PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

ESTUDIO, DISEÑO E IMPLEMENTACIÓN DE UN DRIVER DE RELOJ PARA CCDS UTILIZANDO LA FUENTE DE CORRIENTE DE HOWLAND MEJORADA

BRAULIO JAVIER CANCINO VERA

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Profesor Supervisor:
ÁNGEL ABUSLEME
.....

Santiago de Chile, Enero 2016

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

ESTUDIO, DISEÑO E IMPLEMENTACIÓN DE UN DRIVER DE RELOJ PARA CCDS UTILIZANDO LA FUENTE DE CORRIENTE DE HOWLAND MEJORADA

BRAULIO JAVIER CANCINO VERA

Tesis presentada a la Comisión integrada por los profesores:
ÁNGEL ABUSLEME
DANI GUZMÁN
NICOLÁS REYES
FRANCO PEDRESCHI

Para completar las exigencias del grado de
Magíster en Ciencias de la Ingeniería

Santiago de Chile, Enero 2016

Ciencia y arte

forman parte de un todo.

MARTIN KEMP

AGRADECIMIENTOS

Más que agradecer quiero dedicar esta tesis a todas las personas que me apoyaron durante estos 8 años de universidad. Entre aquellas personas que me acompañaron se encuentran mi familia, la familia Cancino Vera que siempre se preocuparon y apoyaron en todo momento, mis familiares paternos y maternos, los cuales me acogieron durante mi estancia en la Universidad, y la familia Rojas Fuentes por llegar a convertirse en una segunda familia. Finalmente, quiero agradecer a todos mis amigos de colegio, barrio y universidad sin lo cuales todo esto no hubiese sido posible.

Gracias a todos ustedes por ayudarme a finalizar este ciclo y por los buenos momentos que compartimos.

ÍNDICE GENERAL

AGRADECIMIENTOS IV

ÍNDICE GENERAL V

ÍNDICE DE FIGURAS VIII

ÍNDICE DE TABLAS XI

RESUMEN XII

ABSTRACT XIII

1. Introducción 1
   1.1 Contexto 1
   1.2 Motivación y Problema de Tesis 2
   1.3 Objetivos y Metodología 3
   1.4 Organización del Documento 3

2. Transferencia de Carga en Detectores CCD 5
   2.1 Desempeño en el Proceso de Transferencia de Carga 5
      2.1.1 Eficiencia de Transferencia de Carga 7
      2.1.2 Generación de Cargas Espurias 14
      2.1.3 Blooming 15
   2.2 Driver de Reloj para Detectores CCDs Controlado por Corriente 16
      2.2.1 Soluciones Actuales 17
      2.2.2 Solución Propuesta 18
      2.2.3 Análisis Comparativo 19

3. Fuente de Corriente de Howland Mejorada 27
   3.1 Análisis Ideal 28
   3.2 Análisis de Primer Orden 29
   3.3 Análisis de Estabilidad 31

3.3.1 Respuesta Frente a Carga RC . . . 33
3.3.2 Respuesta Frente a Carga RC con Inductancias Parásitas . . . 35
3.3.3 Compensación Frente a Carga RC con Inductancias Parásitas . . . 38
3.4 Análisis de Ruido . . . 39
3.4.1 Análisis de Ruido Frente a una Carga RC . . . 41

4. Implementación . . . 44
4.1 Especificaciones . . . 44
4.1.1 Especificaciones del CCD . . . 44
4.1.2 Especificaciones Circuitales . . . 47
4.2 Diseño Circuital . . . 49
4.2.1 Máquina de Estados Principal . . . 49
4.2.2 Conversores Digital - Análogos (DAC) . . . 50
4.2.3 Amplificadores de Transimpedancia (TIA) . . . 50
4.2.4 Amplificadores de Transconductancia (TCA) . . . 53
4.2.5 Compensación . . . 56
4.2.6 Simulación . . . 57
4.3 Hardware . . . 64
4.3.1 PCB . . . 64
4.3.2 Microcontrolador . . . 69
4.4 Software . . . 71
4.4.1 FPGA . . . 71
4.4.2 Microcontrolador . . . 81
4.4.3 PC . . . 82

5. Resultados Experimentales . . . 84
5.1 Configuración Experimental . . . 84
5.2 Resultados . . . 86
5.2.1 Transconductancias Efectivas Estáticas . . . 86
5.2.2 Señales de Entrada . . . 87

5.2.3 Múltiples Pendientes de Carga . . . 88
5.2.4 Slew Rate Máximo . . . 90
5.2.5 Frecuencia Máxima . . . 92
5.2.6 Distorsión Cerca de los Rieles . . . 94
5.2.7 Respuesta Frente a Inductancia Parásita . . . 94

6. Conclusiones y Trabajo Futuro . . . 96
6.1 Discusión y Resultados . . . 96
6.2 Contribuciones Principales . . . 98
6.3 Trabajo Futuro . . . 99

BIBLIOGRAFÍA . . . 100

ANEXOS . . . 104

A. Especificaciones CCDs Comerciales. . . . 105

B. Códigos. . . . 107
B.1 Módulos Verilog FPGA . . . 107
B.2 Código C Microcontrolador . . . 139

ÍNDICE DE FIGURAS

2.1. Diagrama simplificado proceso de transferencia de carga en un pixel de Tres Fases 6

2.2. Señal de voltaje de una fase típica utilizada en el proceso de transferencia de carga. 7

2.3. Esquema de vaciado de carga del CCD luego de la captura de un flat-field para medir la CTE serial. 10

2.4. Esquema de lectura del CCD luego del vaciado de carga para medir la CTE serial. 10

2.5. Esquema que representa la lectura de un flat-field para la obtención de la CTE mediante el método EPER. 12

2.6. Esquema simplificado de un CCD clock driver de tres fases. 16

2.7. Diagrama típico de un driver de reloj controlado por voltaje. 17

2.8. Diagrama de la segunda arquitecutra de control por voltaje. 18

2.9. Diagrama del driver de reloj controlado por corriente propuesto. 19

2.10. Diagrama circuital simplificado arquitecturas de control por voltaje y corriente. 20

2.11. Establecimiento normalizado comparativo entre las arquitecturas de control por corriente y voltaje. 22

2.12. Cifra de mérito Λ_e para las arquitecturas de control por voltaje y corriente en función del número de constantes de tiempo n. 25

3.1. Diagrama Circuital de la IHCS. 27

3.2. Diagrama Circuital de la IHCS. 28

3.3. Diagrama de Bloques Modelo Ganancia Asintótica. 32

3.4. $T(s)$ frente a una carga RC serie. 35

3.5. $T(s)$ frente a una carga RC serie con inductancia parásita. 37

3.6. $T(s)$ frente a una carga RC serie con inductancia parásita y compensación. . 38
3.7. Equivalente circuital de la IHCS para el análisis de ruido. 39
3.8. Densidad espectral de corriente a la salida de la IHCS considerando una Carga RC serie. 42

4.1. Diagrama general de una señal de reloj. 45
4.2. Diagrama general sistema diseñado. 49
4.3. Diagrama circuital amplificador de transimpedancia. 51
4.4. Diagrama circuital amplificador de transconductancia. 54
4.5. Diagrama circuital etapa de compensación. 56
4.6. Formas de onda a la entrada y a la salida del amplificador de transimpedancia. 58
4.7. Formas de onda a la entrada y a la salida del amplificador de transconductancia de imagen frente a una carga resistivo-capacitiva serie de $0.1\,\Omega$ y $160\,\mathrm{nF}$. . . 59
4.8. Formas de onda a la entrada y a la salida del amplificador de transconductancia de registro frente a una carga resistivo-capacitiva serie de $0.1\,\Omega$ y $1\,\mathrm{nF}$. 60
4.9. Respuesta al escalón del amplificador de transconductancia de imagen utilizando una carga resistiva de $1\,\Omega$. 61
4.10. Respuesta al escalón del amplificador de transconductancia de registro utilizando una carga resistiva de $1\,\Omega$. 62
4.11. Respuesta del amplificador de registro frente a una carga serie inductivo-capacitivo-resistiva de $10\,\mu\mathrm{H}$, $1\,\mathrm{nF}$ y $0.1\,\Omega$. 63
4.12. Diagrama general hardware. 64
4.13. Fotografía PCB implementada. 65
4.14. Secciones circuitales de la PCB implementada. 65
4.15. Diagrama circuital bloque de alimentación. 66
4.16. Diagrama circuital bloque digital. 67
4.17. Diagrama circuital bloque digital. 68

4.18. Tarjeta de desarrollo “MSP430F5529 LaunchPad”. 70
4.19. Diagrama módulo principal FPGA. 71
4.20. Diagrama funcional módulo “RegSPI”. 72
4.21. Diagrama Funcional Módulos DAC. 73
4.22. Diagrama funcional del módulo Config. 75
4.23. Diagrama funcional del módulo “Control”. 80
4.24. Diagrama de periféricos del microcontrolador “MSP430F5529 LaunchPad”. . 82
4.25. Interfaz gráfica de usuario diseñada en Matlab. 83

5.1. Esquema de la configuración experimental utilizada. 85
5.2. Señales de entrada diferenciales provenientes del amplificador de transimpedancia. 87
5.3. Formas de onda a la salida del amplificador de imagen y de registro. 88
5.4. Ajuste lineal y cifra de mérito Λ_e propuesta en (2.18) del tiempo de bajada de la figura 5.3 en su sección de imagen. 89
5.5. Formas de onda a la salida del amplificador de imagen y de registro a máxima corriente se salida. 91
5.6. Formas de onda a la salida del amplificador de imagen y de registro a $500\,\mathrm{kHz}$ para imagen y $1\,\mathrm{MHz}$ para registro utilizando una carga capacitiva de $1\,\mathrm{nF}$. . 93
5.7. Salida del amplificador de registro utilizando una carga inductivo-capacitiva serie de $10\,\mu\mathrm{H}$ y $1\,\mathrm{nF}$. 95

ÍNDICE DE TABLAS

3.1. Parámetros Análisis Estabilidad RC. 34

4.1. Rangos de especificaciones señales de reloj fase de imagen. 46

4.2. Rangos de especificaciones señales de reloj fase de registro. 46

4.3. Especificaciones derivadas señales de reloj fase de imagen. 46

4.4. Especificaciones derivadas señales de reloj fase de registro. 46

4.5. Especificaciones circuitales generales. 47

4.6. Componentes amplificador de transimpedancia. 53

4.7. Características amplificadores operacionales etapa de transconductancia. . . . 55

4.8. Descripción de componentes etapa de transconductancia. 56

4.9. Descripción de entradas y salidas de los módulos DAC. 74

4.10. Descripción de las entradas del módulo Config. 76

4.11. Descripción de las salidas de los módulos Config. 76

4.12. Descripción de las salidas del módulo Config. 77

4.13. Intrucciones externas del módulo Config. 78

4.14. Descripción de las entradas y salidas del módulo Control. 81

5.1. Transconductancias efectivas estáticas obtenidas de los amplificadores de transconductancia. 86

A.1. Datos de CCD Comerciales de e2v. 106

RESUMEN

Los detectores CCD son dispositivos ampliamente utilizados en la astronomía que cumplen la función de generar carga eléctrica medible a partir de fotones. El proceso de lectura de los CCDs implica una etapa de transferencia de carga, la cual traslada la carga recolectada en cada pixel hacia los amplificadores de salida. Este proceso se realiza mediante la variación del voltaje aplicado a los electrodos de cada pixel del detector. Parámetros de las señales de lectura tales como la excursión de voltaje, tiempos de subida/bajada y tasa de subida/bajada, se relacionan en forma directa con el desempeño del proceso de transferencia de carga.

Si consideramos la naturaleza capacitiva de los pixeles del CCD, los drivers de generación de señales de lectura existentes no son eficaces, debido a que el control de la forma de onda se realiza mediante un amplificador de voltaje. Esta arquitectura de control no permite establecer con precisión la tasa de subida/bajada de la señal de voltaje, ya que su establecimiento siempre respetará la respuesta dinámica del amplificador.

Este trabajo estudia y propone el uso de la fuente de corriente de Howland mejorada para generar las señales de reloj para la lectura de los CCDs. Esta idea aprovecha la característica capacitiva de los pixeles del CCD, lo que permite establecer con precisión la tasa de subida/bajada de las señales de lectura, y en consecuencia, mejorar el desempeño del proceso de transferencia de carga.

Palabras Claves: Detectores, CCD, transferencia de carga, eficiencia, fuente de corriente de Howland mejorada, astronomía.

ABSTRACT

CCD detectors are devices widely used in astronomy and their function is to generate measurable electrical charge from incident photons. The CCD reading process involves a charge transfer stage, in which the charge collected in each pixel is transferred to the output amplifiers. This process is performed by applying a set of clock waveforms to each pixel electrode of the detector. Parameters reading signals such as voltage excursion, rise/fall times and rise/fall rates, are related directly to the performance of the charge transfer process.

If we consider the capacitive nature of CCD pixels, the existing clock drivers are not effective, because the waveform control is performed by a voltage amplifier. This control architecture does not allow to accurately set the rise/fall rate of the voltage signal, since its settling always follows the dynamic response of the amplifier.

This paper examines and proposes the use of a current-based CCD clock driver implemented through the improved Howland current source (IHCS). This idea takes advantage of the capacitive characteristic of CCD pixels, and allows to accurately set the rise/fall rate of the reading signals, and in consequence, improve the performance of the charge transfer process. Finally, the effectiveness of the proposed current-controlled architecture is proved.

Keywords: Detectors, CCD, charge transfer, efficiency, improved Howland current source, astronomy

1. INTRODUCCIÓN

Este capítulo tiene por objetivo introducir al lector el contexto, la motivación y el problema que este documento de tesis busca solucionar. Además describe los obtetivos, la metodología y la organización de este trabajo.

1.1 Contexto

Desde hace ya varias décadas Chile se ha ido consolidando como el principal centro de observaciones astronómicas a nivel mundial. La riqueza de nuestros cielos y las condiciones geográficas hacen del norte de nuestro país una zona propicia para la inversión de capital internacional en astronomía. Esto se ha traducido en que los parques astronómicos más importantes del mundo se encuentren en nuestro territorio.

Por otro lado, nuestro país ha ido desarrollando a través del tiempo una necesidad de profesionales cada vez más especializados. En este contexto, los ingenieros y científicos formados en Chile juegan un rol fundamental en el desarrollo del sector minero y su presencia comienza a ser preponderante en la astronomía nacional.

A raíz de las necesidades existentes en nuestro país, nace hace ya hace algunos años el grupo de investigación *Integrated Circuits UC* "IC-UC", del Departamento de Ingeniería Eléctrica de la Pontificia Universidad Católica de Chile, dedicado a la investigación y desarrollo en electrónica. Este grupo de investigación, liderado por el profesor Ángel Abusleme se ha enfocado en el desarrollo en microelectrónica de conversores, microelectrónica para experimentos de física de partículas y en el último tiempo, a la investigación en electrónica para astronomía.

Con el fin de captar imágenes del universo, los científicos en los observatorios utilizan detectores de alto desempeño, en donde destaca la presencia de los CCDs o *Charge-Coupled Devices*. Para su operación, los CCDs necesitan de un acabado montaje y un robusto control electrónico para obtener el mayor de los desempeños posibles.

En la actualidad, la búsqueda por aumentar aún más el desempeño de los detectores CCD se ha enfocado en reducir el ruido en base a post-procesamiento [J. R. Janesick (2001); Liu, Lu, Ding, y Liu (2011); Stefanov y Murray (s.f.)] y no ha puesto el énfasis necesario en mejorar la calidad de los datos obtenidos a la salida. Éste es, por ejemplo, el caso de los detectores *Electron Multiplying CCD* (EMCCD), cuyo desempeño se encuentra limitado por la generación de cargas espurias durante el proceso de transferencia de carga.

1.2 Motivación y Problema de Tesis

Dado el contexto de progreso tecnológico que en nuestro país se ha ido suscitando en el último tiempo, la principal motivación de este documento es ser un real aporte al crecimiento y al desarrollo tecnológico de Chile. La motivación particular de este trabajo tiene relación con la investigación y desarrollo en tópicos relacionados con detectores en astronomía.

Tal y como se analizará con profundidad en capítulos posteriores, el proceso de transferencia de carga en la etapa de lectura de los CCDs no es ideal [J. R. Janesick (2001)]. Debido a múltiples factores físicos, la carga eléctrica acumulada durante el proceso de integración debe lidiar con problemas como la ineficiencia en el proceso de transferencia de carga, generación de cargas espurias, y blooming. La problemática que aborda este documento consiste en analizar y evaluar el desempeño del proceso de transferencia de carga si se utiliza una nueva topología de driver de lectura por corriente basado en la fuente de corriente de Howland mejorada, tomando en consideración la naturaleza capacitiva de las fases de lectura de los CCDs.

1.3 Objetivos y Metodología

El objetivo principal de este trabajo consiste en estudiar, analizar y validar las ventajas de efectuar el proceso de transferencia de carga de un detector CCD utilizando fuentes de corriente de Howland mejorada para generar las señales de lectura. El propósito secundario de esta tesis consiste en motivar e incentivar la investigación y desarrollo en electrónica en nuestro país.

La metodología utilizada en este documento consiste en una primera instancia, analizar los parámetros de CCDs disponibles en la industria y establecer los parámetros de diseño. Luego, se analiza en forma detallada el comportamiento de la fuente de corriente de Howland mejorada, proponiendo ecuaciones de diseño. En forma posterior, se diseña e implementa un prototipo funcional de driver de lectura de CCD basado en la fuente de corriente de Howland mejorada, utilizando herramientas de software EDA. Finalmente, se realizan pruebas experimentales y se analizan los resultados obtenidos.

1.4 Organización del Documento

El documento de tesis se organiza en variadas secciones y sub-secciones que dan cuenta de todo el proceso de estudio, diseño e implementación del prototipo de driver de reloj controlado por corriente basado en la fuente de corriente de Howland mejorada. A continuación se describe brevemente cada uno de los capítulos de la tesis.

- **Transferencia de Carga en Detectores CCD:** Este capitulo estudia el proceso de transferencia de carga en CCDs, describe las ineficiencias del proceso y métodos para medir desempeño de la etapa de lectura.
- **Fuente de Corriente de Howland Mejorada:** Este capítulo estudia la fuente de corriente de Howland mejorada y propone ecuaciones de diseño.
- **Implementación:** Este capítulo describe todo el proceso de diseño y prototipado del driver de reloj controlado por corriente. Este capítulo comprende desde las especificaciones hasta el diseño de software y hardware.

- **Resultados Experimentales:** Este capítulo describe y discute los resultados experimentales obtenidos con el fin de validar la topología de driver de reloj controlado por corriente.
- **Conclusiones y Trabajo Futuro:** Este capítulo final describe y discute todo el trabajo realizado, incluyendo los resultados obtenidos. Además, propone los pasos futuros a realizar.

2. TRANSFERENCIA DE CARGA EN DETECTORES CCD

Este capítulo tiene por objetivo describir y analizar el proceso de transferencia de carga en CCDs, y además, exponer acerca de los procesos físicos y los métodos experimentales que describen su desempeño.

2.1 Desempeño en el Proceso de Transferencia de Carga

La transferencia de carga corresponde al proceso por el cual la carga eléctrica generada en el proceso de integración (o proceso de recolección de carga) es transferida de un pixel a otro. El objetivo de este proceso es transferir la carga desde el pixel recolector de carga hasta un amplificador de salida con el fin de transformar la carga eléctrica almacenada en una señal de voltaje. Para más detalle acerca del proceso físico involucrado en la transferencia de carga acudir a Howell (2006); J. R. Janesick (2001); McLean (2008).

A diferencia del proceso de integración de carga, que corresponde al fenómeno físico en el cual el CCD a un potencial eléctrico constante en sus fases, genera carga eléctrica a partir de fotones incidentes, el proceso de transferencia de carga se realiza mediante la variación de los potenciales eléctricos aplicados a sus fases. La figura 2.1 muestra un esquema simplificado de la transferencia de carga de un pixel de tres fases.

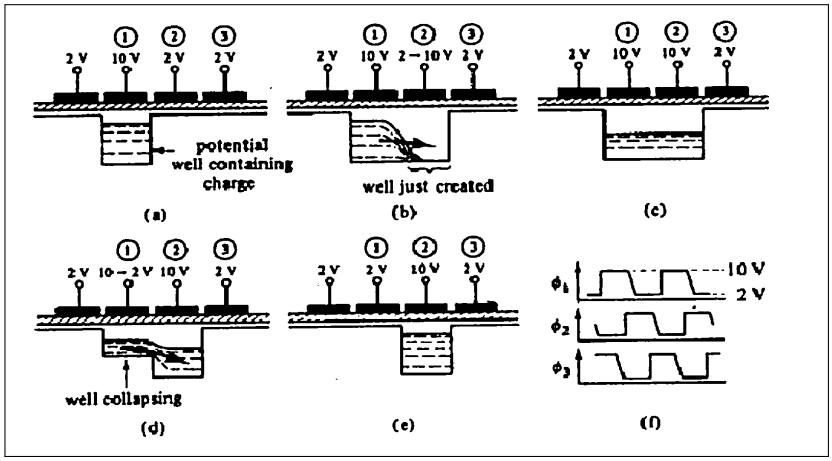


FIGURA 2.1. Diagrama simplificado proceso de transferencia de carga en un pixel de Tres Fases. Obtenido de The University of Oregon Physics Department web site (1997).

En la figura 2.1 se aprecian cinco figuras de transferencia de carga, (a) a la (e), y un gráfico de voltaje de fases (f). En (a), la carga eléctrica generada tras el proceso de integración se encuentra almacenada en un pozo de potencial bajo la fase uno. En (b) se muestra el transiente de movimiento de carga una vez que la fase dos sube su potencial. En (c) se muestra la carga ya repartida entre las fases uno y dos una vez que el transiente de transferencia de carga termina. En (d) se muestra el transiente de movimiento de carga una vez que la fase uno disminuye su potencial. Y por último en (e) se muestra la carga transferida de la fase uno a la dos.
Por otro lado, el la figura 2.1 (f) se muestran las formas de onda de potencial o voltaje de las fases uno, dos y tres en todo el proceso ilustrado de (a) a (e).

A continuación analizaremos en mayor detalle la relación entre forma de onda de voltaje aplicado a las fases y el desempeño del proceso de transferencia de carga.

En el desempeño del proceso de transferencia de carga, la forma de onda de voltaje aplicada a las fases juega un rol fundamental. La figura 2.2 muestra una forma de onda de voltaje típica aplicada a la fase de un pixel.

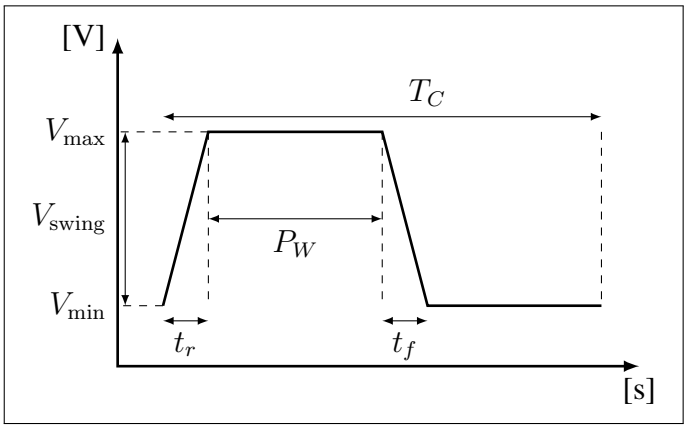


FIGURA 2.2. Señal de voltaje de una fase típica utilizada en el proceso de transferencia de carga.

En la figura 2.2, $V_{\max}$ y $V_{\min}$ corresponden a los voltajes extremos de la señal, V_{swing} y P_W corresponden a la amplitud de la señal y al ancho de pulso, respectivamente, t_r y t_f corresponden a los tiempos de subida y bajada, respectivamente, y T_C corresponde al periodo de la señal de reloj. Como veremos a continuación, estos parámetros poseen una relación directa con el desempeño de la transferencia de carga.

Para analizar el desempeño de la transferencia de carga, separaremos en tres el efecto de la forma de onda de voltaje: eficiencia de transferencia de carga (Charge Transfer Efficiency), generación de cargas espurias y blooming.

2.1.1 Eficiencia de Transferencia de Carga

La eficiencia de transferencia de carga (CTE) es una medida de desempeño fundamental en sistemas de imágenes basados en detectores CCD. Esta medida corresponde a

la eficiencia con la cual la carga de cada pixel (recolectada en el periodo de integración del proceso de lectura) es transferida de un pixel a otro.

Los detectores CCD poseen defectos que crean nuevos niveles entre las bandas de valencia y conducción. Estos nuevos niveles o trampas son capaces de atrapar electrones durante el proceso de transferencia de carga. El tiempo necesario para liberar estos portadores de carga atrapados y devolverlos a la banda de conducción para ser transferidos depende de factores como las características de las trampas (nivel energético, concentración y sección transversal de captura), temperatura de operación y frecuencia de lectura del CCD [Sopczak (2005)]. En particular, la CTE aumenta al disminuir la temperatura o al aumentar la frecuencia de lectura del CCD, y disminuye a un menor nivel de señal capturada.

Si la eficiencia de transferencia de carga entre dos pixeles está dada por η, entonces la ineficiencia de transferencia de carga ϵ estará dada por:

$$\epsilon = 1 - \eta \tag{2.1}$$

Un pixel que posea n electrones en promedio perderá ϵn electrones por cada transferencia de carga en el proceso de lectura. Luego de N transferencias de carga, la carga perdida total media μ_t y su fluctuación rms σ_t estarán dadas por [Holland (1990)]:

$$\mu_t = \epsilon n N \tag{2.2}$$

$$\sigma_t = \sqrt{\epsilon n N} \tag{2.3}$$

Las relaciones (2.2) y (2.3) nos indican que la carga total perdida por un pixel distribuye Poisson.

En astronomía el requerimiento mínimo de CTE es de $0{,}999995$ para ambas direcciones (CTE vertical y horizontal) [Christen, Kuijken, y Baade (2006)].

Continuación se expondrán y analizarán los métodos comúnmente utilizados para determinar o estimar la CTE.

a) Rayos X

El método de rayos X mediante la estimulación con Fe^{55} corresponde al método estándar más preciso para medir la CTE de un detector CCD [Howell (2006)]. Su principio de operación se basa en que un fotón de rayos X genera en promedio 1620 electrones en el silicio dentro de un diámetro menor a $1\,\mu\mathrm{m}$.

Luego de la estimulación de rayos X, se procede a la lectura del CCD, se establece con precisión la cantidad de carga perdida ΔQ, y se calcula la CTE como:

$$\mathrm{CTE} = 1 - \frac{\Delta Q}{NQ} \tag{2.4}$$

donde N corresponde a la cantidad de transferencias necesarias para la lectura del pixel y $Q \approx 1620\mathrm{e}^{-}$ es la carga generada por un fotón de rayos X [J. Janesick (1997)].

La desventaja de este método radica en el acceso al equipamiento necesario para efectuar estas pruebas, por lo que en algunos casos se prefieren métodos estadísticos por sobre este procedimiento.

b) First Pixel Response (FPR)

El método FPR (First Pixel Response) estima la CTE a partir de la carga perdida en un pixel mediante un esquema de reloj especial [Chiaberge, Riess, Mutchler, Sirianni, y Mack (2005)]. Este procedimiento consiste en obtener un flat-field del CCD y luego vaciar rápidamente la mitad del CCD mediante una señal de reloj especial. La siguiente figura nos muestra esta idea:

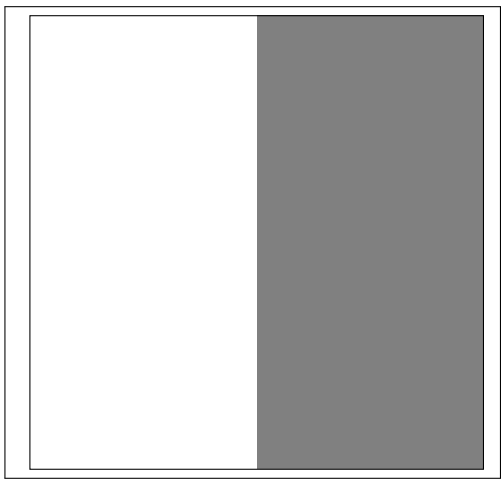


FIGURA 2.3. Esquema de vaciado de carga del CCD luego de la captura de un flat-field para medir la CTE serial. En gris se representa la imagen obtenida, mientras que en blanco se representa la carga vaciada.

Luego de vaciar la carga, se debe leer el CCD de tal manera que las columnas que no han sido vaciadas se transfieran al sector vaciado. Esto se muestra en la siguiente figura:

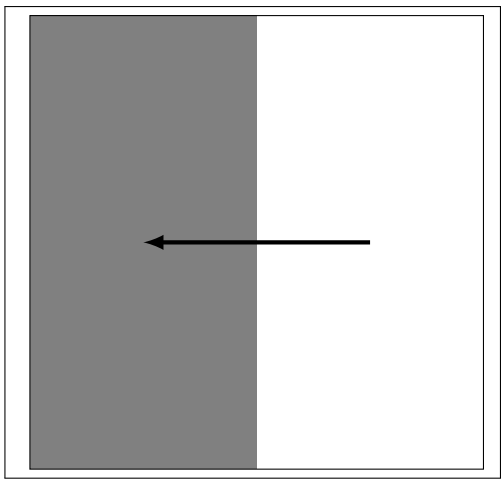


FIGURA 2.4. Esquema de lectura del CCD luego del vaciado de carga para medir la CTE serial. En gris se representa la carga de la imagen.

Finalmente se procede a la lectura de cada fila y se compara la carga del primer pixel (izquierda) con los demás pixeles de la fila.

El fundamento de este método se basa en que al vaciar inicialmente la carga del CCD luego de la captura del un flat-field se vacían todas las trampas de carga en esa zona. Luego al traspasar la carga de imagen, la carga del primer pixel queda atrapada en todas las trampas a medida que su carga se transifiere para ser leída, mientras que los demás pixeles casi no pierden carga. Si Q_f es la carga obtenida del primer pixel y Q es la carga promedio de los demás pixeles en la fila, entonces la CTE puede ser calculada como:

$$\mathrm{CTE} = \left(\frac{Q_f}{Q}\right)^{1/N} \tag{2.5}$$

donde N es la cantidad de pixeles transferidos de la primera carga o pixel.

c) Extended Pixel Edge Response (EPER)

EL método EPER (Extended Pixel Edge Response) corresponde a otro procedimiento de estimación de CTE a partir de una imagen. Éste consiste en tomar un flat-field de un CCD con regiones de overscan. La siguiente figura nos muestra un esquema de esta idea:

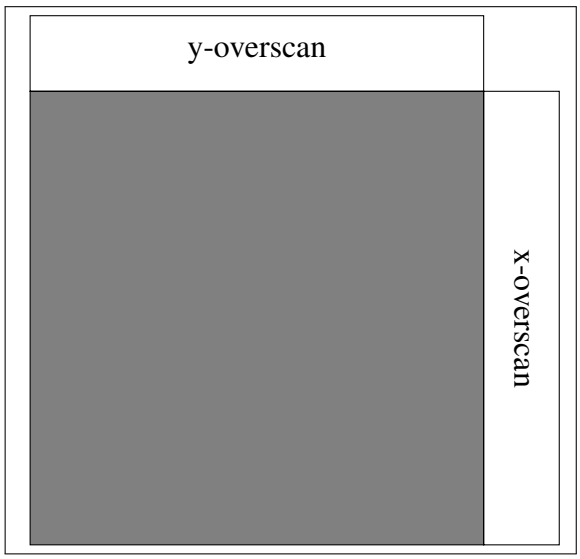


FIGURA 2.5. Esquema que representa la lectura de un flat-field para la obtención de la CTE mediante el método EPER. La zona gris representa la carga obtenida en el CCD debido a la captura del flat-field.

Finalmente, se lee todo el CCD incluyendo las zonas X e Y del overscan. La CTE resultante se calcula como [Jones, Clampin, Meurer, y Schrein (1999)]:

$$\mathrm{CTE} = \left(1 - \frac{\Delta Q}{Q_N}\right)^{1/N} \tag{2.6}$$

donde Q_N es la carga detectada en el último pixel de una fila (columna), ΔQ representa la carga por sobre el piso ruido oscuro de los primeros pixeles de la región de overscan y N es la cantidad de pixeles en la fila (columna) de la región sensible del CCD .

Antes de comenzar a leer el flat-field, todas las trampas de carga se encuentran en equilibrio y poseen una alta probabilidad de estar ocupadas por electrones. En el proceso de lectura, la condición de equilibrio se mantiene hasta que los pixeles del overscan sin carga alguna pasan a través de la zona de imagen sensible y por el registro de salida. Los pixeles del ovserscan solo contienen carga proveniente de carga oscura acumulada durante el proceso de lectura. La carga transferida a los pixeles de overscan mediante trampas de

pixeles de imagen poseen una baja probabilidad de ser re-capturados debido a que la carga se transfiere al siguiente pixel antes de ser atrapada por trampas. Por ello, la carga de las primeros pixeles de la zona de overscan representa la carga aplazada o cola debida a una mala transferencia de carga [Holland (1990)].

d) Change in Variance in Flat Field (CVF)

El método CVF (Change in Variance in Flat Field) corresponde a otro método de estimación de CTE basado en imágenes capturadas mediante un CCD. Tal y como su nombre lo expresa, este procedimiento mide el cambio en la varianza de los pixeles de una fila (columna) a medida que la carga es transferida y leída [Christen et al. (2006)].

Si se toma una flat-field normal, la varianza de la fila (columna) medida en ADU estará dada por:

$$\sigma_a^2(i) = \sigma_{a0}^2 - 2(1 - \mathrm{CTE})\sigma_{a0}^2 i + \mathcal{O}(\mathrm{CTI}) \tag{2.7}$$

donde σ_{a0}^2 representa la varianza del photon noise de la fila (columna) antes de cualquier transferencia de carga. El término $\mathcal{O}(\mathrm{CTI})$ es despreciable.

Para aplicar este método, se requieren dos flat-field F_k y dos bias B_k. El objetivo es eliminar la corriente oscura y el photon response non uniformity (PRNU) de la imagen. Luego creamos una imagen I_f dada por:

$$I_f = \frac{F_1 - B_2}{F_2 - B_2}\mathcal{M}(F_2 - B_2) \tag{2.8}$$

donde $\mathcal{M}()$ corresponde al valor promedio de la imagen. Luego, la varianza en una fila (columna) de I_f puede ser expresada como:

$$\sigma_{I_f}^2(i) = 2\left(\sigma_a^2(i) + \sigma_{\mathrm{ron}}^2\right) \tag{2.9}$$

donde σ_{ron}^2 corresponde al ruido de lectura (ADU). El ruido de lectura se calcula como:

$$\sigma_{\mathrm{ron}} = \frac{\mathcal{S}(B_1 - B_2)}{\sqrt{2}} \tag{2.10}$$

donde $\mathcal{S}()$ corresponde a la desviación estándar de la imagen. Finalmente, con un ajuste lineal es posible obtener la CTE utilizando la ecuación (2.7)

2.1.2 Generación de Cargas Espurias

Las cargas espurias son una fuente importante de carga indeseada en los detectores CCD. Cuando el CCD es polarizado en inversión, huecos de los channel stops migran y se recolectan bajo la compuerta [Dorn (2001); J. R. Janesick (2001)]. Algunos huecos quedan atrapados en la interfaz Si-SiO_2. Cuando la compuerta cambia su polarización a no inversión, los huecos atrapados en la interfaz Si-SiO_2 que poseen energía suficiente crean pares electrón-hueco cuando colisionan con los átomos de Si (ionización por impacto). Finalmente estos electrones o cargas espurias se recolectan en el pozo de potencial más cercano.

Algunas características relevantes de las cargas espurias son las siguientes:

- La carga espuria sólo se genera cuando el reloj pasa de inversión a no inversión (flanco de subida del reloj). Cuando el CCD pasa de no inversión a inversión (flanco de bajada del reloj) no se generan cargas espurias.
- Las cargas espurias disminuyen exponencialmente con al aumentar el tiempo de subida del reloj y aumenta exponencialmente con el aumento de la excursión de voltaje del reloj.
- La carga espuria aumenta cuando aumenta el ancho de pulso de la señal de reloj.
- La carga espuria aumenta cuando disminuye la temperatura de operación.
- La carga espuria aumenta linealmente con el número de transferencias de carga.
- Las cargas espurias distribuyen como ruido de disparo (Poisson). Esto quiere decir que la carga media generada es igual a su varianza en cada pixel.

Es posible medir la generación de cargas espurias mediante la obtención de una imagen bias. La señal medida en el bias posee principalmente componentes de corriente oscura

y de cargas espurias. Estas dos fuentes de ruido pueden ser fácilmente separadas. Por un lado, la señal proveniente de cargas espurias sólo depende de la señal de reloj, por lo que se pueden separar los componentes de ruido variando el tiempo de integración. Por otro lado, la dependencia de la corriente oscura con la temperatura es mucho más alta, por lo que se puede separar los componentes de ruido variando la temperatura de operación.

2.1.3 Blooming

El Blooming es otra fuente de ruido que depende de la señal de reloj del CCD. Por blooming se entiende cuando, debido a exceso de cargas o a una mal acondicionamiento de las señales del reloj, un porcentaje de carga del pixel se desborda y cae en los pixeles adyacentes.

