CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE BIOELECTRÓNICA

Diseño e implementación de un sistema de adquisición para registro de ECoG en rata, con lesión cortical

Tesis que presenta:

Andrés Francisco Morín Sánchez

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis: Dr. Daniel Lorias Espinoza

Ciudad de México febrero 2020

Agradecimientos

Dedico este trabajo a la memoria del Dr. David Elías Viñas, mi maestro y amigo. Con sus sabios consejos he logrado alcanzar muchas de las metas que me propuse al iniciar esta ruta en mi preparación académica. A él, mi más profundo respeto y admiración. “En donde quiera que se encuentre, con mucho cariño, gracias Doctor”.

A mi esposa y a mis hijos que me han apoyado siempre, en cada momento. “Ericka Claudia, tú y nuestros hijos son las personas más importantes en mi vida, gracias por los largos ratos que nos hemos pasado estudiando juntos”.

Agradezco el gran apoyo que me brindaron mis asesores, el Dr. Daniel Lorias Espinoza y el Dr. Francisco Estrada Rojo, que ante la terrible pérdida del Dr. David se mostraron muy solidarios para ayudarme a concluir este proyecto, favorablemente.

Agradezco a todos los miembros del laboratorio 12, trabajadores y alumnos. En particular a Esteban Ruiz Hernández por sus consejos y sugerencias, y a Alfonso Márquez Robles que con su valioso trabajo de maquinado de piezas y gabinetes se logró el realce estético del instrumento desarrollado.

En general agradezco a la Sección de Bioelectrónica del Cinvestav, profesores, personal de apoyo, y alumnos, que de algún modo han contribuido a la realización de este trabajo y a mi formación académica.

Agradezco al CONACYT por el apoyo económico otorgado al programa de Maestría en Ciencias en Ingeniería Eléctrica.

Diseño e implementación de un sistema de adquisición para registro de ECoG en rata, con lesión cortical

Contenido

1. Resumen

El traumatismo craneoencefálico TCE, es una condición clínica que acontece cuando se ha liberado una gran cantidad de energía sobre la masa encefálica, provocando lesiones que pueden ir desde leves hasta muy severas, que si no son atendidas de inmediato el paciente traumatizado puede sufrir consecuencias graves de carácter permanente e incluso la muerte

Es por ello que resulta necesario implementar mecanismos de ayuda para la pronta recuperación de los pacientes que sufren de este problema. Los esfuerzos que se han hecho son muchos, pero todavía faltan bastantes cosas por conocer y la investigación en estos campos logra que se mejoren los mecanismos de atención, haciéndolos más confiables cada vez. Por ello es que surge la necesidad de desarrollar nuevos instrumentos que nos muestren un panorama más amplio de las respuestas fisiológicas del tejido dañado. Se desarrolló un registrador de señales de ECoG para ayudar a la investigación en la utilización de fármacos para minimizar el daño tisular alrededor del foco del trauma. Se implementó un sistema electrónico modular que puede tener hasta 16 canales de captura de datos acondicionando las señales analógicamente, para obtener un rango de voltaje de 0 a 4.096 V y luego digitalizarlas por medio de un convertidor simultaneo. Además, se implementó un sistema de adquisición de los datos por medio de un módulo de desarrollo que tiene como circuito principal el SoC FPGA Cyclone V de la marca Intel. También se diseñó una interfaz gráfica, para poder visualizar los datos en un monitor de computadora conectado al propio sistema y contar con un sistema operativo que contiene un escritorio, desde el cual se lanza la aplicación del registrador. El resultado fue un instrumento autónomo que permite la visualización de bioseñales, el cual, puede ser fácilmente configurado para la observación de otros biopotenciales e incluso con alguna adaptación sencilla puede ser empleado para dar estimulación eléctrica a tejidos in vivo.

2. Abstract

TBI cranial trauma is a clinical condition that occurs when a large amount of energy has been released on the brain mass, causing injuries that can range from mild to very severe that if not treated immediately, the traumatized patient can suffer serious consequences of permanent character and even death. That is why it is necessary to implement support mechanisms for the speedy recovery of patients suffering from this problem. The efforts that have been made are many, but there are still many things to know and research in these fields improves the mechanisms of care, making them more reliable every time. Therefore, there is a need to develop new instruments that show us a broader picture of the physiological responses of damaged tissue. An ECoG signal recorder was developed to assist research in the use of drugs to minimize tissue damage around the focus of the trauma. A modular system that can have up to 16 channels of data capture was implemented by conditioning the signals analogically to obtain a voltage range of 0 to 4,096 V and then digitizing them by means of a simultaneous converter. In addition, a data acquisition system was implemented through a development module whose main circuit is the FPGA Cyclone V SoC of the Intel brand. A graphic interface was also designed to be able to visualize the data on a computer monitor connected to the system itself and to have an operating system that contains a desk from which the recorder application is launched. The result was an autonomous instrument that allows the visualization of biosignals, which can be easily configured for the observation of other biopotentials and even with some simple adaptation it can be used to give electrical stimulation to tissues in vivo.

3. Introducción

3.1. Traumatismo Craneoencefálico (TCE)

El TCE es una condición clínica que se establece inmediatamente después de un impacto al cerebro, producto de una agresión proveniente del exterior como es un golpe en la cabeza, una aceleración o desaceleración abrupta, provocada por una explosión, incluyendo en ocasiones, la introducción de material externo en la cavidad encefálica, en resumen es: “Cualquier lesión física o deterioro funcional del contenido craneal secundario a un intercambio brusco de energía” (1). Puede ser clasificado de varias maneras, la OMS y la mayoría de las sociedades científicas realizan la clasificación basándose en la afectación del nivel de conciencia cuantificada a través de la escala de coma de Glasgow clasificándolo en Leve (Glasgow 13-15), TCE Moderado (Glasgow 9-12) y TCE Grave (Glasgow <8) (2).

En México es la cuarta causa de muerte, teniendo su mayor índice en los accidentes de tráfico, y en menor medida las caídas y accidentes deportivos. Es por esta razón que la atención pronta y expedita de los servicios de emergencia es un importante factor para la recuperación de las personas que sufren este tipo de lesión. Varias líneas de investigación se han derivado de este problema alrededor del mundo y nuestro país está contribuyendo de manera importante al mejoramiento de los procedimientos y terapias adecuadas para minimizar los efectos derivados de un TCE. Sin embargo, todavía hay mucho por hacer.

La fisiopatología del TCE identifica dos tipos de lesiones, una primaria y posteriormente una secundaria. La primera tiene un período de ocurrencia muy corto y es la que establece las condiciones iniciales de la lesión. Inicia justo en el momento del evento traumático y termina una vez que el tejido encefálico ha alcanzado un

estado de pseudo estabilidad. Durante este período, existe un daño que es irreversible porque implica desgarramiento de tejidos, contusión hematomas, edema. Por su parte la lesión secundaria es potencialmente reversible y durante esta etapa vale la pena aplicar las técnicas de rehabilitación conocidas como neuro protección. A partir la lesión primaria se desencadenan mecanismos endógenos que son protectores del tejido que queda sano pues se liberan sustancias que promueven la atracción de leucocitos y plaquetas que ayudan a la reparación de la zona dañada. Sin embargo, en el cerebro, la permanencia de estas condiciones es contraproducente, generando otro tipo de problema al provocar la liberación excesiva del neurotransmisor glutamato y la sobreproducción de radicales libres como Ca^{++}, Na^{+} y K^{+}, además de un proceso inflamatorio. Si esta condición permanece por largo tiempo, puede degenerar en una hiper excitación de las neuronas debido al exceso de glutamato (exocitosis) y radicales libres, desencadenando un ingreso masivo de Ca^{++}, lo que puede derivar en apoptosis y necrosis. El proceso inflamatorio favorece la liberación de más glutamato y la permanencia del edema, lo que provoca muerte celular.

El conocimiento de estos mecanismos, así como del cambio en la actividad eléctrica del cerebro que estos provocan, permitirá el desarrollo de blancos terapéuticos y farmacológicos que permitan genera neuroptotección (3).

3.2. Experimentación en modelo animal de pequeños roedores

Para poder entender los mecanismos que subyacen al presentarse un daño en el cerebro, se han desarrollado diversos modelos de TCE, la gran mayoría de ellos se han desarrollado en animales de experimentación.

Dada su enorme capacidad de reproducción y aclimatación a condiciones de laboratorio y su enorme similitud fisiológica con el humano, los roedores se han convertido en excelentes modelos. Gracias a ello, se pueden tener resultados muy

cercanos a los que se obtendrían al experimentar con humanos. Por otro lado, la experimentación es mucho menos costosa sobre todo en las fases iniciales, ya que en estos casos se tiende a desechar muchas pruebas por no tener resultados favorables, y el costo de algunos fármacos se eleva a cientos de dólares estadounidenses por cada miligramo. Se sabe que la dosis que se debe administrar tanto para provocar la enfermedad como la posible curación depende principalmente de la masa corporal, por lo que, trabajar con estas especies es muy buena opción (4).

Otra razón por la que se trabaja con rata en el laboratorio es que el modelo se tiene perfectamente estudiado y se cuenta con Atlas muy completos (5), que describen las regiones cerebrales en lugares muy focalizados, de modo que el modelo pueda ser reproducido fácilmente por otros laboratorios para la corroboración de los experimentos.

3.2.1. Modelos experimentales del TCE.

El diseño de un modelo experimental de TCE es un reto para los investigadores. Estos modelos deben reproducir las lesiones cerebrales traumáticas observadas en la práctica clínica, con la finalidad de poder cuantificarlos y ser reproducibles entre diferentes investigadores. Un modelo es factible cuando cumple los siguientes criterios: (I) la fuerza mecánica usada para inducir el daño es controlable y cuantificable; (II) la lesión cerebral es reproducible y simula adecuadamente la observada en el daño cerebral humano y (III) existe un rango de gravedad del daño (leve, moderado, grave) que puede medirse con parámetros morfológicos, fisiológicos, bioquímicos o de conducta. Actualmente la mayoría de los estudios usan protocolos estándar que incluyen el uso de animales sham (sin daño) a los que se les realiza todo el procedimiento quirúrgico excepto el trauma y que sirven para controlar el efecto en las variables sistémicas de la anestesia, la técnica quirúrgica o la sujeción de la cabeza en un marco estereotáxico. La importancia de los modelos animales es conseguir que

la lesión inducida sea reproducible y que haya una mínima variabilidad, por ello se debe monitorizar y controlar estrictamente todas las variables fisiológicas que puedan modificar el efecto del daño cerebral traumático y que por lo tanto pueden influir en la lesión tisular final. Las variables que se deben mantener en el rango normal a lo largo de todo el experimento son la temperatura, presión arterial, glucemia y gases arteriales. Por supuesto, todos los animales incluidos en un estudio experimental deben ser del mismo sexo y pertenecer al mismo grupo de edad.

3.3. Procedimientos y análisis

El procedimiento habitual para la experimentación de los distintos fármacos es inducir el daño en el modelo animal de manera sistemática y bajo las mismas condiciones en todos los eventos. De esta manera se reduce al mínimo las variaciones debidas a los cambios que puedan suceder al momento del TCE. Son seleccionados los ejemplares por peso y tamaño y se habitúan al ambiente en el que serán sometidos durante algunos días para no generar variables de estrés o algunas otras condiciones psicológicas. Todos los ejemplares son tratados bajo las normas oficiales mexicanas para tratamiento de animales de laboratorio. Para el caso de TCE se ha tomado como tejido Diana la zona que corresponde a la corteza cerebral motora que se encuentra en las coordenadas P=2 y L=1.4 por medios mecánicos con un pistón que genera una presión de 40 libras en el punto de impacto. El grupo es dividido en dos, una mitad es empleada para las pruebas de control y la otra para la inducción del daño. A su vez cada mitad se subdivide en más grupos para realizar las diferentes pruebas de fármacos, incluyendo una muestra con aplicación de placebo. Todos los animales son sometidos a exámenes de escala conductual de daño neurológico.

Finalmente se procede a realizar un análisis histológico de los distintos grupos conservando los cerebros de todos los animales y cortándolos en rebanadas con un

espesor de 20 µm y tiñéndolos de modo que pueda verse el daño en los animales sometidos al TCE comparativamente con los grupos de control (6).

3.4. Registro electrofisiológico

A pesar de que los análisis histológicos proporcionan un conocimiento importante al respecto de los efectos que provoca la aplicación de fármacos y se puede observar de manera directa y clara el estado del tejido nervioso después de un evento de TCE, para poderlo realizar es necesario sacrificar al animal. Se tiene que seguir una serie de pasos sistemáticos para que los resultados no sean afectados por el mal trato de las muestras, además de la posible contaminación pues se requiere sacrificar a todo el grupo, y como se lleva a cabo en un mismo lugar, los últimos especímenes sufren de una condición de estrés que puede afectar las muestras que se desean preparar. Por otro lado, el análisis conductual con evaluación de nivel de daño neuronal es muy subjetivo, pues depende estrictamente del criterio que tenga el evaluador para determinar si el individuo presenta una mejoría en su comportamiento. Aún que las pruebas se realizan todas bajo las mismas condiciones, es necesario llevar a cabo la observación durante todo el tiempo que se haga la prueba, y a pesar de que se cuente con un sistema de videograbación, para hacer el análisis se tendrá que revisar toda la grabación, lo cual lleva demasiado tiempo, y sigue estando el factor subjetivo del observador.

Por ello es necesario tomar otro tipo de registro que no sea tan subjetivo como el análisis conductual y tampoco se tenga que esperar a sacrificar al animal para hacer el análisis. El sistema que permite obtener esta información es el registro electrofisiológico. Tratándose de una lesión que se ubica en la corteza cerebral como es la provocada por un TCE, el tipo de registro a tomar en cuenta es el electrocorticograma (ECoG) que es un sistema que nos proporciona información

acerca del estado de salud de las células corticales y las vías de conexión con las regiones talámicas. Son los ritmos cerebrales los que nos permiten obtener información acerca de las funciones de control que realiza la corteza cerebral sobre el sistema motor del individuo (7). Si por alguna causa se ha interrumpido la comunicación entre ellos, ya sea por una lesión en las vías que los interconectan o directamente en uno de estos elementos, entonces el control motor se disloca. Durante un TCE la región que resulta lesionada es la que realiza justamente el control, es decir la corteza cerebral, específicamente cuando sucede en la región motora, el individuo pierde el control de sus habilidades motoras, por lo que resulta de gran interés contar con los registros de ECoG para tener pleno conocimiento de los que está sucediendo con el tejido que se encuentra en condición isquémica y si la aplicación de fármacos neuro-protectores tiene efectividad en la pronta recuperación.

4. Objetivos

4.1. Objetivo general

Con este trabajo se pretende desarrollar un sistema de adquisición de datos, que sea capaz de registrar señales de electrocorticografía (ECoG), para apoyar en la generación de nuevo conocimiento en el estudio de la neuroprotección

4.2. Objetivos específicos

4.2.1. Desarrollo del instrumento.

Diseñar la electrónica necesaria, empleando técnicas de adquisición de datos para señales de ECoG, cuyo ancho de banda se encuentra dentro del rango de 0.1 a 70 Hz y una amplitud de 5 a 200 µV.

4.2.2. Calibración.

Una vez que se ha obtenido el diseño del instrumento, lograr la calibración de este, para cumplir con los rangos determinados y almacenar los datos en un archivo de datos, compatible con las aplicaciones computacionales más convencionales.

4.2.3. Adecuación para el estudio con rata.

Lograda la calibración, preparar la interfaz necesaria, para conectar los electrodos implantados en la rata, y así permitir el registro de ECoG esperado.

5. Antecedentes

Son muchos los trabajos que se han realizado para la identificación de características en patologías del cerebro como la epilepsia y el Parkinson, pero con respecto al TCE es poca la investigación que se ha hecho, por lo que resulta difícil saber que se debe esperar de los resultados obtenidos con el instrumento que se está desarrollando. En los últimos 10 años se han realizado experimentos que han dado alguna luz sobre los efectos producidos por el TCE y que han tenido relación con algunos de los síntomas de cuadros epilépticos, como las tormentas eléctricas que se presentan en pacientes con lesión cortical aguda.

El electrocorticograma (ECoG) intraoperatorio permite el registro de la actividad eléctrica cortical, mediante la colocación de electrodos directamente sobre la superficie de la duramadre. Desde su desarrollo en la década de 1950 por Penfield y Jasper en el Instituto Neurológico de Montreal, el ECoG se ha utilizado para identificar zonas epileptogénicas durante el tratamiento quirúrgico de pacientes con epilepsia severa, desde su introducción la identificación de la zona de descarga epileptiforme interictal contribuyó sustancialmente a reducir el número de exploraciones negativas.

El registro ECoGráfico se lleva a cabo con electrodos colocados en la corteza expuesta. En este punto el cirujano realiza una craneotomía, remueve una parte del hueso para exponer la superficie cerebral. En la rata este procedimiento se realiza con anestesia general. Los electrodos pueden ser de plata o acero inoxidable y se colocan encima de la duramadre (epidurales). La banda de frecuencia utilizada para el registro es de 0.5-70Hz y la sensibilidad de 30-50uV/mm. La anestesia es considerada como una desventaja en los registros del ECoG, la anestesia general induce abundante

actividad rápida en el rango de frecuencia de 15-30Hz y la actividad ECoGráfica puede observarse con disminución de la amplitud en el área de la exploración. Algunas técnicas incluyen la electroestimulación, y en humano se ha encontrado correlación entre los síntomas inducidos por la estimulación eléctrica de las estructuras temporales mesiales y las auras habituales, en pacientes con epilepsia del lóbulo temporal (ELT) sometidos a lobectomía temporal. Otra importante utilidad de la estimulación eléctrica intraoperatoria es la evaluación de las áreas corticales involucradas en las funciones de memoria, brindando información sobre los mecanismos de la memoria y contribuyendo a evitar los déficits de memoria posquirúrgicos (8).

En un trabajo desarrollado por Leão en 1944 se describió un fenómeno conocido como Cortical Spreading Depression (CSD) y es “un artefacto obtenido al intentar inducir crisis epilépticas mediante estimulación eléctrica en el córtex frontal del conejo”. Las CSD afectan al sistema nervioso central (SNC) y se caracteriza por una despolarización de la actividad sináptica y por la propagación de la onda de despolarización por la sustancia gris cortical acompañada de una redistribución masiva de los iones a nivel intra y extracelular. La característica fundamental de este fenómeno es su propagación por el neocórtex en forma de onda expansiva lenta.

