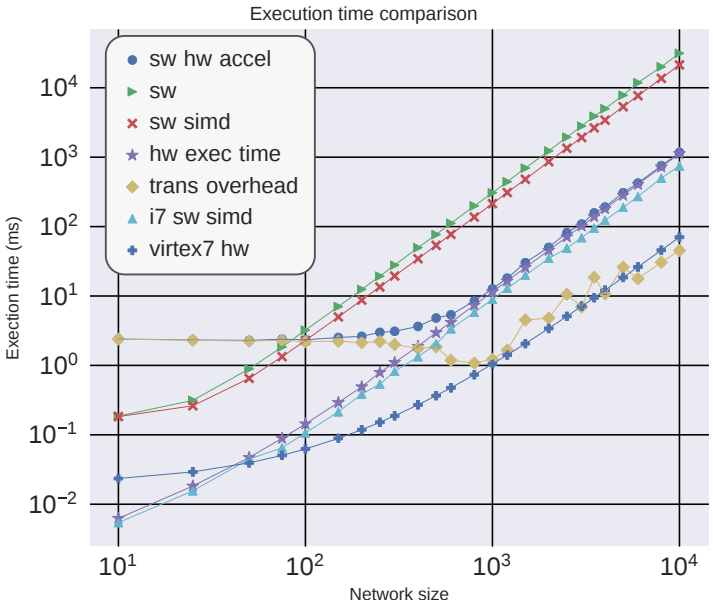

Figura 6.7: Comparación de tiempos para el modelo eHH model sobre: implementación de la segunda aproximación (*sw-hw-accel*), sobre un hilo del Zynq's ARM A9 (*sw*), sobre el mismo ARM A9 con código optimizado NEON SIMD, (*sw-simd*), sobre un Intel i7 de 64 bits, optimizado con AVX2 SIMD para un solo hilo (*i7 sw simd*), y sobre una Virtex XC7VX485T @100Mhz (*virtex hw*). Nótese que *sw hw accel* se descompone en las curvas *hw exec time* y *trans overhead*, lo que significa que esta segunda iteración, las transacciones de memoria absorben el tiempo útil de computación. Imagen tomada de [3].

un diseño óptimo. Un detalle de las directivas utilizadas para la herramienta de Vivado se ofrece en [19].

Tabla 6.1: Uso de los recursos de la FPGA y capacidad de rendimiento para operaciones de precisión sencilla de coma flotante, ejecutadas en la tela de la FPGA, comparada contra una red ION todos contra todos [22] y una de conectividad a ocho vías presentada en [29]. La columna t_{step} representa el tiempo de ejecución por paso de simulación. Se ofrece una densidad de desempeño (razón entre FLOPS y recursos FPGA) más eficiente, para los dadas unidades DSPs y LUTs (tabla tomada de [2]

Source	FPGA	f_{clk}	DSP	LUT	SimC	t_{step} (ms)	MFP opts	MFLOPS	MFLOPS /DSP	MFLOPS /LUT	FLOPS /(DSP $\cdot f_{clk}$)
This work	Zynq-7000	120MHz	91	15k	1056	4.8	13.38	2788	30.64	0.183	0.2532
					1188	6.03	16.94	2809	30.86	0.184	0.2551
Smaragdos	Virtex7	100MHz	1600	251k	1056	1.1	14.29	12990	8.119	0.052	0.0812
Zajo	Virtex7	100MHz	1008	190k	1188	0.05	1.135	22690	22.51	0.012	0.2251

Una comparación entre los resultados originales en [22] y los obtenidos hasta ahora se muestra en la figura Fig. 6.9 (escala logarítmica). La resolución promedo se ha mejorado hasta 4× contra los resultados en [3] para poblaciones de celdas ION de 1000 o más, hasta alcanzar 10× y más para poblaciones debajo de 1000 celdas (conexión todas contra todas). El tiempo sin embargo no es determinístico aún, y debe disminuirse la latencia introducida

