ORDENAMIENTO ÓPTIMO DE LAS FASES DE UN CONVERTIDOR MULTIFÁSICO DE POTENCIA MEDIANTE FPGA

Javier Hernán Moviglia

Este Trabajo Final de carrera fue presentado al Departamento de Electrónica de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata el 24 de agosto de 2020, como requisito parcial para la obtención del título de

Ingeniero en Electrónica

Director: Dr. Pablo Daniel Antoszczuk

Co-Directora: Dra. Paula Cervellini

ORDENAMIENTO ÓPTIMO DE LAS FASES DE UN CONVERTIDOR MULTIFÁSICO DE POTENCIA MEDIANTE FPGA

Javier Hernán Moviglia

Este Trabajo Final de carrera fue presentado al Departamento de Electrónica de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata el 24 de agosto de 2020, como requisito parcial para la obtención del título de

Ingeniero en Electrónica

Director: Dr. Pablo Daniel Antoszczuk

Co-Directora: Dra. Paula Cervellini

El presente trabajo de tesis fue realizado en el Laboratorio de Instrumentación y Control del Departamento de Electrónica, Facultad de Ingeniería, Universidad Nacional de Mar del Plata.

A mis padres

Índice general

Índice de tablas

Índice de figuras

Índice de algoritmos

Agradecimientos

Si bien es cierto que en toda carrera el mayor esfuerzo debe venir de uno mismo, debo reconocer que Dios ha puesto en mi lugar varias personas para que este trabajo se pudiese materializar. En última instancia es el reflejo de varios años de estudio y dedicación en mi transcurso en la carrera de ingeniería electrónica. Por ello mis agradecimientos a:

Mis padres por brindarme todos los recursos para que esto fuese posible, su motivación permanente para que diese siempre lo mejor de mí, su afecto, cariño y acompañamiento constante.

Mis amigos y compañeros de facultad con quienes no sólo intercambié información y conocimientos sino también experiencias juntos.

Mis directores de tesis, Dr. Pablo Daniel Antoszczuk y Dra. Paula Cervellini. Ellos me acompañaron en todo el desarrollo del trabajo, me enseñaron a usar herramientas y programas necesarios para su funcionamiento y estuvieron siempre predispuestos a aconsejarme y asesorarme. Además, su ayuda y consejos fueron clave para que el sueño de hacer una pasantía en Alemania se pudiese concretar.

Por último, quiero agradecer al Dr. Rogelio García, quien aportó valiosas correcciones y sugerencias en las etapas finales de este informe y la presentación del proyecto.

Javier Hernán Moviglia

Mar del Plata, 24 de agosto de 2020.

Resumen

Los convertidores DC/DC interleaved son muy utilizados cuando se requiere controlar elevadas corrientes de forma eficiente. Esto se debe a la posibilidad de dividir la corriente entre N fases. Sin embargo, las ventajas obtenidas mediante el uso de este tipo de convertidores, como reducción en el *ripple* total Δi_T y aumento de la frecuencia en N veces respecto a la frecuencia de conmutación, se ven deterioradas ante desbalances entre los inductores de fase. Dichos desbalances generan diferentes amplitudes en los *ripples* de fase, que al ser sumadas, dan lugar a un Δi_T de mayor amplitud y con componentes armónicas de menor frecuencia. Consecuentemente, las exigencias de filtrado se incrementan. Este problema puede ser mitigado a partir de un ordenamiento adecuado de las fases del convertidor. En Antoszczuk *et al.* [1, 2] se propone un método para hallar el ordenamiento óptimo de las fases de un convertidor operando en modo de conducción continua, de forma tal de minimizar el contenido de frecuencia de conmutación y sus armónicos en Δi_T. El ordenamiento se realiza empleando técnicas de algoritmos genéticos, a partir de las cuales se logra obtener una secuencia óptima que minimiza un funcional relacionado con la amplitud del ripple total. Esta propuesta fue validada a través de simulaciones.

En esta tesis se desarrolla la implementación práctica del método mencionado, la cual comprende el desarrollo del hardware de adquisición (acondicionamiento de señales y conversión analógica a digital), y el diseño de la plataforma digital en

una FPGA (modelado y descripción en VHDL). Por último, se presentan ensayos experimentales y de simulacion para validar la implementación propuesta.

Nomenclatura

$FPGA$	Arreglo de compuertas programable en campo *(Field Programmable Gate Array)*
HDL	Lenguaje de descripción de Hardware *(Hardware Description Language)*
$ASIC$	Circuito integrado para aplicaciones específicas *(Average Current Mode Control)*
CBL	Bloque lógico configurable *(Configurable Logic Block)*
$CMOS$	Semiconductor complementario de óxido metálico *(Complementary metal-oxide-semiconductor)*
$LVDS$	Señal diferencial de bajo voltaje *(low-voltage differential signaling)*
SPI	Interface periférica serie *(Serial Peripheral Interface)*
DCM	Gestor de reloj digital *(Digital Clock Manager)*
$PCMC$	Control modo corriente pico *(Peak Current Mode Control)*
$LFSR$	Registro de desplazamiento de realimentación lineal *(Linear Feedback Shift Register)*

Capítulo 1

Introducción

1.1. Introducción general

Un convertidor multifásico resulta de la asociación en paralelo de N convertidores iguales, los cuales individualmente se denominan fases. En la Fig. 1.1 se muestra un ejemplo de una estructura clásica de este tipo de convertidores: el convertidor buck multifásico. Como se puede apreciar, la corriente total está determinada por la suma de las corrientes de cada fase, i.e., $i_T = \sum_{x=1}^{N} i_x$. Por lo tanto, para una dada i_T, la corriente media en cada fase es inversamente proporcional a N, lo cual permite reducir las pérdidas por conducción y por conmutación de las llaves semiconductoras empleadas en cada fase[3, 4, 5]. Algunas de las aplicaciones en las que se los utiliza son: inversores conectados a la red, sistemas de almacenamiento para vehículos híbridos, módulos reguladores de tensión, física de altas energías, entre otros.

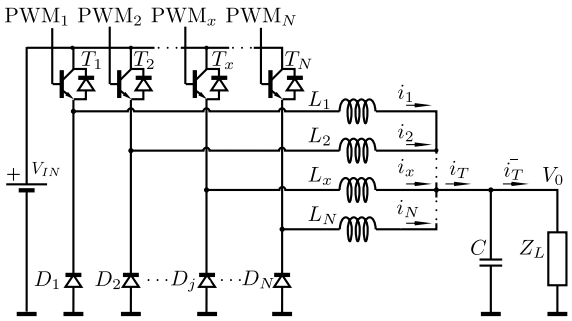
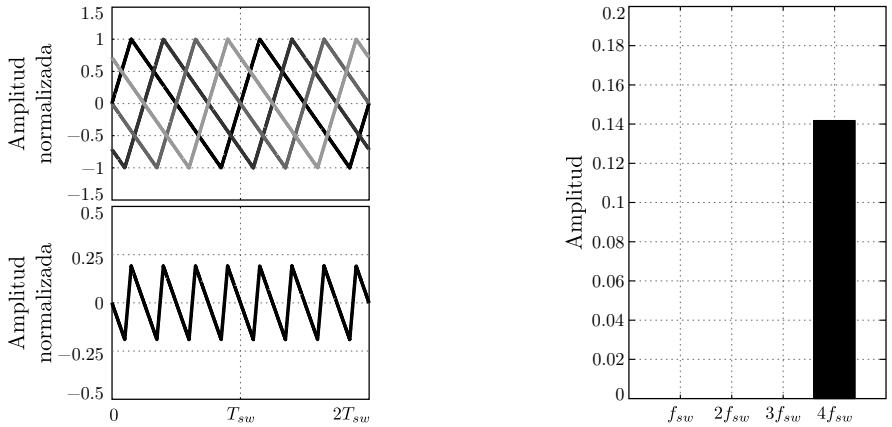


Figura 1.1: Convertidor buck multifásico

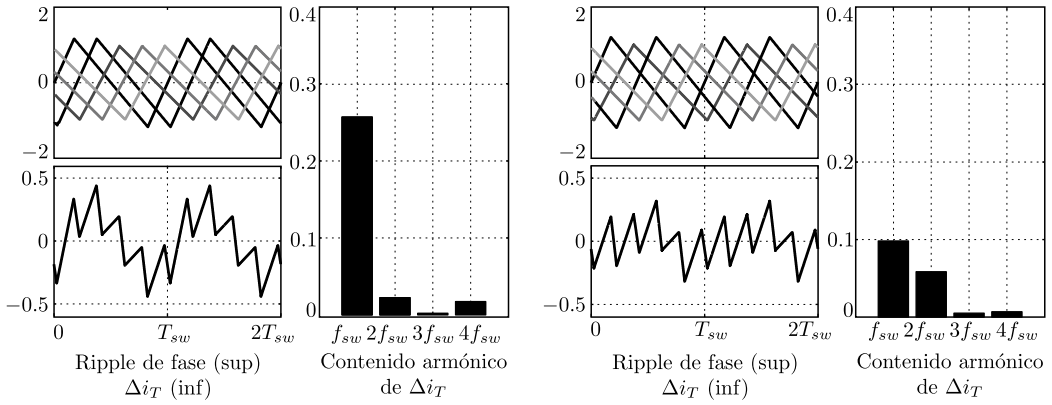
En caso de que la amplitud, el ciclo de trabajo y la frecuencia de conmutación, f_{sw}, del ripple de las diferentes fases sean iguales, condición denominada balanceada, un intercalado equidistante de las señales PWM de control de los dispositivos (2π/N) permite reducir al mínimo el ripple de la corriente total (Δi_T) e incrementar N veces su frecuencia ($N \cdot f_{sw}$). Esta característica es muy deseada debido a que, al incrementarse la frecuencia de Δi_T, se puede obtener un mejor filtrado de la corriente i_T o menores requerimientos sobre el capacitor de filtrado [6]. En la Fig. 1.2 se pueden observar los ripples de fase y el contenido armónico en condiciones balanceadas de un convertidor de 4 fases. Se debe notar que el ripple de la corriente total tiene sólo la componente de la frecuencia ($N \cdot f_{sw}$).



(a) Ripples de fase (sup); Δi_T (inf). (b) Contenido armónico de Δi_T.

Figura 1.2: Ripples de fase, Δi_T, y contenido armónico de Δi_T en condiciones ideales.

Sin embargo, las características del ripple total pueden alejarse del caso balanceado debido, principalmente, a dos factores: desfase no ideal entre fases y diferencias de amplitud entre los ripples de cada fase. Estos factores introducen en el ripple total componentes de la frecuencia de conmutación y sus armónicos incrementando así las exigencias del filtrado. El primer factor es despreciable si se utilizan llaves semiconductoras y drivers del mismo tipo y, además puede ser mitigado mediante técnicas de control. En relación a las diferencias de amplitud entre los ripples de fase, las mismas se producen principalmente por diferencias en el valor de los inductores de fase. Adicionalmente, como se describe en [2], el orden en el que son intercalados los ripples de fase influye en las características del ripple total. En la Fig. 1.3 se muestran los ripples de fase con dos ordenamientos distintos, en donde se observa un ripple total de menor amplitud en el segundo ordenamiento.



(a) Ordenamiento 1: ($A1$ $A2$ $A3$ $A4$ $A5$). (b) Ordenamiento 2: ($A1$ $A3$ $A2$ $A4$ $A5$).

Figura 1.3: Ripple total para distintos ordenamientos de los ripples de fase.

Llevar a cabo este ordenamiento a partir de evaluar todas las combinaciones posibles puede ser un trabajo muy intenso. La cantidad posible de permutaciones es igual al factorial del número fases ($N!$). Debido a la dificultad práctica de encontrar el caso óptimo, principalmente en convertidores con un número elevado de fases, en [1, 2] se desarrolla un método para encontrar el óptimo mediante el uso de algoritmos genéticos, reduciendo significativamente la cantidad de casos a analizar.

En este sentido, este proyecto final consiste en la implementación del algoritmo de reordenamiento en una FPGA y en el diseño y construcción del hardware de acondicionamiento y adquisición de señales. La implementación propuesta se ensaya en un convertidor interleaved de 8 fases, previamente desarrollado en el Laboratorio de Instrumentacion y Control.

1.2. Esquema general del sistema

La Fig. 1.4 muestra un esquema general del sistema en el que se observan cada uno de sus bloques principales.

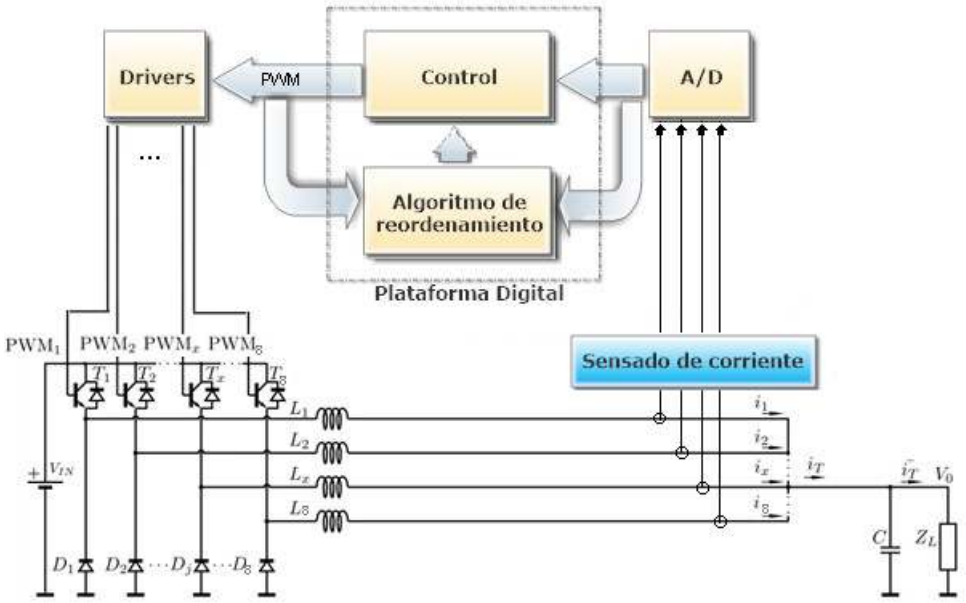


Figura 1.4: Esquema general del sistema.

Como se mencionó anteriormente, el convertidor utilizado para el desarrollo del sistema es de 8 fases. Este posee sensado de corriente en cada una de sus fases y puede ser configurado para operar con topología boost o buck. Para poder llevar a cabo la implementación del método de reordenamiento, cada una de las ocho corrientes sensadas son acondicionadas y digitalizadas para su posterior procesamiento en una FPGA. Esta última se encarga de medir la amplitud de cada fase y obtener, a partir de la técnica de algoritmos genéticos, el ordenamiento óptimo de las fases. Asimismo, en esta misma plataforma digital, se implementa el control de las señales PWM que actúan en los drivers de las llaves de conmutación del convertidor.

En este proyecto se desarrolla el diseño y construcción del circuito de adquisición y acondicionamiento, y la implementación del método de reordenamiento en FPGA. Previo al desarrollo del mismo, en el siguiente capítulo, se presenta una revisión teórica de las metodologías utilizadas.

1.3. Objetivos

El objetivo de este proyecto final de grado es la implementación de un sistema de ordenamiento para las señales de comando de las llaves de un convertidor interleaved con el objetivo de optimizar las características del ripple de la corriente total.

Esto involucra:

- Diseñar y construir el hardware de adquisición y acondicionamiento de las corrientes de fase medidas.
- Medir la amplitud del ripple de las corrientes de fase utilizando la metodología de procesamiento de señales propuesta en [7].
- Generar un funcional que permita ponderar el ordenamiento de las fases a partir de la técnica de caracterización presentada en [6].
- Implementar la técnica de algoritmos genéticos presentada en [1, 2], utilizando la función de costo planteada en el inciso anterior.

Capítulo 2

Marco teórico

2.1. Introducción

Como se dijo en la presentación de los objetivos, es necesario medir la amplitud de los ripple para caracterizar la corriente total y, de esta manera, encontrar un ordenamiento óptimo, utilizando la técnica de algoritmos genéticos. Lo que sigue es una descripción teórica de las técnicas empleadas para el cumplimiento de los objetivos.

2.2. Medición de amplitud

Por lo dicho anteriormente, el primer paso necesario en el desarrollo del proyecto es medir la amplitud de los ripples de fase. A continuación, se describen los aspectos más importantes del método de medición presentado en Cervellini *et al.* [7].

La medición de la amplitud de los ripples de fase presenta una serie de cuestiones que se ilustran en la figura Fig. 2.1. En la misma se muestra el ripple de corriente de fase (i_{L_k}) donde se aprecia el ruido de alta frecuencia que se genera

por la conmutación de los dispositivos semiconductores y resonancias con los elementos parásitos del convertidor. Además, la figura muestra la versión muestreada ($i^*_{L_k}$), utilizando una frecuencia de muestreo $f_s = 20 f_{sw}$.

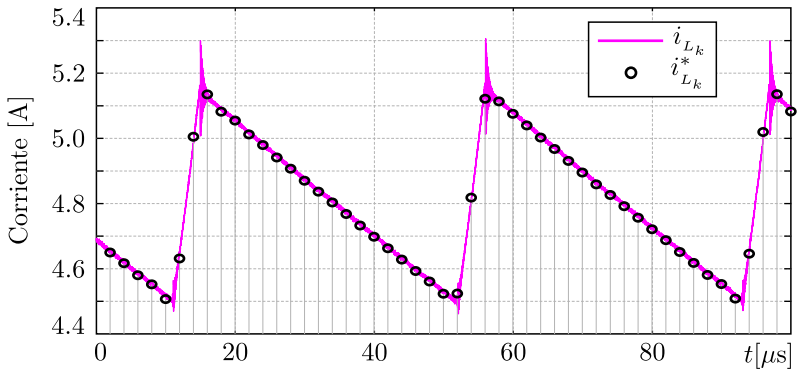


Figura 2.1: Ripple de la corriente de fase (i_{L_k}) y versión muestreada del ripple de corriente con $f_s = 20 f_{sw}$ ($i^*_{L_k}$).

Tal como se observa en la Fig. 2.1, para determinar el valor de la amplitud se deben realizar las mediciones en el instante de conmutación. Sin embargo, la ubicación de esos instantes varía dependiendo de las tensiones de entrada y salida. Una alternativa para determinar la amplitud es sincronizar los instantes de muestreo con la señal de conmutación. No obstante, este enfoque puede verse afectado por el ruido de conmutación, y además requiere del conocimiento del ancho de banda del circuito y de los diferentes retardos del sistema, para alcanzar la sincronización adecuada.

Los problemas de medición descriptos pueden ser evitados haciendo un procesamiento de la señal en el dominio de la frecuencia de modo de aumentar el rechazo al ruido de medición e independizar la determinación de la amplitud de los efectos transitorios generados en los instantes de conmutación.

Teniendo en cuenta las cuestiones planteadas con anterioridad, a continuación

se presenta la metodología desarrollada en [7] para la medición de la relación entre los valores del ripple de corriente entre las fases de los convertidores multifásicos en el dominio de la frecuencia:

Descripción del método de medición de amplitud

Es conocido que el módulo de cada una de las componentes frecuenciales de una señal guarda una relación proporcional con su amplitud y que ésta puede ser representada por medio de una ganancia K (2.1). De esta manera, se recurre al cálculo de la componente fundamental C_1 para identificar la relación de proporcionalidad que existe con la amplitud de la señal A_k, teniendo en cuenta que K depende de la forma de onda de la señal. En este caso la forma de onda de la corriente queda definida por el modo de conducción del convertidor y por los tiempos de encendido y apagado de las llaves, T_{on} y T_{off} respectivamente.

$$\Delta i_{L_k} = K \cdot C_1 \tag{2.1}$$

donde C_1 se define como

$$C_1 = \frac{1}{T} \int_0^T i_{L_k}(t).e^{-jt2\pi/T} \ \mathrm{d}t \tag{2.2}$$

A partir del cálculo de C_1, el factor K para el caso general correspondiente a la operación en DCM resulta:

$$\begin{aligned} K_{DCM} =& \pi[a^2 + b^2 + c^2 - 2ab\cos(xT_{off}) + \\ &+ 2bc\cos(x.(T_{on} + T_{off})) - 2ac\cos(x.T_{on})]^{(-1/2)} \end{aligned} \tag{2.3}$$

donde

$$a = \frac{T}{2\pi}\frac{T_{on}+T_{off}}{T_{on}T_{off}}\ ; \qquad b = \frac{T}{2\pi}\frac{1}{T_{off}}$$
$$c = \frac{T}{2\pi}\frac{1}{T_{on}}\ ; \qquad x = \frac{2\pi}{T}$$

Teniendo en cuenta que en CCM, $T_{on}+T_{off} = T$ y $d = T_{on}/T$, K_{CCM} resulta:

$$K_{CCM} = \frac{\pi^2 d(1-d)}{\sin(\pi d)} \tag{2.4}$$

Luego, el problema de la medición de la amplitud pico a pico del ripple se reduce al cálculo de la componente fundamental del mismo a partir de la versión muestreada de la corriente, denominada $i^*_{L_k}$. Para obtener esta componente, es posible recurrir a la transformada de *Goertzel* de ventana deslizante (SGT), que permite calcular el valor de una única componente armónica en forma eficiente. La función de transferencia de la SGT, de longitud N_s y configurada para recuperar el armónico de orden k, se muestra en (4.1).

$$H_{SGT}(z) = \frac{(1-z^{-N_s})(1-e^{-j2\pi k/N_s}z^{-1})}{1-2cos(2\pi k/N_s)z^{-1}+z^{-2}} \tag{2.5}$$

Como se puede apreciar, las singularidades de H_{SGT} en el plano z están compuestas por N_s ceros equiespaciados sobre el círculo unitario, producto de la ventana deslizante de longitud N_s, un cero adicional en $e^{-j2\pi k/N_s}$ y un par de polos en $e^{\pm j2\pi k/N_s}$. En la Fig. 2.2 se muestra la distribución de ceros y polos en el plano z, y la respuesta en frecuencia para un filtro con $k = 1$ y $N_s = 20$ y utilizando una frecuencia de muestreo f_s.