En el ECoG la CSD se detecta por una depresión súbita de la amplitud del registro eléctrico que se propaga sobre el córtex a manera de mancha a una velocidad de 1-5 mm por minuto. Otro fenómeno llamado CSD-like tiene unas características similares a la CSD clásica, pero se inicia de forma espontánea en las áreas de penumbra isquémica. Los fenómenos CSD-like son frecuentes en el córtex adyacente a las

lesiones focales traumáticas y casi constantes en las áreas de penumbra isquémica en pacientes con infartos malignos de la arteria cerebral media (9). En la figura 1 se muestra un ECoG con la presencia del fenómeno CSD.

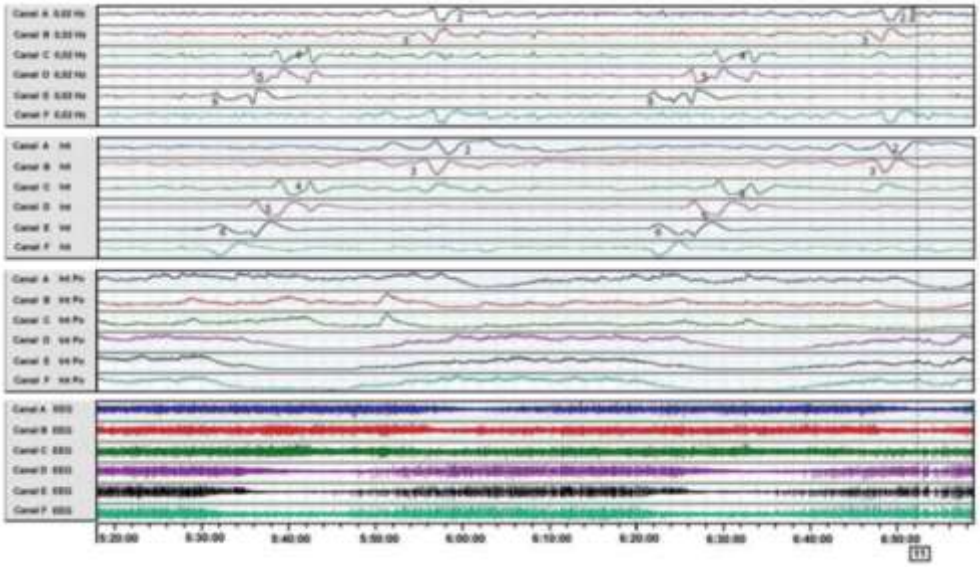

Fig. 1 ECoG con presencia de CSD.

6. Desarrollo del instrumento de medición

Para la primera fase se propone implementar un sistema de registro para señales que se encuentran en el orden de los 10 µV hasta 400 µV con un ancho de banda de 0.1 a 300 Hz[8]. El rango dinámico que se empleará es de 80 dB, de modo que se requiere de un convertidor analógico a digital con una resolución que está por encima de 12 bits por lo que el valor comercial a emplear será de 16 bits. Como la ganancia requerida para el amplificador es de 10000 se debe tener especial cuidado con respecto al ruido. Una manera de resolverlo es dividiendo la ganancia en dos etapas, la primera comprende un amplificador diferencial con salida única. Para esta etapa la ganancia quedará establecida en 1000, y se pretende utilizar un amplificador de instrumentación de alta precisión y de muy bajo ruido para lograr que la resolución del convertidor pueda discriminar el ruido que provenga de la amplificación. La siguiente etapa de amplificación estará formada por un amplificador operacional convencional en configuración no inversora con ganancia 10. Se puede aprovechar para implementar la etapa de filtrado de segundo orden, implementando un pasa bajos de topología Sallen Key estableciendo una frecuencia de corte de 300 Hz. Finalmente se utilizará un amplificador de aislamiento de ganancia unitaria para proveer la protección adecuada al paciente y al personal que llevará a cabo las pruebas. El circuito descrito anteriormente corresponde a la implementación de un solo canal para el registro de ECoG. Como se hará la prueba con un total de cuatro canales para la primera fase del proyecto, es necesario replicar tres veces más esta sección. Todos estos canales corresponden a la sección analógica, por lo que los cuidados que se deberán tener en cuenta para eliminar el ruido tendrán que ver con las técnicas minimización para el procesamiento analógico.

Para la adquisición de datos se empleará un convertidor analógico a digital de 16 bits de ocho canales simultáneos cuya frecuencia de muestreo será de 1 KHz por canal.

Este dispositivo estará controlado por un FPGA que servirá de interfaz para la parte del procesamiento digital de la señal. Los datos se mostrarán en un monitor convencional.

Debido a que la tecnología del FPGA desarrollada por el fabricante Intel, está formada por dos niveles de programación, es indispensable emplear dos plataformas para el desarrollo del sistema de adquisición. Por un lado, se tiene la parte del circuito digital que controlará al convertidor y adquirirá los datos y debe ser sintetizada por medio de un lenguaje de descripción de hardware. La siguiente etapa digital estará formada por un procesador que se encuentra integrado en el FPGA y que puede ejecutar un sistema operativo basado en Linux. Esta plataforma crea un ambiente gráfico en el que se pueden ejecutar aplicaciones por medios de GUI's, por lo que se deberá programar esta aplicación con una plataforma basada en lenguaje C++.

Se planea diseñar un soporte que contenga todos los electrodos necesarios junto con los electrodos de referencia, y que serán colocados en el cráneo de la rata que deberá ser sometida a una intervención quirúrgica. El hecho de colocar los electrodos en un soporte es para que dicha operación resulte sencilla. Una de las opciones que se tiene es diseñar una pieza impresa en 3D con plástico PLA, por ser biocompatible. Se deberá tomar en cuenta las consideraciones necesarias para evitar que este material afecte la prueba, debido a que todo material implantado a pesar de ser biocompatible produce inflamación (8). Por un lado, se busca desarrollar una matriz de electrodos que abarquen la zona M1 del cerebro de la rata que corresponde a la región cortical del control motriz, de acuerdo con los atlas que se tienen para la correcta localización de ella. También se buscará incluir los electrodos atornillables para cumplir con dos propósitos al mismo tiempo, en primer lugar, se espera fijar el soporte al cráneo del roedor y en segundo término permitir que la cirugía se ejecute en un tiempo corto para afectar lo menos posible las condiciones de la prueba.

En una fase posterior al desarrollo del proyecto se buscará que sea el propio sistema quien lleve a cabo un procesamiento características de las señales registradas y con ello identificar las condiciones clínicas de los diferentes casos estudiados. Para este propósito se propone hacer un análisis en frecuencia buscando principalmente las alteraciones en las oscilaciones cerebrales. Se sabe con certeza que algunas patologías neurológicas como las enfermedades de Parkinson, la Epilepsia y la Esquizofrenia producen alteraciones en diferentes regiones de la banda de frecuencias, siendo muy específicas para cada enfermedad, de manera que se piensa que se puede llegar a determinar un patrón de oscilaciones anormales para el TCE.

7. Diseño del “hardware”

7.1. Canal analógico

El circuito que describe a continuación comprende toda la parte analógica y corresponde a un solo canal, por lo que el equipo contendrá un número de circuitos iguales a este, conforme a la cantidad de canales que requiera la aplicación. El canal está formado por cinco etapas para entregar una señal que varía en el rango de ± 4.096 V.

La primera etapa consiste en un filtro diferencial pasivo pasa altos, que tiene como propósito eliminar la componente de corriente directa debida al potencial de electrodo y las bajas frecuencias que se producen cuando se conecta la interfaz y los electrodos. Estos artefactos son interferencias que pueden saturar el amplificador muy rápidamente, por lo que el filtro permitirá mantener el nivel basal de la señal. Se emplean resistencias de 5 MΩ con tolerancia de ±0.1% y capacitores de tantalio de 0.33 µF a ±10%. La frecuencia de corte es de 96.5 mHz, y está definida por los valores de los componentes que se muestran en la figura 2, y debido a que es un filtro simétrico la ecuación nos entrega un polo y un cero con constante de tiempo T=RC (9).

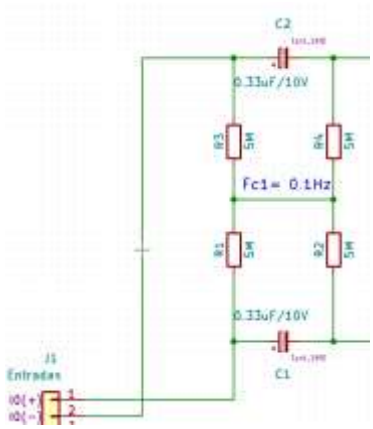

Fig. 2 Diagrama del circuito de entrada

La segunda y tercera etapa es la de amplificación y refuerzo del filtro pasa altos. Como se muestra en la figura 3, se empleando un amplificador de instrumentación AD620, por su alto CMRR y la buena inmunidad al ruido que posee. La ganancia que se requiere en esta etapa es de 1000, por lo que la resistencia R_g debe ser de 50 Ω, conforme a lo establecido en la hoja de datos del dispositivo.

$$R_g = \frac{49.4\ k\Omega}{G-1} = 49.5\ \Omega$$

Además, se cuenta con una etapa de filtrado pasa altos generado por el integrador invertido que regresa una parte de la señal de salida a la terminal de referencia. Como el integrador proporciona un polo que se agrega a la entrada no inversora del amplificador diferencial el resultado agrega también un cero en el origen reforzando el efecto del filtro pasa altos de la entrada (9), convirtiéndolo en un filtrado de segundo orden, y como la frecuencia de corte es exactamente la misma el comportamiento final corresponde al de un filtro de tipo Chebichev con una pendiente de atenuación de -40 dB/Década.

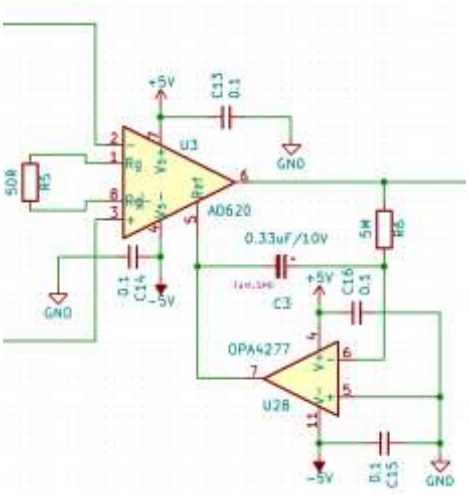

Fig. 3 Etapas de amplificación y filtrado pasa altos.

La cuarta etapa del canal analógico está formada por un filtro pasa bajos de segundo orden de topología Sallen Key, (fig. 4). Su frecuencia de corte se estableció en 239 Hz. Además del filtrado esta etapa proporciona la segunda ganancia, que es de 10 para obtener una ganancia total de 10000.

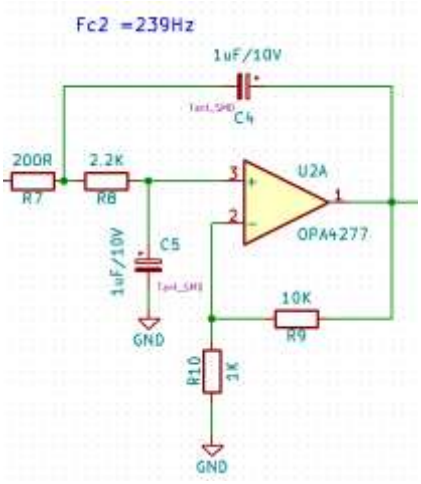

Fig. 4 Filtro de segundo orden pasa bajos Sallen Key.

La última etapa del canal analógico que se muestra en la figura 5, es un amplificador de aislamiento ISO124 de la marca Burr Brown. Se alimenta con dos fuentes de ± 5 V. La primera proporciona la alimentación de la salida que va conectada al convertidor analógico a digital. Esta parte del circuito analógico es la única que va conectada a esta fuente de alimentación. La otra fuente alimenta la sección aislada para proveer seguridad tanto al sujeto de estudio como al personal que manipulará el equipo. Este amplificador tiene ganancia unitaria por lo que no modificará las características de la señal capturada.

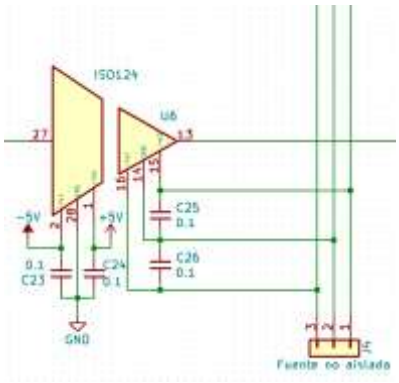

Fig. 4. Amplificador de aislamiento

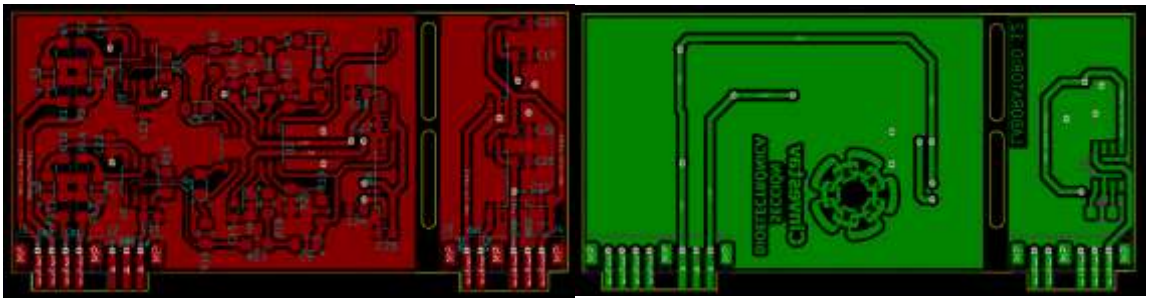

Fig. 6 Diseño del PCB para el módulo analógico.

La figura 6 muestra el diseño del circuito impreso para el módulo analógico. Esta planeado para que se inserte en las ranuras de la tarjeta madre activando dos canales por cada módulo.

Por su parte, la fuente que proporciona la alimentación a la parte aislada es un módulo convertidor CD-CD NMA0505SC de la marca muRata PS (fig. 7), el cual cuenta con un voltaje de aislamiento de 1000 V presentando una impedancia para dicho voltaje de 10 GΩ. La alimentación que requiere es de 5 V y su salida es ±5V proporcionando una corriente de 100 mA.

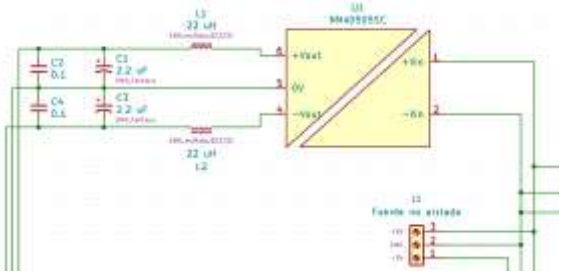

Fig. 7 Convertidor DC-DC

7.2. Sistema de adquisición

Una vez procesada la señal analógicamente para obtener un voltaje que varía en el rango de ± 4.096 V se convierte a formato digital con un rango dinámico de 96 dB, es decir que la resolución que se está empleando es de 16 bits. Este proceso se lleva a cabo por medio del convertidor Analógico-Digital AD7761. Es un convertidor analógico a digital de muestreo simultáneo de 8 canales con un modulador y filtro digital por canal, que permite el muestreo sincronizado de CA y señales de CC. Alcanza el rango dinámico requerido para esta aplicación y su ancho de banda de entrada es suficientemente bueno. El dispositivo se puede configurar en cualquiera de dos modos: Modo de control de pin; y Modo de control SPI. En el encendido, el estado del pin PIN / SPI determina el modo utilizado. Inmediatamente después del encendido, el dispositivo reconoce el estado de dicha terminal y comienza su operación en el modo reconocido. Para nuestro propósito se emplea el modo de control de pin.

7.2.1. Modo de control de PIN.

El modo de control de pin elimina la necesidad de una interfaz de comunicación SPI. El control de pin ofrece un subconjunto de funcionalidades de ajuste fácil y garantiza un estado de funcionamiento conocido después del encendido, un reinicio o una condición de falla en la fuente de alimentación. Después de seleccionado el modo de operación se establece el formato de salida de los datos a través de las terminales denominadas FORMATx. Como se busca que las señales se entreguen de manera

independiente, se fijó el valor de estos pines en valor digital 00 para que las salidas del convertidor correspondan una a una con las entradas analógicas. En la figura 8 se muestra el formato de entrega de las señales, en donde se puede observar que el dispositivo entrega los 8 datos digitales correspondientes a los 8 canales analógicos en formato serial, pero de manera concurrente

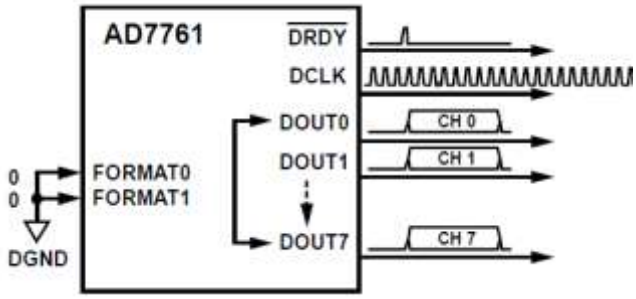

Fig. 8 Formato de datos digitales del convertidor

Por medio de la terminal CLK_SEL se ha seleccionado que la fuente de reloj provenga del exterior, ésta debe tener una frecuencia de 32.768 MHz, la cual será dividida entre 32 de acuerdo con el código programado por las terminales MODE3:0 como se observa en la figura 9 para obtener la frecuencia del reloj DCLK, en este caso se emplea el modo de baja potencia.