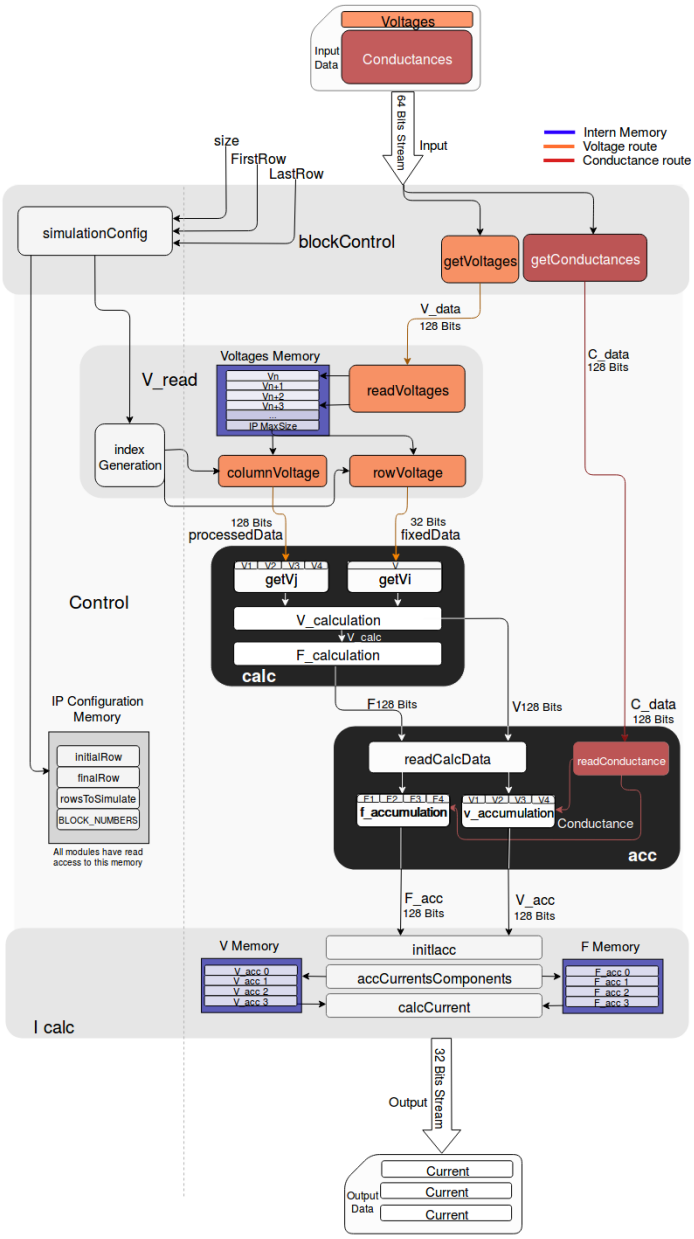

Figura 6.8: Detalle del flujo de datos implementado en el código HLS para resolver las dependencias de datos y paralelizar operaciones de una manera adecuada para un diseño sincrónico. Imagen tomada de [2].

por la intercomunicación si esto fuera a distribuirse entre varista etapas. Se ofrece como referencia la comparación de dos simulaciones (ver figura 6.10). El error está acotado en 0.00001 para el peor caso. El tiempo requerido para tres segundos de simulación de actividad cerebral es de nueve minutos para el TECBrain en con sola Zedboard, mientras que una PC multi-hilo PC de 64 bits(i7-7820HQ@3.9GHz) lo completa en tres minutos. Solo un tercio de la velocidad comparado con una plataforma mucho más lenta y barata en términos de costo y potencia: dos Watts, contra los 45 Watts reportados para un i7-7820HQ@3.9GHz a plena capacidad.

