UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ACCIONAMIENTO Y RECEPCIÓN DE SEÑALES ULTRASÓNICAS PARA UN ECÓGRAFO ULTRA-PORTATIL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRICISTA

MANUEL ENRIQUE TOLEDO CHAMORRO

PROFESOR GUÍA:
SR. NICOLÁS BELTRÁN MATURANA

MIEMBROS DE LA COMISIÓN:
SR. HECTOR AGUSTO ALEGRÍA
SR. MANUEL DUARTE MERMOUD

SANTIAGO DE CHILE
ABRIL 2012

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO
DE INGENIERO CIVIL ELECTRICISTA
POR: MANUEL TOLEDO CHAMORRO
FECHA: 25 DE ABRIL DE 2012
PROF.GUÍA: SR. NICOLÁS BELTRÁN MATURANA

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ACCIONAMIENTO Y RECEPCIÓN DE SEÑALES ULTRASÓNICAS PARA UN ECÓGRAFO ULTRA-PORTATIL"

Hoy en día es imposible imaginar no contar con la disponibilidad de acceso a exámenes médicos, que permitan la visualización de órganos y tejidos mediante sistemas no invasivos y de fácil aplicación, como son los ecógrafos. Sin embargo, en el presente se buscan más aplicaciones a este versátil método de examinación, como lo es la construcción de dispositivos ultra-portátiles, que permitan la ejecución de exámenes rutinarios de imagenología, pero en espacios reducidos y con un equipo energizado por baterías. Estadísticamente, es sabido que del total de las ecografías solicitadas; solo el 30% son realmente necesarias, por lo tanto, la inclusión de equipos de rápido acceso y de categoría portable, es un eslabón deseable en el proceso de diagnóstico médico. Es por ello que en esta memoria se plantea el diseño y fabricación de una plataforma electrónica que permita el manejo completo de un transductor ultrasónico, para realizar la construcción de este tipo de dispositivos.

En este documento se describen los fundamentos, procedimientos y resultados obtenidos en la implementación de un sistema de accionamiento y recepción de señales, para la generación de imágenes de ultrasonido. En esta dirección, se presentan diseños, simulaciones, implementaciones y la construcción de diversas plataformas que otorgaron el aprendizaje necesario para la confección de un sistema de alta tecnología con condiciones de estado del arte en pulsación ultrasónica. Estas iteraciones, son directa consecuencia de una búsqueda estricta de optimizaciones para la plataforma definitiva. Adicionalmente, se presenta todo el desarrollo y análisis necesario para lograr la puesta en marcha de un transductor ultrasónico activo con multiplexión local. Esto, con el fin de poder utilizar un dispositivo de bajo costo "*after market*", en contraposición a la expectativa de fabricar un transductor ultrasónico personalizado con altos costos de inversión y construcción, asociado a la baja cantidad de unidades a producir en relación al mercado mundial abastecido por China. Finalmente, se expone la generación de una completa plataforma digital basada, que permite capturar los datos salientes de un sistema de conversión análogo/digital con interfaz serializada, grabar temporalmente en una memoria de acceso aleatorio de alta velocidad, y finalmente volcar datos a un servidor donde se realizará la conformación.

Tanto los desarrollos electrónicos como los de software, fueron íntegramente diseñados partiendo desde cero, contando únicamente con los conocimientos recabados durante una larga etapa de investigación. El diseño, cálculo y simulaciones de las plataformas, la confección de los circuitos impresos y el montaje de los componentes fue realizado en los laboratorios del Departamento de Ingeniería Eléctrica de la Universidad de Chile, en el contexto del proyecto Corfo Innova 09IEI-7183.

Los resultados obtenidos permiten concluir que se completó de forma íntegra la cadena de diseño, partiendo desde el estudio de la tecnología actual, llegando a la construcción de un dispositivo que se puede considerar estado del arte en el ámbito de pulsación de piezoeléctricos para equipos de ultrasonografía. Se logró obtener una plataforma funcional, eficiente y con gran robustez a los ciclos de trabajo estrictos a los que fue sometido.

"…Sé tu mismo.
Especialmente no finjas afectos.
Tampoco seas cínico respecto al amor,
porque frente a toda aridez y desencanto,
el amor es tan perenne como la hierba.
Acepta con cariño el consejo de los años,
renunciando con elegancia a las cosas de juventud.
Nutre la fuerza de tu espíritu para que te proteja en la inesperada desgracia,
pero no te angusties con fantasías.
Muchos temores nacen de la fatiga y la soledad.
Más allá de una sana disciplina,
sé amable contigo mismo.
Eres una criatura del universo,
al igual que los árboles y las estrellas;
tienes derecho a estar aquí.
Y, te resulte o no evidente,
sin duda el universo se desenvuelve como debe.
Por lo tanto, mantente en paz con Dios,
de cualquier modo que le concibas,
y cualesquiera sean tus trabajos y aspiraciones.
Mantente en paz con tu alma en la ruidosa confusión de la vida."

Desiderata, Max Ehrmann.

Agradecimientos

Este documento es el ícono que cierra un proceso de largos 18 años de estudio, en los cuales busqué hambriento las herramientas para poder darle al mundo mi pasión, mi arte.

El camino recorrido hasta aquí está entregando un gran fruto y trascenderá como un granito de conocimiento más entregado a la humanidad, sin embargo este gran fruto del que me siento orgulloso no habría sido posible sin la existencia del motor de mi pasión: mi gente. Por eso he decidido dedicar estas líneas a quienes también les pertenece este trabajo.

Antes que a nadie, quiero agradecer a Dios, pues es la razón de hacer, la razón de creer, la razón de sentir y la razón de amar.

A mis padres, mi padre Manuel Toledo por su sabiduría y amor, por la disciplina que inculcó, y que siempre llevaré en mi corazón; a mi madre Verónica Chamorro, por su preocupación y su afecto, por entregarlo todo sin medida, por entregarme amor en cada segundo de su vida. Por el esfuerzo y el cariño que ambos depositaron siempre en mi, muchas gracias.

A mi hermano Patricio Toledo, mi favorito y mi elegido, un luchador del cual me siento inmensamente orgulloso y que cada día admiro más. Recuerda que siempre estaré contigo.

A mi amada, Rocío Clavijo, mi compañera de vida, la inspiración, motivación y razón de mis sueños; su amor, comprensión y respeto son mi tesoro más preciado, y el motor de mis ideales.

A mi madrina Nina, María Yolanda Chamorro por su afecto y por sus constantes plegarias para que Dios me acompañara.

A mis buenos amigos, Eduardo Ubilla, Claudio Tapia, Matías Cerda y Benjamín Olivares por su cariño y preocupación hacia mí y hacia mi familia.

A mis guías y compañeros en este fantástico proyecto: Profesor Nicolás Beltrán, Profesor Manuel Duarte, Rodrigo Maureira, Vader Johnson y Javier Moya.

Y por supuesto al Profesor Marcos Orchard, un verdadero ícono en mi vida universitaria que recordaré por siempre.

Índice Generales

Agradecimientos 4
Índice Generales 5
Índice de Figuras 7
Capítulo 1 Introducción 9
1.1 Motivación 9
1.2 Hipótesis de trabajo y metodología 10
1.3 Objetivos generales 10
1.4 Objetivos específicos 10
1.5 Estructura de la memoria 11
Capítulo 2 Fundamentos Generales 12
2.1 Funcionamiento básico de un equipo de ultrasonido 12
2.2 Transductores ultrasónicos 15
2.2.1 Piezoeléctricos 16
2.2.2 Contextualización de los piezoeléctricos en el proyecto 17
2.3 Multiplexión 18
2.3.1 Contextualización de la multiplexión en el proyecto 19
2.4 Pulsación 20
2.4.1 Transistores MOSFET para conmutación 20
2.4.2 Estado del arte de los sistemas de pulsación 21
2.4.3 Contextualización de la pulsación en el proyecto 24
2.5 Conector 26
2.6 Etapa de adquisición de señales 27
2.6.1 FPGA 27
2.6.2 Pares diferenciales LVDS 28
2.7 Etapa de procesamiento de señales 29
2.8 Dominio y aporte del trabajo 30
Capítulo 3 Diseño e implementación de sistemas. 31
3.1 Diseño e implementación de sistema de pulsación Pulser v.1.0 31
3.1.1 Drivers de corriente para MOSFETs 31
3.1.2 Translación de referencias y pulsadores 32
3.1.3 Confección de PCBs 33
3.2 Análisis y puesta en marcha del sistema de multiplexión 36
3.2.1 Situación inicial del sistema de multiplexión 36
3.2.2 Diseño e implementación del sistema de control para la multiplexión 38
3.3 Diseño e implementación del sistema de pulsación Pulser v.2.0 42
3.3.1 Reemplazo de transistores MOSFET 42
3.3.2 Confección de PCB 43
3.4 Migración de sistemas de control a FPGA 45
3.4.1 Operación quasi-asíncrona del sistema de pulsación. 46
3.5 Diseño e implementación del sistema de pulsación Pulser v.2.5 48
3.5.1 Simulación e inclusión del sistema de clamping 51
Capítulo 4 Resultados, fabricación y pruebas de nuevas versiones, y desarrollos en recepción 54
4.1 Resultados de implementación para el Pulser v.2.5 54
4.2 Diseño de sistema de pulsación Pulser v.3.0 55
4.2.1 Circuito integrado de pulsación 56
4.2.2 Confección de PCB para Pulser v.3.0 57
4.2.3 Resultados de operación para el Pulser v.3.0 59

4.3 Desarrollo de núcleos para recepción digital 61
4.3.1 SERDES LVDS 63
4.3.2 SRAM 63
4.3.3 RS232 64
4.3.4 Implementación del esquema final completo 65
Capítulo 5 Conclusiones Generales y Trabajo Futuro 67
5.1 Conclusiones Generales 67
5.2 Trabajo Futuro 68
Referencias 70
Anexos 73
Anexo 1: Diagramas de sistema de pulsación "Pulser v.1.0" 73
Anexo 2: Código C para 16F877A decodificador de multiplexión 75
Anexo 3: Esquema funcional de plataforma de control para multiplexión y pulsación. 76
Anexo 4: Módulo de comandos abstractos de multiplexión y pulsación 77
Anexo 5: Módulo de interrelación y coherencia de datos 78
Anexo 6: Módulo de interpretación de direccionamiento. 78
Anexo 7: Módulo de comunicación con los multiplexores. 85
Anexo 8: Módulo Verilog de pulsación asíncrona 90
Anexo 9: Núcleo para utilización y pruebas del módulo de manejo para SRAM 92
Anexo 10: Núcleo para utilización del módulo de transmisiones RS232. 96
Anexo 11: Código de software de recepción para el host en Visual Basic. 101
Anexo 12: Códigos de módulos y núcleos para el funcionamiento del sistema de adquisición completo. 104

Índice de Figuras

Figura 1: (Izquierda) Imagen original, (Derecha) Imagen procesada a través de un Scan Converter.. 14
Figura 2: Diagrama de bloques general de operación de un ecógrafo ultra-portatil..... 15
Figura 3: Transductor lineal de 4 cm. de excursión. 15
Figura 4: Transductor ultrasónico de 80 elementos, con conector convencional..... 16
Figura 5: Trozo de cerámica perovskita en bruto. 17
Figura 6: Fotografía del cabezal piezoeléctrico utilizado. 17
Figura 7: Esquema básico de funcionamiento de un multiplexor. 18
Figura 8: Fotografías de la situación interna del transductor ultrasónico..... 19
Figura 9: Modelo Físico Electrónico de un transistor MOSFET. 20
Figura 10: Circuito clásico de configuración push-pull. 21
Figura 11: Esquema de pulsación unipolar clásico usado. 22
Figura 12: Arquitectura electrónica de un pulser bipolar. 23
Figura 13: Topología de pulsación con clamper..... 24
Figura 14: Arquitectura de pulsación propuesta en las notas de aplicación de Supertex® Inc. 25
Figura 15: Arquitectura Propuesta por National Semiconductor ®. 25
Figura 16: Comparativa de tamaños para los conectores de transductores. 26
Figura 17: Encapsulado FPGA que se usará para los desarrollos. 28
Figura 18: Esquema de funcionamiento para un sistema de transmisión LVDS..... 29
Figura 19: Configuración de salidas paralelas para obtención de mayor Fan-Out..... 31
Figura 20: Esquemático del circuito para un pulsador (Versión 1.0) 32
Figura 21: Modelo de la PCB para construcción del Pulser v1.0..... 33
Figura 22: Espacio de trabajo inicial que entrega CadSoft Eagle para rutear las conexiones entre componentes. 34
Figura 23: Fotografías de la PCB del Pulser v.1.0 34
Figura 24: Imágenes del montaje de componentes del Pulser v.1.0..... 35
Figura 25: Topología del Analog Switch ECN3290TF de HITACHI®. 36
Figura 26: Circuitos integrados de multiplexión ECN3290TF. 37
Figura 27: Imagen que representa la identificación manual de pines del transductor. 37
Figura 28: Montaje para detección e identificación de posición geométrica de elementos. 38
Figura 29: Esquema de distribución de pines para el cabezal transductor. 38
Figura 30: Esquema de implementación para pruebas de multiplexión. 39
Figura 31: Fotografía del montaje de pruebas para decodificación de la multiplexión. 40
Figura 32: Dibujo CAD de la placa para el sistema de pulsación Pulser v.2.0. 43
Figura 33: Imagen de la construcción y armado de la plataforma de pulsación Pulser v.2.0. 44
Figura 34: Forma de onda para plataforma Pulser v.2.0. 45
Figura 35: Plataforma MORPH IC de FTDI® para desarrollos en FPGA..... 45
Figura 36: Esquema básico del funcionamiento de la plataforma digital de control. 46
Figura 37: Máquina de estados para el proceso de pulsación. 47
Figura 38: Esquema de topología equivalente para simulación electrónica. 48
Figura 39: Resultados de simulación inicial del Pulser v.2.5 49
Figura 40: Corriente de drenaje vs. Tensión de compuerta para un MOSFET. 50
Figura 41: Resultados para nuevas resistencias de limitación de corriente. 50
Figura 42: Efecto de inercia presente en la carga piezoeléctrica. 51
Figura 43: Inclusión de sistema de clamping a simulación. 51
Figura 44: Resultados obtenidos con la inclusión de un sistema de clamping. 52
Figura 45: Fotografías de implementación de la plataforma de pulsación Pulser v.2.5. 52
Figura 46: Evolución del modelo electrónico equivalente del transductor ultrasónico. 53

Figura 47: Resultados de pulsación y multiplexión con plataforma Pulser v.2.5. 55
Figura 48: Diagrama funcional del IC de pulsación MAX4940A. 56
Figura 49: Fotografía del circuito integrado de pulsación MAX4940A. 57
Figura 50: Vista preliminar del diseño CAD para el Pulser v.3.0. 57
Figura 51: Fracción de la PCB correspondiente a la plataforma de la pulsación Pulser v.3.0. 58
Figura 52: Fotografía de la implementación de la plataforma de pulsación Pulser v.3.0. 58
Figura 53: Prueba del sistema de pulsación Pulser v.3.0 con una carga resistiva. 59
Figura 54: Prueba de pulsación para el Pulser v.3.0 con carga piezoeléctrica. 60
Figura 55: Resultados de pulsación para el Pulser v.3.0 con frecuencia de operación a 7 [MHz]. 60
Figura 56: Resultados de pulsación y recepción de ecos satisfactoria. 61
Figura 57: Fotografía de la plataforma de desarrollo ALTERA® DE2. 62
Figura 58: Esquema relacional de cores a desarrollar para interfaz digital de recepción. 62
Figura 59: Captura de programa de recepción de datos en host. 64
Figura 60: Detalle de un módulo de pulsación para la plataforma Pulser v.1.0. 73
Figura 61: Esquemático general para la plataforma de pulsación Pulser v.1.0. 74
Figura 62: Diagrama de bloques global del sistema de multiplexión y pulsación asíncrono. 76
Figura 63: Esquema funcional del módulo de interrelación y coherencia de datos. 78

Capítulo 1

Introducción

1.1 Motivación

El acceso a exámenes de diagnóstico en los sistemas de salud chileno es muchas veces prohibitivo respecto de sus costos, efecto que se provoca por una demanda inelástica y además equipos y mano especializada extremadamente costosa. Particularmente, para el caso de Chile, todo el equipamiento es importado principalmente desde Estados Unidos y Europa, con marcas de alto prestigio que comercializan equipos de calidad, y que por lo tanto tienen costos muy elevados; esto hace que los equipos no estén en todos lados, por lo tanto para ofertas reducidas y demandas inelásticas, el parámetro que se dispara es el precio.

Hoy en día es fundamental, el acceso a exámenes que ayuden a la generación de un diagnóstico por el médico tratante, pero este tipo de procedimientos para los usuarios de la salud pública son lentos por la demanda de ellos en un sistema con oferta limitada. Por esta razón, los pacientes terminan esperando meses para hacerse un examen.

Aquí nace la idea del pre-diagnóstico, especialmente dada la estadística que señala, que del universo de exámenes de ultrasonido que se ejecutan, solo el 30% eran realmente necesarios; que pasaría entonces si se pudiera reducir ese 70% remanente. Una disminución de la demanda en ambos sistemas (público y privado), claramente reduce los costos por procedimiento.

La idea de este proyecto, se enmarca en suplir la necesidad de disponer de instrumental de pre-diagnóstico de bajo costo, orientado a generar información relevante en el primer examen del paciente. Así, se propone la construcción de un escáner de ultrasonido de bajo costo para examinación rápida, lo cual automáticamente permite tomar decisiones al médico respecto del paso siguiente en el tratamiento de un paciente. Por ejemplo, para un paciente que presenta aumento del volumen abdominal se puede asumir que:

- Es producto de acumulación de grasa
- Tiene un embarazo
- Acumulación anormal de líquido

Para un médico que no tiene acceso inmediato a equipamiento de imagenología, solo le queda enviar al paciente a realizarse el examen correspondiente; por otro lado, si éste tuviera acceso a un equipo de inspección rápida, podría desarrollar mucho mejor el siguiente paso y aliviar la demanda de los laboratorios de examinación detallada, como los departamentos de imagenología.

Además de ser un equipo de bajo-costo, este debe tener condiciones de portabilidad, de manera tal que pueda ser utilizado en servicios móviles de urgencia, salas de atención primaria, consultorios y hasta consultas particulares. En este contexto se propone el diseño de un ecógrafo ultra-portátil operado por baterías que pueda ser manejado por médicos que no sean necesariamente imagenólogos.

1.2 Hipótesis de trabajo y metodología

Este trabajo postula que es posible implementar una plataforma integral de multiplexión, pulsación y recepción de señales ultrasónicas, en un dispositivo pequeño y de muy bajo-consumo basándose en los productos que actualmente se comercializan en el mundo. En este sentido, se dedicará especial cuidado a adoptar e implementar estrategias de diseño que eleven la eficiencia y permitan la miniaturización de los esquemas clásicos de construcción de ecógrafos.

Se utilizará la mayor cantidad de ingeniería aplicada posible, con énfasis en el empleo de circuitos integrados especializados en los diseños. Adicionalmente, se tendrá acceso a equipamiento de examinación ultrasónica funcional, para su respectivo análisis, permitiéndose incluso el desarme de estos para lograr una visión mucho más cercana a la filosofía de diseño que actualmente se utiliza en este tipo de equipos médicos. Finalmente, se buscará construir módulos electrónicos de alta densidad, que realicen las tareas antes mencionadas de forma específica, para más tarde poder integrarlos en una única solución de manejo de señales de ultrasonido.

1.3 Objetivos generales

En esta memoria se persiguen los siguientes objetivos generales:

- Análisis y puesta en marcha de un sistema de multiplexión estándar para un transductor de ultrasonido, logrando obtener el know-how para realizar el diseño y construcción de un multiplexor personalizado para la aplicación.
- Diseño y construcción de un módulo de pulsación de ultrasonido, que cumpla con los requerimientos de bajo-consumo y alta eficiencia impuestos por la portabilidad y operación desde baterías.
- Diseño e implementación del módulo digital de recepción de señales ultrasónicas para su posterior procesamiento.

1.4 Objetivos específicos

De los conceptos antes mencionados se pueden desprender los siguientes objetivos específicos:

- Investigación de topología y arquitectura electrónica utilizada para la realización de la multiplexión de forma activa en el transductor ultrasónico.
- Decodificación de la estructura de instrucciones para realizar un direccionamiento de multiplexión.

- Investigación los fenómenos y efectos generados, incluyendo las variaciones en los modelos electrónicos equivalentes, debida la inclusión de la multiplexión, el cable y el conector del transductor ultrasónico.
- Investigación del estado del arte en sistemas de pulsación para equipos de ultrasonografías médicas.
- Diseñar, simular e implementar una plataforma de pulsación operativa.
- Iterar sobre los diseños de la plataforma de pulsación para optimizar los consumos eléctricos, los tamaños y la eficiencia.
- Planificación de módulos digitales a diseñar, para importar datos obtenidos a alta velocidad hasta un computador host.
- Determinar la necesidad de inclusión de memorias de acceso aleatorio (RAM), para la captura de datos a alta velocidad.
- Diseñar medio de comunicación estándar para volcar los datos en un computador host.

1.5 Estructura de la memoria

Capítulo 1 : Introducción

Se muestra la motivación de la realización del proyecto, la hipótesis en la que se sostiene, la metodología de trabajo y los objetivos que este desarrollo persigue.

Capítulo 2 : Fundamentos Generales

En este capítulo se expone el contexto teórico en el que se basa el trabajo a desarrollar, se investiga el estado del arte relacionado y se describe el aporte que este tiene.

Capítulo 3 : Diseño e implementación de sistemas.

En esta sección se muestran los diseños y análisis realizados para lograr implementar una plataforma de pulsación y multiplexión funcional. Se incluyen optimizaciones como la inclusión de clampers a los módulos y la migración de los sistemas distribuidos a núcleos en una FPGA condensada.

Capítulo 4 : Resultados, fabricación y pruebas de nuevas versiones, y desarrollos en recepción

En este capítulo se exponen los resultados obtenidos de la versión distribuida más operacional y se muestran los resultados de la última plataforma de pulsación implementada, la cual posee circuitos integrados dedicados para las tareas propuestas. Adicionalmente se documenta la generación de todo el módulo digital de recepción de señales ultrasónicas, el cual está basado en el diseño de núcleos para su implementación en una FPGA.

Capítulo 5 : Conclusiones Generales y Trabajo Futuro

Se presentan las conclusiones del trabajo realizado, y las tareas que deben realizarse para concluir el diseño del producto final en contexto del proyecto.

Capítulo 2

Fundamentos Generales

En este capítulo, se introducirán los temas que fundamentan y contextualizan este trabajo de investigación y desarrollo. Hoy en día es imposible imaginar no contar con la disponibilidad de acceso a exámenes médicos, que permitan la visualización de órganos y tejidos mediante sistemas no invasivos y de fácil aplicación como son los ecógrafos. Sin embargo, en el presente se buscan más aplicaciones a este versátil método de examinación, como lo es la construcción de dispositivos ultra-portátiles que permitan la ejecución de exámenes rutinarios de imagenología, pero en espacios reducidos y con un equipo energizado por baterías.

Estadísticamente se sabe que del total de las ecografías solicitadas; solo el 30% eran realmente necesarias, por lo tanto, la inclusión de equipos de rápido acceso y de categoría portable, es un eslabón deseable en el proceso de diagnóstico médico. De esta forma se plantea el diseño y fabricación de una plataforma electrónica que permita el manejo completo de un transductor ultrasónico para realizar la construcción de este tipo de dispositivos.

El problema que se abordará será el diseño e implementación de la electrónica requerida para la multiplexión, pulsación y recepción de datos de un transductor activo como arreglo de múltiples elementos piezoeléctricos. Dentro de este desarrollo se mantendrá un exhaustivo control del estado del arte en este aspecto, ya que no solo debe desarrollarse un sistema funcional, sino que adicionalmente se debe lograr miniaturizar todos los diseños para ajustarlos a los requerimientos de portabilidad. Es también de vital importancia, considerar la eficiencia de éstos y sus consumos respectivos, ya que el dispositivo no solo será portátil, sino que además será operado mediante baterías recargables, por lo que debe cumplir con ciertas restricciones mínimas de autonomía.

2.1 Funcionamiento básico de un equipo de ultrasonido

Un ecógrafo, es un equipo de uso médico que utiliza características específicas del fenómeno acústico para generar imágenes interpretables, donde se pueden visualizar tejidos y órganos, permitiendo de esta forma diagnosticar condiciones que no son identificables a simple vista. El equipo antes mencionado se basa en la condición física de reflexión de las ondas acústicas en medios materiales; en este caso, se produce un frente de ondas ultrasónico a través de un transductor

piezoeléctrico el cual es excitado a una frecuencia muy específica[1], dependiendo de la arquitectura, geometría y prestaciones de cada sonda ecográfica. A partir de este momento, un frente de ondas acústicas penetra los tejidos del cuerpo, los cuales irán generando de forma coherente ecos, dependiendo de sus capacidades de reflexión y absorción, estos ecos vuelven al transductor, luego de un intervalo de tiempo directamente proporcional a la distancia del punto donde se reflejó la onda incidente.

El transductor genera entonces un trabajo mecánico el cual provoca la aparición de un frente de onda, luego este queda por unos instantes en estado de reposo, y a continuación debe recibir señales de ecos en magnitudes muy inferiores a las de emisión. Asumiendo una pequeña distancia, como es, la superficie del transductor y la zona muerta de éste, se puede estimar una distancia mínima de 3[mm], por lo tanto se recorrerá a lo menos 6[mm]; por otro lado, las máximas distancias que manejan los ecógrafos en general no superan los 250[mm], por lo que tenemos una cota superior de recorrido igual a 500[mm]. De esta forma se pueden calcular las magnitudes de tiempo que se deben manejar, para lograr capturar correctamente todos los ecos resultantes.

$$v_{sonido\ en\ tejidos} \cdot t = 2 \cdot d_{propagación}$$

$$\rightarrow t_{mín} = \frac{2 \cdot d_{propagación}}{v_{sonido}} = \frac{2 \cdot 3[mm]}{1540000[\frac{mm}{s}]} = 3.8961[\mu s]$$

$$\rightarrow t_{máx} = \frac{2 \cdot d_{propagación}}{v_{sonido}} = \frac{2 \cdot 250[mm]}{1540000[\frac{mm}{s}]} = 324.675[\mu s]$$

En el tiempo especificado se deben capturar las ondas resultantes, que son convertidas a señales eléctricas mediante los mismos transductores que antes emitieron el frente de ondas; estas señales eléctricas son 2 órdenes de magnitud más pequeñas que las de emisión, por lo que la arquitectura electrónica que debe manejar ambas magnitudes debe ser adecuadamente diseñada[2]. A continuación, se deben adquirir las señales para ser llevadas a un ambiente digital donde se puede aplicar todo el procesamiento para transformar señales crudas de eco en una imagen ultrasónica interpretable; la etapa de adquisición consiste fundamentalmente en conversores análogo-digital, de muy bajo ruido, gran profundidad de bits, pre-amplificadores incorporados, alta frecuencia de muestreo y una salida digital serializada de alta velocidad para el gran flujo de datos que se debe manejar.

Con los datos en formato digital, se procede al proceso de conformación, el cual está compuesto por las siguientes etapas:

- Se pre-filtra la señal con un filtro pasa-banda centrado en la frecuencia de pulsación, dejando así solo las componentes en torno a esta frecuencia, y atenuando el ruido y los artefactos.

[1] El rango de frecuencias utilizadas para ultrasonografías médicas esta comprendido entre 1 y 10 [MHz]
[2] Nótese que las tensiones asociadas a la señal de emisión y recepción, las cuales distan en 2 órdenes de magnitud, están presentes en la misma ruta eléctrica.

- Considerando la forma y la inercia de las señales es posible realizar un proceso de up-sampling, el cual se lleva a cabo introduciendo ceros intercalados en la señal y luego filtrando la resultante con un filtro pasa-bajo, el cual se encarga de realizar una interpolación a la señal original, pero con una pseudo-frecuencia de muestreo mucho mayor.

- Con una matriz de retardos pre-calculada que introduce consideraciones geométricas y físicas respecto de cada elemento piezo-eléctrico, se pueden escoger estratégicamente datos de la señal, recuperando la frecuencia de muestreo original, pero con suficiente precisión como para calcular retardos en puntos no muestreados.

- Finalmente, se realiza un proceso de detección de envolvente, el cual es representable de forma directa en lo que se llama un "*scanline*" o línea de ultrasonido, la cual es visible en una ecografía como una línea vertical o descriptora de profundidad. Es un hecho que la aplicación de un algoritmo estándar de detección de envolventes no es suficiente para obtener imágenes de calidad media, lo que genera la necesidad de implementar esquemas más complejos como la Transformada de Hilbert.

La unión de estas líneas de forma secuencial una al lado de la anterior es finalmente el despliegue de la imagen; para un transductor de geometría lineal el posicionamiento de líneas adyacentes es representativo de la realidad, pero para un transductor convexo se debe realizar una conversión de coordenadas y aproximaciones de renderización[1] en base a interpolaciones bilineales, lo cual aumenta la complejidad de todo el problema de despliegue.

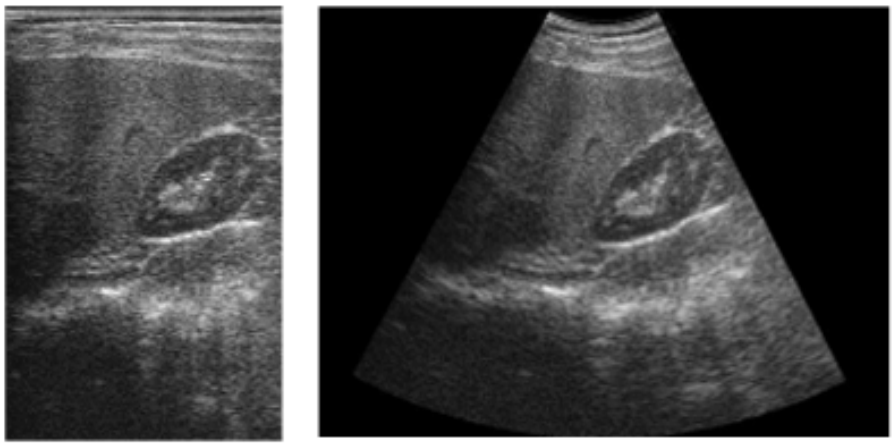

Figura 1: (Izquierda) Imagen original, (Derecha) Imagen procesada a través de un Scan Converter

En la Figura 1, se puede revisar la imagen obtenida luego de la etapa de conformación, pero sin la inclusión del efecto geométrico de que los datos fueron rescatados a través de un transductor ultrasónico convexo y no lineal; dado esto, se debe aplicar un algoritmo de cambio de coordenadas y renderización Scan Converter, el cual se encarga de generar una nueva imagen con las relaciones de aspecto y distancias correctas. Se debe señalar que la regeneración de esta imagen de forma adecuada[2], permite realizar mediciones en pantalla, que reflejan distancias reales en los tejidos examinados.

[1] Proceso de generar una imagen a partir de un modelo matemático.
[2] Del aspecto de una imagen generada por un transductor lineal al de una generada por un transductor convexo.

Un esquema general de todo el proceso de captura de imágenes ultrasónico en grandes bloques funcionales se puede visualizar en la Figura 2.

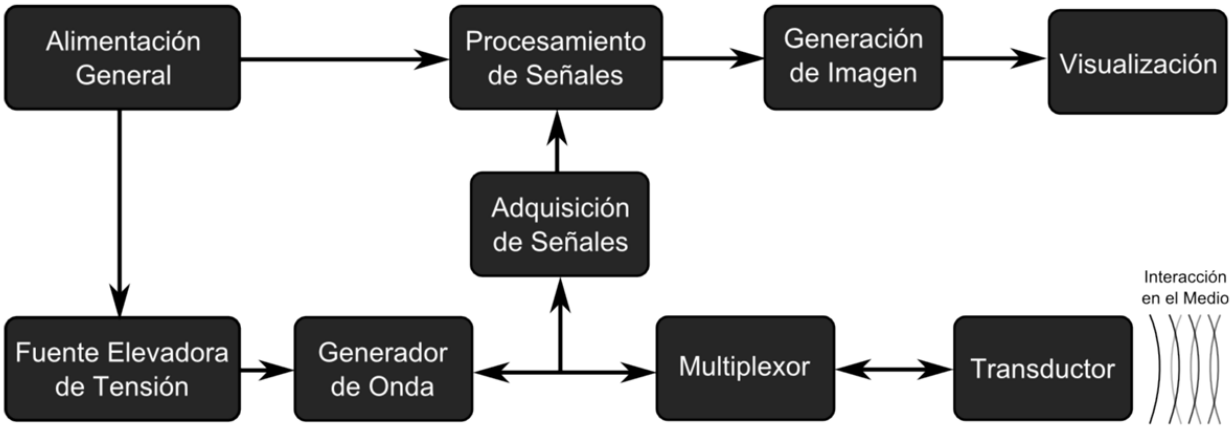

Figura 2: Diagrama de bloques general de operación de un ecógrafo ultra-portatil

2.2 Transductores ultrasónicos

Los transductores ultrasónicos para ecografías de examinación humana de hoy en día consisten, por lo general, en un arreglo unidimensional de piezoeléctricos equiespaciados que permiten el escaneo en una dimensión lateral y una dimensión de profundidad[1]. En general, los arreglos son cantidades estándares de 80, 128, 192 y hasta 256 elementos; estas especificaciones permiten manejar la resolución axial de las ultrasonografías, esto dado que existen modelos de transductores con la misma excursión espacial, pero con distintas cantidades de elementos.

Figura 3: Transductor lineal de 4 cm. de excursión.

El transductor que se muestra en la Figura 3, corresponde a uno de geometría lineal con excursión de exploración de 4 cm; para este modelo, existen distintas arquitecturas de arreglos de piezoeléctricos, con versiones desde 80 hasta 256 elementos, esto implica que para la misma distancia de exploración se pueden obtener diversas resoluciones axiales.

Los ecógrafos tipo carro de examinación rutinaria, son por lo general equipos de gran tamaño que pueden tener soporte de hasta 6 transductores, con sus respectivos sistemas de interconexión. En este aspecto, todos los transductores ultrasónicos están conformados por 3 partes fundamentales:

- Empuñadura, donde se encuentra el arreglo de piezoeléctricos soportado por una estructura que permite su fácil manipulación y direccionamiento.

[1] Generando por lo tanto, una imagen en 2 dimensiones.

- Cable, el cual por lo general tiene una medida de 1.5 metros, cumple con requerimientos de aislación eléctrica y además está compuesto de materiales de alta flexibilidad y resistencia para soportar el estrés de tracción y torsión al que está sometido diariamente.

- Conector, por lo general estructuras grandes y robustas, con inserciones de tipo ZIF[1], en las cuales se encuentran expuestas las conexiones de cada elemento piezoeléctrico(dos polos).

En el caso de un transductor ultrasónico lineal para examinación de baja penetración de 256 elementos, el conector posee un mínimo de 512 contactos, lo que lo hace fuertemente incompatible con aplicaciones portátiles, dado su gran tamaño y peso; en este sentido se deben hacer importantes adaptaciones al sistema tradicional de obtención de datos a través de una gran cantidad de celdas piezoeléctricas, migrándolo a un esquema donde se pueda negociar de forma eficiente el tamaño del conector y la cantidad de elementos disponibles en paralelo.

Figura 4: Transductor ultrasónico de 80 elementos, con conector convencional.

2.2.1 Piezoeléctricos

Los elementos piezoeléctricos son dispositivos pasivos de materiales que son capaces de convertir ondas mecánicas en eléctricas y viceversa, fenómeno en el que esta basada la técnica médica de imagenología del ultrasonido. Estos elementos pueden ser sintonizados a frecuencias preestablecidas, dependiendo de los materiales de construcción, forma y su disposición geométrica.