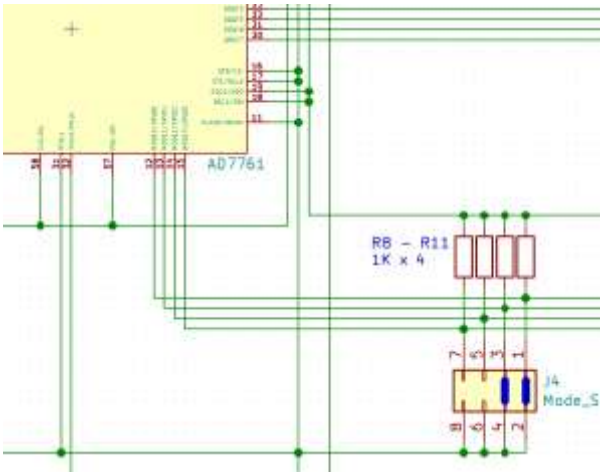

Fig. 9 Configuración del convertidor analógico a digital con modo de control de PIN

También se puede observar que las terminales ST1:ST0 están conectadas a 0 V, con lo que se habilita la salida de todos los canales. Con DEC1:0 se selecciona la decimación de la frecuencia, para este caso se divide la frecuencia del reloj DCLK entre 1024 para obtener la frecuencia de muestreo la cual es de 1 KHz. Se eligió esta frecuencia para cumplir plenamente con el teorema de Nyquist sin tener que manejar grandes cantidades de memoria, pues por cada segundo se emplearán 1000 datos por cada canal, y si se requiere trabajar un almacenamiento de una hora se tendrá que ocupar un espacio de 68.67 MB tan solo por cuatro canales. La capacidad máxima del sistema es suficiente para capturar 16 canales, de manera que, para almacenar una hora se requerirá un espacio de memoria de 223.16 MB.

7.2.2. La tarjeta madre

Todo el sistema está montado de manera modular, con el propósito de contar con escalamiento desde un sistema básico hasta un sistema completo. El mínimo de elementos que se requieren para completar un sistema básico es una tarjeta madre y un módulo analógico. Cada módulo cuenta con dos canales y consiste en un PCB que contiene todos los dispositivos descritos en la sección 6.1 para 2 canales (Fig. 10). Separado de la sección analógica se encuentra la sección digital en donde se ubica el espacio para dos convertidores, CAD de 8 canales, cada uno y el zócalo para el sistema de adquisición. Este último está implementado por una tarjeta de desarrollo de la marca Altera (DE10-Nano), que posee un Sistema integrado (SoC por sus siglas en ingles “System on Chip”) modelo Cyclone V de la marca Intel, el cual tiene integrado un FPGA de 44000 elementos lógicos y un procesador ARM A9 de dos núcleos. El sistema cuenta con memoria RAM de 4 GB, periféricos como puertos USB-UART, manejador de tarjeta microSD, un puerto USB-OTG, además de un puerto de salida para video y hasta un total de 64 puertos de entrada salida de propósito general.

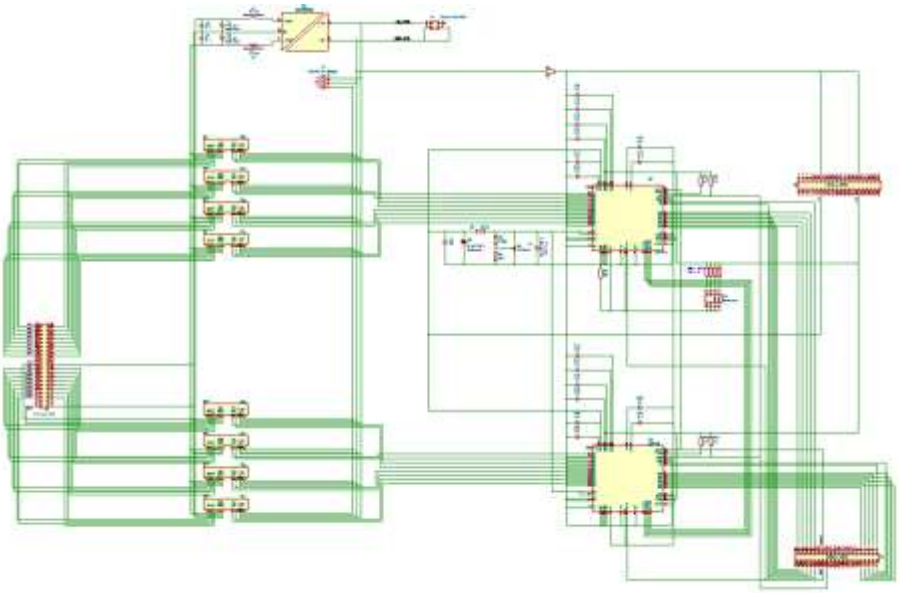

Fig.10 Tarjeta madre. La sección analógica (izquierda) esta formada por 8 módulos. La sección digital (derecha) contiene los convertidores y el zócalo para el sistema de adquisición.

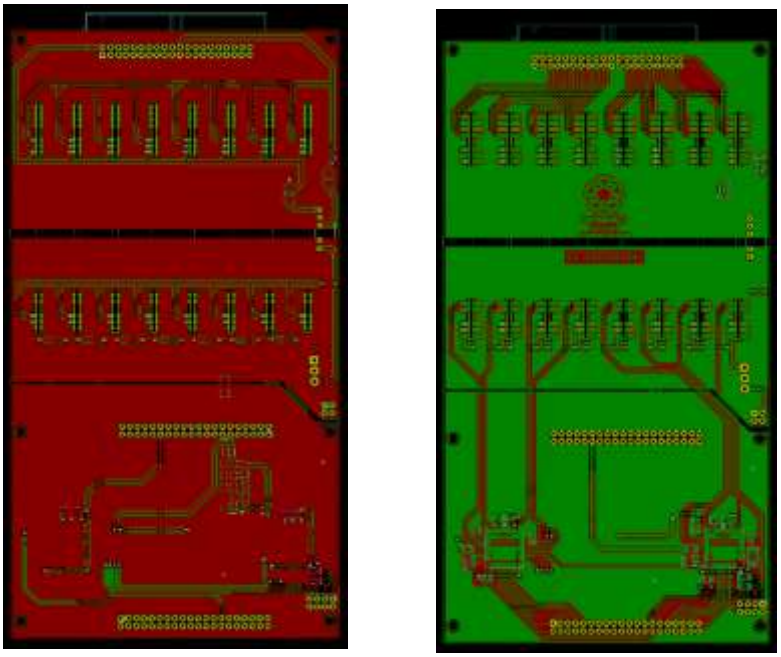

Fig. 11 Diseño del PCB para la tarjeta madre

En la figura 11 se muestra el diseño del PCB de la tarjeta madre, nótese que se contemplaron dos espacios para convertidores analógico a digital, y 8 ranuras para módulos analógicos. Por lo que se puede tener un máximo de 16 canales.

7.3. Módulo de desarrollo DE10-Nano

DE10-Nano es un módulo de desarrollo para diseño de múltiples aplicaciones de tiempo real. Presenta una sólida plataforma de diseño de hardware basada en el FPGA de Intel SoC (“System-on-Chip”), el cual combina un núcleo doble Cortex-A9 integrado con una lógica programable de última generación, proporcionando una gran flexibilidad de diseño, al aprovechar el poder de la reconfigurabilidad de la tarjeta sumada con un procesador de alto rendimiento y baja potencia. El SoC de Intel integra un sistema de procesador físico (HPS, “Hard Proccesor System”) basado en la arquitectura ARM que consta de procesador, periféricos e interfaces de memoria vinculadas perfectamente con el FPGA empleando un ancho de banda alto para la manipulación de datos dentro de la red principal del dispositivo. La placa de desarrollo DE10-Nano está equipada con memoria de alta velocidad DDR3, convertidores analógicos a digitales, redes Ethernet y mucho más.

Las figuras 12 y 13 presentan las diferentes funciones con las que cuenta la tarjeta DE10-Nano. Las indicaciones en color verde presentan todos los periféricos que son controlados por el FPGA mientras que las indicadas con naranja son aquellas que están controladas por el HPS

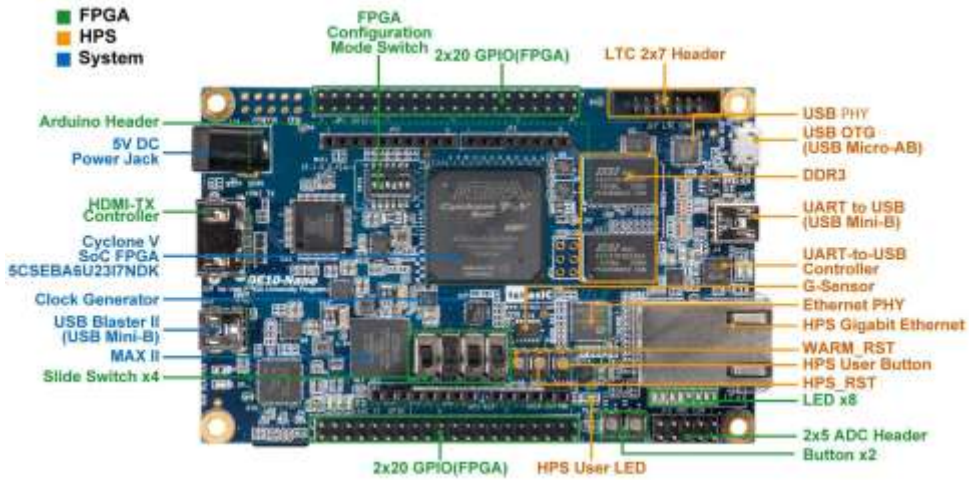

Fig. 12 Vista superior de la tarjeta DE10-Nano

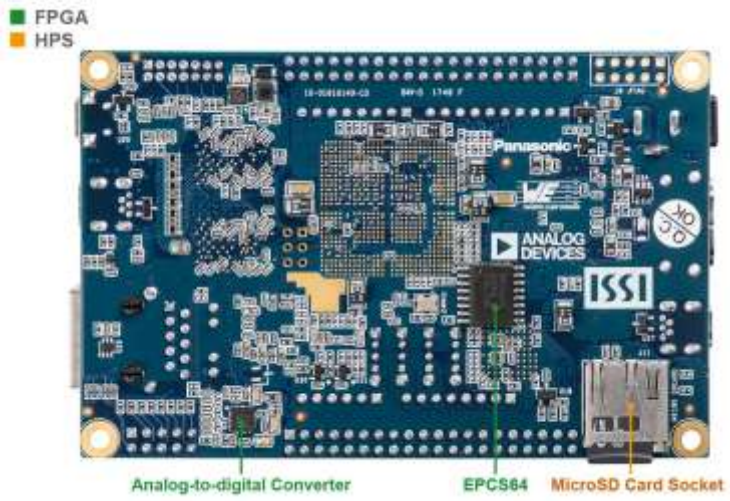

Fig. 13 Vista inferior de la tarjeta DE10-Nano

Las características que posee la tarjeta son las siguientes:

- Para el FPGA
  - Dispositivo Cyclone® V SE 5CSEBA6U23I7NDK.
  - Dispositivo de configuración, formato serial - EPCS64.
  - USB-Blaster II integrado para la programación; Modo JTAG.
  - 2 botones.
  - 4 interruptores deslizantes.
  - 8 LED de usuario verdes.
  - Tres fuentes de reloj de 50MHz del generador de reloj.
  - Dos zócalos de expansión de 40 pines de entrada/salida de propósito general.
  - Un zócalo de expansión Arduino Uno R3.
  - Un zócalo de expansión de entradas analógicas de 10 pines. (compartido con las entradas analógicas del Arduino).
  - Convertidor A / D, interfaz SPI de 4 hilos con el FPGA.
  - Puerto HDMI TX, compatible con DVI v1.0 y HDCP v1.4

- Para el HPS (Sistema de procesador duro).
  - Procesador ARM Cortex-A9 de doble núcleo a 800 MHz.
  - SDRAM DDR3 de 1 GB (bus de datos de 32 bits).
  - Puerto Ethernet PHY de 1 Gigabit con conector RJ45.
  - Puerto USB OTG, conector USB Micro-AB.
  - Interfaz para tarjeta microSD.
  - Acelerómetro (interfaz I2C + interrupción)

- UART a USB, conector USB Mini-B.
- Botón de reinicio por sobrecalentamiento y botón de reinicio de trabajo en frío.
- Un botón de usuario y un LED de usuario.
- Zócalo de expansión LTC 2x7.

De las diversas formas de configuración que posee la tarjeta, la aplicación que se implementa en el presente trabajo es por medio de la operación desde un sistema operativo ligero basado en la plataforma de Linux. Las siguientes secciones describen los diferentes módulos que se emplean en esta aplicación. Información más detallada sobre el funcionamiento de cada uno de ellos se encuentra en el documento del fabricante (10)

7.3.1. Circuitería de reloj.

La frecuencia predeterminada de todos los relojes externos para el SoC FPGA Cyclone V emplea un generador de reloj para distribuir múltiples señales de baja fluctuación. Hay dos señales de 50MHz conectadas a la sección FPGA y se utilizan como fuentes de reloj para la lógica de usuario. Otras dos señales de 25 MHz son utilizadas por la sección HPS, también se proporcionan dos señales más de 24 MHz para dar sincronía a los circuitos USB Blaster II y el USB OTG PHY. Y una última señal de 25 MHz se encuentra conectada al reloj del transceptor Gigabit Ethernet. La figura 14 muestra el diagrama de bloques de está circuitería.

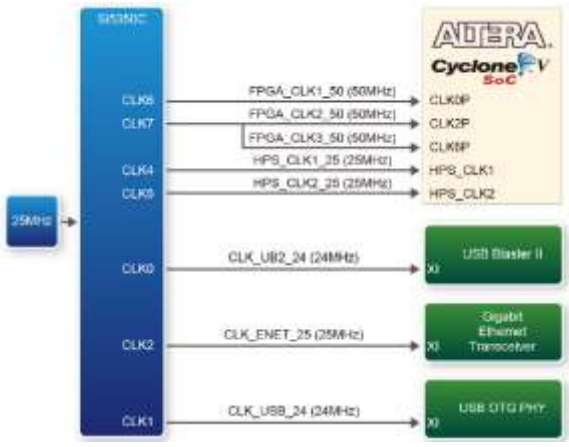

Fig. 14 Diagrama de bloques de la circuitería de reloj para el FPGA

7.3.2. Zócalo de expansión de puertos de entrada salida GPIO_1

La placa cuenta con dos zócalos de expansión de 40 pines. Cada uno contiene 36 terminales de usuario de propósito general conectadas directamente a la FPGA. También provee de alimentación de DC + 5V y DC + 3.3V (VCC3P3) y dos conexiones para GND. La Figura 15 muestra la distribución de terminales del zócalo GPIO_1 que es el que se empleará para nuestra aplicación, tal como se describe en el apartado de la tarjeta madre, en este capítulo.

Fig.15 Zócalo de expansión GPIO_1

7.3.3. Puerto de transmisión de video HDMI

La tarjeta proporciona un transmisor HDMI de alto rendimiento a empleando el dispositivo ADV7513 que incorpora características HDMI v1.4 con soporte de video 3D a 165 MHz. Admite todos los formatos de video hasta 1080p y UXGA. Este dispositivo se controla mediante una interfaz serial I2C, que está conectada a los pines del FPGA. UNA El diagrama de bloques se muestra en la Figura 16. Nótese que no se cuenta con señales de audio por lo que no es posible reproducir sonido alguno.

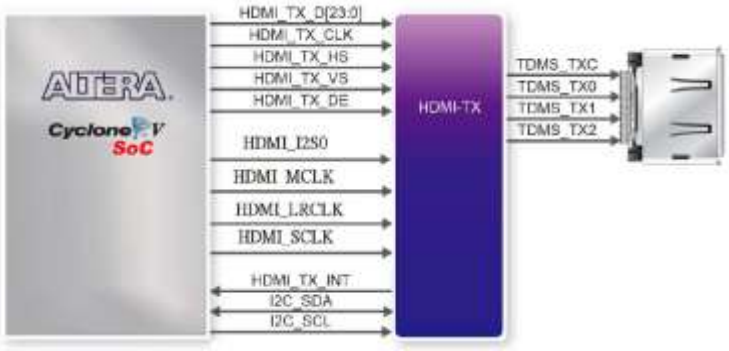

Fig. 16 Interfaz de transmisión HDMI.

7.3.4. Memoria RAM DDR3 de 1 GB

Se cuenta también con una memoria RAM de 1 GB que consta de dos dispositivos DDR3 de 16 bit´s controlados por el manipulador de memoria del HPS con una velocidad de trabajo de 400 MHz. En la figura 17 se indica la forma como están conectadas estas memorias.

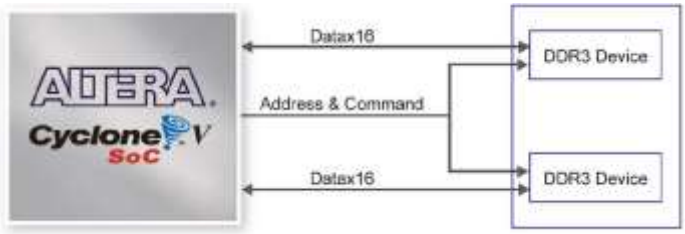

Fig17. Conexiones de la memoria RAM con el FPGA

7.3.5. Interfaz para tarjeta microSD

Se tiene la posibilidad de manejar una interfaz para tarjeta microSD de cuatro canales de datos. Sirve no solo para almacenamiento externo para el HPS, sino también como una opción de arranque alternativa para el DE10-Nano, la cual es utilizada por la aplicación para albergar el sistema operativo del equipo. La Figura 18 muestra las señales conectadas entre el HPS y el zócalo de la tarjeta Micro SD.

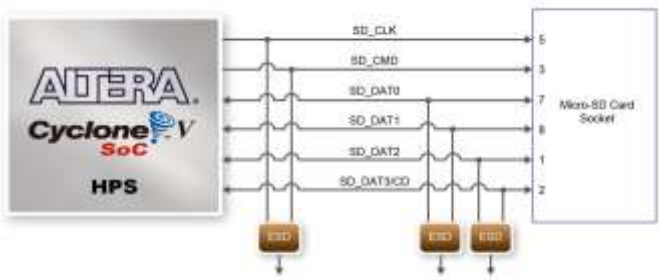

Fig. 18 Conexiones para el zócalo de la interfaz para la tarjeta microSD

7.3.6. Puerto USB OTG PHY 2.0

La tarjeta de desarrollo tiene una interfaz USB a través del controlador SMSC USB3300. se usa para interactuar con una conexión tipo AB para conector micro-USB. Este dispositivo admite la interfaz UTMI + Low Pin (ULPI) para comunicarse con el controlador USB 2.0 en el HPS. Según lo definido por el modo OTG, el PHY puede funcionar en modo Host o dispositivo. Cuando se opera en modo Host, la interfaz suministrará la energía al dispositivo a través de la interfaz Micro-USB. La Figura 19 muestra las conexiones de USB PTG PHY al HPS.