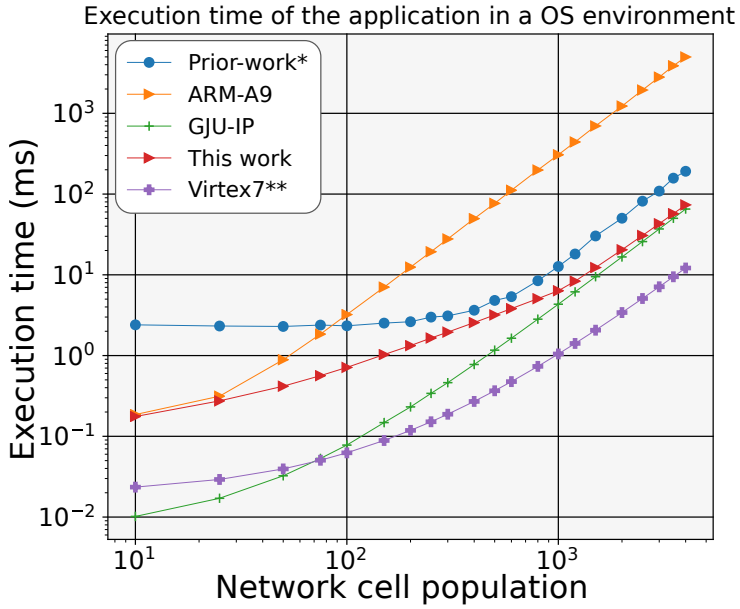

Figura 6.9: Desempeño comparando distintas implementaciones del modelo eHH para varias poblaciones.

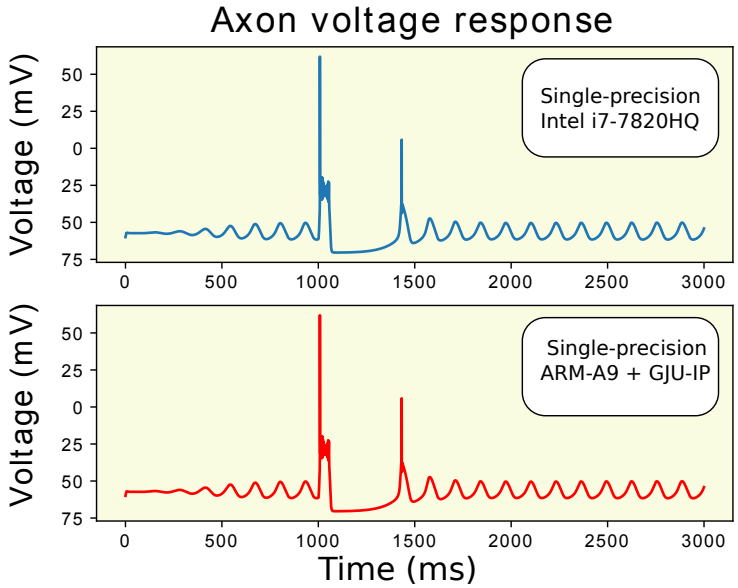

Figura 6.10: Respuesta de una neurona sencilla en el TECBrain, comparado contra el modelo áureo en C++ para una red ION eHH de neuronas conectadas todas contra todas.

6.3.4 Cuarta aproximación: variar la interconexión entre placas a una plataforma personalizada

Los resultados obtenidos en los puntos anteriores fueron prometedores, pero algunas cosas quedaron por mejorar aún para que la implementación en una sola FPGA fuera competitiva. Estos aspectos se están trabajando ya en el proyecto VIE 1360043, "Diseño de Arquitecturas Multinúcleo para Aplicaciones de Procesamiento Masivo de Datos (Big Data)", incubado a partir de los resultados de este proyecto. Además, es necesario disminuir la latencia en la intercomunicación entre los distintos nodos de procesamiento del TECBrain (que debe recordarse andan en cerca de 10ms). Ello implica orientarse hacia una plataforma de interconexión personalizada, trabajo que también se ha iniciado en el marco del proyecto anterior, con la colaboración del Ing. Ronny Ramírez García que ya ha generado la base de este marco dentro de su tesis doctoral en ingeniería.

6.4 Interfaz web de usuario

Como parte final del proyecto, era necesario crear una plataforma de visualización para las simulaciones, con el control necesario para que el usuario ajustara variables como tiempo de simulación, tipo de modelo (región del cerebro a modelar), número de neuronas, densidad de interconexión, y demás. Una primera versión de esta plataforma, desarrollada sobre un servidor web HTML se describe completamente en [23], donde se analizan además sus falencias (sobre todo, lentitud en el despliegue y recolección de datos). Se ofrece una imagen de la misma en la figura 6.11. Se nota el nivel de control existente, y cómo se despliegan tanto el mapa completo de la red neuronal, como la capacidad de visualizar la respuesta de una sola neurona a los impulsos de entrada.