Para el caso de los transductores ultrasónicos, estos se construyen conectados mecánicamente a un bloque de amortiguación el cual debe tener una impedancia acústica muy cercana al del material piezoeléctrico, ya que de esta forma se suprimen los fenómenos de resonancia y por lo tanto permite aumentar la fidelidad y ancho de banda del sistema. Adicionalmente, la arquitectura incluye una capa de protección y adaptación en el frente de éste, la cual asegura que se transmita la mayor cantidad de energía acústica desde el medio hacia cada elemento; por otro lado, también protege los transductores de daños mecánicos o químicos.

Respecto de la construcción, los materiales usados en los primeros años del ultrasonido como el cuarzo, sulfato de litio o el titanato de bario ya no son implementados hoy en día; en efecto, nuevos y más poderosos materiales están disponibles con características acústicas y eléctricas muy diversas. Este fenómeno se debe a que en el presente es posible generar compuestos cristalinos que

[1] Zero Insertion Force, conectores que para su instalación solo deben ser puesto frente al placar de conexiones y luego girar un cerrojo que genera la fuerza para mantener unidas ambas partes.

posean características de sensibilidad, estabilidad y costo objetivo, por lo que es posible desarrollar un tipo de transductor para cada aplicación particular dependiendo de las restricciones del problema.

La gran masa de fabricantes de transductores para ultrasonografías generan sus dispositivos con Zirconato-Titanato de Plomo, el cual es una cerámica perovskita con grandes características piezoeléctricas y una incomparablemente alta eficiencia del 52%. Posee una de las impedancias acústicas más altas dentro del espectro de los piezocerámicos para frecuencias de resonancia menor a 25 [MHz], y es estable a temperaturas de operación de hasta unos 365ºC.

Figura 5: Trozo de cerámica perovskita en bruto.

2.2.2 Contextualización de los piezoeléctricos en el proyecto

Dadas las circunstancia industriales de fabricación presentes en Chile, no es posible contar con fabricantes de componentes y partes tan específicas como son los arreglos de piezoeléctricos para equipos de ultrasonografía. La fabricación de esta tecnología requiere de maquinaria muy especializada, la cual permite laminar de forma muy precisa estas cerámicas, logrando afinar su frecuencia de resonancia en la deseada para la generación de ondas ultrasónicas del espectro de interés.

De esta forma, se debió conseguir tecnología de fabricación en el extranjero para poder acceder a los transductores necesarios para el desarrollo del trabajo. En efecto, parte de las metas iniciales de este proyecto consistió en la generación de una cartera de contactos de donde adquirir este tipo de componentes.

La motivación a realizar esto, está fundamentada especialmente en la aceleración que recibe el proceso de investigación y desarrollo; ya que no solo se busca bibliografía y literatura asociada, si no que además se cuenta con la tecnología que actualmente se desarrolla, permitiendo dilucidar muchas interrogantes en un tiempo muy reducido.

Figura 6: Fotografía del cabezal piezoeléctrico utilizado.

En la Figura 6: Fotografía del cabezal piezoeléctrico utilizado.Figura 6 se puede visualizar una fotografía del cabezal transductor utilizado para el desarrollo de este trabajo, este consiste en un arreglo convexo unidimensional de 80 elementos piezocerámicos calibrados a una frecuencia de máxima sensibilidad en 3.5 [MHz]. Este transductor es uno de los más clásicos utilizados para examinación de órganos abdominales e incluso torácicos, ya que su penetración puede llegar a unos 25 [cm] de profundidad con resoluciones cercanas a 1 [mm].

En la imagen presente a la izquierda es posible notar las conexiones exteriores de cada elemento piezoeléctrico montados en un sustrato amortiguador, por otro lado en el lado derecho es posible visualizar la capa de adaptación acústica y protección contra daño mecánico y químico de color gris.

2.3 Multiplexión

El proceso mediante el cual se puede tener acceso a un conjunto amplio de canales a través de un número reducido de estos mismos, se llama "multiplexión"; este consiste en una topología electrónica donde se seleccionan las señales entrantes mediante un bus selector, el cual indica que parte de la entrada se transmitirá por los canales de salida (Figura 7).

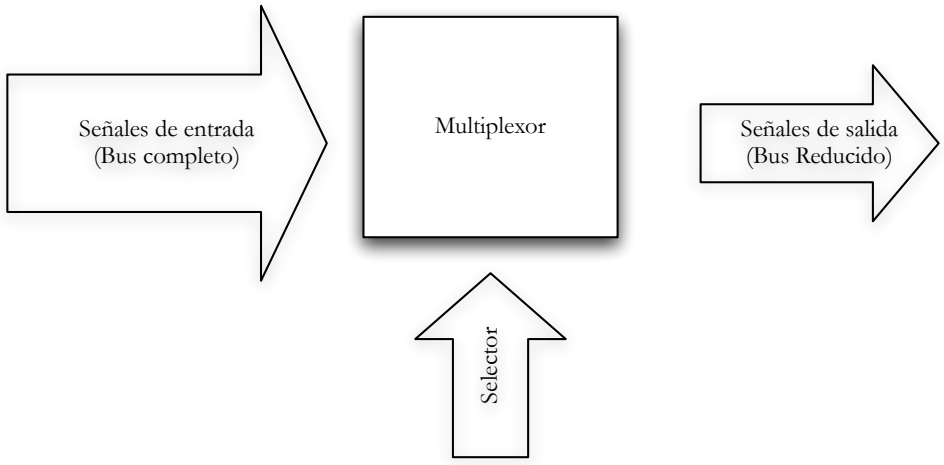

Figura 7: Esquema básico de funcionamiento de un multiplexor.

Este procedimiento es realizado electrónicamente mediante transistores MOSFET, funcionando en modo de corte y saturación como interruptores. La idea principal consiste en 2 capas fundamentales: la primera corresponde al arreglo de transistores que permiten hacer converger todas las señales deseadas a un bus de ancho definido, los cuales se disponen de forma óptima para lograr impedancias de interrupción iguales en todos los canales; la segunda etapa, corresponde al control de estos transistores, es decir, las señales que encienden o apagan cada bifurcación para hacer llegar la señal entrante al puerto de salida deseado. Por lo general esta capa de control no es un selector de entrada paralela, sino que por lo general son entradas serializadas que permiten realizar configuraciones de varios bits en solo un pin de entrada del encapsulado. La arquitectura de manejo de este tipo de configuraciones es un *stack* FIFO[1], por lo tanto se deben ingresar los bits de forma ordenada y síncrona a través de un reloj, el cual en cada flanco de subida empuja un dato nuevo al

[1] FIFO: Sistemas de encolamiento First-In-First-Out, donde el primer elemento que entra al arreglo, es el primero que sale.

arreglo; finalmente cuando el dato está completamente determinado y posicionado en el buffer interno de la etapa de control, se genera un flanco de bajada en una señal de "latch"[1] para entregar este bus de datos estabilizados a la etapa de disparo de los transistores. Todo lo mencionado implica directamente que se debe incorporar una maquinaria completa de lógica electrónica, para lograr operar el selector de multiplexión, y por supuesto también, las señales necesarias para la pulsación.

Finalmente se debe considerar que este tipo de sistemas de multiplexión e interrupción deben tener resistencias de conexión muy bajas, pues deben adaptar señales de gran excursión como son los pulsos ultrasónicos, los cuales operan entre +100[V] y -100[V]; y por otro lado, otorgar mínima atenuación a los ecos de retorno, los cuales son unas 100 veces más pequeños que las señales de pulsación.

2.3.1 Contextualización de la multiplexión en el proyecto

Dados los proveedores disponibles se adquirió un transductor completo, lo que implica el cabezal transductor, la empuñadura, el cable de conexión y el conector principal. El transductor especificado no es pasivo, es decir, posee electrónica que realiza labores de multiplexión en el mismo dispositivo.

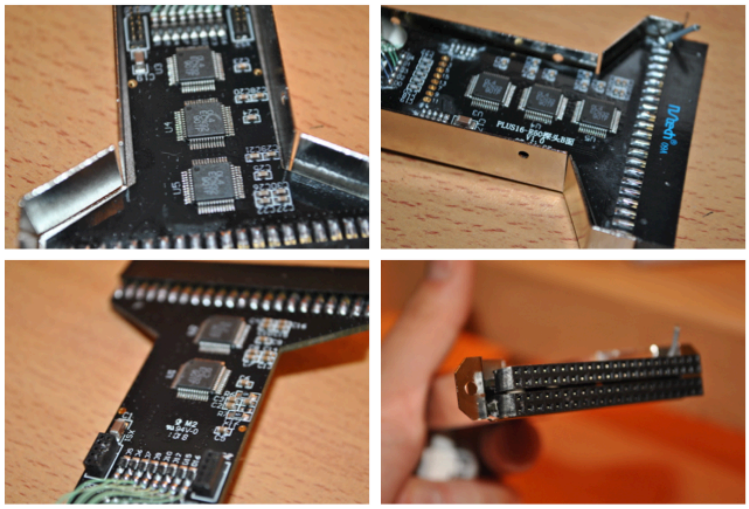

Figura 8: Fotografías de la situación interna del transductor ultrasónico.

En la Figura 8 se muestra la situación del interior del transductor, particularmente en la zona de la empuñadura de la sonda, donde se albergan las tarjetas electrónicas.

Si bien, este hecho genera grandes ventajas respecto al aumento en la velocidad del desarrollo dado que se tiene un transductor activo con multiplexión listo para operar, se tiene la gran desventaja que implica no tener información respecto del funcionamiento y la arquitectura, ya que son dispositivos de consumo ya producidos que justamente intentan proteger sus diseños, ocultando su funcionamiento interno.

[1] Esquema asíncrono clásico de electrónica digital que permite guardar de forma estable un dato entrante en un cierto instante.

Se apuesta entonces a la decodificación de la arquitectura y las comunicaciones para poder usar este dispositivo como instrumento, sin tener que estar re-fabricándolo con tecnología mas cara y escasa. Esto constituirá también, uno de los desafíos que incluye este trabajo en su desarrollo.

2.4 Pulsación

Esta etapa consiste en la generación de pulsos ultrasónicos a nivel eléctrico correctamente adaptados, logrando de esta forma excitar de forma adecuada cada elemento piezoeléctrico en el arreglo. Este corresponde a un proceso no-trivial, ya que se deben generar pulsos de más de 1 [MHz] de frecuencia con niveles de excursión de hasta 200 [Vpp], esto genera directamente dos temáticas asociadas: en primer lugar, se deben realizar diseños con transistores de muy baja capacitancia de compuerta y grandes niveles de tensión *drenador-fuente*; por otro lado se deben incluir drivers de bajos tiempos de propagación y altas corrientes de salida, las cuales alimentarán las compuertas de los MOSFETs.

2.4.1 Transistores MOSFET para conmutación

Los transistores que se utilizan para fines de pulsación son deseablemente pequeños, esto es dados los beneficios físicos electrónicos que tiene en este tipo de aplicaciones. Un modelo muy clásico de los transistores MOSFETs es la vista como un interruptor semi-conductor con una compuerta aislada y que tiene características de capacitor, esto es dado que si se aplica un voltaje positivo[1] a la compuerta, se estará creando un campo eléctrico entre esta y el resto del sustrato del transistor; cuando la tensión aplicada en la compuerta es suficientemente alta para superar la barrera de potencial generada por el sustrato tipo P, se genera un canal de conducción, apareciendo un flujo de corriente desde el drenador (*drain*) hacia la fuente (*source*), generando un camino de baja impedancia que se asocia a un interruptor cerrado.

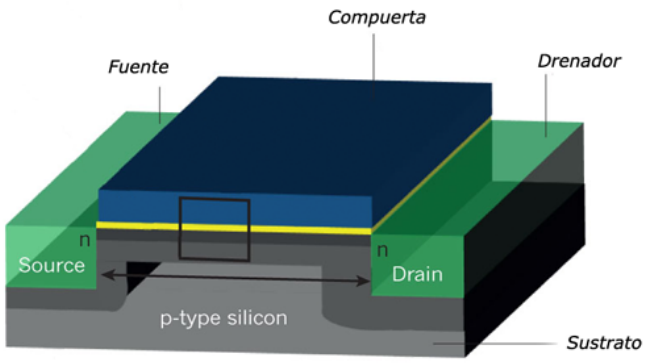

Figura 9: Modelo Físico Electrónico de un transistor MOSFET.

El modelo antes mencionado es representable mediante capacitancias[2], particularmente se puede establecer que la tensión "*gate-source*" V_{gs} va directamente relacionada con el campo eléctrico producido, a su vez la tensión V_{gs} es proporcional a la resistencia de conducción y por lo tanto a la condición de encendido del transistor. Si esta curva se recorre lentamente, se generarán grandes pérdidas por conmutación, dado que la tensión "*drain-source*" V_{DS} instantánea será considerablemente alta, la cual es directamente proporcional con la potencia disipada sobre el transistor; este efecto no

[1] En este caso a un transistor MOSFET canal N
[2] Particularmente por los fenómenos electrostáticos presentes.

es deseable y se puede evitar recorriendo la curva de carga muy rápido. Se conoce la ecuación de corriente como:

$$I = \frac{q}{t}$$

donde "q" corresponde a la carga inyectada y "t" el tiempo que toma, luego si se desea mover grandes cantidades de carga en un tiempo reducido, una forma directa de realizarlo es aumentar la corriente impuesta a la compuerta. Para esto se requiere de una fuente de corriente controlada electrónicamente que permita encender rápidamente el transistor en cuestión por un tiempo preestablecido.

El concepto de fuente de corriente electrónicamente controlada antes mencionado, se puede implementar mediante un MOSFET driver, el cual en estricto rigor corresponde a un dispositivo que entrega corriente en términos de una entrada digital. Estos por lo general consisten en un arreglo de transistores dispuestos en configuración "*push-pull*", la cual mediante dos transistores conectados en serie es capaz de entregar corriente y absorber corriente.

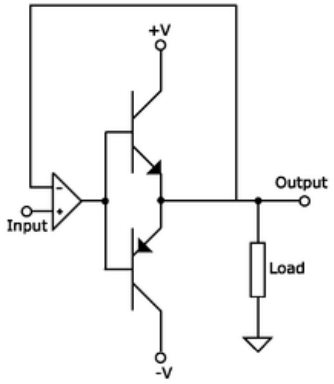

Figura 10: Circuito clásico de configuración push-pull.

Según se menciona anteriormente, es de suma importancia inyectar carga de forma muy rápida en la compuerta, ya que de esta manera es posible encender en corto tiempo el transistor, evitando así pérdidas por conmutación y sus ineficiencias e inestabilidades asociadas; de esta misma forma, es también vital poder apagar rápidamente el MOSFET. Esto es realizado mediante la absorción o drenaje de corriente desde la compuerta, lo cual es directamente logrado habilitando un camino de baja impedancia entre la compuerta y su nodo de referencia a la fuente del transistor. Con esta configuración driver-mosfet, es posible controlar de forma eficiente el encendido y apagado del transistor con una señal de carácter digital, la cual será comandada desde la plataforma de procesamiento.

2.4.2 Estado del arte de los sistemas de pulsación

Con estos antecedentes es posible hacer un análisis del estado del arte respecto a pulsación de ultrasonido, donde se generan otros fenómenos que deben ser analizados, particularmente dadas las características eléctricas del modelo del cristal piezoeléctrico.

La arquitectura planteada, compuesta por un *pulser* basado en transistores MOSFET, un sistema de driving para el encendido eficiente y su correspondiente transductor, es la más básica, y es llamada arquitectura de "*pulser* unipolar". Tal como se menciona en Haider 2006[1][1], el elemento piezoeléctrico se puede modelar como un capacitor, por lo tanto este se carga cuando es energizado por el sistema de pulsación. La desventaja que presenta este hecho es que cada elemento tiene un comportamiento capacitivo y por lo tanto tiempos de respuesta perturbados por las distorsiones en frecuencia y en fase que generan cargas pasivas con estas características. En efecto, tal como se menciona en la publicación, la mayoría de los sistemas de pulsación unipolares existentes deben usar sistemas de drenaje para lograr un comportamiento adecuado.

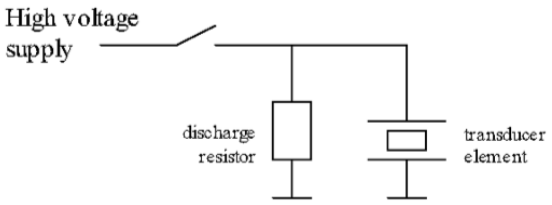

Figura 11: Esquema de pulsación unipolar clásico usado.

La necesidad de inclusión de la resistencia de descarga tiene un único sentido, y es quitar inercia a la carga del sistema de pulsado. Este fenómeno se requiere, ya que si bien la generación y desvanecimiento del canal de conducción del transistor MOSFET es dependiente de la cantidad de carga electrostática presente en la compuerta, es también fundamental, para que se cumpla esta dinámica, que exista una carga R_L correctamente adaptada en la salida del transistor. Esto efectivamente no se cumple con cargas tan capacitivas como un elementos transductor ultrasónico, por lo que la única forma de adaptarlo de forma pasiva es agregando un sistema de drenaje.

El problema que conlleva esta solución es que, si bien efectivamente se cumple el requisito, es un sistema altamente ineficiente. Esto dada la diferencia abismante de impedancias en régimen de corriente continua presentes; por un lado el modelo de impedancia equivalente del transductor ultrasónico a 3.5 [MHz] es de aproximadamente 2 [kΩ] resistivos en paralelo a una capacitancia de 300 [pF], equivalente a una impedancia de 952 [Ω] a la frecuencia antes mencionada; por otro lado, en cambio, la resistencia de drenaje para una operación correcta, debe ser de valores cercanos a los 25 [Ω]. Así es posible realizar de forma directa el siguiente cálculo en régimen de operación.

$$I_{transductor} = \frac{100\ [V]}{2000\ [\Omega] - j \cdot 952[\Omega]} = 0.0411 + 0.01905 \cdot j\ [A]$$

$$\rightarrow |I_{transductor}| = 0.04538\ [A]$$

Por otro lado, si calculamos la corriente circulante en el conjunto paralelo:

$$I_{conjunto} = \frac{100\ [V]}{(2000\ [\Omega] - j \cdot 952[\Omega])\ ||\ 25[\Omega]} = 4.04118 + 0.019050 \cdot j\ [A]$$

$$\rightarrow \left|I_{conjunto}\right| = 4.04123\ [A]$$

[1] B. Haider, "*Power Drive Circuits for Diagnostic Medical Ultrasound*", 18th International Symposium on Power Semiconductor Devices & IC's, Naples, Italy, June 2006.

Es posible concluir que la corriente del conjunto es una 100 veces mayor a la del transductor de forma aislada, esto implica que casi el 99% de la energía esta destinada a alimentar la resistencia de drenaje y no el elemento que se busca energizar. Si bien, esto funciona y permite un apagado rápido de los transistores de pulsación, es un esquema que desperdicia mucha energía, la cual es valiosa en equipos e implementaciones portátiles.

La configuración antes mencionada es utilizada en aproximadamente el 20% de los sistemas de ultrasonido, ya que permite desarrollar arquitecturas electrónicas muy sencillas y funcionales, sin embargo, presenta una gran desventaja respecto de la sub-utilización de la excursión total de cada elemento transductor. Los elementos piezoeléctricos poseen un estado de régimen permanente en el cual su deformación es nula y por lo tanto su tensión lo es también, pero dada la naturaleza elástica encontrada, el fenómeno es simétrico respecto del estado de reposo, lo que implica que la excursión máxima es alcanzada cuando se lleva a estados de máxima-compresión y máxima-tracción. Este efecto mecánico en términos eléctricos, corresponde a la aplicación de tensiones positivas y negativas al sistema piezocerámico, por lo que al adoptar una arquitectura con fuentes unipolares, solo estaremos usando una fracción de la excursión máxima del transductor.

Dado lo anterior, es que el 70% del mercado de los equipos de ultrasonido usa la siguiente estructura electrónica para sus sistemas de pulsación.

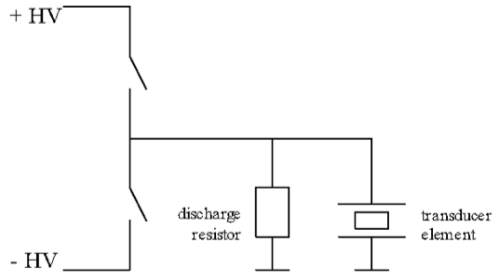

Figura 12: Arquitectura electrónica de un pulser bipolar.

En este caso claramente se alimenta el transductor con tensiones positivas y negativas, lo que genera directamente mayor excursión, ondas más estables y por lo tanto ecos de mejor calidad y amplitud. Aún así, el sistema sigue incluyendo una resistencia de drenaje que otorga bajísimas eficiencias al sistema, que para la gran masa de equipos comercializables no es un problema, ya que operan conectados directamente a la red eléctrica del recinto.

Así y dadas todas las condiciones anteriores, en [1][1] se sugiere una nueva arquitectura que usan los fabricantes más innovadores y que efectivamente maneja esta gran ineficiencia.

[1] B. Haider, "*Power Drive Circuits for Diagnostic Medical Ultrasound*", 18th International Symposium on Power Semiconductor Devices & IC's, Naples, Italy, June 2006.

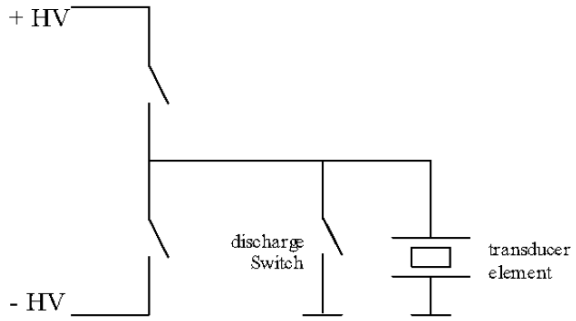

Figura 13: Topología de pulsación con clamper.

El objetivo en este esquema es incluir la resistencia de drenaje, solo cuando ésta es necesaria, es decir, en la etapa de apagado; particularmente se establece una conexión de descarga de forma controlada por una fracción muy pequeña de tiempo justo después del término de la última señal de pulsación.

Para esto se utiliza una arquitectura, como la mostrada en la Figura 13 donde en paralelo al transductor piezoeléctrico (de la misma forma que estaba conectada la resistencia de drenaje), se conecta un transistor de conmutación, que permitirá generar de forma controlada un camino de baja impedancia cuando se desee eliminar la inercia inherente del sistema capacitivo del piezoeléctrico.

La idea fundamental detrás la revisión de este modelo, es que este esquema sea simulado, para probar su implementabilidad, y si es factible la construcción de un circuito de esta categoría, entonces se tendrán reducciones sustanciales en el consumo basal del sistema de pulsación de ultrasonido, siendo esto un interesante resultado dentro del desarrollo de la memoria; particularmente dadas las condiciones de portabilidad implicadas en este proyecto.

2.4.3 Contextualización de la pulsación en el proyecto

Las arquitecturas utilizadas en el equipo que se inspeccionó para la fluidez del desarrollo de la investigación, no cumplían con los requisitos de tamaño, consumo y eficiencia que el planteamiento inicial de portabilidad consideraba, por lo tanto se descartó su análisis para así abrir camino a una amplia línea de investigación, diseño, desarrollo e implementación que efectivamente corresponde a uno de los objetivos fundamentales de esta memoria.

Para los desarrollos se comienza la investigación a partir de notas de aplicación y uso de componentes electrónicos específicos para la generación de pulsos ultrasónicos de examinación médica, donde se presenta una opción proveniente de Estados Unidos respecto a fabricación de circuitos integrados para su uso en la construcción de ecógrafos. La ventaja corresponde en este caso, a un fabricante que entrega diseños de arquitecturas para la implementación de sistemas de pulsación y recepción de señales ultrasónicas médicas.

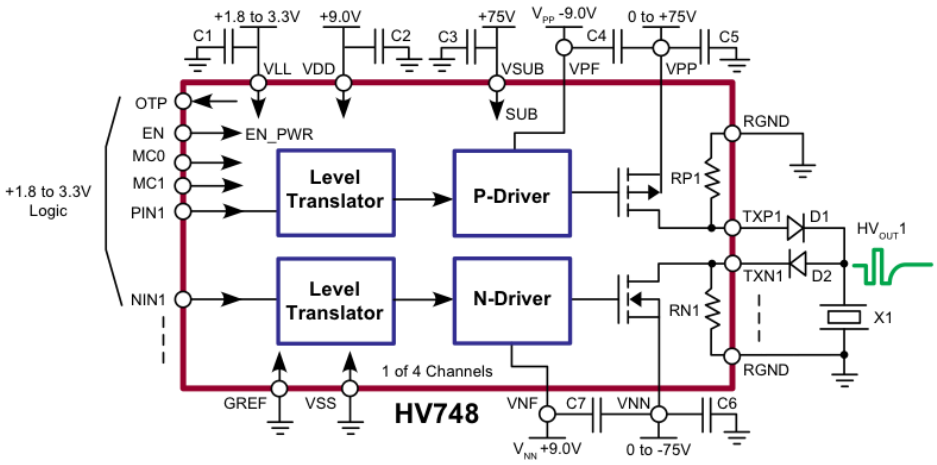

Figura 14: Arquitectura de pulsación propuesta en las notas de aplicación de Supertex® Inc.

Acceder a esta información resulta útil, pues entrega datos importantes respecto de la arquitectura y las consideraciones que se deben tener al diseñar un sistema de pulsación ultrasónica. Particularmente la propuesta de Supertex®[1] consiste en una aplicación basada en transistores MOSFET en configuración "push-pull", sus respectivas etapas de disparo con levantadores de referencia, y sistemas de protección y drenaje para la carga equivalente al cristal piezoeléctrico. Estos planteamientos permiten consolidar algunos conceptos y mejorar las hipótesis de planificación de diseño, por ejemplo la inclusión de una resistencia de drenaje es decidor respecto de las arquitecturas usualmente utilizadas y como estas son implementadas en equipos vigentes.

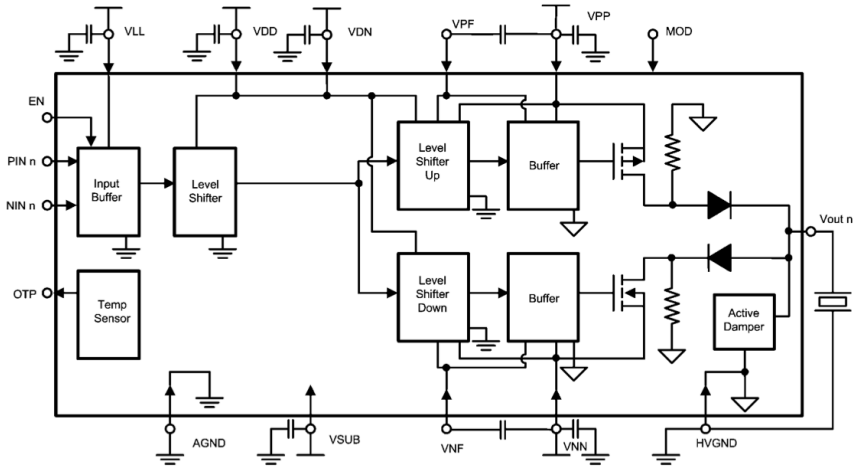

Figura 15: Arquitectura Propuesta por National Semiconductor ®.

En la Figura 15, ya es posible ver la inclusión de sistemas de clamper que permiten el drenaje eficiente de las corrientes almacenadas temporalmente en los cristales ultrasónicos, sin embargo es considerado como un accesorio y sigue viéndose la presencia de resistencias pasivas de adaptación de carga. Respecto de la etapa de potencia, esta se considera del tipo "push-pull", con toda la maquinaria previa que permite su correcto funcionamiento.

Dentro de la investigación se identifica un componente de estado-del-arte de la pulsación para ecógrafos desarrollado por la compañía MAXIM Integrated Products® el cual cuenta con todas

[1] Información disponible en www.supertex.com

las prestaciones esperadas por el equipo a desarrollar, sin embargo al ser un componente de categoría "prototipo", debe ser lanzado y puesto a disposición, para luego comercializarlo, por lo que la disponibilidad de este circuito integrado es incierta. Aún así se abre una hebra de trabajo donde se incluye el circuito integrado MAX4940[1] como solución integral del problema de pulsación. Así mismo, se enrutan los diseños a realizar con esta concepción, ya que cumple las restricciones del problema y además posee grandes muestras de ser un esquema óptimo para la solución que se pretende desarrollar.

2.5 Conector

Aunque pudiese parecer poco importante, uno de los flancos a atacar en el contexto de este proyecto es la miniaturización del sistema que conecta el transductor a través de su cable con el equipo propiamente tal.

El estándar de los transductores ultrasónicos es fabricado para escáneres de ultrasonografía estacionarios, los cuales además de estar conectados a la red eléctrica del recinto permanentemente, también tienen mucho espacio para el hardware, puesto que su arquitectura es muy similar a la de un mueble. Esto implica por lo general que los ecógrafos tienen hasta 8 entradas para distintos transductores con distintas geometrías y frecuencias de operación, lo cual ocupa espacios considerables, ya que en función de mantener las mejores resoluciones posibles, no existe etapa de multiplexión y por lo tanto los elementos se encuentran uno a uno alambrados hasta el equipo de ultrasonido; así, suponiendo un transductor clásico de 128 elementos, el conector tendrá a lo menos 256 pines que deben tener una distancia considerable para su correcto funcionamiento y aislación de alta tensión. Estas características hacen que los transductores convencionales para ecógrafos estacionarios tengan tamaños de magnitudes como : 12 [cm] x 10 [cm] x 6 [cm]; algo absolutamente inviable para los diseños planificados, pues la idea es que el producto final completo tenga dimensiones similares a esas.

Este problema fue resuelto mediante la utilización de transductores construidos para equipos portátiles los cuales incluyen multiplexión y por lo tanto, el cable y su conector son varias veces más pequeños.

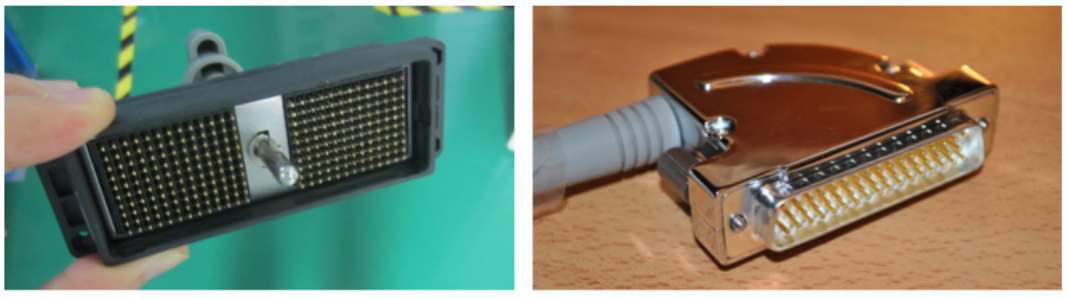

Figura 16: Comparativa de tamaños para los conectores de transductores.

En la Figura 16, se puede ver a la izquierda solo el frontal de un conector para un transductor pasivo de ecógrafo estacionario, al cual aún le falta el ensamblado de el chasis de éste el cual es metálico, pesado y muy grande; en el lado derecho de la figura es posible ver el conector de un

[1] Dual/Quad, Unipolar/Bipolar, High-Voltage Digital Pulsers (www.maxim-ic.com)

transductor activo y multiplexado, el cual fue diseñado para un ecógrafo portátil tipo "laptop" y que tiene dimensiones muy inferiores, del orden de : 5 [cm] x 4 [cm] x 1.2 [cm].

Finalmente, cabe mencionar que no solo tiene ventajas respecto de los tamaños, si no que además el hecho de ser de procedencia China, hace que el costo del conjunto transductor, cable, conector; sea una diez veces menor. Este hecho, determina la conveniencia de usar la sonda tal como viene desde el origen.

2.6 Etapa de adquisición de señales

Una vez que los pulsos de alta tensión se disparan desde los piezoeléctricos, estos viajan por los tejidos y rebotan generando ecos que volverán eventualmente al transductor, estos ecos ultrasónicos deben ser capturados, ya que son las señales que poseen toda la información del entorno, incluyendo densidades y profundidades. El proceso de adquisición analógica es algo directo una vez que se tiene implementada la multiplexión, pero también se debe realizar la conversión digital y la captura de estos datos.

El sistema completo posee a lo menos 80 elementos piezoeléctricos, los cuales serán multiplexados, escogiendo 16 del conjunto completo; los datos provenientes de estos 16 elementos deben ser capturados, convertidos y procesados en tiempos del orden de los milisegundos, ya que esto se debe realizar varias veces tal que se recorra el total de los elementos. Adicionalmente se necesita reiteración puesto que las imágenes de ultrasonido son en tiempo real, y por lo tanto las imágenes se deben refrescar de forma periódica y frecuente de modo tal que se pueda generar el efecto de video.

Las 16 señales que se reciben en paralelo deben ser convertidas a datos digitales de alta precisión con 12 bits de resolución, y además se deben capturar varias miles de muestras en cada disparo, pues estas dirán donde están los obstáculos que encontró el frente de ondas ultrasónicas emitido. Todo este enorme caudal de información debe ser llevado hasta la plataforma digital escogida, para poder realizar la conformación. En efecto, los flujos de información son suficientemente grandes como para impedir la utilización de plataformas de procesamiento secuenciales como micro-procesadores, en su defecto, se hace uso de sistemas de procesamiento paralelo como son las FPGAs. Adicionalmente, se debe contar con esquemas electrónicos de comunicación adecuados para este flujo de datos, como lo son los pares diferenciales LVDS, los cuales serán expuestos más adelante.

2.6.1 FPGA

Las FPGAs (Field-Programmable Gate Array) son circuitos integrados diseñados para ser configurados y programados por los desarrolladores. Estas son configuradas a través de lenguajes de descripción de hardware[1] y son usadas para implementar cualquier tipo de función lógica que un circuito integrado de aplicación específica (ASIC)[2] pueda realizar. Sus grandes ventajas corresponden a la factibilidad de cambiar su configuración y funcionalidades, y por su puesto a las cualidades de procesamiento paralelo que puede lograr.

[1] HDL: Hardware Description Language.
[2] ASIC: Application-Specific Integrated Circuit.

Estos dispositivos contienen lógica programable basada en los llamados "elementos lógicos", y una jerarquía de interconexiones reconfigurables que permiten a los bloques comunicarse entre ellos; de esta forma estos bloques lógicos pueden ser agrupados para conformar funciones combinacionales complejas o sistemas de memoria volátil.

Estos circuitos integrados, no solo se usarán en las etapas de adquisición y procesamiento, sino que además se implementarán desarrollos en FPGAs para la etapa de pulsación y multiplexión, particularmente para las señales de control digital que estos bloques funcionales requieren. Algunas de las razones por las cuales se genera este concepto, es por su gran versatilidad y las capacidades de optimización de energía, dada la naturaleza de dispositivos dedicados que estos tienen.

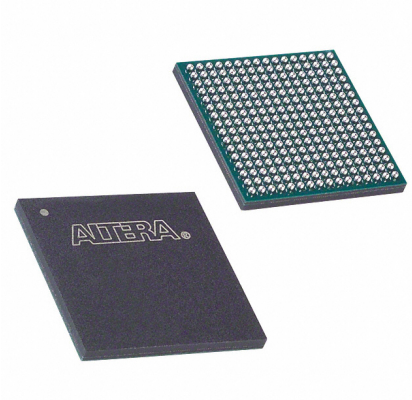

Figura 17: Encapsulado FPGA que se usará para los desarrollos.

2.6.2 Pares diferenciales LVDS

Los pares diferenciales LVDS (low-voltage differential signaling) son un estándar de transmisión de señales digitales, que puede lograr grandes velocidades de transferencia a través de medios de muy bajo costo como los pares trenzados de cobre. El hecho de que sean diferenciales implica que transmite información como la diferencia de dos voltajes en un par de conductores, lo cual es logrado mediante la incorporación de una resistencia de aproximadamente 100 [Ω], adaptada a la impedancia característica del cableado en el terminal receptor del sistema. Esta topología electrónica permite que el receptor interprete la polaridad de este voltaje (dado que es diferencial), para determinar el nivel lógico correspondiente.