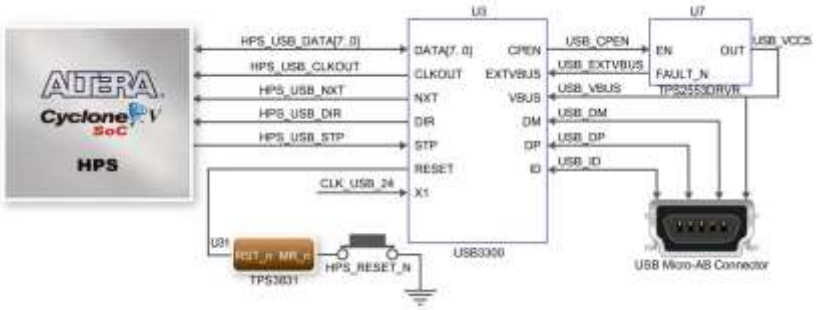

Fig. 19 Conexiones de la interfaz USB OTG PHY con el HPS

7.4. Estructura de la aplicación

La aplicación que se desarrolló requirió de tres niveles de implementación del “harware”, por un lado se requiere de un procesamiento de la señal por medios analógicos, los cuales ya se describieron en la sección 6.2, el segundo nivel se realizó en la plataforma de desarrollo del FPGA de Intel, que emplea el lenguaje de

descripción de hardware Verilog, y se lleva a cabo por medio de la suite de desarrollo Quartus Prime 18.1 que contiene un analizador, un sintetizador de hardware y un ensamblador, para crear la lógica de la aplicación dentro de la sección FPGA del dispositivo Cyclone V SoC. El tercer nivel se desarrolla por medio de una plataforma de descripción que permite realizar las conexiones necesarias entre el HPS y el FPGA, esta plataforma se llama Qsys y consiste en un software que interconecta por medio gráfico periféricos manipulados por el procesador de doble núcleo con los periféricos creados en el FPGA a través de puentes de comunicación, logrando interconectarlos a través del manejo de la memoria RAM de 1 G. Esta plataforma permite mapear dentro de dicho espacio todos los periféricos otorgándole una sección para cada uno y poder así manejarlos desde las localidades de memoria que los controlan.

7.4.1. Control de la conversión

A pesar de que la tarjeta de desarrollo cuenta con un módulo convertidor analógico a digital se requiere implementar una interfaz que controle el dispositivo convertidor AD7761 pues se requiere de una resolución de 16 bits y un manejo simultaneo de hasta 16 canales. La primera implementación que se realizó fue de cuatro canales simultáneos. Es por esa razón que el diseño del control de la conversión se llevó a cabo por la lógica implementada en la sección del FPGA. El primer paso es establecer la fuente de reloj para el convertidor. Esta se hace con un PLL que utiliza como fuente principal de reloj la señal de 50 MHz provenientes del exterior del FPGA. El PLL entrega una señal de 32.768 MHz que será enviada al exterior por medio de un puerto de salida dentro del GPIO_1 (PIN25), correspondiente al PIN_AF21 del dispositivo. La figura 20 presenta el código del módulo PLL creado para este propósito.

```
179
180 PLL_admclk  PLL_admclk_inst(
181             .refclk(fpga_clk_50),    //  refclk.clk
182             .rst(1'b0),              //  reset.reset
183             .outclk_0(ad_mclk),      // outclk0.clk
184             .outclk_1( )             //clk_900)     // outclk1.clk
185             );
186
187
```

Fig. 20 Código Verilog para el módulo PLL

7.4.2. Módulos de captura de datos digitales

El siguiente paso es la implementación del módulo de adquisición del dato. Debido a que el formato de transmisión de los datos desde el convertidor hacia el FPGA es de modo serial, se requiere un convertidor de formato serie a formatos paralelo, es decir, que para cada canal se deberá tomar el dato que viene en una sola línea y pasarlo a un registro de 16 bits que entren de manera concurrente a la siguiente etapa de la lógica. Para ello se cuenta con las señales drdy y dclk. La primera establece el momento en el que el dato de conversión está listo para ser leído y la segunda establece la velocidad con la que se extraerá el dato de la interfaz hacia el FPGA. La señal dclk sincroniza el dato bit a bit por medio de flancos de bajada y su frecuencia de trabajo se define al momento de establecer los parámetros del convertidor con el cpntrol de PIN, para este caso se emplea una frecuencia de 1.024 MHz. Por su parte, los pines de control de la decimación están fijos en el valor binario 11 dividiendo la frecuencia del dclk entre 1024 para obtener una frecuencia de muestreo de 1000 Hz.

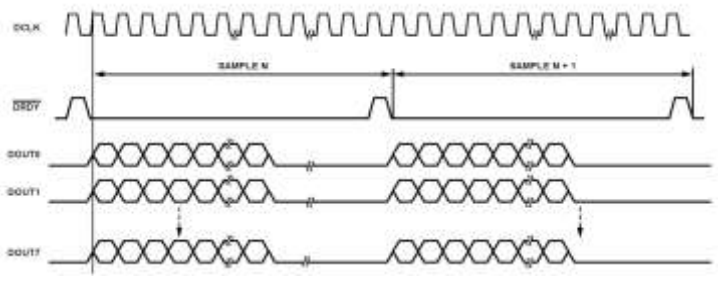

Fig. 21 Protocolo de transmisión de los datos digitales

Debido a que los datos se obtienen en formato serial es necesario que el FPGA convierta a un formato paralelo, para poder manipularlos adecuadamente. El protocolo que emplea el convertidor es el que se muestra en la figura 21. Como se puede observar, la señal de reloj debe activar la lectura del bit correspondiente en el flanco de bajada. El primer bit que se lee es el más significativo y el dato está conformado por un Byte cabecera y luego los 16 bits del dato. En el byte cabecera se observan algunos bits de control del flujo y el número de canal que se está transmitiendo. Estos parámetros están descritos en el documento de datos del fabricante (11). Seguido de la cabecera se entrega el dato de 16 bits comenzando también por el más significativo. Es importante indicar que el tipo de dato es signado y los valores negativos se muestran en complemento a 2. Este es un aspecto muy relevante para el tratamiento de dicho dato en el momento que se pasa al código de la interfaz gráfica.

Para hacer la conversión a formato paralelo se emplea un módulo sintetizado en lenguaje Verilog, que conforma la primera sección del arreglo lógico del FPGA. Está constituido por un puerto que contiene 3 entradas de un bit y tres salidas. Una de las salidas es un registro de 16 bits y las otras dos son registros de un bit que se utilizan para control de la siguiente etapa. El código del módulo es el siguiente:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                              //
//          MODULO ConvSerPar_v.v                                                                                               //
//                                                                                                                              //
// El presente módulo realiza la conversión de formato serial a paralelo de un dato que proviene de                            //
// un convertidor AD7761 o AD7768.                                                                                              //
// El formato de conversión puede ser de 16 bits o de 24 bits según sea el convertidor.                                         //
// Autor:           Ing. Andrés Francisco Morín Sánchez    (18/ago/2019).                                                       //
//Proyecto:         Registrador ECoG                                                                                            //
//                                                                                                                              //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module ConvSerPar_v (dclk, drdy, sdi, Q, clko, sdo);

// Declaracion de parámetros
        parameter       Formato = 16;    // Formato del dato en numero de bits
```

```
// Puertos de entrada y salida
        input wire          dclk;     // La frecuencia del ADC para este reloj es de 4.096 MHz
        input wire          drdy;     // flanco de bajada activo para arranuque de la transmisióndel dato
        input wire          sdi;      // El dato inicia con un header de 8 bits, y luego el MSB del dato (24 o 16 bit)
        output reg [Formato-1:0] Q; // Dato en formato paralelo
        output reg          clko;     // Salida de reloj para habilitar dato paralelo debe permitir la lectura
        output reg          sdo;      // Salida serial

// Declaracion de variables
        reg             [6:0]                   contr;    // Contador de filo de subida
        reg             [6:0]                   contf;    // Contador de filo de bajada
        reg             [6:0]                   tope;
        reg             [31:0]                  Dato;

// Comportamiento del módulo
        always @(negedge dclk)  begin       // Detecta el flanco de bajada del reloj
                if (drdy == 1'b1) begin             // Inicializa datos
                        Dato    =       32'h00000000;
                        clko    =       1'b0;
                        contf   =       6'h00;
                        end

                else if (clko == 1'b0) begin
                        Dato[31:1] = Dato[30:0];// Recorre el dato un bit a la izquierda
                        Dato[0] = sdi;
                        contf = contf + 1'b1;
                        if (contf == Formato + 8) begin
                                Q = Dato[Formato - 1:0]   // Toma solo el dato sin la cabecera
                                clko = 1'b1;              // El dato está listo en formato paralelo
                                end
                        end
                end

        always @(posedge dclk) begin                    // Detecta el flanco de subida del reloj
                if (clko == 1'b0) begin                 // Inicializa datos
                        contr   =       6'h00;
                        sdo     =       1'b0;
                        end

                else if (clko == 1'b1) begin
                        case (contr)
                                6'h00 :  sdo = Dato[31];
                                6'h01 :  sdo = Dato[30];
                                6'h02 :  sdo = Dato[29];
                                6'h03 :  sdo = Dato[28];
                                6'h04 :  sdo = Dato[27];
                                6'h05 :  sdo = Dato[26];
                                6'h06 :  sdo = Dato[25];
                                6'h07 :  sdo = Dato[24];
                                6'h08 :  sdo = Dato[23];
                                6'h09 :  sdo = Dato[22];
                                6'h0A :  sdo = Dato[21];
                                6'h0B :  sdo = Dato[20];
                                6'h0C :  sdo = Dato[19];
                                6'h0D :  sdo = Dato[18];
```

```
                    6'h0E :  sdo = Dato[17];
                    6'h0F :  sdo = Dato[16];
                    6'h10 :  sdo = Dato[15];
                    6'h11 :  sdo = Dato[14];
                    6'h12 :  sdo = Dato[13];
                    6'h13 :  sdo = Dato[12];
                    6'h14 :  sdo = Dato[11];
                    6'h15 :  sdo = Dato[10];
                    6'h16 :  sdo = Dato[9];
                    6'h17 :  sdo = Dato[8];
                    6'h18 :  sdo = Dato[7];
                    6'h19 :  sdo = Dato[6];
                    6'h1A :  sdo = Dato[5];
                    6'h1B :  sdo = Dato[4];
                    6'h1C :  sdo = Dato[3];
                    6'h1D :  sdo = Dato[2];
                    6'h1E :  sdo = Dato[1];
                    6'h1F :  sdo = Dato[0];
                    default: sdo = 1'b0;
                    endcase
                contr = contr + 1'b1;
                end
        end
endmodule
```

Revisando la arquitectura del módulo en la primera sección se atiende la señal de reloj dclk en el flanco de bajada, si la señal drdy está en nivel alto entonces se está iniciando una transmisión de dato de conversión, por lo que es necesario comenzar la variable de cambio de formato en el valor cero, al mismo tiempo se inicia el contador de bits y se pone en nivel bajo la señal de salida clko. Mientras esta señal esté en bajo se habilita el cambio de formato de serie a paralelo en la siguiente sección, pues se pregunta por el valor de dicho registro. El método empleado es haciendo un corrimiento a través del dato desde el bit menos significativo, pues el convertidor transmite en primer lugar el bit más significativo. Se ocupa un registro de 32 bits por el hecho de que el protocolo de transmisión del convertidor incluye una cabecera de 8 bits que se transmite antes del dato, por lo que el contador deberá llegar hasta el valor máximo de 24 bits para nuestro caso. Se ha dejado de manera genérica por medio del parámetro formato para incluir los casos en los que se ocupe un convertidor de 24 bits, en cuyo caso se emplearía todo el registro de 32. Una vez terminada la

transmisión el contador llegará a la cuenta máxima y colocará el dato en la salida Q del módulo, que consiste en un registro del tamaño del Formato, (16 bits para este caso). Después de colocada la salida en formato paralelo se activa la señal clko en nivel alto para indicar que el dato está listo para ser leído por la siguiente etapa del sistema digital. Esta acción controla además que el dato se siga recorriendo a pesar de que la señal dclk continúe operando, pues al estar en nivel alto no actuará esta parte del circuito lógico. La última sección del módulo genera la señal de salida sdo en formato serial y es atendida durante el flanco de subida de la señal dclk. Esta acción también es controlada por la señal de reloj de salida clko, pues solamente se generará la salida serial mientras clko este en nivel alto, y solo se realizará una vez.

La siguiente etapa es un arreglo de memoria de tipo FIFO, también llamada tubería y su propósito es el de almacenar los datos en un buffer antes de entregarlos al procesador que los ocupará para desplegarlos en la ventana de una interfaz gráfica. Consiste en un conjunto de registros ordenados por medio de un buffer de direcciones. Sin embargo, el acceso a los registros del banco de memoria es solo para lectura, es decir, que el bus de direcciones se emplea solamente para apuntar la localidad que se desea leer. El módulo tiene un puerto con cinco entradas y tres salidas. Primeramente, se tiene la señal de entrada clkpush que fuerza la entrada del dato a través de DataIn, empujando todos los demás hacia la salida, es decir que hace un corrimiento de registros desde el más bajo hasta el más alto, sacando el primer dato que entró, por el registro de salida DataOut. Cuando el banco de memoria se encuentra lleno, es decir, que todos los registros están ocupados, se activa la salida full. Por su parte el bus de direcciones se emplea para apuntar a la localidad que se desea leer del banco, cuyo contenido aparecerá en el registro de salida rddata. El procedimiento de lectura es el siguiente: Primero se debe activar la lectura por medio de la entrada rdena y luego se coloca el valor de dirección para lectura en el puerto

de entrada address, seguido de esto se genera un pulso en el puerto de entrada rdclk, y finalmente se toma el dato desde el registro de salida rddata. El código Verilog para este módulo es el que se describe a continuación:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                                //
//                                                MODULO RamTubo.v                                                                //
//                                                                                                                                //
// El presente módulo almacena los datos provenientes del convertidor de formato serie a paralelo en                             //
// un banco de memoria y los recorre cada vez que entra un dato nuevo.                                                            //
// El tamaño y capacidad de la memoria esta determinado por los parámetros del banco.                                             //
// Autor:          Ing. Andrés Francisco Morín Sánchez    (18/ago/2019).                                                          //
//      Proyecto:       Registrador de ECoG                                                                                       //
//                                                                                                                                //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module RamTubo(clkpush, DataIn, DataOut, rdclk, rdena, address, rddata, full);

// Declaracion de parámetros
        parameter       data_width      = 16;   // Ancho del tunel en bits
        parameter       addr_width      = 7;    // Ancho del bus de dirección
        parameter       depth           = 2**addr_width;// Profundidad del tubo en numero de datos

// Puertos de entrada
        input   wire                            clkpush;
        input   wire    [data_width - 1:0]      DataIn; //Genera tantos canales como esté indicado
        input wire                              rdclk;
        input wire                              rdena;
        input wire      [addr_width - 1:0]      address;  // Puertos de salida
        output reg      [data_width - 1:0]      DataOut; // Genera tantos canales como esté indicado
        output reg      [data_width - 1:0]      rddata;   // Genera tantos canales como esté indicado
        output reg                              full;

// Construcción del banco de memoria
        reg     [data_width - 1:0]      bank[depth - 1:0];

// Declaración de variables internas y señales
        integer         i,cont;

// Asignaciones
// Comportmiento del módulo
// Procedimiento de entrada de dato
        always @(posedge clkpush)
                begin
                if (rdena == 1'b0)
                        begin
                        DataOut = 32'bz;        // Desconecta el dato de salida
                        end
                else
                        begin
```

```
                DataOut = bank[depth - 1];        // Saca el primer dato que entró al tubo
                end
        for(i=0; i<depth-1; i=i+1)
                begin
                bank[depth - 1 - i] = bank[depth - 2 - i];  // Recorre los datos del tubo
                end
        bank[0] = DataIn;                   // Mete el nuevo dato al tubo
        cont = cont + 1;
        if (cont > depth - 1)
                begin
                        cont = 0;
                        full = 1'b0;
                end
        else if (cont == depth - 1)
                begin
                        full = 1'b1;
                end
        else
                begin
                        full = 1'b0;
                end
        end

// Procedimiento de lectura de datos
        always @(posedge rdclk)  // Siempre que haya flanco de subida  en el reloj de lectura
                begin
                if (rdena == 1'b0)  // ¿Está deshabilitada la lectura de datos?
                        begin       // Si, desconecta la lectura de datos
                        rddata = 32'bz;
                        end
                else
                        begin      // Entrega el dato apuntado
                        rddata = bank[address];
                        end
                end
endmodule
```

La arquitectura de este módulo está constituida por dos procedimientos. El primero de ellos es atendido en el flanco de subida de la señal clkpush y realiza el embutido del dato nuevo al tubo, empleando un corrimiento de registros y sacando el dato más viejo por el registro de salida. El segundo procedimiento se atiende en el flanco de subida de la señal rdclk y lleva a cabo la lectura del dato apuntado por el puntero el bus de direcciones. Es importante resaltar en ambos procedimientos que si la señal rdena está en nivel bajo los registros de salida DataOut y rddata se mostrarán en estado de

alta impedancia, por lo que dicha señal deberá estar en alto para poder tomar el dato de lectura.

7.4.3. Sistema integrado en el dispositivo (SoC)

El sistema integrado en el dispositivo, (SoC) por sus siglas en Ingles, *"System on Chip"* es un sistema embebido que cuenta con un procesador ARM-A9 de doble núcleo, un tejido lógico compuesto por la sección FPGA y un control de manipulación de datos que establece la comunicación entre las dos secciones, el FPGA y el HPS por medio de puentes de alta velocidad, con capacidad de manejar grandes cantidades de datos entre los diferentes periféricos conectados al dispositivo. La sección del HPS contiene un manejador de memoria con capacidad de manipular hasta un total de 4 GB acomodados en una memoria DDR3 que, al ser ordenados en registros de 32 bits, se tiene un total de 1 G de capacidad de almacenamiento. El sistema puede manejar periféricos importantes como la comunicación Eternet, para poder tener acceso a la WEB, además de puertos USB con interfaz UART y un puerto USB-OTG para usarlo como *"Host"* para el manejo de memoria externa y periféricos como teclado o ratón. También cuenta con un manejador de tarjeta microSD para el almacenamiento del *"software"* y el propio sistema operativo.