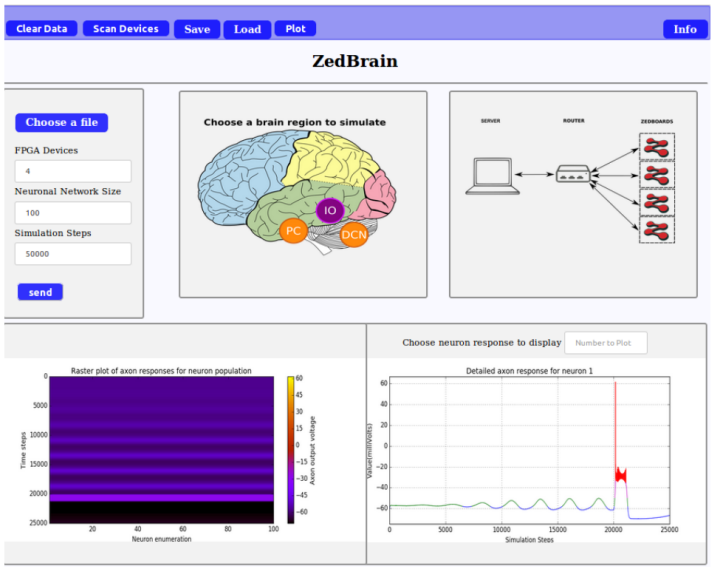

Figura 6.11: Página principal del sistema de visualización web que controla el TECBrain (en ese momento llamado ZedBrain). Imagen tomada de [23].

Ante la situación, se decidió realizar una segunda versión del sistema, montado sobre una arquitectura de recolección y archivo de datos más eficiente en MondoDB y NodeJS, que ha sido descrita con detalle en [15]. Esta arquitectura, llamada de *back-end*, es analizada en términos de rendimiento y escalabilidad en [15], y en la figura 6.12 se muestra un diagrama a bloques de cómo se integran todas las distintas unidades de procesamiento y despliegue de datos que componen el TECBrain, desde el bloque MultiFPGA en sí mismo hasta la aplicación Web de despliegue.

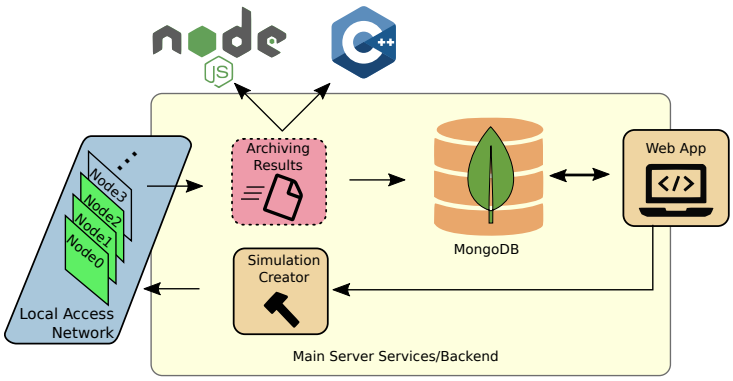

Figura 6.12: Plataforma completa integrada de configuración, manejo y despliegue de datos del TECBrain. El acceso a través de una aplicación Web permite visualizar los datos administrados via MongoDB y NodeJS. Imagen tomada de [15].

Esta estructura es clave pues ofrece una versatilidad más amigable para el usuario. La interfaz final de usuario, llamada *front-end* en la figura 6.12, sigue en proceso de diseño bajo la orientación del Dr. Franklin Hernández Castro dentro del proyecto ya mencionado "Diseño de Arquitecturas Multinúcleo para Aplicaciones de Procesamiento Masivo de Datos (Big Data)". Un avance de todo el sistema bajo funcionamiento se reporta en [24]. En la figura 6.13 se muestra la topología creada para el manejo del *front-end* (imagen tomada de [24]).

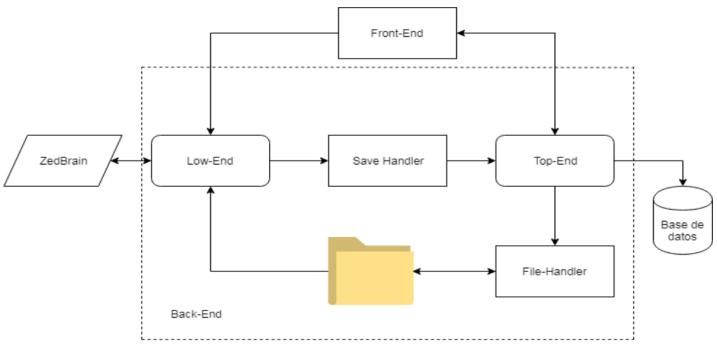

Figura 6.13: Topología usada para el manejo del *front-end* del TECBrain (originalmente llamado ZedBrain). Imagen tomada de [24].