Sin embargo, la representación de los coeficientes de esta función en sistemas

con aritmética de precisión finita puede ocasionar una incorrecta cancelación entre ceros y polos. Adicionalmente, el error de redondeo puede provocar que las singularidades se desplacen fuera del círculo unitario, afectando así la estabilidad del filtro. De modo de asegurar la estabilidad, es necesario introducir un coeficiente de amortiguamiento que desplaza las singularidades de H_{SGT} hacia el interior del círculo unitario. Si bien el amortiguamiento asegura la estabilidad del filtro, la atenuación a frecuencias diferentes de la frecuencia central se ve reducida y la respuesta de fase distorsionada.

El problema de representación de los coeficientes puede ser evitado multiplicando la señal bajo análisis por otra de igual frecuencia que el armónico de interés. A modo de ejemplo, en (2.6) se presenta el resultado de la multiplicación de dos señales V_1 y V_2.

$$\begin{aligned} V_0(t) &= V_1(t).V_2(t) \\ &= \frac{A_1.A_2}{2}\left[\cos\left((w_1+w_2)t+\phi\right)+\right. \\ &\left.+\cos\left((w_1-w_2)t+\phi\right)\right] \end{aligned} \tag{2.6}$$

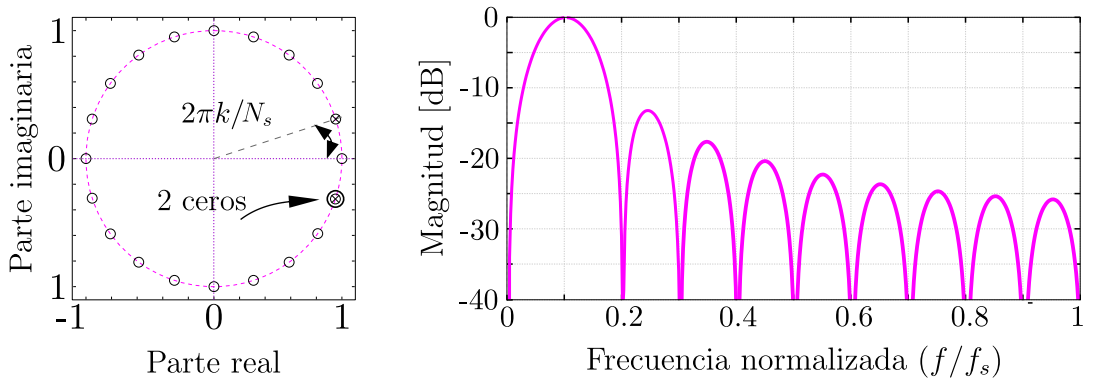


Figura 2.2: Singularidades en el plano z y respuesta en frecuencia de la SGT con $k = 1$ y $N = 20$.

La ecuación (2.6) muestra que $V_0(t)$ tiene una componente de corriente continua (DC) solo cuando w_1 y w_2 son iguales. Por lo tanto, al multiplicar la señal de interés por una señal de frecuencia $f_{sw} = 1/T$, se obtiene un valor de continua que resulta proporcional a la amplitud del ripple de corriente.

La ventaja de la metodología se hace más evidente cuando se configura la SGT para recuperar el nivel de continua. Si $k = 0$, se identifica que no es necesario representar ningún coeficiente distinto de 1 desapareciendo los problemas de representación de coeficientes mencionados, como se muestra en (2.7).

$$H_{SGT}(z) = \frac{(1 - z^{-N_s})}{(1 - z^{-1})} \tag{2.7}$$

Adicionalmente, se debe notar que la frecuencia de conmutación y su fase es conocida, ya que se fija con el control de corriente.

En la Fig. 2.3 se resume el método propuesto para obtener A_k. En el esquema se observa que $i^*_{L_k}$ se multiplica por una señal de frecuencia igual a la frecuencia de conmutación del convertidor ($f_{sw} = 1/T$) y luego utilizando el filtro SGT especializado en $k = 0$, se extrae la componente de continua de la señal resultante. De este modo, se obtiene la amplitud de la componente fundamental de i_{L_k}.

Finalmente, el factor K permite calcular la amplitud pico a pico del ripple.

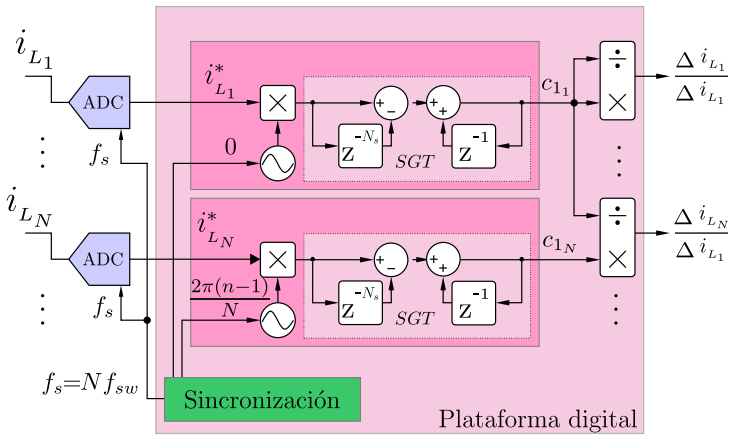


Figura 2.3: Diagrama en bloques del método propuesto de medición de amplitudes.

Es importante destacar que en general, en un convertidor multifásico, la forma de onda de corriente de las distintas fases son iguales entre sí debido a que los tiempos de encendido y apagado de las llaves son los mismos. Luego, el factor K resulta igual para las distintas fases. Por lo tanto, en aquellos casos en donde se requiere calcular la relación entre las amplitudes de las fases, es posible recurrir a la relación entre la componente fundamental del ripple de cada fase, sin necesidad de calcular el factor K.

2.3. Caracterización de la corriente total en estado estacionario

Como se ha mencionado con anterioridad, para aplicar el algoritmo de reordenamiento se necesita una función de costo con la cual evaluar los distintos ordenamientos a fin de encontrar el óptimo. Para ello es necesario obtener la

amplitud pico a pico de la corriente total en función del ciclo de trabajo para cualquier número de fases y bajo cualquier condición de diferencias de amplitud de los ripples de fase. Es por esto que en Antoszczuk *et al.* [8] se presenta un método que permite caracterizar el ripple de la corriente total, a partir de las variables mencionadas, y obtener información del mismo, tal como, la amplitud pico a pico, el valor RMS y el contenido armónico. A continuación se presenta y desarrolla dicho método:

El método parte del análisis del ripple de cada fase en el dominio temporal. Para llevar a cabo este análisis se asume que:

- El convertidor está operando en modo de conducción continua (CCM) y en estado estacionario.
- Las caídas de tensión en los dispositivos semiconductores y en las resistencias parásitas son similares a lo largo de todas las fases, con lo cual el ciclo de trabajo D puede ser considerado igual en todas las fases.
- El ripple de la corriente se puede aproximar por segmentos de recta, debido a que la constante de tiempo, asociada a los inductores y a su componente resistiva, es normalmente mucho mayor que el período de conmutación.
- El error de fase, con respecto al desfase ideal ($\frac{2\pi}{N}$), puede reducirse mediante el control de corriente, por lo cual se asume despreciable.

Bajo estas consideraciones, las posibles diferencias entre las amplitudes del ripple de cada fase dependen principalmente de la tolerancia de los inductores. Además, la aproximación por segmentos de recta permite definir el ripple de cada fase a partir de sus valores pico, y del instante en el que estos picos aparecen.

En la Fig. 2.4 se muestra un ejemplo de los ripples de fase normalizados respecto a un valor nominal, $r_x(t)$, y el ripple de la corriente total, $r_T(t)$, que

surge de la suma de los ripples de fase normalizados para un convertidor buck multifásico con tolerancia en los inductores. El propósito de esta figura es ilustrar un punto de partida, para desarrollar la caracterización del ripple basada en un caso general.

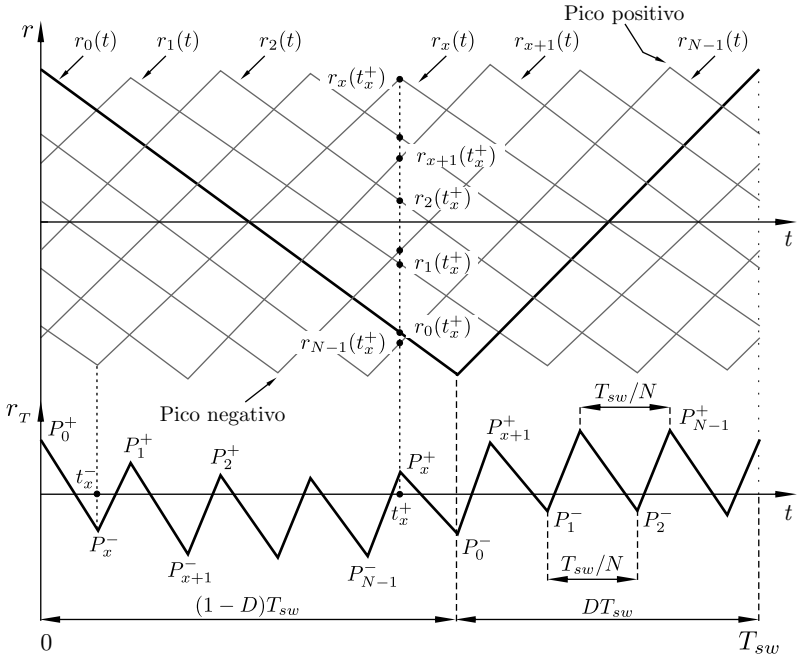


Figura 2.4: Ripple de fase normalizado (sup) y ripple total (inf) para un convertidor buck.

Como se puede observar, los picos en $r_{_T}$ ocurren en el mismo instante que los picos en r_x. Estos picos se definen como positivos cuando la pendiente de la corriente cambia de positivo a negativo, o negativos en el caso contrario donde la pendiente del ripple cambia de negativo a positivo. Los picos positivos se indican con un superíndice $+$, mientras que los negativos con un superíndice $-$. De esta forma, un pico positivo en la fase x produce un pico positivo P_x^+ en $r_{_T}$, mientras que un pico negativo en la misma fase produce el pico negativo P_x^- en el ripple total. Dado que $r_{_T}$ se determina a partir de la suma de los ripples de fase, el pico

positivo P_x^+ que ocurre en el instante t_x^+, se puede expresar como:

$$P_x^+ = r_0(t_x^+) + r_1(t_x^+) + \cdots + r_x(t_x^+) \cdots r_{N-1}(t_x^+). \tag{2.8}$$

Análogamente, el pico negativo en el ripple total P_x^-, que ocurre en el instante t_x^-, está dado por:

$$P_x^- = r_0(t_x^-) + r_1(t_x^-) + \cdots + r_x(t_x^-) \cdots r_{N-1}(t_x^-). \tag{2.9}$$

Dado que los instantes $t_x^\pm$ se pueden calcular de forma analítica, los instantes en los que ocurren los picos en $r_{_T}$ se pueden definir de forma precisa. Como las fases se encuentran desfasadas $\frac{2\pi}{N}$ y tienen igual ciclo de trabajo, el intervalo entre picos del mismo signo en los ripples de fase, y en consecuencia en el ripple total, es constante e igual a $\frac{T_{sw}}{N}$. Si se toma como referencia el instante en el que ocurre el pico positivo en la fase cero, que genera el pico P_0^+ en $r_{_T}$, es posible determinar el instante en el que ocurre un pico positivo en la fase x como:

$$t_x^+ = \frac{xT_{sw}}{N} \tag{2.10}$$

Asimismo, considerando que la ocurrencia de un pico positivo y de uno negativo están relacionadas por el ciclo de trabajo, el instante en el que ocurre un pico negativo en la fase x está dado por:

$$t_x^- = \begin{cases} \frac{xT_{sw}}{N} - DT_{sw} & \text{si } \frac{x}{N} < D \\ \frac{xT_{sw}}{N} - DT_{sw} + T_{sw} & \text{si } \frac{x}{N} > D \end{cases} \tag{2.11}$$

El cálculo de las expresiones (2.8) y (2.9) requiere determinar el valor de los ripples de fase en los instantes t_x^+ y t_x^-. Estos valores se pueden determinar a

partir de una representación genérica del ripple de la corriente de fase, basada en la periodicidad y simetría del sistema. En esta representación, el ripple de corriente de la fase x se define como una función $f(t)$, de forma triangular, con período T_{sw} y pesada por una amplitud normalizada A_x. Esta amplitud está relacionada con el valor pico asociado a la fase x, $\hat{I}_x$, y el valor pico nominal, $\hat{I}_n$:

$$r_x(t) = A_x f(t) = \frac{\hat{I}_x}{\hat{I}_n} f(t). \tag{2.12}$$

En la Tabla 2.1 se muestran las expresiones para determinar $\hat{I}_x$ e $\hat{I}_n$ en función del ciclo de trabajo, para las topologías buck y boost. Como se puede apreciar, $\hat{I}_x$ depende de la inductancia asociada a la fase x, denominada L_x. Por otro lado, $\hat{I}_n$ es función de la inductancia nominal L_n, la cual se define en base a las especificaciones de diseño.

Tabla 2.1: Amplitud pico del ripple de fase en función del ciclo de trabajo.

	$\hat{I}_x$	$\hat{I}_n$
Boost	$\frac{V_i D T_{sw}}{2L_x}$	$\frac{V_i D T_{sw}}{2L_n}$
Buck	$\frac{V_i (1-D) D T_{sw}}{2L_x}$	$\frac{V_i (1-D) D T_{sw}}{2L_n}$

Para calcular P_x^+, es conveniente expresar la función $f(t)$ con su máximo positivo en $t = 0$. Esta función, denominada $f^+(t)$, esta representada por dos secciones con diferente pendiente, separadas por el instante $t = (1 - D)T_{sw}$:

$$f^+(t) = \begin{cases} 1 - \frac{2}{(1-D)T_{sw}} t & 0 \leq t \leq (1-D)T_{sw} \\ -1 + \frac{2}{DT_{sw}} t - \frac{2(1-D)}{D} & (1-D)T_{sw} < t \leq T_{sw} \end{cases} \tag{2.13}$$

Tal como se indica en la expresión (2.8), el cálculo de P_x^+ requiere solo de los valores de las fases en los instantes de picos positivos. Por lo tanto, se define una nueva función f_k^+ a partir del muestreo de $f^+(t)$ en estos instantes, como se muestra en la Fig. 2.5. De esta forma:

$$\begin{aligned} f_0^+ &= f^+(0) \\ f_1^+ &= f^+\left(\frac{T_{sw}}{N}\right) \\ f_2^+ &= f^+\left(\frac{2T_{sw}}{N}\right) \\ &\dots \\ f_{(N-1)}^+ &= f^+\left(\frac{(N-1)T_{sw}}{N}\right). \end{aligned} \tag{2.14}$$

Estas expresiones se pueden expresar en forma general como:

$$f_k^+ = \begin{cases} 1 - \frac{2k}{(1-D)N} & \text{si } 0 \leq k \leq N(1-D) \\ -1 + \frac{2k}{DN} - \frac{2(1-D)}{D} & \text{si } N(1-D) < k \leq N-1 \end{cases} \tag{2.15}$$

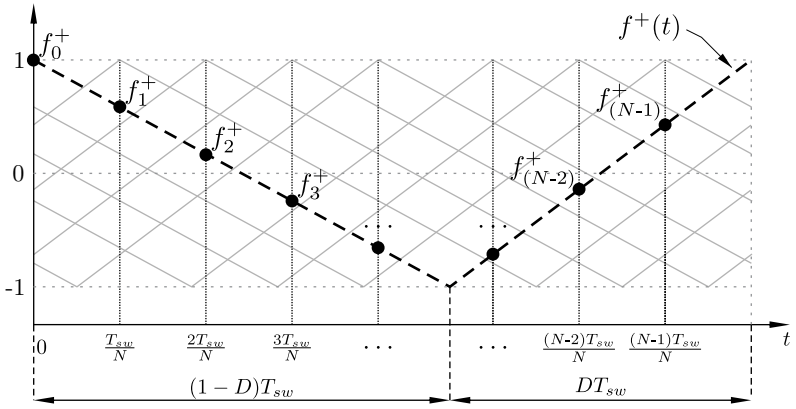


Figura 2.5: $f^+(t)$ (linea de puntos) y f_k^+ (puntos).

El uso de esta representación para el cálculo de P_x^+ se ilustra en la Fig. 2.6, donde se muestran los ripples de fase y el ripple total para un ciclo de conmutación.

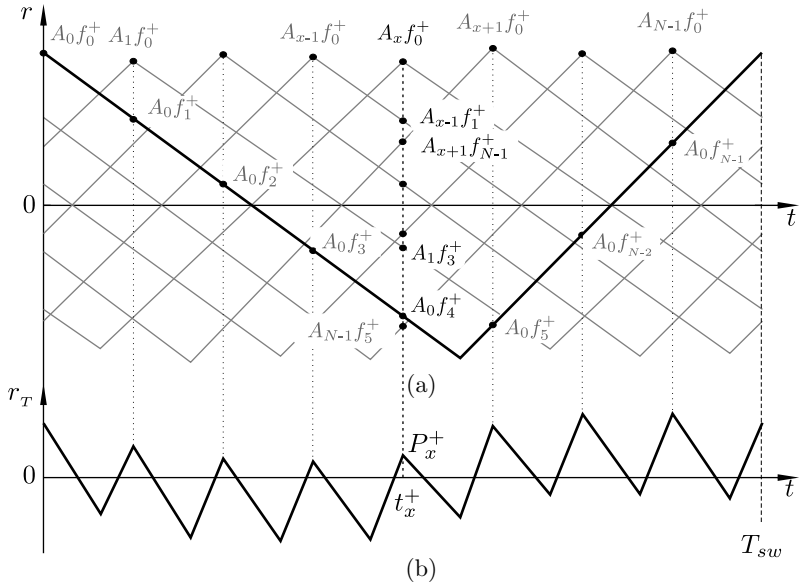


Figura 2.6: Determinación de P_x^+. (a): Ripple de fases. (b): Ripple total.

Como se puede notar, cada fase tiene su propio índice k, con el origen en su respectivo pico positivo. De esta forma, dado que el instante en el que se produce P_x^+ coincide con el del pico positivo de la fase x, la contribución de esta fase a P_x^+ se determina ponderando f_0^+ por la amplitud normalizada A_x. Por otro lado, la fase $x-1$ está desplazada en $-\frac{T_{sw}}{N}$ respecto de t_x^+, por lo que su contribución a P_x^+ se determina ponderando la función f_1^+ por su amplitud normalizada, A_{x-1}. Operando de forma análoga con el resto de las fases, se puede inferir que P_x^+ esta

dado por:

$$\begin{aligned} P_x^+ &= A_x f_0^+ + A_{x-1} f_1^+ \ldots + A_1 f_3^+ + A_0 f_4^+ + A_{N-1} f_5^+ + \ldots + A_{x+1} f_{N-1}^+ \\ &= \sum_{k=0}^{N-1} A_{x-k} \cdot f_k^+. \end{aligned} \tag{2.16}$$

Se debe notar que en (2.16), $x-k$ se puede tornar negativo. En este caso, dada la periodicidad del problema, se debe considerar $x-k+N$.

Operando con (2.15) y (2.16), se obtiene la expresión de P_x^+:

$$\begin{aligned} {P_x^+}_{(D)} = \sum_{k=0}^{N-1} A_{x-k} \cdot f_k^+ = \sum_{k=0}^{N-1} A_{(x-k)} \left[1 - \frac{2k}{(1-D)N}\right] + \\ + \sum_{k>N(1-D)}^{N-1} A_{(x-k)} \left[\frac{2k}{(1-D)DN} - \frac{2}{D}\right]. \end{aligned} \tag{2.17}$$

En el caso de picos negativos, P_x^-, es conveniente expresar la función $f(t)$ con su máximo negativo en $t=0$. Esta función, denominada $f^-(t)$, esta representada por dos secciones de diferente pendiente, separadas por el instante $t = DT_{sw}$. Operando de forma análoga que para el cálculo de P_x^+, la expresión para P_x^- resulta:

$$\begin{aligned} {P_x^-}_{(D)} = \sum_{k=0}^{N-1} A_{x-k} \cdot f_k^- = -\sum_{k=0}^{N-1} A_{(x-k)} \left[1 - \frac{2k}{DN}\right] - \\ - \sum_{k>N\cdot D}^{N-1} A_{(x-k)} \left[\frac{2k}{(1-D)DN} - \frac{2}{(1-D)}\right]. \end{aligned} \tag{2.18}$$

Analizando (2.17) y (2.18), se puede observar que los sumandos en la primer sumatoria son válidos para cualquier k; mientras que en la segunda sumatoria, debido a que la función f está definida por tramos y a la relación entre k y D, solo son válidos los sumandos que verifican la expresión en los límites de la sumatoria.

Es importante destacar que los valores de D que definen las zonas de validez de cada sección coinciden con los casos de cancelación de ripple en el caso ideal.