El blooming es una de las causas que limitan la velocidad de lectura de los CCD. Cuando la señal de reloj cambia su polarización de no inversión a inversión demasiado rápido (tiempo de bajada del reloj), los electrones pueden adquirir energía suficiente como para traspasar la barrera de potencial que separa las regiones de recolección de carga y transferirse en el sentido opuesto de la dirección de transferencia de carga.

El tiempo de bajada necesario para que no ocurra blooming depende de varios factores: temperatura de operación, cantidad de carga transferida, movilidad de los portadores de carga, características del CCD, etc. Éste fenómeno no se produce en el flanco de subida del reloj, es decir, cuando la polarización del CCD pasa de inversión a no inversión [J. Janesick (1997)].

Una forma de medir el blooming consiste en tomar una imagen con una fuente lúminica puntual y calcular la cantidad de carga transferida a los pixeles adyacentes en función de parámetros de la señal de reloj, tales como tiempo de subida y excursión de voltaje.

2.2 Driver de Reloj para Detectores CCDs Controlado por Corriente

Un driver de reloj para CCDs es un circuito electrónico que cumple la función de generar las señales de voltaja durante el proceso de transferencia de carga. Un diagrama simplificado de un driver de reloj de tres fases se muestra en la figura 2.6

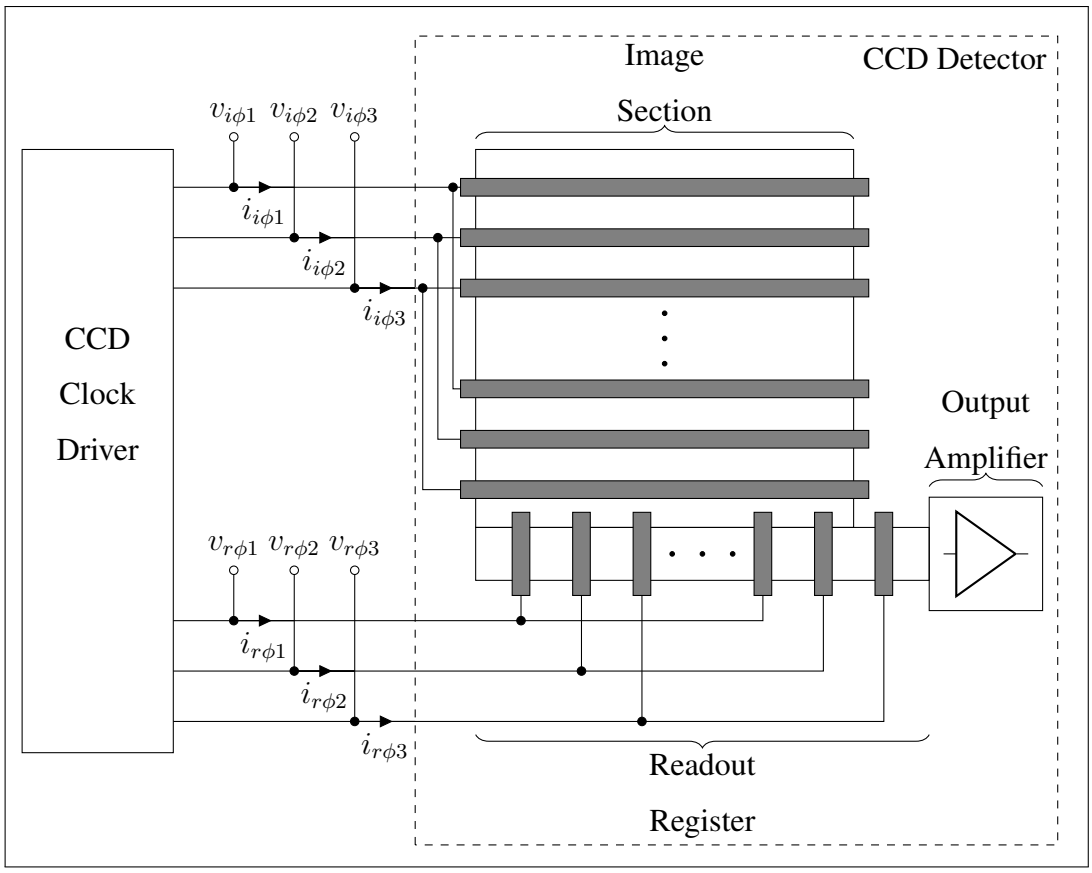


FIGURA 2.6. Esquema simplificado de un CCD clock driver de tres fases.

En la figura 2.6 $v_\phi(t)$ e $i_\phi(t)$ corresponden al voltaje y la corriente en la fase ϕ del CCD, respectivamente. Además, el CCD clock driver debe ser capaz de manejar una carga capacitiva [Dao, Etoh, y Le (2008)].

A continuación se presentará y analizará comparativamente la arquitectura de driver de reloj típicamente utilizado en la práctica y se propondrá la arquitectura de control por corriente.

2.2.1 Soluciones Actuales

En la actualidad, las señales de reloj son generadas utilizando una arquitectura de control por voltaje. La figura 2.7 nos muestra un diagrama de bloques típico de un driver de reloj.

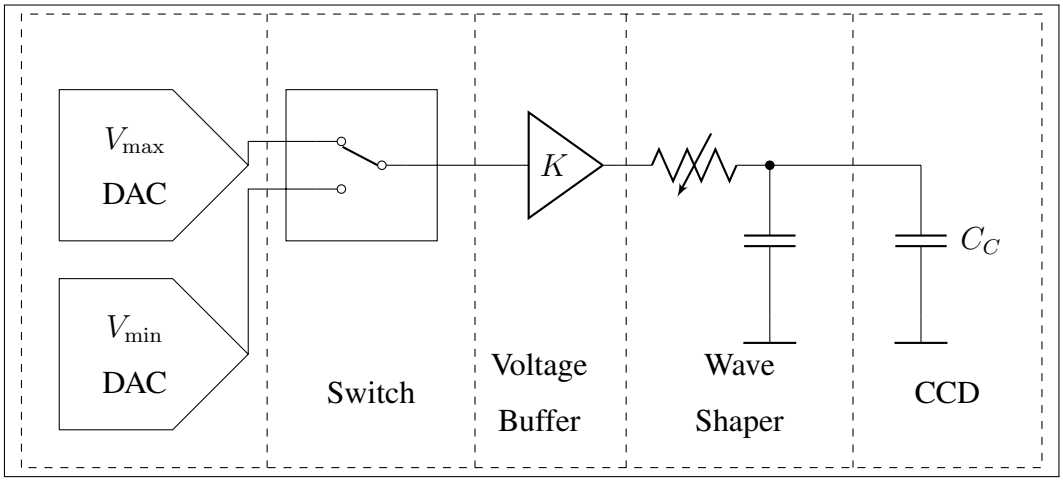


FIGURA 2.7. Diagrama típico de un driver de reloj controlado por voltaje.

En la figura 2.7 se observan cuatro bloques en la generación de la señal de lectura por voltaje. Al inicio, se definen dos niveles de voltaje utilizando un DAC de voltaje, $V_{\min}$ y $V_{\max}$, los cuales son seleccionados mediante un switch. Luego la señal de voltaje conformada a la salida del switch es amplificada en voltaje ($K \geq 1$) y es capaz entregar una corriente adecuada. Opcionalmente, con el fin de limitar la pendiente o el slew rate de la forma de onda en la fase [J. R. Janesick (2001); Wang y Li (2010)], sigue un bloque pasa bajos RC variable que conforma la señal final de voltaje en la fase del CCD. A la salida del bloque pasa bajos puede haber un seguidor de voltaje para evitar que la impedancia de

salida influya en la constante de tiempo del filtro RC.

Otra arquitectura de control por voltaje se muestra en la figura 2.8 [Daigle y Blais-Ouellette (2010)].

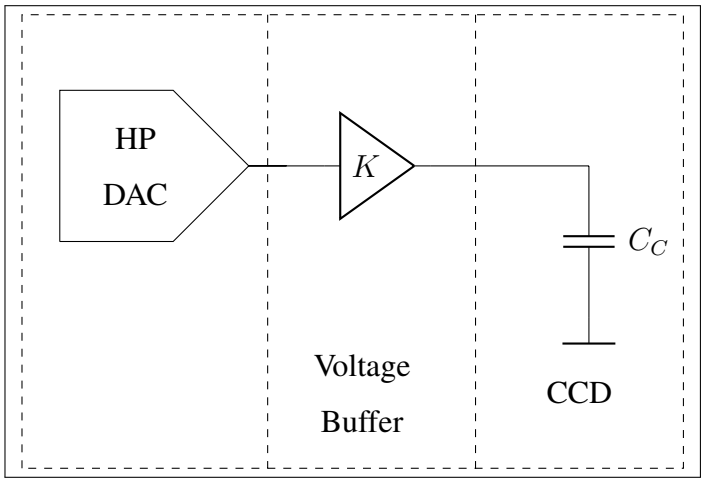


FIGURA 2.8. Diagrama de la segunda arquitecutra de control por voltaje.

Esta arquitectura de control por voltaje se basa en un DAC de voltaje de muy alto desempeño, seguido de un buffer de voltaje el cual se encarga de entregar la corriente necesaria a la carga. El DAC de voltaje de alto desempeño es capaz de establecer una respuesta de voltaje arbitraria en la carga.

2.2.2 Solución Propuesta

La solución propuesta consiste en generar las señales de reloj utilizando una arquitectura de control por corriente. La figura 2.9 nos muestra el diagrama de bloques simplificado de la arquitectura de control por corriente propuesta.

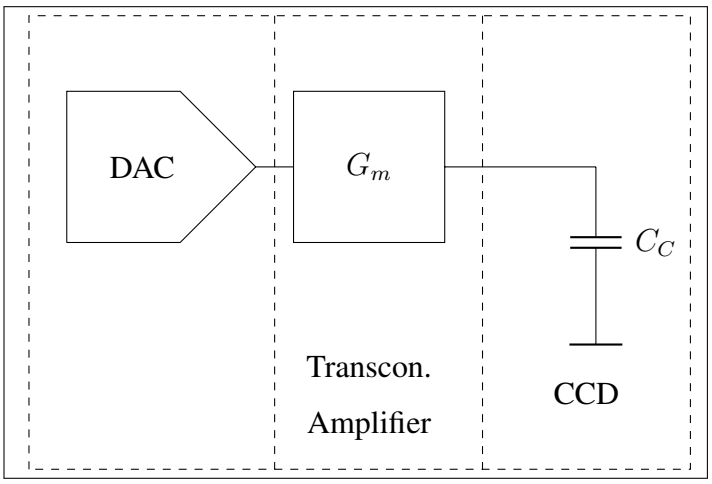


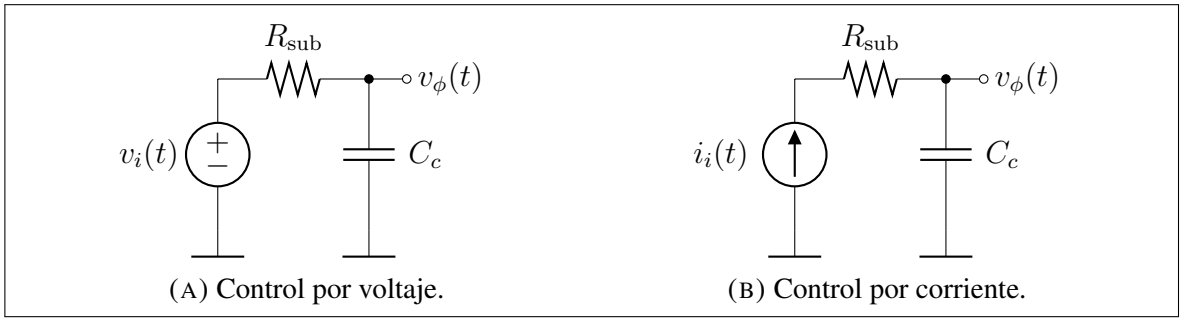
FIGURA 2.9. Diagrama del driver de reloj controlado por corriente propuesto.

En la figura 2.9 se aprecia sólo dos bloques en la generación de la señal de corriente. Éstos son el DAC de voltaje al inicio y la etapa de transconductancia que transforma la señal de voltaje en corriente.

A continuación se realizará un análisis simplificado comparativo entre las arquitecturas de control por voltaje y corriente.

2.2.3 Análisis Comparativo

Para realizar el análisis comparativo utilizaremos el modelo simplificado RC serie de la fase del CCD [Dao et al. (2008)] y fuentes ideales. Las figuras 2.10A y 2.10B muestran los diagramas circuitales simplificados de voltaje y corriente, respectivamente.



(A) Control por voltaje. (B) Control por corriente.

FIGURA 2.10. Diagrama circuital simplificado arquitecturas de control por voltaje y corriente.

En la figura 2.10A, $v_i(t)$, $v_\phi(t)$, R_{sub} y C_c representan el voltaje de entrada, el voltaje en la fase del CCD, el resistencia equivalente del sustrato y el capacitancia equivalente de la fase, respectivamente, mientras que en la figura 2.10B, $i_i(t)$, $v_\phi(t)$, R_{sub} y C_c representan la corriente de entrada, el voltaje en la fase del CCD, el resistencia equivalente del sustrato y el capacitancia equivalente de la fase, respectivamente

Si suponemos que las fuentes de voltaje son de la forma:

$$v_i(t) = Vu(t) \tag{2.11}$$

$$i_i(t) = Iu(t) \tag{2.12}$$

donde $V > 0$, $I > 0$ y $u(t)$ corresponden respectivamente un voltaje DC, una corriente DC y al escalón unitario, y suponiendo condiciones iniciales nulas, entonces el voltaje de salida en la fase v_ϕ para ambas arquitecturas estará dado por:

$$v_{\phi v}(t) = V\left[1 - \exp\left(\frac{-t}{\tau}\right)\right] \tag{2.13}$$

$$v_{\phi i}(t) = \frac{I}{C_c}t \tag{2.14}$$

$$\tau \triangleq R_{\text{sub}}C_c \tag{2.15}$$

La constante de tiempo τ en la realidad depende de la resistencia en serie de la fuente R_{sub} y la de sustrato R_{sub}, las cuales en general se no se encuentran bien definidas. La tasa de cambio de voltaje o slew rate para ambas arquitecturas será:

$$\frac{dv_{\phi v}}{dt} = \frac{V}{\tau}\exp\left(\frac{-t}{\tau}\right) \tag{2.16}$$

$$\frac{dv_{\phi i}}{dt} = \frac{I}{C_c} \tag{2.17}$$

Las ecuaciones anteriores muestran la ventaja del control por corriente por sobre el control por voltaje. Por un lado, el esquema de control por voltaje define una tasa exponencial $dv\phi/dt$, variable en el tiempo, cuyo máximo valor sólo se alcanza en el inicio. Por otro lado, el control por corriente define un slew rate $dv\phi/dt$ que no depende de la resistencia de sustrato R_{sub} y es proporcional a la corriente de entrada I.

Tal y como se discutió con anterioridad, el tiempo de subida (o tiempo de bajada) t_r (t_f) y la excursión de voltaje V_{swing} se relacionan en forma directa con el desempeño del proceso de transferencia de carga. Si consideramos que el voltaje de la fase se encuentra limitado en slew rate (i.e el slew rate en la fase no puede superar cierto valor), entonces el control por corriente, comparado con el control por voltaje, permite establecer un slew rate constante máximo en todo momento, lo que es más eficiente en términos de tiempo.

Este análisis puede ser extendido al caso de la arquitectura de control por voltaje basada en un DAC de alto desempeño. La figura 2.11 muestra una comparación normalizada del establecimiento de la arquitectura de control por corriente y la arquitectura de control por voltaje basada en un DAC de alto desempeño.

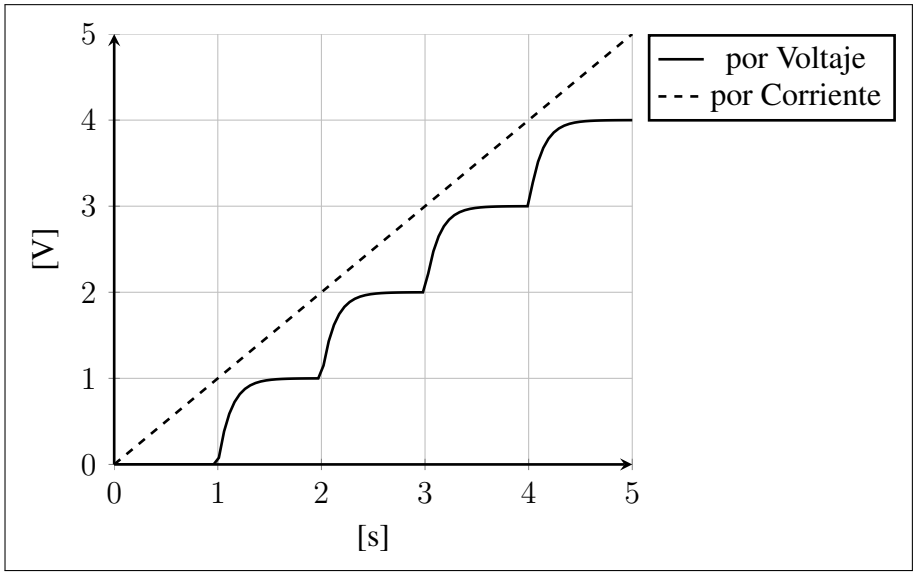


FIGURA 2.11. Establecimiento normalizado comparativo entre las arquitecturas de control por corriente y voltaje.

En la figura 2.11 se aprecia que el control por voltaje basado en un DAC de alto desempeño requiere por un lado una tasa de muestreo muy alta en comparación a la arquitectura de control por corriente. Por otro lado, el control por voltaje no define con precisión el slew rate en los instantes iniciales del muestreo, pudiendo superar su máximo valor establecido por el desempeño de la transferencia de carga.

Con el fin de validar analíticamente los resultados de la comparación anterior, este trabajo propone la siguiente cifra de mérito Λ_e:

$$\Lambda_e \triangleq \frac{\overline{\alpha^2} S^2}{A^2} \tag{2.18}$$

donde $\overline{\alpha^2}$ corresponde al error cuadrático medio entre la señal de voltaje de salida de la arquitectura a analizar y una rampa, S es el número de muestras utilizadas para definir la señal de salida, y A es la amplitud de la rampa. La idea detrás de la cifra de desempeño Λ_e es por un lado medir el error medio de la señal de salida de la arquitectura frente a una tasa de carga constante, y por otro lado, castigar el número de muestras utilizadas. Cada muestra adicional exige una corriente distinta de la fuente, corriente que puede producir ruido de fuente o también kickback noise que se puede introducir a otras partes del circuito. Una cifra de desempeño Λ_e menor indica que la arquitectura define de una mejor manera una tasa de voltaje constante frente a una carga capacitiva sin la necesidad de utilizar demasiadas muestras.

Primero calcularemos la cifra de mérito para la arquitectura de DAC de voltaje de alto desempeño. El error cuadrático medio $\overline{\alpha^2}$ puede ser aproximado como:

$$\overline{\alpha^2} \approx \frac{1}{T_v} S \int_0^{T_s} \left(v_v(t) - v_e(t)\right)^2 \, dt \tag{2.19}$$

donde

$$S = T_v/T_s \tag{2.20}$$

$$T_s = n_v \tau_v \tag{2.21}$$

$$v_v(t) = mT_s \left[1 - \exp\left(-\frac{t}{\tau_v}\right)\right] \tag{2.22}$$

$$v_e(t) = mt \tag{2.23}$$

con T_v el tiempo total, T_s el periodo de muestreo, τ_v es la constante de tiempo de establecimiento de una muestra, S el número de muestras, y m la pendiente de la rampa a definir. Finalmente, para la arquitectura basada en un DAC de alto desempeño, la cifra de mérito resultante es:

$$\Lambda_{e,v}(n_v) = \frac{\overline{\alpha^2}S^2}{A^2} = \frac{1}{6}\frac{(2{n_v}^2\mathrm{e}^{2n_v} - 9\mathrm{e}^{2n_v}n_v + 12\mathrm{e}^{2n_v} - 12\mathrm{e}^{n_v} - 3n_v)\,\mathrm{e}^{-2n_v}}{{n_v}^2} \tag{2.24}$$

donde $A = mT$ es la amplitud de la rampa y $n_v = T_s/\tau_v$ es la cantidad de constantes de tiempos que toma una muestra para establecerse.

El error cuadrático medio $\overline{\alpha^2}$ para la arquitectura de control por corriente estará dado por:

$$\overline{\alpha^2} = \frac{1}{T_i}\int_0^{T_i} \left(v_i(t) - v_e(t)\right)^2\, dt \tag{2.25}$$

donde

$$T_i = n_i\tau_i \tag{2.26}$$

$$v_i(t) = \int_0^t \frac{i_i(t)}{C_c}\, dt \tag{2.27}$$

$$i_i(t) = I\left[1 - \exp\left(-\frac{t}{\tau_i}\right)\right] \tag{2.28}$$

$$v_e(t) = \frac{I}{C_c}t \tag{2.29}$$

$$\tag{2.30}$$

con C_c la capacitancia de carga, I la amplitud de la corriente de carga, T_i el tiempo total de carga o de cálculo de error, τ_i la constante de tiempo de la corriente de carga, y n_i el número de constantes de tiempo en el que se calcula el error. Finalmente, la cifra de mérito Λ_e de la arquitectura de control por corriente propuesta es:

$$\Lambda_{e,i}(n_i) = \frac{1}{2}\frac{(2n_i\mathrm{e}^{2n_i} - 3\mathrm{e}^{2n_i} + 4\mathrm{e}^{n_i} - 1)\,\mathrm{e}^{-2n_i}}{{n_i}^3} \tag{2.31}$$

donde $\Lambda_{e,i}(n_i)$ depende solo del número de constantes de tiempo que transcurren durante el periodo de carga del capacitor.

Las ecuaciones (2.24) y (2.31) nos muestran que la cifra de mérito $\Lambda_e(n)$ para las arquitecturas de voltaje y corriente depende solamente de la cantidad de constantes de tiempo n. Estas cifras no dependen de parámetros como la amplitud de la señal A, la pendiente de la rampa m o la cantidad de muestras S utilizadas para el caso de $\Lambda_{e,v}(n)$. La figura 2.12 muestra ambas cifras de mérito en función de n.

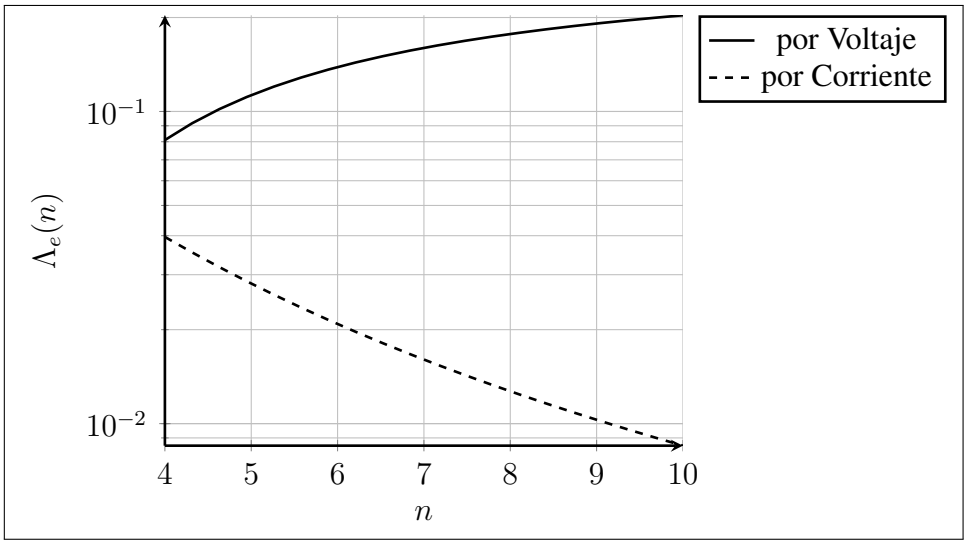


FIGURA 2.12. Cifra de mérito Λ_e para las arquitecturas de control por voltaje y corriente en función del número de constantes de tiempo n.

En la figura 2.12 se puede ver que por un lado que la cifra de mérito de la arquitectura por corriente propuesta $\Lambda_{e,i}(n)$ a diferencia de $\Lambda_{e,v}(n)$ es decreciente. La figura 2.12 solo nos muestra ambas cifras de mérito en función de el número de constantes de tiempo n, pero no nos entrega información comparativa alguna con respecto a la relación que existe entre las constantes de tiempo τ de ambas arquitecturas y el número de muestras utilizadas S. Si igualamos el tiempo de subida o tiempo de carga para ambas arquitecturas, obtenemos:

$$n_v \tau_v S = n_i \tau_i \Leftrightarrow n_v = n_i \frac{\tau_i}{\tau_v S} \tag{2.32}$$

La ecuación 2.32 nos muestra la relación que existe entre el número de constantes de tiempo de la arquitectura por voltaje n_v y el número de constantes de tiempo de la arquitectura de control por corriente propuesta n_i. Esta relación depende del cociente que existe entre la constante de tiempo de la arquitectura por corriente τ_i y la constante de tiempo de la arquitectura de control por voltaje τ_v amplificada por el número de muestras utilizadas S. Esto nos indica por un lado que si la constante de tiempo τ_i aumenta, entonces el gráfico de $\Lambda_{e,v}(n)$ de la figura 2.12 se comprime, si por otro lado S ó τ_v aumentan, entonces es el gráfico de $\Lambda_{e,i}(n)$ el que se comprime. En cualquiera de los dos casos se cumple que $\Lambda_{e,i}(n) < \Lambda_{e,v}(n)$, es decir, la cifra de mérito propuesta nos indica analíticamente que la arquitectura de control por corriente propuesta posee un mejor desempeño que las arquitecturas de control por voltaje tradicionales.

Este último análisis mediante la cifra de mérito $\Lambda_e(n)$ propuesta termina por validar analíticamente las ventajas de la arquitectura de control por corriente por sobre la arquitectura de control por voltaje tradicional a la hora de establecer una tasa de carga constante frente a una carga capacitiva.

3. FUENTE DE CORRIENTE DE HOWLAND MEJORADA

La fuente de corriente de Howland mejorada (IHCS de ahora en adelante) es un amplificador de transconductancia de entrada diferencial y salida referida a tierra bidireccional [Filho (2002); Ortiz, Vergara, y Mercado (2007); Pease (2008)]. La figura 3.1 nos muestra el diagrama circuital de la IHCS, donde $Z_L(s)$ corresponde a la impedancia de la carga.

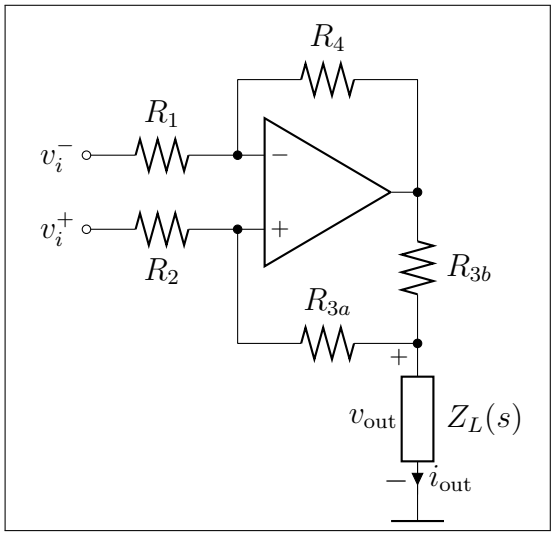


FIGURA 3.1. Diagrama Circuital de la IHCS.

La IHCS presenta las siguientes características:

- Entrada de voltaje diferencial.
- Salida de corriente referida a tierra bidireccional.
- Desempeño dependiente del amplificador operacional.
- Impedancia de salida dependiente de la tolerancia de los resistores.
- Alta eficiencia en el gasto de corriente.
- Simple de implementar.

A continuación se analizará en detalle el comportamiento dinámico teórico de la IHCS.

3.1 Análisis Ideal

El análisis ideal de la IHCS consiste en describir su comportamiento dinámico suponiendo componentes ideales.

La figura 3.2 muestra el equivalente circuital de la IHCS considerando al amplificador operacional como ideal.

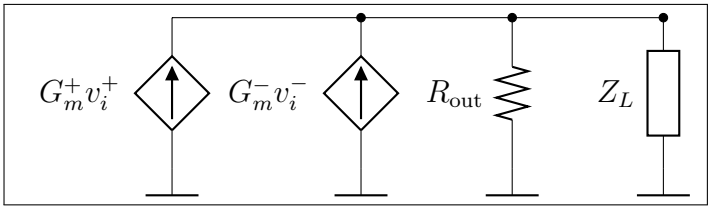


FIGURA 3.2. Diagrama Circuital de la IHCS.

En la figura 3.2, G_m^+, G_m^- y R_{out} corresponden respectivamente a la transconductancia efectiva de la entrada v_i^+, la transconductancia efectiva de la entrada v_i^- y la resistencia de salida de la IHCS. Es posible demostrar que:

$$G_m^+ = \frac{R_1(R_{3a}+R_{3b})+R_4R_{3a}}{R_1R_{3b}(R_2+R_{3a})} \tag{3.1}$$

$$G_m^- = -\frac{R_4}{R_1R_{3b}} \tag{3.2}$$

$$R_{\text{out}} = \frac{R_1R_{3b}(R_2+R_{3a})}{R_1(R_{3a}+R_{3b})-R_2R_4} \tag{3.3}$$

En las ecuaciones (3.1), (3.2) y (3.3) se puede apreciar que las expresiones de las transconductancias efectivas G_m^+ y G_m^- difieren y que el denominador de la resistencia de salida R_{out} posee una diferencia de parámetros.

Si suponemos que:

$$R_4 = R_1 \tag{3.4}$$

$$R_2 = R_{3a} + R_{3b} \tag{3.5}$$

entonces es posible demostrar que las expresiones dadas en (3.1), (3.2) y (3.3) se simplifican a:

$$G_m^+ = \frac{1}{R_{3b}} \tag{3.6}$$

$$G_m^- = -\frac{1}{R_{3b}} \tag{3.7}$$

$$R_{\text{out}} \rightarrow \pm\infty \tag{3.8}$$

De las ecuaciones (3.6), (3.7) y (3.8) podemos ver por un lado que las expresiones de transconductancia se simplifican y dependen sólo de la resistencia R_{3b}. Por otro lado, la impedancia de salida tiende a $\pm\infty$.

Del análisis ideal se desprende que la IHCS se puede comportar como un amplificador de transconductancia diferencial ideal, esto es, con una transconductancia diferencial constante y con una resistencia de salida infinita.

3.2 Análisis de Primer Orden

Si ahora consideramos que el amplificador operacional posee una ganancia $A(s)$ dada por:

$$A(s) = \frac{A_{\text{ol}}}{1 + s\tau} \tag{3.9}$$

donde A_{ol} y τ corresponden a la ganancia de lazo abierto DC y a la constante de tiempo del amplificador operacional, respectivamente, entonces las expresiones de transconductancia efectiva $G_m^i(s)$ e impedancia de salida $Z_{\text{out}}(s)$ toman la siguiente forma:

$$G_m^+(s) = g_m^+\left(\frac{1-s/z_g^+}{1-s/p_g^+}\right) \tag{3.10}$$

$$G_m^-(s) = g_m^-\left(\frac{1}{1-s/p_g^-}\right) \tag{3.11}$$

$$Z_{\text{out}}(s) = z_{\text{out}}\left(\frac{1-s/z_z}{1-s/p_z}\right) \tag{3.12}$$

donde

$$g_m^+ \approx \frac{R_1R_{3a}+R_1R_{3b}+R_4R_{3a}}{R_1R_{3b}(R_2+R_{3a})} \tag{3.13}$$

$$g_m^- \approx -\frac{R_4}{R_1R_{3b}} \tag{3.14}$$

$$z_z \approx -\frac{R_1}{R_1+R_4}\frac{A_{\text{ol}}}{\tau} \tag{3.15}$$

$$z_g^+ \approx -\frac{R_1(R_{3a}+R_{3b})+R_4R_{3a}}{R_{3b}(R_1+R_4)}\frac{A_{\text{ol}}}{\tau} \tag{3.16}$$

$$p_g^+ \approx -\frac{R_1}{R_1+R_4}\frac{A_{\text{ol}}}{\tau} \tag{3.17}$$

$$p_g^- \approx -\frac{R_1}{R_1+R_4}\frac{A_{\text{ol}}}{\tau} \tag{3.18}$$

$$\tag{3.19}$$

$$z_{\text{out}} \approx \frac{A_{\text{ol}}R_1R_{3b}(R_2+R_{3a})}{R_1R_2+R_2R_4+R_1R_{3a}+R_1R_{3b}+R_4R_{3a}+R_4R_{3b}+A_{\text{ol}}[R_1(R_{3a}+R_{3b})-R_2R_4]} \tag{3.20}$$

$$p_z = -\frac{R_1R_2+R_2R_4+R_1R_{3a}+R_1R_{3b}+R_4R_{3a}+R_4R_{3b}+A_{\text{ol}}[R_1(R_{3a}+R_{3b})-R_2R_4]}{\tau(R_1+R_4)(R_2+R_{3a}+R_{3b})} \tag{3.21}$$

Si se cumplen las relaciones de resistencias dadas en (3.4) y (3.5), y además suponemos que la ganancia estática del amplificador operacional A_{ol} posee un valor muy alto, entonces las expresiones anteriores toman la siguiente forma:

$$g_m^+ \approx \frac{1}{R_{3b}} \tag{3.22}$$

$$g_m^- \approx -\frac{1}{R_{3b}} \tag{3.23}$$

$$z_z \approx -\frac{1}{2}\frac{A_{\rm ol}}{\tau} \tag{3.24}$$

$$z_g^+ \approx -\frac{2R_{3a}+R_{3b}}{2R_{3b}}\frac{A_{\rm ol}}{\tau} \tag{3.25}$$

$$p_g^+ \approx -\frac{1}{2}\frac{A_{\rm ol}}{\tau} \tag{3.26}$$

$$p_g^- \approx -\frac{1}{2}\frac{A_{\rm ol}}{\tau} \tag{3.27}$$

$$z_{\rm out} \approx \frac{R_{3b}(2R_{3a}+R_{3b})}{R_{3a}+R_{3b}}\frac{A_{\rm ol}}{4} \tag{3.28}$$

$$p_z \approx -\frac{1}{\tau} \tag{3.29}$$

La utilidad de las ecuaciones presentadas con anterioridad depende principalmente del nivel de tolerancia de los resistores. Si se desea analizar el comportamiento de los parámetros con respecto al nivel de tolerancia de los resistores las ecuaciones a considerar son las expresiones que no consideran la simplificación dadas por (3.4) y (3.5).

3.3 Análisis de Estabilidad

Para analizar la estabilidad del sistema realimentado utilizaremos el modelo de ganancia asintótica o modelo de Rosenstark [Rosenstark (1986)]. Este modelo nos permite escribir la función de transferencia $G_f(s)$ de un sistema realimentado como:

$$G_f(s) \equiv \frac{G_{\rm out}(s)}{G_{\rm in}(s)} = G_\infty(s)\frac{T(s)}{1+T(s)} + G_0(s)\frac{1}{1+T(s)} \tag{3.30}$$

donde $G_\infty(s)$, $G_0(s)$ y $T(s)$ corresponden a la ganacia asintótica, la transmisión directa y la razón de retorno del sistema, respectivamente. La figura 3.3 muestra el diagrama de bloques del modelo de ganancia asintótica.

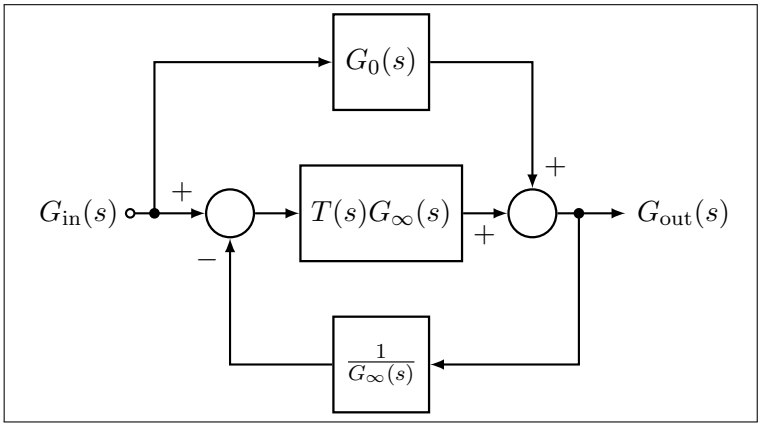


FIGURA 3.3. Diagrama de Bloques Modelo Ganancia Asintótica.

Si ahora consideramos el diagrama circuital de la IHCS mostrado en la figura 3.1, entonces el modelo de ganancia asintótica del sistema estará dado por:

$$G_f^+(s) \equiv \frac{I_{\rm out}(s)}{V_i^+(s)} = G_\infty^+(s)\frac{T(s)}{1+T(s)} + G_0^+(s)\frac{1}{1+T(s)} \tag{3.31}$$

$$G_f^-(s) \equiv \frac{I_{\rm out}(s)}{V_i^-(s)} = G_\infty^-(s)\frac{T(s)}{1+T(s)} \tag{3.32}$$

donde

$$G_\infty^+(s) = G_m^+\frac{R_{\rm out}}{R_{\rm out}+Z_L(s)} \tag{3.33}$$

$$G_\infty^-(s) = G_m^-\frac{R_{\rm out}}{R_{\rm out}+Z_L(s)} \tag{3.34}$$

$$G_0^+(s) = \frac{R_{3b}}{R_2R_{3b}+R_{3a}R_{3b}+(R_2+R_{3a}+R_{3b})Z_L(s)} \tag{3.35}$$

con G_m^+, G_m^- y $R_{\rm out}$ dados por las expresiones (3.1), (3.2) y (3.3), respectivamente, y

$$T(s) = A(s)\frac{R_1R_2R_{3b}+R_1R_{3a}R_{3b}+(R_1R_{3a}+R_1R_{3b}-R_2R_4)Z_L(s)}{(R_1+R_4)(R_2R_{3b}+R_{3a}R_{3b}+[R_2+R_{3a}+R_{3b}]Z_L(s))} \tag{3.36}$$

A continuación se analizará la estabilidad de la IHCS frente a distintos tipos de carga.