A partir de esta estructura, y con el diseño propuesto por el Dr. Hernández Castro, se crearon los módulos de visualización que permiten primero configurar la simulación con todos los parámetros requeridos (ver figura 6.14), visualizar la respuesta completa

de la red en función del tiempo (en un mapa de calor, ver 6.15), y analizar con detalle y comparar entre varias simulaciones para una única neurona, o un juego de las mismas (ver 6.16). Se posee ahora una plataforma atractiva y útil (además de escalable a muchos modelos cerebrales) para que personas del área neurocientífica puedan aprovechar las potencialidades del TECBrain.

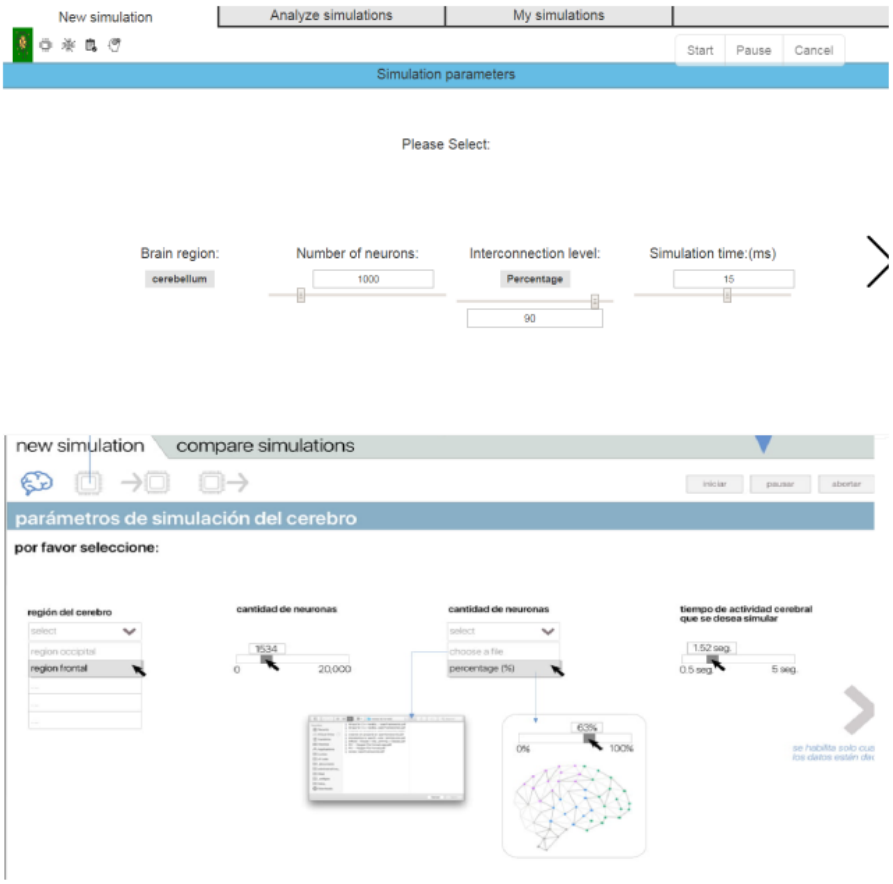

Figura 6.14: Pantallas de configuración de parámetros y control de simulaciones en el TECBrain . Imagen tomada de [24].