A partir de las N amplitudes de los ripples de un sistema de N fases, se obtienen los N picos positivos y los N picos negativos del ripple total, lo cual conforma un vector de $2N$ valores. Con el objetivo de ilustrar en forma práctica el método propuesto, a continuación se presenta un pseudocódigo, que permite obtener el valor de $P_x^{\pm}$ para un determinado D:

Algoritmo 2.1 Pseudocódigo para la determinación de $P_x^{\pm}$ para un dado D.

for $x = 0$ **to** $(N-1)$ **do** ▷ Picos asociados a fase x (desde 0 a $N-1$)
  $P_x^{+} = 0$ ▷ Inicialización de variables
  $P_x^{-} = 0$
  for $k = 0$ **to** $(N-1)$ **do** ▷ Suma de los ripples en el pico de la fase x
    if $x - k < 0$ **then**
      $k = x - k + N$
    end if
    $S1p = A_{(x-k)} \left[1 - \frac{2k}{(1-D)N}\right]$ ▷ 1° sumando de (2.17)
    if $D > 1 - k/N$ **then**
      $S2p = A_{(x-k)} \left[\frac{2k}{(1-D)DN} - \frac{2}{D}\right]$ ▷ 2° sumando de (2.17)
    else
      $S2p = 0$
    end if
    $S1n = -A_{(x-k)} \left[1 - \frac{2k}{DN}\right]$ ▷ 1° sumando de (2.18)
    if $D < k/N$ **then**
      $S2n = -A_{(x-k)} \left[\frac{2k}{(1-D)DN} - \frac{2}{(1-D)}\right]$ ▷ 2° sumando de (2.18)
    else
      $S2n = 0$
    end if
    $P_x^{+} = P_x^{+} + S1p + S2p$
    $P_x^{-} = P_x^{-} + S1n + S2n$
  end for
end for

El conocimiento del valor numérico de $P_x^{\pm}$, determinado a través del algoritmo 2.1, y la secuencia de aparición de estos picos, dada por (2.10) y (2.11), determinan el ripple de la corriente total para cualquier condición de diferencias entre los inductores de fase. Luego, una vez definido el ripple, es posible evaluar aspectos de interés en Δi_T, como amplitud pico a pico, valor rms, contenido

armónico y su impacto en el diseño del convertidor.

2.4. Método de ordenamiento de los ripples de fase

Las diferencias de amplitud entre los ripples de fase es el principal factor en el deterioro de las características del ripple total Δi_T. Adicionalmente, el orden en el que se ubican los ripples de fase influye en Δi_T y, por lo tanto, existe un ordenamiento óptimo que minimiza este ripple.

La principal problemática en la búsqueda del ordenamiento óptimo radica en la cantidad de casos posibles, que es igual al factorial del número de fases. Para sortear este problema, en Antoszczuk *et al.* [1, 2] se presenta un método de ordenamiento de fases para convertidores interleaved operando en modo de conducción continua (CCM), basado en algoritmos genéticos. Esta técnica se basa en la mecánica de selección natural y supervivencia del más apto, lo cual permite encontrar una solución cercana a la óptima sin necesidad de evaluar todos los casos posibles [9, 10, 11].

En los algoritmos genéticos, una población de potenciales soluciones, denominadas cromosomas o individuos, evoluciona a lo largo de sucesivas iteraciones, denominadas *generaciones*. Cada individuo tiene asociado un nivel de ajuste respecto de la solución buscada. El nivel de ajuste se determina por medio de una función de costo, denominada función objetivo. Asimismo, la evolución a lo largo sucesivas generaciones se lleva a cabo utilizando operadores genéticos tales como cruces y mutaciones. El cruce genera una descendencia a partir del intercambio de información entre dos individuos de la generación actual. Por su parte, la mutación es un operador, con baja probabilidad, que altera alguna propiedad de un

individuo en forma aleatoria antes de introducirlo en la nueva generación. El cruce y la mutación cumplen distintos roles dentro del algoritmo. La descendencia obtenida por cruce tiende a mejorar la calidad promedio de la población, mediante la exploración de los espacios ya presentes. La mutación, sin embargo, ayuda a explorar otros espacios, evitando así óptimos locales. Adicionalmente, es deseable preservar los individuos más aptos a lo largo de generaciones sucesivas. En este sentido, se selecciona una cierta cantidad de individuos que mejor ajusten a la función de costo, denominados *elite*, que pasan a la siguiente generación sin sufrir cambios.

En la Fig. 2.7 se muestra el ciclo básico del algoritmo genético. El mismo parte de una población inicial generada aleatoriamente. Luego, se evalúan los individuos que integran la población y se les asigna un nivel de ajuste mediante la función de costo. Seguidamente, se seleccionan los individuos a partir de los cuales se crea la próxima generación, y se aplican los operadores genéticos antes descriptos. Una nueva evaluación devuelve la población a su tamaño original, mediante la selección de los individuos más aptos. El ciclo continúa hasta cumplirse un criterio de finalización. Este criterio puede ser la cantidad máxima de generaciones, una cota de error, o la falta de cambios de una generación respecto de la anterior.

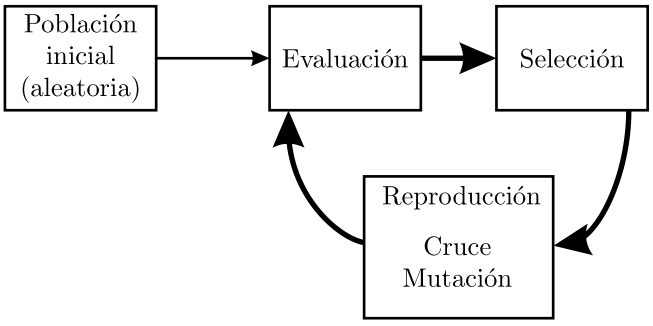


Figura 2.7: Diagrama de flujo del algoritmo genético.

A continuación, se detalla el modelado del problema, la función objetivo y los operadores genéticos antes mencionados en la aplicación de Antoszczuk *et al.* [1, 2].

Modelado y codificación del problema

Las potenciales soluciones del problema se modelan mediante un conjunto de parámetros, denominados *genes*, de forma tal de representar el cromosoma o individuo. Como el problema de optimización de Δi_T es un problema de ordenamiento, cada individuo esta conformado por un vector ordenado compuesto por la amplitud del ripple de cada fase. Por ejemplo, el individuo

$$I_1 = (A1\ A2\ A3\ A4\ A5\ A6\ A7\ A8) \tag{2.19}$$

corresponde a un convertidor de 8 fases, y representa el caso en el que el ripple asociado al inductor de fase L_1 precede al ripple asociado a L_2, y así sucesivamente. Por otro lado, en el individuo

$$I_2 = (A2\ A1\ A3\ A4\ A5\ A6\ A7\ A8) \tag{2.20}$$

se invirtió el orden de los ripples asociados a $L1$ y $L2$ de forma tal que, en este nuevo individuo, $A2$ precede a $A1$.

Es importante destacar que, debido a la naturaleza del problema, la composición del individuo esta sujeta a restricciones que garanticen su validez. En primer lugar, la amplitud del ripple de todas las fases debe estar presentes en cada individuo. En (2.21) se muestra un individuo que no contiene la amplitud de fase $A3$

y, por consiguiente, no representa un ordenamiento de fases válido.

$$I_{erroneo1} = (A1\ A2\ A4\ A5\ A6\ A7\ A8) \tag{2.21}$$

En segundo lugar, no pueden existir amplitudes repetidas dentro de un mismo individuo. En (2.22), se muestra un ejemplo en el que la fase $A1$ se ubica simultáneamente en la primera y tercera posición, determinando así un estado no válido de ordenamiento.

$$I_{erroneo2} = (A1\ A2\ A1\ A3\ A4\ A5\ A6\ A7\ A8) \tag{2.22}$$

De los ejemplos antes expuestos se determina que la cantidad de *genes* en un individuo es igual a la cantidad de fases, y que no existen repeticiones dentro de un mismo individuo. Estas restricciones pueden ser consideradas de dos maneras:

- Mediante penalización en el valor de ajuste de los individuos no válidos.
- Mediante operadores genéticos apropiados, que generen en todo momento individuos válidos.

Se adopta la segunda estrategia, debido a que no se desperdicia esfuerzo de cómputo en la generación y evaluación de individuos que no representan un estado válido del sistema.

Función objetivo

La función objetivo, denominada $f_{fit}(I_j)$, debe entregar un valor que indique el ajuste del individuo I_j, respecto al criterio de optimización seleccionado. En este caso, el objetivo es minimizar la amplitud pico a pico de la corriente total, es decir, Δi_T. Esta función se puede construir a partir de la caracterización propuesta

en la Sección 2.3, como la diferencia entre el valor máximo y mínimo de P_x^+ y P_x^- respectivamente:

$$f_{fit}(I_j) = max(P_x^+) - min(P_x^-) \tag{2.23}$$

Función de cruce

La función de cruce intercambia información entre dos individuos de la generación actual, denominados padres, para generar un nuevo individuo, denominado descendencia. La función de cruce ordenado (*order crossover*) es muy atractiva para esta aplicación, debido a que permite intercambiar la información de orden y posición entre dos individuos [12, 13]. Esta función toma una subsecuencia de fases de uno de los padres y preserva el orden relativo del segundo. Por ejemplo, a partir de los padres P_1 y P_2, se seleccionan dos puntos de corte al azar, señalizados por " | ", tal como se indica en (2.24).

$$\begin{aligned} P_1 &= (A1\ A2\ |A3\ A4\ A5|\ A6\ A7\ A8) \\ P_2 &= (A3\ A4\ |A2\ A5\ A1|\ A6\ A8\ A7) \end{aligned} \tag{2.24}$$

La descendencia, O_1 se genera de la siguiente forma. En primer lugar, los subconjuntos contenidos entre los puntos de corte de P_1 se copian a la descendencia, respetando el orden y la posición, como se muestra en (2.25).

$$O_1 = (-\ -\ |A3\ A4\ A5|\ -\ -\ -) \tag{2.25}$$

Luego, se seleccionan las fases del otro padre a partir del segundo punto de corte, respetando el orden, y se descartan las fases ya presentes en O_1 (2.26).

$$A6\ A8\ A7\ \cancel{A3}\ \cancel{A4}\ A2\ \cancel{A5}\ A1 \tag{2.26}$$

Seguidamente, se completa la descendencia desde el segundo punto de corte, recomenzando desde el principio si se alcanza el final del individuo (2.27)

$$O_1 = (A2\ A1\ |A3\ A4\ A5|\ A6\ A8\ A7) \tag{2.27}$$

Adicionalmente, se puede generar otra descendencia invirtiendo el orden de los padres.

Función de mutación

La función de mutación altera los genes de un individuo, seleccionado de forma aleatoria, antes de colocarlo en una nueva generación. La mutación se realiza invirtiendo el orden de dos fases, de forma tal de generar un nuevo ordenamiento que explore otros campos y evite la convergencia a mínimos locales. Por ejemplo, a partir del individuo P_1, se genera $O1$ invirtiendo las fases $A3$ y $A7$, tal como se ilustra en (2.28) y (2.29).

$$P_1 = (A1\ A2\ \underline{A3}\ A4\ A5\ A6\ \underline{\text{A7}}\ A8) \tag{2.28}$$

$$O_1 = (A1\ A2\ \underline{A7}\ A4\ A5\ A6\ \underline{\text{A3}}\ A8) \tag{2.29}$$

Criterio de finalización

Dado que el algoritmo genético realiza la búsqueda del ordenamiento más adecuado mediante la iteración de sucesivas generaciones, es necesario contar con un criterio que indique la convergencia a una solución. Existen diversos criterios en la literatura para determinar el final del proceso iterativo, tales como una cota máxima de error, un número máximo de generaciones o la mejora relativa de una generación respecto de las anteriores.

En el caso del ordenamiento de los ripples de fase no es posible, en general, establecer una cota de error máximo. Esto se debe a que no se conoce con antelación el máximo o el mínimo Δi_T esperable para una dada condición de desbalances. Por su parte, definir el fin del proceso iterativo a partir de un número máximo de iteraciones no es práctico, ya que es posible que se realicen más iteraciones que las necesarias, o que se detenga el proceso antes de obtener la solución óptima. Por otro lado, si se contrastan el mejor individuo de una generación con los de las generaciones previas, es posible determinar si existen mejoras en la calidad de la solución a lo largo de sucesivas iteraciones. De esta forma, se puede establecer una solución aceptable cuando no existen cambios a lo largo de una determinada cantidad de generaciones.

Es importante destacar que, mediante esta metodología, la solución obtenida no es necesariamente la mejor respecto a la totalidad de combinaciones posibles. Sin embargo, dado que el operador de mutación evita la convergencia a óptimos locales, se arriba a una solución cercana al óptimo global.

2.5. Resumen

Hasta aquí se ha hecho una revisión del marco teórico de los métodos utilizados para el desarrollo de este proyecto. En el próximo capítulo se describe el diseño y desarollo de la placa de adquisición, la cual debe tener la cantidad de canales de entrada requeridas para medir las variables de un prototipo de convertidor multifásico de 8 fases. Además, el diseño debe cubrir los requerimientos de precisión y estabilidad necesarios por el algoritmo de ordenamiento.

Capítulo 3

Diseño y desarrollo de la placa de adquisición

3.1. Introducción

Con el objetivo de implementar el algoritmo de ordenamiento de los ripples de fase, es necesario diseñar el circuito de adquisición para acondicionar y digitalizar cada una de las 8 corrientes sensadas del convertidor. De esta forma, dichas corrientes en conjunto con sus correspondientes señales de control pueden ser procesadas en la FPGA, plataforma en la que se implementan los algoritmos descriptos en el Cap. 2. En particular, el primero de ellos, detallado en la Sec. 2.2, dedicado a la medición de amplitud de las fases, requiere una digitalización apopiada de las señales para obtener un resultado adecuado y, por lo tanto, debe ser tenido en cuenta para el diseño de esta etapa.

Como se mencionó anteriormente, el diseño del hardware debe ser compatible con la etapa de potencia y sensores existentes. En esta etapa, tanto las señales que provienen del convertidor como las entradas y salidas de la FPGA son interconectadas a través de un backplane. Este provee asimismo las fuentes de alimentación

necesarias: +15V, +5V, -5V y 3.3V. Además, cuenta con cuatro conectores de 32x3 pines. Por cuestiones de diseño, cada uno de ellos permite sensar hasta dos fases del convertidor con sus entradas diferenciales en determinados pines previamente establecidos. Por consiguiente, dado que la cantidad de señales a medir es de 8, es necesario construir 4 placas adquisidoras de 2 canales cada una. En la Fig. 3.1 se puede observar el diagrama en bloques correspondiente a una de las placas.

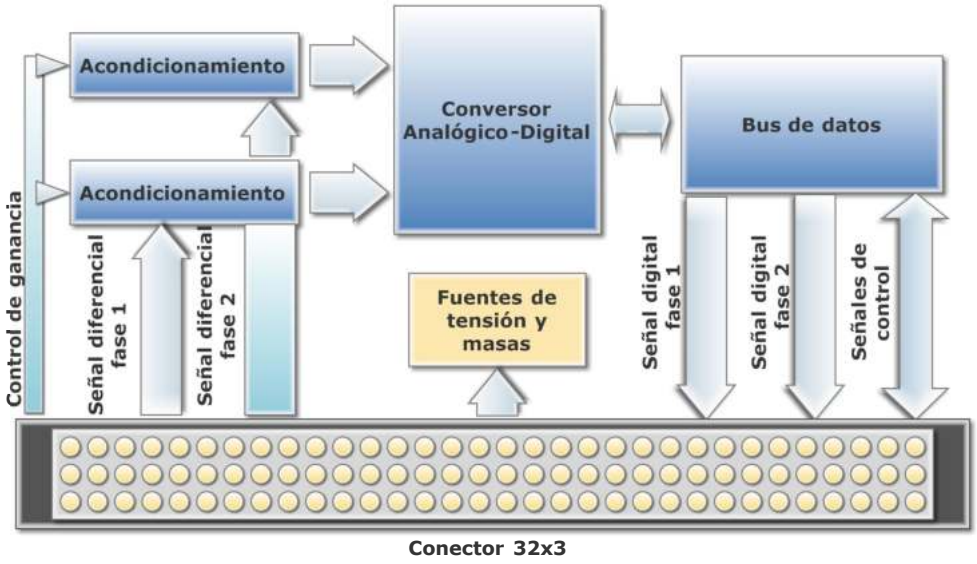


Figura 3.1: Diagrama en bloques de una placa del circuito adquisidor

En las siguientes secciones se describen con mayor detalle cada uno de los bloques.

3.2. Conversor analógico digital

El conversor analógico digital es el componente central del circuito de adquisición y para su elección es necesario definir dos parámetros: la frecuencia de

muestreo y su resolución.

En primer lugar, para establecer la resolución del ADC se asume un rango típico de tolerancia de los inductores de entre el 1 % y el 5 %. Para que en la implementación digital se caracterize el ripple total correctamente a partir de la medición de cada una de las amplitudes de fase, se adopta una resolución al menos 10 veces mejor, es decir, del 0.1-0.5 %, a la salida del bloque de medición de amplitud. En condiciones ideales, 10 bits serían suficientes para alcanzar la resolución deseada. Sin embargo, teniendo en cuenta que los ADC tienen además otras fuentes de error (INL, DNL, ganancia), y que puede haber errores acumulativos en las operaciones, se adopta un valor de 12 bits.

Por otro lado, para determinar la frecuencia de muestreo se debe tener en cuenta que las señales de entrada al adquisidor, es decir, los ripples de corriente de cada una de las fases del convertidor, son señales periódicas que conmutan típicamente en las decenas de kilohertz en las aplicaciones mencionadas en la introducción. Además, como sólo se trabaja con la componente fundamental de la señal para la medición de su amplitud, tal como se explicó en la Sec. 2.2, por el teorema de muestreo de Nyquist sólo se requieren dos muestras por ciclo de señal para obtener la amplitud. Sin embargo, de forma tal de permitir un pre procesamiento de la señal y habilitar el uso de las corrientes adquiridas para otros propósitos dentro de la FPGA (como control o monitoreo), es deseable aumentar la frecuencia de muestro respecto al mínimo indicado por Nyquist.

Con todas estas consideraciones mencionadas, dentro de las ofertas comerciales se optó por el ADC LTC2323. Este es un ADC de 16 bits de aproximaciones sucesivas (SAR) que puede muestrear hasta 5Msps, lo cual satisface Nyquist en el rango de conmutación de los ripples de corriente para este tipo de aplicación. Su SINAD (Signal-to-noise and distortion, acrónimo del inglés) en todo su rango de frecuencia de operación es de 81.7dB. Esto es útil para determinar el número

efectivo de bits (ENOB) mediante (3.1). Este parámetro especifica la resolución del ADC teniendo en cuenta el ruido y la distorsión que introduce este mismo. Aplicando dicha fórmula se obtiene una resolución efectiva de 13 bits. Aunque este valor satisface el requisito propuesto de 12 bits esto es válido en tanto los errores de ganancia y offset del ADC sean calibrados. Esto último se detalla en el Cap. 5.

$$ENOB = \frac{SINAD - 1{,}76dB}{6{,}02} \tag{3.1}$$

Por último, el ADC escogido cuenta con dos canales, la cantidad necesaria por cada placa de adquisición, lo cual ofrece simplicidad y reducción de costo y espacio en su diseño.

En la Fig. 3.2 se puede observar la configuración del ADC, de acuerdo a su hoja de datos, para un correcto funcionamiento.

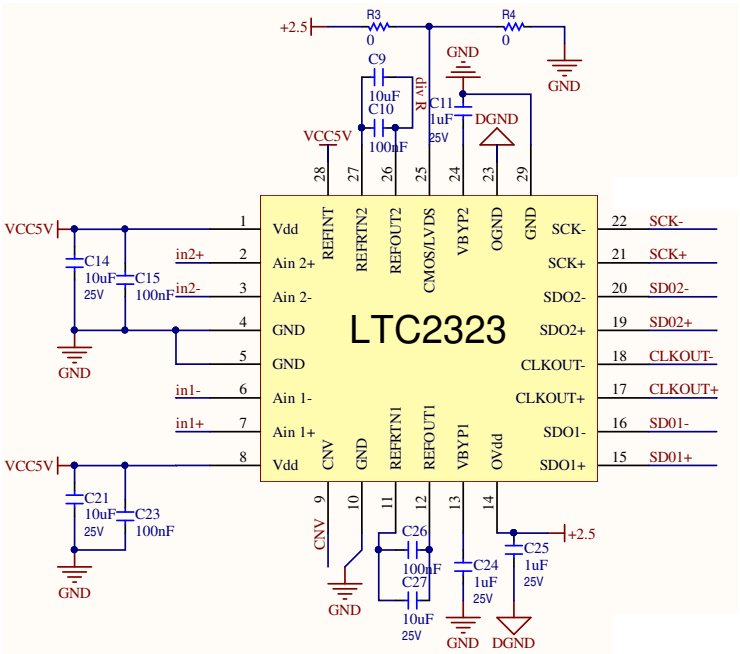


Figura 3.2: Configuración del ADC LTC2323

Este componente permite la transferencia de datos en modo LVDS (modo diferencial) o CMOS (modo común). La selección de un modo u otro se hace mediante una resistencia de valor nulo en el pin 25 (+2.5V para LVDS y 0V para CMOS). Debido a que en la interfaz digital no se espera un entorno ruidoso (conexiones directas vía PCB, aisladas del ciruito de potencia) y a la gran cantidad de canales de adquisicion (que implica una gran cantidad de pines requeridos en la FPGA), se selecciona el modo CMOS. Otro detalle a destacar de la figura anterior es que se necesita una tensión de +2.5V para la alimentación digital del ADC. Este valor no se encuentra dentro de los disponibles en el backplane, mencionados en la introducción de este capítulo. Por eso se utilizó un regulador de tensión LF25CDT para lograr esta tensión a partir de la fuente de +5V.

Por último, los pines 6-7 y 2-3 corresponden a las entradas diferenciales de los canales 1 y 2 respectivamente. Estos deben ser ajustados a los niveles de tensión admisibles por los ADC a partir un bloque de acondicionamiento.

3.3. Acondicionamiento

El bloque de acondicionamiento tiene como propósito adaptar las señales que provienen de los sensores de corriente de cada fase del convertidor a niveles de tensión adecuados de entrada del ADC, de modo de asegurar el mayor aprovechamiento del rango dinámico del conversor. Un diagrama en bloque del mismo se puede ver en la Fig. 3.3, donde IS_P e IS_N representan el borne de entrada de corriente positivo y negativo respectivamente (salida de un transductor de corriente con salida en corriente). A continuación se explican cada uno de estos subbloques.