3.3.1 Respuesta Frente a Carga RC

Si consideramos una carga RC serie dada por:

$$Z_L(s) = R_{\text{sub}} + \frac{1}{sC_c} \tag{3.37}$$

es posible demostrar que la estabilidad del sistema depende sólo de la razón de retorno $T(s)$ del sistema. La expresión de la razón de retorno $T(s)$ para la carga RC está dada por:

$$T(s) = T_0^* \frac{1 - z_T(s)}{1 - p_T(s)} A(s) \tag{3.38}$$

donde

$$T_0^* = \frac{R_1(R_{3a} + R_{3b}) - R_2 R_4}{(R_1 + R_4)(R_2 + R_{3a} + R_{3b})} \tag{3.39}$$

$$z_T(s) = -\frac{1}{C_c \left(R_{\text{sub}} + \frac{R_1 R_{3b}(R_2 + R_{3a})}{R_1(R_{3a} + R_{3b}) - R_2 R_4} \right)} \tag{3.40}$$

$$p_T(s) = -\frac{1}{C_c \left(R_{\text{sub}} + \frac{R_2 R_{3b} + R_{3a} R_{3b}}{R_2 + R_{3a} + R_{3b}} \right)} \tag{3.41}$$

Para describir la estabilidad de este sistema nos bastará con utilizar la expresión de ganancia del amplificador operacional de un polo. Esto es:

$$A(s) = \frac{A_{\text{ol}}}{1 + s\tau} \tag{3.42}$$

Dado lo anterior, la razón de retorno poseerá dos polos y un cero. El cero se encuentra cercano al origen debido a las relaciones dadas por (3.4) y (3.5), mientras que los polos se ubican a más altas frecuencias.

Finalmente, el sistema es siempre estable debido a que la razón de retorno $T(s)$ posee un cero dominante (baja frecuencia) y dos polos de alta frecuencia, lo que asegura un margen de fase de a lo menos $90°$.

Para validar el análisis de estabilidad, se procedió a evaluar la razón de retorno $T(s)$. La tabla 3.1 muestra los parámetros utilizados.

TABLA 3.1. Parámetros Análisis Estabilidad RC.

Parámetro	**Valor**
A_{ol}	$100\,\mathrm{dB}$
ABW	$230\,\mathrm{MHz}$
δ	$0.1\ \%$
R_1	$(1+\delta)270\,\Omega$
R_2	$(1-\delta)(270\,\Omega + 22.1\,\Omega)$
R_{3a}	$(1+\delta)270\,\Omega$
R_{3b}	$(1+\delta)22.1\,\Omega$
R_4	$(1-\delta)270\,\Omega$
C_c	$1\,\mathrm{nF}$
R_{sub}	$1\,\Omega$

La figura 3.4 muestra la razón de retonto $T(s)$ evaluada.

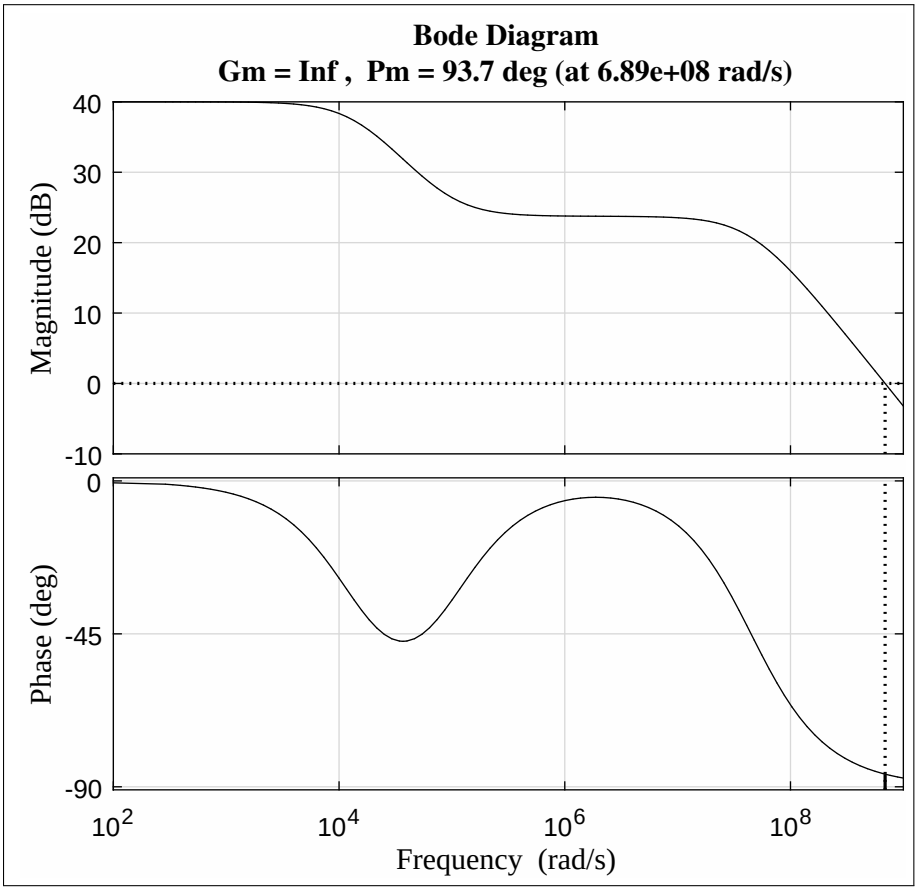


FIGURA 3.4. $T(s)$ frente a una carga RC serie.

En la figura 3.4 se puede apreciar que el sistema frente a una carga RC es siempre estable.

3.3.2 Respuesta Frente a Carga RC con Inductancias Parásitas

Cuando consideramos la inductancia parásita de la línea L_{par}, la carga $Z_L(s)$ toma la siguiente forma:

$$Z_L(s) = R_{\text{sub}} + \frac{1}{sC_c} + sL_{\text{par}} \tag{3.43}$$

Dado lo anterior, la expresión de la razón de retorno puede ser escrita como:

$$T(s) = \frac{a_z s^2 + b_z s + c_z}{a_p s^2 + b_p s + c_p} A(s) = T^*(s) A(s) \tag{3.44}$$

donde

$$a_z = \xi_R L_{\text{par}} \tag{3.45}$$

$$b_z = R_1 R_2 R_{3b} + R_1 R_{3a} R_{3b} + \xi_R R_{\text{sub}} \tag{3.46}$$

$$c_z = \frac{\xi_R}{C_c} \tag{3.47}$$

$$a_p = L_{\text{par}}(R_2 + R_{3a} + R_{3b}) \tag{3.48}$$

$$b_p = R_2 R_{3b} + R_{3a} R_{3b} + (R_2 + R_{3a} + R_{3b}) R_{\text{sub}} \tag{3.49}$$

$$c_p = \frac{R_2 + R_{3a} + R_{3b}}{C_c} \tag{3.50}$$

$$\xi_R = R_1(R_{3a} + R_{3b}) - R_2 R_4 \tag{3.51}$$

De las expresiones anteriores se puede observar que el efecto principal de la inductancia parásita es la disminución del discriminante y del punto medio de los polos y ceros de $T^*(s)$. Por otro lado, la resistencia de línea R_{sub} produce el efecto contrario.

En el análisis de estabilidad, sólo juegan un rol preponderante los polos de $T^*(s)$ y los polos de alta frecuencia de $A(s)$. Si consideramos que el amplificador operacional posee una ganancia $A(s)$ dada por:

$$A(s) = \frac{A_{\text{ol}}}{(1 + s\tau_1)(1 + s\tau_2)} \tag{3.52}$$

entonces el polo no dominante de $A(s)$ en conjunción con los polos de $T * (s)$, cuya ubicación y separación dependen de L_{par}, pueden establecer un margen de fase del sistema que está por fuera de los rangos aceptables de diseño.

Para validar el análisis, esta vez se utilizó el diagrama de Nyquist de la razón de retorno $T(s)$, debido a que el diagrama de Bode obtenido no permite determinar la estabilidad del sistema (J. Hahn, T. Edison, y T. Edgar, 2001). Para ello, se utilizaron los mismos parámetros de la tabla 3.1, con $C_c = 160\,\mathrm{nF}$, $\tau_1/\tau_2 = 6.66\mathrm{E}{-6}$ y $L_{\mathrm{par}} = 150\,\mathrm{nH}$. La figura 3.5 muestra la razón de retorno $T(s)$.

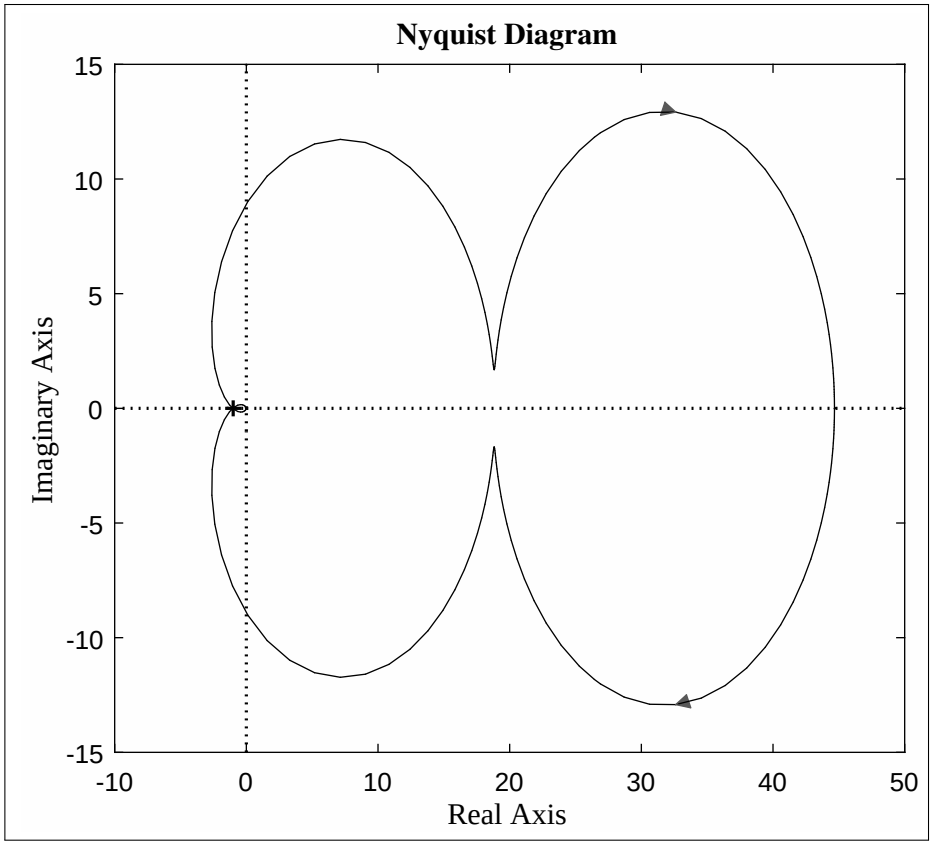


FIGURA 3.5. $T(s)$ frente a una carga RC serie con inductancia parásita.

El efecto de la inductancia parásita L_{par} en el diagrama de Nyquist es disminuir el margen de ganancia y el margen de fase de la razón de retorno $T(s)$. Para nuestro caso evaluado, basta una inductancia parásita de L_{par} de $150\,\mathrm{nH}$ para volver inestable al sistema.

3.3.3 Compensación Frente a Carga RC con Inductancias Parásitas

En la subsección anterior se mostró el efecto de las inductancias parásitas de la carga sobre el margen de fase de la razón de retorno del sistema. Para compensar este efecto, se propone añadir un resistor de compensación R_{comp} en serie con la carga. Esta idea se basa en los resultados descritos en la ecuación (3.44).

Utilizando los mismos datos empleados para la figura 3.5 y un resistor de compensación de $R_{\text{comp}} = 20\,\Omega$ se obtiene la siguiente razón de retorno $T(s)$.

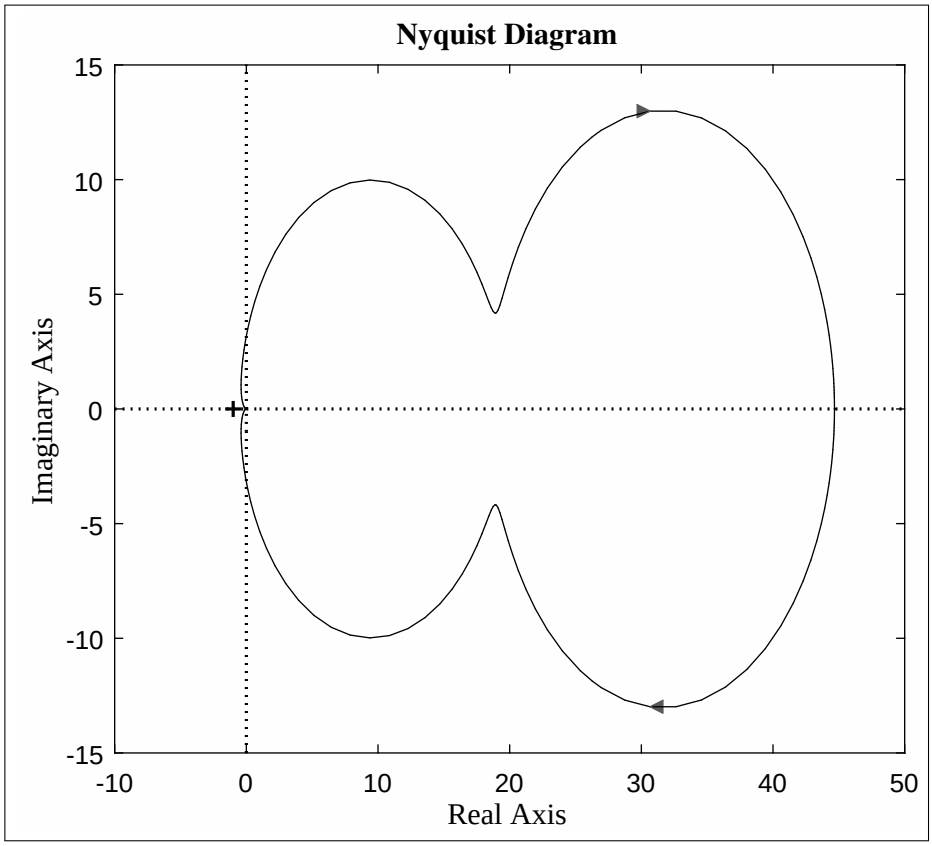


FIGURA 3.6. $T(s)$ frente a una carga RC serie con inductancia parásita y compensación.

El efecto del resistor de compensación es aumentar el margen de ganancia y de fase de la razón de retorno $T(s)$. Para el caso evaluado, el sistema pasó de tener margenes de ganancia y de fase negativos, a tener margenes de $30.5\,\mathrm{dB}$ y $66.5°$.

Tal y como se mencionó en la subsección anterior, la resistencia de compensación en serie con la carga, R_{comp}, se opone al efecto desestabilizador de la inductancia parásita L_{par} cuando la IHCS maneja una carga resistivo-capacitiva serie.

3.4 Análisis de Ruido

Para efectuar el análisis de ruido consideraremos la ganancia del amplificador operacional $A(s)$ y la impedancia de la carga $Z_L(s)$. La figura 3.7 nos muestra el equivalente circuital de la IHCS utilizado para el análisis de ruido.

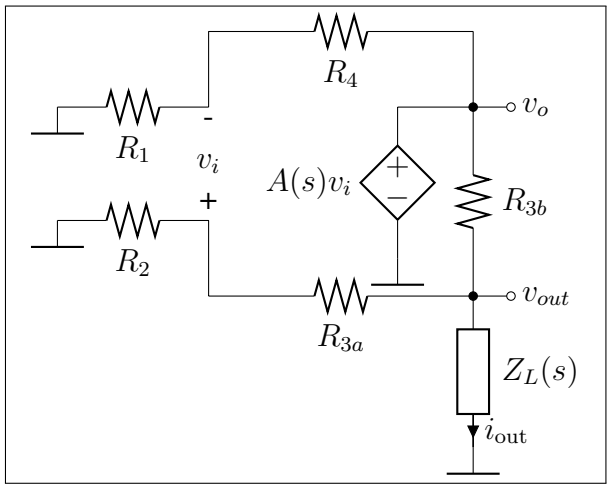


FIGURA 3.7. Equivalente circuital de la IHCS para el análisis de ruido.

Para efectos de análisis, consideraremos el ruido de voltaje del amplificador operacional como el ruido predominante.

Sea $V_n(s)$ el ruido de voltaje del amplificador operacional referido a su entrada. Entonces, es posible demostrar que la transconductancia de ruido a la salida $G_n(s)$ estará dada por:

$$G_n(s) \equiv \frac{I_{\text{out}}(s)}{V_n(s)} = \frac{1}{k_1/A(s) + k_2 Z_L(s)/A(s) + k_3 Z_L(s) + k_4} \tag{3.53}$$

donde:

$$k_1 = R_{3b} \tag{3.54}$$

$$k_2 = \frac{R_2 + R_{3a} + R_{3b}}{R_2 + R_{3a}} \tag{3.55}$$

$$k_3 = \frac{\xi_R}{(R_1 + R_4)(R_2 + R_{3a})} \tag{3.56}$$

$$k_4 = \frac{R_1 R_{3b}}{R_1 + R_4} \tag{3.57}$$

$$\xi_R \equiv R_1(R_{3a} + R_{3b}) - R_2 R_4 \tag{3.58}$$

La relación (3.53) es útil en el sentido de que es posible separar los efectos de $A(s)$, $Z(s)$ y los componentes resistivos de la IHCS.

Si consideramos como ideal al amplificador operacional y que se cumplen las relaciones de igualdad dadas por (3.4) y (3.5), entonces la transconductancia de ruido a la salida estará dada por:

$$G_n(s) \approx \frac{R_1 + R_4}{R_1 R_{3b}} = \frac{2}{R_{3b}} \tag{3.59}$$

resultado que dista del exhibido en [Tucker, Fox, y Sadleir (2013)]. La relación (3.59) nos indica que, idealmente, la transconductancia de ruido a la salida $G_n(s)$ es inversamente proporcional a la resistencia R_{3b}.

A continuación se analizará el ruido frente a una carga RC.

3.4.1 Análisis de Ruido Frente a una Carga RC

Para efectuar el análisis de ruido frente a una carga RC consideraremos que:

$$A(s) = \frac{A_{\text{ol}}}{1+s\tau} \tag{3.60}$$

$$Z_L(s) = R_{\text{sub}} + \frac{1}{sC_c} \tag{3.61}$$

Si separamos los factores dependientes de la frecuencia de la expresión (3.53) encontramos que:

$$1/A(j\omega) = \frac{1}{A_{\text{ol}}} + j\frac{\tau_1\omega}{A_{\text{ol}}} \tag{3.62}$$

$$Z_L(j\omega)/A(j\omega) = \frac{\tau_1 + C_cR_{\text{sub}}}{C_cA_{\text{ol}}} + j\frac{C_cR_{\text{sub}}\tau_1\omega^2 - 1}{\omega C_cA_{\text{ol}}} \tag{3.63}$$

$$Z_L(j\omega) = R_{\text{sub}} + j\frac{-1}{\omega C_c} \tag{3.64}$$

De las expresiones anteriores podemos notar que su componente real no depende de la frecuencia angular ω. Es por esto que el máximo de $|G_n(j\omega)|$ se dará cuando la parte imaginaria del denominador se anule. Esto es:

$$|G_n(j\omega)|^{-1}_{\text{max}} = \frac{k_1}{A_{\text{ol}}} + k_2\frac{\tau_1 + C_cR_{\text{sub}}}{C_cA_{\text{ol}}} + k_3R_{\text{sub}} + k_4 \tag{3.65}$$

$$\approx \frac{R_1R_{3b}(R_2+R_{3a}) + R_{\text{sub}}\xi_R}{(R_1+R_4)(R_2+R_{3a})} \tag{3.66}$$

Por otro lado, el ancho de banda de $|G_n|$ estará dado por:

$$\text{BW}_n \approx \frac{A_{\text{ol}}}{\tau}\frac{1}{R_{3b}+R_{\text{sub}}}|G_n(j\omega)|^{-1}_{\text{max}} \tag{3.67}$$

Finalmente, si consideramos el ruido a la entrada del amplificador operacional $\overline{e_n}$ como blanco, es posible aproximar el ruido integrado total de corriente a la salida N_t como:

$$\frac{N_t}{V_n} \approx 2\sqrt{\mathrm{BW}_n}|G_n(j\omega)|_{\max} \approx 2\sqrt{\frac{A_{\mathrm{ol}}}{\tau_1}\frac{1}{R_{3b}+R_{\mathrm{sub}}}|G_n(j\omega)|_{\max}} \tag{3.68}$$

De la ecuación de ruido (3.68) podemos concluir lo siguiente acerca de la corriente de ruido total a la salida en cuadratura para una carga RC:

- Es proporcional al producto ganancia - ancho de banda ABW del amplificador operacional.
- Disminuye al aumentar R_{3b} o R_{sub}.
- No depende de la capacitancia equivalente del CCD C_C.

Para validar el análisis se evaluó la densidad espectral de corriente a la salida utilizando los valores de la tabla 3.1. La figura 3.8 muestra el resultado.

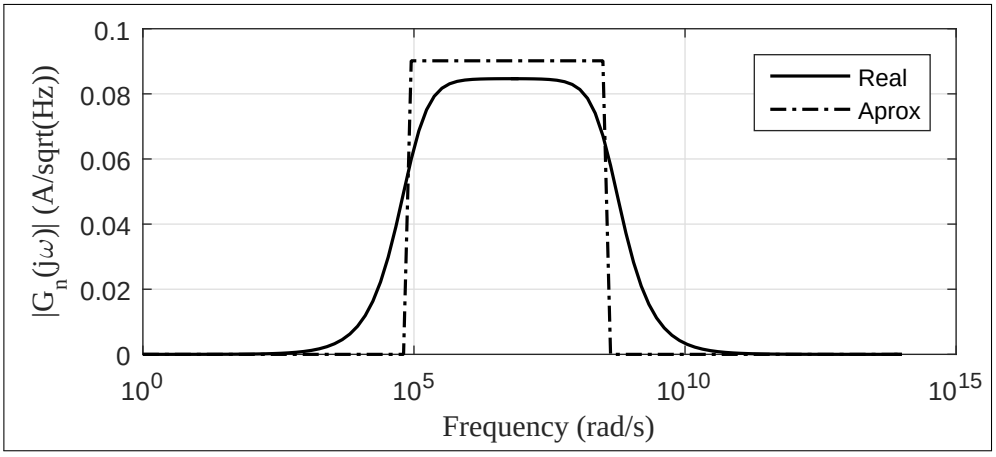


FIGURA 3.8. Densidad espectral de corriente a la salida de la IHCS considerando una Carga RC serie.

El ruido de corriente integrado total a la salida normalizado N_t/V_n es de $2.12\,\mathrm{kA/V}$ para el caso real y $1.76\,\mathrm{kA/V}$ para el caso aproximado, lo que verifica las ecuaciones simplificadas propuestas.

4. IMPLEMENTACIÓN

Este capítulo tiene por objetivo describir tanto las especificaciones de diseño como las consideraciones prácticas en la elaboración del prototipo de driver de reloj controlado por corriente.

Cabe aclarar que el propósito inicial de este trabajo fue desarrollar un ASIC o circuito integrado de aplicación específica que generase las señales de corriente necesarias. Tal y como se verá más adelante, la necesidad de manejar altos voltajes y altas corrientes a la salida descartó la posibilidad de diseñar un ASIC.

4.1 Especificaciones

Para comenzar este capítulo se deben definir las especificaciones que el driver de señales de reloj por corriente debe satisfacer. Para ello dividiremos las especificaciones del sistema en especificaciones de señal de reloj del CCD y especificaciones circuitales. A continuación se detalla cada una de estas sub-secciones.

4.1.1 Especificaciones del CCD

La figura 4.1 nos muestra la forma de señal típica de una fase de reloj de un detector CCD.

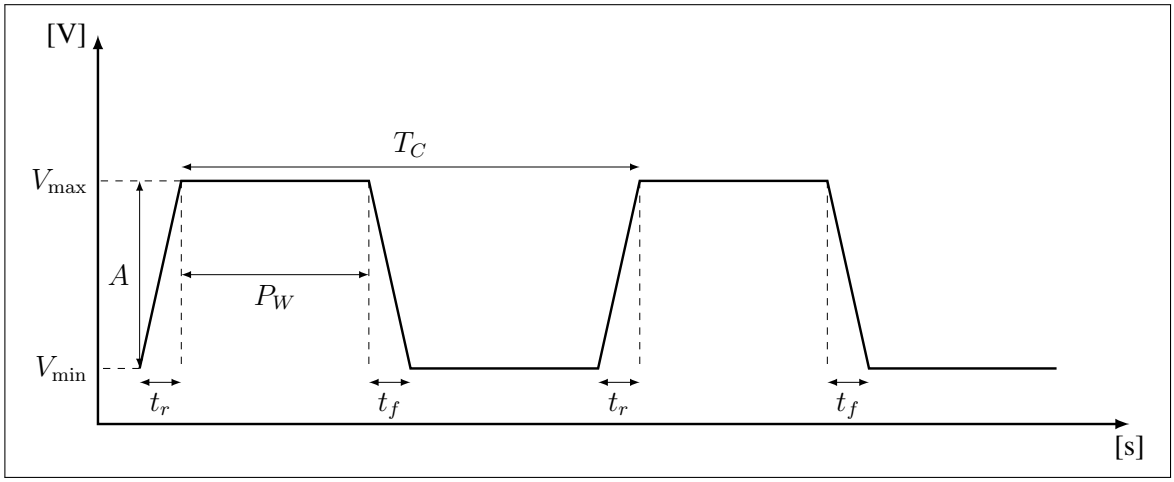


FIGURA 4.1. Diagrama general de una señal de reloj.

Si consideramos la fase del CCD como un capacitor de capacitancia C_c a tierra y que los tiempos de subida y bajada son iguales, entonces es posible demostrar que:

$$|I_{\max}| = \frac{AC_C}{t_r} \tag{4.1}$$

$$I_{\text{rms}} = AC_c\sqrt{\frac{2}{T_c t_r}} \tag{4.2}$$

$$P_{\text{rms}} = AC_c\sqrt{\frac{2\left(V_{\max}^2 + V_{\max}V_{\min} + V_{\min}^2\right)}{3T_c t_r}} \tag{4.3}$$

donde $I_{\max}$, I_{rms} y P_{rms} corresponden a la corriente máxima, la corriente efectiva RMS y la potencia efectiva RMS vista en la fase del CCD dada la forma de onda de la figura 4.1.

Las tablas 4.1 y 4.2 nos muestran los rangos típicos de especificaciones de señales de reloj para distintos CCD comerciales para la fase de imagen y registro, respectivamente[1].

[1]Ver tabla recopilada de datos en anexo A.

TABLA 4.1. Rangos de especificaciones señales de reloj fase de imagen.

	C_c [F]	$V_{\min}$ [V]	$V_{\max}$ [V]	T_r [s]	T_C [s]
Min	2.0E−10	0	10	1.0E−7	2.0E−6
Max	3.2E−7	0	12	1.0E−5	3.0E−4

TABLA 4.2. Rangos de especificaciones señales de reloj fase de registro.

	C_c [F]	$V_{\min}$ [V]	$V_{\max}$ [V]	T_r [s]	T_C [s]
Min	3.4E−11	0	11	5.0E−8	1.0E−6
Max	7.1E−10	1	12	6.0E−5	5.0E−5

Por otro lado, las tablas 4.3 y 4.4 nos muestran las características derivadas utilizando las ecuaciones 4.1, 4.2 y 4.3.

TABLA 4.3. Especificaciones derivadas señales de reloj fase de imagen.

	$\lvert I_{\max}\rvert$ [A]	I_{rms} [Arms]	P_{rms} [Wrms]
Min	1.9E−2	7.6E−3	5.3E−2
Max	8.8E−1	1.6E−1	9.6E−1

TABLA 4.4. Especificaciones derivadas señales de reloj fase de registro.

	$\lvert I_{\max}\rvert$ [A]	I_{rms} [Arms]	P_{rms} [Wrms]
Min	4.0E−5	2.8E−4	1.8E−3
Max	7.4E−2	3.5E−2	2.5E−1

De las tablas 4.1, 4.2, 4.3 y 4.4 podemos ver que las especificaciones para las señales de fase de imagen y registro difieren bastante. Las magnitudes relevantes a comparar enfocados en el diseño son tiempo de subida t_r, la corriente máxima y la corriente máxima RMS.

A continuación, en la subsección 4.1.2 se definirán y discutirán las especificaciones del diseño circuital para las señales de reloj de imagen y registro.

4.1.2 Especificaciones Circuitales

En esta subsección definiremos las especificaciones circuitales generales que el driver de señales de reloj por corriente debe satisfacer. La tabla 4.5 muestra las especificaciones circuitales generales.

TABLA 4.5. Especificaciones circuitales generales.

Fase	$\lvert I_{\max}\rvert$ [A]	I_{rms} [Arms]	$V_{\min}$ [V]	$V_{\max}$ [V]	B	F_s [Hz]
Imagen	1	0.1	0	14	8	1.0E8
Registro	0.1	0.1	0	14	8	1.0E8

A continuación se presenta una breve discusión de cada uno de las especificaciones.

- Corriente Máxima ($\lvert I_{\max}\rvert$): La corriente máxima absoluta se deriva directamente de las tablas 4.3 y 4.4. Este valor propone un reto para el diseño del amplificador de imagen.
- Corriente RMS (I_{rms}): La corriente efectiva se deriva de las tablas 4.3 y 4.4 con la salvedad de que la corriente efectiva para el amplificador de imagen es menor. Dado esto, el driver de imagen deberá tener un periodo de señal de reloj T_C mayor, o alternativamente, manejar cargas capacitivas C_c menores.
- Excursión de Voltaje ($V_{\min}$ y $V_{\max}$): La excursión de voltaje se deriva directamente de las tablas 4.3 y 4.4.

- Resolución (B): La resolución del driver de señales de reloj por corriente hace referencia al rango completo $2|I_{\max}|$. Este valor no posee un sustento teórico en la literatura y es más bien propositivo.
- Frecuencia de muestreo (F_s): Este valor está definido por el amplificador de registro y se utiliza para ambos tipos de amplificadores. El driver de señales de reloj por corriente debe muestrear al menos cinco veces más rápido que el tiempo de subida t_r mínimo de la fase de registro. Este valor asegura un buen establecimiento de la señal de corriente a la salida y permite un control más preciso de la pendiente de la forma de onda de salida.
- No se especifica el nivel de ruido a la salida debido a que no existe sustento teórico en la literatura.

4.2 Diseño Circuital

Esta sección tratará en detalle el diseño circuital de cada una de las secciones del driver de señales de reloj por corriente.

La figura 4.2 muestra el diagrama general del sistema diseñado.

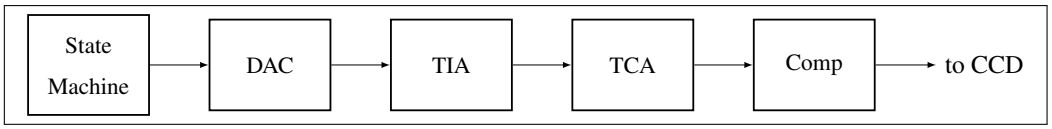


FIGURA 4.2. Diagrama general sistema diseñado.

A continuación se describe en detalle el diseño de cada uno de los bloques que componen el diagrama de la figura 4.2 basados en los resultados teóricos tratados y desarrolladas en la sección 3.

4.2.1 Máquina de Estados Principal

La máquina de estados principal corresponde al "corazón" del sistema. Ésta debe ser capaz de configurar todos los periféricos externos, y de conformar las palabras digitales de los DACs de corriente.

Para este propósito se utilizó el FPGA "XC6SLX9-3TQG144C" de la familia Spartan6 de Xilinx. Este FPGA posee las siguientes características principales [*Spartan-6 LX FPGAs* (s.f.)]:

- Reloj de operación principal de más de $100\,\mathrm{MHz}$.
- 102 entradas y salidas de propósito general.
- Programación mediante puerto JTAG.
- 0 a $3.75\,\mathrm{V}$ de operación.
- Bajo costo.

Este FPGA fue utilizado en proyectos anteriores en donde se verificó su desempeño.

4.2.2 Conversores Digital - Análogos (DAC)

Los conversores digital - análogos (DAC) son los encargados de conformar una señal analógica a partir de una señal digital de entrada. En nuestro diseño se seleccionaron conversores con salida de corriente diferenciales por sobre los de salida de voltaje diferencial. Esto se debe a que el prototipo diseñado con elementos electrónicos discretos debe ser replicable en un "ASIC" o "circuito integrado de aplicación específica" en un futuro.

Dadas las tablas 4.3 y 4.4, se optó por el DAC de video "ADV7125 - LQFP48", que posee las siguientes características [*ADV7125 — datasheet and product info 330MHz Triple 8-Bit High Speed Video DAC — Analog Devices* (s.f.)]:

- Resolución de 8 bits.
- 3 Salidas Diferenciales.
- $330\,\mathrm{MHz}$ de operación.
- Entrada paralela.
- $20\,\mathrm{mA}$ de corriente de salida nominal.
- Posibilidad de referencia interna o externa.
- $75\,\Omega$ de salida para desempeño óptimo.
- $1.4\,\mathrm{V}$ de rango de salida.
- Modo de bajo consumo.
- Bajo costo.

Este DAC de video fue seleccionado por su cantidad de salidas diferenciales y su bajo costo.

4.2.3 Amplificadores de Transimpedancia (TIA)

Los amplificadores de transimpedancia son los encargados de amplificar la corriente diferencial a la salida de los conversores análogo digitales y convertirla en una señal de

voltaje diferencial de amplitud adecuada.

Basados en simulaciones, finalmente se escogió la topología de amplificador de transimpedancia no inversor. La figura 4.3 nos muestra la configuración del amplificador.

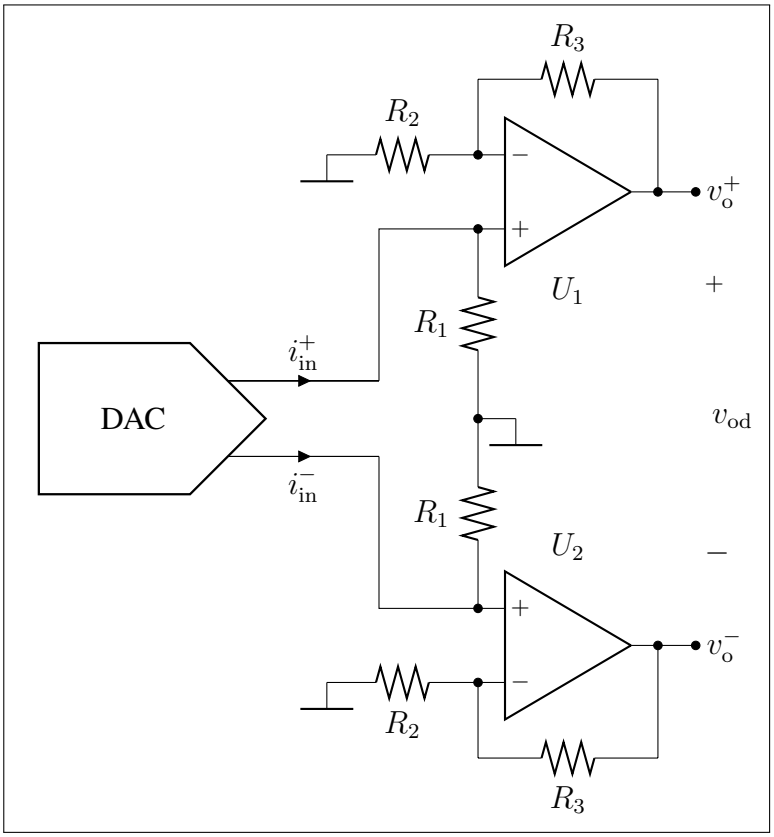


FIGURA 4.3. Diagrama circuital amplificador de transimpedancia.

Esta topología no inversora presenta las siguientes ventajas en comparación con su símil inversor:

- Conserva el producto ganancia-ancho de banda del amplificador operacional.
- No necesita polarizaciones externas para manejar la inversión de la configuración inversora.

- Es simple definir la resistencia de entrada y permanecer dentro de la excursión de voltaje del DAC.

Las especificaciones a cumplir por el amplificador son las siguientes:

- Alimentación de $3.3\,\mathrm{V}$
- Ganancia de voltaje menor a $G = 2{,}5$.
- Resistencia de entrada menor a $70\,\Omega$ debido a la excursión de voltaje de los DAC.
- Excursión de voltaje a la salida de $2.21\,\mathrm{V}$.
- Slew Rate mayor a $250\,\mathrm{V/\mu s}$.
- Bajo nivel de ruido.
- Salida riel a riel.
- Baja distorsión.
- Alta impedancia de entrada.
- Bajo costo.