La implementación de este sistema se realiza por medio de la plataforma de desarrollo Qsys. Esta herramienta se usa junto con el software CAD Quartus Prime y permite al usuario crear fácilmente un sistema basado en el procesador ARM-CortexA9, simplemente seleccionando las unidades funcionales deseadas y especificando sus parámetros. Para mayor información sobre, cómo se emplea y configura la plataforma, consultar el manual de usuario (12)

La aplicación muestra una ventana en la que se incluyen todos los módulos incorporados al sistema y las conexiones que se hacen entre ellos. La figura 22 presenta una imagen de dicha pantalla. En ella se puede observar que se está incorporando el procesador HPS con todos los aditamentos necesarios para manipular los periféricos de la tarjeta DE10-Nano. Hay otros elementos más, como un manejador de señales de reloj y reset, los puentes de interconexión entre la sección del HPS y el FPGA que poseen conexiones de tipo maestro y esclavo. Cada módulo tiene un conjunto de registros dentro del espacio de memoria RAM para poder tener acceso al periférico solicitando los datos provenientes de este hacia el procesador y también controlar las salidas del procesador hacia los periféricos. En dichos registros se encuentran los datos de configuración de cada periférico, y estos estarán sintetizados dentro de la red lógica del FPGA. En el software se debe crear un espacio virtual de memoria para direccionar los registros de configuración de dichos módulos. Funcionan como los registros que se programan a un microcontrolador para tener acceso a los puertos de entrada salida y los diferentes periféricos que internos. Dentro de los módulos importantes que se emplean en la aplicación de nuestro interés, están los que corresponden al periférico de transmisión de video HDMI y el controlador de la memoria externa, además del manejador de interrupciones y el puerto de comunicación JTAG para la interacción entre el dispositivo y algunos otros sistemas que requieran enlace remoto. En esta estructura también se agregan los manejadores de los puertos de entrada y salida de propósito general en donde se están integrando las señales provenientes de las tuberías sintetizadas en el FPGA que son las que proporcionan los datos adquiridos del convertidor analógico a digital. Por cada canal es necesario implementar un manejador para que el procesador pueda realizar las acciones de desplegado en una gráfica dentro de la interfaz de usuario.

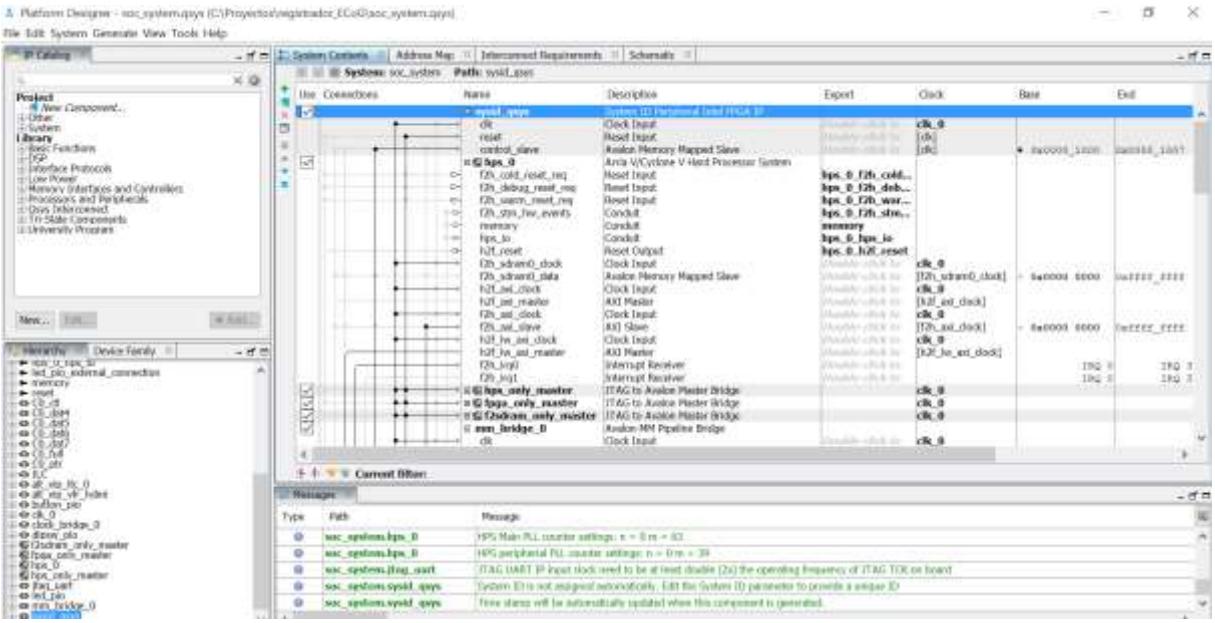

Fig. 22 Ventana principal de la herramienta de ensamble Qsys

Una vez implementado el sistema con la herramienta Qsys se debe generar el módulo Verilog que debe ser incorporado en el tejido de la lógica del FPGA. Lo que se obtiene como producto es un código que corresponde a la síntesis de todo el sistema para ser interconectado con las tuberías y los conversores de formato serie a paralelo que capturan los datos provenientes del convertidor.

En la herramienta CAD Quartus Prime, se hacen todas estas conexiones a través de un último módulo creado en código Verilog, que enlazará las entradas y salidas con los puertos físicos de la tarjeta de desarrollo. El código correspondiente se muestra en el Apéndice 1.

Durante la compilación del sistema se generan diversos archivos. Uno de los cuales se llama DE10_NANO_SOC_FB.sof, el cual deberá ser convertido al archivo soc_system.rbf que es el que se grabará en la tarjeta microSD del sistema operativo para que el FPGA pueda ser iniciado desde este controlador. Es decir que el sistema de arranque del sistema se ejecutará desde el sistema operativo de la tarjeta de desarrollo.

8. Diseño del “software”

8.1. Sistema de arranque

Para poder iniciar la operación del instrumento es necesario considerar que este opera bajo la plataforma del sistema operativo Linux, y para ello se hace necesario contar con una memoria microSD de arranque que se inserta en el zócalo J11 de la tarjeta DE10-Nano. En la microSD se tienen cuatro archivos, zImage como su nombre lo indica es la imagen del sistema operativo Linux, tiene un ambiente gráfico básico con todos los elementos funcionales para la ejecución de aplicaciones con el estilo de los ambientes de escritorio de muchos sistemas intuitivos. Al iniciar el sistema arranca el archivo u-boot.scr que es el encargado de inicializar todas las operaciones del HPS del integrado Cyclone V SoC de la tarjeta. Una vez que se ha inicializado el sistema se carga el archivo soc_system.rbf que es el que contiene todos las conexiones internas que se deben hacer para que el FPGA opere según se sintetizó durante el proceso de generación del tejido lógico. Por último, el archivo soc_system.dbt contiene datos relevantes del sistema para ser utilizados por el kernel del sistema operativo. Después de la inicialización se muestra en el monitor conectado al puerto HDMI la pantalla principal del escritorio para el sistema LDXE como se observa en la figura 23.

Fig. 23 Pantalla del escritorio del sistema LXDE.

8.2. Aplicación “Registrador de ECoG”

Como el sistema operativo es Linux, es indispensable realizar la programación de la interfaz gráfica con una plataforma que pueda operar en dicho ambiente, y no conforme con ello, deberá adaptarse a las condiciones del sistema LXDE que se basa en una distribución que ocupa el mínimo de recursos para poder hacerlo flexible y ligero. Por ello se decidió programar en lenguaje C++ por medio de la plataforma QT Creator versión 5.7.0 de manera que se instaló una terminal virtual dentro del sistema Windows 10 de la PC para instalar en ella el sistema operativo Ubuntu 16.04.06 de 64 bits. Posteriormente se instaló en la misma terminal la herramienta QT Creator anexándole el compilador apropiado para el procesador Cortex-A9.

Con esta herramienta se diseñó la interfaz gráfica de usuario, a la que se le puso por nombre “**Registrador de ECoG**”. Con ella se capturan los datos para presentarlos en una gráfica de dos dimensiones utilizando la librería de código libre QCustomPlot que posee muchas de las prestaciones que proporciona MatLab para generación de gráficas, pero en el ambiente de LXDE. Lo que se busca es mostrar en tiempo real el comportamiento de la señal en función del tiempo y para ello se creó la pantalla que se muestra en la figura 24. La pantalla muestra la gráfica de cuatro canales simultáneos, y en la parte inferior derecha se encuentran dos controles, uno para la base de tiempo que puede ser modificada desde 0.1 s hasta 10 s e indica el ancho de la ventana de tiempo que se está observando en la gráfica. Por su parte el control de amplitud permite variar el máximo de visualización de voltaje que va desde 20 hasta 200 µV e indica el valor máximo que se puede visualizar en la gráfica.

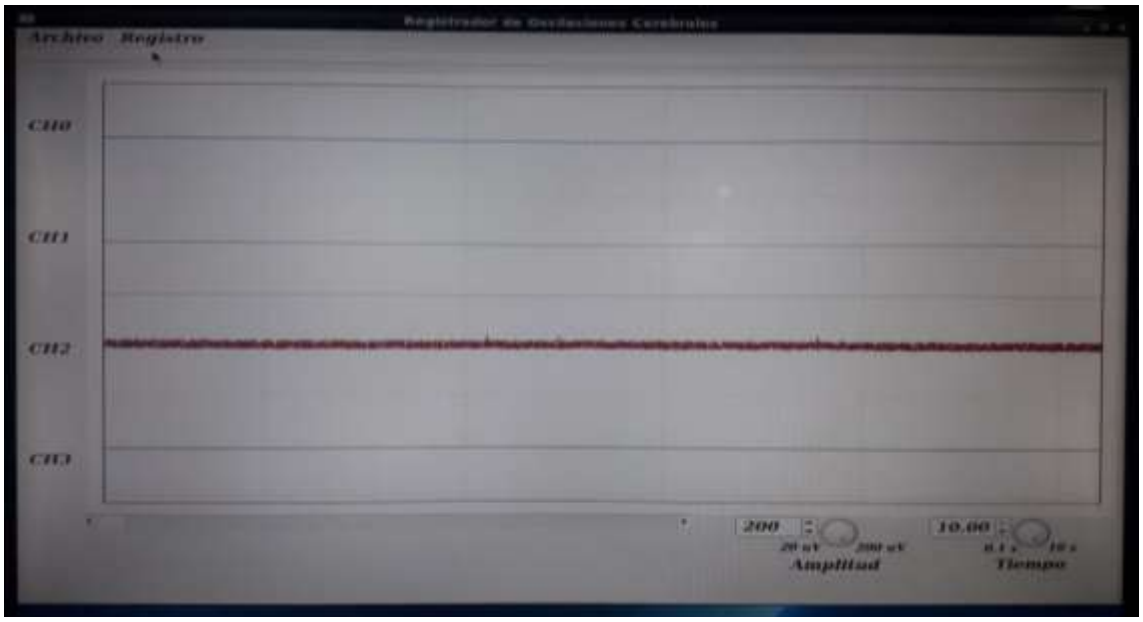

Fig. 24 Ventana de la interfaz gráfica de usuario para el registrador.

Qt Creator es una plataforma cruzada, con un completo entorno de desarrollo integrado (IDE). Está disponible para Linux, Mac OS X y Windows. El código desarrollado para la ventana principal es el siguiente:

```
#include "mainwin.h"
#include "ui_mainwin.h"
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <qtimer.h>
#include <math.h>
#include <qstring.h>
#include "socal.h"
#include "hps.h"
#include "alt_gpio.h"
#include "mman.h"
#include "hps_0.h"

#define HW_REG_BASE     ( 0xFC000000 )
#define HW_REG_SPAN     ( 0X04000000 )
#define HW_REGS_MASK    ( HW_REG_SPAN - 1 )

// Definicion de parámetros del programa y funciones básicas
   // bit(b) regresa el bit x puesto a uno y los demás bits en cero, ej. BIT(3) regresa 00001000
   #define bit(b)        (1<<(b))
   // bitget(x,b) regresa el bit b-esimo de x
   #define bitget(x,b)   ((x) & bit(b))
   // bitset(x,b) establece en '1' el bit b de x
   #define bitset(x,b)   ((x) |= bit(b))
   // bitclr(x,b) establece a '0' el bit b de x
   #define bitclr(x,b) ((x) &= ~bit(b))
   // bittogg(x,b) invierte el valor del bit b de x a su complemento,
   #define bittogg(x,b)  ((x) ^= bit(b))
```

```
    // wrbit(x,b,v) establece el valor 'v' de
    #define wrbit(x,b,v) ((v)? bitset(x,b) : bitclr(x,b))

    void*   virtual_base;
    void*   ptr_addr;
    void*   full_addr;
    void*   ctl_addr;
    void*   CH4_addr;
    void*   CH5_addr;
    void*   CH6_addr;
    void*   CH7_addr;
    void*   led_addr;
    int     fd;

// Genera los vectores de tiempo y magnitud
    QVector<double> t, ch4, ch5, ch6, ch7; // la longitud la determina la funcion PlotAmpl()

    mainwin::mainwin(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::mainwin)
{
// Codigo del constructor mainwin

    ui->setupUi(this);

    def_mem();
    inic_Ploter();

// Inicializaciones de tiempo
    Timer0 = new QTimer(this);
    connect(Timer0, SIGNAL(timeout()), this, SLOT(Ploting()));
}
// Metodos de la clase mainwin

mainwin::~mainwin()
{
    delete ui;
}
void mainwin::def_mem()
{
    fd = open("/dev/mem",(O_RDWR|O_SYNC));
    virtual_base = mmap(NULL, HW_REG_SPAN, (PROT_READ|PROT_WRITE), MAP_SHARED, fd, HW_REG_BASE);
    CH4_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_DAT4_BASE)&(unsigned
long)(HW_REGS_MASK));
    CH5_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_DAT5_BASE)&(unsigned
long)(HW_REGS_MASK));
    CH6_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_DAT6_BASE)&(unsigned
long)(HW_REGS_MASK));
    CH7_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_DAT7_BASE)&(unsigned
long)(HW_REGS_MASK));
    ptr_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_PTR_BASE)&(unsigned
long)(HW_REGS_MASK));
    full_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_FULL_BASE)&(unsigned
long)(HW_REGS_MASK));
    ctl_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + C0_CTL_BASE)&(unsigned
long)(HW_REGS_MASK));
    led_addr = virtual_base + ((unsigned long)(ALT_FPGA_BRIDGE_LWH2F_OFST + LED_PIO_BASE)&(unsigned
long)(HW_REGS_MASK));
}
void mainwin::inic_Ploter(void)
{
    // Creacion de la grafica
        ui->QCP_Canales->addGraph(); // Grafica en azul para el canal 4
        ui->QCP_Canales->graph(0)->setPen(QPen(QColor(20, 20, 160)));
        ui->QCP_Canales->addGraph(); // Grafica en rojo para el canal 5
        ui->QCP_Canales->graph(1)->setPen(QPen(QColor(20, 130, 160)));
        ui->QCP_Canales->addGraph(); // Grafica en azul para el canal 6
        ui->QCP_Canales->graph(2)->setPen(QPen(QColor(160, 20, 20)));
        ui->QCP_Canales->addGraph(); // Grafica en rojo para el canal 7
        ui->QCP_Canales->graph(3)->setPen(QPen(QColor(20, 130, 100)));
```

```
    // Eje horizontal como eje de tiempo
        QSharedPointer<QCPAxisTickerTime> timeTicker(new QCPAxisTickerTime);
        timeTicker->setTimeFormat("%s");
        ui->QCP_Canales->xAxis->setTicker(timeTicker);
    // Visualizacion de las marcas de los ejes apagadas
        ui->QCP_Canales->xAxis->setTicks(false);
        ui->QCP_Canales->yAxis->setTicks(false);
    // Ajuste de los ejes
        ui->QCP_Canales->axisRect()->setupFullAxesBox();
        //ui->QCP_Canales->xAxis->setRange(0, 10);
        ui->QCP_Canales->xAxis->setTickLength(10);
        ui->QCP_Canales->yAxis->setRange(-4.15, 4.15);
        ui->QCP_Canales->yAxis->grid()->setVisible(true);//false);
    // Etiquetas
        //ui->QCP_Canales->xAxis->setLabel("x");
        //ui->QCP_Canales->yAxis->setLabel("y");
        // set axes ranges, so we see all data:
        ui->QCP_Canales->xAxis->setTickLabels(false);
        ui->QCP_Canales->yAxis->setTickLabels(false);
}
void mainwin::Ploting()
{
    static QTime crono(QTime::currentTime());
    uint32_t aa;
    double  ch_dat;
    double tn;         // Instante en que sucedio la muestra N
    tt = crono.elapsed()/1000.0;
    static double ttant = 0.0;
    int N = 128;     // Numero de muestras para capturar

    if (tt-ttant > (N-1)/1000.0) // El tiempo transcurrido es mayor a (N-1) ms
    {
        for (int i=0; i<N; i++)
        {
            tn = tt - 0.127 + i/1000.0;   // indica el tiempo en segundos

            // Apunta al dato que se desea sacar del tubo comenzando por el mas viejo
            *(uint8_t*)ptr_addr = 0x7F - (uint8_t)i;

            //bitset(*(uint32_t*)ctl_addr,12); // Habilita la lectura de datos del tubo 4
            bitset(*(uint32_t*)ctl_addr,4);
            aa = ((*(uint32_t*)CH4_addr) + 0x00008000) & 0x0000FFFF;    // Carga el dato apuntado
            bitclr(*(uint32_t*)ctl_addr,4);
            ch_dat = (double)aa - 32768;
            ch_dat = ch_dat*0.0125/ui->dSBuVolt->value(); //   Resolucion de 4.096V/32768=125uV
            ui->QCP_Canales->graph(0)->addData(tn, ch_dat/4+3.075); // Agrega los datos a la grafica
            //bitclr(*(uint32_t*)ctl_addr,12);    // Deshabilita la lectura de datos para el canal CH4

            // Apunta al dato que se desea sacar del tubo 5 comenzando por el mas viejo
            //bitset(*(uint32_t*)ctl_addr,13); // Habilita la lectura de datos del tubo 5
            bitset(*(uint32_t*)ctl_addr,5);
            aa = ((*(uint32_t*)CH5_addr) + 0x00008000) & 0x0000FFFF;    // Carga el dato apuntado
            bitclr(*(uint32_t*)ctl_addr,5);
            ch_dat = (double)aa - 32768;
            ch_dat = ch_dat*0.0125/ui->dSBuVolt->value(); //   Resolucion de 4.096V/32768=125uV
            ui->QCP_Canales->graph(1)->addData(tn, ch_dat/4+1.025); // Agrega los datos a la grafica
            //bitclr(*(uint32_t*)ctl_addr,13);    // Deshabilita la lectura de datos para el canal CH4

            // Apunta al dato que se desea sacar del tubo 6 comenzando por el mas viejo
            //bitset(*(uint32_t*)ctl_addr,14); // Habilita la lectura de datos del tubo 5
            bitset(*(uint32_t*)ctl_addr,6);
            aa = ((*(uint32_t*)CH6_addr) + 0x00008000) & 0x0000FFFF;    // Carga el dato apuntado
            bitclr(*(uint32_t*)ctl_addr,6);
            ch_dat = (double)aa - 32768;
            ch_dat = ch_dat*0.0125/ui->dSBuVolt->value(); //   Resolucion de 4.096V/32768=125uV
            ui->QCP_Canales->graph(2)->addData(tn, ch_dat/4-1.025); // Agrega los datos a la grafica
            //bitclr(*(uint32_t*)ctl_addr,14);    // Deshabilita la lectura de datos para el canal CH4
            // Apunta al dato que se desea sacar del tubo 7 comenzando por el mas viejo
            //bitset(*(uint32_t*)ctl_addr,15); // Habilita la lectura de datos del tubo 5
            bitset(*(uint32_t*)ctl_addr,7);
```

```
            aa = ((*(uint32_t*)CH7_addr) + 0x00008000) & 0x0000FFFF;    // Carga el dato apuntado
            bitclr(*(uint32_t*)ctl_addr,7);
            ch_dat = (double)aa - 32768;
            ch_dat = ch_dat*0.0125/ui->dSBuVolt->value(); //   Resolucion de 4.096V/32768=125uV
            ui->QCP_Canales->graph(3)->addData(tn, ch_dat/4-3.075); // Agrega los datos a la grafica
            //bitclr(*(uint32_t*)ctl_addr,15);    // Deshabilita la lectura de datos para el canal CH4
        }
        ttant = tt;
        ui->QCP_Canales->xAxis->setRange(tt, ui->dSBTime->value(), Qt::AlignRight);
        ui->QCP_Canales->replot();
    }
    //if (crono.elapsed() >= 60)
    //{
    //    Timer0->stop();   // Deten el registro si ya pasaron 60 segundos
    //}
}
void mainwin::on_actionInicio_triggered()
{
    Timer0->start(0); // Arranca el registro
}
void mainwin::on_actionParo_triggered()
{
    Timer0->stop();   // Deten el registro
}
```