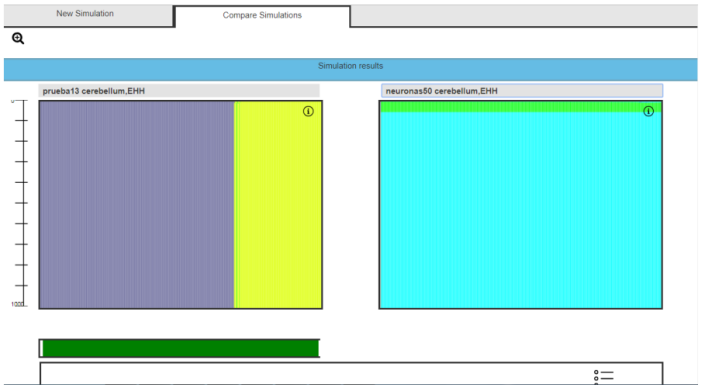

Figura 6.15: Visualización de la respuesta completa en función del tiempo de la red de neuronas. Un mapa de calor revela la intensidad de la espiga o respuesta de cada neurona. Imagen tomada de [24].

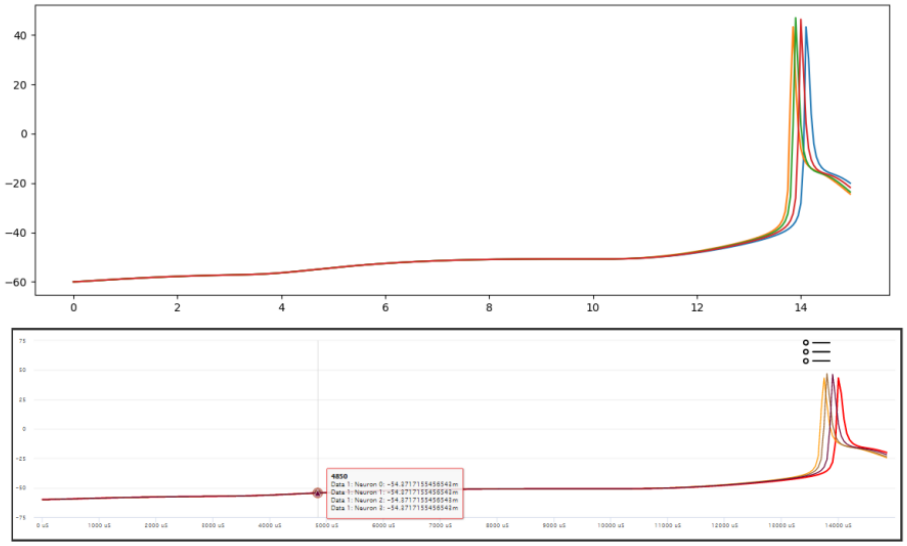

Figura 6.16: Comparación de respuesta temporal de varias neuronas. Imagen tomada de [24].

7. Conclusiones y recomendaciones

7.1 Conclusiones

Se ha mostrado el desarrollo de una plataforma multiFPGA para la aceleración de simulaciones de modelos neuronales biológicamente precisos, que en la actualidad implementa en su totalidad el modelo eHH para la región olivar cerebral, y el modelo general Izhikevich, con un error inferior al 1%, y a una velocidad equivalente a la plataformas computacionales mucho más caras. Esta plataforma, denominada TECBrain, permite actualmente interfazar hasta cuatro FPGAs a través de un sistema de control Web usable y flexible. El sistema es escalable a más FPGAs y puede incoporar distintos modelos de las mismas, además de las placas Zedboard ya existentes, incluyendo una placa propietaria basada en el Zynq 7030 con interfaces seriales de alta velocidad, que se encuentra en estado de desarrollo. Además, se ha dotado de un esquema de manejo y administración de datos a TECBrain para su posterior potencial aplicación a otros problemas de procesamiento paralelo, potencial que será aprovechado en los proyectos VIE actualmente en proceso en el DCILab y el HPCLab.

Los resultados indican que el sistema es competitivo en términos de costo y energía consumidos con respecto a ejemplos de implementaciones recientes en la literatura. Se han producido tres artículos que revelan precisamente la novedad de los resultados obtenidos en el proyecto. Uno de estos está ya publicado en el la base de datos IEEEXplore, con indexación Scopus, y los otros dos se encuentran en proceso de revisión final para su inclusión en un congreso también con memoria indexada Scopus.