Para ello, se utilizó el amplificador operacional "LTC6269", que posee las siguientes características [*LTC6269 - Dual 500MHz Ultra-Low Bias Current FET Input Op Amp - Linear Technology* (s.f.)]:

- $500\,\mathrm{MHz}$ de producto ganancia-ancho de banda.
- Rango de alimentación de $3.1\,\mathrm{V}$ a $5.25\,\mathrm{V}$
- Bajo nivel de ruido ($4.3\,\mathrm{nV/Hz^{1/2}}$ a $100\,\mathrm{MHz}$).
- Salida riel a riel.
- Slew rate de $400\,\mathrm{V/\mu s}$.
- Baja distorsión.
- Dos amplificadores por integrado.
- Resistencia de entrada (modo común y diferencial) mayor a $1000\,\mathrm{G\Omega}$.
- Utilizado como amplificador de transimpedancia.

La tabla 4.6 nos muestra los componentes utilizados en el diseño del amplificador de transimpedancia.

TABLA 4.6. Componentes amplificador de transimpedancia.

Componente	**Descripción**
R_1	Res $51\,\Omega$, ± 1 %
R_2	Res $51\,\Omega$, ± 1 %
R_3	Pot $50\,\Omega$, ± 10 %, 25 Vueltas
U_i	LTC6269

En la tabla 4.6 se puede apreciar que R_3 es un potenciómetro de 25 vueltas. Esto es para definir en forma arbitraria y precisa la ganancia del amplificador.

4.2.4 Amplificadores de Transconductancia (TCA)

Los amplificadores de TransConductancia (TCA) son los encargados de convertir la señal de voltaje diferencial proveniente de la etapa de amplificación de transimpedancia en una corriente de salida referida a tierra bidireccional. La figura 4.4 nos muestra la configuración del amplificador.

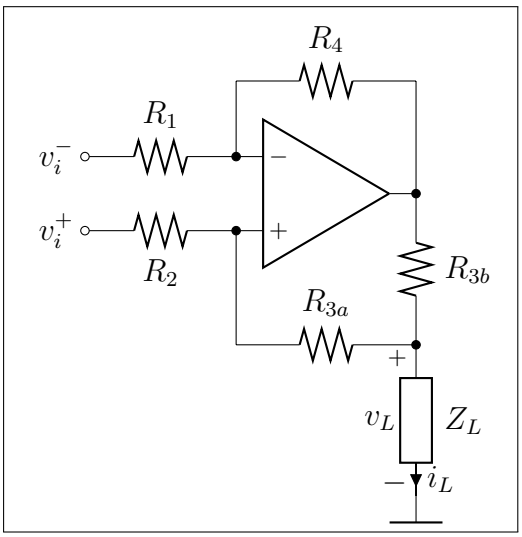


FIGURA 4.4. Diagrama circuital amplificador de transconductancia.

Como ya sabemos, esta topología de amplificador de Transconductancia corresponde a la fuente de corriente de Howland mejorada descrita y analizada en el capítulo 3.

Las especificaciones a cumplir por esta etapa son las siguientes:

- $1/2.21\,\mathrm{S}$ y $1/22.1\,\mathrm{S}$ de transconductancia DC.
- $1\,\mathrm{A}$ y $100\,\mathrm{mA}$ de corriente a la salida para los amplificadores de imagen y registro, respectivamente.
- $10\,\mathrm{MHz}$ y $100\,\mathrm{MHz}$ de ancho de banda para los amplificadores de transconductancia de imagen y registro, respectivamente.
- $100\,\mathrm{mArms}$ de corriente efectiva a la salida.
- $25\,\mathrm{V/\mu s}$ y $250\,\mathrm{V/\mu s}$ de slew rate para los amplificadores de imagen y registro, respectivamente.
- Entrada diferencial.
- Salida riel a riel con una excursión de voltaje de $0\,\mathrm{V}$ a $14\,\mathrm{V}$.
- Impedancia de salida máxima.
- Bajo costo.

Los amplificadores operacionales escogidos fueron el “OPA564” para los amplificadores de imagen y el “THS4226” para los amplificadores de registro. La tabla 4.7 muestra las principales características de estos amplificadores operacionales [*OPA564 — Precision Amplifier — Operational Amplifier (Op Amp) — Description & parametrics* (s.f.)] [*THS4222 — High Speed Amplifier (>=50MHz) — Operational Amplifier (Op Amp) — Description & parametrics* (s.f.)].

TABLA 4.7. Características amplificadores operacionales etapa de transconductancia.

	OPA564	**THS4226**
GBW	$17\,\mathrm{MHz}$	$230\,\mathrm{MHz}$
$I_{\max}$	$1.5\,\mathrm{A}$	$100\,\mathrm{mA}$
Alimentación	$7\,\mathrm{V}$ - $24\,\mathrm{V}$	$3\,\mathrm{V}$ - $15\,\mathrm{V}$
Slew Rate	$40\,\mathrm{V/\mu s}$	$975\,\mathrm{V/\mu s}$
Riel a Riel	No	Si

Tal y como se muestra en la tabla 4.7, el amplificador de imagen no cumple con los requisitos de producto-ganancia ancho de banda y condición de riel a riel expuestas en 4.2.3. La razón por la cual este amplificador fue finalmente utilizado es debido a que en primer lugar, casi no existen posibilidades de amplificadores operacionales que cumplan con tal corriente a la salida a bajo costo. En segundo lugar, tal y como se especifica en la hoja de datos del amplificador, éste alcanza los rieles cuando la corriente a la salida tiende a cero, situación que se cumple cuando los capacitores de las fases de reloj del detector CCD alcanzan su máximo voltaje.

La tabla 4.8 muestra la descripción de componentes utilizados basados en el diagrama de la figura 4.4.

TABLA 4.8. Descripción de componentes etapa de transconductancia.

	OPA564	**THS4226**
R_1	$270\,\Omega$, $\pm 0.1\ \%$	$270\,\Omega$, $\pm 0.1\ \%$
R_2	$R_{3a} + R_{3b}$	$R_{3a} + R_{3b}$
R_{3a}	$270\,\Omega$, $\pm 0.1\ \%$	$270\,\Omega$, $\pm 0.1\ \%$
R_{3b}	$2.21\,\Omega$, $\pm 0.1\ \%$	$22.1\,\Omega$, $\pm 0.1\ \%$
R_4	$270\,\Omega$, $\pm 0.1\ \%$	$270\,\Omega$, $\pm 0.1\ \%$

Tal y como se aprecia en la tabla 4.8, los resistores utilizados son de precisión. Recordemos que la impedancia de salida de la fuente de corriente de Howland mejorada depende en gran medida de este parámetro.

4.2.5 Compensación

Tal y como se analizó en la sección 3.3.3, la inductancia parásita de la carga, provocada principalmente por el largo de la línea, disminuye el margen de fase de la IHCS. La solución propuesta consiste en añadir resistencias en serie con la carga para contraarrestar este efecto.

La figura 4.5 nos muestra la configuración de los compensadores.

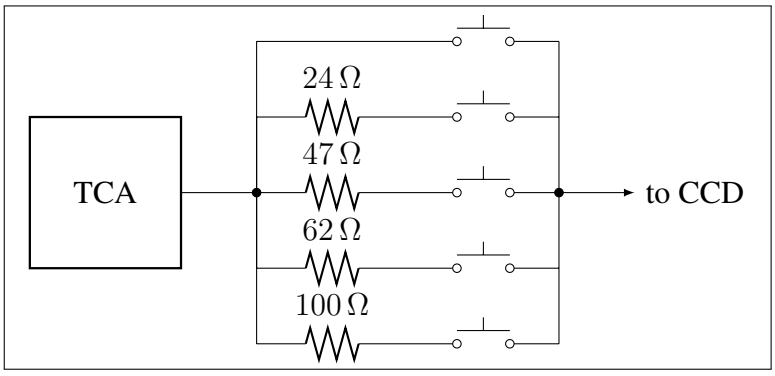


FIGURA 4.5. Diagrama circuital etapa de compensación.

Tal y como muestra la figura 4.5, los resistores de compensación pueden ser escogidos mediante jumpers.

4.2.6 Simulación

Para simular y verificar el diseño se utilizó el sofware LTspice. A continuación se muestran los resultados obtenidos.

a) Etapa de Transimpedancia

Para simular la etapa de transimpedancia se utilizó una fuente de corriente de primer orden con un tiempo de subida de $1\,\mathrm{ns}$ y una amplitud de $25\,\mathrm{mA}$. La figura 4.6 muestra la forma de onda a la entrada y a la salida del amplificador de transimpedancia.

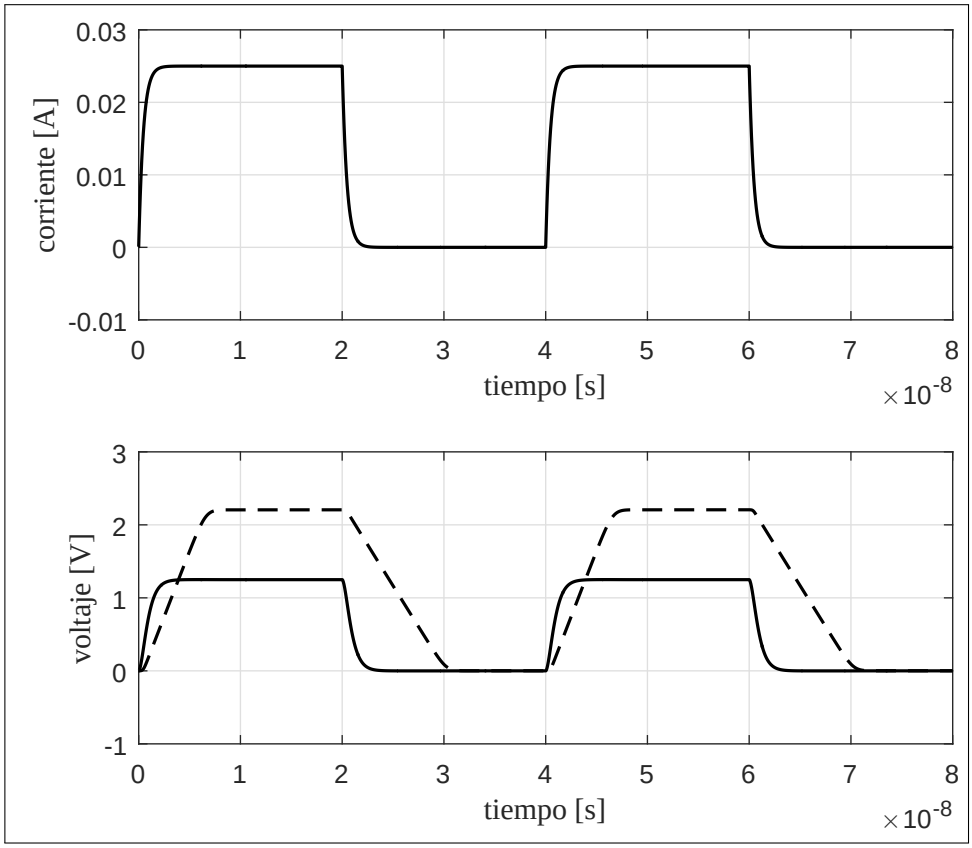


FIGURA 4.6. Formas de onda a la entrada y a la salida del amplificador de transimpedancia. La corriente proveniente del DAC se muestra en la imagen superior, mientras que el voltaje de entrada (línea continua) y el de salida (línea discontinua) se muestran en la parte inferior.

En la figura 4.6 podemos ver que la salida de voltaje a $25\,\mathrm{MHz}$ se encuentra limitada en slew rate dadas las características dinámicas del amplificador operacional utilizado. En la simulación el slew rate de subida corresponde a $360\,\mathrm{V}/\mu\mathrm{s}$ mientras que el de bajada sólo alcanza los $220\,\mathrm{V}/\mu\mathrm{s}$.

b) Etapa de Transconductancia

Para simular el comportamiento de la etapa de transconductancia se utilizó una carga resistivo-capacitiva en serie. Las figuras 4.7 y 4.8 muestran los resultados de simulación.

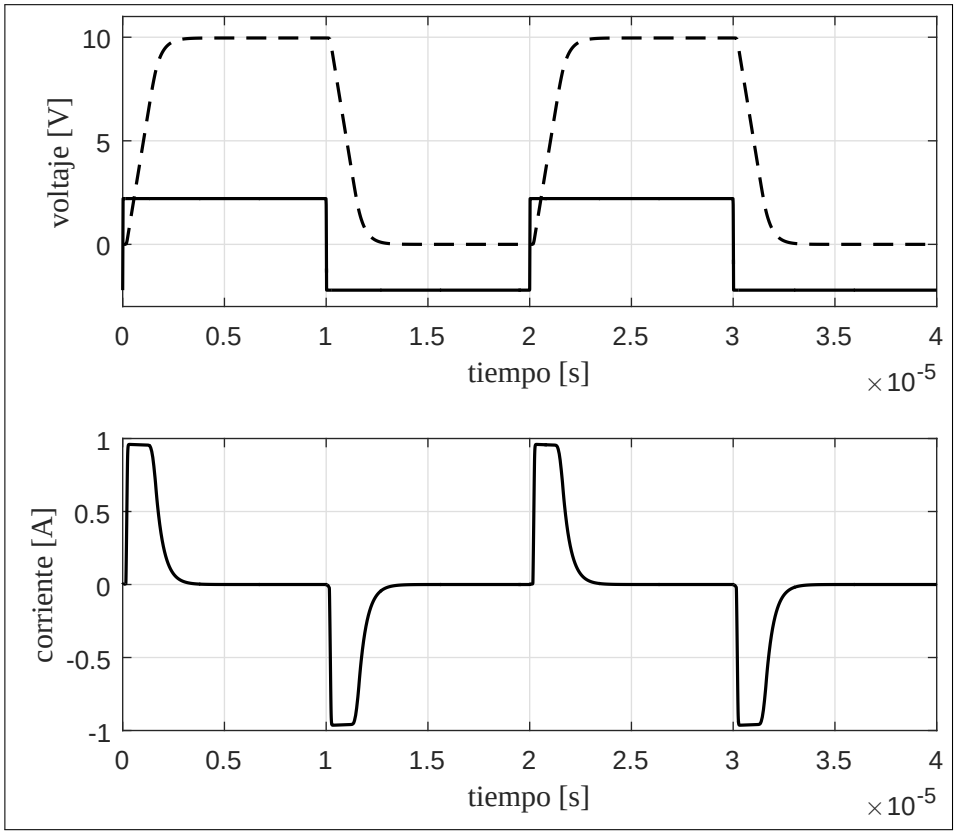


FIGURA 4.7. Formas de onda a la entrada y a la salida del amplificador de transconductancia de imagen frente a una carga resistivo-capacitiva serie de $0.1\,\Omega$ y $160\,\mathrm{nF}$. En la parte superior se muestra el voltaje diferencial de entrada (línea continua) y el voltaje en la carga (línea discontinua), mientras que en la parte inferior se muestra la corriente de la carga.

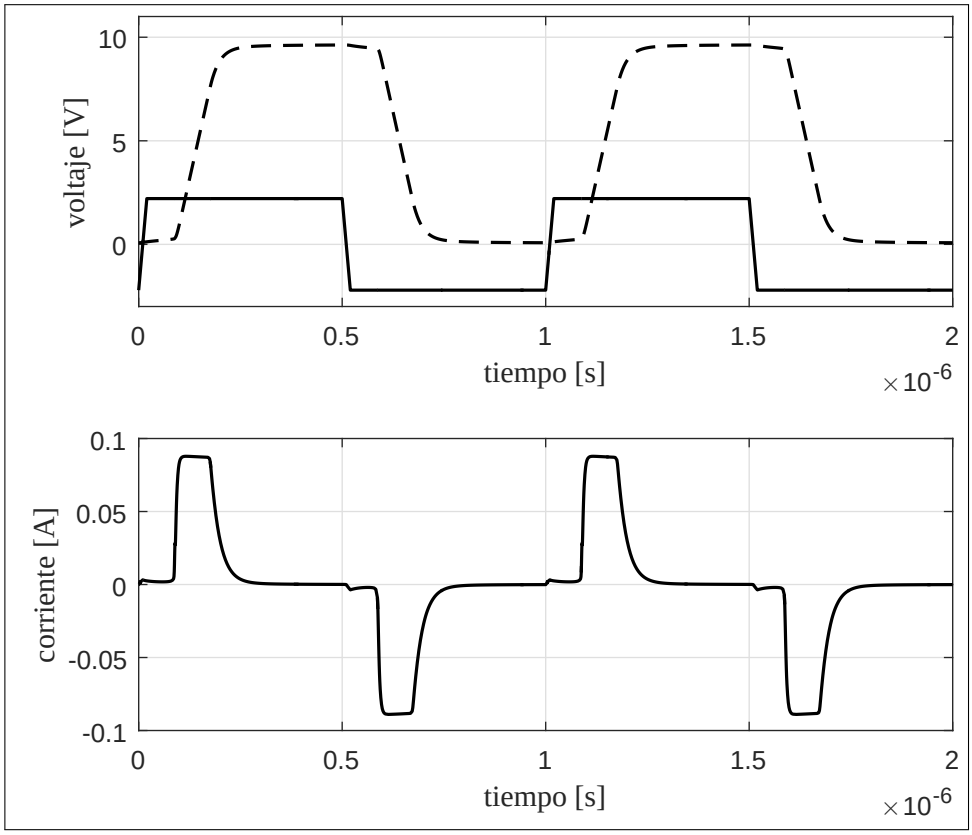


FIGURA 4.8. Formas de onda a la entrada y a la salida del amplificador de transconductancia de registro frente a una carga resistivo-capacitiva serie de $0.1\,\Omega$ y $1\,\mathrm{nF}$. En la parte superior se muestra el voltaje diferencial de entrada (línea continua) y el voltaje en la carga (linea discontinua), mientras que en la parte inferior se muestra la corriente de la carga.

En las figuras 4.7 y 4.8 podemos ver que la corriente de carga alcanza aproximadamente $1\,\mathrm{A}$ y $100\,\mathrm{mA}$ para los amplificadores de imagen y registro, respectivamente.

Para analizar el establecimiento de la corriente, se simuló la respuesta a un escalón utilizando una carga resistiva de $1\,\Omega$. Las figuras 4.9 y 4.10 muestran los resultados de la simulación.

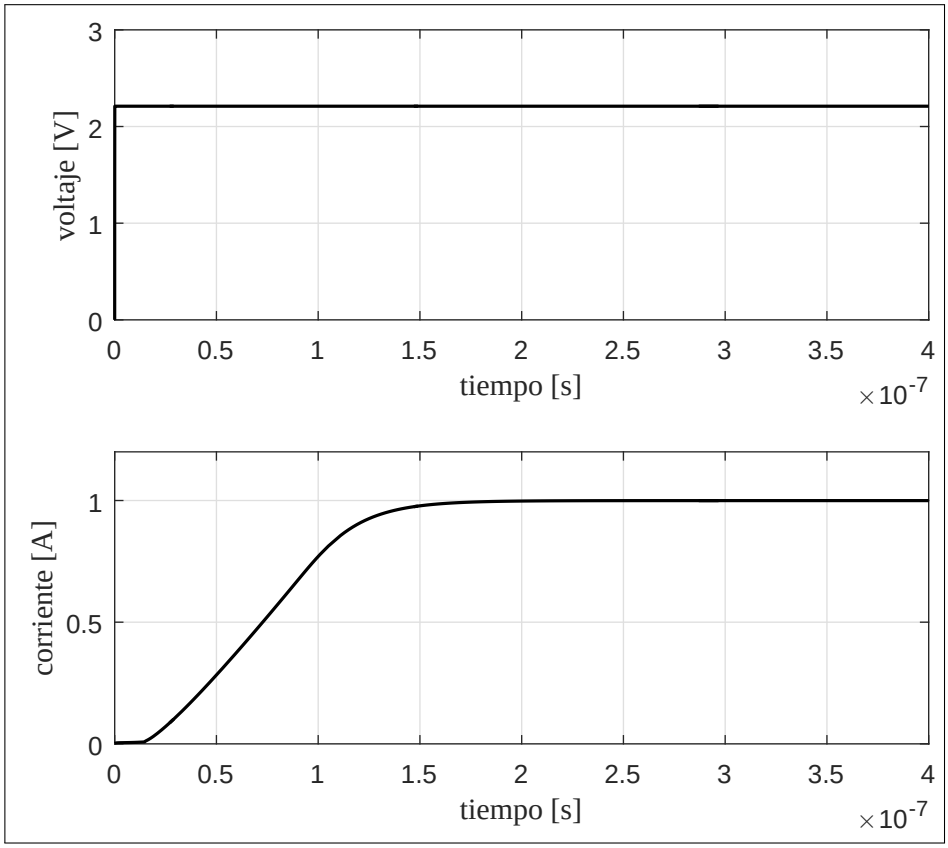


FIGURA 4.9. Respuesta al escalón del amplificador de transconductancia de imagen utilizando una carga resistiva de 1 Ω. En la parte superior se muestra el voltaje diferencial de entrada mientras que en la parte inferior se muestra la corriente a la salida.

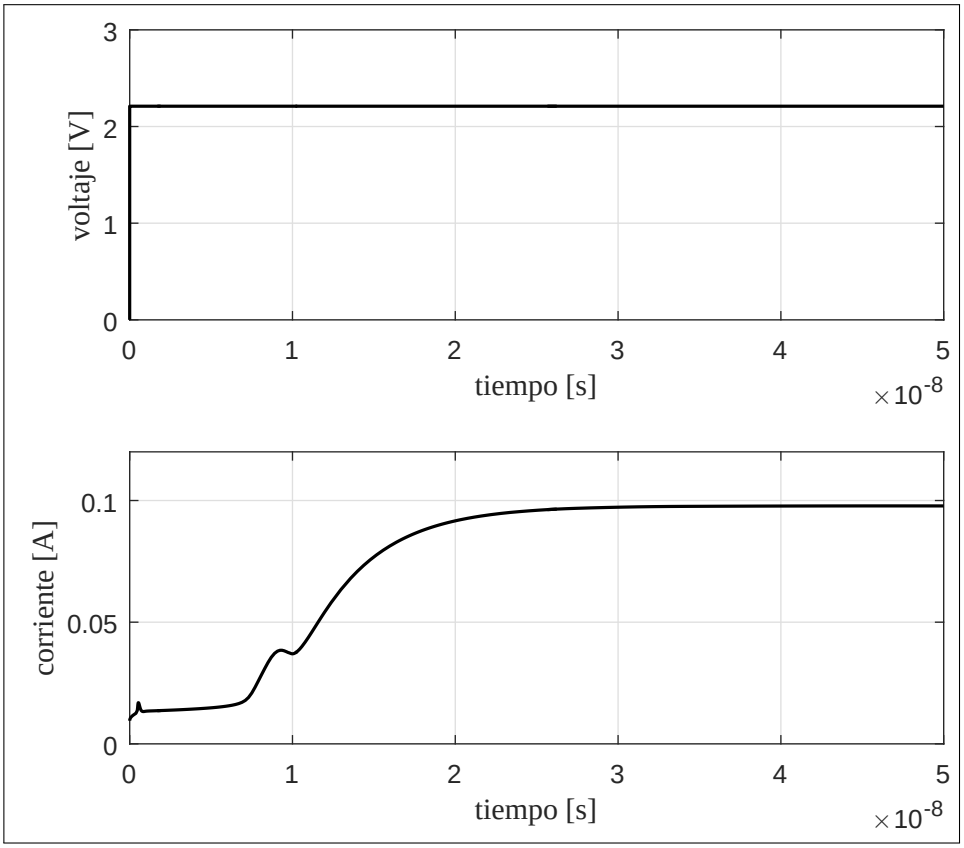


FIGURA 4.10. Respuesta al escalón del amplificador de transconductancia de registro utilizando una carga resistiva de $1\,\Omega$. En la parte superior se muestra el voltaje diferencial de entrada mientras que en la parte inferior se muestra la corriente a la salida.

En las figuras 4.9 y 4.10 podemos ver claramente que el amplificador de registro posee un tiempo de establecimiento menor que el amplificador de imagen. Por otro lado, el amplificador de registro posee un establecimiento no monotónico en la corriente de salida. Esto puede deberse a errores en macromodelo utilizado para simular.

c) Compensación

Por último, para verificar el efecto de los resistores de compensación se utilizó un inductor en serie con la carga de $10\,\mu\text{H}$. La figura 4.11 muestra los resultados simulados.

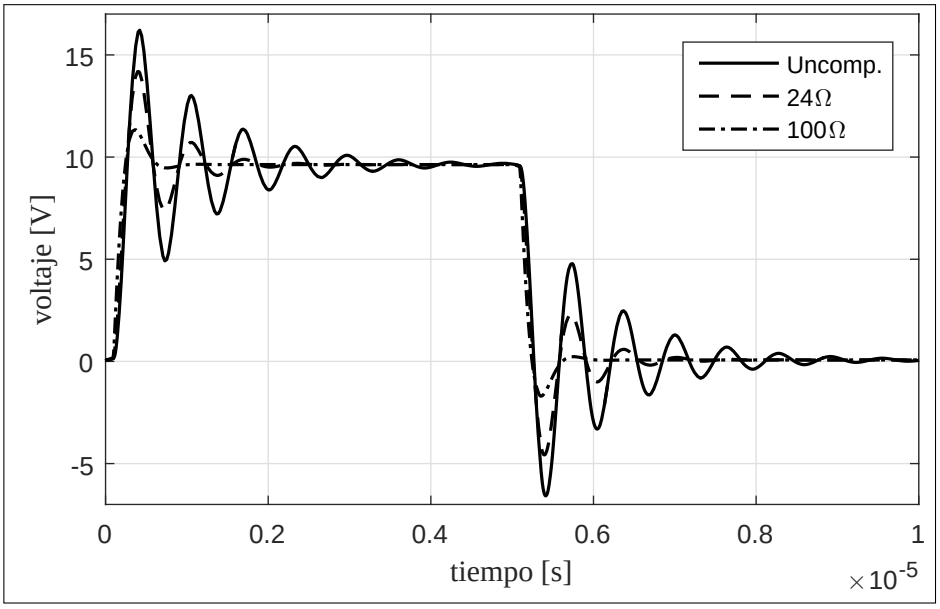


FIGURA 4.11. Respuesta del amplificador de registro frente a una carga serie inductivo-capacitivo-resistiva de $10\,\mu\mathrm{H}$, $1\,\mathrm{nF}$ y $0.1\,\Omega$. En la figura se muestra el voltaje a la salida sin compensar (linea continua), con resistor de compensación de $24\,\Omega$ (linea discontinua) y con resistor de compensación de $100\,\Omega$ (linea de puntos y trazos).

En la figura 4.11 podemos ver que la sobreoscilación disminuye considerablemente, y por ello aumenta el margen de fase, al añadir los resistores de compensación, lo que valida los resultados desarrollados en la sección 3.3.3.

4.3 Hardware

Esta sección tiene por objetivo describir el harware utilizado para conformar el sistema de generación de señales de reloj por corriente. La figura 4.12 nos muestra el diagrama de bloques.

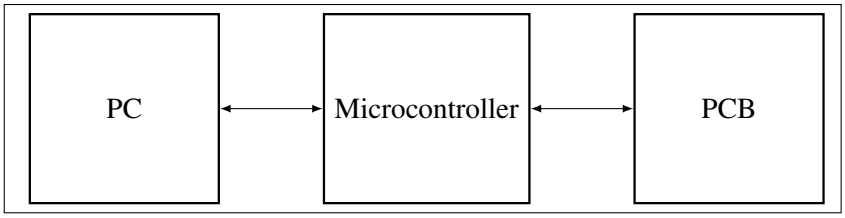


FIGURA 4.12. Diagrama general hardware.

A continuación, se expondrá en detalle cada uno de los bloques que conforman el hardware del sistema.

4.3.1 PCB

Con la finalidad de contar con un circuito personalizado se tomó la decisión de diseñar una PCB. Esta tarjeta posee todos los bloques circuitales necesarios que conforman el driver de señales de reloj. Las figuras 4.13 y 4.14 muestran la PCB operativa y la descripción de las secciones circuitales.

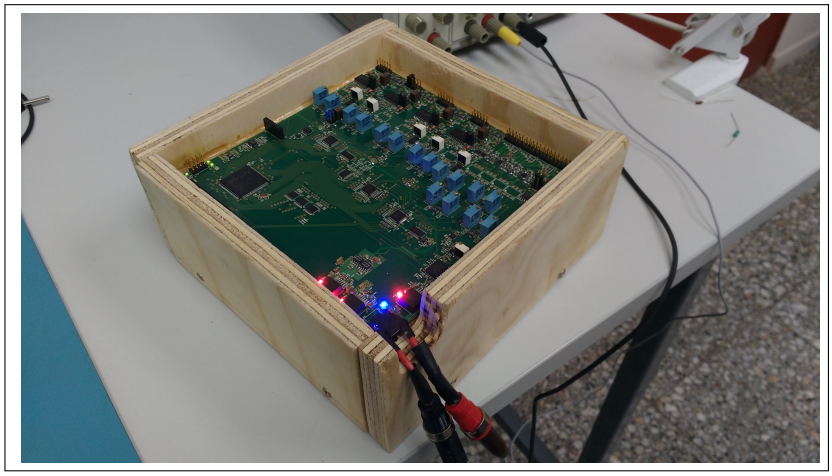


FIGURA 4.13. Fotografía PCB implementada.

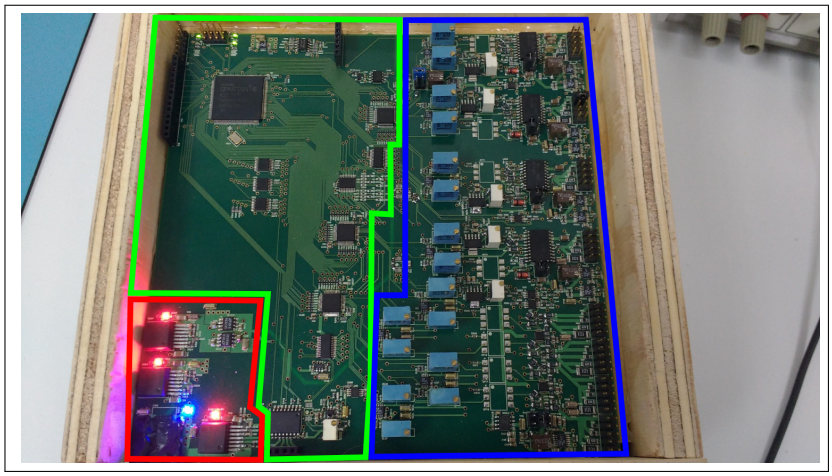


FIGURA 4.14. Secciones circuitales de la PCB implementada. En rojo se muestra la sección de alimentación, en verde la sección digital, mientras que en azul la sección analógica del prototipo.

A continuación, se describirán en forma general cada uno de los bloques que conforman la PCB cuyo diseño circuital fue tratado en la sección 4.2.

a) Alimentación

El bloque de alimentación se compone en su totalidad de fuentes de conmutación y reguladores lineales (fijos y regulables). La figura 4.15 muestra un diagrama simplificado del bloque circuital de alimentación.

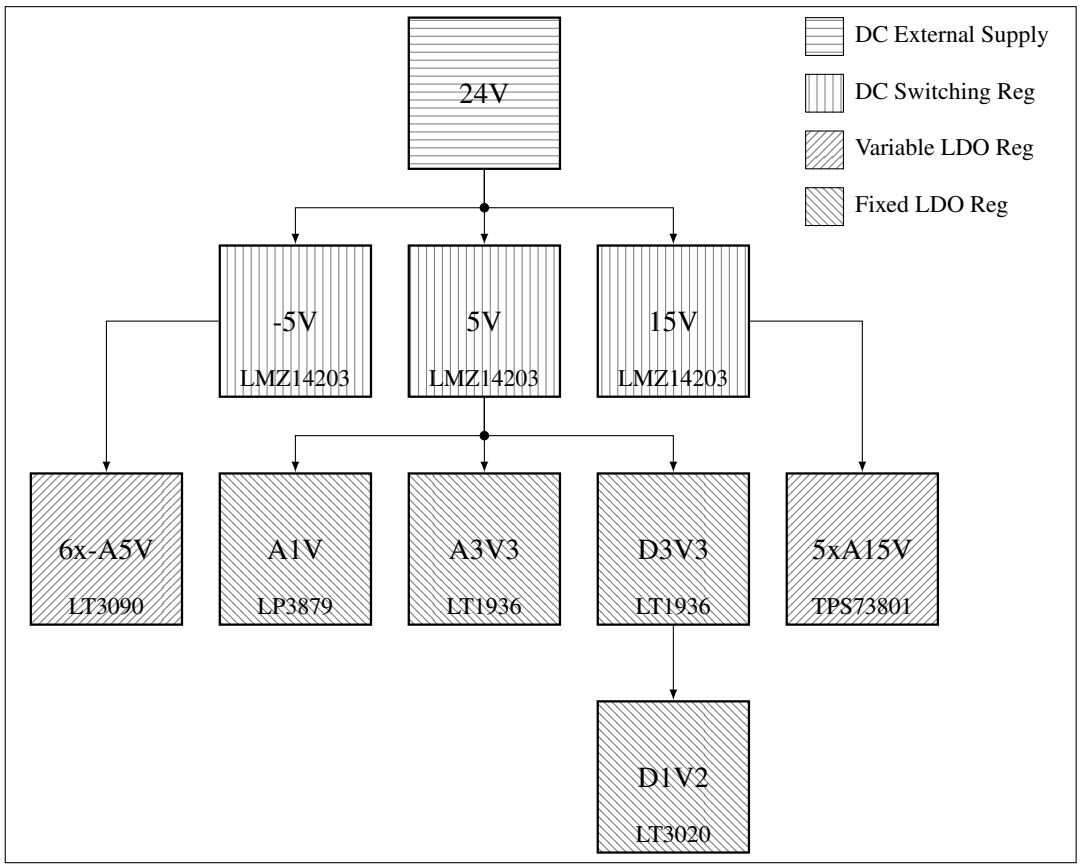


FIGURA 4.15. Diagrama circuital bloque de alimentación.

El bloque de alimentación comienza con una fuente de voltaje externa de 24V. Luego ésta alimenta una etapa de reguladores de conmutación que producen los voltajes necesarios para alimentar reguladores lineales.

Para efectos de disminución de ruido, se decidió separar los elementos circuitales de las etapas digitales y analógicas. Por ello, los reguladores de voltaje lineales “D3V3” y “D1V2” son los encargados de alimentar el bloque digital y el módulo principal del FPGA, respectivamente.

b) Bloque Digital

El bloque digital se compone de un FPGA, DACs de corriente, Shift-Registers, ADCs y un driver de voltaje para las señales de RESET del CCD. La figura 4.16 muestra un diagrama simplificado del bloque circuital digital.

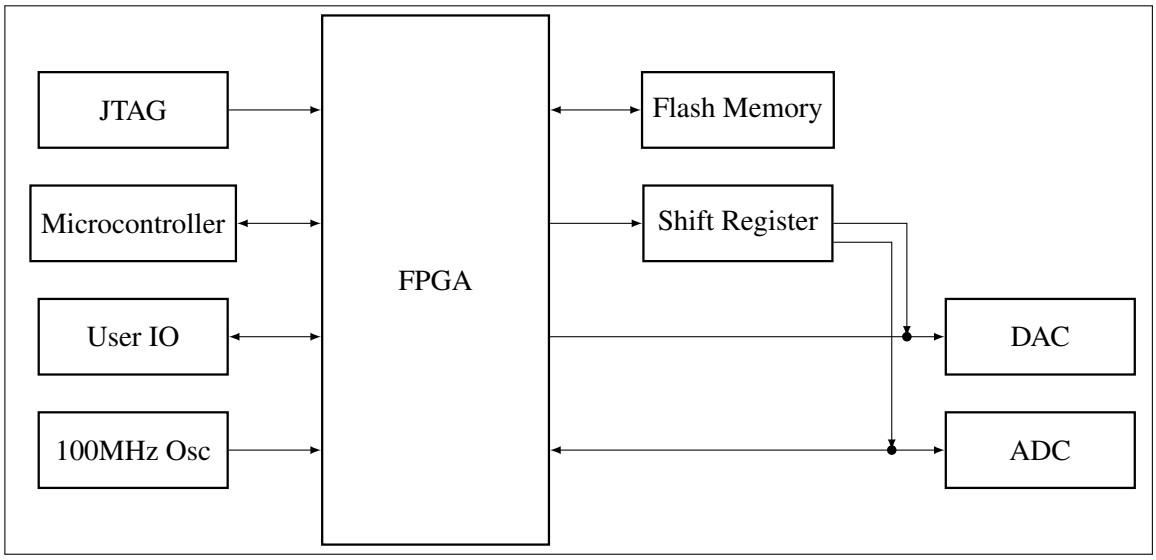


FIGURA 4.16. Diagrama circuital bloque digital.

En términos generales, el FPGA de aproximadamente 120 salidas de propósito general cumple la función de máquina de estados principal. Mediante el protocolo de comunicación “JTAG” es posible programar tanto el FPGA como la memoria flash que permite que el sistema posea memoria no volátil. Además, el FPGA puede conectarse con el microcontrolador mediante un simple protocolo SPI. Por otro lado, el FPGA posee tanto entradas como salidas que permiten definir las señales de reloj del CCD, y un cristal externo de

100 MHz que define la frecuencia de operación de la máquina de estados principal.

Las salidas del FPGA se agrupan en tres bloques: registro serial, DACs y ADCs. La salida de registro serial permite añadir salidas extras para señales que no cambian constantemente en el tiempo. El bloque de DACs, compuesto por 8 conversores digital-análogos de 8 bits permite conformar las señales de corriente diferenciales que posteriormente, serán transformadas en señales de voltaje diferenciales en el bloque analógico. Para finalizar, el FPGA posee la capacidad de conectarse mediante SPI con dos ADCs de 10 bits que monitorean los voltajes de todo el sistema.