La parte medular de este código es la función miembro Ploting(), que se ejecuta cada que se desborda el temporizador Timer0. Una vez que entra a la función se determina si ya han transcurrido 128 ms y si es el caso, entonces la función toma los 128 datos de cada tubo correspondientes a los canales que se están convirtiendo y adquiriendo en el FPGA. Una vez que se han obtenido todos los datos se realiza un ploteo de las variables con los nuevos datos, los cuales se van almacenando cada uno en arreglos de datos de tipo double con un identificador correspondiente al tiempo en milisegundos.

Por su parte el inicio y el paro del registro se hace activando la señal de desbordamiento de Timer0 y realizando al conexión de la variable con la función miembro Pliting(). La activación se lleva a cabo por medio de la función start() del objeto Qtimer en la acción de selección del submenú “Inicio” que se encuentra en la pestaña “Registro” del menú principal. Para la parada del registro se emplea el submenú “Paro” que se encuentra en la misma pestaña y ejecuta la función stop() del objeto Qtimer.

9. El electrodo

Para el electrodo se pensó primero en desarrollar una interfaz que permita conectar el conjunto de terminales del conector peine de la tarjeta madre. En dicho conector se cuenta con alimentación (V+) y (V-) de manera que es posible alimentar un conjunto de amplificadores operacionales que funcionen como buffers de acoplamiento de impedancias y guarda activa. Como se conectarán 4 canales a la cabeza del roedor y se debe conectar un electrodo más para el regreso de la referencia y otro para la conexión de tierra, en total serán seis electrodos. La zona en donde se colocarán los electrodos activos es en la región M1 descrita en el atlas del cerebro de rata (5), con una separación de 2.54 mm entre ellos. La colocación del electrodo de referencia se hará en la unión Lambda y el de tierra se colocará en la unión de Bregma como se indica en la figura 25.

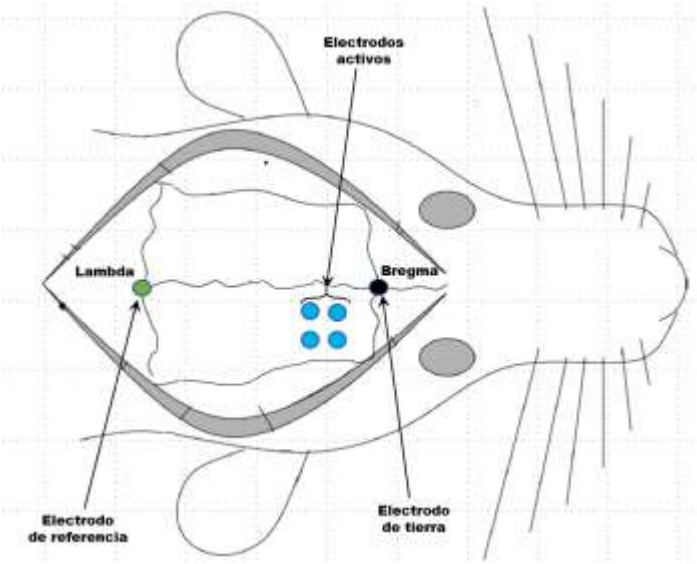

Fig. 25 Ubicación de los electrodos.

El circuito de la guarda activa se muestra en la figura 26, y consiste en un acoplamiento de impedancias por medio de burffers y una red de Wilson. El regreso de la referencia se obtiene conectando el punto central de la red al electrodo de referencia a través de

un acoplamiento de impedancias para cancelar el ruido que se introduce por el modo común.

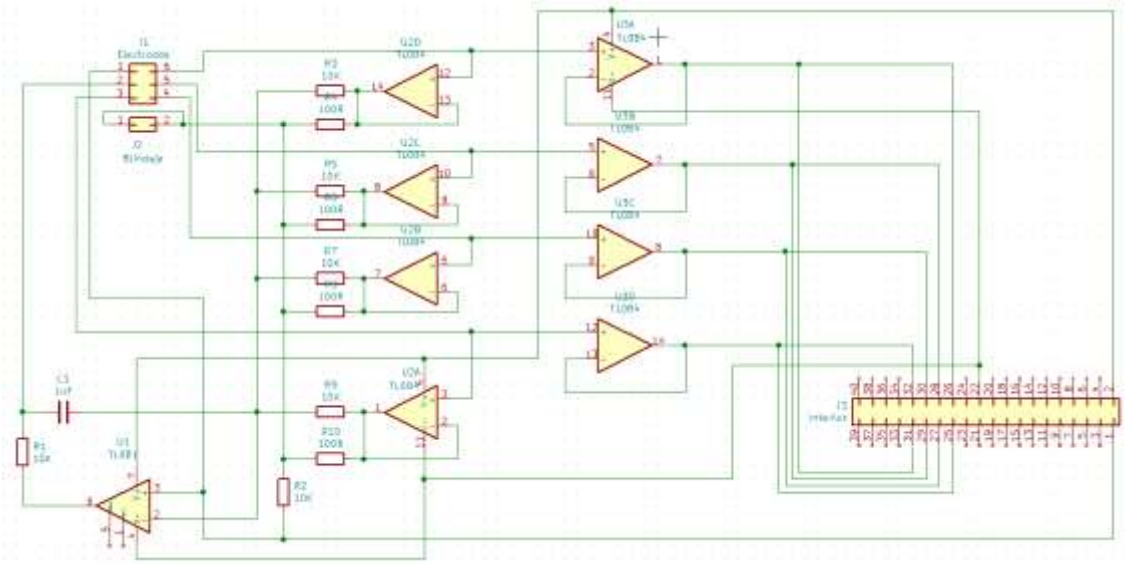

Fig. 26 Circuito de la guarda Activa.

10. Resultados

10.1. Validación del instrumento

Para validar el instrumento se sometió a pruebas de laboratorio, inyectando una señal senoidal a la entrada de uno de los canales, con un generador de funciones de la marca RIGOL, modelo DG1022. Se hizo un barrido de frecuencia desde el valor de 100 mHz hasta 500 Hz, para abarcar todo el ancho de banda correspondiente al tipo de registro que se desea obtener. El generador tiene la capacidad de proporcionar una amplitud mínima de 4 mV, por lo que se necesitó implementar un reductor por medio de un divisor de voltaje diferencial, como se muestra en la figura 27. Los valores medidos de dichas resistencias permitieron establecer una reducción calculada de 108 µV pico a pico.

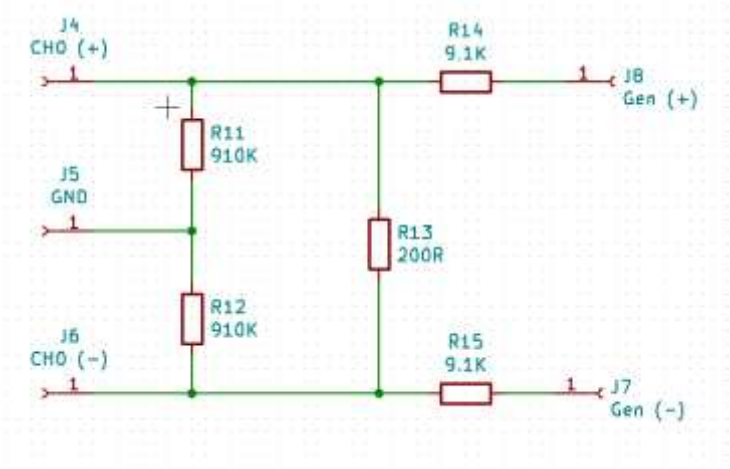

Fig. 27 Reductor de voltaje por divisor de voltaje

Es importante mencionar que se debe conectar la referencia GND para evitar problemas de ruido balanceando el circuito. Para ello se debe elegir el valor de las resistencias lo más parecidas entre sí, estableciendo un porcentaje de error de ±0.1%.

Se midió el voltaje de salida del canal que es el que se aplica a la entrada del convertidor analógico a digital empleando un osciloscopio de la marca Tektronix modelo TDS 2004B obteniéndose la siguiente tabla de valores:

Frecuencia (Hz)	Vod (Vpp)	Ad	Ad (dB)	
0.003	0.1	526.7607405	-31.36403448	
0.03	1.64	8638.876144	-7.06715752	
0.05	2.46	12958.31422	-3.545332339	
0.06	2.78	14643.94859	-2.483138563	*fc1*
0.1	3.24	17067.04799	-1.153134277	
0.3	3.661	19284.71071	-0.092039904	
3	3.7	19490.1474	0	
30	3.7	19490.1474	0	
300	3.9	20543.66888	0.457257659	
325	3.92	20649.02103	0.501688859	
350	3.96	20859.72532	0.589869237	
400	4	21070.42962	0.677165345	
500	4.06	21386.48606	0.80648619	
750	3.84	20227.61244	0.322590006	
1000	2.72	14327.89214	-2.672656401	*fc2*
1500	1.36	7163.946071	-8.693256314	
3000	0.435	2291.409221	-18.59424934	

Tabla de valores de magnitud para diferentes frecuencias

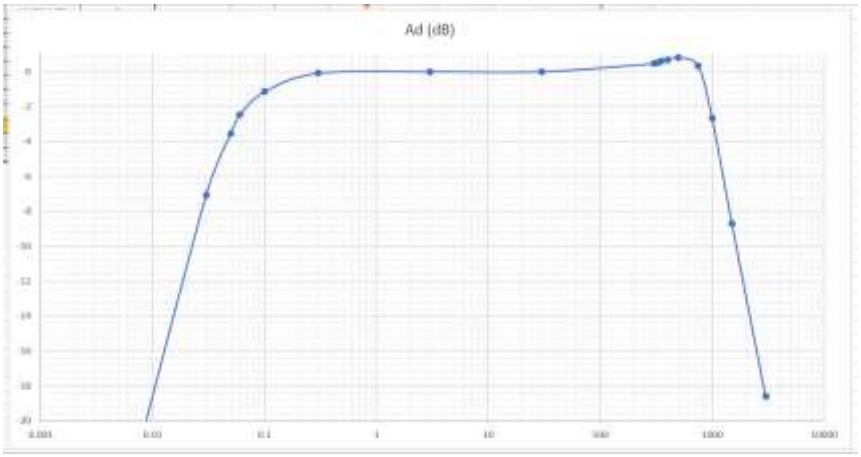

Fig. 28 Función de transferencia para la señal del amplificador de instrumentación.

Se puede observar en la figura 28 que el amplificador es máximamente plano dentro del rango de frecuencias en que se desea operar. La etapa de filtrado recorta la señal a valores de frecuencia de 239 Hz, y es la señal que será convertida a digital para desplegarla en la pantalla de la interfaz gráfica de usuario. Las siguientes imágenes muestran algunas de las señales que se obtuvieron con el registrador y su correspondiente señal analógica medida con el osciloscopio en la salida del amplificador de aislamiento, el canal empleado para la captura de la señal es el indicado como CH0.

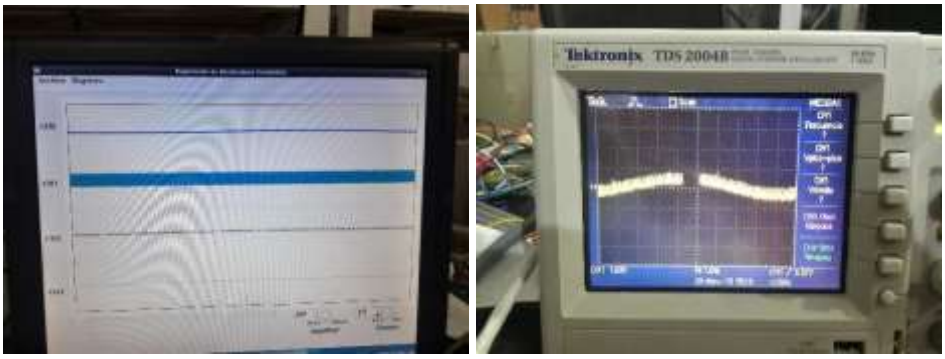

Fig. 29 Señales de salida de la interfaz gráfica y del canal analógico a 10 mHz

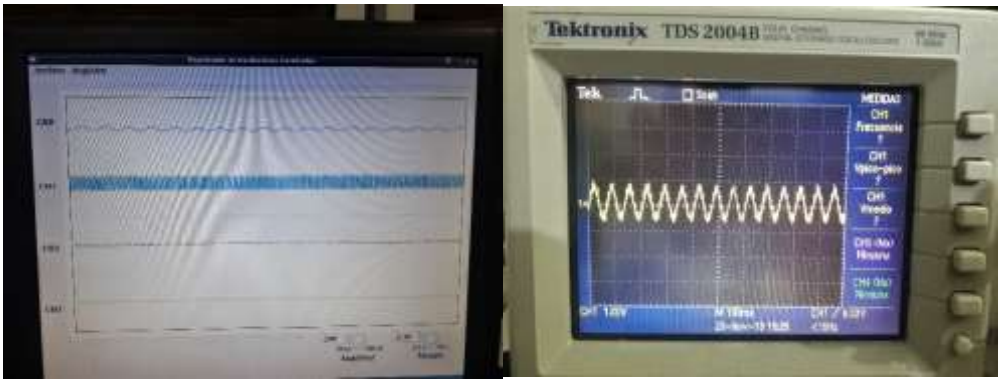

Fig. 30 Señales de salida de la interfaz gráfica y del canal analógico a 15 Hz

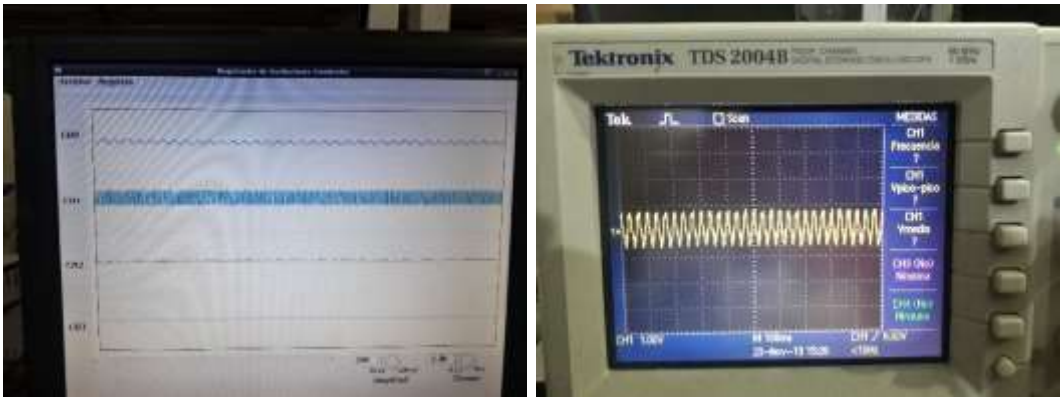

Fig. 31 Señales de salida de la interfaz gráfica y del canal analógico a 30 Hz

Fig. 32 Registrador con el electrodo conectado; del lado derecho una vista interna del circuito

10.2. Ficha técnica del registrador de ECoG

En esta sección se presentan las características técnicas del instrumento de medición que se desarrolló, en las que se especifican aspectos importantes para el buen manejo del equipo y las recomendaciones de seguridad para una buena captura de la señal sin riesgo para el personal que realice la prueba, así como para evitar el daño de la muestra.