Sin embargo, los pares diferenciales no solo se pueden implementar en medios como pares trenzados de cobre, sino que además se pueden incluir en diseños de circuitos impresos (PCB[1]) mediante pistas paralelas, de largos, anchos y espesores correctamente calculados. Es el esquema que se llevará a cabo para la comunicación entre los conversores Análogo/Digital y las FPGAs, fundamentalmente dada la gran cantidad de información digital libre de ruido que se requiere adquirir. Un cálculo rápido nos permite percibir la magnitud de estos flujos:

$$16\ elementos \cdot 40\ MSPS \cdot 12\ bits = 0.89406\ [GBps]$$

[1] Printed Circuit Board.

donde MSPS corresponde a millones de muestras por segundo y 12 bits la resolución de cada muestra. Esta aproximación es suficiente como para enfrentar la situación de transmisión con la importancia que esta realmente tiene.

Adicionalmente, el ambiente donde estarán presentes estas señales, corresponde a uno lleno de ruidos pulsantes que generan incertidumbre respecto de la integridad de los datos; es por esto que la idea de usar un sistema de transmisión LVDS toma validez, ya que las interferencias eléctricas y magnéticas a las que está sujeto un conductor del par diferencial, son muy parecidas a las de su opuesto, por lo que al restar ambas señales, el ruido superpuesto en ambas es eliminado por completo.

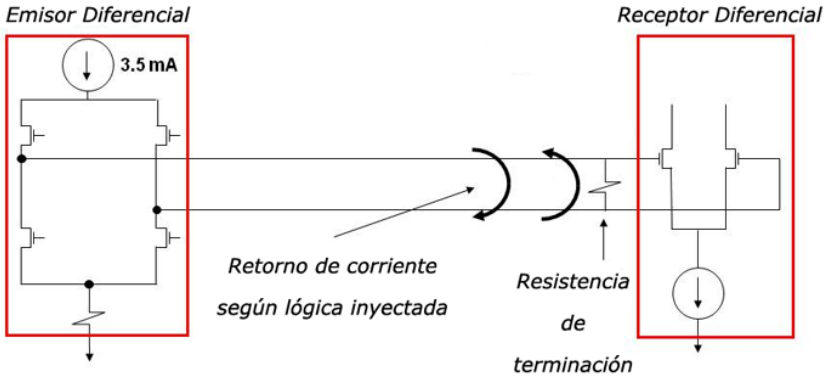

Figura 18: Esquema de funcionamiento para un sistema de transmisión LVDS.

2.7 Etapa de procesamiento de señales

La capa visible para cada usuario de un escáner de ultrasonografía, es efectivamente la etapa de visualización, la cual viene precedida por una sección importante de procesamiento, la cual se encarga de tomar un arreglo de posiciones rescatadas a través de los ecos y formar una imagen interpretable y con escalas coherentes.

La primera etapa justo después de la conformación de ondas, corresponde a filtros de atenuación de ruido e intensificación de bordes, los cuales son ampliamente útiles en este aspecto, dado que permiten una lectura más rápida y eficiente de lo que efectivamente está siendo recibido a través del transductor ultrasónico. La idea detrás de esto consiste en la mejora de las imágenes media post-procesamiento convencional de imágenes el cual por lo general esta compuesto de filtros pasa-bajos o pasa-banda, incluyendo algunos de algoritmos locales como medias móviles o convoluciones de detección de bordes.

La implementación de estos filtros es realizada por lo general en plataformas de procesamiento de señales digitales como los DSPs[1], los cuales poseen librerías y funciones pre-configuradas para el tratamiento de imágenes; en efecto, estos dispositivos están muy bien desarrollados y optimizados para este tipo de funciones, sin embargo poseen algunas consideraciones respecto de la energía, y es que si se desean realizar tareas como las antes mencionadas, las plataformas sugeridas, tienen velocidades de núcleo cercanos a los 1.2 [GHz], lo que implica de forma directa grandes consumos energéticos en pérdidas por conmutación, llevando la electrónica a temperaturas cercanas a los 60ºC.

[1] Digital Signal Processor.

En el contexto del proyecto, particularmente para el caso de la implementación de un equipo ultra-portátil, esto no se ajusta a los requerimientos básicos, los cuales consideran sistemas de bajo consumo y los más eficiente posible. Dado esto se debe optar por la implementación de estos algoritmos basados en FPGAs, las cuales pueden realizar procesamiento paralelos a algunos cientos de Mega Hertz, en vez de procesamiento secuencial a Giga Hertz; adicionalmente es posible controlar el encendido y apagado de relojes de alta velocidad temporalmente, según se requiera para optimizar aún más los consumos asociados.

2.8 Dominio y aporte del trabajo

Se espera como aporte diferenciador para este trabajo, la generación de una plataforma de pulsación ultrasónica de dimensiones muy reducidas y de bajos consumos, para su implementación en la construcción de un ecógrafo ultra-portátil. En el aspecto de innovación tecnológica a nivel nacional y latinoamericano es un concepto novedoso con un mercado atractivo, y con desarrollo muy escaso. En particular se busca la generación de un sistema de pulsación clasificable como estado-del-arte con la inclusión de tecnologías como el *clamping* activo y capacidades de pulsar a frecuencias de hasta unos 7.5 [MHz]. Adicionalmente y como un plus-valor importante se buscará integrar este concepto con sistemas de multiplexión activa de administración abstracta; lo que se puede considerar una etapa completa de emisión-recepción de ondas ultrasónicas análogas, las cuales irán mas tarde al mecanismo de conformación.

Respecto del dominio y los alcances, el proyecto está enmarcado en la construcción de un ecógrafo ultra-portátil para examinación rutinaria de bajo costo; se busca así la concepción de un equipo de tamaño muy reducido con resoluciones de calidad media, que permitan el pre-diagnóstico rápido de un paciente en casi cualquier lugar. Es por esto que el producto estará enfocado a médicos con conocimientos básicos de imagenología.

Dadas todas las partes y la magnitud que involucra el cálculo, diseño y construcción de un equipo médico, el trabajo se desarrolla en el contexto de un grupo multidisciplinario de ingenieros, abarcando así cada uno de los módulos que conforman el conjunto funcional.

Capítulo 3

Diseño e implementación de sistemas.

En este capítulo se describe el diseño e implementación de los sistemas de pulsación, multiplexión y topologías de recepción para la construcción de un ecógrafo ultra-portátil.

Se mostrarán las consideraciones, diseños, simulaciones e implementación de la electrónica y el software desarrollado para conseguir los resultados

3.1 Diseño e implementación de sistema de pulsación Pulser v.1.0

Para el diseño de la primera versión de la plataforma de pulsación, se utiliza directamente un esquema *push-pull* sencillo funcional. A continuación se expone la constitución de cada bloque.

3.1.1 Drivers de corriente para MOSFETs

Para una primera aproximación se propone la utilización de compuertas lógicas en paralelo, con el fin de lograr más corriente entrante en las puertas de los MOSFETs. Particularmente se utiliza un inversor séxtuple de tecnología TTL, con corrientes de salida de 50 [mA] por canal[1]; luego si se conectan dos en paralelo es posible obtener aproximadamente 100 [mA] de salida. El esquema es sencillo y se muestra en la Figura 19

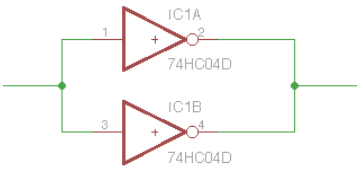

Figura 19: Configuración de salidas paralelas para obtención de mayor Fan-Out.

Se escoge este circuito como primera aproximación, dada su baja complejidad de implementación. Basta con energizar el circuito integrado, y conectar las entradas digitales para obtener los resultados de entrega de corriente deseados, adicionalmente el encapsulado incorpora 6 puertas lógicas, por lo que es posible energizar 3 compuertas de transistores con cada chip.

[1] Componente utilizado corresponde a un Hex Inverter 74HC04

3.1.2 Translación de referencias y pulsadores

Tal como se especificó en los antecedentes de esta memoria, para lograr encender correctamente un transistor MOSFET, se requiere aplicar una tensión en la compuerta suficientemente alta, pero esto es realizable únicamente si se genera un gradiente tal que la corriente fluya hacia la compuerta y no en sentido inverso; en efecto, para un transistor NMOS, la compuerta está referenciada respecto de su terminal de fuente[1], por lo que la forma de generar el campo eléctrico apropiado es imponiendo una tensión talque $V = 0$ esté en el nodo correspondiente.

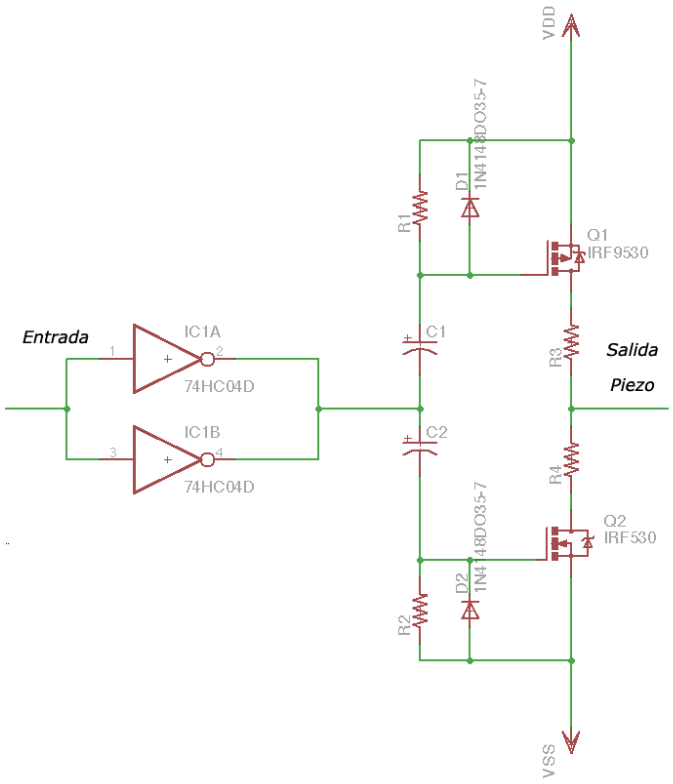

Figura 20: Esquemático del circuito para un pulsador (Versión 1.0)

En este caso la traslación de tensiones se hace a través de dos esquemas:

- Condensadores de desacoplamiento, los cuales permiten eliminar la tensión DC presente en la salida de los inversores TTL y volver móvil la señal saliente de estos.
- Conjunto resistencia-diodo: estos permiten fijar un nuevo punto de operación DC para las señales salientes de los condensadores de desacoplamiento. Electrónicamente, la resistencia lleva el punto de operación DC justo a la tensión VDD y VSS respectivamente, centrando la señal en ese punto; sin embargo el diodo en paralelo, permite generar una configuración de "restaurador DC", la cual no permite que la tensión vaya más allá de la tensión de alimentación, de esta forma la señal completa es referenciada respecto de las tensiones VDD y VSS respectivamente

[1] SOURCE Pin.

Luego, las señales presentes en las compuertas de los MOSFETs estarán referenciadas a sus puntos respectivos, permitiendo así imponer la tensión de compuerta necesaria para su encendido. Finalmente, se agregan resistencias en el nodo común de carga en configuración serial, las cuales están destinadas a la limitación de corriente en los dos sentidos que esta tomará en ese punto de conexión. Dado que la pulsación debe ser bipolar y el transductor como carga tiene una gran inercia, aparece un efecto importante de encendido doble; es decir, se enciende el MOSFET complementario antes que el anterior se haya apagado por completo, esto no es deseable pues eleva las corrientes de conmutación I_{DS} varias veces sobre la nominal de consumo. Es por esto que se deben calibrar estas dos resistencias de limitación de forma que adapten correctamente la impedancia con la carga del piezoeléctrico, y además restrinjan la corriente circulante en esa rama para así evitar la posible destrucción de los semiconductores por sobre-corriente. El diseño final de los circuitos esquemáticos para esta versión del sistema de pulsación se encuentra disponible en el Anexo 1: Diagramas de sistema de pulsación "Pulser v.1.0".

3.1.3 Confección de PCBs

Después del proceso de diseño de estos circuitos se confecciona la tarjeta electrónica con los circuitos impresos para su posterior construcción. Dentro de las consideraciones básicas, está el uso de pistas con ángulos de 45º que permiten la reducción de la impedancia de línea y el agrupamiento de los componentes de forma coherente respecto de su grado de asociación, es decir, para cada módulo de pulsación se localizan los componentes pertenecientes en la vecindad cercana, permitiendo un desarrollo más modular, compacto y robusto a las interferencias.

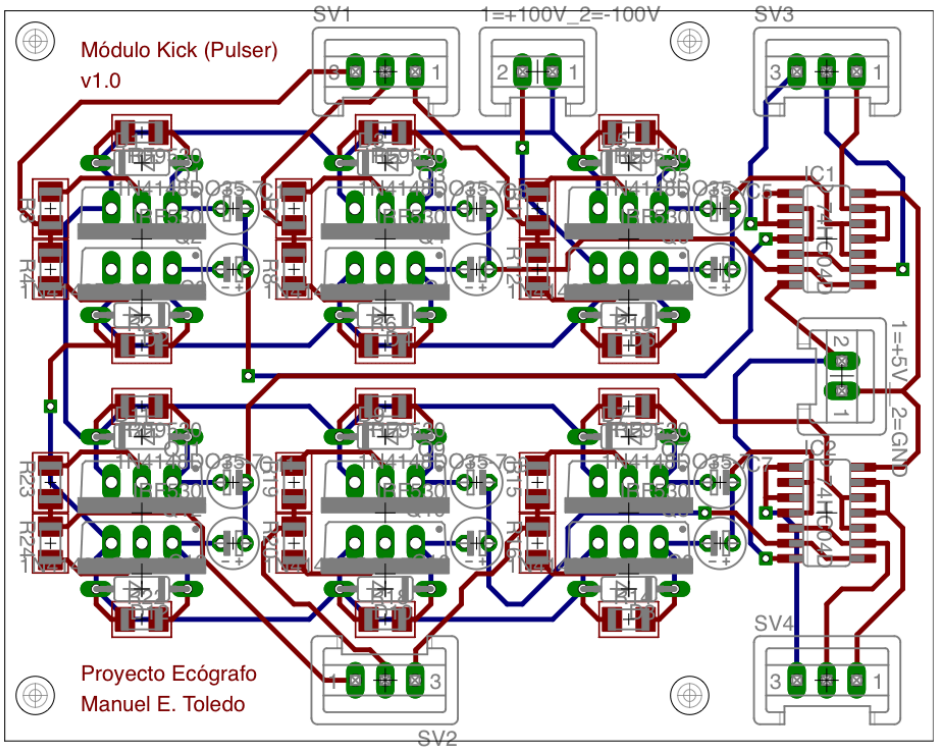

Figura 21: Modelo de la PCB para construcción del Pulser v1.0.

En la Figura 21, se puede visualizar el modelo del circuito impreso que se diseñó para la implementación de la primera versión del sistema de pulsación. Este diseño es realizado en un

software de tipo CAD[1] de la empresa CadSoft, el cual cuenta con las capacidades de generar esquemáticos coherentes a diseños de placas electrónicas, confección de librerías para componentes específicos y sistemas de verificación de coherencia y reglas de fabricación. Este software permite diseñar los esquemáticos, generando automáticamente un espacio de trabajo con los distintos encapsulados de cada componente y conexiones sugeridas que deben ser "ruteadas" según las especificaciones y consideraciones adoptadas para cada señal.

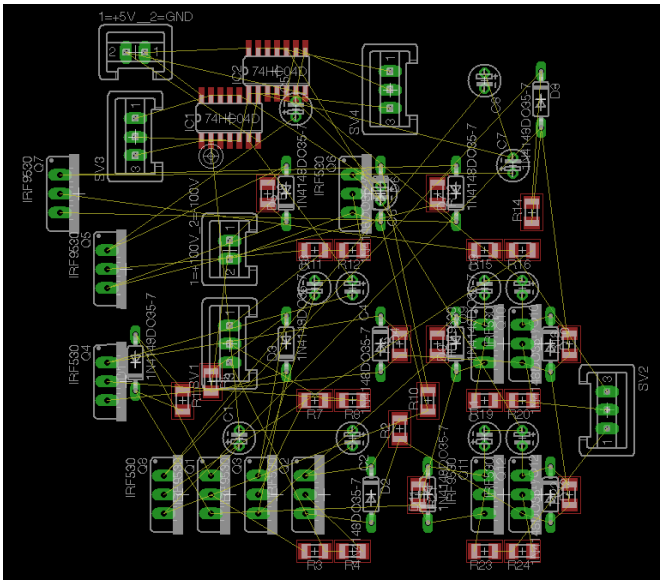

Figura 22: Espacio de trabajo inicial que entrega CadSoft Eagle para rutear las conexiones entre componentes.

Una vez terminado el diseño y la confección, se envían los archivos para su construcción en una placa de fibra con material fotosensible que permite trazar las pistas y pads dispuestos a través de un proceso fotomecánico clásico.

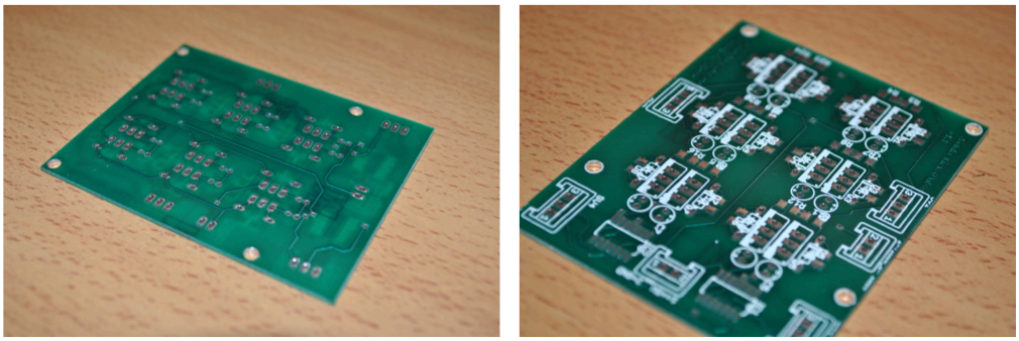

Figura 23: Fotografías de la PCB del Pulser v.1.0

A continuación, se disponen los componentes para su posterior soldado en la placa, de forma consistente a los diseños para obtener una plataforma funcional. Los resultados del proceso de ensamblado y montaje se pueden ver en la Figura 24.

[1] CAD: Computer-Aided Design: Diseño asistido por computadora.

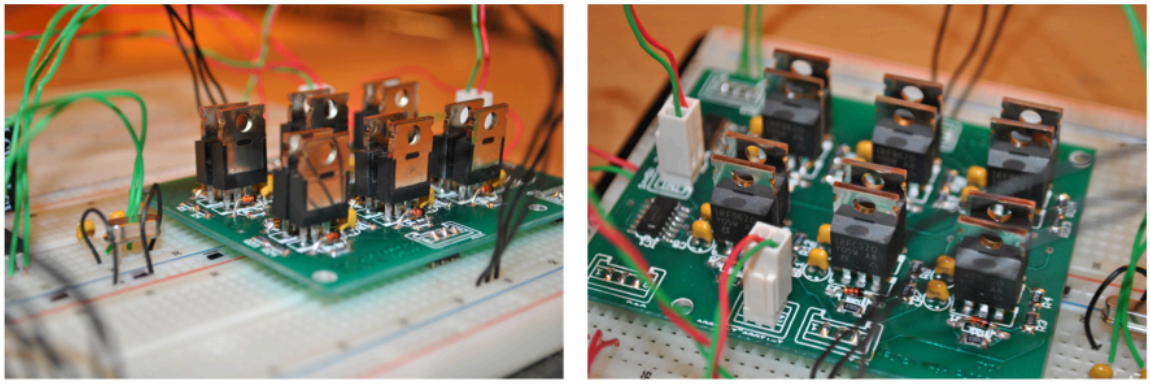

Figura 24: Imágenes del montaje de componentes del Pulser v.1.0

Con las condiciones antes expuestas se comienza con el proceso de pruebas preliminares, que permitan revisar el correcto funcionamiento de los circuitos construidos, particularmente se realizan pruebas de estabilidad y rendimiento con las siguientes consideraciones:

- Pulsos de entrada cuadrados.
- Ciclo de servicio de la señal entrante del 50%.
- Señal entrante con tensión de 5[Vpp] centrada en 2.5[V].
- Carga resistiva pura de 2 [kΩ].
- Alimentación de ± 30 [V] para la etapa de pulsación.

Los resultados arrojados son los mostrados en la Tabla 1.

Frecuencia [MHz]	Pérdidas por conmutación [%]	Corrientes de fuentes[1] [mA]
0.01	2	50
0.1	3	53
0.5	8	60
1	15	65
2	22	71
2.5	27	89
3	29	100
3.5	30	120

Tabla 1: Resultados de pruebas de estabilidad y rendimiento para el Pulser v.1.0.

Las pruebas son explícitas respecto de un sistema relativamente ineficiente a frecuencias cercanas a los 3.5 [MHz], esto fue analizado y se explica de forma directa dadas las condiciones físicas de la electrónica presente.

En primer lugar, los transistores escogidos, si bien poseen la capacidad de oscilar a estas frecuencias, sus cargas de compuertas son demasiado altas, por lo que se debe polarizar las compuertas en un tiempo mínimo para lograr conmutar a alta velocidad, esta es una consideración importante que se tuvo para el diseño de la siguiente versión; por otro lado, se concluye que la característica poco condensada del circuito completo hace que las pérdidas sean mayores por la aparición de inductancias de línea inadecuadas, además de la utilización de transistores de encapsulados muy grandes.

[1] Corriente total sumada de cada una de las fuentes DC, que componen el conjunto bi-polar.

3.2 Análisis y puesta en marcha del sistema de multiplexión

Tal como se especificó en la contextualización, la multiplexión es una etapa que es tomada con otra concepción, pues los transductores ultrasónicos tradicionales son altamente costosos y además muy complejos en su conector, lo que genera una apuesta interesante respecto a la utilización de transductores diseñados para ecógrafos portátiles provenientes de China, los cuales son varias veces más económicos y con arquitecturas físicas mucho más pequeñas.

3.2.1 Situación inicial del sistema de multiplexión

La primera etapa a realizar para poder utilizar los transductores de ultrasonido que incorporan equipos existentes, es desarmarlos para revisar su arquitectura y filosofía de operación. En este caso, la arquitectura encontrada está basada en switches análogos marca HITACHI® los cuales tienen las siguientes características:

- Tecnología MOSFET de switching
- Resistencia de encendido de 22 [Ω].
- Tensiones de alimentación máximas de ± 110 [V].
- Sistema de registro serializado integrado.
- Tensión de alimentación y señales de control TTL.

Con un total de 80 elementos y 16 canales de salida es natural pensar que se generarán 5 grupos de señales multiplexadas; además se tiene la siguiente topología de cada circuito integrado.

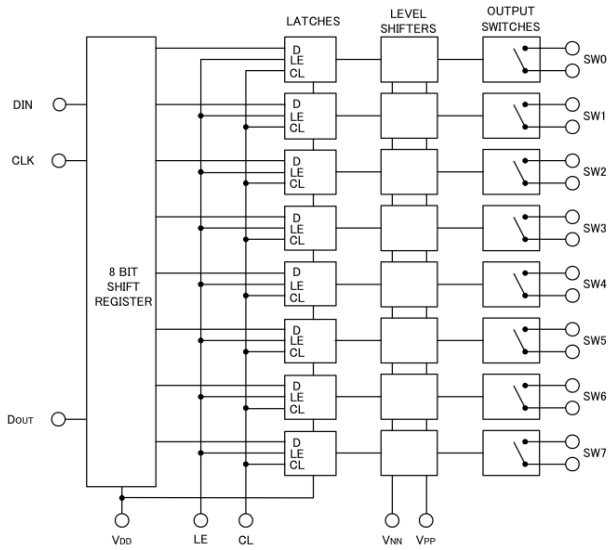

Figura 25: Topología del Analog Switch ECN3290TF de HITACHI®.

Al observar las tarjetas electrónicas en el interior de la empuñadura del transductor, se pueden visualizar 10 chips ECN3290TF, lo cual tiene bastante sentido, en conocimiento de que cada circuito integrado posee 8 switches controlados por una entrada de registro serializado, los cuales deben generar el enrutamiento de 80 elementos piezoeléctricos.

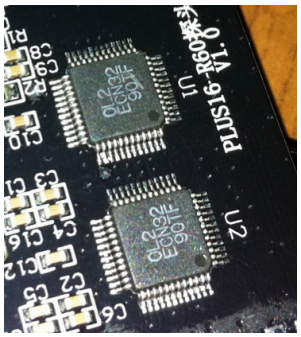

Figura 26: Circuitos integrados de multiplexión ECN3290TF.

El problema a resolver entonces, corresponde a la investigación de la codificación que este sistema utiliza para lograr multiplexar de forma coherente cada posición geométrica en grupos de 16 elementos. Se procede en primera instancia a la identificación del PINOUT[1], la cual arroja el resultado mostrado en la Tabla 2.

Figura 27: Imagen que representa la identificación manual de pines del transductor.

Código Pin	Descripción
1 → 16	Conexión eléctrica para el elemento N del grupo multiplexado
D0	Conexión de datos para los multiplexores 1 y 2
D1	Conexión de datos para los multiplexores 3 y 4
D2	Conexión de datos para los multiplexores 5 y 6
D3	Conexión de datos para los multiplexores 7 y 8
D4	Conexión de datos para los multiplexores 9 y 10
CLK	Conexión del reloj de sincronización para el ingreso de bits al registro serializado.
LE	Conexión del indicador de: dato establecido para ser multiplexado.
-80	Conexión de +80[V] para la polarización de los switches análogos.
+80	Conexión de -80[V] para la polarización de los switches análogos.
+5	Conexión de +5[V] para la alimentación de la lógica TTL interna de serialización.

Tabla 2: Decodificación del conector del transductor.

La siguiente etapa, corresponde a la identificación de los pines de salida del cabezal transductor y su respectiva posición geométrica. Este procedimiento, al igual que el anterior resultó no ser trivial, ya que como se mencionó con anterioridad no hay disponibilidad de ningún tipo de documentación dada la condición de producto final de este dispositivo. Se implementa entonces un

[1] Identificación de funcionalidades de pines de conexión.

montaje que permita determinar la separación geométrica de 80 elementos en una excursión de aproximadamente 5.5 [cm].

Figura 28: Montaje para detección e identificación de posición geométrica de elementos.

Como se puede ver en la Figura 28 se generó un canal unilineal en el centro del transductor (que corresponde a la posición donde están localizados los elementos), además se adapta con gel ecográfico una aguja, que permita la diferenciación entre cada elemento, los cuales tienen una separación muy pequeña.

$$\frac{5.5\ [cm]}{80\ elementos} = 0.68\ \left[{}^{mm}/_{elemento}\right]$$

Los resultados de este procedimiento son los que se muestran en la Figura 29.

GND	GND	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	GND	GND
GND	GND	2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	GND	GND
GND	GND	3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	GND	GND
GND	GND	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	GND	GND

Figura 29: Esquema de distribución de pines para el cabezal transductor.

Los números corresponden a la posición absoluta geométrica de cada elemento respecto del origen de éste; los pines marcados con GND son una masa común que conecta todas las tierras de cada elemento piezoeléctrico.

El siguiente proceso consiste en el desarrollo de la ingeniería inversa de la codificación y topología de conexiones de los switches análogos, implementando un mecanismo que permita manejarlos, tal como lo hace en el producto original.

3.2.2 Diseño e implementación del sistema de control para la multiplexión

Para la decodificación de la estructura de funcionamiento del sistema de multiplexión, se requiere de una plataforma digital controlable, que permita la realización de pruebas con el fin de obtener conclusiones respecto de los modelos de codificación que se propongan.

Dadas las condiciones de lógica TTL y la libertad de poder multiplexar a velocidades bajas, se opta en primera instancia por un microcontrolador Microchip® modelo 16F877A, el cual cuenta con

la opción de oscilar a 20 [MHz] a través de un cristal externo. Se diseña una implementación sencilla de pruebas como la mostrada en la Figura 30, la cual consta de un bloque de gestión de energía el cual entrega las tensiones VPP y VNN necesarias para la polarización de los multiplexores, y tensión de +5[V] necesaria para la alimentación de la electrónica de control y la lógica de la multiplexión; un microcontrolador 16F877A con su respectiva configuración periférica y un conector hembra que permita la interconexión de este diseño con el conjunto: conector, cable, multiplexor y transductor. La programación del microcontrolador es realizada en lenguaje C y traspasada al dispositivo a través de una plataforma de programación y debug USB de Microchip® llamada PicKit3™.

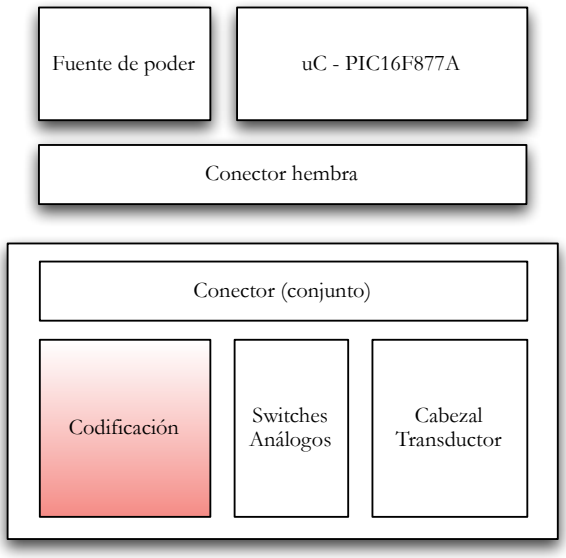

Figura 30: Esquema de implementación para pruebas de multiplexión.

El proceso de decodificación se realiza de forma metódica siguiendo los pasos que se muestran a continuación:

- Análisis de la arquitectura.
- Estimación del modo de funcionamiento.
- Inyección de datos a la lógica del multiplexor en conjunto.
- Análisis espacial de los pulsos obtenidos a través de su visualización en otro ecógrafo.
- Corrección de la arquitectura planteada.
- Implementación de nueva codificación en lenguaje C.

Luego de varias iteraciones sucesivas se logra llegar a los resultados deseados, que corresponden al completo conocimiento de la codificación con la cual el fabricante diseñó su arquitectura de multiplexión. Esto, tal como se explicó con anterioridad, tiene ventajas comparativas respecto de la reducción de costos que el equipo final tendrá y el aumento de la factibilidad de lograr repetitividad de construcción.

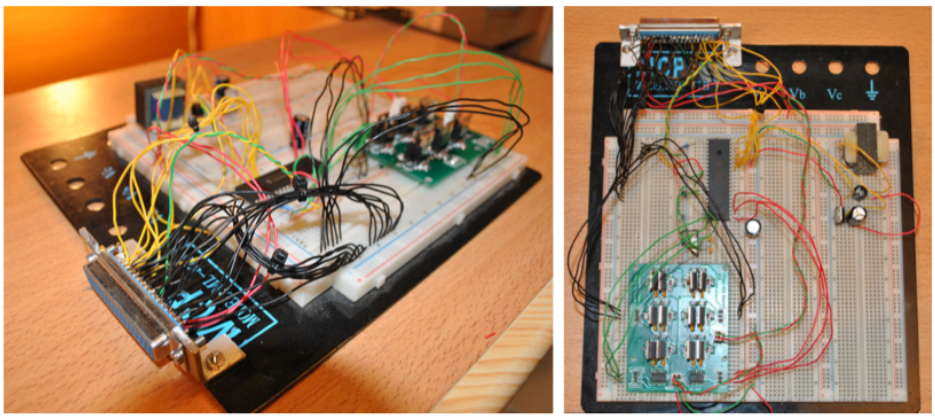

Figura 31: Fotografía del montaje de pruebas para decodificación de la multiplexión.

El código implementado en el microcontrolador se encuentra disponible en el "Anexo 2: Código C para 16F877A decodificador de multiplexión", por otro lado los resultados obtenidos de todo el proceso de ingeniería inversa y decodificación son mostrados en la Tabla 3. La forma de interpretar estos resultados es la siguiente:

- El campo posición muestra la posición absoluta de cada elemento del transductor partiendo desde el origen geométrico de éste.
- Cada elemento tiene una combinación única con un canal de salida hacia el sistema de pulsación/recepción.
- B1 → B16 corresponde a la configuración de bits que debe introducirse de forma síncrona a la lógica del multiplexor. Esto es realizado a través de un "shift register" que se mueve un registro en cada flanco de subida de la señal de reloj de sincronización CLK.
- Los datos B1 → B16 deben ser ingresados en el PIN de conjunto correspondiente, por ejemplo, para el posicionamiento de los primeros 16 elementos solo deben ingresarse señales al PIN D4.
- Una vez que se encuentra correctamente establecido el dato, se genera un flanco de bajada en la señal LE de latch, la cual permite traspasar el dato volátil ingresado, a la lógica de switching de los multiplexores respectivamente.
- Este cambio de direccionamiento se puede realizar como máximo cada 20 [μS].

Dato	Canal	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	Posición	Grupo
D4	16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Z4
D4	15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	Z4
D4	14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	Z4
D4	13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	Z4
D4	12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5	Z4
D4	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	6	Z4
D4	10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	7	Z4
D4	9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	8	Z4
D4	8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	9	Z4
D4	7	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10	Z4
D4	6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	Z4
D4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	12	Z4
D4	4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	13	Z4
D4	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	14	Z4
D4	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	15	Z4

D4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	16	Z4
D3	16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	Z3
D3	15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	18	Z3
D3	14	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	19	Z3
D3	13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	20	Z3
D2	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	Z3
D2	11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	22	Z3
D2	10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	23	Z3
D2	9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	24	Z3
D1	8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	Z3
D1	7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	26	Z3
D1	6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	27	Z3
D1	5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	28	Z3
D0	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	Z3
D0	3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	30	Z3
D0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	31	Z3
D0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	32	Z3
D3	16	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	Z2
D3	15	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	34	Z2
D3	14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	35	Z2
D3	13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	36	Z2
D2	12	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	Z2
D2	11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	38	Z2
D2	10	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	39	Z2
D2	9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	40	Z2
D1	8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	41	Z2
D1	7	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	42	Z2
D1	6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	43	Z2
D1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	44	Z2
D0	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	Z2
D0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	46	Z2
D0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	47	Z2
D0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	48	Z2
D3	16	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	49	Z1
D3	15	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	50	Z1
D3	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	51	Z1
D3	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	52	Z1
D2	12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	53	Z1
D2	11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	54	Z1
D2	10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	55	Z1
D2	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	56	Z1
D1	8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	57	Z1
D1	7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	58	Z1
D1	6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	59	Z1
D1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	60	Z1
D0	4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	61	Z1
D0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	62	Z1
D0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	63	Z1
D0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	64	Z1

D3	16	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	65	Z0
D3	15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	66	Z0
D3	14	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	67	Z0
D3	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	68	Z0
D2	12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	69	Z0
D2	11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	70	Z0
D2	10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	71	Z0
D2	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	72	Z0
D1	8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	73	Z0
D1	7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	74	Z0
D1	6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	75	Z0
D1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	76	Z0
D0	4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	77	Z0
D0	3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	78	Z0
D0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	79	Z0
D0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	80	Z0

Tabla 3: Resultados de la decodificación de multiplexión.

3.3 Diseño e implementación del sistema de pulsación Pulser v.2.0

Dado el bajo rendimiento presentado en la implementación del Pulser v.1.0 y la necesidad imperiosa de reducir los tamaños drásticamente, se genera casi en forma paralela el Pulser v.2.0, el cual cuenta con únicamente electrónica de montaje superficial. En este caso la topología se mantuvo muy similar cambiando únicamente los transistores a utilizar.