En resumen, el bloque digital posee la capacidad de manejar hasta 8 amplificadores del bloque analógico.

c) Bloque Analógico

El bloque analógico tiene por objetivo generar las señales de corriente diferenciales provenientes de las salidas de los DACs. La figura 4.17 muestra un diagrama simplificado del bloque circuital analógico.

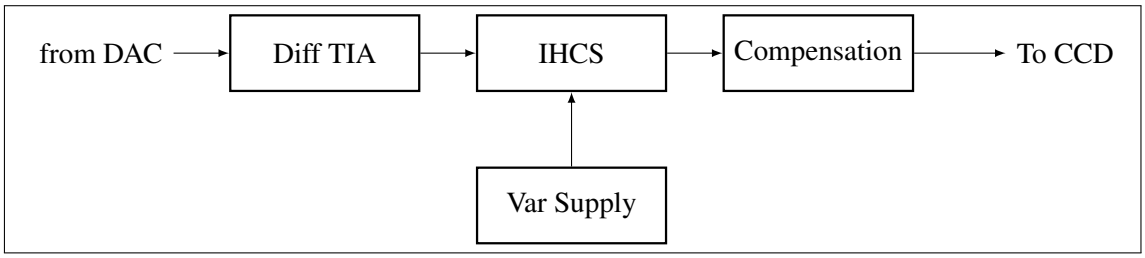


FIGURA 4.17. Diagrama circuital bloque digital.

La figura 4.17 muestra las etapas de uno de los ocho amplificadores. En una primera instancia, las señales diferenciales de corrientes provienientes de los DACs son transformadas a señales de voltaje diferenciales mediante un amplificador de transimpedancia diferencial (“Diff TIA”). Luego estas señales de voltaje diferenciales son utilizadas como

entradas de las fuentes de corriente de Howland mejorada (“IHCS”) las cuales a su vez poseen una alimentación de voltaje regulable proveniente del bloque de alimentación (“Var Supply”) y hacen las veces de amplificador de transconductancia. Recordemos que existen dos diseños de amplificador de transconductancia, los cuales son para las señales de reloj de imagen y de registro. Estos dos tipos de amplificadores poseen parámetros de configuración distintos. Continuando con las etapas del bloque analógico, la señal de corriente a la salida de la fuente de corriente de Howland mejorada puede ser compensada para atenuar los efectos parásitos de la inductancia de línea. Finalmente, la señal de corriente generada y compensada puede dirigirse a la fase de reloj del CCD.

4.3.2 Microcontrolador

El microcontrolador cumple la función de canal de comunicación entre el PC y los registros externamente configurables del FPGA. Debido a los múltiples puertos de salida serial, conexión USB, y a su disponibilidad, se escogió la tarjeta de desarrollo “MSP430F5529 LaunchPad”. La figura 4.18 muestra una fotografía de la tarjeta.

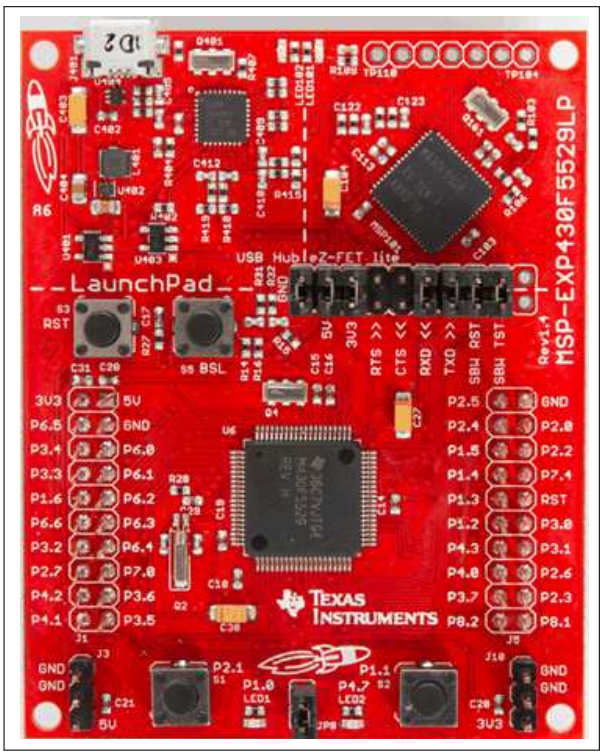


FIGURA 4.18. Tarjeta de desarrollo “MSP430F5529 LaunchPad”.

Las características principales de la tarjeta de desarrollo utilizadas son:

- Operación a 3.3V (compatible con el FPGA).
- Velocidad de operación de hasta 25MHz.
- 4 periféricos de interfaz serial.
- 128 kB de memoria Flash, 8 kB memoria RAM.
- Dos cristales: XT1 a 32kHz y XT2 a 4MHz con una precisión de ± 2500 ppm.

4.4 Software

4.4.1 FPGA

El código del FPGA Spartan6 fue implementado en Verilog con la ayuda del software PlanAhead 14v7. La figura 4.19 describe el funcionamiento general del módulo principal[2].

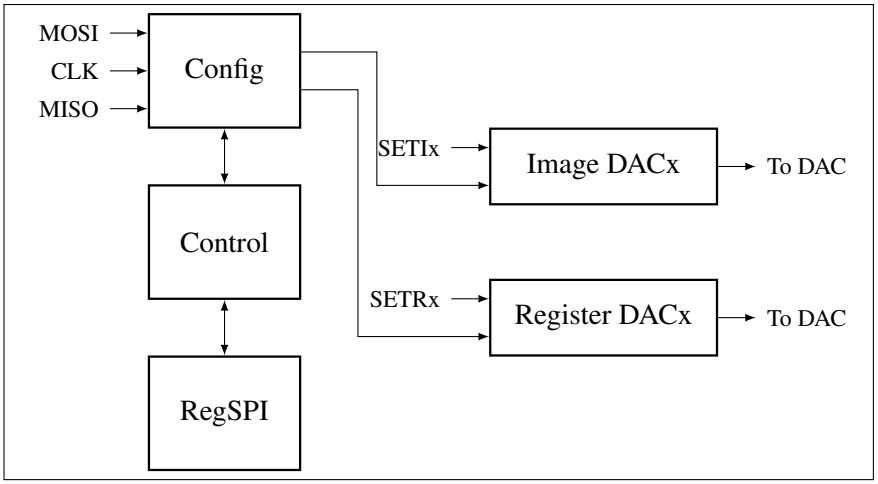


FIGURA 4.19. Diagrama módulo principal FPGA.

El funcionamiento descrito en la figura 4.19 nos muestra los cinco módulos generales en el funcionamiento del FPGA. El bloque principal o de "Control" es el encargado de la máquina de estados principal. El módulo "RegSPI" es el encargado de las comunicaciones entre el FPGA y los componentes del PCB, tales como los registros seriales y los ADC. El módulo "Config" es el encargado de la comunicación del FPGA con el exterior mediante un simple protocolo de SPI. Finalmente, los dos últimos bloques de DACs de imagen y registro son los que conforman la señal para cada amplificador de salida.

A continuación se explica en detalle el funcionamiento de cada módulo.

[2]Los códigos Verilog implementados se muestran en el anexo B.1

a) Módulo REGSPI

Tal y como se dijo con anterioridad, este módulo es el encargado de las comunicaciones internas del sistema. El diagrama funcional de este módulo se muestra en la figura 4.20.

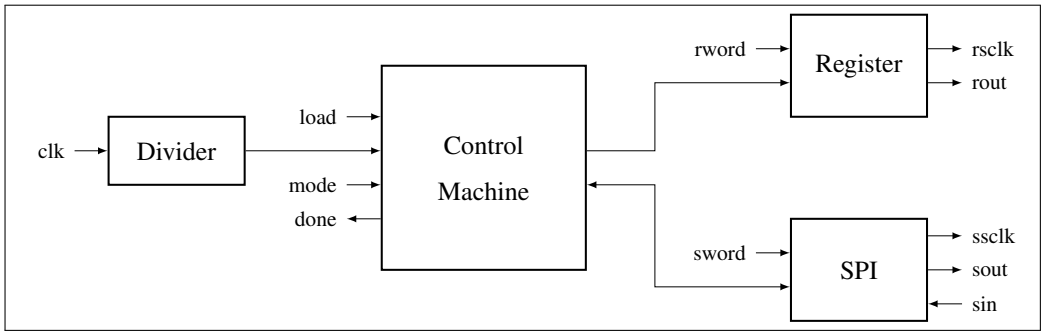


FIGURA 4.20. Diagrama funcional módulo “RegSPI”.

En un primer lugar, la señal de reloj es dividida para cumplir con la tasa máxima de escritura y lectura. Luego podemos ver la máquina de control del módulo. Ésta se acciona mediante la señal “load”, posee una entrada de selección de modo “mode” y regresa una señal de proceso finalizado “done”. Por último, la máquina de control controla dos submódulos de registro y SPI, los cuales se comunican con los componentes internos del sistema.

b) Módulos de DAC

En la figura 4.19 los módulos de DAC corresponden a “Image DACx” y “Register DACx”. Existen cuatro módulos de imagen y registro, cada uno indicado por “x”. El diagrama funcional de este módulo se muestra en la figura 4.21.

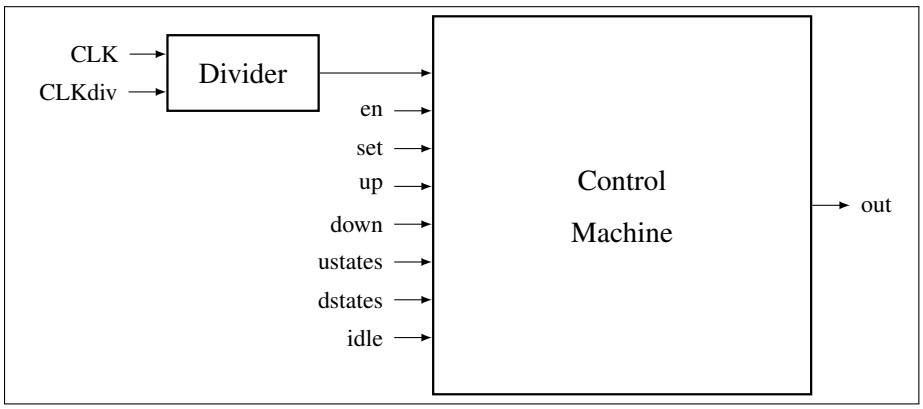


FIGURA 4.21. Diagrama Funcional Módulos DAC.

Las señales “clkdiv”, “en”, “up”, “down”, “ustates”, “dstates” e “idle” son entradas provenientes del módulo “Config”, mientras que la señal “set” gatilla el flanco de subida o bajada de la señal de salida “out” y corresponde a una entrada externa del FPGA. Una descripción más detallada de las entradas y salidas de este módulo se muestra en la tabla 4.9.

TABLA 4.9. Descripción de entradas y salidas de los módulos DAC.

Señal	**Tipo**	**Bits**	**Descripción**
clk	Entrada	1	Clock de entrada del oscilador (100 MHz)
clkdiv	Entrada	16	Divisor del clock de entrada
en	Entrada	1	Enable del módulo DACx
set	Entrada	1	Gatillador Salida DAC
up	Entrada	80	Bus de 10 bytes con la descripción de la forma de onda de salida en subida
down	Entrada	80	Bus de 10 bytes con la descripción de la forma de onda de salida en bajada
ustates	Entrada	8	Número de estados de subida
dstates	Entrada	8	Número de estados de bajada
idle	Entrada	8	Estado en la salida del módulo cuando EN=1'b0
out	Salida	8	Salida del módulo

c) Módulo Config

Este módulo es el encargado de inicializar variables de configuración y cumple la función de enlace externo entre el FPGA y el microcontrolador. El diagrama funcional de este módulo se muestra en la figura 4.22.

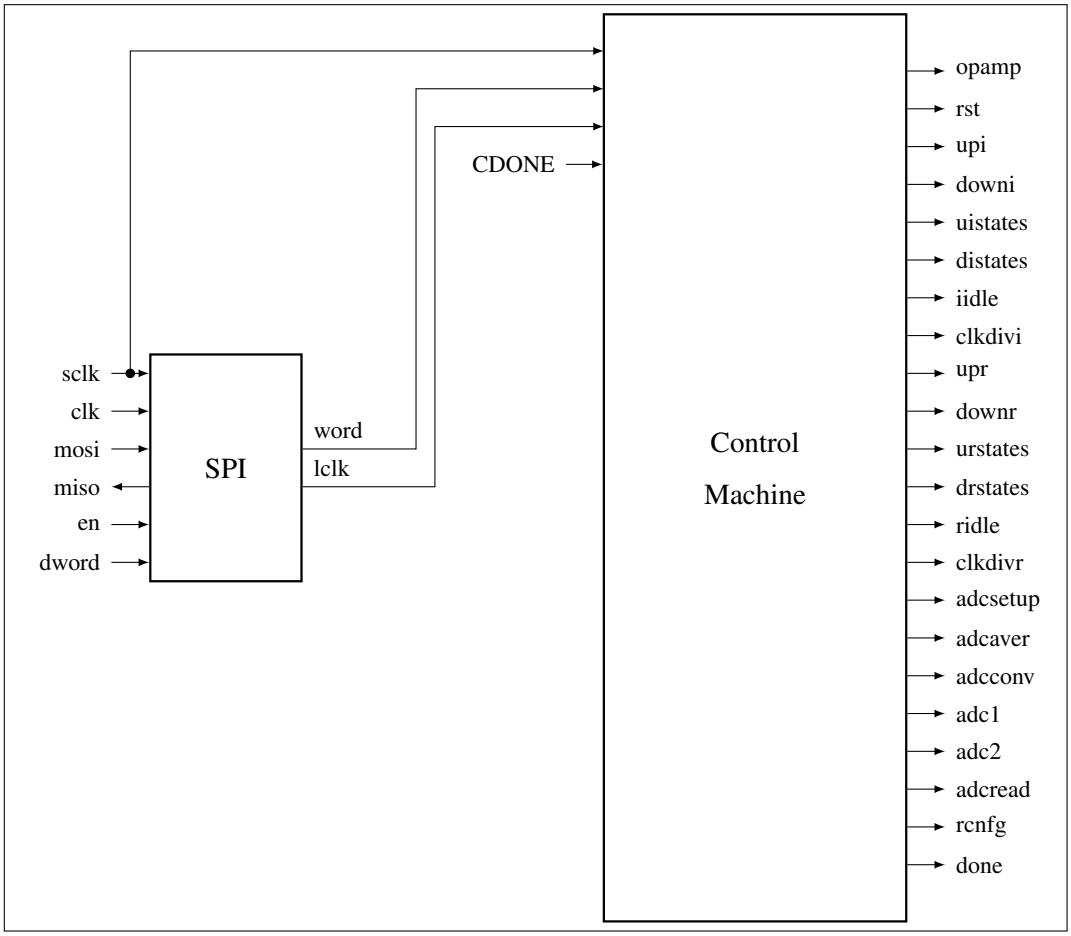


FIGURA 4.22. Diagrama funcional del módulo Config.

Las tablas 4.10, 4.11 y 4.12 nos muestran la descripción de las entradas y salidas de este módulo.

TABLA 4.10. Descripción de las entradas del módulo Config.

Señal	**Tipo**	**Bits**	**Descripción**
clk	Entrada	1	Clock de Entrada del Oscilador (100 MHz)
sclk	Entrada	8	Señal de reloj SPI master
mosi	Entrada	1	SPI master-output slave-input
miso	Salida	1	SPI master-input slave-output
en	Entrada	1	Señal de enable para cargar los datos de dword en MISO
dword	Entrada	8	Señal de 15 bits con los datos de lectura del ADC
word	Interna	8	Instrucción a ejecutar en la máquina de estados
lclk	Interna	1	Señal interna que carga la instrucción en la máquina de estados
cdone	Entrada	1	Señal de proceso finalizado proveniente del módulo CONTROL

TABLA 4.11. Descripción de las salidas de los módulos Config.

Señal	**Tipo**	**Bits**	**Descripción**
opamp	Salida	8	Registro de amplificadores de imagen y registro
rst	Salida	1	Registro del amplificador de reset
upi	Salida	80	Bus de 10 bytes con UP de los amplificadores de imagen
downi	Salida	80	Bus de 10 bytes con DOWN de los amplificadores de imagen
uistates	Salida	8	ustates de los amplificadores de imagen
distates	Salida	8	dstates de los amplificadores de imagen
iidle	Salida	8	idle de los amplificadores de imagen
clkdivi	Salida	16	clkdiv de los amplificadores de imagen
upr	Salida	80	up de los amplificadores de registro
downr	Salida	80	down de los amplificadores de registro
urstates	Salida	8	ustates de los amplificadores de registro

TABLA 4.12. Descripción de las salidas del módulo Config.

Señal	Tipo	Bits	Descripción
drstates	Salida	8	DSTATES de los amplificadores de registro
ridle	Salida	8	IDLE de los amplificadores de registro
clkdivr	Salida	16	CLKdiv de los amplificadores de registro
adcsetup	Salida	8	Registro de configuración de los ADC
adcaver	Salida	8	Registro de promediado de los ADC
adcconv	Salida	8	Registro de conversión de los ADC
adc1	Salida	1	Señal de lectura del ADC1
adc2	Salida	1	Señal de lectura del ADC2
adcread	Salida	1	Señal de espera en el proceso de lectura los ADC
rcnfg	Salida	1	Señal de reconfiguración del FPGA
done	Salida	1	Señal de terminado

A excepción de las señales “adcread” y “done”, todas las señales de salida de este módulo son configurables por el usuario mediante el puerto SPI.

El puerto SPI implementado en el FPGA es de tres puertos con polaridad de reloj nula y polaridad de fase uno. Las instrucciones que puede recibir este módulo para configurar las salidas se muestran en la tabla 4.13.

TABLA 4.13. Intrucciones externas del módulo Config.

Intrucción	N Bytes	Descripción
opamp	2	Instrucción para configurar el registro de amplificadores de imagen y registro
rst	2	Instrucción para configurar el registro de amplificador de reset
daci	25	Instrucción para configurar la forma de señal de los amplificadores de imagen
dacr	25	Instrucción para configurar la forma de señal de los amplificadores de registro
adcconfig	3	Instrucción para configurar los registros de configuración y promediado de los ADC
reconfig	2	Instrucción para reconfigurar el FPGA frente cambios en los registros de opamp, rst y adcconfig
adcmeasx	4	Instrucción para leer el ADCx

A continuación se describe el funcionamiento de cada una de estas instrucciones:

- **OPAMP:** Esta instrucción de dos bytes permite configurar el registro de amplificadores de imagen y registro. Esta instrucción es de la forma:

```
opamp = {8'h01 , rop[3:0], iop[3:0]};
```
(4.4)

  donde "iop[i]" y "rop[i]" corresponden al bit de encendido (1'b1) o apagado (1'b0) del amplificador de imagen y registro "i", respectivamente. Por defecto, todos los amplificadores se encuentran desactivados al inicio.
- **RST:** Esta instrucción de dos bytes permite configurar el registro del amplificador de reset. Esta instrucción es de la forma:

```
rst = {8'h02, 7'hx, rst};
```
(4.5)

donde “rst” corresponde al bit de encendido (1’b1) o apagado (1’b0) del amplificador de reset. Por defecto, el amplificador de reset se encuentra desactivado en el inicio.

- **DACI y DACR:** Esta instrucción de 25 bytes permite configurar la forma de señal de los amplificadores de imagen (DACI) y registro (DACR). Esta instrucción es de la forma:

```
dacx = {dacx, clkdivx,xidle,downxstates,upxstates,downx,upx};
```
(4.6)

donde “dacx” corresponde a “8’h03” (Imagen) y “8’h04” (Registro).

- **ADCConfig:** Esta instrucción de 3 bytes permite configurar los registros de configuración y promediado de los ADC. La instrucción es de la forma:

```
adcconfig = {8'h5, adcsetup, adcaver};
```
(4.7)

donde “adcsetup” y “adcaver” corresponden a los registros de configuración y promediado de los ADCs descritos en su hoja de datos [*ADV7125 — datasheet and product info 330MHz Triple 8-Bit High Speed Video DAC — Analog Devices* (s.f.)]. Por defecto, los registros de los ADCs se encuentran configurados como adcsetup = 8’b01100100 y adcaver = 8’b00111100. El FPGA se encuentra programado para leer un puerto de entrada de los ADCs por instrucción. Aún no se encuentra configurado el modo de múltiples puertos.

- **RECONFIG:** Esta instrucción de dos bytes permite reconfigurar el FPGA y los ADCs tras ejecutar las instrucciones de cambios de registro “OPAMP”, “RST” o “ADCConfig”. La instrucción es de la forma:

```
reconfig = 16'h060a;
```
(4.8)

Una vez finalizada la configuración, el pin de salida del FPGA “done” será reestablecido en “1’b1”.

- **ADCMeasx:** Esta instrucción permite la lectura de una entrada del ADCx (ADC1 ó ADC2). La instrucción es de la forma:

```
adcmeasx = {adcmeasx, adcconv, adcr};
```
(4.9)

donde “adcmeasx” corresponde a “8’h07” (ADC1) y “8’h08” (ADC2), “adcconv” corresponde al registro de conversión especificado en la hoja de datos del ADC, y finalmente “adcr” corresponden a los dos bytes de lectura del resultado de la conversión.

Una vez escrito el byte ADCConv, es preciso esperar al pin de salida “done” del FPGA para proceder a leer los dos bytes del resultado de lectura de un puerto del ADCx. Esta instrucción permite la lectura de un sólo puerto de entrada del ADCx.

d) Módulo Control

Este módulo corresponde a la máquina de estados principal del FPGA. El diagrama funcional de este módulo se muestra en la figura 4.23.

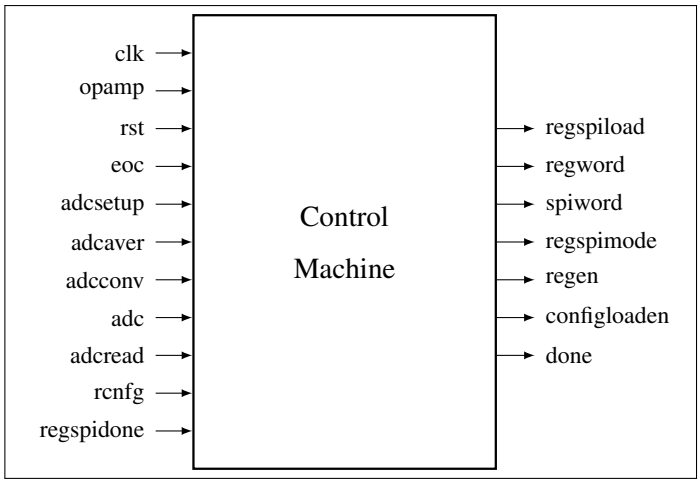


FIGURA 4.23. Diagrama funcional del módulo “Control”.

La tablas 4.14 nos muestran la descripción de las entradas y salidas de este módulo.

TABLA 4.14. Descripción de las entradas y salidas del módulo Control.

Señal	Tipo	Bits	Descripción
clk	Entrada	1	Clock de entrada del oscilador (100 MHz).
opamp	Entrada	8	Registro de amplificadores de imagen y registro
rst	Entrada	1	Registro de amplificadores de reset
eoc	Entrada	2	Señal de término de conversión de los ADC
adcsetup	Entrada	8	Registro de configuración de los ADC
adcaver	Entrada	8	Registro de promediado de los ADC
adcconv	Entrada	8	Registro de conversión de los ADC
adc	Entrada	2	Señal de lectura de los ADC
adcread	Entrada	1	Señal de espera en el proceso de lectura de los ADC
rcnfg	Entrada	1	Señal de reconfiguración del FPGA
regspidone	Entrada	1	Señal de finalizado proveniente del módulo RegSPI
regspiload	Salida	1	Señal de carga de datos e inicialización del módulo RegSPI
regword	Salida	24	Palabra de 3 bytes con la información del registro serial
spiword	Salida	8	Palabra de 1 byte con la información de salida del puerto SPI
regspimode	Salida	4	Modo del módulo RegSPI
regen	Salida	1	Enable de los registros seriales externos
configloaden	Salida	1	Enable del módulo CONFIG
done	Salida	1	Señal de proceso finalizado

4.4.2 Microcontrolador

Tal y como se mencionó con anterioridad, el microcontrolador es el encargado de hacer de enlace entre el FPGA y el PC. Para ello, se utilizaron dos periféricos de puerto serial del microcontrolador. El microcontrolador fue programado utilizando el software “Code

Composer Studio v6.1". La figura 4.24 nos muestra el esquema de periféricos utilizados por el microcontrolador.

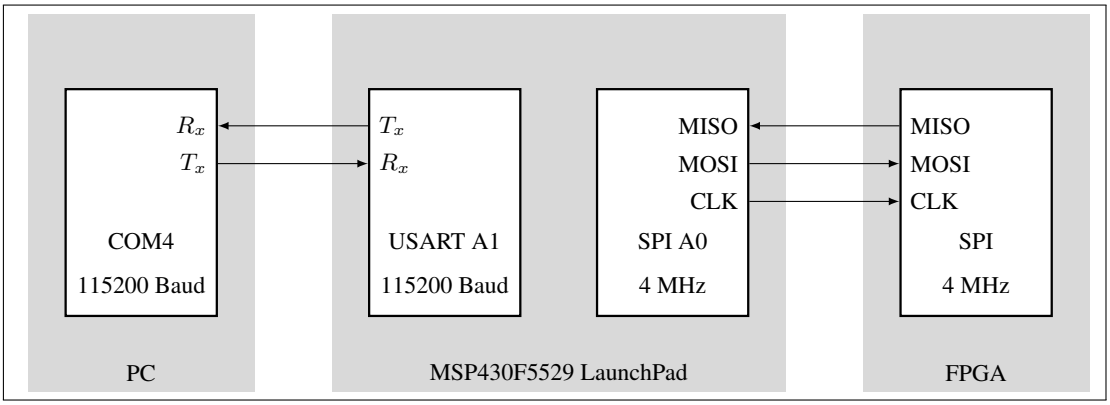


FIGURA 4.24. Diagrama de periféricos del microcontrolador "MSP430F5529 LaunchPad".

Como se aprecia en la figura 4.24, se utilizan los periféricos de comunicación serial "A0" (SPI) y "A1" (USART) para comunicarse con el FPGA y el PC, respectivamente. La tasa de símbolos para el caso de la comunicación USART fue de 115200 Baudios, mientras que la frecuencia del reloj en la comunicación SPI fue de 4 MHz.

El código utilizado para la programación del microcontrolador se muestra en el anexo B.2.

4.4.3 PC

El PC cumple la función de enviar las instrucciones de configuración del FPGA mediante el microcontrolador. Para ello, se diseñó una interfaz gráfica de usuario en Matlab R2014b. La figura 4.25 nos muestra la vista de la interfaz.

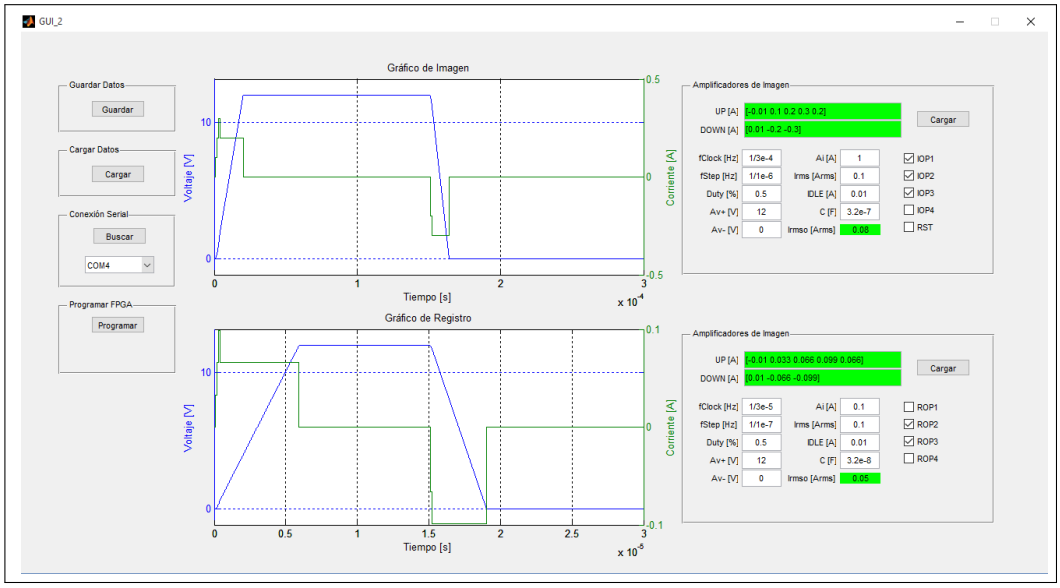


FIGURA 4.25. Interfaz gráfica de usuario diseñada en Matlab.

La interfaz posee las siguientes funcionalidades:

- Graficar la forma de onda de la señal de reloj sujeta a parámetros de señal y capacitancia de la carga.
- Verificar y corregir formas de onda en función de parámetros como corriente rms de salida máxima o excursión de corriente de salida.
- Definir los amplificadores de salida a utilizar en el FPGA (imagen y registro).
- Guardar y cargar parámetros de señal y carga.
- Buscar y seleccionar puerto de comunicación serial.
- Programar el FPGA con los datos de usuario ingresados.

5. RESULTADOS EXPERIMENTALES

Este capítulo tiene por objetivo exhibir, analizar y discutir los resultados experimentales obtenidos con el fin de validar el funcionamiento del driver diseñado y construido.

Para validar y verificar el correcto funcionamiento del prototipo sólo fue posible realizar pruebas con capacitores que emulan el comportamiento de una fase del detector CCD. Esto debido a que los detectores y cámaras a las que tendremos acceso sólo estarán disponibles para pruebas a mediados del 2016.

5.1 Configuración Experimental

Para realizar los experimentos se utilizaron los siguientes instrumentos, softwares y componentes:

- PC Samsung RF511.
- Software Matlab 2014b.
- Tarjeta de desarrollo microcontrolador Texas Instrument, MSP430F5529b.
- Prototipo clock driver CCD.
- Osciloscopio Agilent Technologies, MSO-X 3054A, 500 MHz.
- Fuente de Voltaje Leader, LPS 152.
- Resistor R de $22\,\Omega$, 10 %.
- Capacitor C_c de $1\,\mathrm{nF}$, 5 % de poliester.
- Inductor L_p de $10\,\mu\mathrm{H}$, 10 %.
- Protoboard.

La figura 5.1 nos muestra un esquema de la configuración experimental utilizada.

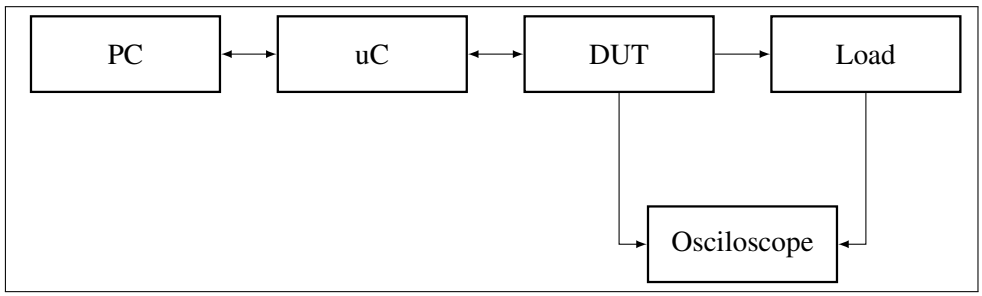


FIGURA 5.1. Esquema de la configuración experimental utilizada.

En la sección siguiente se exhiben y se analizan los resultados obtenidos.

5.2 Resultados

Para mostrar adecuadamente el funcionamiento del prototipo, separaremos los resultados experimentales en distintas sub-secciones. A continuación se exhiben cada una de ellas.

5.2.1 Transconductancias Efectivas Estáticas

La tabla 5.1 muestra las transconductancias efectivas estáticas medidas para cada uno de los 8 amplificadores de transconductancia, con sus respectivos rangos de error y valores esperados. En la tabla I_i corresponde a los amplificadores de imagen y R_i a los amplificadores de registro.

TABLA 5.1. Transconductancias efectivas estáticas obtenidas de los amplificadores de transconductancia.

Amp	G_m [S]	min(G_m) [S]	max(G_m) [S]	G_m Esperado [S]
I_1	$5.48\text{E}{-1}$	$5.16\text{E}{-1}$	$5.82\text{E}{-1}$	$4.52\text{E}{-1}$
I_2	$5.71\text{E}{-1}$	$5.39\text{E}{-1}$	$6.06\text{E}{-1}$	$4.52\text{E}{-1}$
I_3	$5.31\text{E}{-1}$	$5.01\text{E}{-1}$	$5.64\text{E}{-1}$	$4.52\text{E}{-1}$
I_4	$5.43\text{E}{-1}$	$5.11\text{E}{-1}$	$5.77\text{E}{-1}$	$4.52\text{E}{-1}$
R_1	$4.59\text{E}{-2}$	$4.56\text{E}{-2}$	$4.63\text{E}{-2}$	$4.52\text{E}{-2}$
R_2	$4.52\text{E}{-2}$	$4.48\text{E}{-2}$	$4.55\text{E}{-2}$	$4.52\text{E}{-2}$
R_3	$4.56\text{E}{-2}$	$4.53\text{E}{-2}$	$4.60\text{E}{-2}$	$4.52\text{E}{-2}$
R_4	$4.60\text{E}{-2}$	$4.57\text{E}{-2}$	$4.64\text{E}{-2}$	$4.52\text{E}{-2}$

En la tabla 5.1 podemos apreciar que los valores esperados de las transconductancias efectivas estáticas concuerdan con el valor de diseño. Para el caso de los amplificadores de imagen, estos valores difieren de lo esperado. Es posible que la razón detrás de la discrepancia en los valores obtenidos de los amplificadores de imagen se deba a su arquitectura

interna. Estos amplificadores están diseñados para aplicaciones switching de alta potencia y no para aplicaciones analógicas.

5.2.2 Señales de Entrada

La figura 5.2 muestran las señales diferenciales de entrada de la IHCS provenientes del amplificador de transimpedancia.

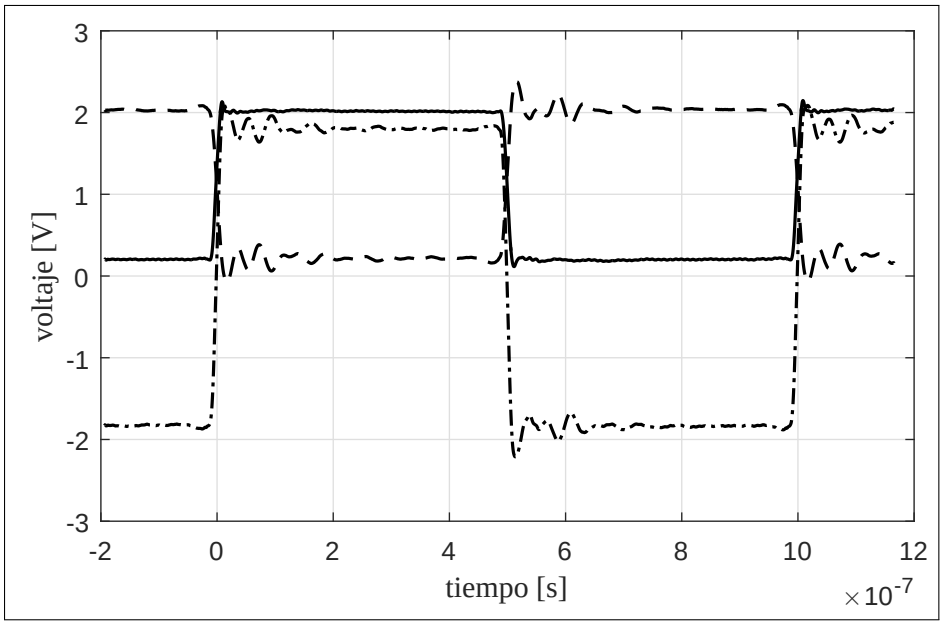


FIGURA 5.2. Señales de entrada diferenciales provenientes del amplificador de transimpedancia. En la figura se muestra la señal diferencial positiva (línea continua), la señal diferencial negativa (linea discontinua) y la señal diferencial (línea de puntos y trazos). La frecuencia de reloj es de $1\,\mathrm{MHz}$ y las señales de corriente de subida y bajada de entrada están dadas por $80\,\mathrm{mA}$ y $80\,\mathrm{mA}$ para el amplificador de registro.

En la figura 5.2 podemos apreciar que las señales de voltaje poseen bastante sobreoscilación. Esto se debe a que en el diseño del prototipo no se consideraron aspectos como adaptación de impedancia.

5.2.3 Múltiples Pendientes de Carga

La figura 5.3 muestra las salidas de los amplificadores de imagen y registro cuando se generan múltiples pendientes de carga del capacitor.