10.2.1. Características generales

A continuación, se indican las características generales para el registrador de ECoG:

Tipo de medida:	Oscilaciones cerebrales corticales ECoG
Medición:	Diferencial
Número de canales:	4 (expandible a 16)
Rango de entrada:	0.2896 a 289.6 μV_{RMS}
Frecuencia de trabajo:	100 mHz a 300 Hz
Aislamiento:	1000 kVDC
Alimentación de la sección aislada:	± 5 V @ ± 100 mA
Tipo de respuesta:	Lineal para el rango de entrada
Tiempo de recuperación por sobretiro:	2 s
Sensibilidad:	0.01 V/µV
Temperatura de trabajo:	0 a 80 °C
Humedad:	< 90 %
Dimensiones: (Ancho x Alto x Largo)	24 x 8 x 22.5 cm
Peso:	1355 g

10.2.2. Características del procesamiento analógico

El método de procesamiento empleado es una amplificación con compensación en amplitud y frecuencia con atenuación de la banda alta a partir de los 239 Hz, y para la banda baja hasta antes de los 100 mHz, mostrando como respuesta en frecuencia, la siguiente función de transferencia:

$$H(s) = \frac{jwGR}{jw\tau_1 + (G+1)} * \frac{{w_0}^2}{jw\frac{w_0}{Q} + ({w_0}^2 - w^2)}$$

Donde:
G = 10000
R = 5 MΩ
$\tau_1 = 1.649\ s$
$w_0 = 1507.5\ rad/s\ ;\ f_0 = 239\ Hz$
Q = 0.276

Que representa un sistema de amplificación con ganancia de 10000 con un filtrado pasa-altos de 0.1 Hz. La constante de tiempo τ_1 indica un ajuste de cero ejecutado por la retroalimentación de las bajas frecuencias en la entrada de referencia del amplificador de instrumentación. El factor de calidad Q, y la frecuencia característica w_0 de la segunda etapa representan un filtrado pasa-bajos con frecuencia de corte de 239 Hz. Otras especificaciones para el procesamiento analógico se indican a continuación:

Parámetro	Símbolo	Valor			Unidad
		Mínimo	Típico	Máximo	
Razón de rechazo al modo común	CMRR	75			dB
Inmunidad al ruido	SNR		31.28		dB
Impedancia de entrada	Zi	3.14		7508	MΩ
Ganancia de voltaje	G		10000		

10.2.3. Características del procesamiento digital

El tipo de conversión empleado está basada en la tecnología Σ-Δ que es un convertidor de sobre muestreo con un filtro digital de banda ancha, que proporcionan las siguientes características:

Parámetro	Símbolo	Valor			Unidad
		Mínimo	Típico	Máximo	
Frecuencia de muestreo	Fs		1000		Hz
Factor de amplificación digital			1		
Resolución	Resol		1		bit
Rango Dinámico	RD		96.32		dB
Convertidor	n		16		bit

10.2.4. Características de salida

El dato de salida se muestra por medio de una ventana dentro del ambiente gráfico del sistema operativo. Los valores se indican en la imagen por medio de los controles de tiempo y amplitud, que determinan los rangos que se observarán durante la captura de datos. A continuación, se presentan las características para la salida:

Tipo de salida:	Señal de tiempo real
Interfaz:	Puerto HDMI Tx
Presentación de los datos:	Interfaz gráfica GUI Programada en C++
Sistema operativo:	Linux LXDE de baja densidad
Almacenamiento:	Tarjeta microSD y puerto de salida USB para almacenamiento masivo

Parámetro	Símbolo	Valor			Unidad
		Mínimo	Típico	Máximo	
Rango de salida en amplitud	Vo	20		200	µV
Ventana de tiempo de visualización	B_T	0.1		10	s
Velocidad de pantallazo	St		100		ms
Retardo de señal	dt		100		ms

10.2.5. Otras especificaciones

Sistema de protección: Fusible tipo europeo 250 V_{AC} @ 500 mA

Condiciones de seguridad: Conectar a tierra física segura. No usar adaptador de alimentación

Higiene: Limpiar con trapo húmedo estando desconectado el equipo. No usar solventes.

Parámetro	Símbolo	Valor			Unidad
		Mínimo	Típico	Máximo	
Tensión de alimentación	V_{POW}	90	127	140	V_{AC}
Consumo de corriente	I_{POW}	100		350	mA
Frecuencia de la alimentación	f_{POW}		60		Hz

11. Conclusión

El traumatismo cráneo encefálico (TCE) es un problema de salud de gran relevancia a nivel mundial. En México es una de la principales causas de morbilidad y mortandad, por lo que se están haciendo muchas investigaciones sobre los tratamientos que reduzcan los efectos del trauma secundario, pues es la fase en la que una atención pronta y expedita, pero además la aplicación del tratamiento adecuado es determinante para la recuperación con el mínimo daño para el paciente.

Equipos como el que se desarrolló en el presente trabajo dan lugar al mejoramiento de las condiciones en los tratamientos de emergencia, pues permiten obtener datos relevantes del estado del paciente y acertar en las medidas curativas. Todo con el propósito de evitar daños posteriores al trauma inicial. Es bien sabido que en el primer momento del TCE no es posible hacer cosa alguna, pero las horas siguientes son las más importantes para reducir los efectos adversos como la isquemia. El instrumento desarrollado nos permite registrar señales de los efectos fisiológicos que se presentan en este cuadro clínico, y para efectos de investigación es una herramienta muy poderosa para identificar características y tener conocimiento de la recuperación del tejido afectado.

Si bien, no se pudo hacer registro en el modelo animal, como era previsto, se hicieron mediciones de laboratorio, que le dan un grado de validez al instrumento, pues las señales que se capturan con él, son iguales a las que se observan en el osciloscopio y a pesar de que la interfaz gráfica solamente permite hacer una estimación de amplitud y tiempo se puede corroborar fácilmente que las frecuencias que se miden con el instrumento corresponden a las medidas con el osciloscopio, partiendo de que la base de tiempo ajustada en el control contiene el número de ciclos correspondiente. De la misma manera, puede hacerse para la amplitud, y con ello determinar la calibración del equipo desarrollado.

11.1. Trabajos futuros

El instrumento desarrollado es una herramienta muy poderosa para la implementación de aplicaciones, pues es un sistema embebido con múltiples prestaciones. El simple hecho de contar con un sistema operativo lo convierte en un equipo versátil pues con desarrollar una interfaz gráfica de usuario se pueden modificar las características del equipo permitiendo hacer a futuro procesamiento de las señales que se capturan.

Por otro lado, el contar con una interfaz física intercambiable provoca que con una simple modificación en dicha interfaz el instrumento se pueda configurar para capturar otro tipo de señales como son las del ECG, y como se tienen hasta un total de 16 canales, fácilmente se puede configurar como un registrador ECG de 12 derivaciones.

También es posible, haciendo mínimos cambios al instrumento, que tenga algunos canales de salida para electroestimulación y con ello poderlo configurar como un equipo terapéutico. Son muchas las posibilidades que tiene el sistema que se desarrolló, por lo que los trabajos futuros pueden tomar diferentes caminos, pero todos ellos para el mejoramiento de las condiciones de vida de los pacientes traumatizados.

12. Referencias

1. *Manejo del trauma craneoencefálico en la atención primaria en salud. MéD. UIS. 2015; 28(1):153-158.* **Piña Tornés, Arlines Alina.** 1, s.l. : MeD. UIS, 2015, Vol. 28.

2. **Rada Martin, Saraí.** Recomendaciones de cuidados de enfermería para el paciente con traumatismo craneoencefálico severo ingresado en la UCI-A del complejo Hospitalario de Navarra. Universidad Pública de Navarra. *https://academica-e.unavarra.es/bitstream/handle/2454/11278/SaraiRadaMartin.pdf?sequence=1&isAllowed=y.* [En línea] 2014.

3. *Neuroprotección y traumatismo craneoencefálico.* **Estrada Rojo, Francisco, y otros.** 2012, Revista de la Facultad de Medicina de la UNAM, págs. 16-29.

4. *¿Por qué los científicos experimentan con ratas y ratones?* **Bonilla, Armando.** 2018, Agencia Informativa Conacyt.

5. *Ratat1: A Digital Rat Brain Stereotaxic Atlas Derived from High-Resolution MRI Images Scanned in Three Dimensions.* **Wisner, Kurt, y otros.** 2016, Frontiers in Systems Neuroscience.

6. *En la recuperación ante un traumatismo.* **G. E., Arón, y otros.** 2010.

7. *NEUROCIENCIA, La exploración del cerebro.* s.l**. : Wolters Klower, 2016.**

8. *Biocompatibility and safety of PLA and its copolymers.* **Ramot, Yuval, y otros. 2016, Advanced Drug Delivery Reviews, págs. 153-162.**

9. *AC-Coupled Front-End for Biopotential Measurements.* **Spinelli, Enrique Mario, Pallàs-Areny, Ramon y Mayosky, Miguel Angel. 2003, IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, págs. 391-395.**