3.3.1 Reemplazo de transistores MOSFET

El cambio más importante en esta versión del pulsador corresponde al reemplazo de los transistores de pulsación, los cuales deben ser más pequeños, con cargas de compuerta mucho menores y características mejor ajustadas al desarrollo en cuestión.

Característica	**IRF530**	**ZXMC10A816N8**
Encapsulado	Simple TO-220	Complementario N+P – SOIC8
Corriente I_D	9.2 [A]	2.1 [A]
Resistencia de encendido	0.27 [Ω]	0.23 [Ω]
Carga de compuerta	30 [nC]	9.2 [nC]
Retardo de encendido	13 [nS]	2.9 [nS]
Tiempo de encendido	30 [nS]	2.1 [nS]

Tabla 4: Comparativa de especificaciones para los transistores del Pulser v.1.0 y Pulser v.2.0

En la Tabla 4, se puede visualizar directamente la condición comparativa entre los transistores, donde el ZXMC10A816N8 se ajusta mucho mejor a los requerimientos de tiempo y de complementariedad, ya que el encapsulado no incluye un solo transistor, sino que a favor del diseño posee canal N y el canal P; adicionalmente es bastante pequeño ya que posee un encapsulado SOIC8 de montaje superficial. Las cargas de compuerta son también considerablemente inferiores respecto de las del transistor IRF530, lo que implicará polarizar la compuerta con mayor facilidad y en menos tiempo, logrando un encendido de alta velocidad; respecto de las corriente de drenaje, es posible ver que para el nuevo transistor es de 2.1 [A], lo cual según los cálculos es suficiente para lo operación en

régimen permanente, particularmente dado el bajo ciclo de trabajo que tienen las ráfagas de pulsación en el proceso completo.

3.3.2 Confección de PCB

Dada la similitud del diseño no se mostraran los esquemáticos pues, son equivalentes a los anteriores solo que con encapsulados distintos, de esta forma lo que se expondrá será el diseño y ruteo de la placa electrónica para esta versión.

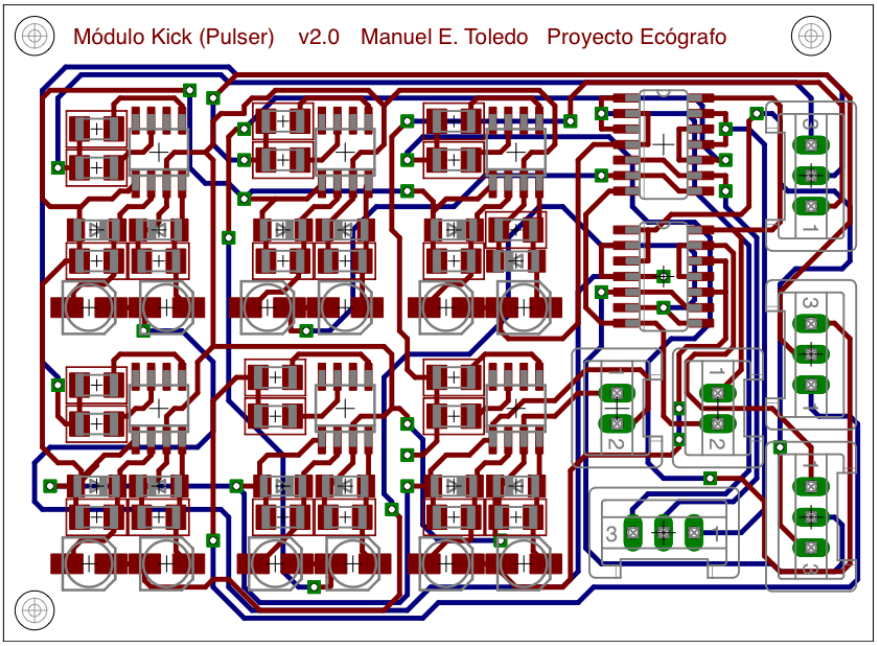

Figura 32: Dibujo CAD de la placa para el sistema de pulsación Pulser v.2.0.

En la Figura 32, es posible ver el diseño computarizado de la placa del sistema de pulsación Pulser v.2.0, el cual posee varias ventajas comparativas con su versión anterior, especialmente por la reducción en tamaño, modularización y mejora en eficiencia que debiese presentar. Se redujo también el tamaño de algunos condensadores de estabilización, permitiendo implementar encapsulados superficiales para estos también, en efecto, es posible ver que los únicos componentes *thru-hole* presentes son los conectores de energía, entradas y salidas de señales, en el lado derecho de la PCB. Siguiendo el mismo procedimiento, se generan los archivos para la construcción de esta PCB, lo que permitirá realizar pruebas incluyendo multiplexión.

Luego de construida, se procede al ensamblado de lo componentes con una estación de temperatura controlada, logrando así mantener los requerimientos para un correcto funcionamiento de todos los circuitos integrados, en particular los transistores MOSFET, los cuales no solo son electrostáticamente sensibles, sino que además son sensibles a la temperatura a la que son sometidos.

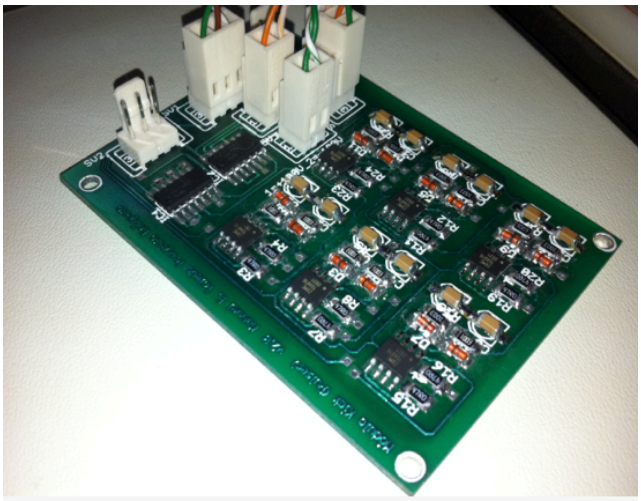

Figura 33: Imagen de la construcción y armado de la plataforma de pulsación Pulser v.2.0.

En la Figura 33, se puede visualizar la placa del Pulser v.2.0 terminada y lista para ser puesta en operación, se debe notar la importante disminución del tamaño de los componentes y de la placa electrónica también, pues en este caso se hizo uso solamente de dispositivos con encapsulado de montaje superficial. La respuesta de operación para esta plataforma se expone en la Tabla 5:

Frecuencia [MHz]	Pérdidas por conmutación [%]	Corrientes de fuentes[1] [mA]
0.01	2	50
0.1	3	53
0.5	8	58
1	10	62
2	12	67
2.5	15	71
3	17	76
3.5	19	80

Tabla 5: Resultados de eficiencia y estabilidad para plataforma Pulser v.2.0.

En este caso los resultados son muy superiores y aceptables, incluso en términos de forma de onda generada como puede ser visto en la Figura 34. Las pérdidas por conmutación son menores y además el sistema es mucho más estable térmicamente; a pesar de esto se concluye que el conjunto aún no es suficientemente operacional, pues se debe obtener la máxima eficiencia posible con el fin de cumplir los requisitos de portabilidad del equipo, y además conseguir bajas disipaciones térmicas, ya que en el interior de los encapsulados plásticos la gradiente térmica y la ventilación es mucho menor.

El Pulser v.2.0 como plataforma se encuentra en condiciones de ser operada, pero requiere ajustes; particularmente durante el estudio de los fenómenos, se encontró una situación relacionada con la etapa de driving de los transistores, donde se vuelve insuficiente la salida de los inversores TTL para suplir la corriente necesaria y así lograr el correcto encendido de éstos en tiempos adecuados. Dada esta condición se propone realizar cambios directamente sobre la tarjeta ya construida, ganando los tiempos que toma realizar una nueva construcción.

[1] Corriente total sumada de cada una de las fuentes DC, que componen el conjunto bi-polar.

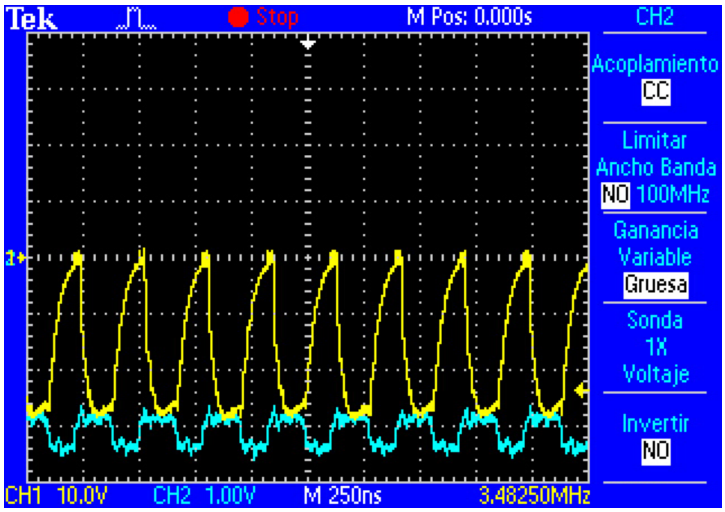

Figura 34: Forma de onda para plataforma Pulser v.2.0.

3.4 Migración de sistemas de control a FPGA

En este contexto también se pretende migrar y compactar la generación de señales de control a una única plataforma basada en una FPGA; esto, dado que el microcontrolador es insuficiente para lograr velocidades de pulsación de 3.5[MHz] y además multiplexar a esas velocidades. La FPGA consiste en un paradigma completamente distinto donde la programación es mediante lenguaje de descripción de hardware y los procesos se pueden paralelizar casi por completo.

Se usa para estos fines una plataforma funcional para prototipaje, la cual posee una FPGA de la serie Cyclone II® de gama baja para realizar pruebas y generación de algoritmos sencillos.

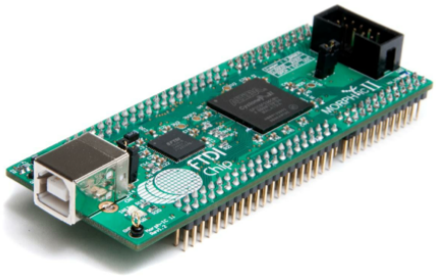

Figura 35: Plataforma MORPH IC de FTDI® para desarrollos en FPGA.

De esta forma se procede al desarrollo de software en lenguaje Verilog HDL, para la implementación de las señales de configuración de multiplexión y pulsación.

El esquema que se implementó tiene una complejidad considerable la cual será mostrada y expuesta en diagramas de bloque; adicionalmente es posible encontrar los códigos y diagramas funcionales asociados en el "Anexo 3: Esquema funcional de plataforma de control para multiplexión y pulsación."

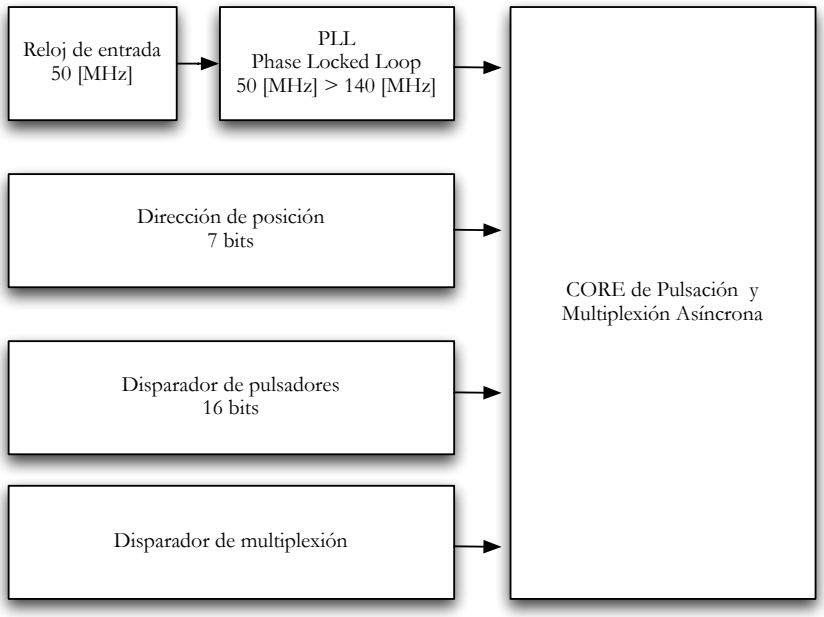

Figura 36: Esquema básico del funcionamiento de la plataforma digital de control.

Se programa entonces un sistema integral que permita el manejo operacional de todo el sistema de multiplexión y pulsación, a un alto nivel de abstracción. En este caso las entradas son datos en paralelo, fácilmente interpretables donde se especifica la posición inicial del arreglo de 16 elementos que se desea pulsar y recibir; así el sistema generado, automáticamente interpreta el direccionamiento ingresado y configura el código de multiplexión para dejar listo el sistema esperando la señal de disparo de multiplexión. El sistema implementado, lleva a cabo la configuración de multiplexión a bajo nivel, enviando las señales necesarias para habilitar los canales con los elementos piezoeléctricos especificados y devuelve un flanco que indica la realización exitosa de la operación. Se construye también un bloque de seguridad que bloquee las señales de pulsación, mientras el sistema esté multiplexando, así una vez que la señal de operación exitosa está en estado activo, el módulo permite el ingreso de señales de disparo, las cuales son ejecutadas de forma quasi-asíncrona.

3.4.1 Operación quasi-asíncrona del sistema de pulsación.

Dadas las constantes de tiempo involucradas en el proceso de obtención de una imagen de ultrasonido, se requiere controlar el instante de pulsación de la forma más precisa posible, en particular si eventualmente se desea conformar las ondas con enfoque basado en la emisión, ya que en este procedimiento la resolución de la imagen está relacionada directamente con la precisión que se tenga en los diferenciales de tiempo de la ráfaga de pulsos. Es por esto que es deseable poder pulsar un elemento transductor en un momento deseado, cualquiera que este sea.

Sin embargo, el proceso de pulsación como tal es síncrono, pues una vez que se envía la instrucción de pulsar, este debe realizarlo con tiempos muy bien definidos, los cuales determinan la potencia que disipará sobre los tejidos el transductor[1], debido a la respuesta en frecuencia de los cristales; esta ráfaga de pulsos generalmente esta compuesta por un pulso positivo seguido de un

[1] En este caso, estos tiempos corresponden a 142.85 [nS], que corresponde al tiempo que toma cada semi-ciclo de una ráfaga de pulsación a 3.5 [MHz]

pulso negativo, repitiendo esta secuencia dos veces y finalmente una etapa de *clamping* para atenuar la fuerte inercia capacitiva de la carga.

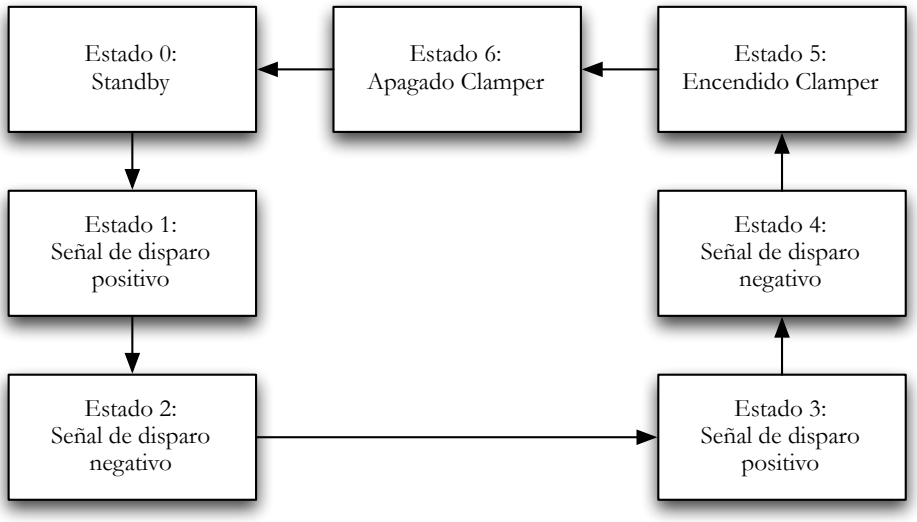

Figura 37: Máquina de estados para el proceso de pulsación.

Como la forma de onda de pulsación debe tener una frecuencia de 3.5[MHz], por cada ciclo de pulsado, entonces se deduce directamente que entre el estado 1 y el estado 3 deben transcurrir 285,71 [nS]. Además deben haber 3 eventos entre ciclo y ciclo, los cuales deben ser activados mediante los flancos positivos de cada reloj, así mediante un cálculo sencillo se concluye que la frecuencia de oscilación para el reloj que sincroniza la máquina de estados mostrada en la Figura 37 debe ser de 10.5 [MHz].

Para lograr buenas resoluciones en enfoques de emisión se requiere la posibilidad de emitir dos pulsos para dos elementos contiguos con diferencias temporales inferiores a 14 [nS], por lo tanto frecuencias de 10.5 [MHz] son insuficientes.

Se debe entonces conseguir una frecuencia superior que sea divisible por un número entero que entregue frecuencias de 3.5 [MHz], así se calcula una frecuencia óptima de 140[MHz], la cual permite 40 pasos de resolución a 3.5[MHz] y cumple requerimientos de diferencias temporales equivalentes a 7.14[nS].

El sistema entonces recibe la señal de activación y la encola para ser procesada justo en el siguiente flanco positivo del reloj de 140 [MHz], el cual es fácilmente generable mediante los PLL[1] incorporados en la FPGA Cyclone 2™ de Altera®. Todo el proceso de encolamiento, ejecución síncrona y control de señales fue correctamente implementado, llegando a un sistema funcional. El código para realizar estas tareas puede ser revisado en el "Anexo 3: Esquema funcional de plataforma de control para multiplexión y pulsación.".

[1] Phase-Locked-Loop: Circuito electrónico consistente en un oscilador de frecuencia variable y un detector de fase, el cual puede funcionar como desfasador, multiplicador y divisor de frecuencia.

3.5 Diseño e implementación del sistema de pulsación Pulser v.2.5

Con la plataforma digital de control implementada, se puede comenzar con la optimización del sistema de pulsación Pulser v.2.0. Tal como se mencionó, las mejoras sugeridas consisten en el cambio de consideraciones en la etapa de driving de los transistores MOSFET complementarios. En particular, se estableció que esta funcionalidad se había llevado a cabo mediante inversores digitales TTL en paralelo, para así generar la corriente que encendería los transistores. Luego de las pruebas respectivas se constata una disminución sustancial en las tensiones de salida del circuito integrado que cumple esta función, por lo tanto, la corriente entregada al circuito de polarización de las compuertas de los MOSFETs se ve directamente limitada.

Se propone entonces el cambio de esta etapa por una que permita la entrega de mayor corriente a las frecuencias de operación. Dada toda la experimentación realizada, se tiene conocimiento profundo de los fenómenos presentes en el sistema globalmente, por lo que se está en condiciones de realizar simulaciones computacionales con datos muy cercanos a los reales; por esto, se procede a la generación del modelo del sistema en un sistema de simulaciones electrónicas de Linear Technologies® llamado LTSPICE™.

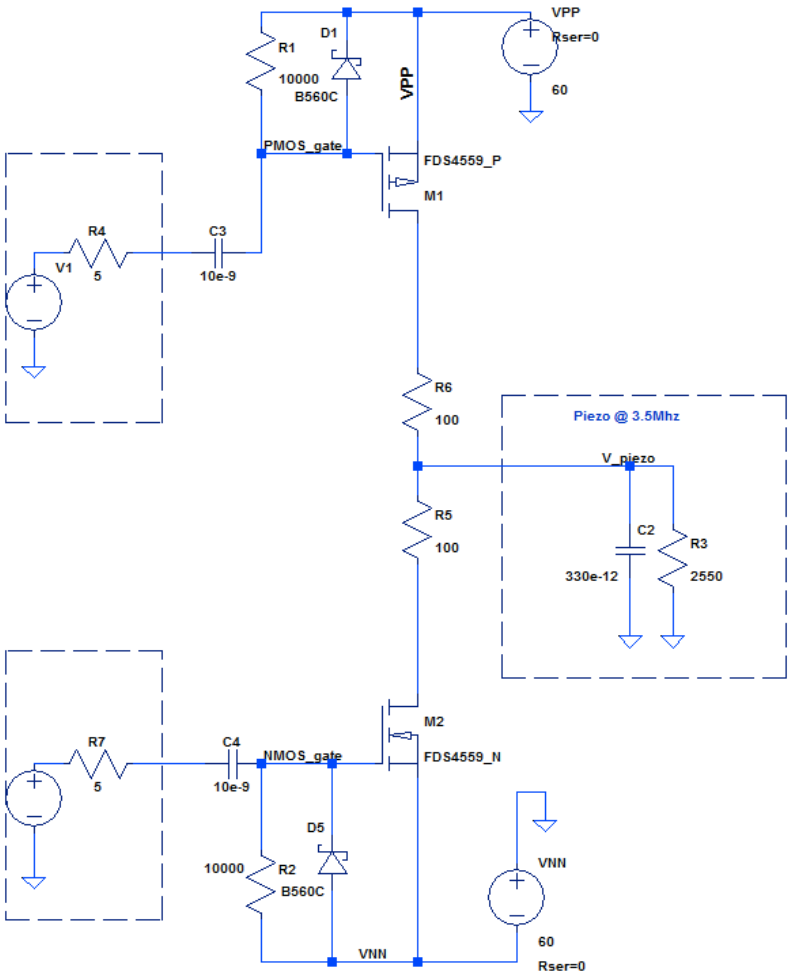

Figura 38: Esquema de topología equivalente para simulación electrónica.

En la Figura 38 se puede ver la topología para un pulsador bipolar. Todos los componentes montados en esta simulación, particularmente los transistores incorporan valores reales de uso, lo cual genera un escenario más fiel a la realidad y permitirá tener una mejor noción del comportamiento que tendrá el sistema en condiciones reales.

El bloque en la derecha es una representación del elemento piezoeléctrico con su equivalente pasivo, por otro lado al lado izquierdo se ven dos bloques simétricos que representan los drivers que se requerirá implementar para obtener los resultados deseados respecto al encendido de los transistores de pulsación.

Figura 39: Resultados de simulación inicial del Pulser v.2.5

Los resultados de la primera simulación del sistema incluyendo drivers con impedancia de salida de 5 [Ω] se muestran en la Figura 39. Se puede concluir que la inclusión de drivers con salida de mayor corriente efectivamente mejoran la forma de onda y disminuyen las pérdidas por conmutación, pero los resultados de esto siguen no siendo óptimos. Dado esto, se analiza nuevamente el circuito revisando las secciones sensibles al comportamiento observado.

Las corrientes entrantes a los circuitos de polarización de las compuertas de los transistores, esta vez son efectivamente suficientes para el encendido a las velocidades que establece la hoja de datos del fabricante, pero esto no es todo, pues si bien el modelo simplificado de los transistores MOSFET permite separar la etapa de la compuerta y la rama de carga, la física de estos no se comporta así, si estos no son sintonizados en su punto de operación.

En efecto un buen indicador de esto es la transconductancia, la cual viene dada por la siguiente ecuación:

$$g_m = \frac{\Delta I_{out}}{\Delta V_{in}}$$

Si bien este parámetro en general se utiliza para calcular el incremento de la corriente en la carga respecto del incremento en la tensión de compuerta, también es posible utilizarle en el sentido contrario para analizar su comportamiento respecto de corrientes limitadas de drenaje. En efecto la Figura 40, se muestra la condición en que si una carga drena poca corriente, entonces no habrá

opción a encender ese transistor (independiente de la cantidad de tensión que se aplique a su compuerta).

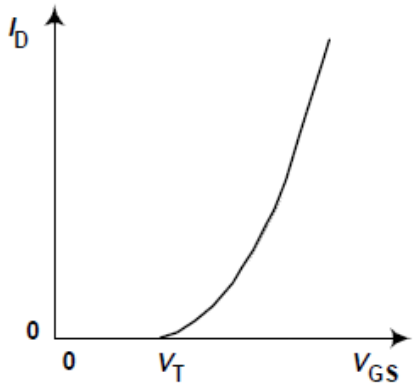

Figura 40: Corriente de drenaje vs. Tensión de compuerta para un MOSFET.

Es mediante este análisis que se puede detectar que el sistema podría requerir un redimensionamiento de las resistencias de limitación de corriente para la rama de carga. Realizando un análisis y modificando los valores para obtener un comportamiento óptimo se llega a los resultados presentes en la Figura 41.

Figura 41: Resultados para nuevas resistencias de limitación de corriente.

Las resistencias fueron modificadas a un valor de 47 [Ω], lo cual permite llevar los transistores a su punto de operación adecuado, obteniendo así formas de onda con bajísimas pérdidas de conmutación y con excursiones adecuadas para la tensión de polarización bipolar presente. En la Figura 41, la curva de mayor excursión (color verde) corresponde a la tensión en la carga; la curva inferior (color azul) corresponde a la tensión de compuerta del transistor canal N; y la curva superior (color rojo) corresponde a la tensión de compuerta del transistor canal P.

3.5.1 Simulación e inclusión del sistema de clamping

En el mismo análisis de curvas de las simulaciones, aparece el antes mencionado efecto de inercia por la condición capacitiva del transductor ultrasónico, tal como se muestra en la Figura 42.

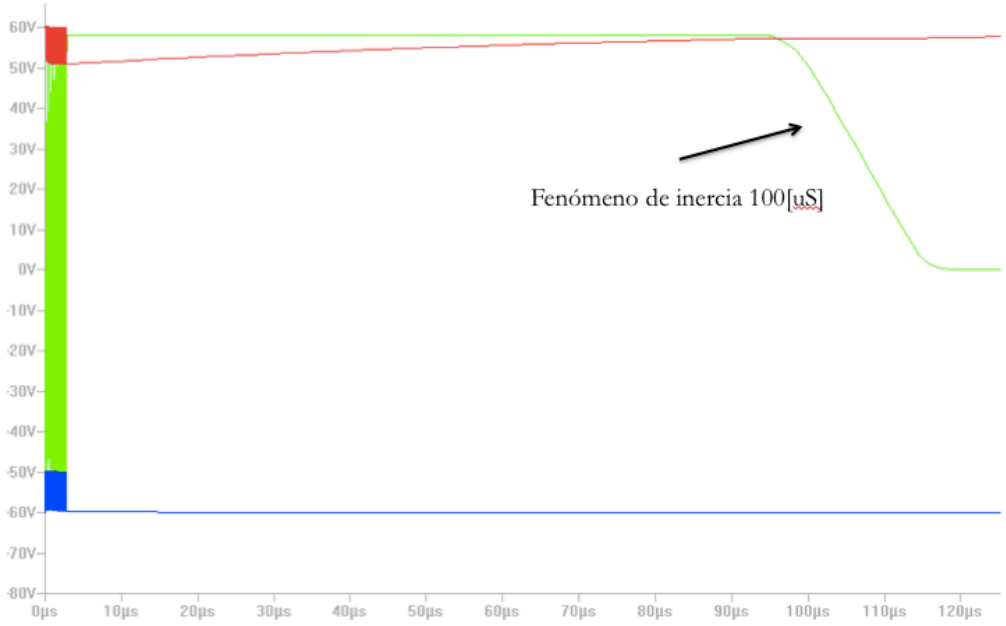

Figura 42: Efecto de inercia presente en la carga piezoeléctrica.

Dada esta condición se procede a la inclusión de un sistema de clamping que permita atenuar este efecto de forma activa y eficiente, a través de un transistor controlado, tal como se muestra en la Figura 43.

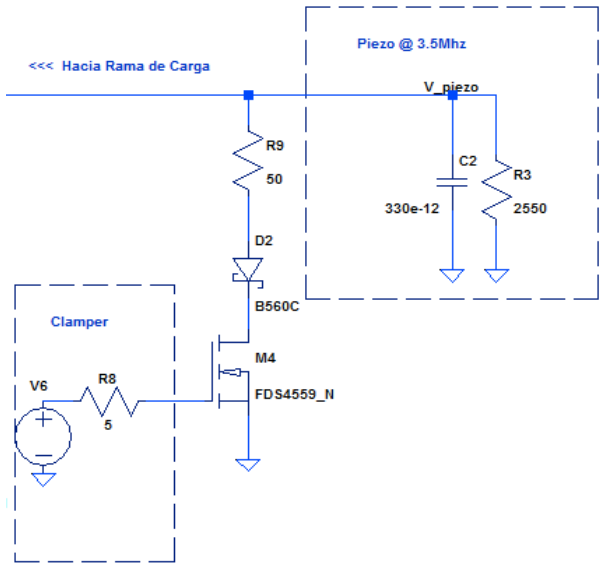

Figura 43: Inclusión de sistema de clamping a simulación.

Los resultados de la instalación de este clamper son explícitos en los gráficos de operación, tal como se puede visualizar en la Figura 44.

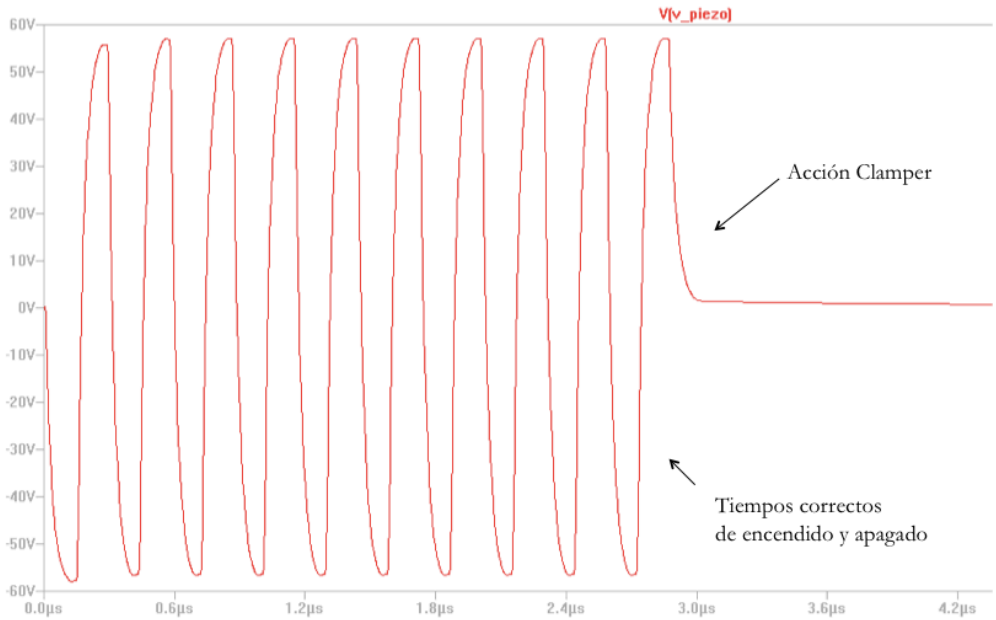

Figura 44: Resultados obtenidos con la inclusión de un sistema de clamping.

En estas condiciones ya es posible implementar los cambios realizados en simulaciones, para analizar su comportamiento bajo condiciones reales. En esta dirección, se debe buscar un circuito integrado de driving para MOSFETs que cumpla con los requisitos de corrientes y tensiones necesarias; el dispositivo escogido es el MIC4428 el que tiene condiciones muy aptas para la aplicación. Se procede entonces a la alteración de la placa para el Pulser v.2.0 y así generar la nueva versión (Figura 45).

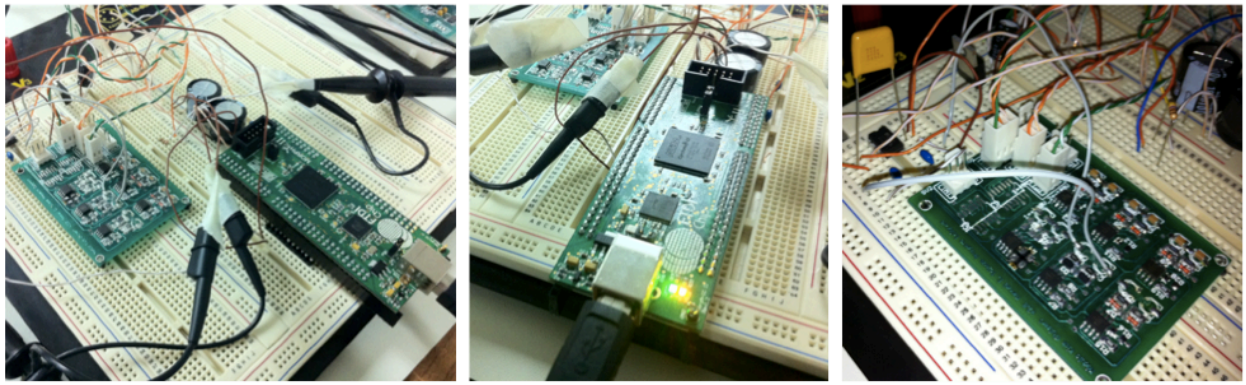

Figura 45: Fotografías de implementación de la plataforma de pulsación Pulser v.2.5.

De forma paralela se realizan iteraciones sobre el modelo del transductor ultrasónico para ajustar parámetros con mayor precisión. Este ajuste se debe realizar dada la existencia de componentes intermediando la comunicación entre el transductor y la salida del sistema de pulsación; estos son el conector, el cable y el sistema de multiplexión. El conector en particular posee un arreglo de inductancias conectadas a tierra para adaptación de impedancias, las cuales deben ser consideradas dentro del modelo de las simulaciones, para los cálculos posteriores.

El resultado de este estudio permitió calcular la inductancia equivalente del cable de comunicaciones del transductor, y el efecto que genera el arreglo de inductancias en paralelo conectadas a tierra en el conector.

Los efectos asociados a la modificación de este modelo, permitieron ajustar los parámetros del circuito de pulsación y así lograr resultados óptimos en la implementación. La evolución del modelo para el transductor ultrasónico se puede ver en la Figura 46.

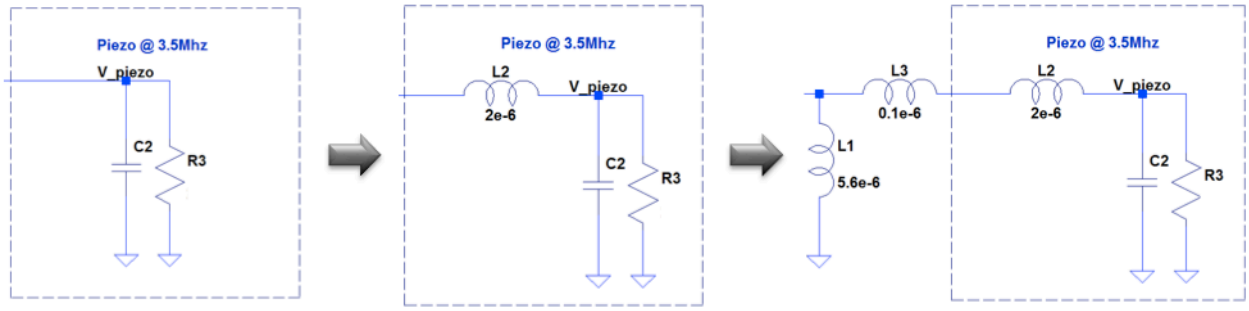

Figura 46: Evolución del modelo electrónico equivalente del transductor ultrasónico.

Capítulo 4

Resultados, fabricación y pruebas de nuevas versiones, y desarrollos en recepción

En este capítulo se exponen y describen los resultados obtenidos con la implementación de la plataforma de pulsación y multiplexión Pulser v.2.5; adicionalmente, se agrega la confección de una plataforma totalmente miniaturizada, que incorpora todas las consideraciones implicadas en la investigación y desarrollo de este trabajo de título. Los resultados de la experimentación con esta plataforma también serán expuestos para concluir el desarrollo.

Finalmente, se mostrarán los desarrollos realizados para la etapa de recepción y como estos comienzan a tomar forma, para que finalmente puedan comunicarse con el resto del prototipo enmarcado en el proyecto de la fabricación de un ecógrafo ultra-portátil.