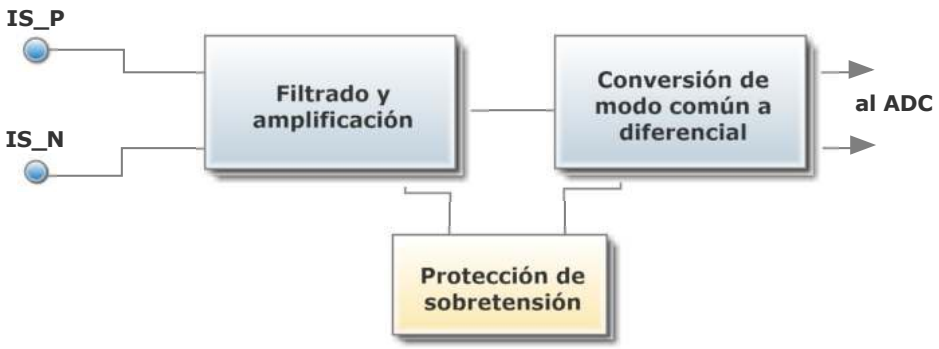


Figura 3.3: Detalle del bloque de acondicionamiento

Filtrado y amplificación

Las señales de las fases que provienen del convertidor interleaved antes de llegar al circuito de adquisición pasan por un transductor de corriente LA25-NP de 1mA/A. Dado que la máxima corriente que puede alcanzar una fase es de 30A, la máxima corriente por canal a la entrada del circuito de adquisición es de 30mA. Por otro lado, la alimentación del ADC es de 5V, con lo cual su tensión de referencia, de acuerdo a su hoja de datos, es de 4.096V. Para que se pueda trabajar con estos niveles de tensión se requiere un amplificador de instrumentación para amplificar la señal deseada sin amplificar el ruido que llega a ambas terminales de entrada (modo común). Además, el amplificador de instrumentación permite realizar la medición sin cargar al sensor de corriente, debido a su alta impedancia de entrada. Por lo tanto, asumiendo una ganancia unitaria, la resistencia a la entrada del amplificador debe ser igual a 4.096V/30mA=136Ω. Para su implementación se seleccionan dos resistencias en paralelo de 270Ω. El amplificador elegido es el AD8250 de ganacia programable G=1, 2, 5 y 10. Su bajo ruido y distorsión en la salida, y su buen rechazo a tensiones de entrada en modo común lo hace ideal para interactuar con sensores y trabajar con conversores analógico-digital de alta frecuencia de muestreo, como el utilizado en este proyecto.

Por otro lado, debe tenerse en cuenta que las señales de entrada al circuito de adquisición no están limitadas en banda y, por lo tanto, deben ser filtradas previamente para evitar aliasing. Para el cálculo del filtro se debe recordar que la frecuencia de muestreo se encuentra en el orden de los MHz. Por ende, un capacitor a la entrada de 2.2nF en conjunto con las dos resistencias en paralelo 270Ω es suficiente para mitigar dicho efecto ya que el polo de este filtro se encuentra en 535.9KHz. El ajuste de esta frecuencia de corte es aceptable a fines prácticos porque es menor a la mitad de la frecuencia de muestreo que se puede establecer y es lo sufientemente grande para no afectar el ancho de banda de la componente fundamental de la señal de entrada.

Por último, a la salida del amplificador se conecta un filtro pasabajos para reducir el nivel de ruido, tal como lo recomienda la hoja de datos del ADC. El filtro pasabajos es un circuito RC que se ajusta para tener un polo en aproximadamente 1MHz.

En la siguiente figura se muestra el circuito completo de la etapa de filtrado y amplificación, cuya ganancia se ajusta por medio de las entradas digitales A0 y A1.

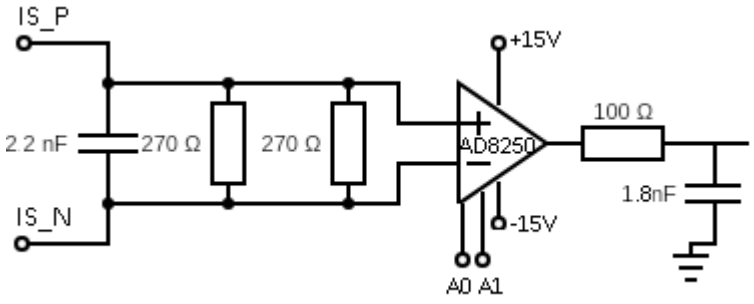
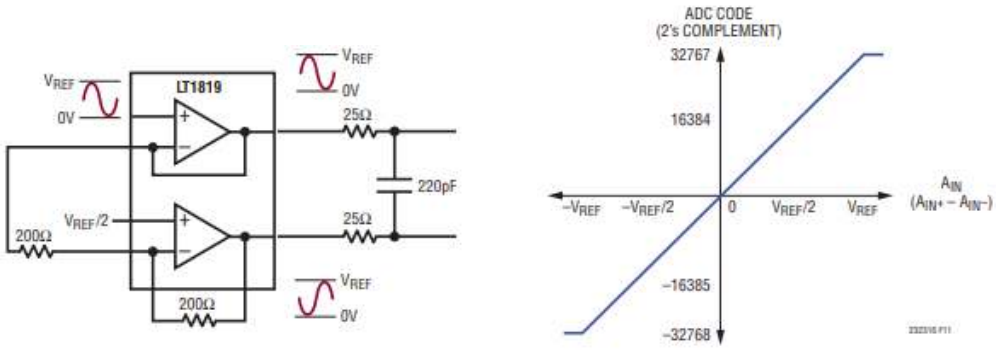


Figura 3.4: Filtrado y amplificación

Conversión de modo común a diferencial

Para aprovechar el rango de conversión completo, el ADC requiere de señales diferenciales. Se pueden generan también señales pseudo diferenciales, es decir, se fija la tensión de una de la entradas y se varía la tensión de la otra pero esto trae aparejado la utilización de sólo la mitad de la escala completa del ADC. Por tal motivo, se prefiere transformarlas en señales completamente diferenciales a partir del circuito de la Fig. 3.5, que ofrece la hoja de datos del ADC, con su correspondiente función de transferencia. Como se observa en la Fig. 3.5b se utiliza la escala completa de -32768 a 32767 en complemento a 2.



(a) Circuito diferencial (b) Función transferencia

Figura 3.5: Conversión de modo común a diferencial

Los AOs que se recomiendan en la hoja de datos son los LT1819 por su elevado slew rate, bajo ruido y baja distorsión. Se requieren dos de ellos, uno como seguidor de tensión y otro como inversor de ganancia unitaria, para de esta forma conseguir las señales diferenciales. Dado que el ADC está alimentado a 5V, las señales entrantes al mismo se esperan que sean entre 0 y un valor de referencia de 4.096V. Para que pueda llegar a este valor, se debe alimentar con una tensión superior a 5.5V en el pin de alimentación positivo ya que este CI posee una caída

de tensión de 1.4V respecto de la alimentación. Esto se consigue fácilmente con la tensión de +15V que provee el backplane. Asimismo, se alimenta su pin negativo con una tensión de -5V por los mismos motivos.

Por otro lado, para conseguir dicho rango en la señal invertida de 0-4.096V, es necesario una tensión de 2.048V en su pin positivo de entrada. Los pines 12 y 26 del ADC (Fig. 3.2) tienen una tensión constante de 4.096V. Luego a partir de cualquiera de los dos mediante un divisor de tensión y un amplificador de ganancia unitaria se consiguen los 2.048V buscados. Asimismo, es importante destacar que las resistencias utilizas en el inversor de ganancia unitaria deben ser precisas para evitar una diferencia de amplitud respecto a la otra señal diferencial de salida del AO en modo seguidor. Por tal motivo, se utilizaron resistencias de 0.1 % de toleracia.

Por último, a la salida del CI LT1819 de la Fig. 3.5a se agrega un filtro pasabajos, recomendado por la hoja de datos, para atenuar el ruido propio del amplificador operacional.

Protección de sobretensión

Debido a la sensibilidad que presentan los circuitos integrados a las sobretensiones generadas por eventuales cortocircuitos con las tensiones de alimentación de +15V o -15V, o un incorrecto seteo de la ganancia del amplificador de instrumentación, se agregan diodos como mecanismo de protección en diferentes terminales de entrada. En la Fig. 3.6 se puede ver el esquema correspondiente al bloque completo de acondicionamiento. Como se observa, los pines A0 y A1 están protegidos mediantes diodos de protección de ESD (Electro-Static Discharge, por sus siglas en inglés) y, a continuación de las etapas de amplificación, diodos Schottky para enclavar la tensión de salida de cada etapa a la tensión de alimentación del ADC (0V a 5V). Se utilizan diodos Schottky debido a su baja

caída de tensión en directa.

Por un lado, para determinar el diodo de protección de ESD se debe tener en cuenta la tensión nominal de los pines de selección de ganancia en estado alto de 5V. Por lo tanto, se selecciona un diodo que posea una tensión de operación superior a 5V, con baja corriente de fuga y con una tensión de ruptura lo más baja posible. El componente elegido fue el NSQA6V8AW5T2G que cuenta con 4 diodos con una tensión de ruptura típica de 5V, y una baja corriente de fuga a la tensión de operación de 5V.

Por otro lado, para seleccionar los diodos Schottky apropiados se debe asegurar, en caso de que el amplificador sature a alguna de las fuentes de +15V o -15V, que la corriente que circula por los diodos no sea mayor a su valor máximo admisible. En el peor de los escenarios, se asume que la corriente que circularía por los diodos es la corriente de cortocircuito del amplificador. Dicha corriente a la salida del AD8250 y el LT1819 es de 37mA y 200mA respectivamente. El diodo elegido fue el BAT54SWFILM ya que soporta una corriente máxima en directa de 300mA y cuya caída de tensión en directa es menor a 900mV en el peor caso, protegiendo de esta manera las entradas del ADC.

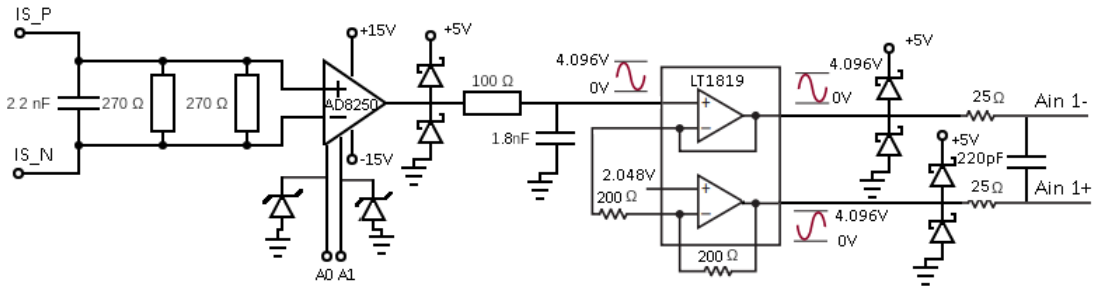


Figura 3.6: Circuito completo del bloque de acondicionamiento

3.4. Adecuación de niveles de tensión

Una vez digitalizadas las señales, sus tensiones de salida deben adaptarse a las de la FPGA para poder ser luego procesadas y, al mismo tiempo, las señales de control generadas por la FPGA deben adaptarse al ADC. Las señales de salida del ADC en alto son de 2.5V y la FPGA trabaja con 3.3V. Para este propósito se utilizó el transductor SN74AVC4T245DR de 4 bits. La Fig. 3.7 resume la interconexión del mismo.

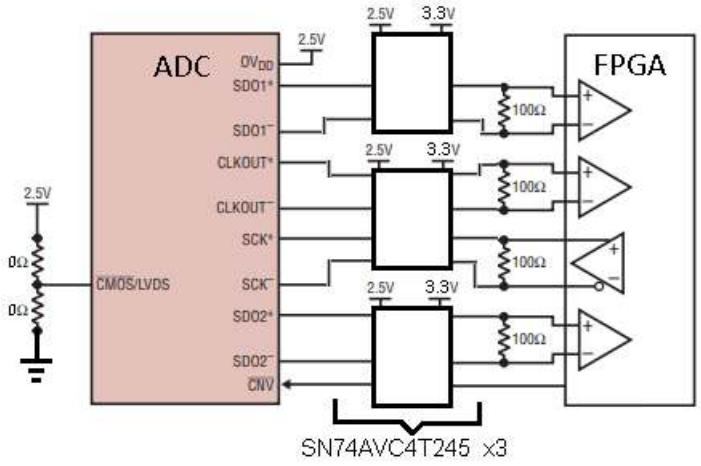


Figura 3.7: Interconexión del transductor entre el ADC y la FPGA

3.5. Diseño final del PCB

El diseño del PCB del circuito de adquisición se realizó en cuatro placas de 160mm x 100mm mediante el software Altium Designer. A continuación se presentan los esquemáticos.

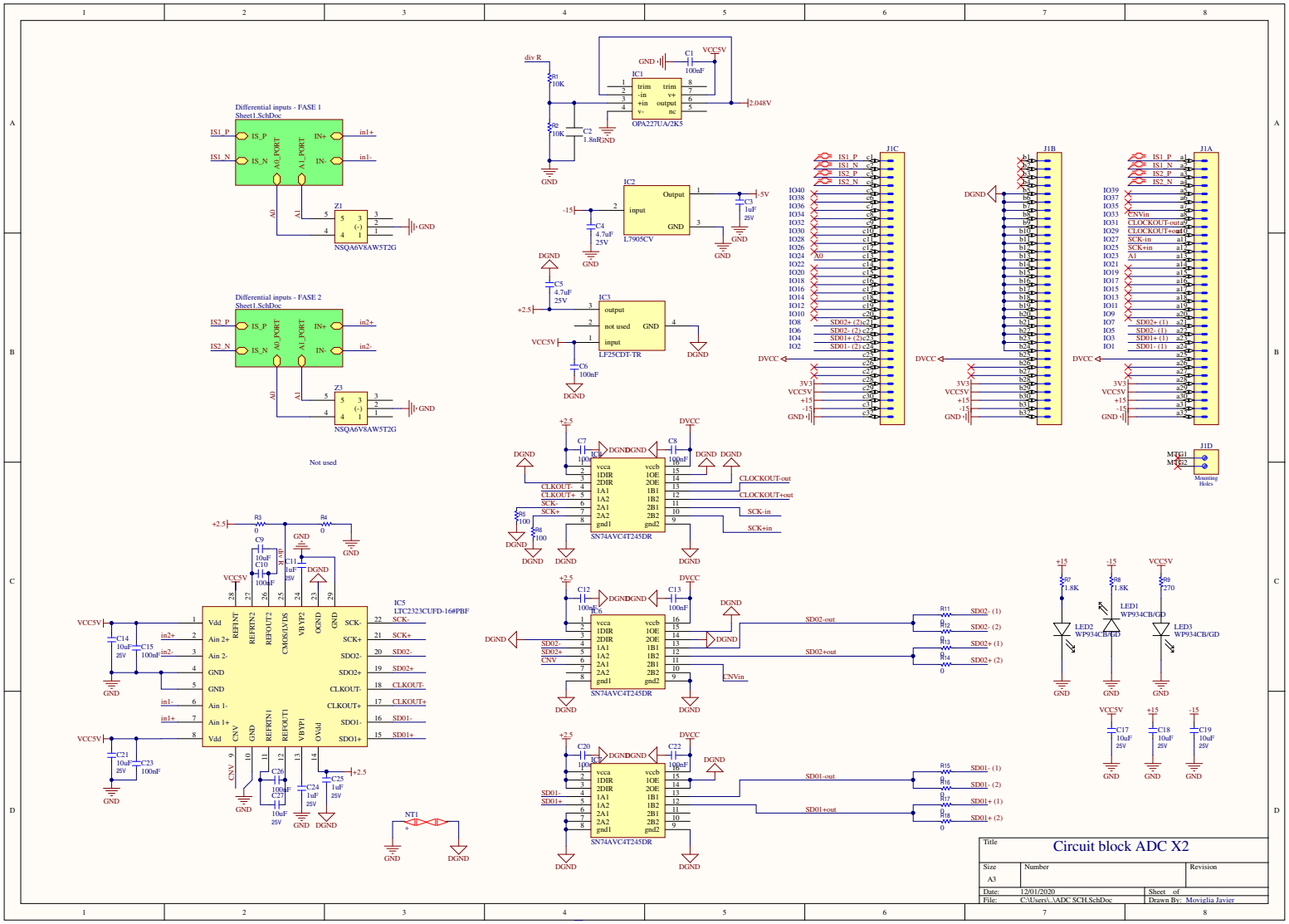


Figura 3.8: Esquemático del circuito de adquisición

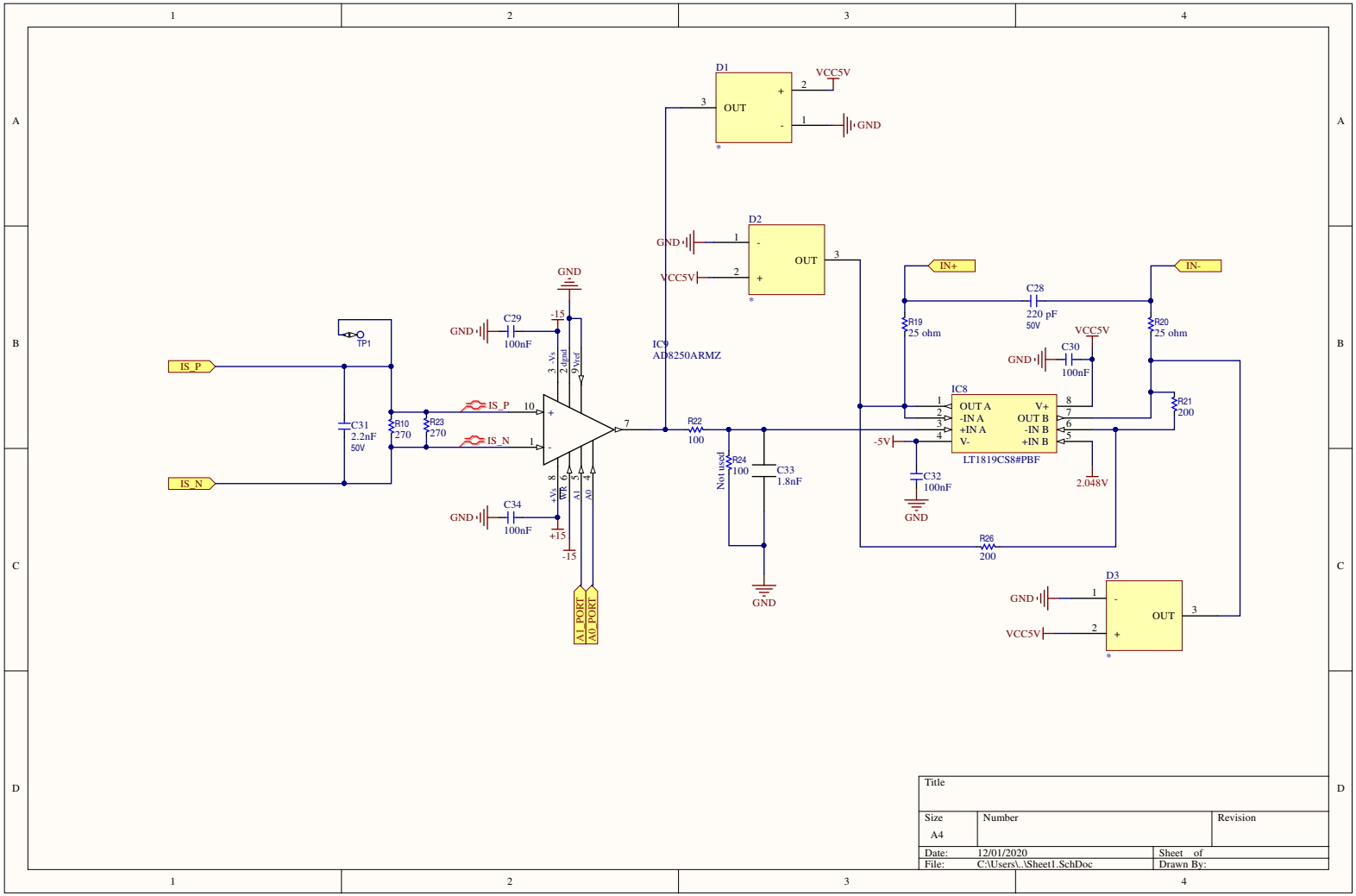


Figura 3.9: Esquemático del bloque de acondicionamiento

Se usaron cuatro capas con una distribución estándar: dos de señal, una de alimentación y una de masa. Respecto a la capa dedicada a los planos de masa, se enfatizó una separación entre la masa digital y la masa analógica para evitar que el contenido de corrientes digitales de alta frecuencia afecte a los circuitos analógicos. Además, se procuró que cada plano tuviese la menor cantidad de cortes para que la corriente tenga un retorno sin interrupciones. El punto de conexión entre ambas masas se realizó en el ADC. Respecto a la distribución de capas, se colocaron las capas de señal en TOP (capa superior del PCB) y BOTTOM (capa inferior del PCB), y las de masa y alimentación en las capas internas. De esta forma, se logra una simetría entre el contenido de cobre de cada capa y también respecto al centro. Esto permite evitar que la placa sufra efectos de torsión (warping) debido a las diferencias entre el coeficiente de expansión térmico del cobre y del FR4. Asimismo se ubicó el plano de masa adyacente a la capa TOP para reducir el área de los lazos y en consecuencia las inductancias parásitas.

En el diseño se contempló utilizar la menor cantidad de vías posible de modo de no introducir efectos parásitos. En relación a las dimensiones de los trazos, al no trabajar con corrientes elevadas no existen restricciones al respecto, con lo cual el criterio utilizado consistió en usar trazos mayores a los mínimos especificados por el fabricante.