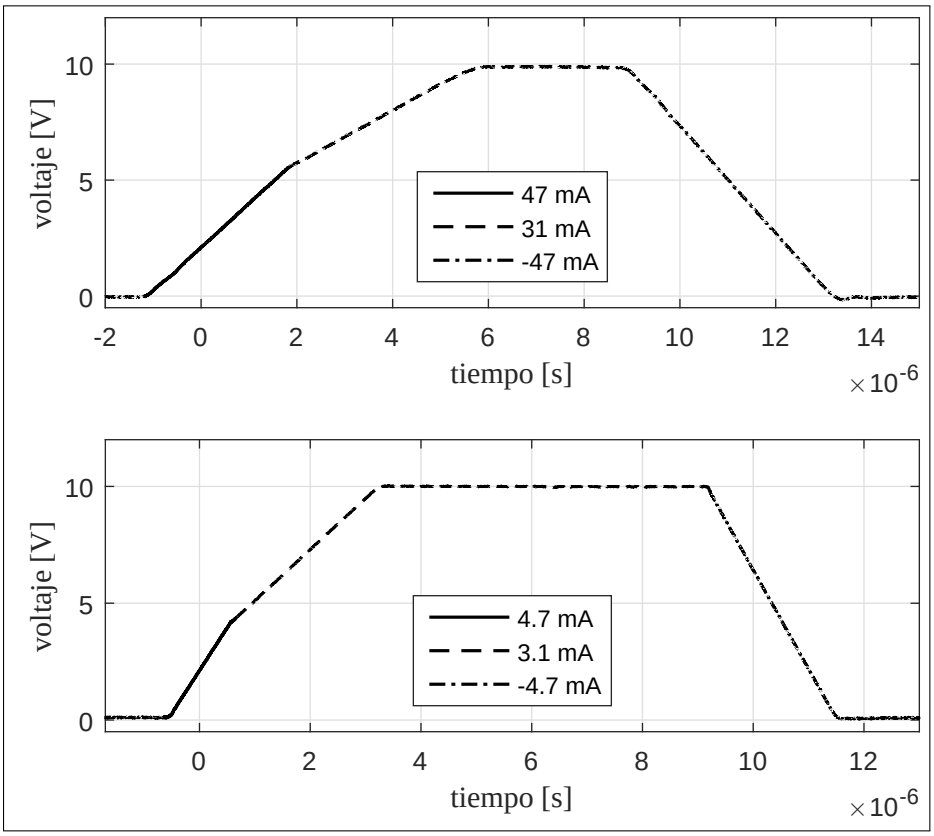


FIGURA 5.3. Formas de onda a la salida del amplificador de imagen (superior) y de registro (inferior) para una señal de reloj de $50\,\mathrm{kHz}$. Las señales de subida y bajada para el amplificador de imagen son $[47\,\mathrm{mA}\ 31\,\mathrm{mA}]$ y $[-47\,\mathrm{mA}]$, respectivamente, para una carga de $10\,\mathrm{nF}$, mientras que para el amplificador de registro son $[4.7\,\mathrm{mA}\ 3.1\,\mathrm{mA}]$ y $[-4.7\,\mathrm{mA}]$, respectivamente, para una carga de $1\,\mathrm{nF}$.

En la figura 5.3 se pueden apreciar distintos resultados. El primero de ellos consiste en que ambos tipos de amplificadores son capaces de establecer en un mismo periodo de subida y/o bajada distintas pendientes de carga. El segundo, es que a pesar de que las señales de corriente del amplificador de imagen sea diez veces superior a la de registro, las formas de onda son muy similares. Esto se debe al menor ancho de banda que poseen los

amplificadores de imagen en comparación con su símil de registro. Por último, la señal de salida del amplificador de imagen presenta distorsiones o no linealidades cerca de ambos rieles.

Con el fin de medir el desempeño del prototipo implementado, se midió la cifra de mérito Λ_e propuesta en (2.18) del tiempo de bajada en la respuesta mostrada en la figura 5.3. Para ello, se efectuó un ajuste lineal, tal y como se muestra en la figura 5.4.

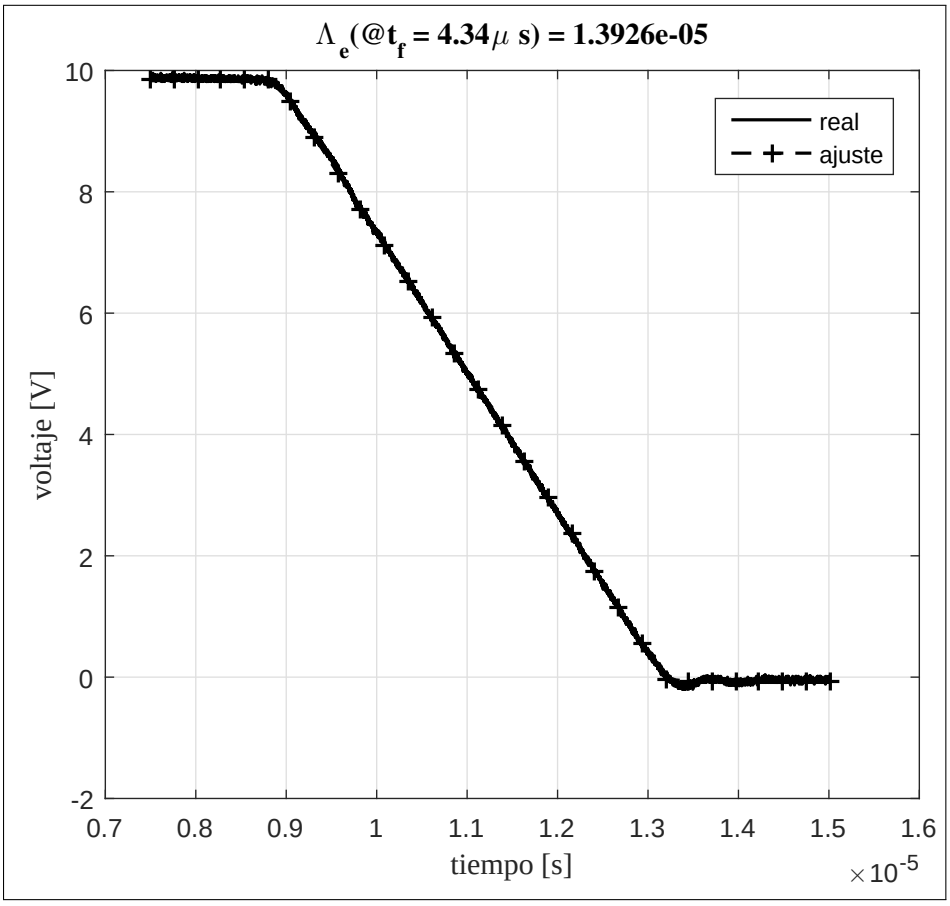


FIGURA 5.4. Ajuste lineal y cifra de mérito Λ_e propuesta en (2.18) del tiempo de bajada de la figura 5.3 en su sección de imagen.

En la figura 5.4, el cálculo de la cifra de mérito Λ_e sólo es realizado en la zona lineal y no en los extretremos o rieles, en donde se obtiene un valor de $\Lambda_e(@t_f = 4.34\,\mu\text{s}) = 1.3926\text{E}{-5}$. Para el amplificador de registro, la cifra de mérito obtenida fue de $\Lambda_e(@t_f = 2.32\,\mu\text{s}) = 8.7826\text{E}{-6}$. Los resultados de las cifras de mérito obtenidas para ambos amplificadores nos sugieren que el amplificador de registro serial establece de una mejor manera una tasa de carga constante debido a que posee un mayor ancho de banda.

5.2.4 Slew Rate Máximo

La figura 5.5 muestra las señales de salida de ambos tipos de amplificadores a corriente máxima utilizando distintos resistores de compensación.

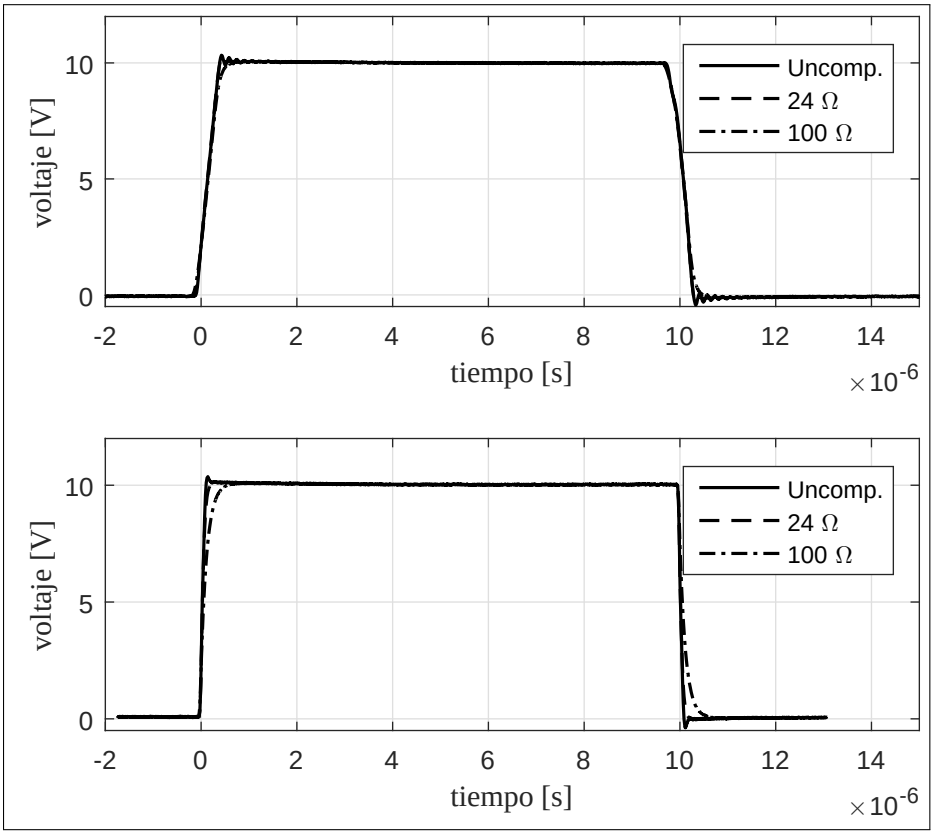


FIGURA 5.5. Formas de onda a la salida del amplificador de imagen (superior) y de registro (inferior) a máxima corriente se salida ($1\,\mathrm{A}$ imagen y $0.1\,\mathrm{A}$ registro) utilizando una carga capacitiva de $1\,\mathrm{nF}$. Para ambos gráficos se muestra la salida sin resistor de compensación, con resistor de compensación de $24\,\Omega$ y con resistor de compensación de $100\,\Omega$

.

En la figura anterior se pueden apreciar distintos resultados. El primero de ellos es que a corriente máxima ambos tipos de amplificadores quedan limitados en ancho de banda, es decir, no son capaces de establecer la corriente de salida DC en el pequeño tiempo de subida y/o bajada. Lo segundo que se observa es que a mayor corriente, mayor es el porcentaje de sobreoscilación de la forma de onda sin compensar. Lo tercero consiste en notar el efecto de la adición de los resistores de compensación en serie. Tal y como se discutió en capítulos anteriores, los resistores de compensación aumentan el margen de

fase de la IHCS disminuyendo consigo las sobre-oscilaciones del sistema. Lo último consiste en notar que el amplificador de imagen posee una mayor sobreoscilación que el de registro para la señal sin compensar. Esta diferencia puede deberse a los distintos tipos de amplificadores operacionales utilizados.

La cifra de mérito Λ_e de la figura 5.5 para una compensación de $24\,\Omega$ fue de $\Lambda_e(@t_f = 564.9\,\mathrm{ns}) = 2.1\mathrm{E}{-3}$ para el amplificador de imagen y de $\Lambda_e(@t_f = 126.3\,\mathrm{ns}) = 1.7\mathrm{E}{-3}$ para el amplificador de registro. Estos resultados nos indican que a medid que el tiempo de subida o bajada disminuye, la capacidad para establecer una tasa de caraga constante en ambos tipos de amplificadores disminuye, tal y como lo expresa la relación (2.31).

5.2.5 Frecuencia Máxima

La figura 5.6 muestra las señales a la salida a la máxima frecuencia descrita en las especificaciones de diseño.

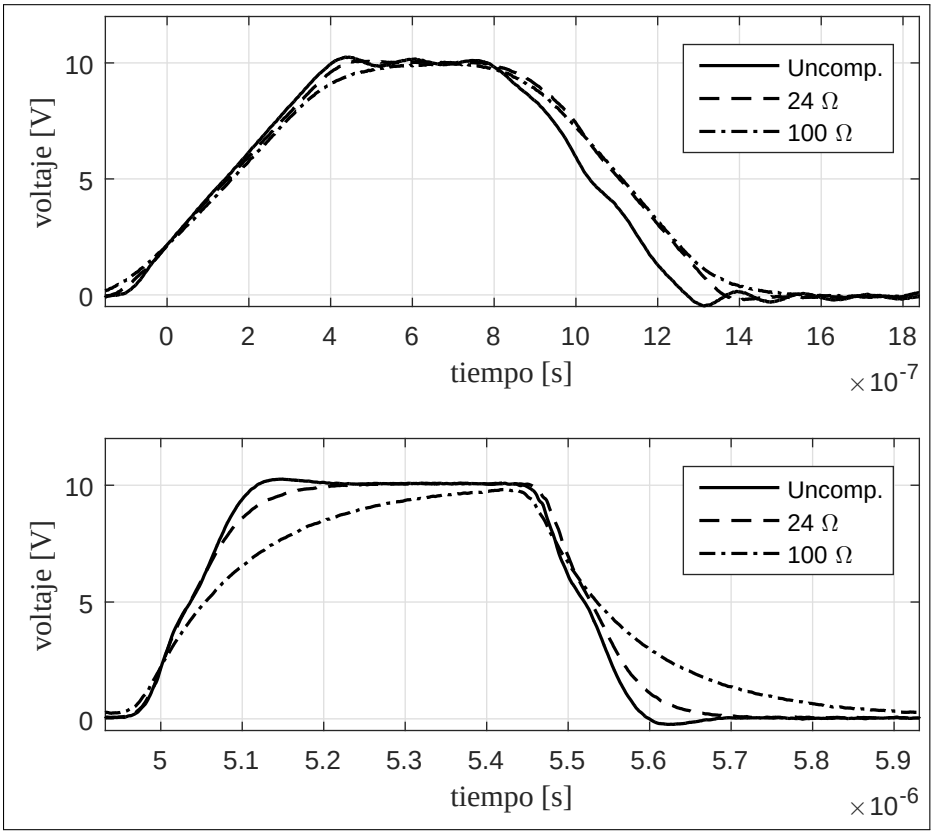


FIGURA 5.6. Formas de onda a la salida del amplificador de imagen (superior) y de registro (inferior) a $500\,\mathrm{kHz}$ para imagen y $1\,\mathrm{MHz}$ para registro utilizando una carga capacitiva de $1\,\mathrm{nF}$. Las señales de entrada para los amplificadores de imagen y registro son de $[0.5\,\mathrm{A}\ -0.5\,\mathrm{A}]$ y $[80\,\mathrm{mA}\ -80\,\mathrm{mA}]$, respectivamente. Para ambos gráficos se muestra la salida sin resistor de compensación, con resistor de compensación de $24\,\Omega$ y con resistor de compensación de $100\,\Omega$.

En la figura 5.6 podemos ver que la respuesta del amplificador de imagen está limitada en slew rate. Por otro lado, como se describió anteriormente, el efecto de los resistores de compensación es por un lado disminuir las oscilaciones y por el otro, aumentar el tiempo de establecimiento de la salida.

La cifra de mérito Λ_e de la figura 5.6 para una compensación de $24\,\Omega$ fue de $\Lambda_e(@t_f = 496.6\,\mathrm{ns}) = 1.5056\mathrm{E}{-4}$ para el amplificador de imagen y de $\Lambda_e(@t_f = 144.5\,\mathrm{ns}) =$

$8.2\mathrm{E}{-4}$ para el amplificador de registro. Este resultado nos indica que el prototipo implementado presenta una mayor cifra de mérito cuando está al límite de la frecuencia de operación que cuando la corriente de salida es máxima (slew rate máximo).

5.2.6 Distorsión Cerca de los Rieles

Si se observa la respuesta cerca de los rieles de la figura 5.3, podemos apreciar que para ambos amplificadores tiene un establecimiento aproximadamente exponencial. Este comportamiento puede ser justificado si tomamos en consideración la región de operación de los transistores de salida de los amplificadores operacionales. Cuando el voltaje de salida está cercano a los rieles de operación, los transistores de salida dejan de estar en región activa y pasan a triodo. Una vez en triodo, éstos pueden ser modelados como resistores, y en conjunto con la carga capacitiva, se explicaría el establecimiento exponencial.

Por otro lado, el amplificador de imagen posee otro tipo distorsiones en su operación cuando el voltaje de salida se encuentra cercano a los rieles. Aproximadamente a $1\,\mathrm{V}$ de los rieles existe una distorsión que puede deberse a la configuración de transistores de su etapa de salida.

5.2.7 Respuesta Frente a Inductancia Parásita

Para verificar la capacidad de compensación de los amplificadores se utilizó una inductancia de $10\,\mu\mathrm{H}$ conectada en serie con la capacitancia de carga de $1\,\mathrm{nF}$. La figura 5.7 muestra el resultado de la compensación para el amplificador de registro.

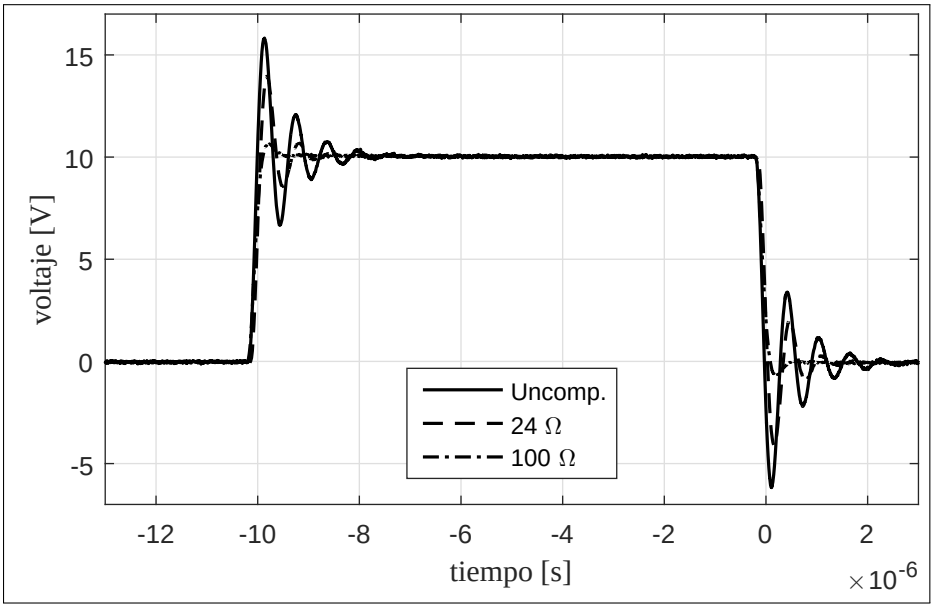


FIGURA 5.7. Salida del amplificador de registro utilizando una carga inductivo-capacitiva serie de $10\,\mu\mathrm{H}$ y $1\,\mathrm{nF}$. La señal de entrada es de $[80\,\mathrm{mA}\ -80\,\mathrm{mA}]$ y se muestra la salida sin resistor de compensación (azul), con resistor de compensación de $24\,\Omega$ (verde) y con resistor de compensación de $100\,\Omega$ (rojo).

Tal y como se discutió en la sección 3.3.3, la inductancia parásita disminuye el margen de fase de la IHCS cuando maneja cargas resistivo-capacitivas. En la figura 5.7 podemos ver que sin resistor de compensación la respuesta del sistema posee una alta sobreoscilación. El sistema sin resistor de compensación posee una sobreoscilación del 64.7 %, que disminuye a 40.4 % y a 8.3 % con los resistores de compensación en serie. Si consideraramos un sistema de segundo orden, eso equivale a reducir el margen de fase del sistema de $15°$, a $30°$ y $60°$.

6. CONCLUSIONES Y TRABAJO FUTURO

Este capítulo tiene por objetivo analizar y discutir los resultados obtenidos de todos los capítulos, así como establecer las contribuciones principales de este documento de tesis y proponer el trabajo futuro a realizar.

6.1 Discusión y Resultados

Con el fin de discutir el contenido de este trabajo, a continuación se resumen los análisis y resultados obtenidos en cada uno de los capítulos.

- **Introducción:** Este capítulo inicial describe el contexto, las motivaciones y el problema que ataca este trabajo. Además, establece los objetivos, la metodología y la organización de este documento de tesis.
- **Transferencia de Carga:** En este capítulo se describe el principio físico del proceso de transferencia de carga en detectores CCD y se expone acerca de la relación entre su desempeño y la forma de onda de las señales de lectura. Además, en este capítulo se describen los métodos experimentales a utilizar para medir el desempeño del proceso y se compara la arquitectura de control propuesta del driver de reloj con las arquitecturas tradicionales.
- **IHCS:** En este capítulo se analizó en detalle la IHCS y se propusieron simples ecuaciones de diseño en base modelos de primer y segundo orden. Se analizó el comportamiento de la IHCS frente a cargas inductivo - capacitivas y se propuso un método simple de compensación. Finalmente, se validaron los resultados mediante simulaciones en Matlab.
- **Implementación:** En este extenso capítulo se establecen las especificaciones de diseño y se exhibe el proceso de diseño circuital del prototipo, tanto en hardware como software. Finalmente, se valida el diseño mediante simulaciones de SPICE.

- **Resultados Experimentales:** Este capítulo final exhibe los resultados experimentales obtenidos con el fin de validar el correcto funcionamiento del prototipo de driver.

Tal y como se muestra en la síntesis anterior, este trabajo de tesis comienza con una descripción del contexto, el establecimiento del problema a resolver y los objetivos de este documento. Luego, se describe el proceso de transferencia de carga en CCDs y se analiza en profundidad la fuente de corriente de Howland mejorada. Finalmente, se describe el proceso de diseño e implementación del prototipo funcional y se exhiben los resultados experimentales obtenidos.

Tras todo el proceso de análisis teórico, de diseño, implementación y de obtención de resultados experimentales, el prototipo de driver de reloj controlado por corriente basado en la fuente de corriente de Howland mejorada cumple con los propósitos para los que fue diseñado. Si bien es cierto que no fue posible validar su desempeño frente a detectores CCDs reales, las pruebas frente a cargas resistivo-capacitivo-inductivas corroboraron algunas ecuaciones de diseño propuestas y ratificaron el principio detrás de esta nueva arquitectura, la cual se basa en que la mejor forma de controlar la pendiente de voltaje de cargas capacitivas es mediante fuentes de corriente y no de voltaje.

Por otro lado, las ecuaciones de diseño derivadas para la fuente de corriente de Howland mejorada poseen un gran valor en sí mismas. Los amplificadores de corriente, o más bien, pensar en “corriente” en vez de “voltaje” es una filosofía de diseño electrónico no muy usual, no obstante muy útil. Es por ello que el desarrollo analítico de la fuente de corriente de Howland mejorada corresponda a un paso que ayude a desmitificar el uso de amplificadores de corriente.

Por último, una vez que esta nueva topología pueda ser probada utilizando CCDs reales, el nuevo reto consistirá en obtener en forma precisa y acuciosa cifras de desempeño que corroboren la idea propuesta por este trabajo.

6.2 Contribuciones Principales

Dentro de las contribuciones principales de este trabajo podemos citar en primer lugar que se diseña e implementa un driver electrónico con una aplicación acorde a las necesidades de nuestro país, tal como lo es la astronomía. Tal y como se describió en el capítulo introductorio, el norte de nuestro país ha sido el principal foco de inversión astronómica a nivel mundial debido a sus inigualables condiciones climáticas y de contaminación lumínica. Es por ello que cualquier avance nacional en esta materia es un directo aporte a un desarrollo que nos pertenece.

Por otro lado, se proponen nuevas ecuaciones de diseño para la fuente de corriente de Howland mejorada, las cuales pudieron ser validadas mediante simulaciones y resultados experimentales. Tal y como se mencionó con anterioridad, estas ecuaciones tienen valor en sí mismas ya que simplifican en gran medida el proceso de diseño y otorgan un nuevo punto de vista del amplificador.

También se propone una cifra de mérito que permite comparar las distintas arquitecturas de driver de reloj existentes con la arquitectura propuesta. ESta cifra de mérito permite objetivizar las ventajas y desventajas de cada arquitectura.

Por último, se propone una nueva arquitectura de driver de reloj para detectores CCD, que utiliza señales de corriente para establecer de forma precisa la pendiente de carga de las señales de lectura, y que busca mejorar aún más de manera simple el desempeño de los detectores en el proceso de transferencia de carga. Esta contribución se definirá en su totalidad una vez que se realicen las pruebas con CCDs reales.

6.3 Trabajo Futuro

El trabajo expuesto en este documento de tesis corresponde a un inicio en el proceso de desarrollo de esta nueva arquitectura de generación de señales de lectura por corriente para CCDs.

El primer paso a realizar consistirá en validar el desempeño de la topología propuesta por este trabajo utilizando CCDs reales. Es necesario obtener cifras de desempeño que avalen y justifiquen el uso de esta nueva arquitectura por sobre las tradicionales.

Un segundo paso consistirá en corroborar experimentalmente todas las ecuaciones de diseño propuestas, incluyendo las ecuaciones referentes a ruido electrónico. Esto tiene por objetivo darle completitud al trabajo realizado en este documento.

Un tercer paso consistirá en realizar una nueva versión de este prototipo. Esto con el fin de corregir todos los errores de diseño y mejorar algunos aspectos no considerados en la primera versión. Por ejemplo, podría considerarse la adaptación de impedancias de líneas con el fin de mejorar el porcentaje de sobreoscilaciones de las señales conformadas o mejorar la forma en que se definen las ganancias, evitando el uso de potenciómetros de ajuste.

Una vez validada la arquitectura con CCDs reales, el último paso será traspasar la mayor parte de este diseño y/o idea a un circuito integrado de aplicación específica (ASIC). La idea inicial de este trabajó siempre tuvo en consideración un futuro diseño en circuito integrado.

BIBLIOGRAFÍA

ADV7125 — datasheet and product info 330MHz Triple 8-Bit High Speed Video DAC — Analog Devices. (s.f.). Descargado 2015-10-19, de http://www.analog.com/en/products/digital-to-analog-converters/da-converters/adv7125.html{#}product-overview

Chiaberge, M., Riess, A., Mutchler, M., Sirianni, M., y Mack, J. (2005). ACS Charge Transfer Efficiency . Results from Internal and External Tests. *2005 HST Calibration Work.*, 36–40.

Christen, F., Kuijken, K., y Baade, D. (2006). CCD Charge Transfer Efficiency (CTE) Derived from Signal Variance in Flat Field Images. *Sci. Detect. ...*, 543–548. Descargado de http://link.springer.com/content/pdf/10.1007/1-4020-4330-9{_}61.pdf

Daigle, O., y Blais-Ouellette, S. (2010). Photon counting with an EMCCD. En *Proc. spie* (Vol. 7536, p. 753606). Descargado de http://link.aip.org/link/PSISDG/v7536/i1/p753606/s1{&}Agg=doi doi: 10.1117/12.840047

Dao, V., Etoh, T., y Le, C. (2008). Estimation Of Attenuation And Phase Delay In Driving Voltage Waveform Of An Ultra-High-Speed Image Sensor by Dimensional Analysis. *Int. J. ...*, *2*(10), 1104–1109.

Dorn, R. J. (2001). *A CCD based curvature wavefront sensor for adaptive optics in astronomy* (Tesis Doctoral no publicada).

E2v. (2014). *Imaging Sensors (CMOS, CCD, EMCCD) Resources.* Descargado 2014-12-18, de http://www.e2v.com/products/imaging/imaging-sensors-cmos-ccd-emccd/?resources{#}literature

Filho, P. B. (2002). Tissue Characterisation using an Impedance Spectroscopy Probe. *Med. Phys.*(September).

Holland, A. D. (1990). Radiation Effects in CCD X-Ray Detectors. (April 1990), 1–174. Descargado de http://hdl.handle.net/2381/27614

Howell, S. B. (2006). *Handbook of Ccd Astronomy* (Vol. second edi). Descargado de http://books.google.ca/books/about/Handbook{_}of{_}Ccd{_}Astronomy.html?id=ZZCTqanpsZUC{&}pgis=1 doi: 10.2277/0521617626

J. Hahn, T. Edison, y T. Edgar. (2001). *A note on stability analysis using Bode plots.*

Janesick, J. (1997). CCD transfer method: standard for absolute performance of CCDs and digital CCD camera systems. *Electron. Imaging'97*, *3019*. Descargado de http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=920029

Janesick, J. R. (2001). *Scientific Charge-coupled Devices*. Descargado de http://books.google.com/books?hl=es{&}lr={&}id=rkgBkbDie7kC{&}pgis=1

Jones, M. R., Clampin, M., Meurer, G., y Schrein, R. (1999). Justification and Requirements for On-Board ACS FPR/EPER CTE Calibration. *ACS Tech. Instrum. Rep.*(Ngc 5139), 1–30.

Liu, R., Lu, J., Ding, J., y Liu, Y. (2011). CCD Signal Processing Based on Correlated Double Sampling. , 3–5.

LTC6269 - Dual 500MHz Ultra-Low Bias Current FET Input Op Amp - Linear Technology. (s.f.). Descargado 2015-10-20, de http://www.linear.com/product/LTC6269

McLean, I. S. (2008). *Electronic Imaging in Astronomy: Detectors and Instrumentation*. Springer Science & Business Media. Descargado de https://books.google.com/books?id=FGHhZf-k8SkC{&}pgis=1

OPA564 — Precision Amplifier — Operational Amplifier (Op Amp) — Description & parametrics. (s.f.). Descargado 2015-10-21, de http://www.ti.com/product/opa564

Ortiz, O. A. A., Vergara, O. J. B., y Mercado, D. A. M. (2007). *Criterios de Diseño de la Fuente de Corriente de Holwand* (Vol. 6) (n.º 1). Descargado de http://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/1950

Pease, R. (2008). A comprehensive study of the Howland current source. *Natl. Semicond. St. Clara, CA*(April), 1–17. Descargado de http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:A+comprehensive+study+of+the+Howland+current+source{#}0

Rosenstark, S. (1986). *Feedback amplifier principles*. Macmillan Pub. Co. Descargado de http://books.google.cl/books/about/Feedback{_}amplifier{_}principles.html?id=Vg9TAAAAMAAJ{&}pgis=1

Sopczak, A. (2005, jul). Charge Transfer Inefficiency Studies for CCD Vertex Detectors at a LC. Descargado de http://arxiv.org/abs/physics/0507028

Spartan-6 LX FPGAs. (s.f.). Descargado 2015-10-19, de http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/lx.html

Stefanov, K. D., y Murray, N. J. (s.f.). Optimal digital correlated double sampling for CCD signals.

doi: 10.1049/el.2014.0759

The University of Oregon Physics Department web site. (1997). *Evolving Towards The Perfect CCD*. Descargado 2015-11-01, de http://zebu.uoregon.edu/ccd.html

THS4222 — High Speed Amplifier (>=50MHz) — Operational Amplifier (Op Amp) — Description & parametrics. (s.f.). Descargado 2015-10-21, de http://www.ti.com/product/ths4222

Tucker, A., Fox, R., y Sadleir, R. (2013). Biocompatible, high precision, wideband, improved Howland current source with lead-lag compensation. *Biomed. Circuits ...*, *7*(1), 63–70. Descargado de http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6221965

Wang, Y., y Li, T. (2010, may). Wave-shaping & Engineering Realization of CCD Driving Signals. , *7658*. Descargado de http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=761472
doi: 10.1117/12.865985

ANEXOS

A. ESPECIFICACIONES CCDS COMERCIALES.

La tabla A.1 se muestra la información recopilada de CCDs comerciales [E2v (2014)]:

TABLA A.1. Datos de CCD Comerciales de e2v.

Modelo	N Pixeles (H)	N Pixeles (V)	Tamaño Pyxel [m]	Sección	Número Secciones	Fases/Sección	Vmin [V]	A [V]	Vmax [V]	Cap/Fase [F]	Cap Total [F]	T_C [s]	T_r [s]	I_{max} [A]	I_{rms} [Arms]	P_{rms} [Wrms]	Cap/Area [F/m2]
CCD231-84	4096	4112	1,50E-6	Imagen	4	4	0	10	10	2,50E-8	1,00E-7	7,50E-5	1,00E-6	8,00E-1	1.63E-1	9.43E-1	2,64E-3
				Registro	4	3	1	10	11	4,75E-11	1,90E-10	2,00E-6	5,00E-8	3,04E-2	8,50E-3	5.66E-2	
				Reset			0	12	12								
CCD30-11	1024	255	2,60E-5	Imagen	1	3	0	12	12	1,50E-8	1,50E-8	3,00E-5	2,00E-6	7,20E-2	3,29E-2	2,28E-1	8,50E-5
				Registro	1	3	1	10	11	3,25E-10	3,25E-10	2,22E-5	2,22E-6	1,17E-3	6,54E-4	4,35E-3	
				Reset			0	12	12								
CCD42-10	2048	515	1,35E-5	Imagen	1	3	0	12	12	2,50E-8	2,50E-8	3,00E-5	2,00E-6	1,20E-1	5,48E-2	3,79E-1	1,30E-4
				Registro	1	3	1	10	11	3,10E-10	3,10E-10	5,00E-5	5,00E-6	4,96E-4	2,77E-4	1,85E-3	
				Reset			0	12	12								
CCD47-10	1056	1027	1,30E-5	Imagen	2	3	0	12	12	8,00E-9	1,60E-8	1,40E-5	5,00E-6	3,07E-2	3,25E-2	2,25E-1	8,73E-5
				Registro	2	3	1	10	11	1,40E-10	2,80E-10	1,00E-6	1,00E-7	2,24E-2	1,25E-2	8,34E-2	
				Reset			0	12	12								
CCD203-82	4096	4136	1,20E-5	Imagen	2	4	0	10	10	6,80E-8	1,36E-7	9,00E-5	5,00E-7	2,18E+0	2,87E-1	1,66E+0	5,57E-5
				Registro	4	2,5	1	10	11	3,80E-10	1,52E-9	1,00E-6	2,00E-8	6,08E-1	1,52E-1	1,01E+0	
				Reset			0	10	10								
CCD39-01	80	80	2,40E-5	Imagen	1	3	0	12	12	2,00E-10	2,00E-10	2,00E-6	1,00E-7	1,92E-2	7,59E-3	5,26E-2	5,43E-5
				Registro	1	3	1	11	12	3,40E-11	3,40E-11	1,00E-6	1,00E-7	2,99E-3	1,67E-3	1,21E-2	
				Reset			0	12	12	1,00E-11							
CCD47-20	1024	1024	1,30E-5	Imagen	1	3	0	12	12	1,15E-8	1,15E-8	1,40E-5	5,00E-7	2,21E-1	7,38E-2	5,11E-1	6,49E-5
				Registro	2	3	1	10	11	1,50E-10	3,00E-10	1,00E-6	1,00E-7	2,40E-2	1,34E-2	8,93E-2	
				Reset			0	12	12								
CCD44-82	2048	4096	1,50E-5	Imagen	1	3	0	10	10	9,00E-8	9,00E-8	1,00E-4	1,00E-5	7,20E-2	4,02E-2	2,32E-1	4,77E-5
				Registro	2	3	1	11	12	3,55E-10	7,10E-10	1,00E-6	1,00E-7	6,25E-2	3,49E-2	2,53E-1	
				Reset			0	12	12	2,00E-11							
CCD230-42	2048	2064	1,50E-5	Imagen	4	4	0	11	11	1,00E-8	4,00E-8	7,20E-5	1,00E-6	3,52E-1	7,33E-2	4,66E-1	4,21E-5
				Registro	4	3	0	11	11	2,25E-11	9,00E-11	1,33E-6	5,00E-8	1,58E-2	5,42E-3	3,44E-2	
				Reset			0	11	11								
CCD230-84	4096	4112	1,50E-5	Imagen	4	4	0	11	11	4,00E-8	1,60E-7	1,35E-4	2,00E-6	7,04E-1	1,51E-1	9,62E-1	4,22E-5
				Registro	4	3	0	11	11	4,75E-11	1,90E-10	1,33E-6	5,00E-8	3,34E-2	1,14E-2	7,27E-2	
				Reset			0	11	11								
CCD231-C6	6144	6160	1,50E-5	Imagen	4	4	0	10	10	5,50E-8	2,20E-7	2,25E-4	2,00E-6	8,80E-1	1,47E-1	8,47E-1	2,58E-5
				Registro	4	3	1	10	11	7,50E-11	3,00E-10	2,00E-6	6,00E-5	4,00E-5	3,87E-4	2,58E-3	
				Reset			0	12	12								
CCD290-99	9216	9232	1,00E-5	Imagen	2	4	0	11	11	1,60E-7	3,20E-7	3,00E-4	5,00E-6	5,63E-1	1,29E-1	8,16E-1	3,76E-5
				Registro	4	3	1	10	11	1,63E-10	6,50E-10	2,00E-6	7,00E-8	7,43E-2	2,46E-2	1,64E-1	
				Reset			0	12	12								