4.1 Resultados de implementación para el Pulser v.2.5

El sistema de pulsación y multiplexión Pulser v.2.5 implementado, cumple con las especificaciones de acuerdo con los resultados de las simulaciones con el software LTSPICE™, no solo en las formas de onda sino que respecto de los consumos y las condiciones físicas que se darían al momento de energizar los circuitos electrónicos. La inclusión del sistema de *clamping* también se comportó de forma esperable, aunque fue modificado en su tiempo de acción, retrasando levemente el instante de inicio; esto fue realizado para eliminar las distorsiones generadas al intentar apagar un transistor con una carga a tierra, cuando la dinámica de este aún tiene inercia en su carga de compuerta. Estas distorsiones son fundamentalmente armónicas del circuito resonante que se forma entre las capacitancias de las compuertas de ambos transistores (el de encendido de rama y el *clamper*), por lo que se retrasó el proceso de *clamping* en medio ciclo.

La salida presente en la rama de carga, con el transductor conectado y multiplexado correctamente, se muestra en la Figura 47, donde la curva de mayor excursión (color amarillo) es la tensión en la rama de carga (tensión de pulsación); y las dos curvas centrales (colores magenta y cian), corresponden a las señales de control de pulsado, salientes de la FPGA asociada.

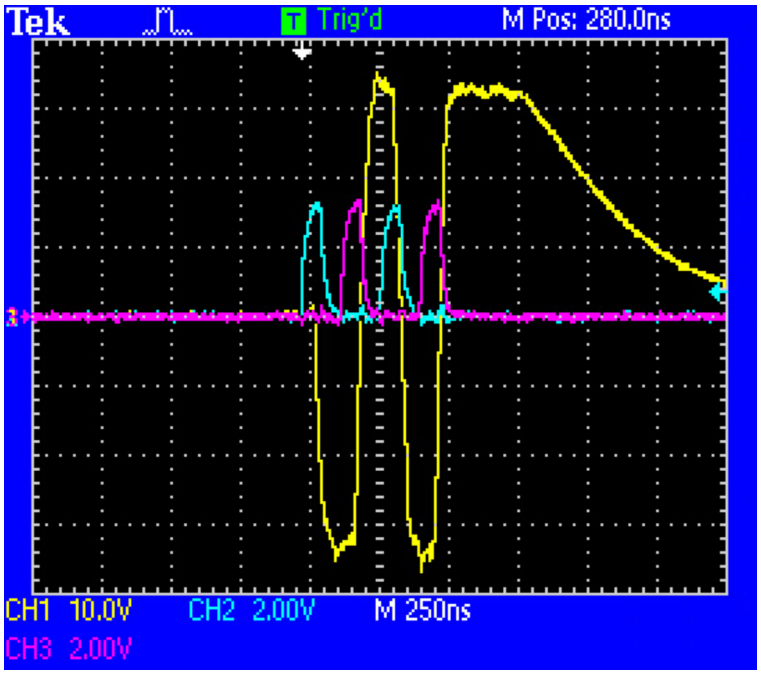

Figura 47: Resultados de pulsación y multiplexión con plataforma Pulser v.2.5.

Como se puede ver, las simulaciones fueron coherentes con la realidad observada, implicando que la generación de los modelos fue adecuada para los componentes presentes en la implementación. La modificación de los tiempos de activación del clamper mejoraron la respuesta transiente, evitando sobre-niveles y distorsiones en la señal que pueden generar pulsos ultrasónicos en frecuencias no deseadas y emisiones electromagnéticas asociadas; sin embargo, esta modificación no causa problemas ya que el sistema de recepción está concebido para comenzar a almacenar datos 50 [μS] después del último pulso de activación, por lo tanto, tiempos de atenuación del orden de 1[μS] no genera problema alguno en la recepción de los ecos que conformarán la imagen.

4.2 Diseño de sistema de pulsación Pulser v.3.0

Se estima entonces, que el sistema ya es suficientemente estable y eficiente como para escalarlo y miniaturizar, cumpliendo así con los requisitos de portabilidad del dispositivo. La generación de la última versión miniaturizada consiste fundamentalmente, en la elección de los dispositivos adecuados que contengan la tecnología necesaria integrada en un solo encapsulado. El centro de este estudio corresponde a la búsqueda de un encapsulado que colapse la mayor cantidad de funciones relacionadas con el sistema de pulsación.

El diseño e implementación orientado a reducir los tamaños físicos a la menor expresión posible, permite optimizar el espacio, integrando tecnología con los dispositivos de encapsulados más reducidos disponibles. Esta nueva plataforma, el Pulser v.3.0 corresponde al resultado de todo este trabajo de investigación y desarrollo, donde se incorporará electrónica de potencia como la mostrada y la unión con la plataforma digital de control desarrollada en una sola tarjeta electrónica condensada.

4.2.1 Circuito integrado de pulsación

El circuito óptimo disponible en el mercado para la aplicación desarrollada es el MAX4940A de la compañía MAXIM Integrated Products®, el cual tiene el diagrama funcional de la Figura 48 para un canal.

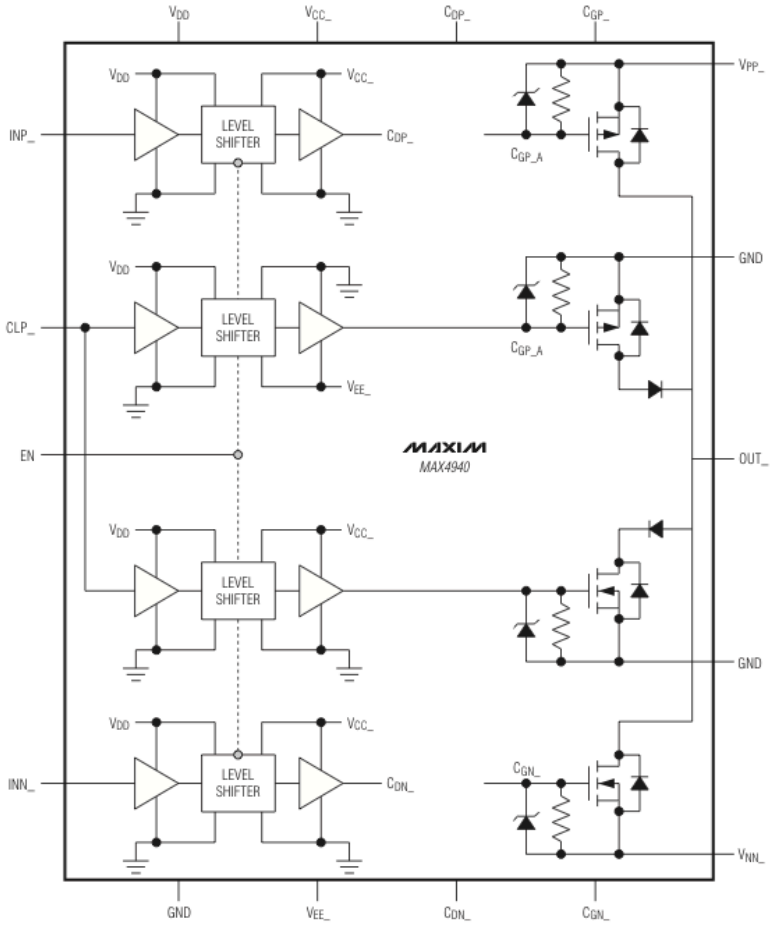

Figura 48: Diagrama funcional del IC de pulsación MAX4940A.

Como se puede apreciar en el diagrama, el circuito integrado posee la opción de operar con todas las consideraciones planteadas en este trabajo respecto del sistema de pulsación y el de *clamping*. La topología también es correspondiente con lo estudiado y lo implementado en las versiones anteriores, por lo que se visualiza un componente que efectivamente cumple con el estado-del-arte investigado, lo que permitirá eficiencia y robustez para el diseño portátil que se está abordando.

El circuito integrado adicionalmente posee 4 canales, cada uno con su sistema de *clamping* controlable a través de señales lógicas de tecnología CMOS para la FPGA Cyclone 2™ que se considera usar en esta aplicación; una fotografía del encapsulado se puede ver en la Figura 49, el cual tiene un tamaño reducido de 8 [mm] x 8 [mm].

Para producir el esquema establecido por la arquitectura de multiplexión, se deben instalar 4 de estos integrados de forma conjunta, para así poder operar 16 canales de piezoeléctricos simultáneamente multiplexados. En beneficio del tamaño del sistema final, estos chips poseen un sistema de disipación alojado en el dorso inferior del encapsulado y además poseen electrónica que

permite el control y protección contra estados prohibido y situaciones de riesgo para la electrónica de potencia.

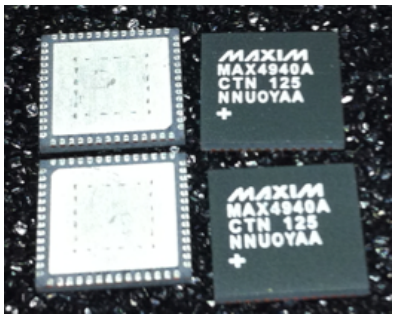

Figura 49: Fotografía del circuito integrado de pulsación MAX4940A.

Análoga a la plataforma de pulsación Pulser v.2.5, los MAX4940A permiten corrientes máximas en la rama de carga de 2[A] y tensiones máximas de operación de ±110 [V]. Adicionalmente y de forma natural, dada la existencia de sistemas de driving al interior, se debe alimentar con las tensiones de polarización y pulsado, incluyendo ±12 [V], para la energización de los drivers incorporados.

Se transforma así el MAX4940A en una solución muy adecuada para la aplicación otorgando grandes ganancias en espacio, eficiencia y escalabilidad.

4.2.2 Confección de PCB para Pulser v.3.0

Dentro de este contexto se confecciona una placa de circuitos impresos para operar este nuevo sistema, la cual dada la complejidad, debe ser de 6 capas para conseguir las condensación necesaria para su correcta operación. Este desarrollo fue realizado en Altium Designer™ de la compañía Altium Limited®, el cual al igual que Eagle™ de Cadsoft® permite el diseño CAD de tarjetas electrónicas.

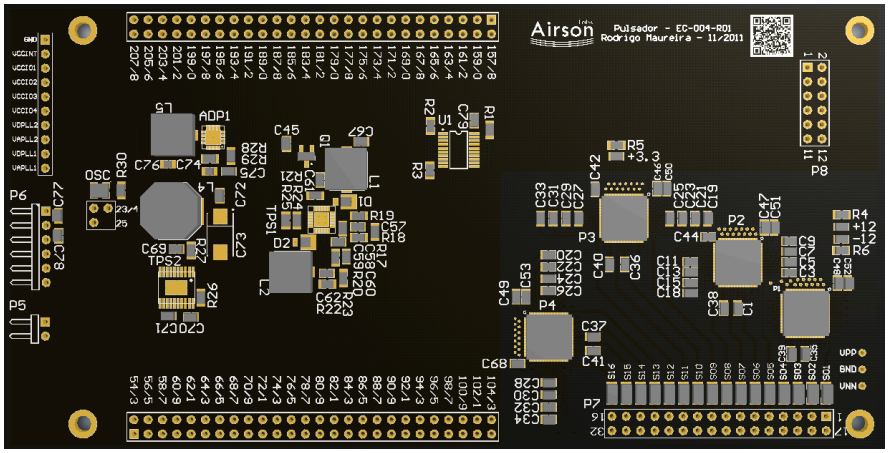

Figura 50: Vista preliminar del diseño CAD para el Pulser v.3.0.

En la Figura 50 se puede ver la versión final del Pulser v.3.0, donde la plataforma de pulsación solo corresponde a una fracción de la tarjeta electrónica visible, en este caso la parte mostrada en la Figura 51.

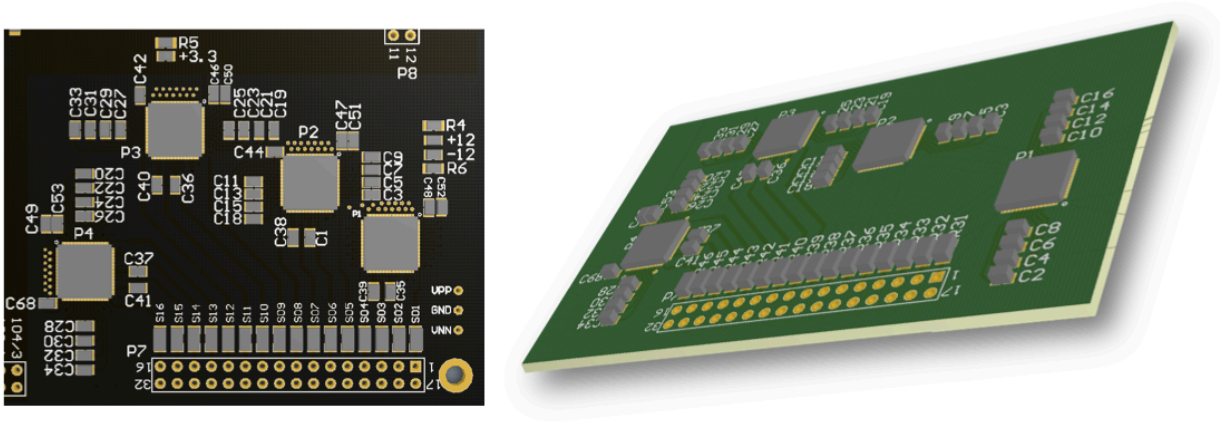

Figura 51: Fracción de la PCB correspondiente a la plataforma de la pulsación Pulser v.3.0.

El resto de la tarjeta electrónica integra la instalación de una FPGA Cyclone 2™ que controlará la pulsación y multiplexión, además y como se puede visualizar incluye las fuentes de energía de categoría switching para todas las alimentaciones necesarias en el diseño.

Tal como en el resto de las versiones, se procede a la fabricación de esta placa electrónica para su posterior montaje en los laboratorios, en particular, esto también representa un desafío y experiencia pues se debe realizar soldadura manual de un componente con un encapsulado TQFN de pads expuestos, lo cual también tiene complicaciones inherentes al tamaño de éste. Fotografías de la implementación se puede ver en la Figura 52, donde al lado izquierdo se visualiza el montaje completo de la placa de integración del Pulser v.3.0 con la plataforma digital de control basada en una FPGA Cyclone 2™; y al lado derecho una ampliación donde se detalla el montaje de los MAX4940A con los componentes periféricos necesarios para su correcta operación.

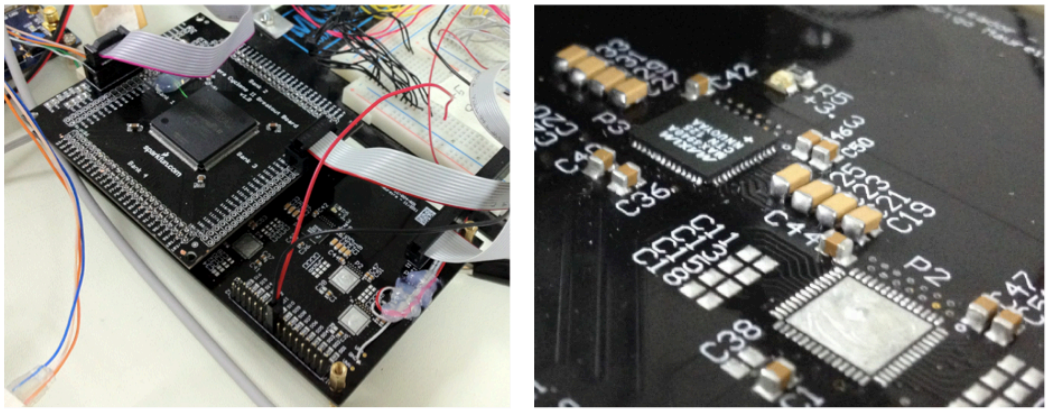

Figura 52: Fotografía de la implementación de la plataforma de pulsación Pulser v.3.0.

Los tamaños en este caso, aún no se encuentran totalmente reducidos pues aún son prototipos, la idea final es condensar todos los diseños en un espacio muy confinado, donde estará alojada la electrónica de potencia y la de control en la misma placa, la cual se comunicará con las otras etapas del equipo a través de conectores verticales que formarán un sándwich de placas operacionales sincronizadas.

4.2.3 Resultados de operación para el Pulser v.3.0

Se procede entonces a la instalación de todos los programas desarrollados para el Pulser v.2.5, en esta nueva plataforma para analizar su comportamiento, estabilidad y fenómenos asociados; donde algunas de las pruebas a realizar será la capacidad de mantener un régimen de rendimiento estable a altos ciclos de servicio y la opción de operar a un rango amplio de frecuencias, moviéndose desde los 3.5 [MHz] hasta los 7 [MHz].

Se realizan ensayos de cargas resistivas puras para probar el funcionamiento básico del sistema y poder avanzar hacia pruebas de mayor exigencia. Para esta primera etapa, se obtienen pulsos cuadrados muy bien definidos con un rizo muy pequeño, además se observa una temperatura de operación diferencialmente nula. Una captura de la forma de onda puede ser visualizada en la Figura 53

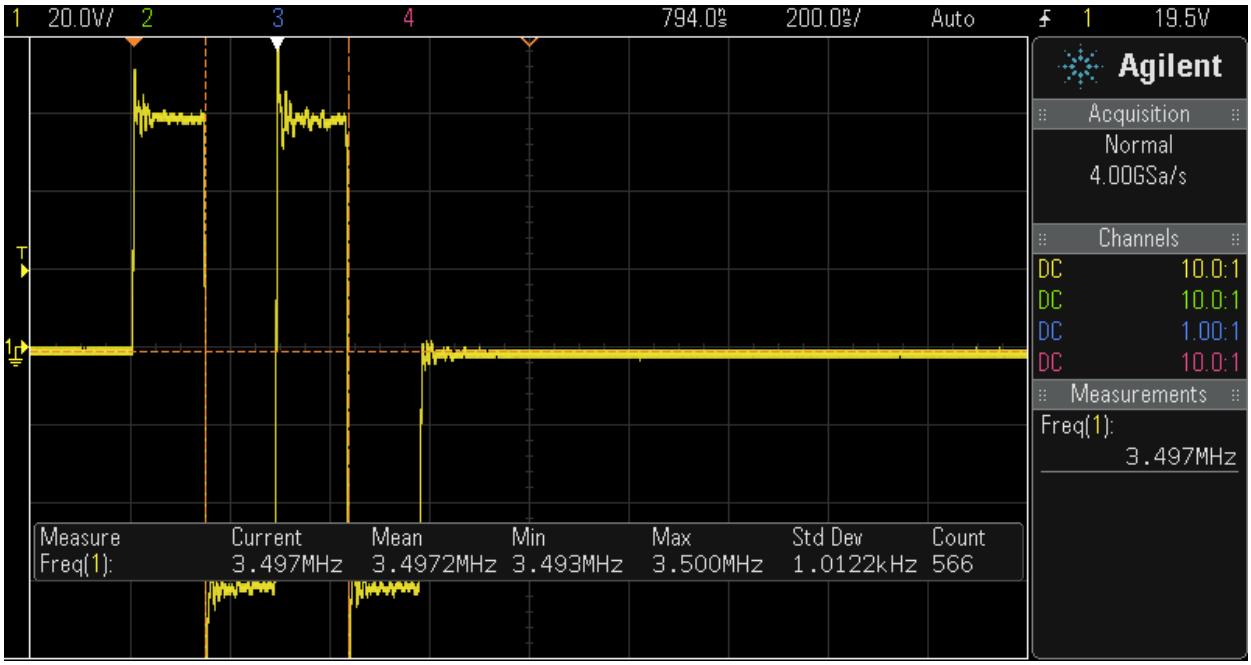

Figura 53: Prueba del sistema de pulsación Pulser v.3.0 con una carga resistiva.

La segunda prueba consiste en la inclusión del transductor piezoeléctrico y además aumentar el ciclo de servicio al triple, para así comenzar a observar la estabilidad del sistema respecto de altos desempeños y por lo tanto medir la temperatura de operación de los circuitos integrados y las variaciones de forma de onda en función del tiempo de operación y la exigencia. En este caso la prueba fue exitosa también, obteniendo formas de onda muy bien definidas y estables, además se puede observar la inercia de la carga piezoeléctrica, la cual es atenuada activamente por el sistema de *clamping* en el final de la secuencia de pulsación; adicionalmente, los pulsos pierden un poco de excursión y aumentan el efecto "*ringing*" lo cual se atribuye a las condiciones capacitivas tanto de la carga como de los mismos transistores de conmutación. Una captura de los resultados obtenidos para esta prueba se pueden ver en la Figura 54.

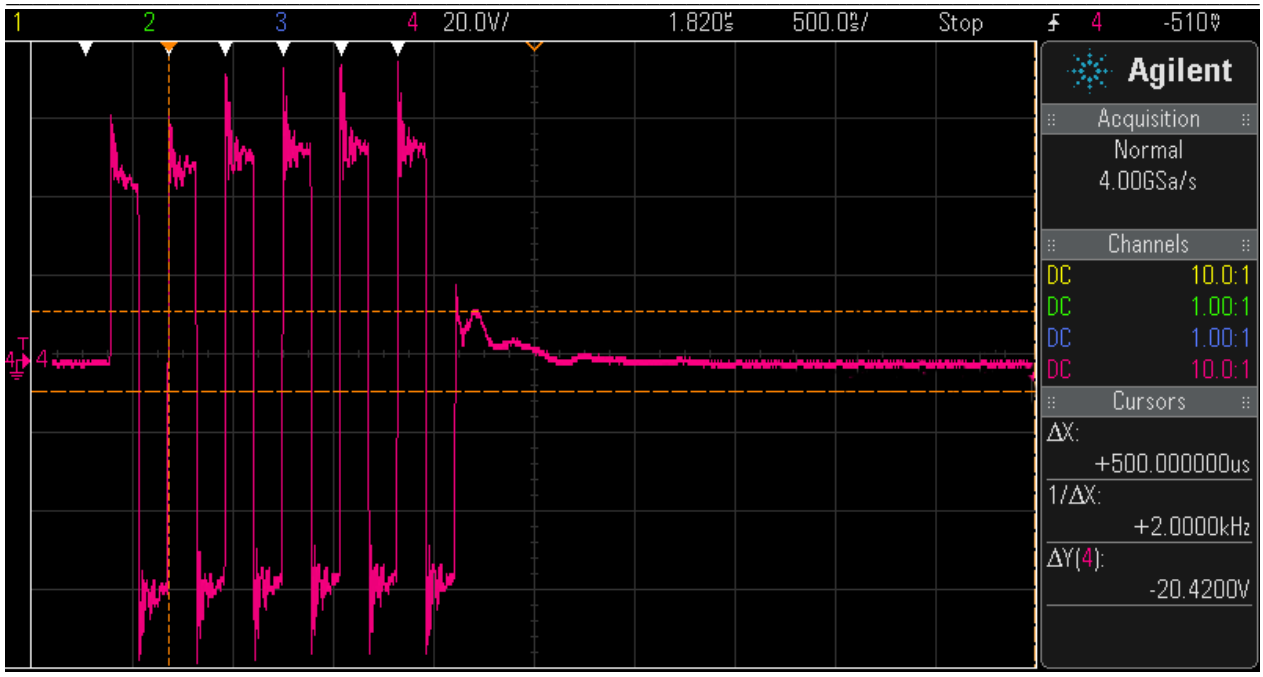

Figura 54: Prueba de pulsación para el Pulser v.3.0 con carga piezoeléctrica.

La tercera prueba a realizar consiste en el aumento de la frecuencia de pulsación a 7 [MHz], la cual es usada para transductores de pequeño tamaño, baja penetración y gran resolución en los tejidos superficiales y el campo cercano. Esta prueba es importante, ya que se busca que el equipo final de ultrasonografía sea suficientemente versátil como para conectarle un transductor lineal de examinación superficial o uno transvaginal, y que éste pueda cambiar su régimen de operación a uno de mayor frecuencia sin la necesidad de construir y adaptar otra arquitectura, sino que por el contrario, solo hacer cambios a nivel de software.

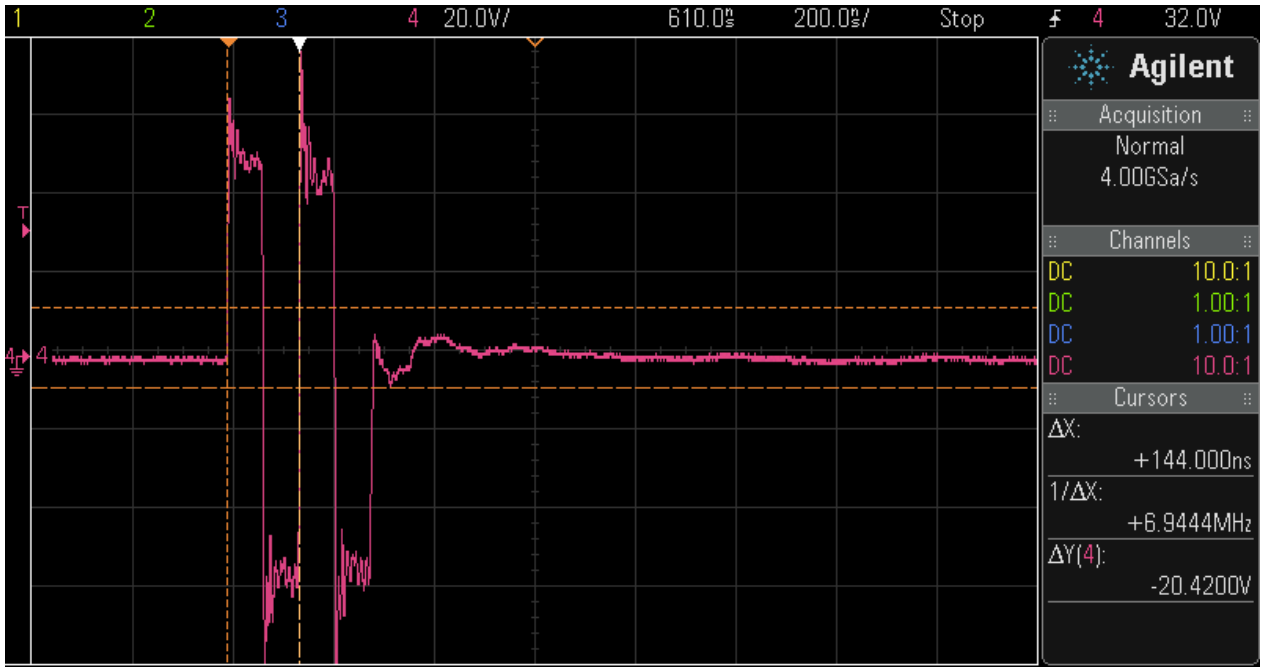

Figura 55: Resultados de pulsación para el Pulser v.3.0 con frecuencia de operación a 7 [MHz].

Esta tercera prueba, como se puede ver en la Figura 55, también se llevo a cabo de forma satisfactoria, logrando formas de onda bien definidas, un clamping funcional y generando un poco más de pérdidas por conmutación, lo que se vuelve razonable, pues se aumentó al doble la frecuencia de operación.

Con las pruebas de funcionamiento básico ya aprobadas, se pueden asumir cumplidos los requerimientos mínimos para llevar a cabo la prueba más significativa, representativa e importante de esta memoria, que corresponde a la visualización de ecos en los mismos terminales de pulsación.

Se procede entonces a la aplicación de gel ecográfico al transductor de ultrasonido, y se enfrenta a un antebrazo humano para visualizar las señales de retorno.

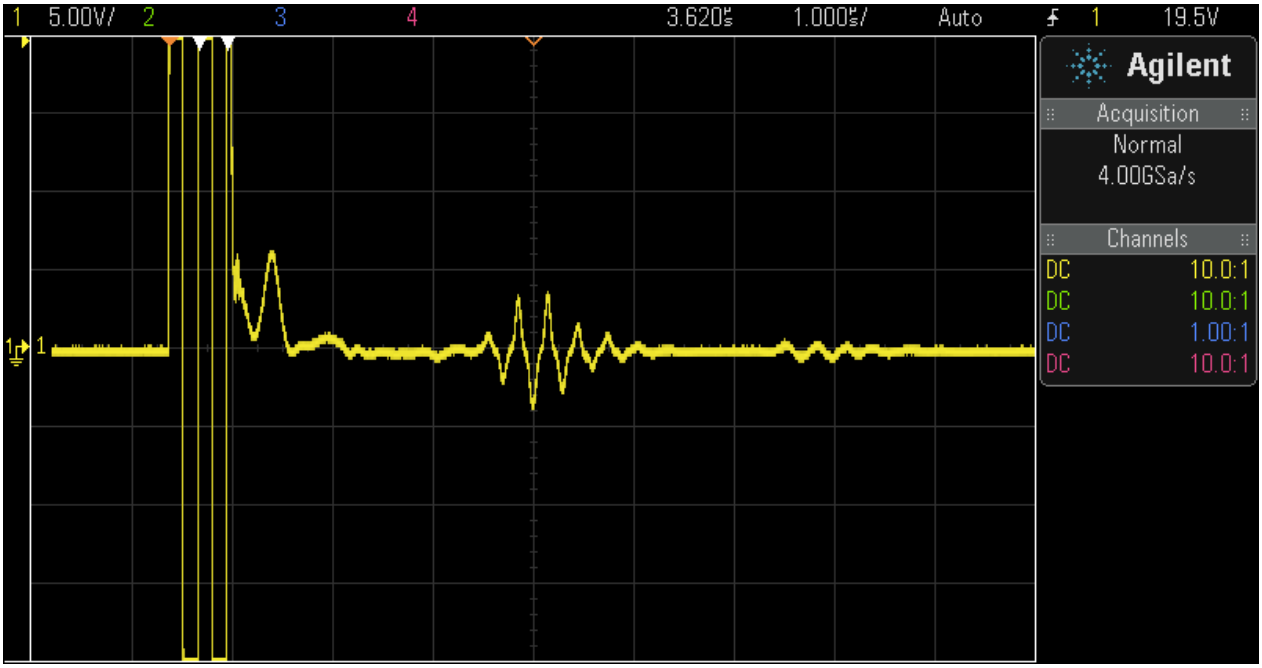

Figura 56: Resultados de pulsación y recepción de ecos satisfactoria.

Efectivamente, como se puede visualizar en la Figura 56, retornan señales de eco con la frecuencia de pulsación como portadora en distintas magnitudes a lo largo del horizonte de tiempo. Este corresponde al resultado más importante de este trabajo, pues el diseño, cálculo, simulaciones e implementaciones logran el objetivo de conseguir ecos volviendo desde los tejidos hacia el transductor, los cuales podrán ser conformados y finalmente procesados para mostrar una imagen de ultrasonido. Se concluye de forma exitosa el proceso de diseño y pruebas para el sistema de pulsación y multiplexión.

4.3 Desarrollo de núcleos para recepción digital

En este apartado se mostrarán los desarrollos realizados para la implementación, puesta en marcha y generación de pruebas, que requiere parcialmente el sistema de recepción en su etapa digital. Los desarrollos aquí mostrados están íntegramente programados en lenguaje de descripción de hardware Verilog® HDL, lo que tiene ventajas comparativas de versatilidad, escalabilidad y rapidez, respecto de sistemas basados en microprocesadores secuenciales.

Las pruebas son realizadas en una plataforma de desarrollo de Altera® basada en una FPGA Cyclone 2™ de alta gama, con periféricos listos para operar; tal como se muestra en la Figura 57. Esto permite disminuir los tiempos de diseño e implementación y optimizar los algoritmos en el vuelo, dejando solo la necesidad de migrar los sistemas, lo cual son procesos de un paso en las FPGAs por su arquitectura inherente.

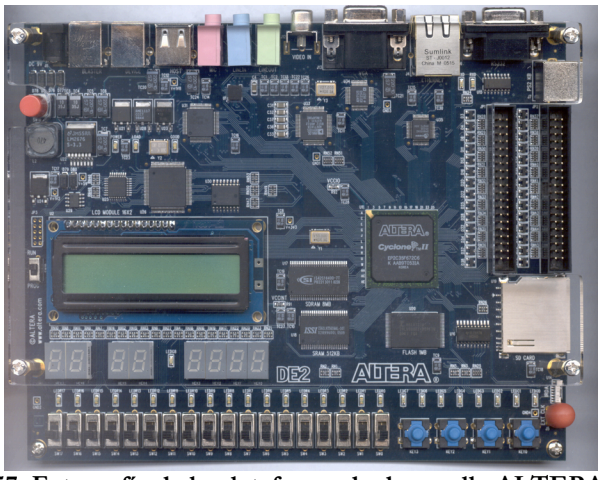

Figura 57: Fotografía de la plataforma de desarrollo ALTERA® DE2.

Para la generación del sistema de adquisición digital básico, se propone la implementación de los siguientes *cores*[1], con el esquema relacional mostrado en la Figura 58:

- Serializador/deserializador para protocolo de comunicación a través de pares diferenciales LVDS.
- Etapa de interfaz para abstracción de hardware de memoria SRAM.
- Generador ajustable de transmisiones RS232 para comunicaciones con PC HOST.

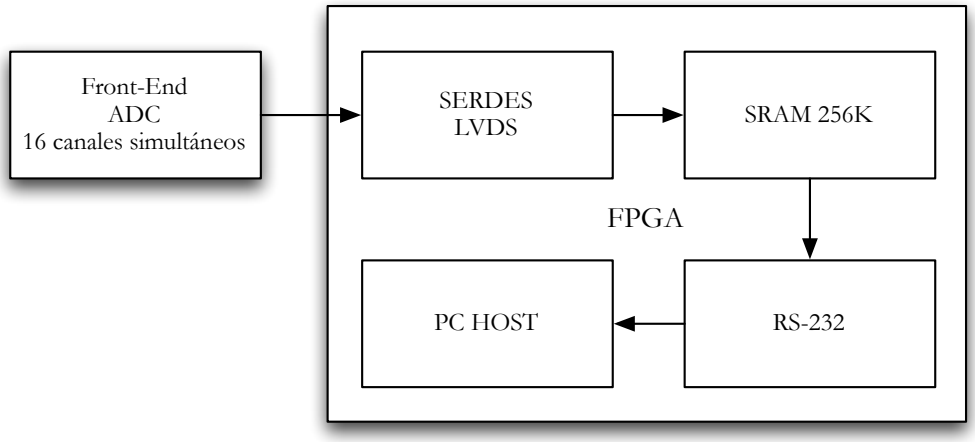

Figura 58: Esquema relacional de cores a desarrollar para interfaz digital de recepción.

[1] Núcleos funcionales y atómicos programados en HDL, que permiten su integración a cualquier plataforma de FPGA en forma modular. Su análogo de programación secuencial son los procedimientos y funciones.

4.3.1 SERDES LVDS

Este núcleo es bastante sencillo, ya que consiste únicamente en un serializador y deserializador de datos que tienen un protocolo de transmisión robusto a las interferencias electromagnéticas, permitiendo manejar altas velocidades de comunicación. De esta forma, el módulo no posee una gran dificultad a nivel de software, si no que lo interesante es la electrónica de emisión/recepción; sin embargo, esta ya está resuelta dentro de estas FPGAs, las cuales poseen canales dedicados para manejo de señales LVDS y mini-LVDS.

Adicionalemente, el serializador y el deserializador, está disponible como *core* pre-fabricado en el Quartus II™ de Altera®, por lo tanto el paso de implementación es directo en este entorno, donde la configuración está manejada por un asistente especialmente diseñado para esto.

Luego de la rápida implementación se hicieron pruebas de transmisión a 100 [Mbps], las cuales arrojaron resultados positivos con integridad de datos al 100%

4.3.2 SRAM

Este módulo es desarrollado íntegramente al bajo nivel, realizando un estudio del periférico específico incluido en la plataforma de desarrollo DE2; para este caso se busca construir un núcleo que permita el manejo intuitivo de la memoria SRAM al alto nivel, y adicionalmente un sistema pequeño que permita hacer pruebas de integridad de datos a distintas velocidades, con el fin de poder establecer los límites críticos de frecuencia de acceso al circuito integrado tanto para lectura como para escritura.