Respecto a la ubicación de los componentes, teniendo en cuenta que la placa debía adecuarse al backplane del prototpo existente, primero se colocaron los conectores y luego se ubicaron el resto de los componentes por bloques lógicos, cuidando que las pistas sean lo más cortas posibles para minimizar los efectos parásitos. En este sentido, los capacitores de desacople se ubicaron junto a los pines de alimentación de cada circuito integrado, respetando los valores recomendados por los fabricantes. Esto permite reducir el ruido generado por los circuitos

digitales en las tensiones de alimentación y mejorar el rechazo al ruido de fuente en los circuitos integrados analógicos. Las figuras de las siguientes páginas muestran el circuito impreso de cada capa.

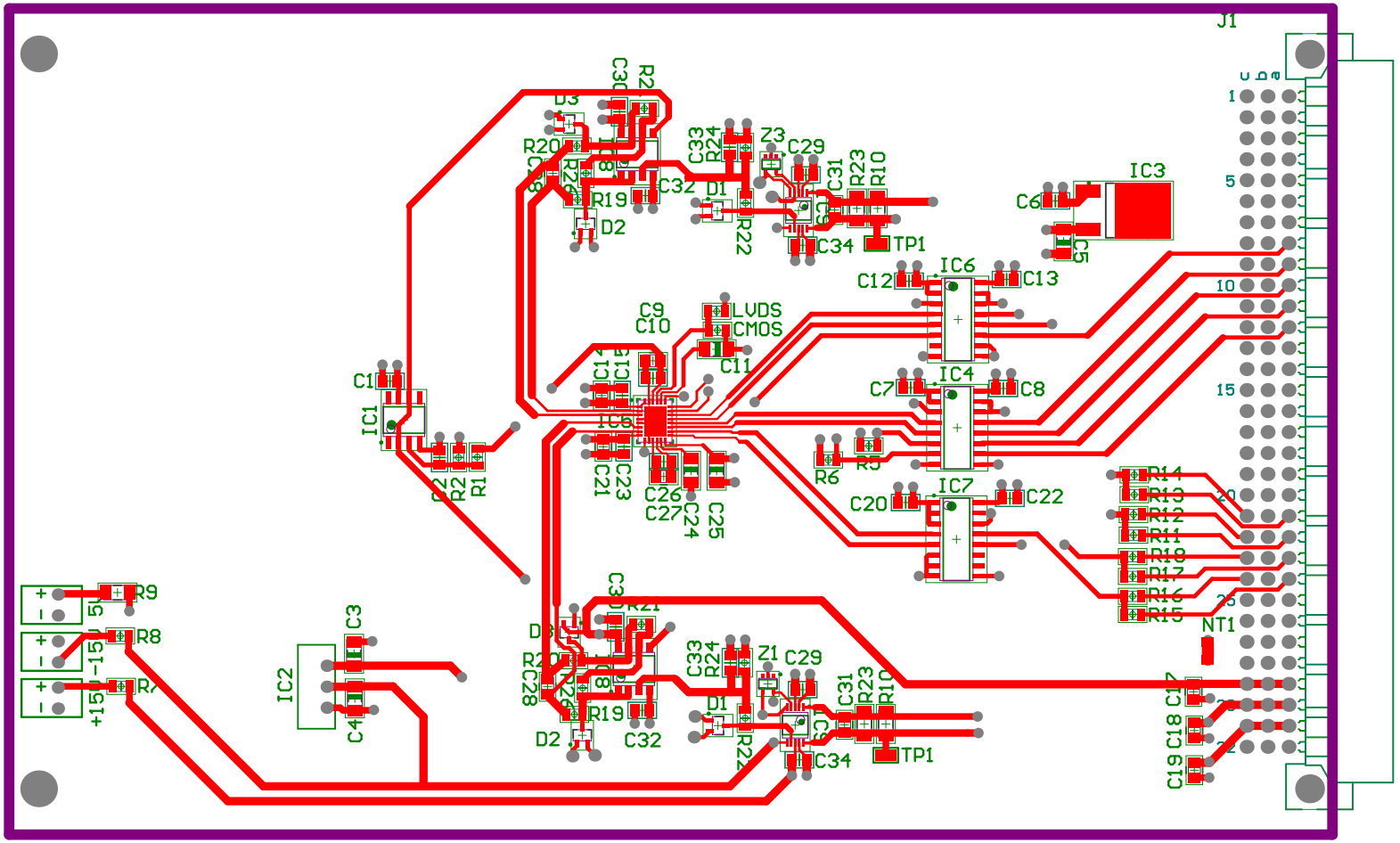


Figura 3.10: Capa superior del PCB

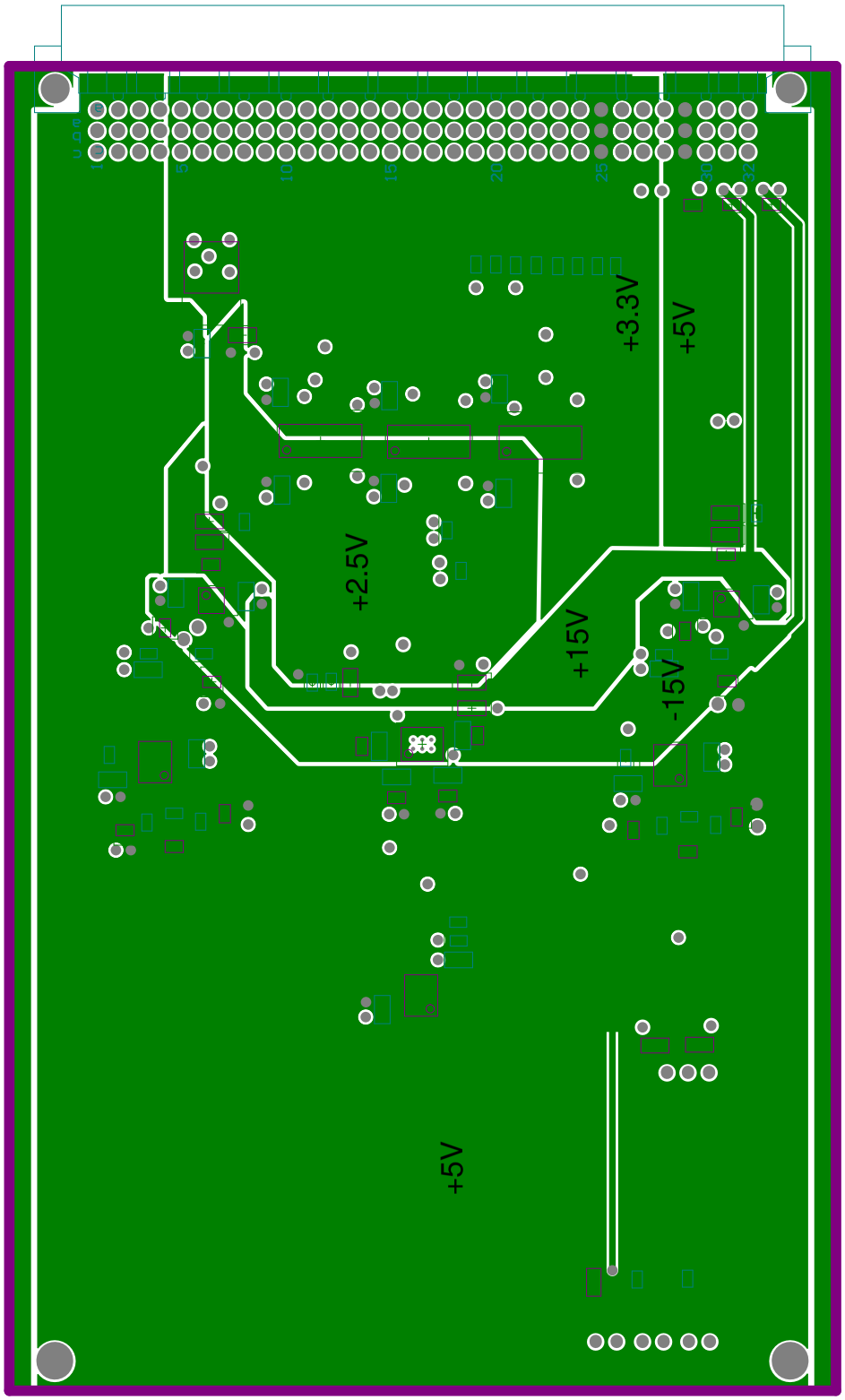


Figura 3.11: Plano de tensiones de alimentación

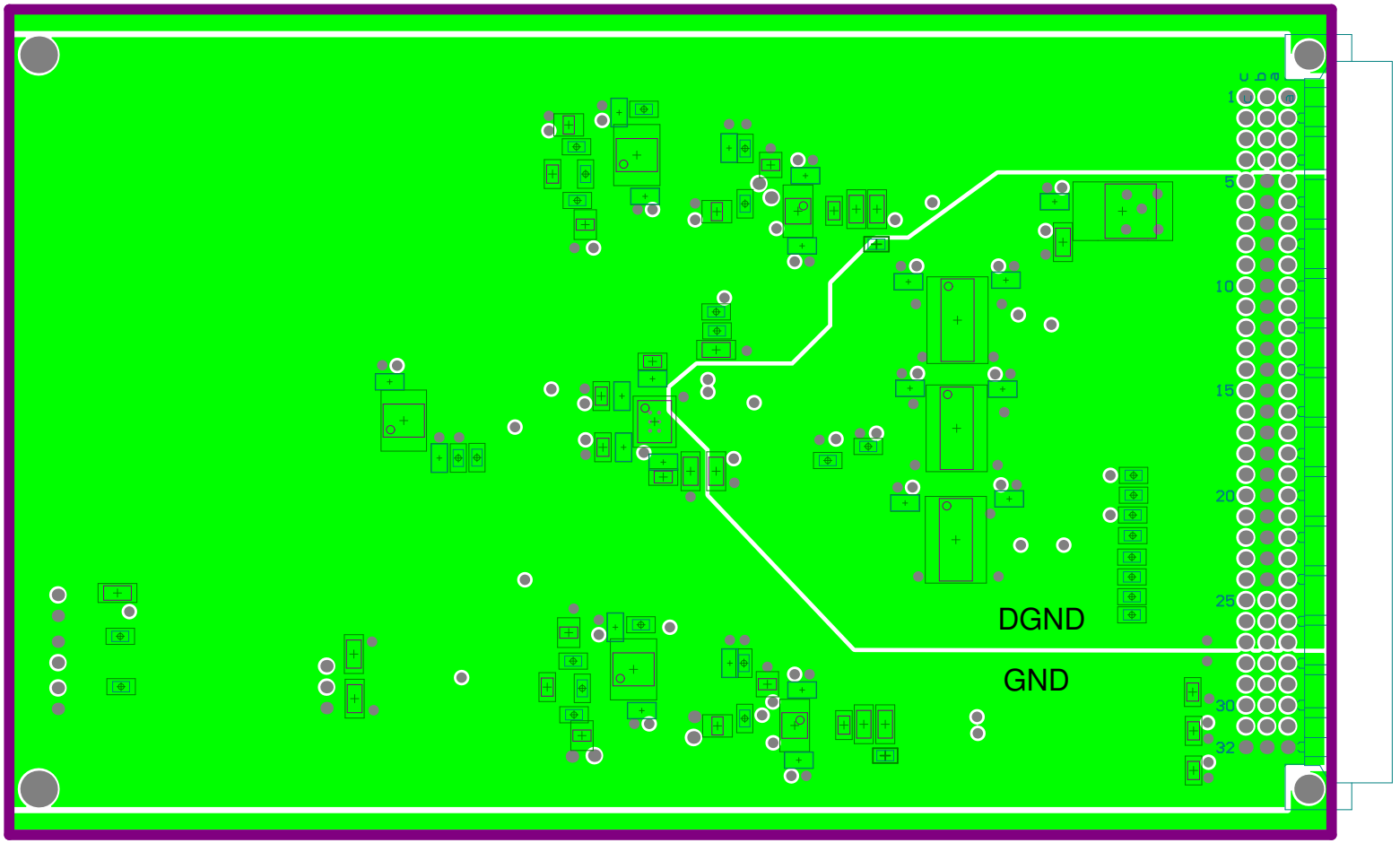


Figura 3.12: Plano de masa analógica y digital

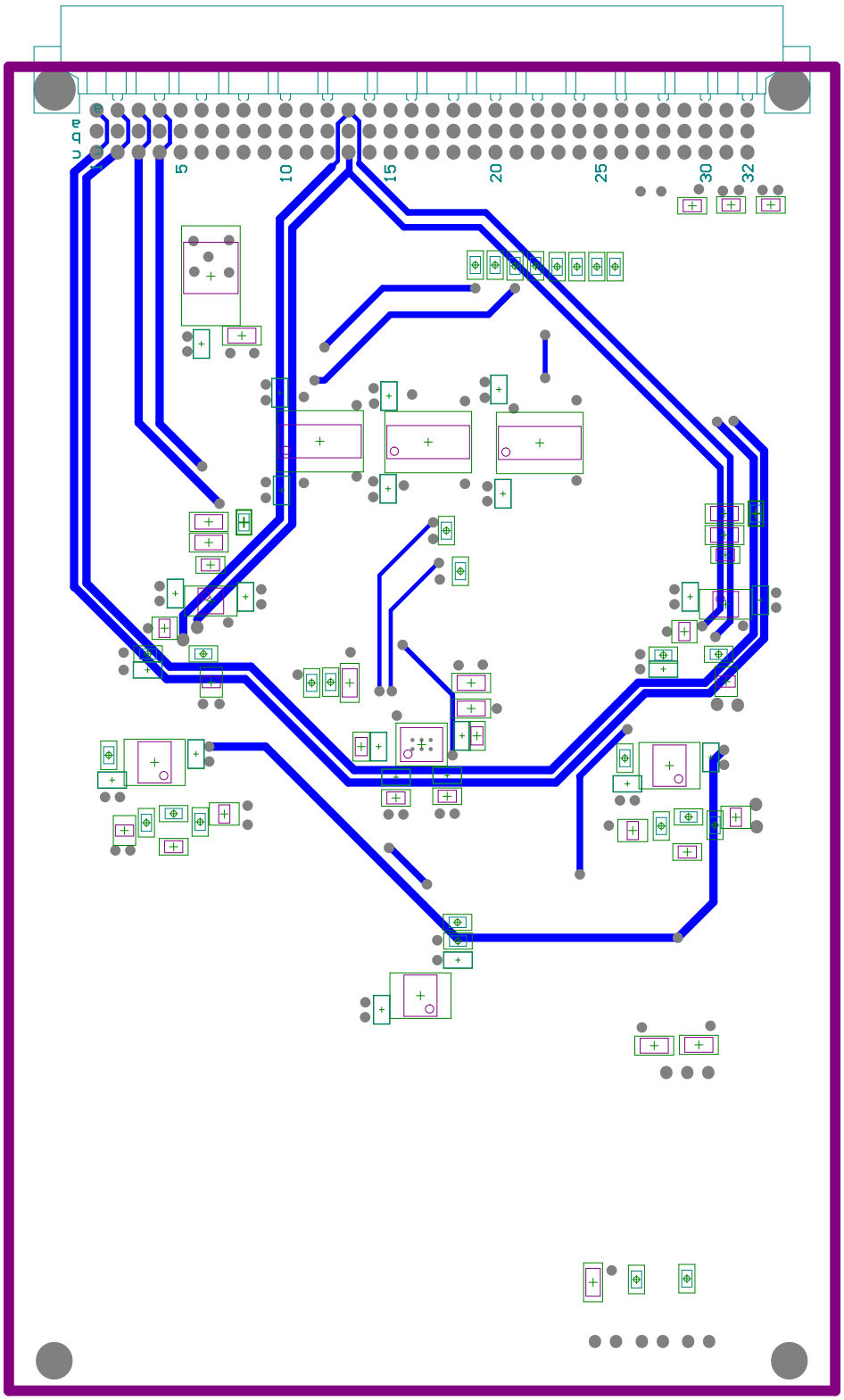


Figura 3.13: Capa inferior del PCB

El resultado del diseño final de las placas se puede ver en la Fig. 3.14 y 3.15. En la primera figura se muestra el diseño 3D obtenido mediante el programa Altium Designer y en la segunda figura se muestra las cuatro placas ya terminadas. Se ha utilizado un tamaño estandar para un gabinete de 19" y altura 3U, haciéndolo compatible con el backplane existente, tal como se observa en la Fig. 3.16.

En estas condiciones, es posible manipular las fases del convertidor digitalizadas y aplicar el algoritmo para hallar el ordenamiento óptimo. Esto se aborda en el siguiente capítulo.

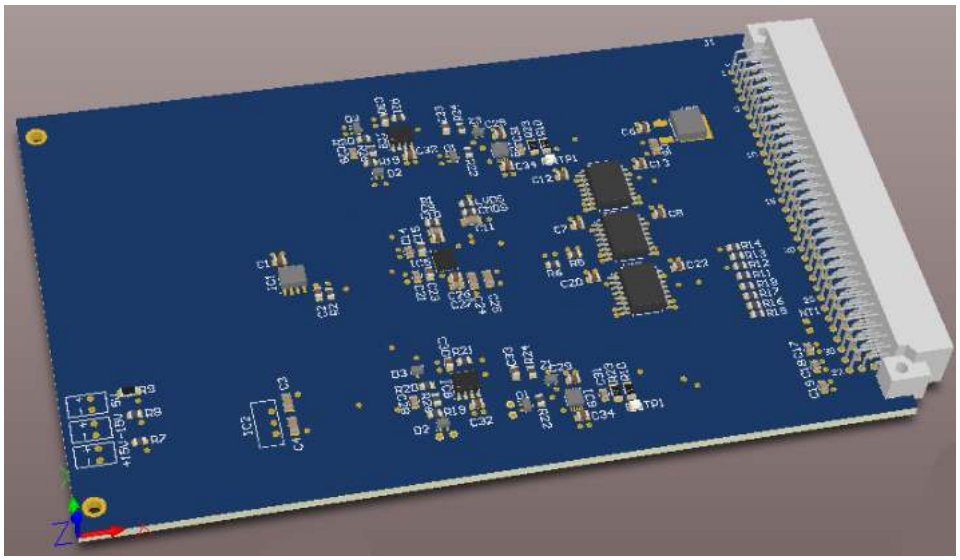


Figura 3.14: Diseño final de la placa de adquisición realizado mediante el programa Altium Designer

Figura 3.15: Placas de adquisición terminadas

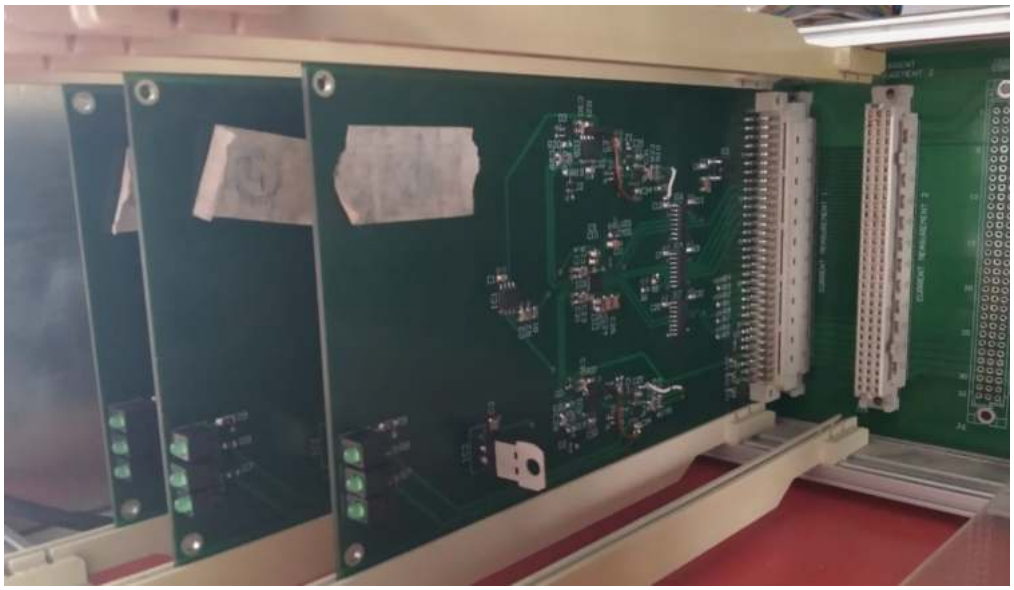


Figura 3.16: Placas ya conectadas al backplane (sólo se visualizan 3 placas)

Capítulo 4

Implementación del sistema en una plataforma digital

4.1. Introducción

En este capítulo se desarrollan los aspectos más relevantes de la implementacion del sistema en la plataforma digital.

4.2. Selección de la plataforma

En este proyecto se requiere procesar varias señales, ejecutar un gran número de operaciones de cálculo y modularidad. Esto último resulta de importancia ante posibles variaciones futuras de las especificaciones del convertidor o la forma de operar del algoritmo, como puede ser el número de fases o la cantidad de bits de resolución de las señales. Una FPGA reune todas las condiciones para implementar estas características. Este es un dispositivo semiconductor conformado por una matriz de bloques lógicos configurables (CBLs) interconectados entre sí. Su funcionalidad puede ser programada y reprogramada a través de lenguajes de

descripción de hardware HDL. Además, en una FPGA el código escrito se procesa en paralelo, a diferencia de un microcontrolador que lo realiza de manera secuencial, facilitando así la operación modular requerida en este proyecto.

La Fig. 4.1 presenta un esquema del proyecto en donde se muestra los bloques principales que fueron configurados en la FPGA. Allí se evidencia el concepto de modularidad y la interconexión entre los mismos. Estos son luego inicializados e interconectados en un único bloque. Cada uno de ellos es escrito en código VHDL e implementado posteriormente en una Spartan XC3S1600E.