Por otra parte, este proyecto ha solidificado la experticia de los investigadores participantes, y los laboratorios DCILab y HPCLab de la Escuela de Ingeniería Electrónica y las áreas de Ingeniería Mecatrónica e Ingeniería en Computadores, en áreas como procesamiento paralelo, sistemas empotrados basados en Linux, de síntesis de alto nivel de aceleradores de hardware sobre FPGA, y su integración en sistemas de cómputo heterogéneos sobre plataformas programables, interconectadas a alta velocidad. Además, se ha generado una experiencia en el rápido desarrollo de sistemas de procesamiento masivo de datos en hardware para aplicaciones complejas, dentro de una metodología de optimización de codificación de HLS orientada a sistemas FPGA.

Como ejemplo de lo anterior, se está concluyendo ya en el HPCLab un subproyecto generado a partir del TECBrain que se piensa utilizar para el monitoreo energético de granjas de paneles fotovoltaicos, en apoyo a laboratorio de Sistemas Energétivos sostenibles (SESLab). Además, existe ya el interés de una empresa de sistemas empotrados en adquirir los servicios del labotario para uno de sus proyectos de procesamiento de señales en ciernes.

7.2 Recomendaciones

Se debe generar un esquema de interconexión alternativo que se aproveche de las interfaces seriales de alta velocidad, disponibles en algunas versiones más avanzadas de la plataforma Zynq 7000. Este esquema debe dotarse de un esquema multicapa que facilite su integración con las aplicaciones de simulación de alto nivel.

Es necesario introducir al menos de manera opcional algún curso en el área de síntesis de alto nivel sobre dispositivos flexibles, tanto en la carrera de Ingeniería Electrónica, como de sus carreras hermanas: Ingeniería en Computadores e Ingeniería Mecatrónica. Ello pues es apreciable cada vez más el énfasis que se está dando a estas metodologías en el desarrollo de arquitecturas más avanzadas de procesamiento digital. Ello disminuirá el tiempo necesario de curva de aprendizaje para los estudiantes de grado y de maestría que se incorporen al HPCLab a los futuros proyectos.

Bibliografía

[1] Marco Acuña. Extensión de las capacidades de comunicación y memoria para sistema de modelado de redes neuronales del olivo inferior. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Agosto 2015.

[2] K. Alfaro-Badilla, A. Arroyo-Romero, C. Salazar-García, Luis León-Vega, J. Espinoza-González, Franklin Hernández-Castro, A. Chacón-Rodríguez, G. Smaragdos, y C. Strydis. Improving the simulation of biologically accurate neural networks using data flow HLS transformations on heterogeneous SoC-FPGA platforms. In *2019 Latin American Conference on High Performance Computing, CARLA*, Aug 2019.

[3] K. Alfaro-Badilla, A. Chacón-Rodríguez, G. Smaragdos, C. Strydis, A. Arroyo-Romero, J. Espinoza-González, y C. Salazar-García. Prototyping a biologically plausible neuron model on a heterogeneous CPU-FPGA board. In *2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS)*, págs. 5–8, Feb 2019.

[4] Y. Arias-Esquivel et al. Real-time vibration analysis for structure fault detection. In *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*, págs. 1–4, Nov 2016.

[5] Kaleb Alfaro Badilla. Diseño de un acelerador de hardware para simulaciones de redes neuronales biológicamente precisas utilizando un sistemamulti-fpga. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Agosto 2017.

[6] A. Cervantes, F. Lopez, J. Quiros, D. Rodriguez, C. Salazar-Garcia, C. Meza, y A. Chacon-Rodriguez. Implementation of an open core ieee 754-based fpu with nonlinear arithmetic support. In *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*, págs. 1–6, Nov 2016.

[7] A. Chacon-Rodriguez, P. Julian, y F. Masson. Fast and low power integrated circuit for impulsive sound localisation using kalman filter approach. *Electronics Letters*, 46:533–534, April 2010.

[8] A. Chacon-Rodriguez, F. Martin-Pirchio, S. Sanudo, y P. Julian. A low-power integrated circuit for interaural time delay estimation without delay lines. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 56(7):575–579, July 2009.

[9] A Chacón-Rodriguez, FN Martín-Pirchio, P Julián, y PS Mandolesi. A verilog hdl digital architecture for delay calculation. *Latin American applied research*, 37(1):41–45, 2007.

[10] Alfonso Chacón-Rodríguez y José Pablo Alvarado-Moya. Sistema electrónico integrado en chip (SoC) para el reconocimiento de patrones de disparos y motosierras en una red inalámbrica de sensores para la protección ambiental. 2014.