La memoria RAM estática en este caso es la IS61LV25616AL de ISSI®, la cual cuenta con 256 [KB] de memoria volátil accesibles a través de un bus de 16 bits de datos y un bus de 8 bits de dirección de memoria, que está físicamente alambrada de forma directa con la FPGA central de la plataforma de desarrollo. Los códigos implementados pueden ser consultados en el Anexo 9: Núcleo para utilización y pruebas del módulo de manejo para SRAM.

Se procede a la etapa de pruebas de este sistema, para así poder medir las capacidades de velocidad máxima con la integridad de datos deseada. Esta prueba consiste en el grabado de un set de memoria con un dato conocido, el cual luego es leído, para comparar los resultados y concluir respecto de la validez de los datos rescatados desde la memoria. Este proceso es realizado con velocidad y datos variables, con una frecuencia máxima establecida; implementando dos indicadores monoestables mostrados en LEDs. Las condiciones para el encendido de estos son:

Flag de dato correcto : Se enciende si el dato recibido fue correcto.

Flag de dato incorrecto : Se enciende si el dato recibido fue incorrecto.

La ventaja de este sistema es que los LEDs se inicializan apagados y al encenderse no vuelven a apagarse a menos que se resetee el sistema, además en cada iteración solo tiene dos opciones, o el dato es correcto o incorrecto, así la única forma de que haya datos íntegros durante toda la duración de la prueba es que solo el "Flag de dato correcto" esté encendido y el otro apagado todo el tiempo. Los resultados de estabilidad para este sub-sistema son los mostrados en la Tabla 6.

Pruebas de velocidad de acceso				
Main clock [MHz]	Acción	Max Freq. [MHz]	Min. Time[nS]	Resultado
100	Lectura/Escritura	50	20	✓
125	Lectura/Escritura	62.5	16	✓
150	Lectura/Escritura	75	13.3	×

Tabla 6: Resultados de pruebas exhaustivas de integridad de datos para SRAM durante 30 minutos.

Por lo tanto, se pueden realizar ciclos de escritura/lectura a 62.5[MHz] de forma efectiva, con integridad de datos asegurada en un 100%.

4.3.3 RS232

Se implementa un *core* en Verilog que permita la transmisión de datos a través del protocolo RS232 a un computador host en un puerto USB, a través de un adaptador FTDI RS232-USB. Este corresponde a una gran máquina de estados, con pequeños procesos anidados, ya que el protocolo RS232 es una transmisión serial secuencial, por lo tanto se deben respetar tiempos de ejecución y tasas de transferencia. El código implementado está disponible en el Anexo 10: Núcleo para utilización del módulo de transmisiones RS232..

Con el módulo diseñado se pueden realizar pruebas de velocidad de transmisión, tal que los datos lleguen correctamente al computador host. La prueba se realiza enviando un mensaje predefinido que cambia de forma conocida y se compara en el computador con la consigna. Para este fin se diseña una pequeña aplicación en Microsoft® Visual Basic™ que permita llevar a cabo la recepción y las comparaciones respectivas. Los códigos están disponibles en el Anexo 11: Código de software de recepción para el host en Visual Basic., también se puede ver una captura del programa en ejecución en la Figura 59.

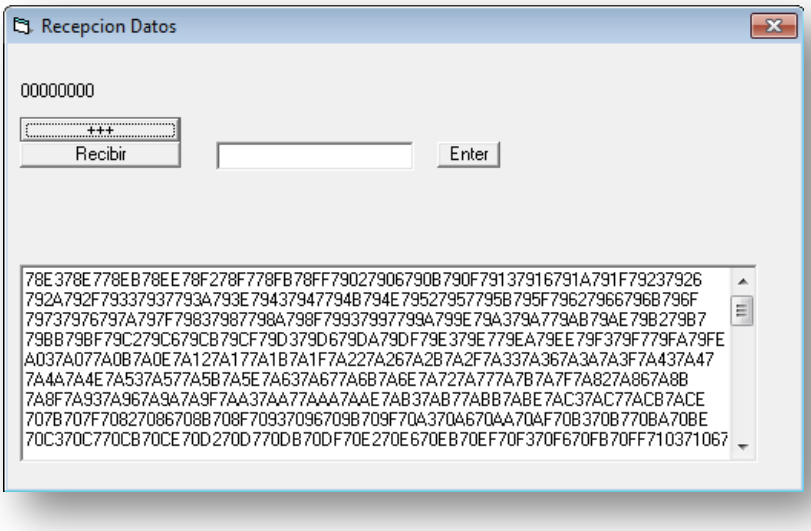

Figura 59: Captura de programa de recepción de datos en host.

Los resultados obtenidos de estas pruebas de velocidad de transmisión se muestran en la Tabla 7, donde se puede observar el funcionamiento del sistema mediante dos interfaces:

- Adaptador MAX232, el cual tiene la norma original RS232 con tensiones de ± 12[V] y adapta las señales para comunicarse con tecnologías CMOS como el host USB.
- Adaptador FT232, el cual tiene normas de tensión CMOS manteniendo el protocolo de comunicaciones pero con excursiones de tensión mucho menores, lo que permite tiempos de conmutación mas cortos, y por lo tanto mayores velocidades de transmisión.

Pruebas de velocidad de acceso				
Acción	Velocidad [bps]	Velocidad [kbps]	Resultado MAX232	Resultado FT232
Envío DE2 a PC	115200	10	✓	✓
Envío DE2 a PC	230400	20	✓	✓
Envío DE2 a PC	460800	40	✓	✓
Envío DE2 a PC	921600	80	×	✓
Envío DE2 a PC	1843200	160	×	✓
Envío DE2 a PC	3686400	320	×	×

Tabla 7: Resultados para pruebas de transmisión de datos por RS232 a distintas velocidades.

Como se puede analizar de la Tabla 7, el núcleo desarrollado y la conexión mediante un adaptador FT232 de FTDI® permite transmitir datos al host a una velocidad de 160 [kbps], resultados muy buenos en considerando las transmisiones comunes mediante comunicación serial USART[1].

4.3.4 Implementación del esquema final completo

En el contexto de generar un sistema completo que realice todo el ciclo mostrado en la Figura 58, se procede a la implementación del módulo en Verilog HDL, que une de forma coherente los núcleos desarrollados para así conformar el método de transmisión para datos provenientes desde los conversores análogos digitales, hasta un archivo que los albergue para realizar su conformación offline en un computador.

El sistema se dividió en varios bloques funcionales, separados en archivos como se muestra a continuación:

- **Módulo TOP**: Este es el maestro global de procesos, por lo que es el que llama a los núcleos en distintos instantes y es capaz de controlar funciones al alto nivel
- **PLLALP**: Consiste en la implementación mediante asistente de un PLL para entregar la mayor frecuencia necesaria, la cual será dividida (mediante contadores binarios), para conseguir sub-frecuencias asociadas.
- **f_divisor**: Módulo de división de frecuencias, que son utilizadas para el envío de datos.
- **send_16_hex**: Genera la tira de datos que se enviarán por protocolo RS232 al pin de salida.
- **SerialTransmitter**: Realiza el envío de la tira de datos, manejando el control a bajo nivel para códigos de standby, inicio de transmisión y tasas de transferencia.
- **dacsgn**: Genera las señales de sincronía para la adquisición de datos análogos a través de conversores análogo/digitales con codificación sigma-delta serializada.

[1] Universal Synchronous Asynchronous Receiver/Transmitter, protocolo de comunicación serial de gran utilización a nivel doméstico e industrial, para comunicaciones de rápida instalación y versatilidad. Consiste básicamente, en una señal de standby, la cual apenas aparece un flanco de bajada envía los datos en forma serial a una frecuencia conocida por el receptor y el emisor, coincidente con la tasa de transmisión.

- **multiplexor_dirección**: Permite realizar el manejo de permutaciones y cambios en el bus de direcciones de la memoria SRAM.
- **muestra_digitos**: Un pequeño núcleo de conversión que permite el despliegue de información en displays de 7 segmentos.
- **monoestable_50mhz**: Un núcleo que genera un pulso monoestable de 20 [nS].

Capítulo 5

Conclusiones Generales y Trabajo Futuro

5.1 Conclusiones Generales

El proyecto presentado en esta memoria, si evoluciona a una fase comercial podrá tener un impacto social importante, pues podrá aumentar ostensiblemente el acceso a la salud de una gran cantidad de personas que requieren la espera de meses para dilucidar un diagnóstico. En este contexto, el desarrollo, implementación y construcción de este dispositivo, generará un nuevo paradigma respecto de la atención primaria de salud en Chile, mejorando los tiempos de emisión de diagnósticos para distintas enfermedades que hoy deben esperar a la realización de exámenes ecográficos.

Respecto a los aspectos más técnicos, podemos ver que los objetivos fueron cumplidos a cabalidad, ya que se completó de forma integra la cadena de diseño, partiendo desde el estudio de la tecnología actual, llegando a la construcción de un dispositivo que se puede considerar estado del arte en el ámbito de pulsación de piezoeléctricos para equipos de ultrasonografía. El factor de sostener ligadas la teoría con la implementación, tuvo una implicancia directa en la rápido prototipaje y análisis de fallas, permitiendo así un desarrollo ágil y consistente. Si bien, se montaron varias plataformas antes de llegar a la definitiva, se logró obtener un dispositivo funcional, eficiente y con gran robustez a los ciclos de trabajo. En particular, la última versión se mantuvo en operación durante 2 días continuos sin cesar, prueba de la cual el sistema tuvo resultados exitosos.

Algunos de los temas debieron profundizarse para enfrentar los diseños, son por ejemplo: la operación de transistores a altas frecuencias, tema no solo útil en ultrasonido, si no que para aplicaciones de iluminación LED, backlights de pantallas, fuentes switching de pequeña escala y muchas más. Se adquirió experiencia respecto del cálculo de corrientes y tensiones necesarias para el encendido de un transistor MOSFET, como se acoplan los fenómenos de compuerta con los de carga, y las consideraciones que se deben tener manejar cargas capacitivas. La selección del componente adecuado también se vuelve fundamental, no solo en capacidades sino que sus tamaños, la forma en que se pueden localizar dentro de un circuito impreso y aspectos relevantes a considerar si se apunta hacia la miniaturización.

De la misma forma que aparecen muchos cuidados que se deben tener a la hora de diseñar, también desaparecen muchos mitos que siempre rondan los proyectos de desarrollo; una situación real respecto de esto son las fuentes de poder, y la factibilidad de poder manejar parámetros sensibles, como la frecuencia de conmutación que hacen que sea más robusta o más adecuada para

una cierta aplicación, sin la necesidad de tener que crear grandes construcciones de control y monitoreo para que sean estables.

El sistema de multiplexión fue también un tema de importancia tratado, pues permitió realizar otro tipo de desarrollos, más desconectados de diseño, y orientados más bien al uso de la experiencia en diseños para lograr dilucidar como una plataforma opera, llegando incluso a poder manejarla de forma integra, tal como si se fuese el fabricante. En este caso puntual, el hecho de haber realizado este trabajo exitosamente, permite economizar grandes costos para la manufactura de un transductor con todo lo que esto implica; así, se puede comprar el sistema ya fabricado en China por miles de unidades a valores muy inferiores a los obtenidos en Europa o Estados Unidos y con soporte en variedad de alternativas. Este desarrollo de ingeniería inversa, permite finalmente reducir los costos del producto final, y por lo tanto generar un impacto social mucho más grande aún.

Se demostró también, como una simulación con modelos y parámetros bien ajustados, puede ser muy representativa de la realidad. El uso de herramientas como LTSPICE™ de Linear Technologies®, fue fundamental para afinar parámetros de componentes pasivos y así evitar horas de experimentación empírica; el hecho de poder cambiar el valor de una capacitancia con un clic y que efectivamente ese cambio sea fidedigno de los fenómenos reales es muy apreciado. De hecho se puede ver como muchas de las herramientas que se usaron como las plataformas de desarrollo de Altera®, permiten disminuir fuertemente el llamado "time-to-market", ya que se puede hacer prototipaje y pruebas en cuestión de horas, y no tener la estricta necesidad de estar fabricando placas para cada prueba que se desea realizar.

Los resultados de la plataforma de pulsación, son aceptables considerando un sistema que deberá excitar 80 canales de transmisión de pulsos ultrasónicos de dimensiones muy pequeñas. Los consumos alcanzados para un régimen de operación típico y continuo fueron de 0.8[W], lo cual asegura que el diseño efectivamente es adecuado para implementaciones portátiles. Los circuitos de pulsación MAX4940A, disiparon potencias muy bajas en el encapsulado por lo que también se puede concluir que se pueden incorporar en carcasas sin ventilación forzada, lo que también es apto para mantener la eficiencia y el tamaño reducido.

Finalmente y considerado el resultado más importante de este trabajo, por la demostración explicita del correcto funcionamiento, se obtuvieron ecos de retorno en los pines de pulsación, los cuales se capturarán, conformarán y procesarán para obtener una imagen de ultrasonido. Este resultado es sin duda el más importante, pues demuestra que el sistema está multiplexando y pulsando correctamente, generando un frente de ondas que se recepciona con los retardos y atenuaciones correspondientes, entregando la información necesaria para construir una imagen de ecografía.

5.2 Trabajo Futuro

En el contexto del proyecto, el trabajo futuro está siendo actualmente realizado, el cual consta del diseño y la implementación de los sub-sistemas adicionales necesarios para construir completamente un ecógrafo ultra-portátil, esto incluye el diseño de una fuente de poder que posea los conversores de potencia que toman la energía de una batería y entregan el arreglo de fuentes que permiten energizar todo el sistema; particularmente este es un completo tema de investigación, pues los niveles de tensión que se deben tener presentes son variadas en magnitud, capacidad y polaridad.

Adicionalmente, se debe llevar a cabo el diseño e implementación de la etapa de adquisición análoga de los ecos, la cual consiste en un front-end de alta precisión, con filtros incorporados que permitan reducir los ruidos no deseados, y amplificadores sintonizados para procesar señales de

niveles coherentes. Este diseño tiene una gran cantidad de aristas, ya que las alimentaciones deben tener zumbidos muy controlados, y se debe tener especial cuidado en el ruteo de las pistas y la aislación de circuitos para evitar la máxima cantidad de interferencias en las señales muestreadas.

Las ondas capturadas deben ser conformadas, siguiendo exhaustivamente los pasos mencionados en la fundamentación teórica de este trabajo; en este aspecto, hay también trabajo a realizar, puesto que estos desarrollos deben ser realizados en arquitecturas basadas en FPGA, por lo tanto, se debe generar una serie de núcleos que llevarán a cabo todas las tareas de pre-filtrado, up-*sampling*, cálculo de muestras asociadas a retardos y detección de envolvente entre otras.

Finalmente, se deben realizar los diseños que incorporan el post-procesamiento de la imagen, lo que incluye filtrado de reducción de ruidos, detección de bordes y finalmente el proceso de Scan Converter. Si bien, por lo general estos sub-sistemas están implementados en librerías ya existentes de plataformas de procesamiento de señales digitales como los DSP de Texas Instruments®, estos tienen consumos muy elevados de energía, en particular por los osciladores que operan en rangos de los Giga Hertz.

Referencias

[1] B. Haider, "*Power Drive Circuits for Diagnostic Medical Ultrasound*", 18th International Symposium on Power Semiconductor Devices & IC's, Naples, Italy, June 2006.

[2] X. Xiaochen, "*A Low-Cost Bipolar Pulse Generator for High-Frequency Ultrasound Applications*", IEEE Transactions on Ultrasonics, Vol. 54, pp. 443-447, 2007.

[3] J. Brown, "*A Low-Cost, High-Performance Pulse Generator for Ultrasound Imaging*", IEEE Transactions on Ultrasonics, Vol. 49., pp. 848-851, 2002.

[4] A. Kassem, "*A Scan Conversion CMOS Implementation for a portable ultrasonic system*", IEEE Transactions on Ultrasonics, Vol. 3, pp. 1461-1464, May 2003.

[5] M.A. Hassan, "*Modular FPGA-based digital ultrasound beamforming*", Biomedical Engineering (MECBME), 1st Middle East Conference, Cairo, Egypt, pp. 134-137, Feb. 2011.

[6] S. Tang, G. Clement, K. Hynynen, "*A Computer-Controlled Ultrasound Pulser-Receiver System for Transkull Fluid Detection using a Shear Wave Transmission Technique*", IEEE Transactions on Ultrasonics, Vol. 9, pp. 1772-1783, Sept. 2007.

[7] M. Fuller, T. Blalock, J. Hossack, W. Walker, "*Novel transmit protection scheme for ultrasound systems*", Dept. of Biomed. Eng., Virginia Univ., Charlottesville, VA, IEEE Transactions on Ultrasonics, Vol. 54, pp. 79-86, January 2007.

[8] L. Jian-yu, "*Development of a linear power amplifier for high frame rate imaging system [biomedical ultrasound imaging applications]",* Dept. of Bioeng., Toledo Univ., OH, USA, IEEE Ultrasonics Symposium, Vol. 2, pp. 1413-1416, Aug. 2004.

[9] G. Xiaorong, "*Transmitting circuit design for ultrasonic transducer*", Electronic and Mechanical Engineering and Information Technology (EMEIT), Chengdu, China, Vol. 3, pp. 1291-1293, Aug. 2011.

[10] B. Dufort, "*Digitally controlled high-voltage analog switch array for medical ultrasound applications in thin-layer silicon-on-insulator process*", SOI Conference, IEEE International, NY, USA. Vol. 3, pp. 78-79, Oct. 2002.

[11] G. Wu, "*A one-way MOS analog switch*", Solid-State and Integrated Circuit Technology, Beijing, China, Vol. 2, pp. 413-415, Oct. 1998.

[12] J. Steensgaard, "*Bootstrapped low-voltage analog switches*", Circuits and Systems, ISCAS '99. Proceedings of the 1999 IEEE International Symposium, Lyngb, Denmark, Vol. 2, pp. 29-32, Jul. 1999.

[13] D. Neamen, "*Semiconductor Physics and Devices*", Third Edition, University of New Mexico, McGraw Hill, 2003

[14] J.M. Albella, J. Martinez-Duart, "*Fundamentos de electrónica física y microelectrónica*", Addison-Wesley, Universidad Autónoma de Madrid, 1999

[15] A. Sedra, "*Circuitos Microelectrónicos*", Cuarta Edición, University of Toronto, Oxford University Press, 2002.

[16] A. Noah, S. Lucas, "*MOSFETs: Properties, Preparations and Performance*", Nova Science Publishers, 2008.

[17] P. Ashenden, "*Digital Design, An Embedded System Approach using Verilog*", University of Adelaide, Morgan Kaufmann Publishers, 2008.

[18] D. Thomas, "*The Verilog Hardware Description Language*", Kluwer Academic Publishers, 2002.

[19] J. Lee, "*VERILOG Quickstart*", Third Edition, Kluwer Academic Publishers, 2002.

[20] J. Hamblen, M. Furman, "*Rapid Prototyping of Digital Systems*", Second Edition, Georgia Institute of Technology, Springer Science+Business Media Inc., 2006.

[21] "*Laboratory Examples and Projects, Advanced Microcontroller Design and system-on-chip*", class notes for ECE 5760, Department of Electrical Engineering, University of Cornell, Fall 2011.

[22] M. Tanabe, "*Ultrasound Imaging*", First Edition, Intech Publishers, Rijeka-Croatia, 2011.

Anexos

Anexo 1: Diagramas de sistema de pulsación “Pulser v.1.0”.

En las figuras a continuación se muestran los esquemáticos en detalle y general de la primera versión del sistema de pulsación.

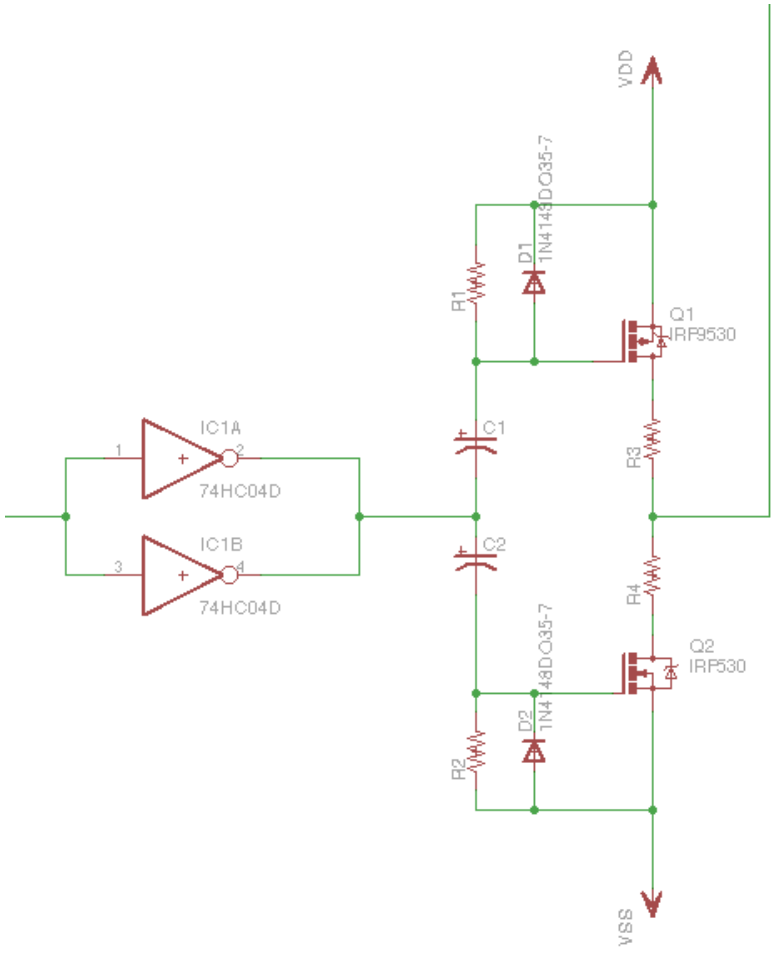

Figura 60: Detalle de un módulo de pulsación para la plataforma Pulser v.1.0.

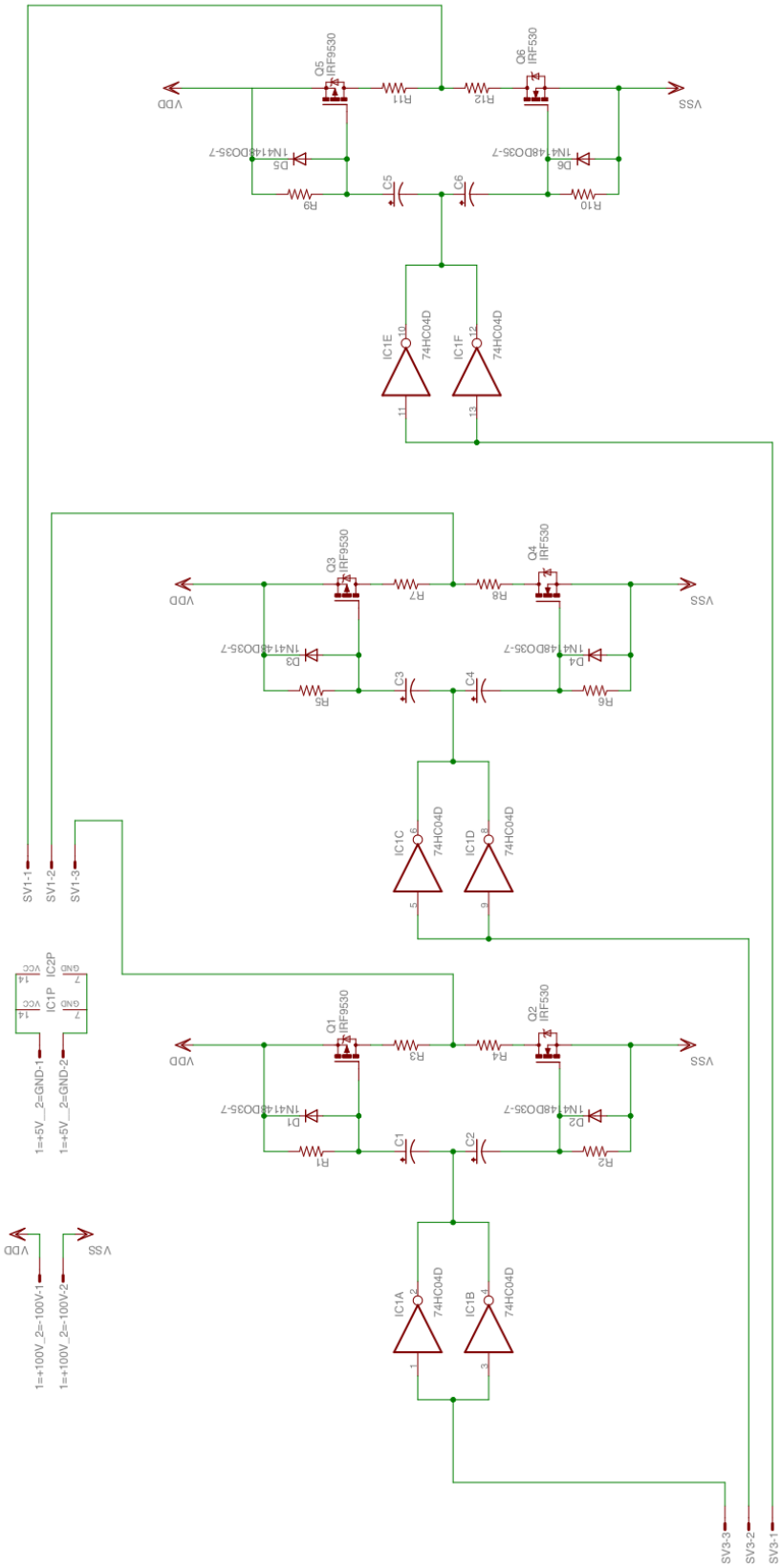

Figura 61: Esquemático general para la plataforma de pulsación Pulser v.1.0.

Anexo 2: Código C para 16F877A decodificador de multiplexión

```
#include "C:\Users\Manuel\Desktop\Multiplexion\mux.h"
int m0[16];
int m1[16];
int m2[16];
int m3[16];
int m4[16];
int i;
int j;
void mensaje(){
   for (i=0;i<=15;i++){
      if (m0[i]==1){
         output_high(PIN_B0);
      }else{
         output_low(PIN_B0);
      }
      if (m2[i]==1){
         output_high(PIN_B2);
      }else{
         output_low(PIN_B2);
      }
      if (m3[i]==1){
         output_high(PIN_B3);
      }else{
         output_low(PIN_B3);
      }
      if (m1[i]==1){
         output_high(PIN_B1);
      }else{
         output_low(PIN_B1);
      }
      if (m4[i]==1){
         output_high(PIN_B4);
      }else{
         output_low(PIN_B4);
      }
      output_low(PIN_B7);
      output_high(PIN_B7);
      output_low(PIN_B7);
   }
   output_low(PIN_B6);
   output_high(PIN_B6);
}
void setzero(){
   m0[0]=0;    m1[0]=0;    m2[0]=0;    m3[0]=0;    m4[0]=0;
   m0[1]=0;    m1[1]=0;    m2[1]=0;    m3[1]=0;    m4[1]=0;
   m0[2]=0;    m1[2]=0;    m2[2]=0;    m3[2]=0;    m4[2]=0;
   m0[3]=0;    m1[3]=0;    m2[3]=0;    m3[3]=0;    m4[3]=0;
   m0[4]=0;    m1[4]=0;    m2[4]=0;    m3[4]=0;    m4[4]=0;
   m0[5]=0;    m1[5]=0;    m2[5]=0;    m3[5]=0;    m4[5]=0;
   m0[6]=0;    m1[6]=0;    m2[6]=0;    m3[6]=0;    m4[6]=0;
   m0[7]=0;    m1[7]=0;    m2[7]=0;    m3[7]=0;    m4[7]=0;
   m0[8]=0;    m1[8]=0;    m2[8]=0;    m3[8]=0;    m4[8]=0;
   m0[9]=0;    m1[9]=0;    m2[9]=0;    m3[9]=0;    m4[9]=0;
   m0[10]=0;   m1[10]=0;   m2[10]=0;   m3[10]=0;   m4[10]=0;
   m0[11]=0;   m1[11]=0;   m2[11]=0;   m3[11]=0;   m4[11]=0;
   m0[12]=0;   m1[12]=0;   m2[12]=0;   m3[12]=0;   m4[12]=0;
   m0[13]=0;   m1[13]=0;   m2[13]=0;   m3[13]=0;   m4[13]=0;
   m0[14]=0;   m1[14]=0;   m2[14]=0;   m3[14]=0;   m4[14]=0;
   m0[15]=0;   m1[15]=0;   m2[15]=0;   m3[15]=0;   m4[15]=0;
}
void main()
{
   setup_adc_ports(AN0_AN1_AN3);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   output_high(PIN_B6);

   output_low(PIN_B4);
   output_low(PIN_B3);
   output_low(PIN_B2);
   output_low(PIN_B1);
   output_low(PIN_B0);
   // B6 = LATCH
```

```
   // B7 = CLOCK
   j=0;
   delay_ms(1000);
   setzero();
   m0[j]=1; m1[j]=0; m2[12]=1; m3[j]=0; m4[j]=0;
   mensaje();
   while (true){
   }
}
```

Anexo 3: Esquema funcional de plataforma de control para multiplexión y pulsación.

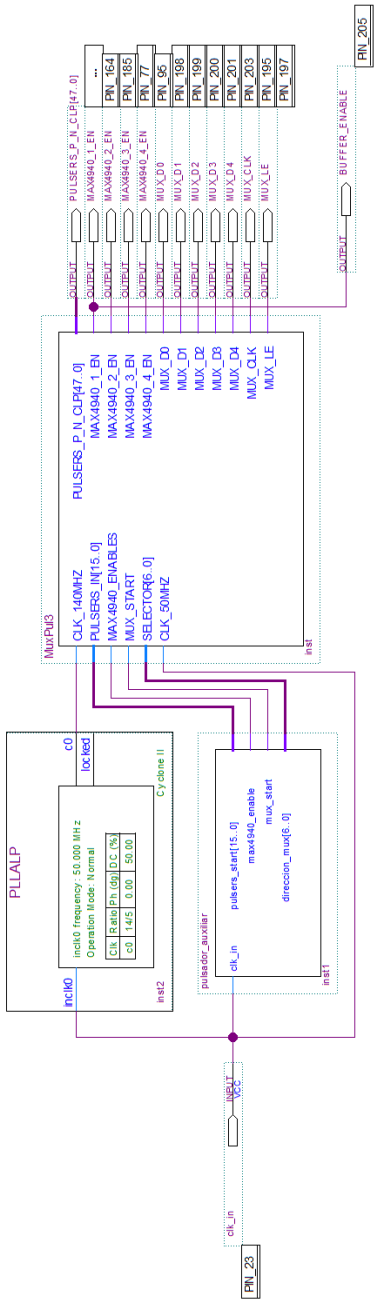

Figura 62: Diagrama de bloques global del sistema de multiplexión y pulsación asíncrono.

Anexo 4: Módulo de comandos abstractos de multiplexión y pulsación.

```
////////////////////////////////////////////////
/////  Modulo de multiplexion y pulsacion ///////
/////                          AUXILIAR                                                    ///////
////////////////////////////////////////////////
module pulsador_auxiliar(clk_in, pulsers_start, max4940_enable, mux_start, direccion_mux);
    input clk_in;
    output reg mux_start;
         output reg [15:0] pulsers_start;
         output reg max4940_enable;
    output reg [6:0] direccion_mux;
    reg internal_salida;
    reg [31:0] contador ;
         initial
                begin
                        direccion_mux <= 7'd0;
                        contador <= 32'd0;
                        mux_start <= 1'b0;
                        pulsers_start <= 16'b0000000000000000;
                        max4940_enable <= 1'b1;
                end
            always @(posedge clk_in)
            begin
                ////////////////////////////////////////////////
                ////    MULTIPLEXION
                ////////////////////////////////////////////////
                if (contador == 32'd0)
                        begin
                                mux_start <= 1'b1;
                                contador <= contador + 32'd1;

                        end
                ////////////////////////////////////////////////
                ////    PULSACION CANAL S09
                ////////////////////////////////////////////////
                else if (contador == 32'd5000)
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b1111111111111111;
                                contador <= contador + 32'd1;
                        end
                else if (contador == 32'd10000)  // Resetea STARTERS
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b0000000000000000;
                                contador <= contador + 32'd1;
                        end
                ////////////////////////////////////////////////
                ////    PULSACION CANAL S10
                ////////////////////////////////////////////////
                else if (contador == 32'd30000)
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b1111111111111111;
                                contador <= contador + 32'd1;
                        end
                else if (contador == 32'd35000)  // Resetea STARTERS
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b0000000000000000;
                                contador <= contador + 32'd1;
                        end
                ////////////////////////////////////////////////
                ////    PULSACION CANAL S11
                ////////////////////////////////////////////////
                else if (contador == 32'd55000)
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b1111111111111111;
                                contador <= contador + 32'd1;
                        end
                else if (contador == 32'd60000)  // Resetea STARTERS
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b0000000000000000;
                                contador <= contador + 32'd1;
                        end
                ////////////////////////////////////////////////
                ////    PULSACION CANAL S12
                ////////////////////////////////////////////////
```

```
                else if (contador == 32'd80000)
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b1111111111111111;
                                contador <= contador + 32'd1;
                        end
                else if (contador == 32'd85000)  // Resetea STARTERS
                        begin
                                mux_start <= 1'b0;
                                pulsers_start[15:0] <= 16'b0000000000000000;
                                contador <= contador + 32'd1;
                        end
                else if ((contador >= 32'd100000) &&(contador <= 32'd4000000100))
                        begin
                                //direccion_mux <= direccion_mux + 7'd1;
                                contador <= 32'd0;
                        end
                else
                        begin
                                contador <= contador + 32'd1;
                        end
        end
endmodule
```

Anexo 5: Módulo de interrelación y coherencia de datos

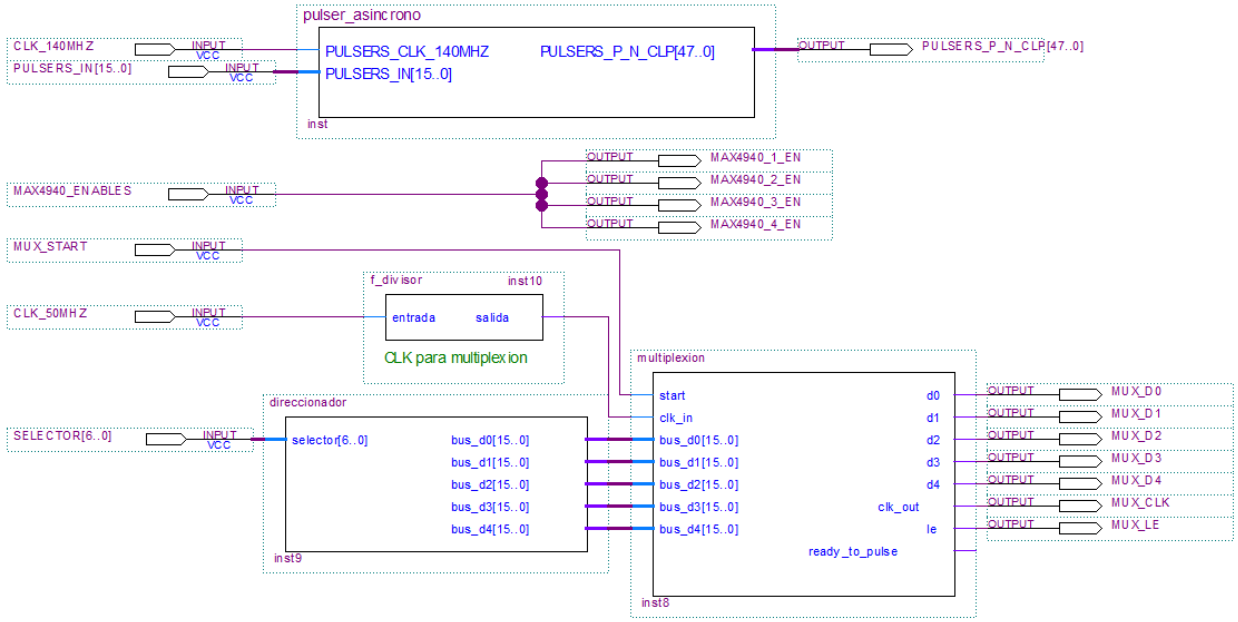

Figura 63: Esquema funcional del módulo de interrelación y coherencia de datos.