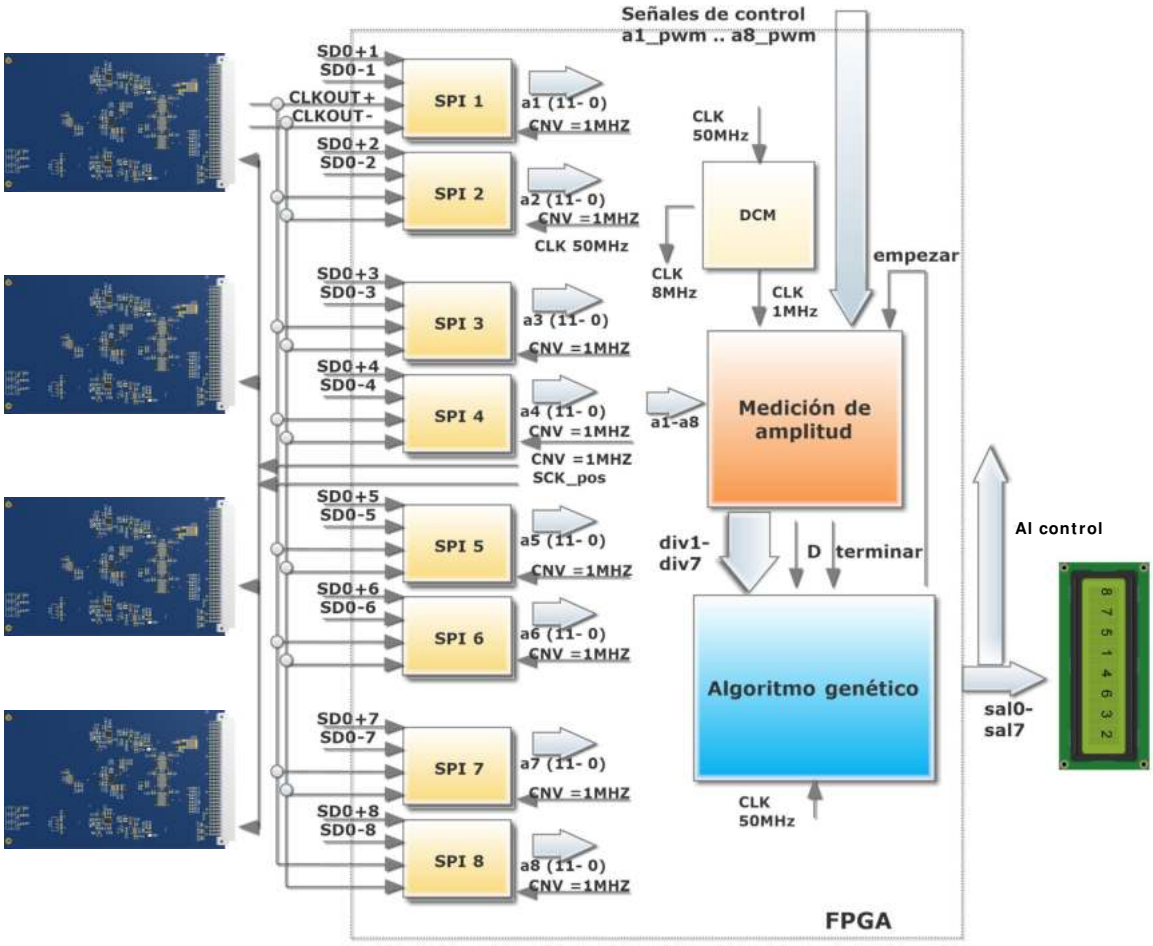


Figura 4.1: Esquema del proyecto con detalles de la configuración de la FPGA

En las siguientes secciones se explica cada uno de los módulos. Debido a la extensión de algunos de los códigos, se utilizan distintos medios para su explicación, como diagramas de flujo, pseudo código o extractos relevantes del mismo. Por otro lado, se ha adoptado una frecuencia de muestreo de 1MHz y una frecuencia de conmutación de 25KHz, es decir, 40 muestras por ciclo de señal, para la experimentación del proyecto.

4.3. SPI: Interfaz Periférica Serie

Cada una de las señales de las fases ya digitalizadas que provienen del ADC son un flujo de bits serie que deben ser reagrupados apropiadamente para la interpretación de sus valores en palabras de 12 bits. Esto se consigue con este primer bloque que no es más que un registro de desplazamiento a la frecuencia con la que arriban los bits de datos. Asimismo, las señales que llegan y salen del ADC deben cumplir determinados requisitos de señalización y temporización para el correcto funcionamiento del mismo. En función de lo anterior, la hoja de datos del ADC LTC2323 provee el siguiente diagrama temporal:

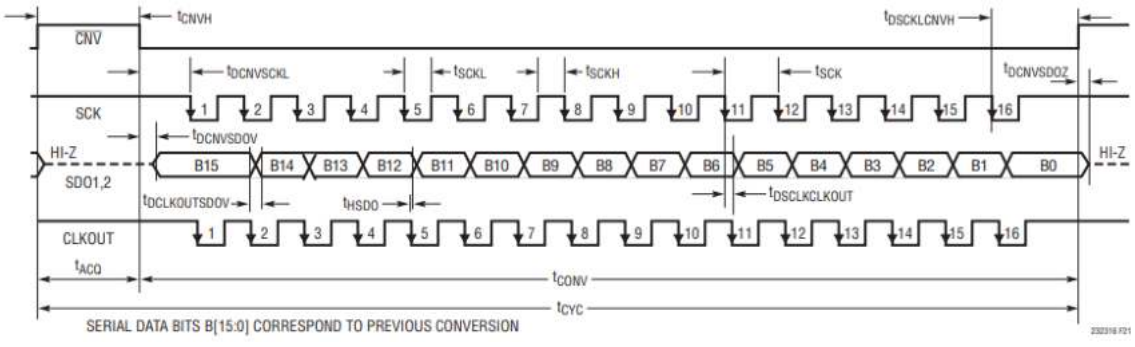


Figura 4.2: Diagrama temporal de la adquisición de señales

En la Fig. 4.2 se observan cuatro señales:

- CNV: Señal de reloj que indica el comienzo de la conversión y, por lo tanto, debe setearse a 1MHz.

- SCK: Entrada de reloj de datos en serie. El flanco descendente de este reloj desplaza el resultado de conversión MSB (bit más significativo) primero a los pines SDO. Por lo tanto requiere de 16 pulsos por cada ciclo de la señal de conversión.

- SDO1,2: Salida de datos serie del ADC (canal 1 y 2).

- CLKOUT: Salida de reloj de datos en serie. Esta señal es similar a SCK pero sincronizada con los datos de salida, para reducir problemas de retardo entre SCK y los datos en el extremo receptor (FPGA).

Las primeras dos señales, CNV y SCK, son salidas de la FPGA y, por lo tanto, deben ser generadas mediante código. Sin embargo, la FPGA Spartan XC3S1600E no cuenta con un clock interno de 1MHz, frecuencia necesaria para crear ambas señales; no obstante, ésta se puede obtener a partir de su clock de 50MHz. En particular, las FPGA Xilinx poseen bloques de lógica dedicados exclusivamente a funciones de control y generación de señales de reloj denominados DCM (Digital Clock Management, por sus siglas en ingles). Entre algunas ventajas de su utilización se encuentran la sincronización de la señal de reloj generada que llega a los distintos componentes, producir corrimientos de fase, multiplicar o dividir la frecuencia de entrada, entre otros. Debido a que la mínima frecuencia que se puede obtener a la salida de un DCM, con la FPGA de operación, es de 5MHz, se escoge un valor de 8MHz. Esto se hace así porque resulta más sencillo la implementación de un divisor de frecuencia por una potencia de 2. El siguiente código VHDL corresponde a un divisor por 8 requerido para lograr la frecuencia deseada de 1MHz:

Algoritmo 4.1 Divisor de frecuencia por 8

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity clock_div_pow2 is
port(
  i_clk          : in  std_logic;
  o_clk_div8     : out std_logic );
end clock_div_pow2;

architecture rtl of clock_div_pow2 is
signal clk_divider: unsigned(2 downto 0);
begin
p_clk_divider: process(i_clk)
begin
  if (rising_edge(i_clk)) then
    clk_divider   <= clk_divider + 1;
  end if;
end process p_clk_divider;
o_clk_div8 <= clk_divider(2);
end rtl;
```

Al contar con la señal de reloj de 1MHz es posible recrear las formas de onda de CNV y SCK en el programa principal. Esto se consigue por medio de los procesos desarrollados en 4.2 y 4.3, respectivamente.

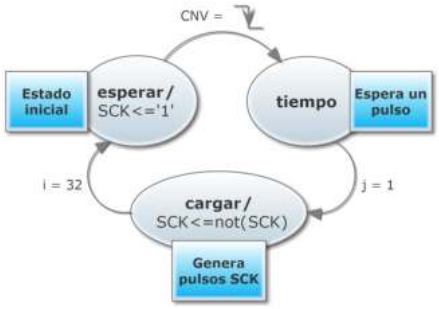
Algoritmo 4.2 Señal CNV

```
CNV<= CNV_aux;

process(clk_buf, clk1)
begin
  if clk_buf'event and  clk_buf='1' then
    case state_next is
    when esperar=>
       clkprevio<=clk1;
       if clkprevio='0' and clk1='1' then
         state_next<= cargar;
         CNV_aux<='1';
       end if;
    when cargar=>
      if (i=10) then
        i<=0;
        CNV_aux<='0';
        state_next <= esperar;
      else
        state_next<= cargar;
        i<=i+1;
        CNV_aux<='1';
      end if;
   end case;
  end if;
end process;
```

Algoritmo 4.3 Señal SCK

```
1  SCK<=SCK_aux;
2
3  process(clk_buf,CNV)
4  begin
5   if clk_buf'event and  clk_buf='1' then
6     case state_next is
7
8       when esperar=>
9          SCK_aux<='1';
10         CNV_previo<=CNV;
11         if CNV_previo='1' and CNV='0' then
12           state_next<= tiempo;
13         end if;
14
15      when tiempo=>
16         if (j=1) then
17             j<=0;
18             state_next <= cargar;
19         else
20             j<=j+1;
21             state_next <= tiempo;
22         end if;
23
24      when cargar=>
25         if (i=32) then
26             i<=0;
27             SCK_aux<='1';
28             state_next <= esperar;
29         else
30             SCK_aux<= not SCK_aux;
31             i<=i+1;
32         end if;
33    end case;
34   end if;
35 end process;
```



En el Alg. 4.2 se crea la señal CNV o de conversión a partir de la señal clk1, es decir la salida del DCM a 1MHz, siendo clk_buf el reloj interno de 50MHz de la FPGA. Cada 10 pulsos de clk_buf pasa a estado alto CNV_aux y luego vuelve a estado bajo hasta el próximo flanco ascendente de clk1.

Por otro lado, en el Alg. 4.3 se muestra el proceso de generación de SCK a partir de CNV. Esta es una máquina de tres estados. En el primero se espera que llegue un flanco descendente de la señal CNV; cuando esto sucede, se cuenta un ciclo de señal de clk de 50 MHz ($j = 1$). La variable j modifica el tiempo $t_{DCNVSCKL}$. En el tercer y último estado, se cuentan 32 pulsos de la señal clk_buf

y en cada uno de sus flancos ascendentes se cambia el valor de SCK_aux dando como resultado 16 pulsos de la misma a una frecuencia de 25MHz.

En el Alg. 4.4 se expone el proceso de muestreo y almacenamiento de los 12 bits más significativos de cada muestra en el registro SD0_pos_reg, ya que el ADC se lo configura en su modo CMOS, mediante la señal de reloj CLOUT_pos. La variable result_aux es un registro auxiliar que almacena los últimos cuatro resultados para luego promediarlos, es decir, un filtro digital. Este es el primer proceso que conforma el bloque SPI.

Algoritmo 4.4 SPI: Muestreo y almacenamiento

```
process(CLKOUT_pos, CNV)
begin
 if CNV='0' then
   if CLKOUT_pos'event and  CLKOUT_pos='1' then
       k<=k+1;
       SD0_pos_reg<= SD0_pos_reg(10 downto 0)&SD0_pos;
       if (k=12) then
         result_aux(3)<=result_aux(2);
         result_aux(2)<=result_aux(1);
         result_aux(1)<=result_aux(0);
         result_aux(0)<=signed(SD0_pos_reg);
       end if;
   end if;
 else
   SD0_pos_reg<= (others=>'0');
   k<=0;
 end if;
end process;
```

Nótese así mismo en la Fig. 4.1 que, por cuestiones de diseño y espacio en el ruteo de las señales, la señal CLKOUT_pos proviene de una sóla placa. Si bien esta práctica no es óptima, ya que el funcionamiento de las cuatro placas depende de sólo una de ellas como se verá en el siguiente capítulo, esto no es crítico para los resultados del proyecto ya que la diferencia de longitud de trazos no es lo suficientemente grande como para generar problemas de sincronización debido a diferencias de retardo.

Por último, se efectúa el promedio del registro result_aux y se lo almacena la señal de salida result. Por otro lado, las palabras que genera el ADC están

representadas en complemento a 2. Como en el módulo de medición de amplitudes se trabaja con señales enteras positivas, un paso previo es sumarle 2048 a cada elemento del registro result_aux, tal como se muestra a continuación:

Algoritmo 4.5 SPI: Promediado y resultado

```
process(clk)
begin
 if clk'event and  clk='0' then
   resultunsigned<=to_unsigned(
   (to_integer(result_aux(0)+"011111111111"+"000000000001")
   +to_integer(result_aux(1)+"011111111111"+"000000000001")
   +to_integer(result_aux(2)+"011111111111"+"000000000001")
   +to_integer(result_aux(3)+"011111111111"+"000000000001")),14);
   result<= std_logic_vector(resultunsigned(13 downto 2));
 end if;
end process;
```

Un detalle a destacar es el flanco con el que se inicia el proceso anterior. Este se activa con el flanco descendente de la señal clk ya que en el Alg. 4.4 se activa con el flanco ascendente. Esto permite la correcta actualización de los datos en la variable result, una vez cargados los 12 bits en el registro auxiliar.

4.4. Algoritmo de medición de amplitud

La descripción teórica de este bloque ya fue abordada en la Sección 2.2. En este apartado se desarrolla su implementación en FPGA.

De acuerdo a (2.7) y sabiendo que la cantidad de muestras por cada ciclo de las señales de corriente de fase es igual a 40, el filtro resultante para llevar a cabo la medición de las ampitudes de las mismas es el siguiente:

$$H_{SGT}(z) = \frac{(1 - z^{-40})}{(1 - z^{-1})} \tag{4.1}$$

O bien, escrito como ecuación en diferencias:

$$y(n) = x(n) + y(n-1) - x(n-40) \tag{4.2}$$

Recordando lo explicado en la Sección 2.2 para obtener la amplitud a partir del cálculo de la componente de continua es necesario multiplicar la señal por una senoidal de la misma frecuencia, en este caso de 25KHz. Además, la amplitud del ripple se ve afectada por un factor de $A/2$ al multiplicarse por la onda senoidal, siendo A su correspondiente amplitud pico. Debido a que el algoritmo de reordenamiento opera con la relación entre amplitudes de fase, este factor no influye en el cálculo debido a que es el mismo en todas las señales que multiplican a cada ripple.

El diagrama completo con todos los bloques utilizados para la realización del código se muestra en la Fig. 4.3

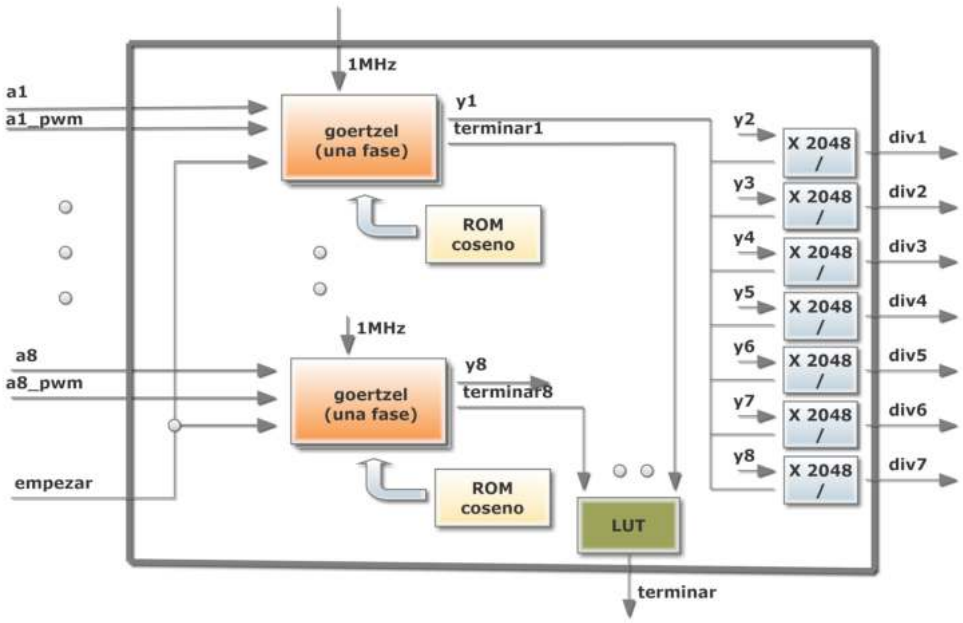


Figura 4.3: Diagrama en bloques del módulo Medición de amplitud

El subloque «goertzel (una fase)» se encarga de resolver la ecuación en diferencias (4.2) para una sola fase. Previamente es multiplicada por un coseno en fase con la misma, que se obtiene de una memoria ROM de 40 muestras. Lo

siguiente es un extracto de su código:

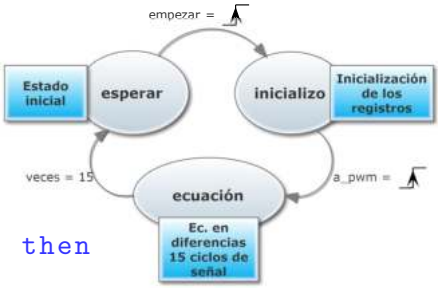
Algoritmo 4.6 Goertzel (una fase)

```
aux<=a*cos;
y<=y2;
process (clk, empezar)

begin
if CLK'event and  CLK='1' then
 case state_next is
    when esperar=>
     terminar_aux<='0';
     y1<= (others => '0');
     empezar_previo<=empezar;
     if empezar='1' and empezar_previo='0' then
      state_next<= inicializo;
     end if;

    when inicializo =>
     a_pwm_previo<=a_pwm;
     if a_pwm='1' and a_pwm_previo='0' then
      for i in 0 to 39 loop
        x1(i)<= (others => '0');
      end loop;
      x1(40)<=aux;
      n<="000001";
      state_next <= ecuacion;
     end if;

    when ecuacion =>
     y1<= x1(40)+y1-x1(0);
     x1(0 to 39)<= x1(1 to 40);
     x1(40)<=aux;
     if n="101000" then
        n<="000001";
        if veces=15 then
           y2<=std_logic_vector(abs(y1));
           terminar_aux<='1';
           state_next <= esperar;
           veces<=0;
        else
           veces<=veces+1;
        end if;
    else
        n <= n+'1';
    end if;
 end case;
end if;
end process;
```



Como se observa en el Algoritmo 4.6, se trata de una máquina de estados para que el proceso sea resuelto de manera secuencial y se compone de los estados *esperar*, *inicializo* y *ecuacion*. En el primero, se espera la señal de inicio del

algoritmo a partir de un flanco ascendente de la entrada *empezar*. En el siguiente, se aguarda a que la señal PWM asociada al ripple de entrada pase a nivel alto para setear la dirección de la memoria ROM que contiene al coseno en el primer valor. En estas condiciones, se puede dar comienzo a la resolución de la ecuación en diferencias en el último estado. El resultado de la multiplicación se almacena en la variable auxiliar *aux* y el resultado final en *y*. Por último, se regresa al primer estado al cabo de 16 ciclos de señal a la espera de un nuevo flanco ascendente de *empezar*.

De esta manera, el módulo medición de amplitud realiza ocho instancias del subloque Goertzel de una fase y además efectúa las divisiones de la Fig. 4.3. En esta imagen se observa que todas las operaciones están referidas a la primera fase. A fin de efectuar el cálculo de cada una de estas divisiones y conservar la resolución de 12 bits del resultado obtenido, se hace previamente un corrimiento del registro del denominador 11 bits hacia la izquierda, lo que equivale a multiplicar el resultado de la división por 2048, tal como se muestra en la figura anterior. Recordando del capítulo anterior que las toleracias de los inductores se asumían entre el 1 y el 5 %, el rango posible de valores a la salida del bloque de medición de amplitud es de 1853 a 2267, y por lo tanto, 12 bits son suficientes para representar estos valores, tal como se propuso.

Finalmente, el subloque «LUT», simplemente espera que el cálculo de la amplitud de cada ripple haya finalizado y lo señaliza mediante un pulso de un ciclo de reloj de la señal *terminar*.

4.5. Método de ordenamiento de los ripples de fase

Este es el módulo central y, al mismo tiempo, más complejo en el desarrollo del proyecto en FPGA encargado de hallar el ordenamiento óptimo de las fases.

En la Fig. 4.4 se observa cada uno de los bloques básicos del algoritmo genético aplicado al problema de interés con sus entradas y salidas correspondientes. A continuación, se detallan las partes que componen el algoritmo y su implementación en código.

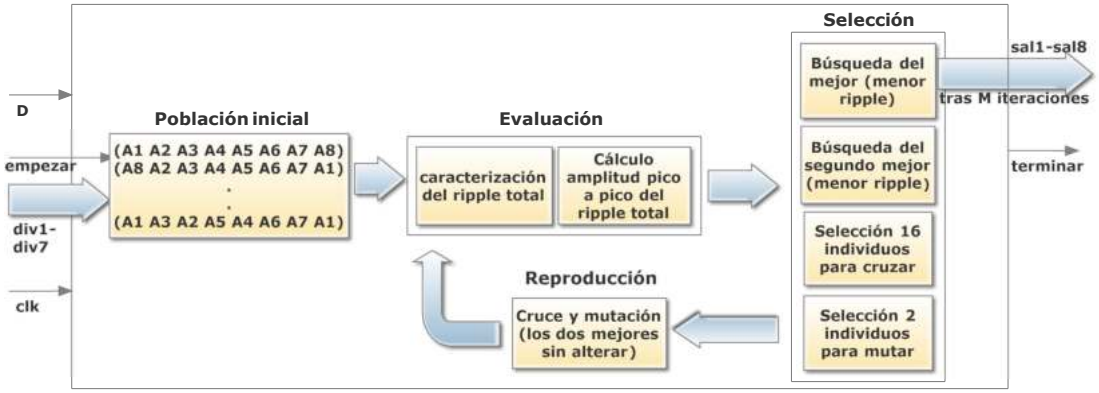


Figura 4.4: Algoritmo genético aplicado al problema

4.5.1. Población inicial

La elección de la cantidad de individuos e iteraciones del algoritmo es un proceso empírico. Esto se debe a que, si bien un número mayor de individuos en la población inicial permite converger a la solución en un menor número de iteraciones, el proceso se torna más lento. Por lo tanto, luego de realizar una serie de pruebas, en este proyecto se establece un valor de 20 para ambos.

Para llevar a cabo este primer paso se crean dos memorias RAM, ambas con los

mismos 20 individuos iniciales elegidos aleatoriamente. Una de ellas conserva la misma información a lo largo de toda una iteración y en la otra se realizan todas las operaciones para la próxima generación, conocidos como *children*. Una vez terminada la iteración, la información de los *children* es pasada a la memoria que no fue alterada, es decir, se transforman en los nuevos «padres». A continuación, el código del componente RAM empleado:

Algoritmo 4.7 Componente RAM

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
        we : in std_logic;
         a : in std_logic_vector(7 downto 0);
        di : in std_logic_vector(11 downto 0);
        do : out std_logic_vector(11 downto 0));
end raminfr;

architecture behavioral of raminfr is

type ramt is array (255 downto 0) of
std_logic_vector (11 downto 0);
signal RAM : ramt;

begin
process (clk)
begin
   if (clk'event and clk = '1') then
     if (we = '1') then
      RAM(conv_integer(unsigned(a))) <= di;
     end if;
  end if;
end process;
do <= RAM(conv_integer(unsigned(a)));
end behavioral;
```

Del código anterior, se observa que se trata de memorias de lectura y escritura de 32Bytes x 12bits. Cada dirección almacena la amplitud del ripple de una fase. Por lo tanto, si se requieren 20 individuos de 8 fases cada uno, se necesitan 160 direcciones.