[11] Alfonso Chacón-Rodríguez. *Circuitos Integrados de Bajo Consumo de Potencia para Detección y Localización de Disparos de Armas de Fuego*. PhD thesis, Universidad Nacional de Mar del Plata Argentina, Argentina, 2009.

[12] Johannes de Fine Licht, Simon Meierhans, y Torsten Hoefler. Transformations of high-level synthesis codes for high-performance computing. *CoRR*, abs/1805.08288, 2018. URL http://arxiv.org/abs/1805.08288.

[13] AVNET electronics marketing. Hardware user's guide- zedboard(zynq evaluation and development), 2014. URL http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf.

[14] Javier Espinoza González. Diseño y optimización de una estrategia de mejora para una red neuronal biológicamente precisa, basada en el modelo hodgkin huxley extendido. Tesis de Licenciatura, Área de Ingeniería en Computadores, ITCR, Cartago, Costa Rica, Junio 2018.

[15] L. G. León-Vega, K. Alfaro-Badilla, A. Chacón-Rodríguez, y C. Salazar-García. Optimizing big data network transfers in FPGA SoC clusters: TECBrain case study. In *2019 Latin American Conference on High Performance Computing, CARLA*, Aug 2019.

[16] W. Maass. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. *Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons*, pág. 211–217, 1996.

[17] F. Martin-Pirchio, A. Chacon-Rodriguez, P. Julian, y P. Mandolesi. A comparison of low power architectures for digital delay measurement. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, págs. 498–499, March 2007.

[18] FN Martin Pirchio, Pedro Julián, Pablo Sergio Mandolesi, y Alfonso Chacon-Rodriguez. An adaptive cross-correlation derivative algorithm for ultra-low power time delay measurement. In *2007 IEEE International Symposium on Circuits and Systems*, págs. 4016–4019. IEEE, 2007.

[19] Andrés Arroyo Romero. Codiseño software/hardware de un simulador neuronal basado en el modelo Hodgkin Huxley extendido. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2019.

[20] C. Salazar-García, L. Alfaro-Hidalgo, M. Carvajal-Delgado, J. Montero-Aragón, R. Castro-Gonzalez, J. A. Rodríguez, A. Chacon-Rodríguez, y P. Alvarado-Moya. Digital integrated circuit implementation of an identification stage for the detection of illegal hunting and logging. In *2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS)*, págs. 1–4, Feb 2015.

[21] C. Salazar-García, R. Castro-González, y A. Chacón-Rodríguez. RISC-V based sound classifier intended for acoustic surveillance in protected natural environments. In

2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS), págs. 1–4, Feb 2017.

[22] Georgios Smaragdos, Sebastian Isaza, Martijn F. van Eijk, Ioannis Sourdis, y Christos Strydis. FPGA-based biophysically-meaningful modeling of olivocerebellar neurons. *FPGA*, págs. 89–98, 2014.

[23] Daniel Zamora Umaña. Desarrollo y validación de un método para la visualización de resultados en la implementación del algoritmo de simulación de redes neuronales. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Agosto 2017.

[24] Keylor Mena Venegas. Desarrollo de plataforma web para control y recolección de datos en dispositivos embebidos. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2019.

[25] G. Wulfram y W. Werner. *Spiking Neuron Models*. Cambridge University Press, 2002.

[26] Xilinx. Bringing ultra high productivity to mainstream systems and platform designers. Technical report, 2015. URL https://www.xilinx.com/support/documentation/backgrounders/vivado-hlx.pdf.

[27] Xilinx. Field programmable gate array (fpga), 2018. URL https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.

[28] Xilinx. Zynq-7000 all programmable soc: Embedded design tutorial a hands-on guide to effective embedded system design. Technical report, 20186.

[29] Amir Zjajo, Jaco Hofmann, Gerrit Jan Christiaanse, Martijn Van Eijk, Georgios Smaragdos, Christos Strydis, Alexander De Graaf, Carlo Galuzzi, y Rene Van Leuken. A Real-Time Reconfigurable Multichip Architecture for Large-Scale Biophysically Accurate Neuron Simulation. *IEEE Transactions on Biomedical Circuits and Systems*, 12(2):326–337, 2018.