Anexo 6: Módulo de interpretación de direccionamiento.

```
module direccionador(selector, bus_d0, bus_d1, bus_d2, bus_d3, bus_d4);
    output reg [15:0] bus_d0 ;
    output reg [15:0] bus_d1 ;
    output reg [15:0] bus_d2 ;
    output reg [15:0] bus_d3 ;
    output reg [15:0] bus_d4 ;
    input [6:0] selector ;
    initial
                begin
                        // SETTING DE DATOS, SE PUEDE MODIFICAR
                        bus_d0 <= 16'b0000000000000000;
                        bus_d1 <= 16'b0000000000000000;
                        bus_d2 <= 16'b0000000000000000;
                        bus_d3 <= 16'b0000000000000000;
                        bus_d3 <= 16'b0000000000000000;
                end
```

```
always @(selector)
begin
              if (selector == 7'd0)
              begin
                      bus_d4<=1111111111111111;
                      bus_d3<=0000000000000000;
                      bus_d2<=0000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd1)
              begin
                      bus_d4<=0111111111111111;
                      bus_d3<=1000000000000000;
                      bus_d2<=0000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd2)
              begin
                      bus_d4<=0111111101111111;
                      bus_d3<=1000000010000000;
                      bus_d2<=0000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd3)
              begin
                      bus_d4<=0011111101111111;
                      bus_d3<=1000100010000000;
                      bus_d2<=0000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd4)
              begin
                      bus_d4<=0011111100111111;
                      bus_d3<=1000100010001000;
                      bus_d2<=0000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd5)
              begin
                      bus_d4<=0001111100111111;
                      bus_d3<=1000100010001000;
                      bus_d2<=1000000000000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd6)
              begin
                      bus_d4<=0001111100011111;
                      bus_d3<=1000100010001000;
                      bus_d2<=1000000010000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd7)
              begin
                      bus_d4<=0000111100011111;
                      bus_d3<=1000100010001000;
                      bus_d2<=1000100010000000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd8)
              begin
                      bus_d4<=0000111100001111;
                      bus_d3<=1000100010001000;
                      bus_d2<=1000100010001000;
                      bus_d1<=0000000000000000;
                      bus_d0<=0000000000000000;
              end
              else if (selector == 7'd9)
              begin
                      bus_d4<=0000011100001111;
                      bus_d3<=1000100010001000;
                      bus_d2<=1000100010001000;
                      bus_d1<=1000000000000000;
                      bus_d0<=0000000000000000;
```

```
end
else if (selector == 7'd10)
begin
        bus_d4<=0000011100000111;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000000010000000;
        bus_d0<=0000000000000000;
end
else if (selector == 7'd11)
begin
        bus_d4<=0000001100000111;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010000000;
        bus_d0<=0000000000000000;
end
else if (selector == 7'd12)
begin
        bus_d4<=0000001100000011;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=0000000000000000;
end
else if (selector == 7'd13)
begin
        bus_d4<=0000000100000011;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000000000000000;
end
else if (selector == 7'd14)
begin
        bus_d4<=0000000100000001;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000000010000000;
end
else if (selector == 7'd15)
begin
        bus_d4<=0000000000000001;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000100010000000;
end
else if (selector == 7'd16)
begin
        bus_d4<=0000000000000000;
        bus_d3<=1000100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000100010001000;
end
else if (selector == 7'd17)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0100100010001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000100010001000;
end
else if (selector == 7'd18)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0100100001001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000100010001000;
end
else if (selector == 7'd19)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0100010001001000;
        bus_d2<=1000100010001000;
        bus_d1<=1000100010001000;
        bus_d0<=1000100010001000;
end
else if (selector == 7'd20)
begin
```

```
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=1000100010001000;
            bus_d1<=1000100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd21)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100100010001000;
            bus_d1<=1000100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd22)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100100001001000;
            bus_d1<=1000100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd23)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001001000;
            bus_d1<=1000100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd24)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=1000100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd25)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=0100100010001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd26)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=0100100001001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd27)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=0100010001001000;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd28)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=0100010001000100;
            bus_d0<=1000100010001000;
    end
    else if (selector == 7'd29)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
            bus_d1<=0100010001000100;
            bus_d0<=0100100010001000;
    end
    else if (selector == 7'd30)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0100010001000100;
            bus_d2<=0100010001000100;
```

```
        bus_d1<=0100010001000100;
        bus_d0<=0100100001001000;
end
else if (selector == 7'd31)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0100010001000100;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001001000;
end
else if (selector == 7'd32)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0100010001000100;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd33)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010010001000100;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd34)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010010000100100;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd35)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100100;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd36)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0100010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd37)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010010001000100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd38)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010010000100100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd39)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100100;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd40)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0100010001000100;
        bus_d0<=0100010001000100;
end
```

```
else if (selector == 7'd41)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010010001000100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd42)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010010000100100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd43)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100100;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd44)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0100010001000100;
end
else if (selector == 7'd45)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010010001000100;
end
else if (selector == 7'd46)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010010000100100;
end
else if (selector == 7'd47)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010001000100100;
end
else if (selector == 7'd48)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0010001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010001000100010;
end
else if (selector == 7'd49)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0001001000100010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010001000100010;
end
else if (selector == 7'd50)
begin
        bus_d4<=0000000000000000;
        bus_d3<=0001001000010010;
        bus_d2<=0010001000100010;
        bus_d1<=0010001000100010;
        bus_d0<=0010001000100010;
end
else if (selector == 7'd51)
begin
        bus_d4<=0000000000000000;
```

```
            bus_d3<=0001000100010010;
            bus_d2<=0010001000100010;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd52)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0010001000100010;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd53)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001001000100010;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd54)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001001000010010;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd55)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010010;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd56)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0010001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd57)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0001001000100010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd58)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0001001000010010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd59)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0001000100010010;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd60)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0001000100010001;
            bus_d0<=0010001000100010;
    end
    else if (selector == 7'd61)
    begin
            bus_d4<=0000000000000000;
            bus_d3<=0001000100010001;
            bus_d2<=0001000100010001;
            bus_d1<=0001000100010001;
```

```
                        bus_d0<=0001001000100010;
                end
                else if (selector == 7'd62)
                begin
                        bus_d4<=0000000000000000;
                        bus_d3<=0001000100010001;
                        bus_d2<=0001000100010001;
                        bus_d1<=0001000100010001;
                        bus_d0<=0001001000010010;
                end
                else if (selector == 7'd63)
                begin
                        bus_d4<=0000000000000000;
                        bus_d3<=0001000100010001;
                        bus_d2<=0001000100010001;
                        bus_d1<=0001000100010001;
                        bus_d0<=0001000100010010;
                end
                else if (selector == 7'd64)
                begin
                        bus_d4<=0000000000000000;
                        bus_d3<=0001000100010001;
                        bus_d2<=0001000100010001;
                        bus_d1<=0001000100010001;
                        bus_d0<=0001000100010001;
                end
                else
                begin
                        bus_d4<=0000000000000000;
                        bus_d3<=0000000000000000;
                        bus_d2<=0000000000000000;
                        bus_d1<=0000000000000000;
                        bus_d0<=0000000000000000;
                end
        end

endmodule
```

Anexo 7: Módulo de comunicación con los multiplexores.

```
////////////////////////////////////////////////////////////////////////////////
//MODULO VERILOG DE MULTIPLEXION
//Manuel E. Toledo - 2011
//
//Asignacion Pines
//d0 -> d4                :               Salida  -       Pines de multiplexion en transductor
//bus_d0 -> d4            :               Entrada -       Bus de entrada para codificación multiplexion
//clk_out                 :               Salida  -       Clock de multiplexion en transductor
//le                      :               Salida  -       Latch de multiplexion en transductor
//ready_to_pulse          :               Salida  -       Flag que indica que esta listo para pulsar
//start                   :               Entrada -       Flanco de subida que indica iniciar disparo
////////////////////////////////////////////////////////////////////////////////

module multiplexion(start, d0, d1, d2, d3, d4, clk_out, le, clk_in,bus_d0, bus_d1, bus_d2, bus_d3, bus_d4,
ready_to_pulse);

        output reg d0,d1,d2,d3,d4;                                  //Definicion de entrada de reloj
        output reg clk_out, le;
        reg [6:0] contador;
        input clk_in;
        input [15:0] bus_d0;
        input [15:0] bus_d1;
        input [15:0] bus_d2;
        input [15:0] bus_d3;
        input [15:0] bus_d4;
        input start;
        reg start_old;
        output reg ready_to_pulse;
        initial
        begin
                d0 <= 1'b0;
                d1 <= 1'b0;
                d2 <= 1'b0;
                d3 <= 1'b0;
                d4 <= 1'b0;

                ready_to_pulse <= 1'b1;
                clk_out <= 1'b0;
                le <= 1'b1;
                start_old <= start;
```

```
        contador <= 7'd85;
end
always @(posedge clk_in)
begin

        if (contador == 7'd0)
                begin
                        ready_to_pulse <= 1'b0;
                        clk_out <= 1'b0;
                        le <= 1'b1;
                        contador <= contador + 7'd1;
                end

        ///////////////// ENVIO DE DATO //////////////////////
        else if (contador == 7'd1)
                begin
                        d0 <= bus_d0[15];
                        d1 <= bus_d1[15];
                        d2 <= bus_d2[15];
                        d3 <= bus_d3[15];
                        d4 <= bus_d4[15];
                        contador <= contador + 7'd1;
                end

        else if (contador == 7'd2)
                begin
                        clk_out <= 1'b1;
                        contador <= contador + 7'd1;
                end
        else if (contador == 7'd4)
                begin
                        clk_out <= 1'b0;
                        contador <= contador + 7'd1;
                end
        ////////////////////////////////////////////////////
        ///////////////// ENVIO DE DATO //////////////////////
        else if (contador == 7'd5)
                begin
                        d0 <= bus_d0[14];
                        d1 <= bus_d1[14];
                        d2 <= bus_d2[14];
                        d3 <= bus_d3[14];
                        d4 <= bus_d4[14];
                        contador <= contador + 7'd1;
                end
        else if (contador == 7'd6)
                begin
                        clk_out <= 1'b1;
                        contador <= contador + 7'd1;
                end
        else if (contador == 7'd8)
                begin
                        clk_out <= 1'b0;
                        contador <= contador + 7'd1;
                end
        ////////////////////////////////////////////////////
        ///////////////// ENVIO DE DATO //////////////////////
        else if (contador == 7'd9)
                begin
                        d0 <= bus_d0[13];
                        d1 <= bus_d1[13];
                        d2 <= bus_d2[13];
                        d3 <= bus_d3[13];
                        d4 <= bus_d4[13];
                        contador <= contador + 7'd1;
                end
        else if (contador == 7'd10)
                begin
                        clk_out <= 1'b1;
                        contador <= contador + 7'd1;
                end
        else if (contador == 7'd12)
                begin
                        clk_out <= 1'b0;
                        contador <= contador + 7'd1;
                end
        ////////////////////////////////////////////////////
        ///////////////// ENVIO DE DATO //////////////////////
        else if (contador == 7'd13)
                begin
                        d0 <= bus_d0[12];
                        d1 <= bus_d1[12];
```

```
                d2 <= bus_d2[12];
                d3 <= bus_d3[12];
                d4 <= bus_d4[12];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd14)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd16 )
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
///////////////// ENVIO DE DATO ////////////////////
else if (contador == 7'd17)
        begin
                d0 <= bus_d0[11];
                d1 <= bus_d1[11];
                d2 <= bus_d2[11];
                d3 <= bus_d3[11];
                d4 <= bus_d4[11];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd18)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd20)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
///////////////// ENVIO DE DATO ////////////////////
else if (contador == 7'd21)
        begin
                d0 <= bus_d0[10];
                d1 <= bus_d1[10];
                d2 <= bus_d2[10];
                d3 <= bus_d3[10];
                d4 <= bus_d4[10];
                contador <= contador + 7'd1;
        end

else if (contador == 7'd22)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd24)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
///////////////// ENVIO DE DATO ////////////////////
else if (contador == 7'd25)
        begin
                d0 <= bus_d0[9];
                d1 <= bus_d1[9];
                d2 <= bus_d2[9];
                d3 <= bus_d3[9];
                d4 <= bus_d4[9];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd26)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd28)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
///////////////// ENVIO DE DATO ////////////////////
else if (contador == 7'd29)
        begin
```

```
                d0 <= bus_d0[8];
                d1 <= bus_d1[8];
                d2 <= bus_d2[8];
                d3 <= bus_d3[8];
                d4 <= bus_d4[8];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd30)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd32)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd33)
        begin
                d0 <= bus_d0[7];
                d1 <= bus_d1[7];
                d2 <= bus_d2[7];
                d3 <= bus_d3[7];
                d4 <= bus_d4[7];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd34)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end

else if (contador == 7'd36)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd37)
        begin
                d0 <= bus_d0[6];
                d1 <= bus_d1[6];
                d2 <= bus_d2[6];
                d3 <= bus_d3[6];
                d4 <= bus_d4[6];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd38)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd40)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd41)
        begin
                d0 <= bus_d0[5];
                d1 <= bus_d1[5];
                d2 <= bus_d2[5];
                d3 <= bus_d3[5];
                d4 <= bus_d4[5];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd42)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd44)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
```

```
else if (contador == 7'd45)
        begin
                d0 <= bus_d0[4];
                d1 <= bus_d1[4];
                d2 <= bus_d2[4];
                d3 <= bus_d3[4];
                d4 <= bus_d4[4];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd46)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd48)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd49)
        begin
                d0 <= bus_d0[3];
                d1 <= bus_d1[3];
                d2 <= bus_d2[3];
                d3 <= bus_d3[3];
                d4 <= bus_d4[3];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd50)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd52)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd53)
        begin
                d0 <= bus_d0[2];
                d1 <= bus_d1[2];
                d2 <= bus_d2[2];
                d3 <= bus_d3[2];
                d4 <= bus_d4[2];
                contador <= contador + 7'd1;
        end

else if (contador == 7'd54)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end

else if (contador == 7'd56)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
        end
////////////////////////////////////////////////////
/////////////// ENVIO DE DATO //////////////////////
else if (contador == 7'd57)
        begin
                d0 <= bus_d0[1];
                d1 <= bus_d1[1];
                d2 <= bus_d2[1];
                d3 <= bus_d3[1];
                d4 <= bus_d4[1];
                contador <= contador + 7'd1;
        end
else if (contador == 7'd58)
        begin
                clk_out <= 1'b1;
                contador <= contador + 7'd1;
        end
else if (contador == 7'd60)
        begin
                clk_out <= 1'b0;
                contador <= contador + 7'd1;
```

```
                        end
                ////////////////////////////////////////////////
                ///////////////// ENVIO DE DATO ////////////////////
                else if (contador == 7'd61)
                        begin
                                d0 <= bus_d0[0];
                                d1 <= bus_d1[0];
                                d2 <= bus_d2[0];
                                d3 <= bus_d3[0];
                                d4 <= bus_d4[0];
                                contador <= contador + 7'd1;
                        end
                else if (contador == 7'd62)
                        begin
                                clk_out <= 1'b1;
                                contador <= contador + 7'd1;
                        end
                else if (contador == 7'd64)
                        begin
                                clk_out <= 1'b0;
                                contador <= contador + 7'd1;
                        end
                ////////////////////////////////////////////////
                ////////    LATCH ///////////////////////////////////
                else if (contador == 7'd66)
                        begin
                                le <= 1'b0;
                                contador <= contador + 7'd1;
                        end
                else if (contador == 7'd68)
                        begin
                                le <= 1'b1;
                                contador <= contador + 7'd1;
                        end
                else if (contador == 7'd70)
                        begin
                                ready_to_pulse <= 1'b1;
                                contador <= contador + 7'd1;
                        end
                ////////////////////////////////////////////////
                else if ((contador >= 7'd71) &&(contador <= 7'd90))
                        begin
                                if ((start_old == 1'b0) &&(start == 1'b1))
                                        begin
                                                start_old <= start;
                                                contador <= 7'd0;
                                        end
                                else
                                        begin
                                                start_old <= start;
                                        end

                        end

                else
                        begin
                                contador <= contador + 7'd1;
                        end
        end
endmodule
```

Anexo 8: Módulo Verilog de pulsación asíncrona.

```
////////////////////////////////////////////////////////////////////////////////
//              MODULO VERILOG DE PULSACION ASINCRONO
//              Manuel E. Toledo - 2011
//
//              Asignacion Pines
//              reloj                   :               Entrada -       Clock de 140 MHZ
//              vpp                     :               Salida  -       Senal de pulsacion positiva
//              vnn                     :               Salida  -       Senal de pulsacion negativa
//              clamper                 :               Salida  -       Senal de pulsacion clamper
//              start                   :               Entrada -       Flanco de subida que indica iniciar
disparo
////////////////////////////////////////////////////////////////////////////////
module genpulsos(reloj, vpp, vnn, clamper, start);
        input start;                            //Flanco para dar inicio a un ciclo de pulsacion
        input reloj;                            //Definicion de entrada de reloj
        output reg vpp, vnn;            //Definicion de salidas de Vpp y Vnn
        output reg clamper;             //Definicion de salida de clamper
        reg [15:0] contador;            //Contador para recorrido de estados
```

```
reg start_old;                          //Auxiliar para deteccion de flanco de subida
initial
begin
        vpp <= 1'b0;
        vnn <= 1'b0;
        clamper <= 1'b0;
        contador <= 16'd121;
        start_old <= start;

end
always @(posedge reloj)
begin
        // Un ciclo son 40 pasos
        // === Generacion de 4 pulsos (1 pos + 1 neg + 1 pos + 1 neg + clamping) ===
        // incluye dead times opcionales
        ////////////////////////////////////////
        /////    1er CICLO DE PULSACION //////////
        ////////////////////////////////////////
        // Levanta Vpp
        if (contador == 16'd0)
                begin
                        vpp <= 1'b1;
                        vnn <= 1'b0;
                        contador <= contador + 16'd1;
                end
        // Se puede incluir dead-time
        // Levanta Vnn
        else if (contador == 16'd20)
                begin
                        vpp <= 1'b0;
                        vnn <= 1'b1;
                        contador <= contador + 16'd1;
                end
        // Se puede incluir dead-time
        ////////////////////////////////////////
        ////////////////////////////////////////
        /////    2do CICLO DE PULSACION //////////
        ////////////////////////////////////////
        // Levanta Vpp
        else if (contador == 16'd40)
                begin
                        vpp <= 1'b1;
                        vnn <= 1'b0;
                        contador <= contador + 16'd1;
                end
        // Se puede incluir dead-time
        // Levanta Vnn
        else if (contador == 16'd60)
                begin
                        vpp <= 1'b0;
                        vnn <= 1'b1;
                        contador <= contador + 16'd1;
                end
        // Se puede incluir dead-time
        ////////////////////////////////////////
        else if (contador == 16'd80)
                begin
                        vpp <= 1'b0;
                        vnn <= 1'b0;
                        clamper <= 1'b1;
                        contador <= contador + 16'd1;
                end
        else if (contador == 16'd280)
                begin
                        clamper <= 1'b0;
                        contador <= contador + 16'd1;
                end
        else if ((contador >= 16'd300) &&(contador <= 16'd800))
                begin
                        if ((start_old == 1'b0) &&(start == 1'b1))
                                begin
                                        start_old <= start;
                                        contador <= 16'b0;
                                end
                        else
                                begin
                                        start_old <= start;
                                end
                end
        else
                begin
                        contador <= contador + 16'd1;
                end
```

```
        end
endmodule
```

Anexo 9: Núcleo para utilización y pruebas del módulo de manejo para SRAM

```
module DE2_TOP();
input           CLOCK_27;               //  27 MHz
input           CLOCK_50;               //  50 MHz
input           EXT_CLOCK;
input   [3:0]   KEY;
input   [17:0]  SW;
output  [6:0]   HEX0;
output  [6:0]   HEX1;
output  [6:0]   HEX2;
output  [6:0]   HEX3;
output  [6:0]   HEX4;
output  [6:0]   HEX5;
output  [6:0]   HEX6;
output  [6:0]   HEX7;
output  [8:0]   LEDG;                   //  LED Green[8:0]
output  [17:0]  LEDR;                   //  LED Red[17:0]
inout   [15:0]  SRAM_DQ;                //  SRAM Data bus 16 Bits
output  [17:0]  SRAM_ADDR;              //  SRAM Address bus 18 Bits
output          SRAM_UB_N;              //  SRAM High-byte Data Mask
output          SRAM_LB_N;              //  SRAM Low-byte Data Mask
output          SRAM_WE_N;              //  SRAM Write Enable
output          SRAM_CE_N;              //  SRAM Chip Enable
output          SRAM_OE_N;              //  SRAM Output Enable
inout   [35:0]  GPIO_0;
inout   [35:0]  GPIO_1;

//////////////////////////////////////////
// Designacion Counter
reg [3:0] shortCount ;
//////////////////////////////////////////
//////////////////////////////////////////
// Designacion de info_estado
// Bit   15:0 = DATA
// Bit  19:16 = MEM. ADDRESS
// Bit     20 = WRITE BIT
reg [20:0] info_estado;
reg falla;
reg bien;
wire reloj_salida;
//////////////////////////////////////////
initial
    begin
        falla <= 1'b0;
        bien <= 1'b0;
        info_estado <= 21'b100001111111111111111;
    end


always @(posedge reloj_salida)
begin
    case (shortCount)

        // ESTADO 1
        4'h0: begin
                    info_estado <= 21'b100001111111111111111;       // Set Datos 1s en Memoria 0
                end
        4'h1: begin
                    info_estado <= 21'b000001111111111111111;       // Set Datos 1s en Memoria 0 (WRITE)
                end
        4'h2: begin
                    info_estado <= 21'b111111111111111111111;       // Set Datos 1s en Memoria 15
                end
        4'h3: begin
                    info_estado <= 21'b011111111111111111111;       // Set Datos 1s en Memoria 15 (WRITE)
                end
        // ESTADO 2
        4'h4:   begin
                    info_estado <= 21'b100000000000000000000;       // Set Datos 0s en Memoria 0
                end
        4'h5: begin
                    if ( (SRAM_DQ-info_estado[15:0]) ==16'b1111111111111111)        // Set Datos 0s en Memoria
0 (READ)
                        begin
                            bien <= 1'b1;
```

```
                    end
                else
                    begin
                        falla <= 1'b1;
                    end
            end
        4'h6:   begin
                    info_estado <= 21'b111110000000000000000;       // Set Datos 0s en Memoria 15
            end
        4'h7: begin
                    if ( (SRAM_DQ-info_estado[15:0]) ==16'b1111111111111111)        // Set Datos 0s en Memoria
0 (READ)
                    begin
                        bien <= 1'b1;
                    end
                else
                    begin
                        falla <= 1'b1;
                    end
            end
        // ESTADO 3
        4'h8: begin
                    info_estado <= 21'b100000000000000000000;       // Set Datos 1s en Memoria 0
            end
        4'h9: begin
                    info_estado <= 21'b000000000000000000000;       // Set Datos 1s en Memoria 0 (WRITE)
            end
        4'ha: begin
                    info_estado <= 21'b111110000000000000000;       // Set Datos 1s en Memoria 15
            end
        4'hb: begin
                    info_estado <= 21'b011110000000000000000;       // Set Datos 1s en Memoria 15 (WRITE)
            end
        // ESTADO 4
        4'hc: begin
                    info_estado <= 21'b100001111111111111111;       // Set Datos 0s en Memoria 0
            end
        4'hd: begin
                    if ( (info_estado[15:0]-SRAM_DQ) ==16'b1111111111111111)        // Set Datos 0s en Memoria
0 (READ)
                    begin
                        bien <= 1'b1;
                    end
                else
                    begin
                        falla <= 1'b1;
                    end
            end
        4'he: begin
                    info_estado <= 21'b111111111111111111111;       // Set Datos 0s en Memoria 15
            end
        4'hf: begin
                    if ( (info_estado[15:0]-SRAM_DQ) ==16'b1111111111111111)        // Set Datos 0s en Memoria
0 (READ)
                    begin
                        bien <= 1'b1;
                    end
                else
                    begin
                        falla <= 1'b1;
                    end
            end
        default begin
                    info_estado <= 21'b100010101010101010101;
                end
    endcase

    shortCount <= shortCount + 1;
end
assign LEDG[3:0] = shortCount;
assign LEDG[7] = falla;
assign LEDG[6] = bien;
PLLALP divisor(CLOCK_50,reloj_salida);
HexDigit Digit4(HEX4, shortCount);
assign SRAM_ADDR = {14'h0, info_estado[19:16]};
assign SRAM_UB_N = 0;
assign SRAM_LB_N = 0;
assign SRAM_CE_N = 0;
assign SRAM_WE_N = info_estado[20];
assign SSRAM_OE_N = 0;
assign SRAM_DQ = (info_estado[20]? 16'hzzzz : info_estado[15:0]);
assign GPIO_1[0]=reloj_salida;
assign GPIO_1[1]=reloj_salida;
```

```
assign GPIO_1[2]=reloj_salida;
assign GPIO_1[3]=reloj_salida;
assign GPIO_1[4]=reloj_salida;
assign GPIO_1[5]=reloj_salida;
assign GPIO_1[6]=reloj_salida;
assign GPIO_1[7]=reloj_salida;
assign GPIO_1[28]=reloj_salida;
assign GPIO_1[29]=reloj_salida;
assign GPIO_1[30]=reloj_salida;
assign GPIO_1[31]=reloj_salida;
assign GPIO_1[32]=reloj_salida;
assign GPIO_1[33]=reloj_salida;
assign GPIO_1[34]=reloj_salida;
assign GPIO_1[35]=reloj_salida;
HexDigit Digit0(HEX0, SRAM_DQ[3:0]);
HexDigit Digit1(HEX1, SRAM_DQ[7:4]);
HexDigit Digit2(HEX2, SRAM_DQ[11:8]);
HexDigit Digit3(HEX3, SRAM_DQ[15:12]);

endmodule //end top level DE2_TOP
////////////////////////////////////////////
////////////////////////////////////////////////
// Decode one hex digit for LED 7-seg display
module HexDigit(segs, num);
    input [3:0] num ;       //the hex digit to be displayed
    output [6:0] segs ;     //actual LED segments
    reg [6:0] segs ;
    always @ (num)
    begin
        case (num)
                4'h0: segs = 7'b1000000;
                4'h1: segs = 7'b1111001;
                4'h2: segs = 7'b0100100;
                4'h3: segs = 7'b0110000;
                4'h4: segs = 7'b0011001;
                4'h5: segs = 7'b0010010;
                4'h6: segs = 7'b0000010;
                4'h7: segs = 7'b1111000;
                4'h8: segs = 7'b0000000;
                4'h9: segs = 7'b0010000;
                4'ha: segs = 7'b0001000;
                4'hb: segs = 7'b0000011;
                4'hc: segs = 7'b1000110;
                4'hd: segs = 7'b0100001;
                4'he: segs = 7'b0000110;
                4'hf: segs = 7'b0001110;
                default segs = 7'b1111111;
        endcase
    end
endmodule
////////////////////////////////////////////

module f_divisor(entrada, salida);
    input entrada   ;       //the hex digit to be displayed
    output salida ;     //actual LED segments
    reg internal_salida;

    reg [31:0] contador ;
    always @(posedge entrada)
    begin
        internal_salida <= contador[1];
        contador <= contador + 1;
    end
    assign salida = internal_salida;
endmodule
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////
`timescale 1 ps / 1 ps
module PLLALP (
    inclk0,
    c0,
    locked);

    input     inclk0;
    output    c0;
    output    locked;

    wire [5:0] sub_wire0;
    wire  sub_wire2;
    wire [0:0] sub_wire5 = 1'h0;
    wire [0:0] sub_wire1 = sub_wire0[0:0];
    wire  c0 = sub_wire1;
    wire  locked = sub_wire2;
```

```
    wire  sub_wire3 = inclk0;
    wire [1:0] sub_wire4 = {sub_wire5, sub_wire3};

    altpll  altpll_component (
                .inclk (sub_wire4),
                .clk (sub_wire0),
                .locked (sub_wire2),
                .activeclock (),
                .areset (1'b0),
                .clkbad (),
                .clkena ({6{1'b1}}),
                .clkloss (),
                .clkswitch (1'b0),
                .configupdate (1'b0),
                .enable0 (),
                .enable1 (),
                .extclk (),
                .extclkena ({4{1'b1}}),
                .fbin (1'b1),
                .fbmimicbidir (),
                .fbout (),
                .fref (),
                .icdrclk (),
                .pfdena (1'b1),
                .phasecounterselect ({4{1'b1}}),
                .phasedone (),
                .phasestep (1'b1),
                .phaseupdown (1'b1),
                .pllena (1'b1),
                .scanaclr (1'b0),
                .scanclk (1'b0),
                .scanclkena (1'b1),
                .scandata (1'b0),
                .scandataout (),
                .scandone (),
                .scanread (1'b0),
                .scanwrite (1'b0),
                .sclkout0 (),
                .sclkout1 (),
                .vcooverrange (),
                .vcounderrange ());
    defparam
        altpll_component.clk0_divide_by = 1,
        altpll_component.clk0_duty_cycle = 50,
        altpll_component.clk0_multiply_by = 2,
        altpll_component.clk0_phase_shift = "0",
        altpll_component.compensate_clock = "CLK0",
        altpll_component.gate_lock_signal = "NO",
        altpll_component.inclk0_input_frequency = 20000,
        altpll_component.intended_device_family = "Cyclone II",
        altpll_component.invalid_lock_multiplier = 5,
        altpll_component.lpm_hint = "CBX_MODULE_PREFIX=PLLALP",
        altpll_component.lpm_type = "altpll",
        altpll_component.operation_mode = "NORMAL",
        altpll_component.port_activeclock = "PORT_UNUSED",
        altpll_component.port_areset = "PORT_UNUSED",
        altpll_component.port_clkbad0 = "PORT_UNUSED",
        altpll_component.port_clkbad1 = "PORT_UNUSED",
        altpll_component.port_clkloss = "PORT_UNUSED",
        altpll_component.port_clkswitch = "PORT_UNUSED",
        altpll_component.port_configupdate = "PORT_UNUSED",
        altpll_component.port_fbin = "PORT_UNUSED",
        altpll_component.port_inclk0 = "PORT_USED",
        altpll_component.port_inclk1 = "PORT_UNUSED",
        altpll_component.port_locked = "PORT_USED",
        altpll_component.port_pfdena = "PORT_UNUSED",
        altpll_component.port_phasecounterselect = "PORT_UNUSED",
        altpll_component.port_phasedone = "PORT_UNUSED",
        altpll_component.port_phasestep = "PORT_UNUSED",
        altpll_component.port_phaseupdown = "PORT_UNUSED",
        altpll_component.port_pllena = "PORT_UNUSED",
        altpll_component.port_scanaclr = "PORT_UNUSED",
        altpll_component.port_scanclk = "PORT_UNUSED",
        altpll_component.port_scanclkena = "PORT_UNUSED",
        altpll_component.port_scandata = "PORT_UNUSED",
        altpll_component.port_scandataout = "PORT_UNUSED",
        altpll_component.port_scandone = "PORT_UNUSED",
        altpll_component.port_scanread = "PORT_UNUSED",
        altpll_component.port_scanwrite = "PORT_UNUSED",
        altpll_component.port_clk0 = "PORT_USED",
        altpll_component.port_clk1 = "PORT_UNUSED",
        altpll_component.port_clk2 = "PORT_UNUSED",
        altpll_component.port_clk3 = "PORT_UNUSED",
```

```
        altpll_component.port_clk4 = "PORT_UNUSED",
        altpll_component.port_clk5 = "PORT_UNUSED",
        altpll_component.port_clkena0 = "PORT_UNUSED",
        altpll_component.port_clkena1 = "PORT_UNUSED",
        altpll_component.port_clkena2 = "PORT_UNUSED",
        altpll_component.port_clkena3 = "PORT_UNUSED",
        altpll_component.port_clkena4 = "PORT_UNUSED",
        altpll_component.port_clkena5 = "PORT_UNUSED",
        altpll_component.port_extclk0 = "PORT_UNUSED",
        altpll_component.port_extclk1 = "PORT_UNUSED",
        altpll_component.port_extclk2 = "PORT_UNUSED",
        altpll_component.port_extclk3 = "PORT_UNUSED",
        altpll_component.valid_lock_multiplier = 1;


endmodule
```