4.5.2. Evaluación

El objetivo del algoritmo de reordenamiento es hallar el orden que minimice la amplitud de Δi_T, para reducir las exigencias de filtrado en el punto de acople común entre fases. El Alg. 2.1 es un pseudocódigo para la caracterización de la corriente total en estado estacionario . A partir de los picos hallados por dicho algoritmo, se pueden encontrar los valores máximos y mínimos y, de este modo, calcular la amplitud de Δi_T como la diferencia entre ambos. Esta es la función de costo, ya que es el parámetro que se pretende minimizar, evaluando cada individuo a partir de la misma.

En el Algoritmo 4.8 se detalla la búsqueda de los picos máximos y mínimos, *pp* y *pn*, siendo la cantidad de elementos de cada uno de estos vectores igual al número de fases, es decir, 8 para este caso. Las variables *min* y *max* fueron asignadas previamente a los últimos dos picos calculados. El resultado de aplicar la función costo se almacena en la variable *f_costo_aux*.

Algoritmo 4.8 Función costo

```
when buscar =>
  if j<8 then
   if pp(j)>min then
     min<=pp(j);
   end if;
   if pn(j)<max then
     max<=pn(j);
   end if;
   j<=j+1;
  else
   state_next<=cargo;
  end if;

when cargo =>
  j<=0;
  f_costo_aux<=min-max;
  terminar_aux<='1';
  state_next <= esperar;
```

4.5.3. Reproducción

Como ya fue explicado con anterioridad en el marco teórico, la evolución a lo largo de las sucesivas generaciones se lleva a cabo utilizando operadores genéticos de cruce y mutación.

Función cruce

Dada la extensión de su código en VHDL, se resume en la Fig. 4.5 en un diagrama de flujo todas las operaciones llevadas a cabo para el cruce de los individuos $P_1 = A$ y $P_2 = B$. Para ello, se sigue como modelo de referencia el desarrollado en el marco teórico en la Sec. 2.4. El resultado obtenido se almacena en el vector C.

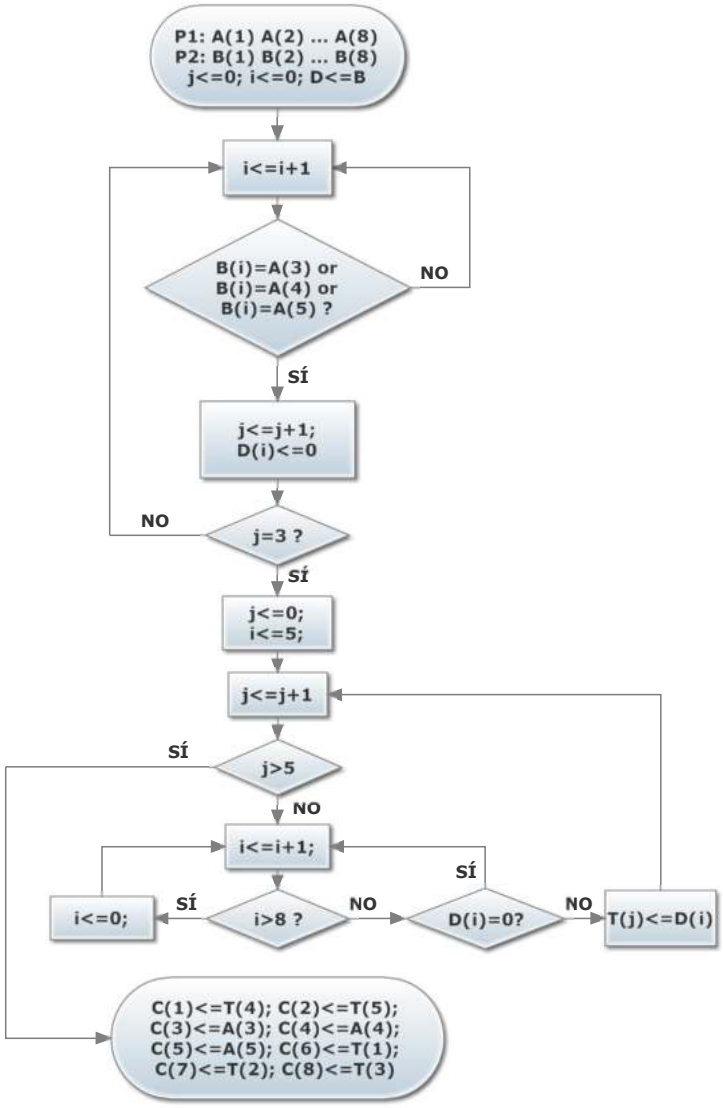


Figura 4.5: Diagrama de flujo de la función cruce

Función mutación

La mutación es un operador genético que ayuda a explorar otros espacios, evitando así óptimos locales. Para el problema en cuestión, esto se consigue sólo por la inversión del orden de dos fases de un individuo seleccionado aleatoriamente.

Para llevarlo a la práctica, se utiliza un generador de secuencia binaria pseudo

aleatoria implementado mediante un registro de desplazamiento de realimentación lineal (LFSR).

Un LFSR de n-bits es una cadena de n registros con una única entrada en uno de los registros de los extremos. Este dato de entrada, bit de entrada al registro de desplazamiento, es el resultado de realizar la operación lógica XOR (o XNOR) entre ciertos y determinados bits del registro de desplazamiento. Obteniendo los valores lógicos a la salida de cada registro de desplazamiento y uniéndolos de modo de formar un vector o bus de datos, se obtiene como resultado una secuencia de $2^n - 1$ números binarios de n-bits que cambia con cada flanco de reloj, cuyo ciclo se repite luego de $2^n - 1$ ciclos de reloj. Se puede demostrar que para cualquier valor de n (numero de bits del registro de desplazamiento) existe al menos una ecuación de realimentación basada en la operación lógica XOR o XNOR que genera una secuencia pseudo-aleatoria de números binarios cuando se unen las n salidas de los n-bits del registro de desplazamiento, que se repite cada $2^n - 1$ ciclos de reloj [14].

Es importante destacar que para que funcione correctamente, el registro debe empezar con todos sus bits en cero en el caso de usar compuertas XNOR y todos en uno si se usan XOR. Como el estado todos unos es el único que no es visitado por el LFSR con compuertas XNOR, para generar 8 números aleatorios se requiere un circuito de al menos 4 registros (Fig. 4.6) y para que se encuentren en el rango 0-7 se debe agregar una condición , donde se restan ocho unidades si el número resultante es mayor o igual a 8. Esto se visualiza en el Algoritmo 4.9.