B. CÓDIGOS.

B.1 Módulos Verilog FPGA

```
1  module top(
2   input osc_gclk,
3   input debug1,
4   input debug2,
5   input debug3,
6   input debug4,
7   input debug5,
8   input debug6,
9   input debug7,
10  input debug8,
11  input debug9,
12  input debug10,
13
14  output config1,
15  input config2,
16  output config3,
17  output config4,
18
19  input spi_dout,
20  input eoc1,
21  input eoc2,
22
23  output [7:0] in1, //salida iop1
24  output [7:0] in2, //salida iop2
25  output [7:0] in3, //salida iop3
26  output [7:0] in4, //salida iop4
27  output [7:0] in5, //salida rop1
28  output [7:0] in6, //salida rop2
29  output [7:0] in7, //salida rop3
30  output [7:0] in8, //salida rop4
31  output reg_en_n,
32  output reg_lclk,
33  output reg_out,
34  output reg_rst,
35  output reg_sclk,
36  output spi_din,
37  output spi_sclk,
38  output dac_gclk
39     );
40
41  ////////////////////////////
42  // instanciacion de modulos
43  ////////////////////////////
44
45  wire clk_rspi;
46  wire clk_i;
47  wire clk_r;
48
49  wire [7:0] divider_i;
50  wire [7:0] divider_i2;
51  wire [7:0] divider_r;
52  wire [7:0] divider_r2;
53
54  wire [15:0] dac2_aux;
55
56  clk_divider clk_regspi(
```

```
57    .clk_in(osc_gclk),
58    .divider(8'd200),
59    .clk_out(clk_rspi)
60    );
61
62    wire regspi_load;
63    wire [3:0] regspi_mode;
64    wire regspi_done;
65    wire [23:0] reg_word;
66    wire [7:0] spi_word_8;
67    wire [15:0] spi_fifo;
68    wire [15:0] reg_spi_aux;
69
70    reg_spi reg_spi(
71     .clk(clk_rspi),
72     .load(regspi_load),
73     .mode(regspi_mode),
74     .done(regspi_done),
75
76     .reg_word(reg_word),
77     .reg_lclk(reg_lclk),
78     .reg_sclk(reg_sclk),
79     .reg_rst(reg_rst),
80     .reg_out(reg_out),
81
82     .spi_word_8(spi_word_8),
83     .spi_dout(spi_dout),
84     .spi_fifo(spi_fifo),
85     .spi_din(spi_din),
86     .spi_sclk(spi_sclk),
87     .aux(reg_spi_aux)
88     );
89
90    wire [79:0] upi;
91    wire [79:0] downi;
92    wire [7:0] upi_states;
93    wire [7:0] downi_states;
94    wire [7:0] iidle;
95
96    wire [79:0] upr;
97    wire [79:0] downr;
98    wire [7:0] upr_states;
99    wire [7:0] downr_states;
100   wire [7:0] ridle;
101
102   dac3 iop1(
103    .clk(osc_gclk),
104    .divider({divider_i2, divider_i}),
105    .en(opamp[0]),
106    .set(debug10),
107    .up(upi),
108    .down(downi),
109    .up_states(upi_states),
110    .down_states(downi_states),
111    .idle(iidle),
112    .out(in1)
113    );
114
115   dac3 iop2(
116    .clk(osc_gclk),
117    .divider({divider_i2, divider_i}),
118    .en(opamp[1]),
119    .set(debug9),
```

```
120     .up(upi),
121     .down(downi),
122     .up_states(upi_states),
123     .down_states(downi_states),
124     .idle(iidle),
125     .out(in2)
126     );
127
128   dac3 iop3(
129     .clk(osc_gclk),
130     .divider({divider_i2, divider_i}),
131     .en(opamp[2]),
132     .set(debug8),
133     .up(upi),
134     .down(downi),
135     .up_states(upi_states),
136     .down_states(downi_states),
137     .idle(iidle),
138     .out(in3)
139     );
140
141   dac3 iop4(
142     .clk(osc_gclk),
143     .divider({divider_i2, divider_i}),
144     .en(opamp[3]),
145     .set(debug7),
146     .up(upi),
147     .down(downi),
148     .up_states(upi_states),
149     .down_states(downi_states),
150     .idle(iidle),
151     .out(in4),
152     .aux(dac2_aux)
153     );
154
155   dac3 rop1(
156     .clk(osc_gclk),
157     .divider({divider_r2, divider_r}),
158     .en(opamp[4]),
159     .set(debug6),
160     .up(upr),
161     .down(downr),
162     .up_states(upr_states),
163     .down_states(downr_states),
164     .idle(ridle),
165     .out(in5)
166     );
167
168   dac3 rop2(
169     .clk(osc_gclk),
170     .divider({divider_r2, divider_r}),
171     .en(opamp[5]),
172     .set(debug5),
173     .up(upr),
174     .down(downr),
175     .up_states(upr_states),
176     .down_states(downr_states),
177     .idle(ridle),
178     .out(in6)
179     );
180
181   dac3 rop3(
182     .clk(osc_gclk),
```

```
183     .divider({divider_r2, divider_r}),
184     .en(opamp[6]),
185     .set(debug4),
186     .up(upr),
187     .down(downr),
188     .up_states(upr_states),
189     .down_states(downr_states),
190     .idle(ridle),
191     .out(in7)
192     );
193
194   dac3 rop4(
195     .clk(osc_gclk),
196     .divider({divider_r2, divider_r}),
197     .en(opamp[7]),
198     .set(debug3),
199     .up(upr),
200     .down(downr),
201     .up_states(upr_states),
202     .down_states(downr_states),
203     .idle(ridle),
204     .out(in8)
205     );
206
207   wire [7:0] opamp;
208
209   wire rst;
210
211   wire [7:0] dout_word;
212   wire config_load_en;
213
214   wire [7:0] adc_setup;
215   wire [7:0] adc_aver;
216   wire [7:0] adc_conv;
217   wire adc_1;
218   wire adc_2;
219   wire adc_read;
220
221   wire reconfig;
222
223   wire control_done;
224
225   wire [15:0] control_aux;
226
227   control control(
228     .clk(osc_gclk),
229
230     .opamp(opamp),
231     .rst(rst),
232
233     .eoc1(eoc1),
234     .eoc2(eoc2),
235
236     .adc_setup(adc_setup),
237     .adc_aver(adc_aver),
238     .adc_conv(adc_conv),
239     .adc_1(adc_1),
240     .adc_2(adc_2),
241     .adc_read(adc_read),
242
243     .reconfig(reconfig),
244
245     .regspi_done(regspi_done),
```

```
246     .regspi_load(regspi_load),
247     .reg_word(reg_word),
248     .spi_word8(spi_word_8),
249     .regspi_mode(regspi_mode),
250     .reg_en_n(reg_en_n),
251
252     .config_load_en(config_load_en),
253
254     .done(control_done),
255
256     .aux(control_aux)
257   );
258
259   wire config_lclk;
260   wire [7:0] din_word;
261   wire [15:0] spiconfig_aux;
262   wire spi_config_done;
263
264   spi_config spi_config(
265     .clk(osc_gclk),
266     .sclk(debug1),
267     .din(config2),
268     .load_en(config_load_en),
269     .dout_word(spi_fifo),
270
271     .lclk(config_lclk),
272     .din_word(din_word),
273     .done(spi_config_done),
274     .aux(spiconfig_aux)
275   );
276
277   wire [7:0] config_aux;
278   wire config_done;
279
280   config config(
281     .sclk(debug1),
282     .din_word(din_word),
283     .lclk(config_lclk),
284     .control_done(control_done),
285
286     .opamp(opamp),
287
288     .rst(rst),
289
290     .upi(upi),
291     .downi(downi),
292     .upi_states(upi_states),
293     .downi_states(downi_states),
294     .iidle(iidle),
295     .divider_i(divider_i),
296     .divider_i2(divider_i2),
297
298     .upr(upr),
299     .downr(downr),
300     .upr_states(upr_states),
301     .downr_states(downr_states),
302     .ridle(ridle),
303     .divider_r(divider_r),
304     .divider_r2(divider_r2),
305
306     .adc_setup(adc_setup),
307     .adc_aver(adc_aver),
308     .adc_conv(adc_conv),
```

```
309     .adc_1(adc_1),
310     .adc_2(adc_2),
311     .adc_read(adc_read),
312 
313     .reconfig(reconfig),
314 
315     .done(config_done),
316 
317     .aux(config_aux)
318   );
319 
320   assign config1 = spi_config_done;
321   assign config4 = config_done;
322 
323   assign dac_gclk = osc_gclk;
324 
325   //////////////////////
326   // otras asignaciones
327   //////////////////////
328 
329   assign config3 = dac2_aux[0];
330 
331 endmodule
```

```
1  module control(
2      input clk,
3  
4      input [7:0] opamp,
5  
6      input eoc1,
7      input eoc2,
8  
9      input rst,
10 
11     input [7:0] adc_setup,
12     input [7:0] adc_aver,
13     input [7:0] adc_conv,
14     input adc_1,
15     input adc_2,
16     input adc_read,
17 
18     input reconfig,
19 
20     input regspi_done,
21     output regspi_load,
22     output [23:0] reg_word,
23     output [7:0] spi_word8,
24     output [3:0] regspi_mode,
25     output reg_en_n,
26 
27     output config_load_en,
28 
29     output done,
30 
31     output [15:0] aux
32     );
33 
34   ////////////////////////////////
35   // configuracion de parametros
36   ////////////////////////////////
37 
38   parameter cs_none  = 6'b111111;
39   parameter cs_adc1  = 6'b111110;
```

```
40    parameter cs_adc2  = 6'b111101;
41
42    parameter en_is1  = 13'd256;    // imagen op1 supply
43    parameter en_is2  = 13'd512;    // imagen op2 supply
44    parameter en_is3  = 13'd1024;   // imagen op3 supply
45    parameter en_is4  = 13'd2048;   // imagen op4 supply
46    parameter en_rs   = 13'd4096;   // register op supply
47
48    parameter psave1 = 3'b001;
49    parameter psave2 = 3'b010;
50    parameter psave3 = 3'b100;
51
52    parameter en_reset1 = 2'b01;    // supply reset
53    parameter en_reset2 = 2'b10;    // enable reset inputs
54
55    parameter reg_a       = 4'd0;
56    parameter reg_spi8  = 4'd1;
57    parameter reg_spi9  = 4'd2;
58    parameter reg_read  = 4'd3;
59    parameter spi8      = 4'd4;
60    parameter spi9      = 4'd5;
61    parameter read      = 4'd6;
62
63    ////////////////////
64    // wires y registros
65    ////////////////////
66
67    reg [7:0] state = 8'b0;
68    wire clk_en;
69    wire [7:0] state_next;
70
71    // shift register
72    wire [12:0] en;
73    wire [5:0] cs;
74    wire [2:0] psave;
75    wire [1:0] en_rst;
76
77
78    /////////////
79    // simulacion
80    /////////////
81
82    assign aux = {en[7:0],state};
83
84    ////////////////////////
85    // always y asignaciones
86    ////////////////////////
87
88    wire [12:0] en_1 = (opamp[0])? en_is1 : 13'b0;
89    wire [12:0] en_2 = (opamp[1])? en_is2 : 13'b0;
90    wire [12:0] en_3 = (opamp[2])? en_is3 : 13'b0;
91    wire [12:0] en_4 = (opamp[3])? en_is4 : 13'b0;
92    wire [12:0] en_5 = (opamp[7:4] > 4'b0)? en_rs : 13'b0;
93
94    wire [7:0] reg_word1 = {en_rst[1], cs[5], en_rst[0], cs[1], psave[2], en[12], cs[4], en[11]};
95    wire [7:0] reg_word2 = {psave[1], en[3], en[2], en[1], en[0], en[10], cs[3], en[7]};
96    wire [7:0] reg_word3 = {en[6], en[5], en[4], cs[0], en[9], psave[0], cs[2], en[8]};
97
98    wire [2:0] psave_1 = (opamp[2:0] > 3'b0)? psave1 : 3'b0;
99    wire [2:0] psave_2 = (opamp[5:3] > 3'b0)? psave2 : 3'b0;
100   wire [2:0] psave_3 = (opamp[7:6] > 2'b0)? psave3 : 3'b0;
101
102
```

```
103   assign  clk_en = ((state[0] == 1'b0) && (state <= 8'd7))?
104      regspi_done     :   1'bz,  // listo
105     clk_en = ((state[0] == 1'b1) && (state <= 8'd7))?
106      ~regspi_done    :   1'bz,
107
108     clk_en = (state == 8'd8)?       adc_read     :   1'bz,
109     clk_en = (state == 8'd9)?       regspi_done  :   1'bz,
110     clk_en = (state == 8'd10)?      ~regspi_done :   1'bz,
111     clk_en = (state == 8'd11)?      regspi_done  :   1'bz,
112     clk_en = (state == 8'd12)?      ~regspi_done :   1'bz,
113     clk_en = (state == 8'd13)?      ~eoc1        :   1'bz,
114     clk_en = (state == 8'd14)?      regspi_done  :   1'bz,
115     clk_en = (state == 8'd15)?      ~regspi_done :   1'bz,
116
117     clk_en = (state == 8'd16)?      adc_read     :   1'bz,
118     clk_en = (state == 8'd17)?      regspi_done  :   1'bz,
119     clk_en = (state == 8'd18)?      ~regspi_done :   1'bz,
120     clk_en = (state == 8'd19)?      regspi_done  :   1'bz,
121     clk_en = (state == 8'd20)?      ~regspi_done :   1'bz,
122     clk_en = (state == 8'd21)?      ~eoc2        :   1'bz,
123     clk_en = (state == 8'd22)?      regspi_done  :   1'bz,
124     clk_en = (state == 8'd23)?      ~regspi_done :   1'bz,
125
126     clk_en = (state >  8'd23)?     1'b1         :   1'bz;
127
128   assign  state_next = (reconfig)?
129      8'd0    :   8'bzzzz_zzzz,   // listo
130     state_next = (adc_1)?
131      8'd8    :   8'bzzzz_zzzz,
132     state_next = (adc_2)?
133      8'd16   :   8'bzzzz_zzzz,
134     state_next = (~reconfig && ~adc_1 && ~adc_2)?
135      8'd5    :   8'bzzzz_zzzz;
136
137   assign  en = (state >= 8'd4)?    (en_1 | en_2 | en_3 | en_4 | en_5) : 13'b0;
138
139   assign  en_rst = ((rst == 1'b1) && (state >= 8'd4))? (en_reset2 | en_reset1) : 2'b0;
140
141   assign psave = psave_1 | psave_2 | psave_3;     // listo
142
143   assign  cs = ((state <= 8'd3))?
144      (cs_adc1 & cs_adc2)     :   6'bzz_zzzz,
145
146     cs = ((state >= 8'd4) && (state <= 8'd7))?
147      cs_none                 :   6'bzz_zzzz,
148
149     cs = ((state >= 8'd8) && (state <= 8'd10))?
150      cs_adc1                 :   6'bzz_zzzz,
151     cs = ((state >= 8'd11) && (state <= 8'd13))?
152      cs_none                 :   6'bzz_zzzz,
153     cs = ((state >= 8'd14) && (state <= 8'd15))?
154      cs_adc1                 :   6'bzz_zzzz,
155
156     cs = ((state >= 8'd16) && (state <= 8'd18))?
157      cs_adc2                 :   6'bzz_zzzz,
158     cs = ((state >= 8'd19) && (state <= 8'd21))?
159      cs_none                 :   6'bzz_zzzz,
160     cs = ((state >= 8'd22) && (state <= 8'd23))?
161      cs_adc2                 :   6'bzz_zzzz,
162
163     cs = (state >= 8'd24)?
164      cs_none                 :   6'bzz_zzzz;
165
```

```
166
167   assign reg_word = {reg_word3, reg_word2, reg_word1};
168
169   assign reg_en_n = 1'b0;
170
171   assign  regspi_load = (state <= 8'd7)?
172      ~state[0]   :   1'bz,
173
174     regspi_load = (state == 8'd8)?
175      1'b0        :   1'bz,
176     regspi_load = (state == 8'd9)?
177      1'b1        :   1'bz,
178     regspi_load = (state == 8'd10)?
179      1'b0        :   1'bz,
180     regspi_load = (state == 8'd11)?
181      1'b1        :   1'bz,
182     regspi_load = (state == 8'd12)?
183      1'b0        :   1'bz,
184     regspi_load = (state == 8'd13)?
185      1'b0        :   1'bz,
186     regspi_load = (state == 8'd14)?
187      1'b1        :   1'bz,
188     regspi_load = (state == 8'd15)?
189      1'b0        :   1'bz,
190
191     regspi_load = (state == 8'd16)?
192      1'b0        :   1'bz,
193     regspi_load = (state == 8'd17)?
194      1'b1        :   1'bz,
195     regspi_load = (state == 8'd18)?
196      1'b0        :   1'bz,
197     regspi_load = (state == 8'd19)?
198      1'b1        :   1'bz,
199     regspi_load = (state == 8'd20)?
200      1'b0        :   1'bz,
201     regspi_load = (state == 8'd21)?
202      1'b0        :   1'bz,
203     regspi_load = (state == 8'd22)?
204      1'b1        :   1'bz,
205     regspi_load = (state == 8'd23)?
206      1'b0        :   1'bz,
207
208     regspi_load = (state >  8'd23)?
209      1'b0        :   1'bz;
210
211
212   assign  spi_word8 =  ((state <= 8'd1))?
213      adc_aver    :   8'bzzzz_zzzz,
214     spi_word8 =  ((state >= 8'd2)  && (state <= 8'd3))?
215      adc_setup   :   8'bzzzz_zzzz,
216     spi_word8 =  ((state >= 8'd4)  && (state <= 8'd7))?
217      8'b0        :   8'bzzzz_zzzz,
218     spi_word8 =  ((state >= 8'd8)  && (state <= 8'd23))?
219      adc_conv    :   8'bzzzz_zzzz,
220     spi_word8 = (state > 8'd23)?
221      8'b0        :   8'bzzzz_zzzz;
222
223
224   assign  regspi_mode = ((state <= 8'd1))?
225      reg_spi8 :   4'bzzzz,
226     regspi_mode = ((state >= 8'd2)  && (state <= 8'd3))?
227      spi8  :   4'bzzzz,
228
```

```
229     regspi_mode = ((state >= 8'd4)  && (state <= 8'd7))?
230      reg_a  :   4'bzzzz,
231
232     regspi_mode = ((state >= 8'd8)  && (state <= 8'd10))?
233      reg_spi8 :   4'bzzzz,
234     regspi_mode = ((state >= 8'd11)  && (state <= 8'd13))?
235      reg_a  :   4'bzzzz,
236     regspi_mode = ((state >= 8'd14)  && (state <= 8'd15))?
237      reg_read :   4'bzzzz,
238
239     regspi_mode = ((state >= 8'd16)  && (state <= 8'd18))?
240      reg_spi8 :   4'bzzzz,
241     regspi_mode = ((state >= 8'd19)  && (state <= 8'd21))?
242      reg_a  :   4'bzzzz,
243     regspi_mode = ((state >= 8'd22)  && (state <= 8'd23))?
244      reg_read :   4'bzzzz,
245
246     regspi_mode = (state > 8'd37)?
247      4'd7        :   4'bzzzz;
248
249   assign  config_load_en = ((state == 8'd15) || (state == 8'd23))?
250      1'b1    :   1'b0;
251
252   assign  done = (state == 8'd5)?
253      1'b1 :  1'b0;
254
255   always @(posedge clk)
256   begin
257    if(clk_en)
258    begin
259     if(state == 8'd0)
260      state <= 8'd1;          // configuracion adc aver
261     else if(state == 8'd1)
262      state <= 8'd2;          // fin adc aver
263     else if(state == 8'd2)
264      state <= 8'd3;          // configuracion adc setup
265     else if(state == 8'd3)
266      state <= 8'd4;          // fin adc setup
267
268     else if(state == 8'd4)
269      state <= 8'd5;           // comienzo del fin del encendido de amplificadores
270     else if(state == 8'd5)
271      state <= state_next;    // estado de espera.
272
273     else if(state == 8'd6)
274      state <= 8'd7;           // recongiguración. se escribe en_1_old y en_rst_1_old
275     else if(state == 8'd7)
276      state <= 8'd4;           // se finaliza la pre- reconfiguracion y se vuelve al inicio.
277
278     else if(state == 8'd8)
279      state <= 8'd9;           // adc1. esperamos a adc_read.
280     else if(state == 8'd9)
281      state <= 8'd10;          // se inicia la escritura en el conversion register.
282     else if(state == 8'd10)
283      state <= 8'd11;          // finaliza la escritura en el conversion register.
284     else if(state == 8'd11)
285      state <= 8'd12;          // inicio up todos los cs.
286     else if(state == 8'd12)
287      state <= 8'd13;          // fin up todos los cs.
288     else if(state == 8'd13)
289      state <= 8'd14;          // esperamos a que eoc1 = 0
290     else if(state == 8'd14)
291      state <= 8'd15;          // inicio escritura cs_adc1
```

```
292     else if(state == 8'd15)
293      state <= 8'd5;          // fin escritura cs_adc1.
294
295     else if(state == 8'd16)
296      state <= 8'd17;         // adc2. esperamos a adc_read.
297                 // (la dirección más el registro de conversión).
298     else if(state == 8'd17)
299      state <= 8'd18;         // se inicia la escritura en el conversion register.
300     else if(state == 8'd18)
301      state <= 8'd19;         // finaliza la escritura en el conversion register.
302     else if(state == 8'd19)
303      state <= 8'd20;         // inicio up todos los cs.
304     else if(state == 8'd20)
305      state <= 8'd21;         // fin up todos los cs.
306     else if(state == 8'd21)
307      state <= 8'd22;         // esperamos a que eoc2 = 0
308     else if(state == 8'd22)
309      state <= 8'd23;         // inicio escritura cs_adc2
310     else if(state == 8'd23)
311      state <= 8'd5;          // fin escritura cs_adc2.
312
313     else
314      state <= state;
315    end
316    else
317     state <= state;
318   end
319
320 endmodule
```

```
1  module config(
2      input sclk,
3      input [7:0] din_word,
4      input lclk,
5      input control_done,
6
7      output reg [7:0] opamp,
8      output reg rst,
9
10     output [79:0] upi,
11     output [79:0] downi,
12     output reg [7:0] upi_states,
13     output reg [7:0] downi_states,
14     output reg [7:0] iidle,
15     output reg [7:0] divider_i,
16     output reg [7:0] divider_i2,
17
18     output [79:0] upr,
19     output [79:0] downr,
20     output reg [7:0] upr_states,
21     output reg [7:0] downr_states,
22     output reg [7:0] ridle,
23     output reg [7:0] divider_r,
24     output reg [7:0] divider_r2,
25
26     output reg [7:0] adc_setup,
27     output reg [7:0] adc_aver,
28     output reg [7:0] adc_conv,
29     output adc_1,
30     output adc_2,
31     output adc_read,
32
33     output reconfig,
```

```
34
35      output done,
36
37      output [7:0] aux
38      );
39
40      ////////////////////
41      // wires y registros
42      ////////////////////
43
44      wire clk_en;
45
46      reg [7:0] state_next;
47
48      reg [7:0] state = 8'b0;
49
50      ////////////////////
51      // estados iniciales
52      ////////////////////
53
54      reg [7:0] upi_1 = 8'd115;
55      reg [7:0] upi_2 = 8'd118;
56      reg [7:0] upi_3 = 8'd121;
57      reg [7:0] upi_4 = 8'd124;
58      reg [7:0] upi_5 = 8'd127;
59      reg [7:0] upi_6 = 8'd128;
60      reg [7:0] upi_7 = 8'd131;
61      reg [7:0] upi_8 = 8'd134;
62      reg [7:0] upi_9 = 8'd137;
63      reg [7:0] upi_10 = 8'd140;
64
65      reg [7:0] downi_1 = 8'd140;
66      reg [7:0] downi_2 = 8'd137;
67      reg [7:0] downi_3 = 8'd134;
68      reg [7:0] downi_4 = 8'd131;
69      reg [7:0] downi_5 = 8'd128;
70      reg [7:0] downi_6 = 8'd127;
71      reg [7:0] downi_7 = 8'd124;
72      reg [7:0] downi_8 = 8'd121;
73      reg [7:0] downi_9 = 8'd118;
74      reg [7:0] downi_10 = 8'd115;
75
76      reg [7:0] upr_1 = 8'd115;
77      reg [7:0] upr_2 = 8'd118;
78      reg [7:0] upr_3 = 8'd121;
79      reg [7:0] upr_4 = 8'd124;
80      reg [7:0] upr_5 = 8'd127;
81      reg [7:0] upr_6 = 8'd128;
82      reg [7:0] upr_7 = 8'd131;
83      reg [7:0] upr_8 = 8'd134;
84      reg [7:0] upr_9 = 8'd137;
85      reg [7:0] upr_10 = 8'd140;
86
87      reg [7:0] downr_1 = 8'd140;
88      reg [7:0] downr_2 = 8'd137;
89      reg [7:0] downr_3 = 8'd134;
90      reg [7:0] downr_4 = 8'd131;
91      reg [7:0] downr_5 = 8'd128;
92      reg [7:0] downr_6 = 8'd127;
93      reg [7:0] downr_7 = 8'd124;
94      reg [7:0] downr_8 = 8'd121;
95      reg [7:0] downr_9 = 8'd118;
96      reg [7:0] downr_10 = 8'd115;
```

```
97
98      initial
99      begin
100         opamp = 8'b0;
101         rst = 1'b0;
102
103         upi_states = 8'd10;         // cantidad estados de subida imagen
104         downi_states = 8'd10;       // cantidad estados de bajada imagen
105         iidle = 8'd0;               // estado idle imagen
106         divider_i = 8'd10;          // divisor reloj imagen
107         divider_i2 = 8'd0;
108
109         upr_states = 8'd10;         // registro
110         downr_states = 8'd10;
111         ridle = 8'd0;
112         divider_r = 8'd10;
113         divider_r2 = 8'd0;
114
115         adc_setup = 8'b01_10_01_00; // clock mode 10, ext reference.
116         adc_aver = 8'b001_111_00;   // 32 conversions and return the average.
117         adc_conv = 8'b0;            // no scan, only n channel.
118    //este es el unico modo admitido. toda palabra debe ser de la forma 1_xxxx_11_x.
119     end
120
121     //////////////
122     // simulacion
123     //////////////
124
125     assign aux = opamp;
126
127     ////////////////////////
128     // always y asignaciones
129     ////////////////////////
130
131  assign clk_en = lclk;
132
133  assign done = (control_done && (state == 8'b0))?    1'b1 : 1'b0;
134
135  assign upi[39:0] = {upi_5, upi_4, upi_3, upi_2, upi_1};
136  assign upi[79:40] = {upi_10, upi_9, upi_8, upi_7, upi_6};
137
138  assign downi[39:0] = {downi_5, downi_4, downi_3, downi_2, downi_1};
139  assign downi[79:40] = {downi_10, downi_9, downi_8, downi_7, downi_6};
140
141  assign upr[39:0] = {upr_5, upr_4, upr_3, upr_2, upr_1};
142  assign upr[79:40] = {upr_10, upr_9, upr_8, upr_7, upr_6};
143
144  assign downr[39:0] = {downr_5, downr_4, downr_3, downr_2, downr_1};
145  assign downr[79:40] = {downr_10, downr_9, downr_8, downr_7, downr_6};
146
147     always @*
148     begin
149         if(din_word == 8'd1)        state_next = 8'd1;      // opamp
150         else if(din_word == 8'd2)   state_next = 8'd2;      // rst
151         else if(din_word == 8'd3)   state_next = 8'd3;      // daci
152         else if(din_word == 8'd4)   state_next = 8'd27;     // dacr
153         else if(din_word == 8'd5)   state_next = 8'd51;     // adc config
154         else if(din_word == 8'd6)   state_next = 8'd53;     // reconfiguracion.
155         else if(din_word == 8'd7)   state_next = 8'd54;     // adc1 meas
156         else if(din_word == 8'd8)   state_next = 8'd55;     // adc2 meas
157         else if(din_word == 8'd9)   state_next = 8'd58;     // divider_i
158         else if(din_word == 8'd10)  state_next = 8'd59;     // divider_r
159
```

```
160         else                        state_next = 8'b0;
161     end
162
163     always @(negedge sclk)
164     begin
165         if(clk_en)
166         begin
167             if(state == 8'd0)               state <= state_next; // idle
168
169             else if(state == 8'd1)          state <= 8'd0; // opamp
170
171             else if(state == 8'd2)          state <= 8'd0; // rst
172
173             else if(state == 8'd3)          state <= 8'd4; // upi_1
174             else if(state == 8'd4)          state <= 8'd5; // upi_2
175             else if(state == 8'd5)          state <= 8'd6; // upi_3
176             else if(state == 8'd6)          state <= 8'd7; // upi_4
177             else if(state == 8'd7)          state <= 8'd8; // upi_5
178             else if(state == 8'd8)          state <= 8'd9; // upi_6
179             else if(state == 8'd9)          state <= 8'd10; // upi_7
180             else if(state == 8'd10)         state <= 8'd11; // upi_8
181             else if(state == 8'd11)         state <= 8'd12; // upi_9
182             else if(state == 8'd12)         state <= 8'd13; // upi_10
183             else if(state == 8'd13)         state <= 8'd14; // downi_1
184             else if(state == 8'd14)         state <= 8'd15; // downi_2
185             else if(state == 8'd15)         state <= 8'd16; // downi_3
186             else if(state == 8'd16)         state <= 8'd17; // downi_4
187             else if(state == 8'd17)         state <= 8'd18; // downi_5
188             else if(state == 8'd18)         state <= 8'd19; // downi_6
189             else if(state == 8'd19)         state <= 8'd20; // downi_7
190             else if(state == 8'd20)         state <= 8'd21; // downi_8
191             else if(state == 8'd21)         state <= 8'd22; // downi_9
192             else if(state == 8'd22)         state <= 8'd23; // downi_10
193             else if(state == 8'd23)         state <= 8'd24; // upi_states
194             else if(state == 8'd24)         state <= 8'd25; // downi_states
195             else if(state == 8'd25)         state <= 8'd26; // iidle
196             else if(state == 8'd26)         state <= 8'd58; // divider_i
197
198             else if(state == 8'd27)         state <= 8'd28; // upr_1
199             else if(state == 8'd28)         state <= 8'd29; // upr_2
200             else if(state == 8'd29)         state <= 8'd30; // upr_3
201             else if(state == 8'd30)         state <= 8'd31; // upr_4
202             else if(state == 8'd31)         state <= 8'd32; // upr_5
203             else if(state == 8'd32)         state <= 8'd33; // upr_6
204             else if(state == 8'd33)         state <= 8'd34; // upr_7
205             else if(state == 8'd34)         state <= 8'd35; // upr_8
206             else if(state == 8'd35)         state <= 8'd36; // upr_9
207             else if(state == 8'd36)         state <= 8'd37; // upr_10
208             else if(state == 8'd37)         state <= 8'd38; // downr_1
209             else if(state == 8'd38)         state <= 8'd39; // downr_2
210             else if(state == 8'd39)         state <= 8'd40; // downr_3
211             else if(state == 8'd40)         state <= 8'd41; // downr_4
212             else if(state == 8'd41)         state <= 8'd42; // downr_5
213             else if(state == 8'd42)         state <= 8'd43; // downr_6
214             else if(state == 8'd43)         state <= 8'd44; // downr_7
215             else if(state == 8'd44)         state <= 8'd45; // downr_8
216             else if(state == 8'd45)         state <= 8'd46; // downr_9
217             else if(state == 8'd46)         state <= 8'd47; // downr_10
218             else if(state == 8'd47)         state <= 8'd48; // upr_states
219             else if(state == 8'd48)         state <= 8'd49; // downr_states
220             else if(state == 8'd49)         state <= 8'd50; // ridle
221             else if(state == 8'd50)         state <= 8'd59; // divider_r
222
```

```
223             else if(state == 8'd51)         state <= 8'd52; // adc_setup
224             else if(state == 8'd52)         state <= 8'd0; // adc_aver
225
226             else if(state == 8'd53)         state <= 8'd0; // reconfig
227
228             else if(state == 8'd54)         state <= 8'd56; // adc_1 conversion register
229             else if(state == 8'd55)         state <= 8'd56; // adc_2 conversion register
230             else if(state == 8'd56)         state <= 8'd57; // adc read 1
231             else if(state == 8'd57)         state <= 8'd0; // adc read 2
232
233             // nuevos estados
234
235             else if(state == 8'd58)         state <= 8'd0; // divider_i2
236
237             else if(state == 8'd59)         state <= 8'd0; // divider_r2
238
239             else                            state <= state;
240         end
241
242         else    state <= state;
243     end
244
245     // asignaciones
246
247     assign reconfig =   (state == 8'd53)?   1'b1    :   1'b0;
248
249     assign adc_1 =      (state == 8'd54)?   1'b1    :   1'b0;
250     assign adc_2 =      (state == 8'd55)?   1'b1    :   1'b0;
251     assign adc_read =   (state == 8'd56)?   1'b1    :   1'b0;
252
253     always @(negedge sclk)
254     begin
255   opamp <= (clk_en && (state == 8'd1))?   din_word     :   opamp;
256
257   rst <= (clk_en && (state == 8'd2))?  din_word[0]   :   rst;
258
259   upi_1 <= (clk_en && (state == 8'd3))?   din_word     :   upi_1;
260   upi_2 <= (clk_en && (state == 8'd4))?   din_word     :   upi_2;
261   upi_3 <= (clk_en && (state == 8'd5))?  din_word      :   upi_3;
262   upi_4 <= (clk_en && (state == 8'd6))?  din_word      :   upi_4;
263   upi_5 <= (clk_en && (state == 8'd7))?  din_word      :   upi_5;
264   upi_6 <= (clk_en && (state == 8'd8))?  din_word      :   upi_6;
265   upi_7 <= (clk_en && (state == 8'd9))?  din_word      :   upi_7;
266   upi_8 <= (clk_en && (state == 8'd10))?  din_word     :   upi_8;
267   upi_9 <= (clk_en && (state == 8'd11))?  din_word     :   upi_9;
268   upi_10 <= (clk_en && (state == 8'd12))?  din_word     :   upi_10;
269   downi_1 <= (clk_en && (state == 8'd13))?  din_word     :   downi_1;
270   downi_2 <= (clk_en && (state == 8'd14))?  din_word     :   downi_2;
271   downi_3 <= (clk_en && (state == 8'd15))?  din_word     :   downi_3;
272   downi_4 <= (clk_en && (state == 8'd16))?  din_word     :   downi_4;
273   downi_5 <= (clk_en && (state == 8'd17))?  din_word     :   downi_5;
274   downi_6 <= (clk_en && (state == 8'd18))?  din_word     :   downi_6;
275   downi_7 <= (clk_en && (state == 8'd19))?  din_word     :   downi_7;
276   downi_8 <= (clk_en && (state == 8'd20))?  din_word     :   downi_8;
277   downi_9 <= (clk_en && (state == 8'd21))?  din_word     :   downi_9;
278   downi_10 <= (clk_en && (state == 8'd22))?  din_word     :   downi_10;
279   upi_states  <= (clk_en && (state == 8'd23))?  din_word     :   upi_states;
280   downi_states <= (clk_en && (state == 8'd24))?  din_word     :   downi_states;
281   iidle <= (clk_en && (state == 8'd25))?  din_word     :   iidle;
282   divider_i <= (clk_en && (state == 8'd26))?  din_word     :   divider_i;
283   divider_i2 <= (clk_en && (state == 8'd58))?  din_word     :   divider_i2;
284
285   upr_1 <= (clk_en && (state == 8'd27))?  din_word     :   upr_1;
```

```
286    upr_2 <= (clk_en && (state == 8'd28))?  din_word     :   upr_2;
287    upr_3 <= (clk_en && (state == 8'd29))?  din_word     :   upr_3;
288    upr_4 <= (clk_en && (state == 8'd30))?  din_word     :   upr_4;
289    upr_5 <= (clk_en && (state == 8'd31))?  din_word     :   upr_5;
290    upr_6 <= (clk_en && (state == 8'd32))?  din_word     :   upr_6;
291    upr_7 <= (clk_en && (state == 8'd33))?  din_word     :   upr_7;
292    upr_8 <= (clk_en && (state == 8'd34))?  din_word     :   upr_8;
293    upr_9 <= (clk_en && (state == 8'd35))?  din_word     :   upr_9;
294    upr_10 <= (clk_en && (state == 8'd36))?  din_word     :   upr_10;
295    downr_1 <= (clk_en && (state == 8'd37))?  din_word     :   downr_1;
296    downr_2 <= (clk_en && (state == 8'd38))?  din_word     :   downr_2;
297    downr_3 <= (clk_en && (state == 8'd39))?  din_word     :   downr_3;
298    downr_4 <= (clk_en && (state == 8'd40))?  din_word     :   downr_4;
299    downr_5 <= (clk_en && (state == 8'd41))?  din_word     :   downr_5;
300    downr_6 <= (clk_en && (state == 8'd42))?  din_word     :   downr_6;
301    downr_7 <= (clk_en && (state == 8'd43))?  din_word     :   downr_7;
302    downr_8 <= (clk_en && (state == 8'd44))?  din_word     :   downr_8;
303    downr_9 <= (clk_en && (state == 8'd45))?  din_word     :   downr_9;
304    downr_10 <= (clk_en && (state == 8'd46))?  din_word     :   downr_10;
305    upr_states  <= (clk_en && (state == 8'd47))?  din_word     :   upr_states;
306    downr_states <= (clk_en && (state == 8'd48))?  din_word     :   downr_states;
307    ridle <= (clk_en && (state == 8'd49))?  din_word     :   ridle;
308    divider_r <= (clk_en && (state == 8'd50))?  din_word     :   divider_r;
309    divider_r2 <= (clk_en && (state == 8'd59))?  din_word     :   divider_r2;
310
311    adc_setup <= (clk_en && (state == 8'd51))?  din_word     :   adc_setup;
312    adc_aver <= (clk_en && (state == 8'd52))?  din_word     :   adc_aver;
313
314    adc_conv <= (clk_en && ((state == 8'd54) || (state == 8'd55)))?
315     din_word     :   adc_conv;
316      end
317  endmodule
```

```
1   module reg_spi(
2    input clk,
3    input load,
4    input [3:0] mode,
5    output done,
6
7    input [23:0] reg_word,
8    output reg_lclk,
9    output reg_sclk,
10   output reg_rst,
11   output reg_out,
12
13   input [7:0] spi_word_8,
14   input [8:0] spi_word_9,
15   input spi_dout,
16   output [15:0] spi_fifo,
17   output spi_din,
18   output spi_sclk,
19   output [15:0] aux
20   );
21
22   //////////////
23   // comentarios
24   //////////////
25
26   // clk : clock del shift register (<=10 mhz).
27   // load: carga la palabra en el flanco de subida.
28   // x_word_i : palabra a escribir.
29   // mode:
30   //      0 -> escribe solo el registro
```

```
31  //      1 -> escribe el registro mas el spi_8
32  //      2 -> escribe el registro mas el spi_9
33  //      3 -> se escribe el registro y se lee el spi
34  //      4 -> se escribe el spi_8
35  //      5 -> se escribe el spi_9
36  //      6 -> se lee el spi
37
38
39  ////////////////////
40  // wires y registros
41  ////////////////////
42
43  reg [7:0] state = 8'b0;
44  wire [7:0] state_ini;
45  wire [7:0] state_next;
46  wire clk_en;
47
48  wire reg_load;
49  wire reg_done;
50  wire [15:0] reg_aux;
51
52  wire spi_load;
53  wire spi_done;
54  wire [1:0] spi_mode;
55  wire [15:0] spi_aux;
56
57  ////////////////////
58  // modulos
59  ////////////////////
60
61  shift_register shift_register(
62   .clk(clk),
63   .load(reg_load),
64   .word(reg_word),
65   .out(reg_out),
66   .rst(reg_rst),
67   .lclk(reg_lclk),
68   .sclk(reg_sclk),
69   .done(reg_done),
70   .aux(reg_aux)
71   );
72
73  spi spi(
74   .clk(clk),
75   .load(spi_load),
76   .dout(spi_dout),
77   .mode(spi_mode),
78   .word_9(spi_word_9),
79   .word_8(spi_word_8),
80   .sclk(spi_sclk),
81   .din(spi_din),
82   .done(spi_done),
83   .fifo(spi_fifo),
84   .aux(spi_aux)
85  );
86
87  //////////////
88  // simulacion
89  //////////////
90
91  assign aux = {8'b0, state};
92
93
```

```
94   ////////////////////////
95   // always y asignaciones
96   ////////////////////////
97
98   assign  clk_en = (state == 8'd0)?    1'b1 : 1'bz,
99     clk_en = (state == 8'd1)?    reg_done : 1'bz,
100    clk_en = (state == 8'd2)?    spi_done : 1'bz,
101    clk_en = (state >= 8'd3)?    1'b1 : 1'bz;
102
103  assign done = (state == 8'd3)? 1'b1 : 1'b0;
104
105  assign state_ini = (mode < 4'd4)?  8'd1 : 8'bzzzzzzzz,
106      state_ini = (mode > 4'd3)?  8'd2 : 8'bzzzzzzzz;
107
108  assign state_next = (mode == 4'd0)? 8'd3 : 8'd2;
109
110  assign reg_load = (state == 8'd1)? 1'b1 : 1'b0;
111
112  assign spi_load = (state == 8'd2)? 1'b1 : 1'b0;
113
114  assign  spi_mode = ((mode == 4'd1) || (mode == 4'd4))?  2'd0 : 2'bzz,
115    spi_mode = ((mode == 4'd2) || (mode == 4'd5))?  2'd1 : 2'bzz,
116    spi_mode = ((mode == 4'd3) || (mode == 4'd6))?  2'd2 : 2'bzz,
117    spi_mode = ((mode == 4'd0) || (mode > 4'd6))?   2'd3 : 2'bzz;
118
119  always @(posedge clk)
120  begin
121   if(clk_en)
122   begin
123    if(load == 1)
124    begin
125     if(state == 8'd0)
126      state <= state_ini;
127
128     else if(state == 8'd1)  // se escribe el registro.
129      state <= state_next;
130
131
132     else if(state == 8'd2) // se escribe el spi. spi_load <= 1.
133      state <= 8'd3;
134
135     else if(state == 8'd3)   // se espera a que load baje a 0
136      state <= 8'd3;
137    end
138
139    else
140    begin
141     state <= 8'b0;
142    end
143   end
144
145   else
146    state <= state;
147  end
148
149  endmodule
```

```
1   module dac3(
2       input clk,
3       input [15:0] divider,
4       input en,
5       input set,
6       input [79:0] up,
```

```
7       input [79:0] down,
8       input [7:0] up_states,
9       input [7:0] down_states,
10      input [7:0] idle,
11      output [7:0] out,
12      output [15:0] aux
13      );
14
15      ////////////////////
16      // wires y registros
17      ////////////////////
18
19      wire clk_out;
20
21      parameter mid = 8'd10;
22
23      reg [7:0] state = 8'd19;
24      reg clk_en = 8'b0;
25      wire [7:0] state_next;
26
27      wire up_result;
28      wire down_result;
29      wire mid_result;
30
31      wire [7:0] down_aux;
32      wire [7:0] up_aux;
33
34      ///////////
35      // modulos
36      ///////////
37
38      clk_divider2 clk_div(
39          .clk_in(clk),
40          .divider(divider),
41          .en(clk_en),
42          .clk_out(clk_out),
43          .aux(aux)
44          );
45
46      comparator up_comp(
47          .a(state),
48          .b(up_aux),
49          .lower(up_result)
50          );
51
52      comparator down_comp(
53          .a(state),
54          .b(down_aux),
55          .lower(down_result)
56          );
57
58
59      ////////////////////////
60      // always y asignaciones
61      ////////////////////////
62
63      assign mid_result = (state < mid)?      1'b1 : 1'b0;
64
65      assign up_aux = up_states - 8'b1;
66
67      assign down_aux = down_states + 8'd9;
68
69      assign  state_next = ((up_result == 0) && (down_result == 0) && (mid_result == 0))?
```

```
70      8'd0            :   8'bzzzz_zzzz,
71              state_next = ((up_result == 0) && (down_result == 0) && (mid_result == 1))?
72      8'd0            :   8'bzzzz_zzzz,
73              state_next = ((up_result == 0) && (down_result == 1) && (mid_result == 0))?
74      (state + 8'b1)  :   8'bzzzz_zzzz,
75              state_next = ((up_result == 0) && (down_result == 1) && (mid_result == 1))?
76      8'd10           :   8'bzzzz_zzzz,
77              state_next = ((up_result == 1) && (down_result == 0) && (mid_result == 0))?
78      8'd0            :   8'bzzzz_zzzz,
79              state_next = ((up_result == 1) && (down_result == 0) && (mid_result == 1))?
80      8'd0            :   8'bzzzz_zzzz,
81              state_next = ((up_result == 1) && (down_result == 1) && (mid_result == 0))?
82      8'd0            :   8'bzzzz_zzzz,
83              state_next = ((up_result == 1) && (down_result == 1) && (mid_result == 1))?
84      (state + 8'b1)  :   8'bzzzz_zzzz;
85
86      always @(posedge clk)
87      begin
88          if(up_result && en)
89              clk_en <= set;
90          else if(~up_result && down_result && en)
91              clk_en <= ~set;
92          else if(~up_result && ~down_result && en)
93              clk_en <= set;
94          else
95              clk_en <= 1'b0;
96      end
97
98      assign  out = ((state == 8'd0) && (en == 1))?
99      up[7:0]         :       8'bzzzz_zzzz,
100             out = ((state == 8'd1) && (en == 1))?
101     up[15:8]        :       8'bzzzz_zzzz,
102             out = ((state == 8'd2) && (en == 1))?
103     up[23:16]       :       8'bzzzz_zzzz,
104             out = ((state == 8'd3) && (en == 1))?
105     up[31:24]       :       8'bzzzz_zzzz,
106             out = ((state == 8'd4) && (en == 1))?
107     up[39:32]       :       8'bzzzz_zzzz,
108             out = ((state == 8'd5) && (en == 1))?
109     up[47:40]       :       8'bzzzz_zzzz,
110             out = ((state == 8'd6) && (en == 1))?
111     up[55:48]       :       8'bzzzz_zzzz,
112             out = ((state == 8'd7) && (en == 1))?
113     up[63:56]       :       8'bzzzz_zzzz,
114             out = ((state == 8'd8) && (en == 1))?
115     up[71:64]       :       8'bzzzz_zzzz,
116             out = ((state == 8'd9) && (en == 1))?
117     up[79:72]       :       8'bzzzz_zzzz,
118
119             out = ((state == 8'd10) && (en == 1))?
120     down[7:0]       :       8'bzzzz_zzzz,
121             out = ((state == 8'd11) && (en == 1))?
122     down[15:8]      :       8'bzzzz_zzzz,
123             out = ((state == 8'd12) && (en == 1))?
124     down[23:16]     :       8'bzzzz_zzzz,
125             out = ((state == 8'd13) && (en == 1))?
126     down[31:24]     :       8'bzzzz_zzzz,
127             out = ((state == 8'd14) && (en == 1))?
128     down[39:32]     :       8'bzzzz_zzzz,
129             out = ((state == 8'd15) && (en == 1))?
130     down[47:40]     :       8'bzzzz_zzzz,
131             out = ((state == 8'd16) && (en == 1))?
132     down[55:48]     :       8'bzzzz_zzzz,
```

```
            out = ((state == 8'd17) && (en == 1))?
    down[63:56]     :       8'bzzzz_zzzz,
            out = ((state == 8'd18) && (en == 1))?
    down[71:64]     :       8'bzzzz_zzzz,
            out = ((state >= 8'd19) && (en == 1))?
    down[79:72]     :       8'bzzzz_zzzz,

            out = (en == 0)?
    idle            :       8'bzzzz_zzzz;

    always @(posedge clk_out)
    begin
        state <= state_next;
    end

endmodule
```

```
module clk_divider(
 input clk_in,
 input [7:0] divider,
 output clk_out,
 output [7:0] aux
 );

 ////////////////////
 // wires y registros
 ////////////////////

 reg [7:0] state_par = 8'b0;
 reg [7:0] state_impar = 8'b0;

 wire clk_par;
 wire clk_impar;

 wire result_par;
 wire result_par2;
 wire result_impar;
 wire result_impar2;

 wire [7:0] divdiv;

 ////////////
 // modulos
 ////////////

 comparator comparador_par(
  .a(state_par),
  .b(divider),
  .lower(result_par)    // a<b
 );

 comparator comparador_par2(
  .a(state_par),
  .b(divdiv),
  .lower(result_par2)    // a<b
 );

 comparator comparador_impar(
  .a(state_impar),
  .b(divider),
  .lower(result_impar)    // a < b
 );