10. Terasic Inc. *DE10-Nano, Cyclon V SoC with Dual-core ARM Cortex-A9, User Manual.* 2017**.**

11. Analog Devices, Inc. *8-Channel, 16-Bit, Simultaneous Sampling ADC with Power Scaling, 110.8 kHz BW.* 2017.

12. Altera Corporation. *Introduction to the Altera Qsys System Integration Tool.* 2014.

Apéndice 1 Código Verilog para el módulo principal del sistema.

```
`default_nettype none

//=======================================================
//  This code is generated by Terasic System Builder
//=======================================================

module DE10_NANO_SOC_FB(

	//////////// CLOCK //////////
	input 		          		FPGA_CLK1_50,
	input 		          		FPGA_CLK2_50,
	input 		          		FPGA_CLK3_50,

	//////////// HDMI //////////
	inout 		          		HDMI_I2C_SCL,
	inout 		          		HDMI_I2C_SDA,
	inout 		          		HDMI_I2S,
	inout 		          		HDMI_LRCLK,
	inout 		          		HDMI_MCLK,
	inout 		          		HDMI_SCLK,
	output 		          		HDMI_TX_CLK,
	output 		    [23:0]		HDMI_TX_D,
	output 		          		HDMI_TX_DE,
	output 		          		HDMI_TX_HS,
	input 		          		HDMI_TX_INT,
	output 		          		HDMI_TX_VS,

	//////////// HPS //////////
	inout 		          		HPS_CONV_USB_N,
	output 		    [14:0]		HPS_DDR3_ADDR,
	output 		     [2:0]		HPS_DDR3_BA,
	output 		          		HPS_DDR3_CAS_N,
	output 		          		HPS_DDR3_CK_N,
	output 		          		HPS_DDR3_CK_P,
	output 		          		HPS_DDR3_CKE,
	output 		          		HPS_DDR3_CS_N,
	output 		     [3:0]		HPS_DDR3_DM,
	inout 		    [31:0]		HPS_DDR3_DQ,
	inout 		     [3:0]		HPS_DDR3_DQS_N,
	inout 		     [3:0]		HPS_DDR3_DQS_P,
	output 		          		HPS_DDR3_ODT,
	output 		          		HPS_DDR3_RAS_N,
	output 		          		HPS_DDR3_RESET_N,
	input 		          		HPS_DDR3_RZQ,
	output 		          		HPS_DDR3_WE_N,
	output 		          		HPS_ENET_GTX_CLK,
	inout 		          		HPS_ENET_INT_N,
	output 		          		HPS_ENET_MDC,
	inout 		          		HPS_ENET_MDIO,
	input 		          		HPS_ENET_RX_CLK,
	input 		     [3:0]		HPS_ENET_RX_DATA,
	input 		          		HPS_ENET_RX_DV,
	output 		     [3:0]		HPS_ENET_TX_DATA,
	output 		          		HPS_ENET_TX_EN,
	inout 		          		HPS_GSENSOR_INT,
	inout 		          		HPS_I2C0_SCLK,
	inout 		          		HPS_I2C0_SDAT,
	inout 		          		HPS_I2C1_SCLK,
	inout 		          		HPS_I2C1_SDAT,
	inout 		          		HPS_KEY,
	inout 		          		HPS_LED,
	inout 		          		HPS_LTC_GPIO,
	output 		          		HPS_SD_CLK,
	inout 		          		HPS_SD_CMD,
	inout 		     [3:0]		HPS_SD_DATA,
	output 		          		HPS_SPIM_CLK,
```

```
        input                                   HPS_SPIM_MISO,
        output                                  HPS_SPIM_MOSI,
        inout                                   HPS_SPIM_SS,
        input                                   HPS_UART_RX,
        output                                  HPS_UART_TX,
        input                                   HPS_USB_CLKOUT,
        inout           [7:0]                   HPS_USB_DATA,
        input                                   HPS_USB_DIR,
        input                                   HPS_USB_NXT,
        output                                  HPS_USB_STP,

        ////////// KEY //////////
        input           [1:0]                   KEY,

        ////////// LED //////////
        output          [7:0]                   LED,

        ////////// SW //////////
        input           [3:0]                   SW,

        //// Conexiones FPGA ////
        // Entradas/Salidas de propósito general banco 0 en la tarjeta DE10-NANO
        //inout         [35:0]                  GPIO0,

        // Entradas/Salidas de propósito general banco 0 en la tarjeta DE10-NANO
        //inout         [35:0]                  GPIO1,
        output                                  MCLK,           // gpio_1(25)       PIN_AF21
        output                                  adc_sync,       // gpio_1(24)       PIN_AG20
        input                                   ad0_drdy,       // gpio_1(26)       PIN_AG19
        input                                   ad0_dclk,       // gpio_1(27)       PIN_AH19
        input           [7:0]                   canala,         // Canales digitales del convertidor 0 en formato serie
                //      gpio_1(30)  canala_0                    PIN_AF18
                //      gpio_1(31)  canala_1                    PIN_AF20
                //      gpio_1(28)  canala_2                    PIN_AG18
                //      gpio_1(29)  canala_3                    PIN_AH18
                //      gpio_1(33)  canala_4                    PIN_AE20
                //      gpio_1(32)  canala_5                    PIN_AG15
                //      gpio_1(35)  canala_6                    PIN_AE17
                //      gpio_1(34)  canala_7                    PIN_AE19
        input                                   ad1_drdy,       // gpio_1(1)        PIN_AC24
        input                                   ad1_dclk,       // gpio_1(0)        PIN_Y15
        input           [7:0]                   canalb          // Canales digitales del convertidor 1 en formato serie
                //      gpio_1(5)   canalb_0    PIN_AF28
                //      gpio_1(4)   canalb_1    PIN_AG28
                //      gpio_1(3)   canalb_2    PIN_AD26
                //      gpio_1(2)   canalb_3    PIN_AA15
                //      gpio_1(6)   canalb_4    PIN_AE25
                //      gpio_1(7)   canalb_5    PIN_AF27
                //      gpio_1(8)   canalb_6    PIN_AG26
                //      gpio_1(9)   canalb_7    PIN_AH27
);

        ////////// Definición de parámetros  //////////
        parameter       adFmt           =       16;
        parameter       pipeaddrw       =        7;
//=======================================================
//  REG/WIRE declarations
//=======================================================
  wire                          hps_fpga_reset_n;
  wire [1:0]                    fpga_debounced_buttons;
  wire [7:0]                    fpga_led_internal;
  wire [2:0]                    hps_reset_req;
  wire                          hps_cold_reset;
  wire                          hps_warm_reset;
  wire                          hps_debug_reset;
  wire [27:0]                   stm_hw_events;
  wire                          fpga_clk_50;
  wire                          clk_65;
  wire                          clk_130;
  //wire                                clk_900;
```

```
// Señales internas para la aplicación  Alambrados
  wire                      ad_mclk;                  // Señal MCLK para los convertidores
  wire                      drdy0;                    // Señal drdy del convertidor AD0
  wire                      dclk0;                    // Señal dclk del convertidor AD0
  wire [6:0]                C0_ptr;                   // Señal del puntero
  wire [adFmt - 1:0]        C0_full;
  wire [31:0]               C0_ctl;                   // Señal de control para el tubo
  wire [adFmt - 1:0]        C0_D4;                    // Señal para el dato del canal 4
  wire [adFmt - 1:0]        Q_ch4;
  wire                      clko_ch4;
  wire [adFmt - 1:0]        C0_D5;                    // Señal para el dato deñ canal 5
  wire [adFmt - 1:0]        Q_ch5;
  wire                      clko_ch5;
  wire [adFmt - 1:0]        C0_D6;                    // Señal para el dato del canal 6
  wire [adFmt - 1:0]        Q_ch6;
  wire                      clko_ch6;
  wire [adFmt - 1:0]        C0_D7;                    // Señal para el dato del canal 7
  wire [adFmt - 1:0]        Q_ch7;
  wire                      clko_ch7;

// connection of internal logics
  assign LED[7:0] = fpga_led_internal;// | {7'b0000000, led_level};
  assign fpga_clk_50=FPGA_CLK1_50;
  assign stm_hw_events    = {{15{1'b0}}, SW, fpga_led_internal, fpga_debounced_buttons};
// Conexiones internas del FPGA
  assign   MCLK             = ad_mclk;          // Al pin MCLK asignar la señal de 32.768 MHz
  assign   adc_sync = 1'b1;             // Señal de sincronía siempre en alto
  assign drdy0        = ad0_drdy;       // Al alambre de la señal drdy convertidor 0 asignar el valor del pin
  assign   dclk0      = ad0_dclk;       // Al alambre de la señal dclk convertidor 0 asignar el valor del pin

//=======================================================
//  Structural coding
//=======================================================
I2C_HDMI_Config u_I2C_HDMI_Config (
            .iCLK(FPGA_CLK1_50),
            .iRST_N( 1'b1),
            .I2C_SCLK(HDMI_I2C_SCL),
            .I2C_SDAT(HDMI_I2C_SDA),
            .HDMI_TX_INT(HDMI_TX_INT)
            );

PLL_admclk  PLL_admclk_inst(
            .refclk(fpga_clk_50),   //  refclk.clk
            .rst(1'b0),                         //   reset.reset
            .outclk_0(ad_mclk),                 // outclk0.clk
            .outclk_1( )                              //clk_900)                  // outclk1.clk
);


vga_pll  vga_pll_inst(
            .refclk(fpga_clk_50),   //  refclk.clk
            .rst(1'b0),      //   reset.reset
            .outclk_0(clk_65), // outclk0.clk
            .outclk_1(clk_130), // outclk1.clk
            .locked()    //  locked.export
);

assign HDMI_TX_CLK = clk_65;

soc_system u0 (
           //Clock&Reset
             .clk_clk                       (FPGA_CLK1_50 ),  // clk.clk
             .reset_reset_n                 (hps_fpga_reset_n ),              //                reset.reset_n
             .clk_130_clk                   (clk_130 ),                 //                clk.clk
             .alt_vip_itc_0_clocked_video_vid_clk      (HDMI_TX_CLK ),     // alt_vip_itc_0_clocked_video.vid_clk
             .alt_vip_itc_0_clocked_video_vid_data     (HDMI_TX_D ),     //                .vid_data
             .alt_vip_itc_0_clocked_video_underflow    ( ),    //                .underflow
             .alt_vip_itc_0_clocked_video_vid_datavalid (HDMI_TX_DE), //                .vid_datavalid
             .alt_vip_itc_0_clocked_video_vid_v_sync   (HDMI_TX_VS ),   //                .vid_v_sync
```

```
.alt_vip_itc_0_clocked_video_vid_h_sync    (HDMI_TX_HS ),    //                .vid_h_sync
.alt_vip_itc_0_clocked_video_vid_f         ( ),        //                .vid_f
.alt_vip_itc_0_clocked_video_vid_h         ( ),        //                .vid_h
.alt_vip_itc_0_clocked_video_vid_v         ( ),        //                .vid_v
//HPS ddr3
.memory_mem_a                    ( HPS_DDR3_ADDR),                 //          memory.mem_a
.memory_mem_ba                   ( HPS_DDR3_BA),                   //          .mem_ba
.memory_mem_ck                   ( HPS_DDR3_CK_P),                 //          .mem_ck
.memory_mem_ck_n                 ( HPS_DDR3_CK_N),                 //          .mem_ck_n
.memory_mem_cke                  ( HPS_DDR3_CKE),                  //          .mem_cke
.memory_mem_cs_n                 ( HPS_DDR3_CS_N),                 //          .mem_cs_n
.memory_mem_ras_n                ( HPS_DDR3_RAS_N),                //          .mem_ras_n
.memory_mem_cas_n                ( HPS_DDR3_CAS_N),                //          .mem_cas_n
.memory_mem_we_n                 ( HPS_DDR3_WE_N),                 //          .mem_we_n
.memory_mem_reset_n              ( HPS_DDR3_RESET_N),              //          .mem_reset_n
.memory_mem_dq                   ( HPS_DDR3_DQ),                   //          .mem_dq
.memory_mem_dqs                  ( HPS_DDR3_DQS_P),                //          .mem_dqs
.memory_mem_dqs_n                ( HPS_DDR3_DQS_N),                //          .mem_dqs_n
.memory_mem_odt                  ( HPS_DDR3_ODT),                  //          .mem_odt
.memory_mem_dm                   ( HPS_DDR3_DM),                   //          .mem_dm
.memory_oct_rzqin                ( HPS_DDR3_RZQ),                  //          .oct_rzqin
//HPS ethernet
.hps_0_hps_io_hps_io_emac1_inst_TX_CLK (HPS_ENET_GTX_CLK),
.hps_0_hps_io_hps_io_emac1_inst_TXD0  ( HPS_ENET_TX_DATA[0] ),  //          .hps_io_emac1_inst_TXD0
.hps_0_hps_io_hps_io_emac1_inst_TXD1  ( HPS_ENET_TX_DATA[1] ),  //          .hps_io_emac1_inst_TXD1
.hps_0_hps_io_hps_io_emac1_inst_TXD2  ( HPS_ENET_TX_DATA[2] ),  //          .hps_io_emac1_inst_TXD2
.hps_0_hps_io_hps_io_emac1_inst_TXD3  ( HPS_ENET_TX_DATA[3] ),  //          .hps_io_emac1_inst_TXD3
.hps_0_hps_io_hps_io_emac1_inst_RXD0   ( HPS_ENET_RX_DATA[0] ),
.hps_0_hps_io_hps_io_emac1_inst_MDIO   ( HPS_ENET_MDIO ),       //          .hps_io_emac1_inst_MDIO
.hps_0_hps_io_hps_io_emac1_inst_MDC    ( HPS_ENET_MDC ),        //          .hps_io_emac1_inst_MDC
.hps_0_hps_io_hps_io_emac1_inst_RX_CTL ( HPS_ENET_RX_DV),
.hps_0_hps_io_hps_io_emac1_inst_TX_CTL ( HPS_ENET_TX_EN),
.hps_0_hps_io_hps_io_emac1_inst_RX_CLK ( HPS_ENET_RX_CLK),
.hps_0_hps_io_hps_io_emac1_inst_RXD1   ( HPS_ENET_RX_DATA[1] ),
.hps_0_hps_io_hps_io_emac1_inst_RXD2   ( HPS_ENET_RX_DATA[2] ),
.hps_0_hps_io_hps_io_emac1_inst_RXD3   ( HPS_ENET_RX_DATA[3] ),
//HPS SD card
.hps_0_hps_io_hps_io_sdio_inst_CMD     ( HPS_SD_CMD    ),       //          .hps_io_sdio_inst_CMD
.hps_0_hps_io_hps_io_sdio_inst_D0      ( HPS_SD_DATA[0]    ),   //          .hps_io_sdio_inst_D0
.hps_0_hps_io_hps_io_sdio_inst_D1      ( HPS_SD_DATA[1]    ),   //          .hps_io_sdio_inst_D1
.hps_0_hps_io_hps_io_sdio_inst_CLK     ( HPS_SD_CLK    ),       //          .hps_io_sdio_inst_CLK
.hps_0_hps_io_hps_io_sdio_inst_D2      ( HPS_SD_DATA[2]    ),   //          .hps_io_sdio_inst_D2
.hps_0_hps_io_hps_io_sdio_inst_D3      ( HPS_SD_DATA[3]    ),
.hps_0_hps_io_hps_io_usb1_inst_D0      ( HPS_USB_DATA[0]    ),  //          .hps_io_usb1_inst_D0
.hps_0_hps_io_hps_io_usb1_inst_D1      ( HPS_USB_DATA[1]    ),  //          .hps_io_usb1_inst_D1
.hps_0_hps_io_hps_io_usb1_inst_D2      ( HPS_USB_DATA[2]    ),  //          .hps_io_usb1_inst_D2
.hps_0_hps_io_hps_io_usb1_inst_D3      ( HPS_USB_DATA[3]    ),  //          .hps_io_usb1_inst_D3
.hps_0_hps_io_hps_io_usb1_inst_D4      ( HPS_USB_DATA[4]    ),  //          .hps_io_usb1_inst_D4
.hps_0_hps_io_hps_io_usb1_inst_D5      ( HPS_USB_DATA[5]    ),  //          .hps_io_usb1_inst_D5
.hps_0_hps_io_hps_io_usb1_inst_D6      ( HPS_USB_DATA[6]    ),  //          .hps_io_usb1_inst_D6
.hps_0_hps_io_hps_io_usb1_inst_D7      ( HPS_USB_DATA[7]    ),  //          .hps_io_usb1_inst_D7
.hps_0_hps_io_hps_io_usb1_inst_CLK     ( HPS_USB_CLKOUT    ),   //          .hps_io_usb1_inst_CLK
.hps_0_hps_io_hps_io_usb1_inst_STP     ( HPS_USB_STP    ),      //          .hps_io_usb1_inst_STP
.hps_0_hps_io_hps_io_usb1_inst_DIR     ( HPS_USB_DIR    ),      //          .hps_io_usb1_inst_DIR
.hps_0_hps_io_hps_io_usb1_inst_NXT     ( HPS_USB_NXT    ),
.hps_0_hps_io_hps_io_spim1_inst_CLK    ( HPS_SPIM_CLK ),        //          .hps_io_spim1_inst_CLK
.hps_0_hps_io_hps_io_spim1_inst_MOSI   ( HPS_SPIM_MOSI ),       //          .hps_io_spim1_inst_MOSI
.hps_0_hps_io_hps_io_spim1_inst_MISO   ( HPS_SPIM_MISO ),       //          .hps_io_spim1_inst_MISO
.hps_0_hps_io_hps_io_spim1_inst_SS0    ( HPS_SPIM_SS   ),       //          .hps_io_spim1_inst_SS0

//HPS UART
.hps_0_hps_io_hps_io_uart0_inst_RX     ( HPS_UART_RX   ),       //          .hps_io_uart0_inst_RX
.hps_0_hps_io_hps_io_uart0_inst_TX     ( HPS_UART_TX   ),
.hps_0_hps_io_hps_io_i2c0_inst_SDA     ( HPS_I2C0_SDAT ),       //          .hps_io_i2c0_inst_SDA
.hps_0_hps_io_hps_io_i2c0_inst_SCL     ( HPS_I2C0_SCLK ),
.hps_0_hps_io_hps_io_i2c1_inst_SDA     ( HPS_I2C1_SDAT ),       //          .hps_io_i2c1_inst_SDA
.hps_0_hps_io_hps_io_i2c1_inst_SCL     ( HPS_I2C1_SCLK ),
.hps_0_hps_io_hps_io_gpio_inst_GPIO09  ( HPS_CONV_USB_N ),  //          .hps_io_gpio_inst_GPIO09
.hps_0_hps_io_hps_io_gpio_inst_GPIO35  ( HPS_ENET_INT_N ),  //          .hps_io_gpio_inst_GPIO35
.hps_0_hps_io_hps_io_gpio_inst_GPIO40  ( HPS_LTC_GPIO   ),  //          .hps_io_gpio_inst_GPIO40
```

```
            .hps_0_hps_io_hps_io_gpio_inst_GPIO53  ( HPS_LED   ), //                    .hps_io_gpio_inst_GPIO53
            .hps_0_hps_io_hps_io_gpio_inst_GPIO54  ( HPS_KEY   ), //                    .hps_io_gpio_inst_GPIO54
            .hps_0_hps_io_hps_io_gpio_inst_GPIO61  ( HPS_GSENSOR_INT ),
            .led_pio_external_connection_export    ( fpga_led_internal          ),   //    led_pio_external_connection.export
            .dipsw_pio_external_connection_export  ( SW    ),  //  dipsw_pio_external_connection.export
            .button_pio_external_connection_export ( fpga_debounced_buttons ), // button_pio_external_connection.export
            .hps_0_h2f_reset_reset_n               ( hps_fpga_reset_n ),          //             hps_0_h2f_reset.reset_n
            .hps_0_f2h_cold_reset_req_reset_n      (~hps_cold_reset ),     //       hps_0_f2h_cold_reset_req.reset_n
            .hps_0_f2h_debug_reset_req_reset_n     (~hps_debug_reset
            .hps_0_f2h_stm_hw_events_stm_hwevents  (stm_hw_events ),
            .hps_0_f2h_warm_reset_req_reset_n      (~hps_warm_reset ),
          //FPGA Fabric
            .c0_ctl_external_connection_export     ( C0_ctl ),       //  c0_ctl_external_connection.export
            .c0_full_external_connection_export    ( C0_full ),      //  c0_full_external_connection.export
            .c0_ptr_external_connection_export     ( C0_ptr ),       //  c0_ptr_external_connection.export
            .c0_dat4_external_connection_export    ( C0_D4 ),        //  c0_dat4_external_connection.export
            .c0_dat5_external_connection_export    ( C0_D5 ),        //  c0_dat5_external_connection.export
            .c0_dat6_external_connection_export    ( C0_D6 ),        //  c0_dat6_external_connection.export
            .c0_dat7_external_connection_export    ( C0_D7 )         //  c0_dat7_external_connection.export
 );

// Convertidores de formato serial a paralelo
ConvSerPar_v CSP_Ch4_inst (
                        .dclk     ( dclk0 ),
                        .drdy     ( drdy0 ),
                        .sdi      ( canala[4] ),
                        .Q        ( Q_ch4 ),
                        .clko     ( clko_ch4 ),
                        .sdo      ( )
);
          defparam CSP_Ch4_inst.Formato = adFmt; // Empata el parámetro de formato de conversión con el parámetro del SoC

// Memoria de tipo tubo de 128 registros de ancho definido por adFmt
RamTubo             RamTch4_inst(
                        .clkpush            (clko_ch4),
                        .DataIn             (Q_ch4),
                        .DataOut            ( ),
                        .rdclk              (C0_ctl[4]),
                        .rdena              (1'b1),  //C0_ctl[12]),          // Los bits de habilitación de escritura están en la parte alta
del registro de control canal_ctl del 16 al 31
                        .address            (C0_ptr),
                        .rddata             (C0_D4),
                        .full                       (C0_full[4])
);
          defparam RamTch4_inst.data_width      = adFmt;  // Ancho del tubo en bits
          defparam RamTch4_inst.addr_width = pipeaddrw;

// Convertidores de formato serial a paralelo
ConvSerPar_v CSP_Ch5_inst (
                        .dclk     ( dclk0 ),
                        .drdy     ( drdy0 ),
                        .sdi      ( canala[5] ),
                        .Q        ( Q_ch5 ),
                        .clko     ( clko_ch5 ),
                        .sdo      ( )
);
          defparam CSP_Ch5_inst.Formato = adFmt; // Empata el parámetro de formato de conversión con el parámetro del SoC

// Memoria de tipo tubo de 128 registros de ancho definido por adFmt
RamTubo             RamTch5_inst(
                        .clkpush            (clko_ch5),
                        .DataIn             (Q_ch5),
                        .DataOut            ( ),
                        .rdclk              (C0_ctl[5]),
                        .rdena              (1'b1),  //C0_ctl[13]),          // Los bits de habilitación de escritura están en la parte alta
del registro de control canal_ctl del 16 al 31
                        .address            (C0_ptr),
                        .rddata             (C0_D5),
                        .full                       ( )
);
```

```
        defparam RamTch5_inst.data_width      = adFmt; // Ancho del tubo en bits
        defparam RamTch5_inst.addr_width = pipeaddrw;

// Convertidores de formato serial a paralelo
ConvSerPar_v CSP_Ch6_inst (
                .dclk     ( dclk0 ),
                .drdy     ( drdy0 ),
                .sdi      ( canala[6] ),
                .Q        ( Q_ch6 ),
                .clko     ( clko_ch6 ),
                .sdo      ( )
);
        defparam CSP_Ch6_inst.Formato = adFmt; // Empata el parámetro de formato de conversión con el parámetro del SoC

// Memoria de tipo tubo de 128 registros de ancho definido por adFmt
RamTubo         RamTch6_inst(
                .clkpush        (clko_ch6),
                .DataIn         (Q_ch6),
                .DataOut        ( ),
                .rdclk          (C0_ctl[6]),
                .rdena          (1'b1),  //C0_ctl[14]),          // Los bits de habilitación de escritura están en la parte alta
del registro de control canal_ctl del 16 al 31
                .address        (C0_ptr),
                .rddata         (C0_D6),
                .full                   ( )
);
        defparam RamTch6_inst.data_width      = adFmt; // Ancho del tubo en bits
        defparam RamTch6_inst.addr_width = pipeaddrw;

// Convertidores de formato serial a paralelo
ConvSerPar_v CSP_Ch7_inst (
                .dclk     ( dclk0 ),
                .drdy     ( drdy0 ),
                .sdi      ( canala[7] ),
                .Q        ( Q_ch7 ),
                .clko     ( clko_ch7 ),
                .sdo      ( )
);
        defparam CSP_Ch7_inst.Formato = adFmt; // Empata el parámetro de formato de conversión con el parámetro del SoC

// Memoria de tipo tubo de 128 registros de ancho definido por adFmt
RamTubo         RamTch7_inst(
                .clkpush        (clko_ch7),
                .DataIn         (Q_ch7),
                .DataOut        ( ),
                .rdclk          (C0_ctl[7]),
                .rdena          (1'b1),  //C0_ctl[15]),          // Los bits de habilitación de escritura están en la parte alta
del registro de control canal_ctl del 16 al 31
                .address        (C0_ptr),
                .rddata         (C0_D7),
                .full                   ( )
);
        defparam RamTch7_inst.data_width      = adFmt; // Ancho del tubo en bits
        defparam RamTch7_inst.addr_width = pipeaddrw;

// Debounce logic to clean out glitches within 1ms
debounce debounce_inst (
  .clk          (fpga_clk_50),
  .reset_n      (hps_fpga_reset_n),
  .data_in      (KEY),
  .data_out      (fpga_debounced_buttons)
);
  defparam debounce_inst.WIDTH = 2;
  defparam debounce_inst.POLARITY = "LOW";
  defparam debounce_inst.TIMEOUT = 50000;               // at 50Mhz this is a debounce time of 1ms
  defparam debounce_inst.TIMEOUT_WIDTH = 16;            // ceil(log2(TIMEOUT))

// Source/Probe megawizard instance
hps_reset hps_reset_inst (
  .source_clk (fpga_clk_50),
```

```
  .source     (hps_reset_req)
);

altera_edge_detector pulse_cold_reset (
  .clk       (fpga_clk_50),
  .rst_n     (hps_fpga_reset_n),
  .signal_in (hps_reset_req[0]),
  .pulse_out (hps_cold_reset)
);
  defparam pulse_cold_reset.PULSE_EXT = 6;
  defparam pulse_cold_reset.EDGE_TYPE = 1;
  defparam pulse_cold_reset.IGNORE_RST_WHILE_BUSY = 1;

altera_edge_detector pulse_warm_reset (
  .clk       (fpga_clk_50),
  .rst_n     (hps_fpga_reset_n),
  .signal_in (hps_reset_req[1]),
  .pulse_out (hps_warm_reset)
);
  defparam pulse_warm_reset.PULSE_EXT = 2;
  defparam pulse_warm_reset.EDGE_TYPE = 1;
  defparam pulse_warm_reset.IGNORE_RST_WHILE_BUSY = 1;

altera_edge_detector pulse_debug_reset (
  .clk       (fpga_clk_50),
  .rst_n     (hps_fpga_reset_n),
  .signal_in (hps_reset_req[2]),
  .pulse_out (hps_debug_reset)
);
  defparam pulse_debug_reset.PULSE_EXT = 32;
  defparam pulse_debug_reset.EDGE_TYPE = 1;
  defparam pulse_debug_reset.IGNORE_RST_WHILE_BUSY = 1;

endmodule
```