Anexo 10: Núcleo para utilización del módulo de transmisiones RS232.

```
module DE2_TOP ();
   assign HEX0 = 7'h7F;
   assign HEX1 = 7'h7F;
   assign HEX2 = 7'h7F;
   assign HEX3 = 7'h7F;
   assign HEX4 = 7'h7F;
   assign HEX5 = 7'h7F;
   assign HEX6 = 7'h7F;
   assign HEX7 = 7'h7F;
   assign LEDG = 9'h0;
   assign SD_DAT = 1'bz;
   assign SD_CLK = 1'b0;
   assign SRAM_ADDR = 18'h0;
   assign SRAM_CE_N = 1'b1;
   assign SRAM_DQ   = 16'hzzzz;
   assign SRAM_LB_N = 1'b1;
   assign SRAM_OE_N = 1'b1;
   assign SRAM_UB_N = 1'b1;
   assign SRAM_WE_N = 1'b1;
   assign I2C_SCLK = 1'b0;
   assign IRDA_TXD = 1'b0;
   assign TD_RESET = 1'b0;
   assign TDO = 1'b0;
   wire reset ;
   assign reset = ~KEY[0] ;
        wire send_done ;
    send_16_hex sender(
            .number_to_send_in(data),
            .send_strobe_in(sendNew),
            .send_done_out(send_done),
            .uart_transmit_Out(UART_TXD),
            .CLOCK_50_in(CLOCK_50),
            .reset_in(reset) );
    reg [15:0] data ;
    reg sendNew ;
    parameter data_rate = 1 ;
    parameter data_rate_set =  50000000/data_rate ;
    reg [31:0] data_rate_counter ;
    assign GPIO_1[27] = UART_TXD;
         // Generador de tasa de transferencia y datos:
         always @(posedge CLOCK_50)
         begin
             if (reset)
             begin
                 data <= 16'd0 ;
                 sendNew <= 0 ;
             end

             if (data_rate_counter == data_rate_set)
             begin
                 data_rate_counter <= 32'd0 ;
                 sendNew <= 1 ;
                 data <= data + 16'd1 ;
             end
             else
             begin
                 data_rate_counter <= data_rate_counter + 1 ;
                 sendNew <= 0 ;
             end
         end
```

```
      assign LEDR[15:0] = data ;

      endmodule

module send_16_hex(
                        number_to_send_in,
                        send_strobe_in,
                        send_done_out,
                        uart_transmit_Out,
                        CLOCK_50_in, reset_in);
input [15:0] number_to_send_in ;
input send_strobe_in ;
output reg send_done_out ;
output uart_transmit_Out ;
input CLOCK_50_in, reset_in ;
parameter data_rate = 10000;
parameter baud_rate = 115200;
parameter data_rate_set =  50000000/data_rate ;
reg sendNew ;
reg [31:0] data_rate_counter ;
parameter send_d1=4'd1, send_d2=4'd2, send_d3=4'd3, send_d4=4'd4, send_cr=4'd9, send_lf=4'd11 ;
parameter wait_d1=4'd5, wait_d2=4'd6, wait_d3=4'd7, wait_d4=4'd8, wait_cr=4'd10, wait_lf=4'd12 ;
parameter cr = 8'h0d, line_feed = 8'h0a ;
wire [7:0] ascii_char1, ascii_char2, ascii_char3, ascii_char4 ;
// Invocar funciones de conversion de codigos a caracteres
     HexDigit d1(ascii_char1, number_to_send_in[15:12]);
HexDigit d2(ascii_char2, number_to_send_in[11:8]);
HexDigit d3(ascii_char3, number_to_send_in[7:4]);
HexDigit d4(ascii_char4, number_to_send_in[3:0]);
SerialTransmitter uart(
    .CLOCK_50(CLOCK_50_in),
    .UART_TXD(uart_transmit_Out),
    .toTransmit(ascii_char),
    .reset(reset_in) ,
    .sendNew(sendNew),
    .baud_clock(baud_clock)
);
reg [3:0] send_state ;
wire uart_idle ;
reg [7:0] ascii_char ;
reg sending ;
always @(posedge CLOCK_50_in)
begin
    if (reset_in)
    begin
        sendNew <= 0 ;
        send_state <= send_d1 ;
        data_rate_counter <= 0;
        sending <= 0;
    end
    if (send_strobe_in) sending <= 1 ;
    if (data_rate_counter == data_rate_set)
    begin
        data_rate_counter <= 0;
        if (sending)
        begin
        case(send_state)
        send_d1:
        begin
            ascii_char <= ascii_char1 ;
            sendNew <= 1 ;
            send_state <= wait_d1 ;
        end
        wait_d1:
        begin
            sendNew <= 0 ;
            send_state <= send_d2 ;
        end
        send_d2:
        begin
            ascii_char <= ascii_char2 ;
            sendNew <= 1 ;
            send_state <= wait_d2 ;
        end
        wait_d2:
        begin
            sendNew <= 0 ;
            send_state <= send_d3 ;
        end
        send_d3:
        begin
            ascii_char <= ascii_char3 ;
            sendNew <= 1 ;
```

```
                send_state <= wait_d3 ;
            end
            wait_d3:
            begin
                sendNew <= 0 ;
                send_state <= send_d4 ;
            end
            send_d4:
            begin
                ascii_char <= ascii_char4 ;
                sendNew <= 1 ;
                send_state <= wait_d4 ;
            end
            wait_d4:
            begin
                sendNew <= 0 ;
                send_state <= send_lf ;
            end
            send_lf:
            begin
                ascii_char <= line_feed ;
                sendNew <= 1 ;
                send_state <= wait_lf ;
            end
            wait_lf:
            begin
                sendNew <= 0 ;
                send_state <= send_cr ;
            end
            send_cr:
            begin
                ascii_char <= cr ;
                sendNew <= 1 ;
                send_state <= wait_cr ;
            end
            wait_cr:
            begin
                sendNew <= 0 ;
                    send_state <= send_d1 ;
                    sending <= 0;
            end

            endcase
            end
        end
        else
        begin
            data_rate_counter <= data_rate_counter + 1 ;
            sendNew <= 0 ;
        end
    end
    reg baud_clock ;
    reg [31:0] baud_counter ;
    parameter baud_set = (50000000/(2*baud_rate)) - 1 ;
    always @(posedge CLOCK_50_in)
    begin
        if (reset_in) baud_counter <= 32'd0 ;
        if (baud_counter == baud_set)
        begin
            baud_counter <= 32'd0 ;
            baud_clock <= ~baud_clock ;
        end
        else baud_counter <= baud_counter + 1 ;
    end

endmodule

module SerialTransmitter(
    input CLOCK_50,
    output  UART_TXD,
    input [7:0] toTransmit,
    input reset,
    output dataSent_out,
    input sendNew,
    output reg idle,
    output [7:0] lastTransmit,
    input baud_clock
    );

    reg [7:0] txd, last_txd;
    reg [3:0] currBit;
    reg [3:0] transmitState;
```

```
    parameter [3:0] idleWait = 4'd0,
                    sendStartBit = 4'd1,
                    sendData = 4'd2,
                    sendStopBit = 4'd3,
                    return_to_idle = 4'd4 ;

    reg dataSent;
    reg txd_out;
    reg should_send;
    reg begin_bit ;
    assign UART_TXD = txd_out;
    assign dataSent_out = dataSent;
    assign lastTransmit = last_txd;
    always @(posedge CLOCK_50)
    begin
        if(sendNew & (transmitState == idleWait)) begin
            should_send <= 1'b1;
            idle <= 0;
        end
        if(reset) begin
            transmitState <= idleWait;
            dataSent <= 0;
            currBit <= 0;
            txd <= 0;
            txd_out <= 1'b1;
            should_send <= 1'b0;
            begin_bit <= 1'b1 ;
            idle <= 1'b1;
        end

        else begin
            if (baud_clock & begin_bit)
            begin
                begin_bit <= 0 ;
                case(transmitState)
                    idleWait: begin
                        txd_out <= 1'b1;
                        if(should_send)
                        begin
                            txd <= toTransmit;
                            last_txd <= toTransmit;
                            transmitState <= sendStartBit;
                            dataSent <= 1'b0;
                            should_send <= 0;
                        end
                        else idle <= 1'b1 ;
                    end
                    sendStartBit: begin
                        txd_out <= 1'b0;
                        transmitState <= sendData;
                        currBit <= 4'd0;
                    end
                    sendData: begin
                        txd_out <= txd[currBit];
                        currBit <= currBit + 4'd1;
                        if(currBit == 4'd7) begin
                            transmitState <= sendStopBit;
                        end
                    end
                    sendStopBit: begin
                        txd_out <= 1'b1;
                        dataSent <= 1;
                        transmitState <= return_to_idle;
                    end
                    return_to_idle: begin
                        transmitState <= idleWait;
                    end
                endcase
            end
            else if (~baud_clock & ~begin_bit)
            begin
                begin_bit <= 1'b1 ; //
            end
        end
    end
endmodule

//////////////////////////////////////////////////////
// Decodificacion para los 7-segmentos
//////////////////////////////////////////////////////

module HexDigit(ascii, num);
    input [3:0] num ;
```

```
    output [7:0] ascii ;        //Codigo ascii del digito
    reg [7:0] ascii ;
    always @ (num)
    begin
        case (num)
                4'h0: ascii = 7'h30;
                4'h1: ascii = 7'h31;
                4'h2: ascii = 7'h32;
                4'h3: ascii = 7'h33;
                4'h4: ascii = 7'h34;
                4'h5: ascii = 7'h35;
                4'h6: ascii = 7'h36;
                4'h7: ascii = 7'h37;
                4'h8: ascii = 7'h38;
                4'h9: ascii = 7'h39;
                4'ha: ascii = 7'h41;
                4'hb: ascii = 7'h42;
                4'hc: ascii = 7'h43;
                4'hd: ascii = 7'h44;
                4'he: ascii = 7'h45;
                4'hf: ascii = 7'h46;
                default ascii = 7'h30;
        endcase
    end
endmodule

/////////////// Divisor de frecuencia ////////////////////////////////////////////////
module f_divisor(entrada, salida);
    input entrada   ;
    output salida ;
    reg internal_salida;

    reg [31:0] contador ;
    always @(posedge entrada)
    begin
        internal_salida <= contador[1];
        contador <= contador + 1;
    end
    assign salida = internal_salida;
endmodule


/////////////  PLL Multiplicador de frecuencia  ////////////////////////////////////

`timescale 1 ps / 1 ps
module PLLALP (
    inclk0,
    c0,
    locked);

    input     inclk0;
    output    c0;
    output    locked;

    wire [5:0] sub_wire0;
    wire  sub_wire2;
    wire [0:0] sub_wire5 = 1'h0;
    wire [0:0] sub_wire1 = sub_wire0[0:0];
    wire  c0 = sub_wire1;
    wire  locked = sub_wire2;
    wire  sub_wire3 = inclk0;
    wire [1:0] sub_wire4 = {sub_wire5, sub_wire3};

    altpll  altpll_component (
                .inclk (sub_wire4),
                .clk (sub_wire0),
                .locked (sub_wire2),
                .activeclock (),
                .areset (1'b0),
                .clkbad (),
                .clkena ({6{1'b1}}),
                .clkloss (),
                .clkswitch (1'b0),
                .configupdate (1'b0),
                .enable0 (),
                .enable1 (),
                .extclk (),
                .extclkena ({4{1'b1}}),
                .fbin (1'b1),
                .fbmimicbidir (),
                .fbout (),
                .fref (),
                .icdrclk (),
```

```
                .pfdena (1'b1),
                .phasecounterselect ({4{1'b1}}),
                .phasedone (),
                .phasestep (1'b1),
                .phaseupdown (1'b1),
                .pllena (1'b1),
                .scanaclr (1'b0),
                .scanclk (1'b0),
                .scanclkena (1'b1),
                .scandata (1'b0),
                .scandataout (),
                .scandone (),
                .scanread (1'b0),
                .scanwrite (1'b0),
                .sclkout0 (),
                .sclkout1 (),
                .vcooverrange (),
                .vcounderrange ());
    defparam
        altpll_component.clk0_divide_by = 1,
        altpll_component.clk0_duty_cycle = 50,
        altpll_component.clk0_multiply_by = 2,
        altpll_component.clk0_phase_shift = "0",
        altpll_component.compensate_clock = "CLK0",
        altpll_component.gate_lock_signal = "NO",
        altpll_component.inclk0_input_frequency = 20000,
        altpll_component.intended_device_family = "Cyclone II",
        altpll_component.invalid_lock_multiplier = 5,
        altpll_component.lpm_hint = "CBX_MODULE_PREFIX=PLLALP",
        altpll_component.lpm_type = "altpll",
        altpll_component.operation_mode = "NORMAL",
        altpll_component.port_activeclock = "PORT_UNUSED",
        altpll_component.port_areset = "PORT_UNUSED",
        altpll_component.port_clkbad0 = "PORT_UNUSED",
        altpll_component.port_clkbad1 = "PORT_UNUSED",
        altpll_component.port_clkloss = "PORT_UNUSED",
        altpll_component.port_clkswitch = "PORT_UNUSED",
        altpll_component.port_configupdate = "PORT_UNUSED",
        altpll_component.port_fbin = "PORT_UNUSED",
        altpll_component.port_inclk0 = "PORT_USED",
        altpll_component.port_inclk1 = "PORT_UNUSED",
        altpll_component.port_locked = "PORT_USED",
        altpll_component.port_pfdena = "PORT_UNUSED",
        altpll_component.port_phasecounterselect = "PORT_UNUSED",
        altpll_component.port_phasedone = "PORT_UNUSED",
        altpll_component.port_phasestep = "PORT_UNUSED",
        altpll_component.port_phaseupdown = "PORT_UNUSED",
        altpll_component.port_pllena = "PORT_UNUSED",
        altpll_component.port_scanaclr = "PORT_UNUSED",
        altpll_component.port_scanclk = "PORT_UNUSED",
        altpll_component.port_scanclkena = "PORT_UNUSED",
        altpll_component.port_scandata = "PORT_UNUSED",
        altpll_component.port_scandataout = "PORT_UNUSED",
        altpll_component.port_scandone = "PORT_UNUSED",
        altpll_component.port_scanread = "PORT_UNUSED",
        altpll_component.port_scanwrite = "PORT_UNUSED",
        altpll_component.port_clk0 = "PORT_USED",
        altpll_component.port_clk1 = "PORT_UNUSED",
        altpll_component.port_clk2 = "PORT_UNUSED",
        altpll_component.port_clk3 = "PORT_UNUSED",
        altpll_component.port_clk4 = "PORT_UNUSED",
        altpll_component.port_clk5 = "PORT_UNUSED",
        altpll_component.port_clkena0 = "PORT_UNUSED",
        altpll_component.port_clkena1 = "PORT_UNUSED",
        altpll_component.port_clkena2 = "PORT_UNUSED",
        altpll_component.port_clkena3 = "PORT_UNUSED",
        altpll_component.port_clkena4 = "PORT_UNUSED",
        altpll_component.port_clkena5 = "PORT_UNUSED",
        altpll_component.port_extclk0 = "PORT_UNUSED",
        altpll_component.port_extclk1 = "PORT_UNUSED",
        altpll_component.port_extclk2 = "PORT_UNUSED",
        altpll_component.port_extclk3 = "PORT_UNUSED",
        altpll_component.valid_lock_multiplier = 1;
endmodule
```

Anexo 11: Código de software de recepción para el host en Visual Basic.

```
Option Explicit
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Dim Puerto
Dim lngLocal As Long
```

```
Dim blnEstado As Boolean
Dim lngX As Long
Dim i
Dim buffer_input As String
Dim datos_actuales As String
Public fs As FileSystemObject
Public ts As TextStream
Dim fso As New FileSystemObject
Dim fsoStream As TextStream
Private blnConnected As Boolean

Private Sub cmdReset_Click()
    dato_hex.Caption = ""
    dato_dec.Caption = ""
    dato_bin.Caption = ""
    log.Text = ""

End Sub

Private Sub Command1_Click()
    recibido.Caption = "RECIBIDO : " + MSComm1.Input

End Sub

Private Sub cliente4espera_Timer()

End Sub

Private Sub Command2_Click()
    MSComm1.Output = "+++"
    recibido.Caption = "ENVIADO +++"
End Sub

Private Sub Command3_Click()
    MSComm1.Output = Trim(Text1.Text) + Chr$(13)
    recibido.Caption = "ENVIADO " + Trim(Text1.Text) + " ENTER"

End Sub


Private Sub Command6_Click()



End Sub

Private Sub Command7_Click()


End Sub

Private Sub Command4_Click()
    Timer3.Enabled = True
End Sub

Private Sub Command5_Click()
    Timer3.Enabled = False
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    '49 = 1
    '50 = 2
    '51 = 3
    '52 = 4
    '80 = p

    If KeyCode = 49 Then Call cliente1_Click
    If KeyCode = 50 Then Call cliente2_Click
    If KeyCode = 51 Then Call cliente3_Click
    If KeyCode = 52 Then Call cliente4_Click

    If KeyCode = 80 Then Call mensajep_Click

    If KeyCode = 27 Then recibido.Caption = ""

End Sub

Sub apagarleds()
    led(1).Visible = False
    led(2).Visible = False
    led(3).Visible = False
    led(4).Visible = False
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    fsoStream.Close
End Sub

Private Sub log_Change()
    Dim dato_decimal As Double

    If led(3).Visible = False Then
        led(3).Visible = True
    Else
        led(3).Visible = False
    End If
    ' Escritura dato en hexadecimal
    dato_hex.Caption = Trim(log.Text)
    dato_hex.Caption = Left(dato_hex.Caption, 4)

    If Len(Trim(dato_hex.Caption)) = 4 Then
        ' Escritura dato en binario
        dato_bin.Caption = ""
        dato_bin.Caption = dato_bin.Caption + hex_a_bin(Right(Left(dato_hex.Caption, 1), 1))
        dato_bin.Caption = dato_bin.Caption + hex_a_bin(Right(Left(dato_hex.Caption, 2), 1))
        dato_bin.Caption = dato_bin.Caption + hex_a_bin(Right(Left(dato_hex.Caption, 3), 1))
        dato_bin.Caption = dato_bin.Caption + hex_a_bin(Right(Left(dato_hex.Caption, 4), 1))

        dato_bin.Caption = Trim(dato_bin.Caption)
        'Escribe dato decimal
        dato_decimal = 0
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 16), 1)) * 1)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 15), 1)) * 2)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 14), 1)) * 4)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 13), 1)) * 8)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 12), 1)) * 16)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 11), 1)) * 32)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 10), 1)) * 64)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 9), 1)) * 128)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 8), 1)) * 256)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 7), 1)) * 512)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 6), 1)) * 1024)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 5), 1)) * 2048)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 4), 1)) * 4096)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 3), 1)) * 8192)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 2), 1)) * 16384)
        dato_decimal = dato_decimal + (Val(Right(Left(dato_bin.Caption, 1), 1)) * 32768)

        dato_dec.Caption = dato_decimal

        fsoStream.WriteLine Trim(dato_dec.Caption)

    End If
End Sub

Function hex_a_bin(hexa As String) As String

    If hexa = "0" Then
        hex_a_bin = "0000"
    End If
    If hexa = "1" Then
        hex_a_bin = "0001"
    End If
    If hexa = "2" Then
        hex_a_bin = "0010"
    End If
    If hexa = "3" Then
        hex_a_bin = "0011"
    End If
    If hexa = "4" Then
        hex_a_bin = "0100"
    End If
    If hexa = "5" Then
        hex_a_bin = "0101"
    End If
    If hexa = "6" Then
        hex_a_bin = "0110"
    End If
    If hexa = "7" Then
        hex_a_bin = "0111"
    End If
    If hexa = "8" Then
        hex_a_bin = "1000"
    End If
    If hexa = "9" Then
        hex_a_bin = "1001"
```

```
    End If
    If hexa = "A" Then
        hex_a_bin = "1010"
    End If
    If hexa = "B" Then
        hex_a_bin = "1011"
    End If
    If hexa = "C" Then
        hex_a_bin = "1100"
    End If
    If hexa = "D" Then
        hex_a_bin = "1101"
    End If
    If hexa = "E" Then
        hex_a_bin = "1110"
    End If
    If hexa = "F" Then
        hex_a_bin = "1111"
    End If

End Function

Private Sub MSComm1_OnComm()
    log.Text = MSComm1.Input

End Sub

Private Sub Form_Load()
    blnConnected = False
    MSComm1.CommPort = InputBox("Puerto?")
    MSComm1.PortOpen = True
    datos_actuales = "0000000000"
    Set fsoStream = fso.CreateTextFile("c:\salida.txt", True)
End Sub

Private Sub Timer2_Timer()
End Sub

Private Sub timerwaiting_Timer()
    waiting.Visible = False
    timerwaiting.Enabled = False
End Sub
```

Anexo 12: Códigos de módulos y núcleos para el funcionamiento del sistema de adquisición completo.

```
//////////////////////////////////////////////////////////////
//                          MODULO TOP
//////////////////////////////////////////////////////////////

module DE2_TOP
        ();

input                   CLOCK_27;
input                   CLOCK_50;
input                   EXT_CLOCK;
input   [3:0]   KEY;
input   [17:0]  SW;
output  [6:0]   HEX0;
output  [6:0]   HEX1;
output  [6:0]   HEX2;
output  [6:0]   HEX3;
output  [6:0]   HEX4;
output  [6:0]   HEX5;
output  [6:0]   HEX6;
output  [6:0]   HEX7;
output  [8:0]   LEDG;
output  [17:0]  LEDR;
output  UART_TXD;
input           UART_RXD;
output  IRDA_TXD;
input           IRDA_RXD;
inout   [15:0]  SRAM_DQ;
output  [17:0]  SRAM_ADDR;
output                  SRAM_UB_N;
output                  SRAM_LB_N;
output                  SRAM_WE_N;
output                  SRAM_CE_N;
```

```
output                  SRAM_OE_N;
inout                   AUD_ADCLRCK;
input                   AUD_ADCDAT;
inout                   AUD_DACLRCK;
output                  AUD_DACDAT;
inout                   AUD_BCLK;
output                  AUD_XCK;
inout   [35:0]  GPIO_0;
inout   [35:0]  GPIO_1;
assign  LCD_ON          =       1'b0;
assign  LCD_BLON=       1'b0;

// Todos los puertos en alta impedancia
assign  DRAM_DQ         =       16'hzzzz;
assign  FL_DQ           =       8'hzz;
assign  OTG_DATA=       16'hzzzz;
assign  LCD_DATA=       8'hzz;
assign  SD_DAT          =       1'bz;
assign  I2C_SDAT=       1'bz;
assign  ENET_DATA       =       16'hzzzz;
assign  AUD_ADCLRCK     =       1'bz;
assign  AUD_DACLRCK     =       1'bz;
assign  AUD_BCLK=       1'bz;
assign  GPIO_0          =       36'hzzzzzzzzz;
assign  GPIO_1          =       36'hzzzzzzzzz;

//////////////////////////////////////////
// Designacion Counter
reg [5:0] shortCount ;
//////////////////////////////////////////

////////////////////////////////////////////////////////////
///     Declaracion de variables de uso interno SRAM
////////////////////////////////////////////////////////////
reg falla;                                                      //Flag - Grabacion incorrecta SRAM
reg bien;                                                       //Flag - Grabacion correcta   SRAM
reg [17:0] direccion_memoria;     //Direccion memoria 18 bits       SRAM
reg escritura_memoria;                  //Flag - Modo escritura                 SRAM
reg [15:0] datos_memoria;         //Datos memoria 16 bits                 SRAM
wire [17:0] direccion_final;
reg [17:0] direccion_para_leer;
////////////////////////////////////////////////////////////
///     Declaracion de variables de uso interno CODEC
////////////////////////////////////////////////////////////
reg dacdat;
reg [15:0] periodos;
reg [15:0] muestra_actual;              //Registro Swaping
reg [15:0] muestra_anterior;      //Registro Swaping
reg [15:0] muestra_preparada;     //Registro Swaping
reg [15:0] muestra_led;                 //Bus de 16 bits con dato actual

reg flag_enviar_ahora;                  //Flag que indica que se enviara un nuevo paquete (este se resetea
                                                                                        //una vez concluido)
reg flag_listo_para_enviar;       //Flag que indica listo para enviar datos por UART
reg flag_key_presionado;          //Flag que indica si presiono el boton
wire reloj_salida;                              //Reloj de salida de PLL
wire clock_envio_datos;                 //Reloj de salida de divisor de frecuencia para
                                                                                        //frecuencia  de  entrega  de
datos

reg [1:0] etapa;                                //CONTADOR DE ETAPAS (MAQUINA DE ESTADOS PRINCIPAL)
reg [2:0] estado_envio;                 //Contador de estados de la secuencia de envio

wire daclrck;                                                   //Clock DAC Ventanas LR
wire clkdata;                                                   //Clock Datos CODEC

reg [6:0] seg7_1;                                               //Display 7-segmentos
reg [6:0] seg7_2;                                               //Display 7-segmentos
reg [6:0] seg7_3;                                               //Display 7-segmentos
reg [6:0] seg7_4;                                               //Display 7-segmentos


assign HEX7 = seg7_1;
assign HEX6 = seg7_2;
assign HEX5 = seg7_3;
assign HEX4 = seg7_4;

////////////////////////////////////////////////////////////
// BLOQUE DE INICIALIZACION
////////////////////////////////////////////////////////////
initial
        begin
```

```
                falla <= 1'b0;                                                                  //    Reset
flag falla de transmision
                bien <= 1'b0;                                                                   //    Reset
flag transmision correcta
                direccion_memoria <= 18'b0;                                     //  Reset  contador  direccion  de
memoria
                periodos <= 16'd100;
                dacdat <= 1'b0;
                muestra_actual <= 16'b0;                                        // Reset swaper muestreador
                muestra_anterior <= 16'b0;                                              // Reset swaper muestreador
                muestra_led <= 16        'b0000000000000000;
                escritura_memoria <= 1'b0;                                              //   Reset   de   flag   para
escritura
                seg7_1 <= 7'b1000111;                                                   // Escribe LoAd
                seg7_2 <= 7'b0100011;                                                   // Escribe LoAd
                seg7_3 <= 7'b0001000;                                                   // Escribe LoAd
                seg7_4 <= 7'b0100001;                                                   // Escribe LoAd
                estado_envio <= 3'b000;                                         // Reset contador de estados
                flag_listo_para_enviar <= 1'b0;                   //      Reset  de  flag  de  listo  para  envio
UART
                flag_key_presionado <= 1'b0;                                    // Reset de flag de boton presionado
                etapa <= 2'b00;
                direccion_para_leer <= 18'b000000000000000000;
                flag_enviar_ahora <= 1'b0;


        end
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////


assign LEDG[8] = flag_listo_para_enviar;

assign AUD_XCK = reloj_salida;
assign AUD_BCLK = dacbclk;
assign AUD_DACDAT = dacdat;
assign AUD_DACLRCK = daclrck;
assign AUD_ADCLRCK = daclrck;


/////////////////////////////////////////////////////
//              DEFINICION DE SALIDAS GPIO SENALES CODEC (Testpoints)
/////////////////////////////////////////////////////
/*
assign GPIO_1[5]=AUD_ADCDAT;
assign GPIO_1[4]=AUD_DACDAT;
assign GPIO_1[3]=clkdata;
assign GPIO_1[2]=AUD_BCLK;
assign GPIO_1[1]=AUD_DACLRCK;
assign GPIO_1[0]=AUD_XCK;
*/

assign GPIO_1[7:0]=TD_DATA;
assign GPIO_1[8]=TD_HS;
assign GPIO_1[9]=TD_VS;

assign TD_RESET = 1'b1;


/////////////////////////////////////////////////////
//              DEFINICION DE CLOCKS (CODEC, PLL, DATOS)
/////////////////////////////////////////////////////
PLLALP divisor(CLOCK_50, reloj_salida);
dacsgn senales(reloj_salida, daclrck, dacbclk,clkdata);
f_divisor clk_frecuencia_envio_datos(CLOCK_50,clock_envio_datos);
multiplexor_direccion address_mux(etapa, direccion_memoria, direccion_para_leer, direccion_final);

monostable_50mhz pulso_rs232(CLOCK_50, flag_enviar_ahora, sendNew);

muestra_digitos Digit0(HEX0, SRAM_DQ[3:0]);
muestra_digitos Digit1(HEX1, SRAM_DQ[7:4]);
muestra_digitos Digit2(HEX2, SRAM_DQ[11:8]);
muestra_digitos Digit3(HEX3, SRAM_DQ[15:12]);

/////////////////////////////////////////////////////
//      DEFINICION DE ETAPAS
/////////////////////////////////////////////////////
//
//              ETAPA 0 :               ADQUISICION Y ESCRITURA EN SRAM
//
//              ETAPA 1 :               ENVIO POR UART
//
//              ETAPA 2 :               STANDBY // ESPERANDO RE-MUESTRA
```

```
//
////////////////////////////////////////////////////

////////////////////////////////////////////////////
//      ETAPA DE ADQUISICION DE DATOS
////////////////////////////////////////////////////
always @(negedge clkdata)
begin
if (etapa == 2'b00)
begin
        if (shortCount == 6'd0)
                begin
                        dacdat <= 1'b1;
                        muestra_actual[15] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd2)
                begin
                        dacdat <= 1'b0;

                        muestra_actual[14] <= AUD_ADCDAT;
                end

        if (shortCount == 6'd4)
                begin
                        muestra_actual[13] <= AUD_ADCDAT;
                end

        if (shortCount == 6'd6)
                begin
                        muestra_actual[12] <= AUD_ADCDAT;
                end

        if (shortCount == 6'd8)
                begin
                        muestra_actual[11] <= AUD_ADCDAT;
                end

        if (shortCount == 6'd10)
                begin
                        muestra_actual[10] <= AUD_ADCDAT;
                end

        if (shortCount == 6'd12)
                begin
                        muestra_actual[9] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd14)
                begin
                        muestra_actual[8] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd16)
                begin
                        muestra_actual[7] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd18)
                begin
                        muestra_actual[6] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd20)
                begin
                        muestra_actual[5] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd22)
                begin
                        muestra_actual[4] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd24)
                begin
                        muestra_actual[3] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd26)
                begin
                        muestra_actual[2] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd28)
                begin
                        muestra_actual[1] <= AUD_ADCDAT;
                end
        if (shortCount == 6'd30)
                begin
                        muestra_actual[0] <= AUD_ADCDAT;
                end
```

```
        if (shortCount == 6'd32)
                begin
                        muestra_anterior <= muestra_led;
                end

        // Calcula el valor del diferencial a sumar
        if (shortCount == 6'd33)
                begin
                        if (muestra_actual[15] == 1'b0)
                                begin
                                        muestra_preparada <= muestra_actual[14:0];
                                end

                        if (muestra_actual[15] == 1'b1)
                                begin
                                        muestra_preparada          <=          ((16'b111111111111111          -
muestra_actual[14:0])+1'b1);
                                end

        end

        // Actualiza el cambio e incrementa (decrementa) la cuenta integrada
        if (shortCount == 6'd36)
                begin
                        if (muestra_actual[15] == 1'b0)
                                begin
                                        muestra_led <= (muestra_anterior + muestra_preparada[15:0]);
                                end

                        if (muestra_actual[15] == 1'b1)
                                begin
                                        muestra_led <= (muestra_anterior - muestra_preparada[15:0]);
                                end
                end
        // Setting Datos y Direccion de Memoria en SRAM
        if (shortCount == 6'd37)
                begin
                        datos_memoria <= muestra_led;
                end
        // Setting flag de escritura de memoria (escritura activada)
        if (shortCount == 6'd38)
                begin
                        escritura_memoria <= 1'b1;
                end
        // Setting flag de escritura de memoria (escritura desactivada)
        if (shortCount == 6'd39)
                begin
                        escritura_memoria <= 1'b0;
                end
        // Incrementa la direccion de memoria
        if (shortCount == 6'd40)
                begin
                        if (direccion_memoria == 18'b111111111111111111)
                                begin
                                        seg7_1 <= 7'b0001100;
                // Escribe Push
                                        seg7_2 <= 7'b1100011;
                // Escribe Push
                                        seg7_3 <= 7'b0010010;
                                        seg7_4 <= 7'b0001011;
                                        flag_listo_para_enviar <= 1'b1;                                 //
Set Flag Listo para enviar
                                        etapa <= 2'b01;
                        // Pasa a la etapa de envio
                                        direccion_memoria <= 18'b000000000000000000; // Reset a la posicion
en memoria
                                        end
                        else
                                begin
                                        direccion_memoria <= direccion_memoria +1;
                                end
                end



        shortCount <= shortCount + 1;

end
end


assign LEDG[7] = estado_envio[0];
```

```
////////////////////////////////////////////////////
//      ETAPA DE ESPERA Y ENTREGA DE DATOS POR UART
////////////////////////////////////////////////////
always @(posedge clock_envio_datos)
begin
        if (etapa == 2'b01)
        begin
                if (flag_listo_para_enviar)
                begin
                        if (flag_key_presionado)
                        begin
                                if (estado_envio == 4'd0)
                                begin
                                        // Resetea la senal de disparo de envio de paquete
                                        flag_enviar_ahora <= 1'b0;
                                end
                                if (estado_envio == 4'd1)
                                begin
                                        // Levanta el flag de envio por UART
                                        flag_enviar_ahora <= 1'b1;
                                end
                                if (estado_envio == 4'd2)
                                begin
                                        // Estabilizacion de seguridad del sistema
                                end
                                if (estado_envio == 4'd3)
                                begin
                                        // Aumenta en uno el contador de la direccion de memoria
                                         direccion_para_leer <= direccion_para_leer + 1;
                                end


                                //seg7_1 <= 7'b0010010;                                         //  Escribe
SEnd
                                //seg7_2 <= 7'b0000110;
                                //seg7_3 <= 7'b0101011;
                                //seg7_4 <= 7'b0100001;
                                estado_envio <= estado_envio + 1;
                        end
                end
        end
end


always @(negedge KEY[3])
begin
        flag_key_presionado <= 1'b1;
end
////////////////////////////////////////////////////////////////////////
///     ASIGNACION DE VARIABLES A LEDS Y 7-SEGMENTOS
////////////////////////////////////////////////////////////////////////

// Muestra ciclos de la maquina de estados principal
assign LEDG[5:0] = shortCount;

// Muestra direccion de memoria actual en LEDR[17:0]
assign LEDR[17:0] = direccion_final;

////////////////////////////////////////////////////////////////////////



assign SRAM_ADDR = direccion_final;
assign SRAM_UB_N = 0;                                   // Hi-Byte
assign SRAM_LB_N = 0;                                   // Lo-Byte
assign SRAM_CE_N = 0;                                   // Enable
assign SRAM_WE_N = !escritura_memoria;    // Bit de escritura en memoria (NEGADO)
assign SSRAM_OE_N = 0;                                  // Output enable
// Cambia entre modo escritura y lectura (alta impedancia)
assign SRAM_DQ = ((!escritura_memoria)? 16'hzzzz : datos_memoria);



    ////////////////////////////
    // reset define
    wire reset;
    assign reset = ~KEY[0] ;

    wire send_done ;
    send_16_hex sender(
            .number_to_send_in(SRAM_DQ),
            .send_strobe_in(sendNew),
            .send_done_out(send_done),
```

```
            .uart_transmit_Out(UART_TXD),
            .CLOCK_50_in(CLOCK_50),
            .reset_in(reset) );

   reg [15:0] data ;
   wire sendNew ;

   parameter data_rate = 1 ;
   parameter data_rate_set =  50000000/data_rate ;
         reg [31:0] data_rate_counter ;

   assign GPIO_1[27] = UART_TXD;
endmodule

module dacsgn(entrada, lrck, bclk, clockdatos);
         input entrada    ;               //the hex digit to be displayed
         output lrck ;            //actual LED segments
         output bclk;
         output clockdatos;
         reg internal_lrck;
         reg internal_bclk;
         reg internal_clockdatos;
         reg [31:0] contador ;

         always @(negedge entrada)
         begin
                  internal_lrck <= contador[7];
                  internal_bclk <= contador[2];
                  internal_clockdatos <= contador[1];
                  contador <= contador + 1;
         end
         assign lrck = internal_lrck;
         assign bclk = internal_bclk;
         assign clockdatos = internal_clockdatos;
endmodule

////////////////////////////////////////////////
/////  Modulo de multiplexion de SRAM_DQ
////////////////////////////////////////////////

module multiplexor_direccion(etapa_sistema, direccion_grabacion, direccion_lectura, direccion_salida);
         input [17:0] direccion_grabacion;         // BUS que indica la direccion de grabacion
         input [17:0] direccion_lectura;                   // BUS que indica la direccion de lectura
         input [1:0] etapa_sistema;                                        // Indica la etapa y por lo tanto el
direccionador
                  // del multiplexor
         output reg [17:0] direccion_salida ;                                          //   BUS   de   salida   del
multiplexor

         always @(direccion_grabacion or direccion_lectura or etapa_sistema)
         begin
                  if (etapa_sistema == 2'b00)
                           begin
                                    direccion_salida = direccion_grabacion;
                           end
                  if (etapa_sistema == 2'b01)
                           begin
                                    direccion_salida = direccion_lectura;
                           end
         end

endmodule

////////////////////////////////////////////
// Decode one hex digit for LED 7-seg display
module muestra_digitos(segs, num);
    input [3:0] num ;       //the hex digit to be displayed
    output [6:0] segs ;     //actual LED segments
    reg [6:0] segs ;
    always @ (num)
    begin
        case (num)
                4'h0: segs = 7'b1000000;
                4'h1: segs = 7'b1111001;
                4'h2: segs = 7'b0100100;
                4'h3: segs = 7'b0110000;
                4'h4: segs = 7'b0011001;
                4'h5: segs = 7'b0010010;
                4'h6: segs = 7'b0000010;
                4'h7: segs = 7'b1111000;
                4'h8: segs = 7'b0000000;
                4'h9: segs = 7'b0010000;
                4'ha: segs = 7'b0001000;
```

```
                4'hb: segs = 7'b0000011;
                4'hc: segs = 7'b1000110;
                4'hd: segs = 7'b0100001;
                4'he: segs = 7'b0000110;
                4'hf: segs = 7'b0001110;
                default segs = 7'b1111111;
        endcase
    end
endmodule
////////////////////////////////////////////

module monostable_50mhz(clk50_in, enviar_ahora_in, send_data_out);
   input clk50_in;
   input enviar_ahora_in;
   output reg send_data_out;
   reg estado_anterior;
         reg estado_actual;
         reg [1:0] etapa;
         initial
                 begin
                         estado_anterior <= 1'b1;
                         send_data_out <= 1'b0;
                 end
                 always @ (posedge clk50_in)
   begin
                 if ((enviar_ahora_in == 1'b1) &&(estado_anterior == 1'b0))
                         begin
                                 send_data_out <= 1'b1;
                                 estado_anterior <= enviar_ahora_in;
                         end
                 else
                         begin
                                 send_data_out <= 1'b0;
                                 estado_anterior <= enviar_ahora_in;
                         end
   end
endmodule
```