```

```
47   comparator comparador_impar2(
48    .a(state_impar),
49    .b(divdiv),
50    .lower(result_impar2)    // a < b
51   );
52
53   //////////////
54   // simulacion
55   //////////////
56
57   assign clk_impar =  (result_par2 || result_impar2)?     1'b1    :   1'b0;
58   assign clk_par =    (result_par2)?     1'b1    :   1'b0;
59
60   assign aux = {state_par[3:0], state_impar[3:0]};
61
62   ////////////////////////
63   // always y asignaciones
64   ////////////////////////
65
66   assign divdiv = {1'b0, divider[7:1]} + 8'b1; // division por 2
67
68   assign clk_out = (divider <= 8'd1)?       clk_in      :   1'bz,
69     clk_out = ((divider > 8'd1) && (divider[0] == 1'b0))? clk_par     :   1'bz,
70     clk_out = ((divider > 8'd1) && (divider[0] == 1'b1))? clk_impar   :   1'bz;
71
72   always @(posedge clk_in)
73   begin
74    state_par <= (result_par)?  (state_par + 8'b1) : 8'b1;
75   end
76
77   always @(negedge clk_in)
78   begin
79    state_impar <=  (result_impar)?  (state_impar + 8'b1) : 8'b1;
80   end
81
82  endmodule
```

```
1   module clk_divider2(
2    input clk_in,
3    input [15:0] divider,
4    input en,
5    output clk_out,
6    output [15:0] aux
7    );
8
9    ////////////////////
10   // wires y registros
11   ////////////////////
12
13   reg [15:0] state_par = 16'b0;
14   reg [15:0] state_impar = 16'b0;
15
16   reg clk_en_par;
17   reg clk_en_impar;
18
19   reg clk_par = 1'b0;
20   reg clk_impar = 1'b0;
21
22   wire result_par;
23   wire result_par2;
24   wire result_impar;
25   wire result_impar2;
26
```

```
27  wire [15:0] divdiv;
28
29  ////////////
30  // modulos
31  ////////////
32
33  comparator2 comparador_par(
34   .a(state_par),
35   .b(divider),
36   .lower(result_par)    // a<b
37  );
38
39  comparator2 comparador_par2(
40   .a(state_par),
41   .b(divdiv),
42   .lower(result_par2)    // a<b
43  );
44
45  comparator2 comparador_impar(
46   .a(state_impar),
47   .b(divider),
48   .lower(result_impar)    // a < b
49  );
50
51  comparator2 comparador_impar2(
52   .a(state_impar),
53   .b(divdiv),
54   .lower(result_impar2)    // a < b
55  );
56
57  //////////////
58  // simulacion
59  //////////////
60
61  assign aux = {state_par[7:0], 7'b0, clk_out};
62
63  ////////////////////////
64  // always y asignaciones
65  ////////////////////////
66
67  always @(posedge clk_in)
68  begin
69   clk_impar <= result_par2 | result_impar2;
70   clk_par <= result_par2;
71  end
72
73
74  assign divdiv = {1'b0, divider[15:1]} + 16'b1; // division por 2
75
76  always @*
77  begin
78   if(result_par == 1'b0 || state_par == 16'b0)
79    clk_en_par = en;
80   else
81    clk_en_par = 1'b1;
82  end
83
84  always @*
85  begin
86   if(result_impar == 1'b0 || state_impar == 16'b0)
87    clk_en_impar = en;
88   else
89    clk_en_impar = 1'b1;
```

```
 end

 assign  clk_out = (divider <= 16'd1)?       (clk_in & en) :   1'bz,
   clk_out = ((divider > 16'd1) && (divider[0] == 1'b0))? clk_par   :   1'bz,
   clk_out = ((divider > 16'd1) && (divider[0] == 1'b1))? clk_impar  :   1'bz;

 always @(posedge clk_in)
 begin
  if(clk_en_par)      state_par <= (result_par)?  (state_par + 16'b1) : 16'b1;
  else                state_par <= state_par;
 end

 always @(negedge clk_in)
 begin
  if(clk_en_impar)
   state_impar <=  (result_impar)?  (state_impar + 16'b1) : 16'b1;
  else
   state_impar <= state_impar;
 end

endmodule
```

```
module comparator(
 input wire [7:0] a,
 input wire [7:0] b,
 output reg equal,
 output reg lower,
 output reg greater
 );

 always @*
 begin
  if (a < b)
  begin
   equal = 0;
   lower = 1;
   greater = 0;
  end
  else if (a==b)
  begin
   equal = 1;
   lower = 0;
   greater = 0;
  end
  else
  begin
   equal = 0;
   lower = 0;
   greater = 1;
  end
 end

endmodule
```

```
module comparator2(
    input [15:0] a,
    input [15:0] b,
    output reg equal,
    output reg lower,
    output reg greater
    );

 always @*
 begin
```

```
    if (a < b)
    begin
     equal = 0;
     lower = 1;
     greater = 0;
    end
    else if (a==b)
    begin
     equal = 1;
     lower = 0;
     greater = 0;
    end
    else
    begin
     equal = 0;
     lower = 0;
     greater = 1;
    end
   end

endmodule
```

```
 module shift_register(
 input clk,
 input load,
 input [23:0] word,
 output out,
 output rst,
 output lclk,
 output sclk,
 output done,
 output [15:0] aux
 );

 //////////////
 // comentarios
 //////////////

 // clk : clock del shift register (<=10 mhz).
 // load: carga la palabra en el flanco de subida.
 // word : palabra a escribir y setear en el shift register.
 // out : salida serial. comienza con el lsb y termina con el msb
 // en : enable de las salidas (negado) esta señal sera externa.
 // rst: reset de los ff (negado).
 // lclk: latch que setea los latch de salida.
 // sclk: shift clock.
 // done: señal de terminado o inactividad.

 ////////////////////
 // wires y registros
 ////////////////////

 parameter state_aux = 6'd50;

 reg [5:0] state_next = 6'b0;
 reg [5:0] state = 6'b0;
 reg [23:0] s_reg = 24'b0;

 ////////////////////
 // estados iniciales
 ////////////////////

 initial
```

```
42  begin
43
44  end
45
46  //////////////
47  // simulacion
48  //////////////
49
50  assign aux = {10'b0, state};
51
52  ////////////////////////
53  // always y asignaciones
54  ////////////////////////
55
56  always @(posedge clk)
57  begin
58   if(load == 1)
59   begin
60    if(state == 6'b0)
61    begin
62     s_reg <= word;
63     state_next <= 6'd1;
64     state <= state_aux;
65    end
66
67    else if(state == 6'd1)
68    begin
69     s_reg <= s_reg;
70     state_next <= state_next;
71     state <= 6'd2;
72    end
73
74    else if((state> 6'b1) && (state < 6'd48) && (state[0] == 1'b0))
75    begin
76     s_reg <= {1'b0,s_reg[23:1]};
77     state_next <= state + 6'b1;
78     state <= state_aux;
79    end
80
81    else if((state> 6'b1) && (state < 6'd48) && (state[0] == 1'b1))
82    begin
83     s_reg <= s_reg;
84     state_next <= state_next;
85     state <= state + 6'b1;
86    end
87
88    else if(state == 6'd48)
89    begin
90     s_reg <= s_reg;
91     state_next <= state_next;
92     state <= 6'd49;
93    end
94
95    else if(state == state_aux)
96    begin
97     s_reg <= s_reg;
98     state_next <= state_next;
99     state <= state_next;
100   end
101
102   else
103   begin
104    s_reg <= s_reg;
```

```
105       state_next <= state_next;
106       state <= 6'd49;
107      end
108     end
109
110     else
111     begin
112      s_reg <= 24'b0;
113      state_next <= 6'b0;
114      state <= 6'd0;
115     end
116    end
117
118    assign rst = 1;
119    assign out = s_reg[0];
120    assign sclk = ((state[0] == 1'b1) && (state < 6'd48))?  1'b1 : 1'b0;
121    assign lclk = ((state == 6'd48) || (state == 6'd49))?     1'b1 : 1'b0;
122
123    assign done =   (state == 6'd49)?   1'b1 : 1'b0;
124   endmodule
```

```
1   module SPI(
2    input clk,
3    input load,
4    input dout,
5    input [1:0] mode,
6    input [8:0] word_9,
7    input [7:0] word_8,
8    output sclk,
9    output din,
10   output done,
11   output reg [15:0] fifo,
12   output [15:0] aux
13   );
14
15   ///////////////
16   // Comentarios
17   ///////////////
18
19   // clk: Clock de entrada (>=10 MHz).
20   // load: cargar palabra
21   // mode: 0 -> 8 bits, 1 -> 9 bits, 2 -> read.
22   // word_i: palabra de i bits a escribir (MSB primero)
23   // sclk: reloj del SPI
24   // din: palabra de salida.
25   // done: señal de terminado o inactividad.
26   // dout: palabra de entrada. Each result contains 2 bytes, with the
27   //       MSB preceded by four leading zeros. After each falling
28   //       edge of CS, the oldest available byte of data is available at
29   //       DOUT, MSB first. When the FIFO is empty, DOUT is zero.
30   //       Serial Data Output. Data is clocked out on the falling edge of
31   //       SCLK. High impedance when CS is connected to VDD.
32
33   ////////////////////
34   // Wires y Registros
35   ////////////////////
36
37   parameter state_aux = 8'd20;
38   parameter state_auxr = 8'd34;
39
40   reg [7:0] state = 8'b0;
41   reg [7:0] state_r = 8'b0;
42
```

```
43  reg [7:0] state_next = 8'b0;
44  reg [7:0] state_nextr = 8'b0;
45
46  reg [8:0] s_reg = 9'b0;
47
48  wire sclk_8, sclk_9, sclk_r;
49  wire done_8, done_9, done_r;
50
51  ////////////////////
52  // Estados Iniciales
53  ////////////////////
54
55  initial
56  begin
57   fifo = 0;
58  end
59
60  //////////////
61  // Simulacion
62  //////////////
63
64  assign aux = {8'b0, state_r};
65
66  ////////////////////////
67  // Always y Asignaciones
68  ////////////////////////
69
70  always @(posedge clk)
71  begin
72   if(load == 1'b1)
73    begin
74    if(mode == 2'd0) // 8 bits
75    begin
76     if(state == 8'b0)
77     begin
78      s_reg <= {word_8, 1'b0};
79      state <= state_aux;
80      state_next <= 8'd1;
81     end
82
83     else if(state == 8'b1)
84     begin
85      s_reg <= s_reg;
86      state <= 8'd2;
87      state_next <= state_next;
88     end
89
90     else if((state> 8'b1) && (state < 8'd16) && (state[0] == 1'b0))
91     begin
92      s_reg <= {s_reg[7:0], 1'b0};
93      state <= state_aux;
94      state_next <= state + 8'b1;
95     end
96
97     else if((state> 8'b1) && (state < 8'd16) && (state[0] == 1'b1))
98     begin
99      s_reg <= s_reg;
100     state <= state + 8'b1;
101     state_next <= state_next;
102    end
103
104    else if(state == 8'd16)
105    begin
```

```
106         s_reg <= s_reg;
107         state <= 8'd17;
108         state_next <= state_next;
109        end
110
111        else if(state == state_aux)
112        begin
113         s_reg <= s_reg;
114         state <= state_next;
115         state_next <= state_next;
116        end
117
118        else
119        begin
120         s_reg <= s_reg;
121         state <= 8'd17;
122         state_next <= state_next;
123        end
124       end
125
126       else if(mode == 2'd1) // 9 bits
127       begin
128        if(state == 8'b0)
129        begin
130         s_reg <= word_9;
131         state <= state_aux;
132         state_next <= 8'd1;
133        end
134
135        else if(state == 8'b1)
136        begin
137         s_reg <= s_reg;
138         state <= 8'd2;
139         state_next <= state_next;
140        end
141
142        else if((state> 8'b1) && (state < 8'd18) && (state[0] == 1'b0))
143        begin
144         s_reg <= {s_reg[7:0], 1'b0};
145         state <= state_aux;
146         state_next <= state + 8'b1;
147        end
148
149        else if((state> 8'b1) && (state < 8'd18) && (state[0] == 1'b1))
150        begin
151         s_reg <= s_reg;
152         state <= state + 8'b1;
153         state_next <= state_next;
154        end
155
156        else if(state == 8'd18)
157        begin
158         s_reg <= s_reg;
159         state <= 8'd19;
160         state_next <= state_next;
161        end
162
163        else if(state == state_aux)
164        begin
165         s_reg <= s_reg;
166         state <= state_next;
167         state_next <= state_next;
168        end
```

```
169
170       else
171       begin
172        s_reg <= s_reg;
173        state <= 8'd19;
174        state_next <= state_next;
175       end
176      end
177
178      else // read
179      begin
180       s_reg <= 9'b0;
181       state <= 8'b0;
182       state_next <= 8'b0;
183      end
184     end
185
186     else
187     begin
188      s_reg <= s_reg;
189      state <= 8'b0;
190      state_next <= 9'b0;
191     end
192    end
193
194    always @(posedge clk)
195    begin
196     if((load == 1'b1) && (mode == 2'd2))
197     begin
198      if(state_r == 8'b0)
199      begin
200       fifo <= fifo;
201       state_r <= 8'd1;
202      end
203
204      else if(state_r == 8'b1)
205      begin
206       fifo <= {fifo[14:0], dout};
207       state_r <= 8'd2;
208      end
209
210      else if((state_r> 8'd1) && (state_r < 8'd32) && (state_r[0] == 1'b0))
211      begin
212       fifo <= fifo;
213       state_r <= state_r + 8'b1;
214      end
215
216      else if((state_r> 8'd1) && (state_r < 8'd32) && (state_r[0] == 1'b1))
217      begin
218       fifo <= {fifo[14:0], dout};
219       state_r <= state_r + 8'b1;
220      end
221
222      else if(state == 8'd32)
223      begin
224       fifo <= fifo;
225       state_r <= 8'd33;
226      end
227
228      else
229      begin
230       fifo <= fifo;
231       state_r <= 8'd33;
```

```
    end
   end

   else
   begin
    fifo <= fifo;
    state_r <= 8'b0;
   end
  end

  assign sclk_8 = ((state[0] == 1'b1) && (state < 8'd16))?  (mode == 4'b0000) : 1'b0;
  assign done_8 = (mode == 4'b0000)?    (state == 8'd17) : 1'b1;

  assign sclk_9 = ((state[0] == 1'b1) && (state < 8'd18))?  (mode == 4'b0001) : 1'b0;
  assign done_9 = (mode == 4'b0001)?    (state == 8'd19) : 1'b1;

  assign sclk_r = ((state_r[0] == 1'b1) && (state_r < 8'd32))?  (mode == 4'b0010) : 1'b0;
  assign done_r = (mode == 4'b0010)?    (state_r == 8'd33) : 1'b1;

  assign din = s_reg[8];
  assign done = done_8 & done_9 & done_r;
  assign sclk = sclk_8 | sclk_9 | sclk_r;

endmodule
```

```
module spi_config(
 input clk,
 input sclk,
 input din,
 input load_en,
 input [15:0] dout_word,
 output lclk,
 output [7:0] din_word,
 output dout,
 output done,
 output [15:0] aux
 );

 //////////////
 // comentarios
 //////////////

 // sclk : set clock
 // din : palabra de entrada (8 bits). msb primero
 // dout_word : palabra de entrada que se va a escribir a la salida
 // lclk : load clock
 // din_word: palabra de salida que se escribio en la entrada.
 // dout : palabra de salida (8 bits). msb primero
 // load_en: señal para cargar la palabra leída por el spi.

 // cpol = cpha = 0

 ////////////////////
 // wires y registros
 ////////////////////

 reg [7:0] state_in = 8'b0;
 reg [7:0] reg_in = 8'b0;

 reg [7:0] state_out = 8'b0;
 reg [15:0] reg_out = 16'b0;

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

```
39  // simulacion
40  ///////////////
41
42  assign aux = {7'b0, state_in};
43
44  ////////////////////////
45  // always y asignaciones
46  ////////////////////////
47
48  assign din_word = reg_in;
49
50  assign done = ((state_in == 8'd0) || (state_in == 8'd8))?       1'b1 : 1'b0;
51
52  assign  lclk =  (state_in == 8'd8)?     1'b1 : 1'b0;
53
54  assign  dout =  (state_out == 8'd0)?    reg_out[15]  :   1'bz,
55    dout =  (state_out == 8'd1)?    reg_out[14]  :   1'bz,
56    dout =  (state_out == 8'd2)?    reg_out[13]  :   1'bz,
57    dout =  (state_out == 8'd3)?    reg_out[12]  :   1'bz,
58    dout =  (state_out == 8'd4)?    reg_out[11]  :   1'bz,
59    dout =  (state_out == 8'd5)?    reg_out[10]  :   1'bz,
60    dout =  (state_out == 8'd6)?    reg_out[9]   :   1'bz,
61    dout =  (state_out == 8'd7)?    reg_out[8]   :   1'bz,
62    dout =  (state_out == 8'd8)?    reg_out[7]   :   1'bz,
63    dout =  (state_out == 8'd9)?    reg_out[6]   :   1'bz,
64    dout =  (state_out == 8'd10)?   reg_out[5]   :   1'bz,
65    dout =  (state_out == 8'd11)?   reg_out[4]   :   1'bz,
66    dout =  (state_out == 8'd12)?   reg_out[3]   :   1'bz,
67    dout =  (state_out == 8'd13)?   reg_out[2]   :   1'bz,
68    dout =  (state_out == 8'd14)?   reg_out[1]   :   1'bz,
69    dout =  (state_out >= 8'd15)?   reg_out[0]   :   1'bz;
70
71  always @(posedge sclk)  // din
72  begin
73   reg_in <= {reg_in[6:0], din};
74   state_in <= (state_in <8'd8)? (state_in +8'b1) : 8'b1;
75  end
76
77  always @(negedge sclk)  // dout
78  begin
79   state_out <= (state_out <8'd15)? (state_out +8'b1) : 8'b0;
80  end
81
82  always @(posedge clk)
83  begin
84   reg_out <= (load_en)? dout_word : reg_out;
85  end
86
87 endmodule
```

B.2 Código C Microcontrolador

```
#include <msp430.h>

//////////////////////
// Variables Globales
//////////////////////

char RXBUF[30];

int READ1;
int READ2;

char aux;

void init(void){

 WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

 ///////////////
 // Osciladores
 ///////////////

 P2DIR |= BIT2; // P2.2 (SMCLK),
 P2SEL |= BIT2;

 P7SEL |= BIT7; // P7.7 (MCLK)
 P7DIR |= BIT7;

 P1SEL |= BIT0; // P1.0 (ACLK)
 P1DIR |= BIT0;

 // Configuración

 UCSCTL3 = SELREF__XT2CLK; // Select XT2
 P5SEL |= BIT2; // Set the PSEL bit for XT2IN

 UCSCTL6 &= ~(XT2OFF); // Enable XT2
 UCSCTL6 &= ~(XT2DRIVE0); // set the correct drive level for 4MHz

 UCSCTL4 |= SELA__DCOCLKDIV; // Cambiamos el ACLK de XT1 A DCOCLKDIV

 __bis_SR_register(SCG0); // Disable the FLL control loop

 UCSCTL0 = 0x0000; // Set lowest possible DCOx, MODx
 UCSCTL1 = DCORSEL_6; // Select DCO range 20MHz operation (6)

 UCSCTL2 = FLLD_0 + 4;  // Set DCO Multiplier for 20MHz
        // (N + 1) * FLLRef = fDCOCLKDIV
        // (4 + 1) * 4MHz = 20MHz
        // Set fDCOCLKDIV = fDCOCLK/1
 __bic_SR_register(SCG0); // Enable the FLL control loop

 // Worst-case settling time for the DCO when the DCO range bits have been
 // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
 // UG for optimization.
 // 32 x 32 x 4 MHz / 20 MHz = 205 = MCLK cycles for DCO to settle
 __delay_cycles(205);

 // Loop until XT1,XT2 & DCO fault flag is cleared
 do
 {
  UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
```

```
62    // Clear XT2,XT1,DCO fault flags
63    SFRIFG1 &= ~OFIFG; // Clear fault flags
64   }while (SFRIFG1&OFIFG); // Test oscillator fault flag
65
66   UCSCTL4 = SELA__XT2CLK | SELS__DCOCLKDIV | SELM__DCOCLKDIV;
67
68   ////////////////////////////////
69   // Puertos de Entrada y Salida
70   ////////////////////////////////
71
72   P2OUT = 0;
73   P2DIR &= ~BIT5;  // Done de FPGA P2.5
74   P2DIR |= BIT4;  // P2.4 como salida
75   P2OUT &= ~BIT4;
76
77   ////////////////////////////////////
78   // Configuración del puerto SPI A0
79   ////////////////////////////////////
80
81   //  P2.7 UCA0CLK
82   // P3.3 UCA0TXD
83   // P3.4 UCA0RXD
84
85   P2SEL |= BIT7; // UCA0CLK
86   P3SEL |= BIT3 | BIT4; //UCA0TXD | UCA0RXD
87
88   UCA0CTL0 |= UCMSB | UCMST | UCSYNC | UCCKPH;
89   // Phase = 1, Polarity = 0, MSB First, 8-bit, Master Mode, 3-Pin, Synchronous.
90   UCA0CTL1 |= UCSSEL__ACLK; // AMCLK Clock Source. (4MHz)
91
92   UCA0BR0 = 0x0A; // Prescaler Clock = (UCxxBR0 + UCxxBR1 * 256)
93   UCA0BR1 = 0x00;
94
95   UCA0CTL1 &= ~UCSWRST; // Se inicia la máquina de estados del módulo SPI.
96
97   ////////////////////////////////////
98   // Configuración del Puerto UART A1
99   ////////////////////////////////////
100
101  P2DIR |= BIT2; // P2.2 SMCLK set out to pins
102  P2SEL |= BIT2;
103
104  P4SEL |= BIT4 | BIT5; // P4.4 (TX), P4.5 (RX)
105  // El registro UCA1CTL0 Se deja nulo. Esto implica
106  //  Paridad desactivada
107  //  LSB First
108  //  8-bit
109  //  1 stop bit
110  //  Modo Uart y Asíncrono.
111
112  UCA1CTL1 |= UCSSEL_2; // SMCLK Clock Source (20MHz)
113  UCA1BR0 = 173; // 115200 (see User's Guide)
114  UCA1BR1 = 0; // COM4
115
116  UCA1MCTL |= UCBRS_5 + UCBRF_0; // Modulacion UCBRSx = 5 , UCBRFx = 0, UCOS16 = 0.
117
118  UCA1CTL1 &= ~UCSWRST; // **Initialize USCI state machine
119
120 }
121
122 int SPI_TXRX(unsigned char input){
123
124  UCA0TXBUF = input; // Enviamos el dato.
```

```
125
126   while(!(UCA0IFG & UCRXIFG));
127   // Esperamos a que el dato se envíe y se reciva correctamente.
128
129   int output = (int) UCA0RXBUF; // Leímos el buffer de entrada
130
131   return output;
132  }
133
134  void UART_TXchar(char c )
135  {
136   while (!(UCA1IFG & UCTXIFG));
137   UCA1TXBUF = c;
138  }
139
140  char UART_RXchar(){
141
142   char out;
143
144   while(!(UCA1IFG & UCRXIFG));
145    out = UCA1RXBUF;
146
147   return out;
148  }
149
150  void UART_TXstring(char* c)
151   {
152   int i;
153
154   for(i=0; i<strlen(c); i++){
155    UART_TXchar(*(c + i));
156   }
157  }
158
159  void UART_TXstring2(char* c, int count)
160  {
161   int i;
162
163   for(i=0; i<count; i++){
164    UART_TXchar(*(c + i));
165   }
166  }
167
168  void UART_RXstring(){
169
170   // Recibimos el primer byte que nos informa cuantos bytes recibiremos a futuro.
171   int count = (int) UART_RXchar();
172   UART_TXchar(0x55);
173
174   // Ahora recibimos los demás bytes y los guardamos en RXBUF que posee un tamaño de 30.
175   int i;
176   for(i=0; i<count; i++){
177    RXBUF[i] = UART_RXchar();
178    UART_TXchar(0x55);
179   }
180  }
181
182  void UART_Response(){
183
184   int i;
185
186   // Primero enviamos la función de la FPGA
187   SPI_TXRX(RXBUF[0]);
```

```
188
189   switch (RXBUF[0]) {
190   case 0x01: // Configure Opamp.
191    SPI_TXRX(RXBUF[1]);
192    break;
193
194   case 0x02: // rst
195    SPI_TXRX(RXBUF[1]);
196    break;
197
198   case 0x03: // DACI
199    for(i=1; i < 26; i++){
200     SPI_TXRX(RXBUF[i]);
201    }
202    break;
203
204   case 0x04: // DACR
205    for(i=1; i < 26; i++){
206     SPI_TXRX(RXBUF[i]);
207    }
208    break;
209
210   case 0x05: // ADC Config
211    SPI_TXRX(RXBUF[1]);
212    SPI_TXRX(RXBUF[2]);
213    break;
214
215   case 0x06: // Reconfiguración
216    SPI_TXRX(0x0A); // Se envía cualquier dato.
217    break;
218
219   case 0x07: // ADC1 Meas
220    SPI_TXRX(RXBUF[1]); // Se escribe el registro de conversión.
221
222    while(!(P2IN & BIT5)); // Esperamos a la FPGA
223
224    READ1 = SPI_TXRX(0x0A); // Leímos el primer byte.
225    READ2 = SPI_TXRX(0x0A); // Leímos el segundo byte.
226
227    UART_TXchar((unsigned char) READ1);
228    UART_TXchar((unsigned char) READ2);
229
230    break;
231   case 0x08:    // AD2 Meas
232    SPI_TXRX(RXBUF[1]); // Se escribe el registro de conversión.
233
234    while(!(P2IN & BIT5)); // Esperamos a la FPGA
235
236    READ1 = SPI_TXRX(0x0A); // Leímos el primer byte.
237    READ2 = SPI_TXRX(0x0A); // Leímos el segundo byte.
238
239    UART_TXchar((char) READ1);
240    UART_TXchar((char) READ2);
241
242    break;
243
244   default:
245     break;
246   }
247  }
248
249  void main(void){
250
```

```
251   init(); // Inicializamos el sistema
252
253   while(1){
254
255    // Esperamos a que la FPGA ya este inicializada.
256
257    //while(!(P2IN & BIT5));
258
259    UART_RXstring();
260    UART_Response();
261    P2OUT &= ~BIT4;
262
263    // Final
264
265    int i;
266    for(i=1; i < 150; i++){
267     __delay_cycles(100000);
268    }
269
270    UART_TXchar(0x55);}}
```