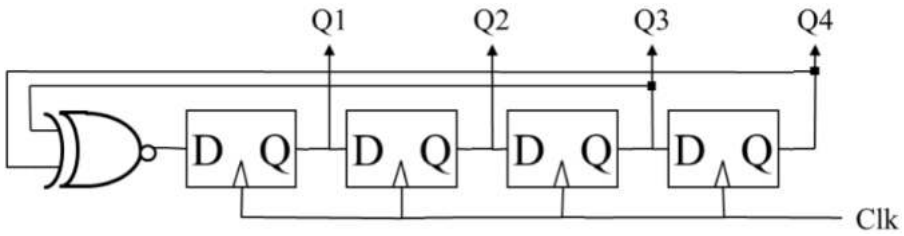


Figura 4.6: 4-bit LFSR

Algoritmo 4.9 Números aleatorios de 0 a 7

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4  use ieee.numeric_std.all;
5
6  entity mutar_fases is
7  port (
8      clk     : in std_logic;
9      i_Enable : in std_logic;
10     o_LFSR_Data : out integer range 0 to 7
11     );
12 end entity mutar_fases;
13
14 architecture RTL of mutar_fases is
15
16 signal r_LFSR : std_logic_vector(4 downto 1) := (others => '0');
17 signal auxiliar: integer range 0 to 15;
18 signal salida: integer range 0 to 7;
19
20 begin
21 p_LFSR : process (clk) is
22  begin
23   if rising_edge(clk) then
24    if i_Enable = '1' then
25     r_LFSR <= r_LFSR(4 downto 1) & (r_LFSR(4) xnor r_LFSR(3));
26     auxiliar<= to_integer(unsigned((r_LFSR(4 downto 1))));
27     if auxiliar>=8 then
28       salida<=auxiliar-8;
29     else
30       salida<=auxiliar;
31     end if;
32    end if;
33   end if;
34 end process p_LFSR;
35 o_LFSR_Data<= salida;
36 end architecture RTL;
```

De forma análoga, para la selección del individuo aleatorio de entre los 20 que se tienen, se necesita un LFSR de 5 registros. Así, cada vez que resulte en

un valor mayor o igual a 21 debe restarse 11 unidades. Con esto se consigue un rango de valores de 1 a 20 (aquí se utilizaron compuertas XOR).

4.5.4. Selección

En cada nueva iteración, para formar la próxima generación de hijos, primero se cruzan los individuos contiguos de a pares almacenados en la RAM padres y en sentido inverso también. Es decir, si P_1 y P_2 son dos individuos contiguos se cruza primero P_1 con P_2 y luego P_2 con P_1. El resultado de esta operación se guarda en la RAM hijos. Posteriormente se mutan dos individuos de los padres y se lo reemplaza por cualquiera de los dos que surgieron del cruce anterior. Finalmente, se hace lo mismo con los dos mejores o *elite* en dos posiciones distintas a la anterior. En resumen, la nueva generación surge de 16 cruces, 2 mutaciones y de los dos mejores. Transcurridas las 20 iteraciones, se selecciona el mejor, se compara el orden de las fases con el inicial y se lo transmite al módulo de la pantalla LCD para su representación. De esta forma, se consigue el orden buscado de las fases que optimiza las exigencias de filtrado.

Capítulo 5

Resultados y discusión

5.1. Introducción

En este capítulo se presentan todos los ensayos realizados para evaluar la implementación y funcionamiento del sistema final. Se analiza para tal fin, el circuito de adquisición y el procesamiento de las señales en FPGA en cada uno de sus bloques principales. Se describe además la puesta en marcha del sistema completo con sus correspondientes opciones de configuración.

5.1.1. Especificaciones de los ensayos

- Frecuencia de conmutación f_{sw}: 25KHz.
- Frecuencia de muestreo f_s: 1MHz.
- Cantidad de bits de procesamiento: 12.
- Ciclo de trabajo de los ripples de fase: 0.18.

5.2. Descripción del sistema final y puesta en marcha

En la Fig. 5.1 se puede ver el sistema físico completo empleado para llevar a cabo la experimentación. Tanto el ordenamiento hallado por la implementación, como la amplitud de cada uno de los ripples de fase pueden ser visualizados en la pantalla LCD de la FPGA.

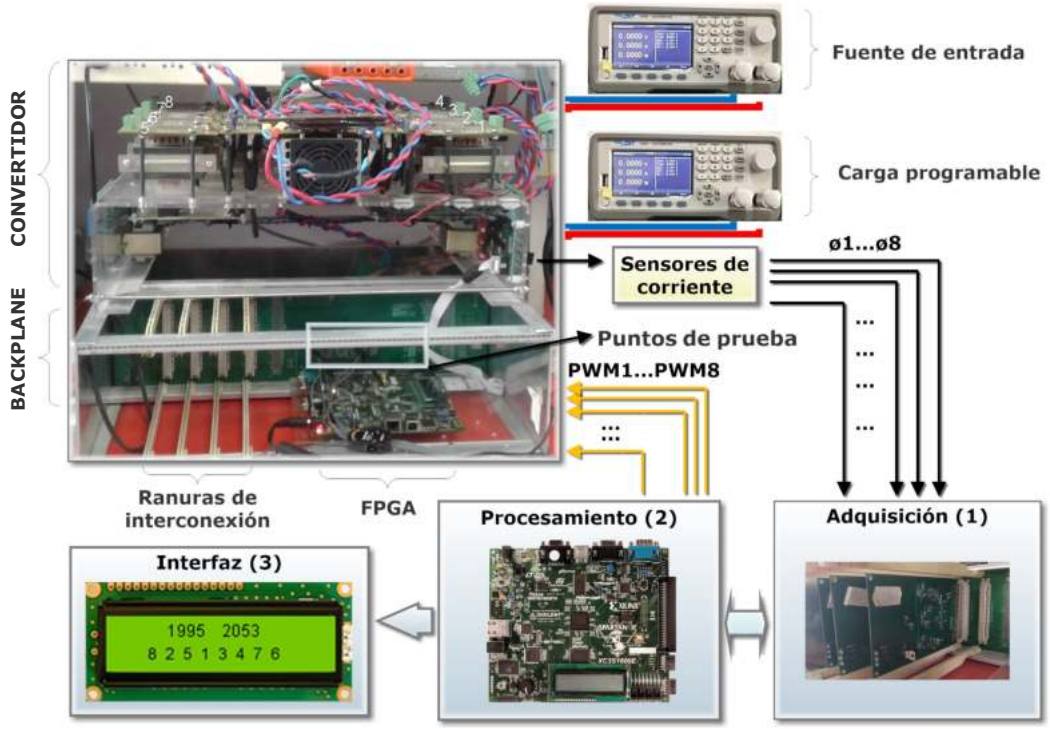


Figura 5.1: Esquema del sistema físico completo

Una vez conectadas las cuatro placas de adquisición y la FPGA en las ranuras correspondientes, y las señales de control al backplane, de acuerdo a la Tabla 5.1, se procede al encendido del equipo. Para la correcta puesta en marcha del equipo, es necesario seguir una determinada secuencia de encendido de sus partes ya que

hay componentes, tales como los amplificadores de instrumentación del circuito adquisidor que pueden dañarse si se ingresa con tensión en sus pines de entrada sin haber sido previamente alimentado. Primero se deben alimentar las placas del circuito adquisidor, luego se conecta el convertidor a la alimentación y por último la fuente de entrada. Finalizado este proceso, debe encenderse T_0 y pulsarse luego P_0, indicados en la Fig. 5.2. Por un lado, T_0 activa el envío de las señales de control al convertidor, mientras que P_0 sirve como señal de Trigger para dar comienzo al procesamiento de las señales en la FPGA. Las llaves S_0 y S_1 permiten cambiar el dato que se visualiza en el LCD: las amplitudes de los ripples de corriente de cada fase. En la Tabla 5.2 se resumen las amplitudes de los ripples que son visualizados para cada combinación S_0-S_1.

Figura 5.2: Pulsador y botones de configuración de la FPGA

Fase	Pin de conexión del Backplane	Fase	Pin de conexión del Backplane
1	IO9	5	IO26
2	IO10	6	IO28
3	IO11	7	IO30
4	IO12	8	IO32

Tabla 5.1: Detalle de conexión de las señales PWM al convertidor

Llaves	Fases visualizadas en el LCD
S_0=0 y S_1=0	1 y 2
S_0=0 y S_1=1	3 y 4
S_0=1 y S_1=0	5 y 6
S_0=1 y S_1=1	7 y 8

Tabla 5.2: Configuraciones de las llaves S_0 y S_1

5.3. Evaluación del hardware de acondicionamiento y adquisición

En esta sección se describen los ensayos realizados para corroborar el funcionamiento del circuito de acondicionamiento y adquisición.

En primer lugar, con las placas conectadas sólamente al backplane, sin conectar aún ninguna señal del convertidor, se midieron y verificaron con el multímetro todas las alimentaciones y salidas de los circuitos integrados.

Luego, con una sóla placa conectada, empezando por la que envía la señal CLKOUT a las otras, y la FPGA conectada al backplane se ingresa con un generador de tensión continua en los pines de entrada de la placa. Para ello, es importante seguir los pasos de la puesta en marcha de la Sec. 5.2. Como ya se indicó en el Cap. 3, el rango de operación de tensión de entrada es de 0 a 4.096V. Por lo tanto, se varía la tensión del generador en estos valores y se toma nota de los valores leídos en la pantalla LCD. De esta manera, se verifica que la adquisición está funcionando correctamente. Este procedimiento se repite para las restantes placas (manteniendo siempre la primera placa conectada) y en cada canal de los ADC. En este paso han podido ser verificadas tres de las cuatro placas satisfactoriamente. La placa faltante tuvo un daño en el ADC y el mismo no pudo ser reemplazado antes del inicio de la pandemia. Una vez verificadas las placas, se propone trazar la curva de valor adquirido vs tension de entrada para cada

canal, y corregir las diferencias de offset y ganancia, tomando como referencia el primer canal.

5.4. Evaluación de la implementación digital

5.4.1. Optimización y análisis de los recursos de la FPGA

Antes de verificar que cada bloque lógico implementado en la FPGA cumpla su función correctamente, se debe sintetizar el código y corroborar que dicha plataforma cuente con todos los recursos para su implementación y se cumplan las restricciones de temporización. Si bien para este proyecto todas estas condiciones son satisfechas sin la necesidad de incluir restricciones en su implementación, es posible optimizar aún más los recursos empleados y la temporización de las señales. Para ello, Xilinx provee la herramienta Planahead que, entre otras funciones, sirve para aplicar restricciones de área. Esto posibilita indicar los sectores en los que deberán ser implementados cada instancia del proyecto en la FPGA. De esta forma, se logra organizar mejor el espacio utilizado de la plataforma digital y reducir los caminos de ruteo de las señales conociendo a priori la interconexión de sus bloques.

La Fig. 5.3 es un esquemático del Planahead de la FPGA empleada (XC3S1600E) con todos los recursos que esta dispone. En dicha figura se puede observar que fueron definidas tres áreas rectangulares, conocidas como Pblocks, para el bloque del algoritmo genético, medición de amplitud y la pantalla LCD. Es en estos sectores donde el ISE implementará luego cada uno de estos bloques. Para la ubicación de los mismos se tuvo en cuenta que las señales de entrada utilizadas se localizan en el borde superior del esquemático. Por lo tanto, el espacio superior es para la implementación de los módulos SPI que deben estar lo más próximo

a las señales de entrada y al módulo de medición de amplitud. En cuanto a la ubicación del bloque de la pantalla LCD, esta se basó en su cercanía al módulo del algoritmo genético y sus pines de salida que se posicionan en el borde lateral derecho. Por otro lado, para definir el tamaño de cada uno de los Pblocks, la misma herramienta provee la información de los recursos necesarios de los Pblocks y los que se disponen con el tamaño establecido.

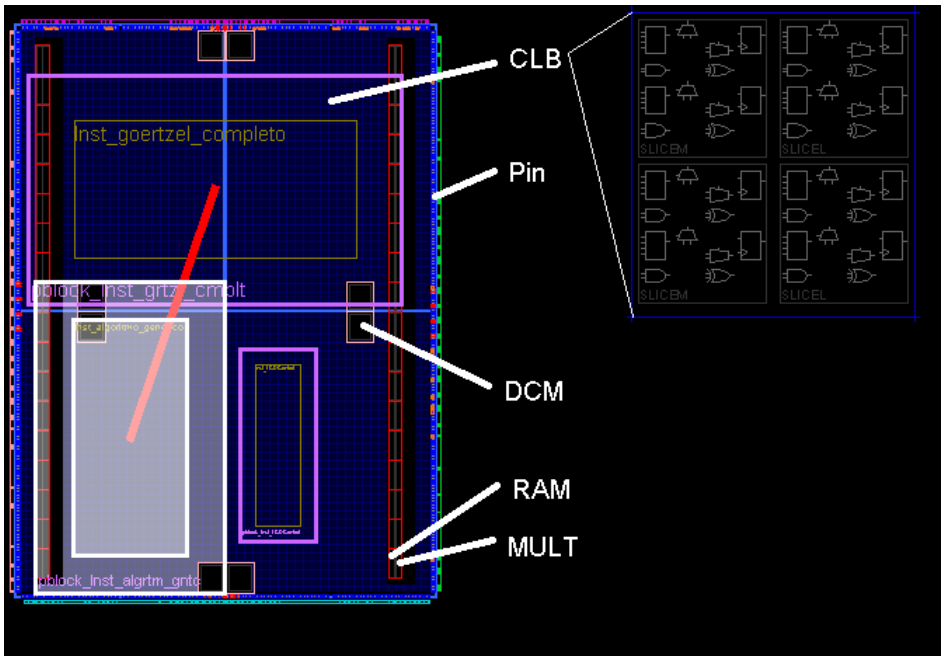


Figura 5.3: Pblocks definidos en Planahead

Asimismo con esta misma aplicación con la opción *Analyze Timing*, se puede ver el ruteo de las señales con su correspondiente análisis temporal tras su implementación. En la Fig. 5.4 se observa el reporte completo de temporización teniendo en cuenta las restricciones temporales definidas en el proyecto, es decir, la señal interna de la FPGA de 50MHz con un ciclo de trabajo del 50 % y la señal de 8MHz generada mediante un DCM. Sólo se muestran, en este caso, el análisis del tiempo de Setup de las dos señales de reloj anteriores para los caminos de

mayor retardo. Las líneas en blanco sobre el diseño representan uno de esos caminos. Aún para estos casos, como se puede ver, todas las condiciones temporales para la correcta implementación del código fueron satisfechas.

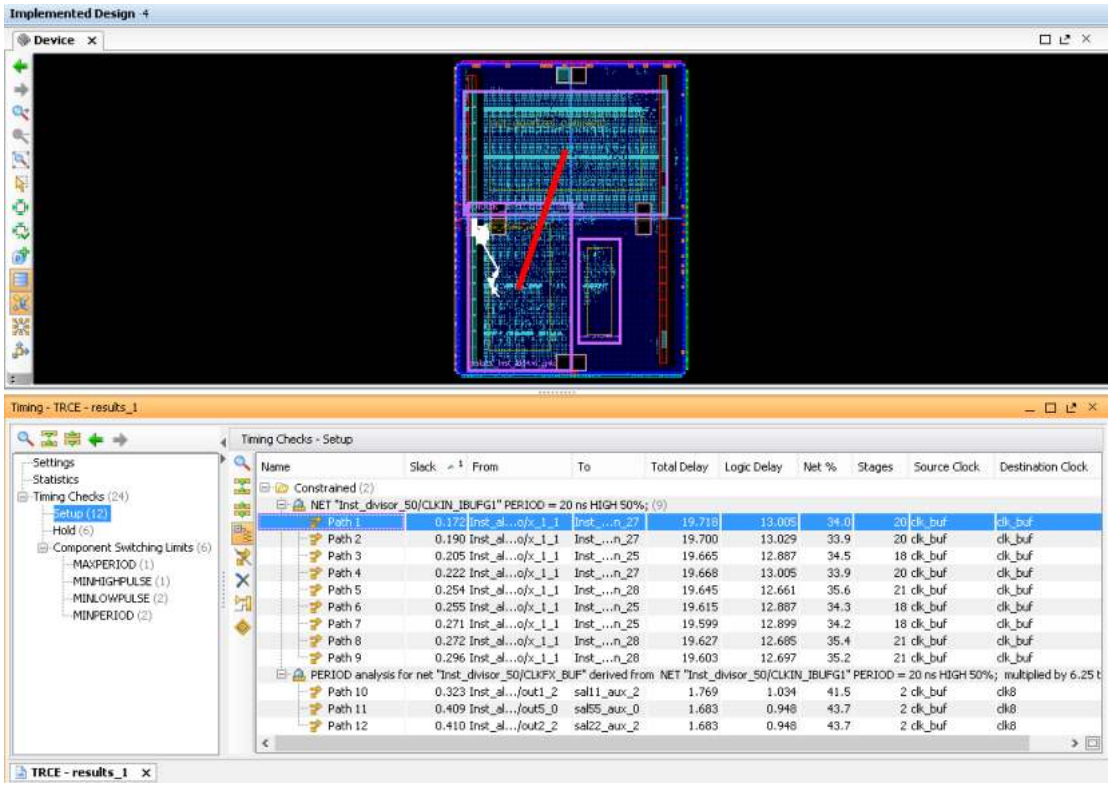


Figura 5.4: Reporte del análisis temporal del PlanAhead

A continuación, en la Fig. 5.5 se muestra el informe de síntesis que se obtuvo del ISE de Xilinx para la FPGA empleada.

main Project Status (07/16/2020 - 16:11:06)			
Project File:	pblocks.xise	**Parser Errors:**	No Errors
Module Name:	main	**Implementation State:**	Placed and Routed
Target Device:	xc3s1600e-4fg320	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	558 Warnings (304 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	12,336	29,504	41%	
Number of 4 input LUTs	15,234	29,504	51%	
Number of occupied Slices	8,947	14,752	60%	
Number of Slices containing only related logic	8,947	8,947	100%	
Number of Slices containing unrelated logic	0	8,947	0%	
Total Number of 4 input LUTs	15,428	29,504	52%	
Number used as logic	14,848			
Number used as a route-thru	194			
Number used for 32x1 RAMs	384			
Number used as Shift registers	2			
Number of bonded IOBs	34	250	13%	
Number of BUFGMUXs	5	24	20%	
Number of DCMs	1	8	12%	
Number of MULT18X18SIOs	26	36	72%	
Average Fanout of Non-Clock Nets	3.53			

Figura 5.5: Informe de síntesis de la FPGA

Como ya se ha indicado antes, una FPGA es un arreglo matricial de CLBs (Bloques Lógicos Configurables). Cada uno de ellos está compuesto por cuatro *Slices* que permiten implementar funciones lógicas, aritméticas y ROM. A su vez, estos contienen otros componentes como Flip Flops, LUTs, multiplexores, RAM, registros de desplzamientos y otros, de acuerdo al tipo de *Slice*. Teniendo

en cuenta esto último, se analiza a continuación cada item de la Fig. 5.5:

- Number of Slice Flip Flops: Indica la cantidad total de Flip Flops utilizados (existen dos por cada *Slice*).
- Number of 4 input LUTs: Cantidad de LUTs. Un LUT (Look Up Table, del inglés) es una pequeña memoria que se desempeña como generador de funciones. Para la FPGA utilizada, estos son de 4 entradas, por lo que uno de ellos puede generar una tabla de verdad de 16x1.
- Number of occupied Slices: Número de *Slices* utilizadas. Como puede verse, se hace la distinción entre dos tipos de *Slice*, de lógica relacionada y no relacionada. El primero refiere a aquellos que comparten conectividad y tienen prioridad en la síntesis del código ya que conducen a un mejor rendimiento de temporización. En este caso, no fue utilizado ninguno del tipo no relacionado por lo que en este sentido se encuentra optimizado. Es decir, el número de slices ocupados es el 60 % del total, y dentro de este 60 %, el 100 % usa lógica relacionada.
- Total Number of 4 input LUTs: Número total de LUTs teniendo en cuenta también aquellos que se utilizan en su modo *route thru*, es decir, aquellos LUTs utilizados para acceder a un punto interno de un *Slice*. Por otro lado, las memorias RAM 32x1 son memorias de 5 líneas de direccionamiento y una salida que se implementan a partir de dos LUTs.
- Number of bonded IOBs: Número de entradas y salidas de la FPGA
- Number of BUFGMUXs: Cantidad de multiplexores.
- Number of DCMs: Cantidad de DCMs. En este caso, se utiliza sólo uno para obtener la frecuencia de 8MHz a partir de la de 50MHz de la FPGA.

- Number of MULT18X18SIOs: Cantidad de multiplicadores sincrónicos, cada uno con dos entradas de 18 bits con una salida de 36 bits.
- Average Fanout of Non-Clock Nets: Número promedio de entradas digitales que una compuerta lógica puede alimentar en su salida (Fan out). Las señales de reloj y habilitación de reloj utilizan rutas dedicadas.

Como puede apreciarse, el porcentaje de utilización de los recursos de la FPGA en todos los casos es menor al 100 %. Además, en la sección Timing Constraints (Restricciones temporales) se puede verificar que todas las restricciones son cumplidas. Respecto a las advertencias del informe, corresponden a señales que no fueron utilizadas en el desarrollo del proyecto y otras que han sido optimizadas por el sintetizador, sin que esto afecte su funcionamiento. Con todo esto, sólo resta evaluar el funcionamiento de cada bloque lógico del código implementado.

5.4.2. Funcionamiento

A fin de poder evaluar el funcionamiento de la implementación digital en cada una de las etapas que lo conforman, se realizó una simulación del hardware sintetizado en la FPGA, utilizando mediciones de los ripples de corriente adquiridas en ensayos previos. Para llevar a cabo esta simulación, se utilizó el bloque HDL Cosimulation de MatLab Simulink 2012b, en conjunto con el simulador de FPGA ModelSim SE6.5, donde se cargaron todos los códigos VHDL del proyecto y se configuraron las entradas y salidas de la FPGA para su visualización.

En la Fig. 5.6 se puede observar la simulación del sistema completo. Como se mencionó anteriormente, para realizar la simulación en un escenario realista, se experimentaron con muestras de los ripples de corriente adquiridas en ensayos previos. Dichas señales, adquiridas con una sonda de medición de corriente, deben ser acondicionadas, tal como se explicó en el Cap. 3. Todo este proceso se puede

ver en mayor detalle en la Fig. 5.7 para una de las fases de entrada del convertidor. Asimismo, en la Fig. 5.8 se muestra cómo fue simulado el ADC con las señales de muestreo (SCK) y conversión (CNV), el valor de la pendiente de su función transferencia (Fig.3.5b) y los bloques "Extract Bit *i*"para la serialización de los datos en palabras de 16 bits, tal como funciona el ADC LTC2323 utilizado.

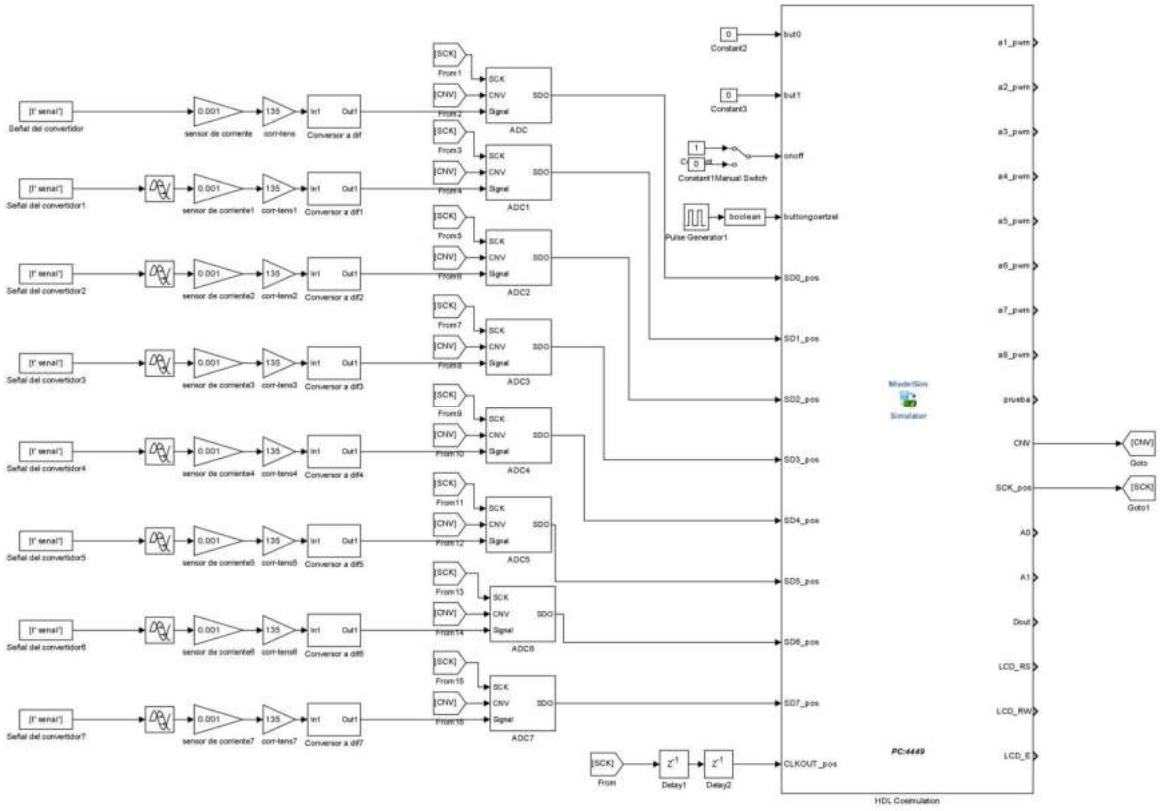


Figura 5.6: Diagrama en bloques del proyecto completo en Simulink

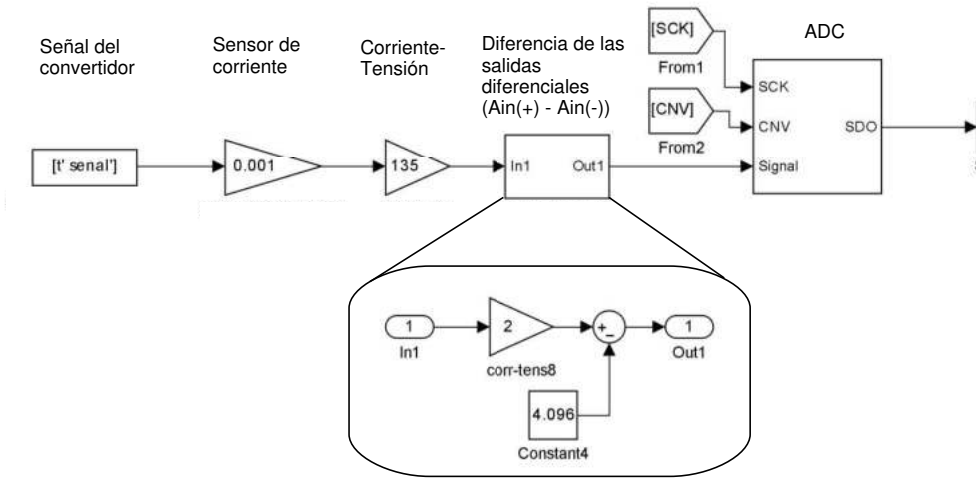


Figura 5.7: Detalle de la simulación del circuito de adquisición

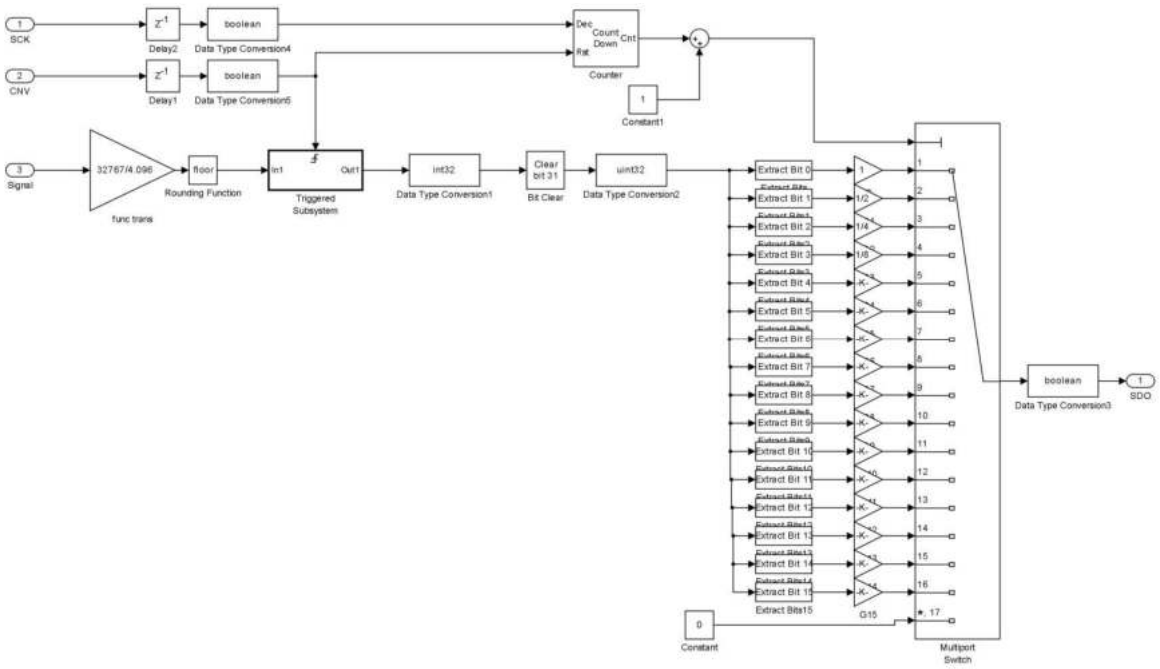


Figura 5.8: Detalle de la simulación del ADC

A continuación se evalúa cada uno de los bloques principales del proyecto, es decir, medición de amplitud de los riples de corriente, la caracterización del

ripple de la corriente total y el método de ordenamiento. Todo este análisis se lleva a cabo comparando los resultados obtenidos a partir del modelo simulado de la Fig. 5.7 y los reales.

Medición de amplitud

En la Fig. 5.9 se visualizan las muestras de cada una de las fases del convertidor utilizadas en la simulación. Estas son señales de 25KHz con un ciclo de trabajo del 18 %.

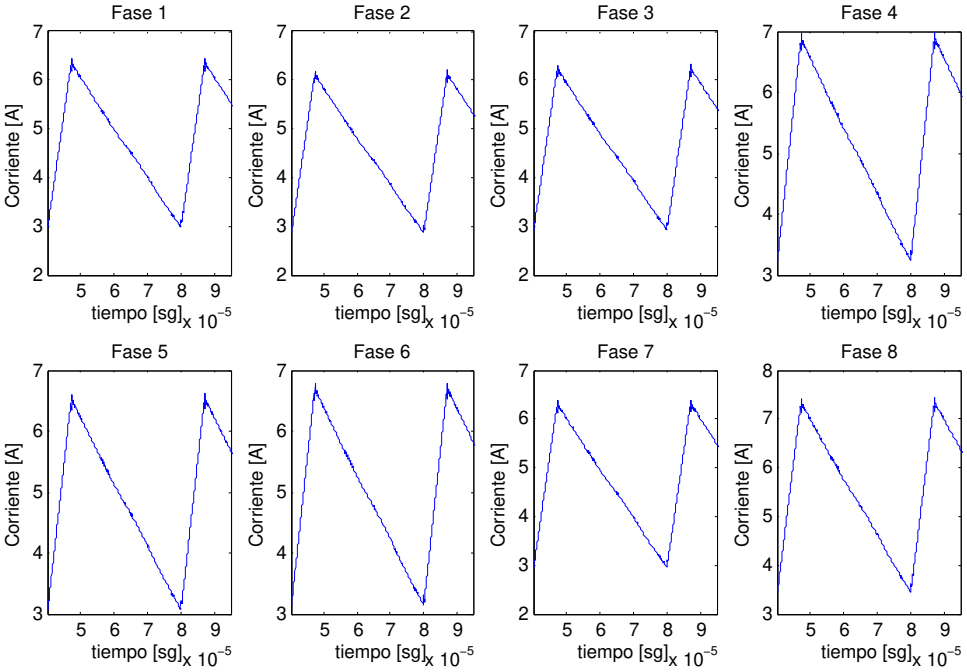


Figura 5.9: Fases del convertidor utilizadas como entradas en la simulación

Para poder calcular la amplitudes de cada uno de los ripples de forma analítica se trazan tres rectas por interpolación lineal tal como se muestra en la Fig. 5.10 y se calcula la amplitud como la diferencia entre los dos puntos de intersección.

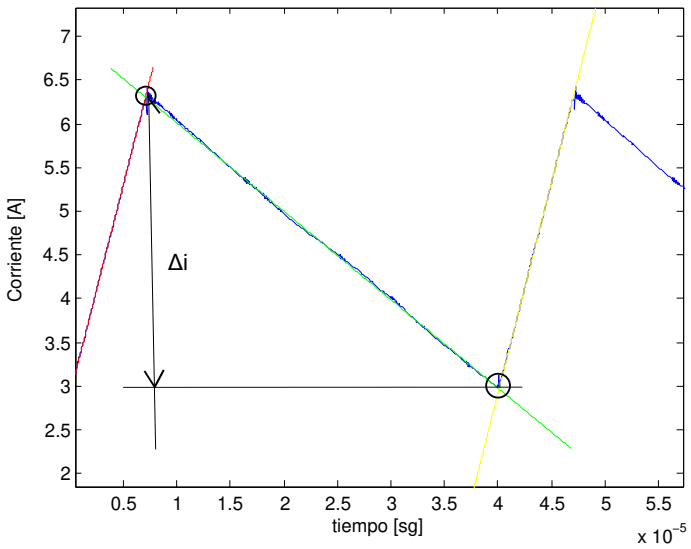


Figura 5.10: Método analítico para el cálculo de la amplitud del ripple

En la Tabla 5.3 se muestran las relaciones de cada una de las amplitudes de las señales ensayadas respecto a la primera aplicando el método descripto anteriormente y las obtenidas mediante simulación. Asimismo, se puede ver en dicha tabla que los errores porcentuales cometidos de los valores calculados por la simulación se encuentran dentro del orden planteado en la etapa de diseño.

Fase	$\Delta i_i/\Delta i_1$ (Teórico)	$\Delta i_i/\Delta i_1$* 2048	Resultado de la simulación	Error porcentual
2	0.962	1970.176	1971	0.04 %
3	0.981	2009,088	2009	0.04 %
4	1.084	2220,032	2218	0.09 %
5	1.028	2105,344	2104	0.06 %
6	1.054	2158,592	2158	0.03 %
7	0.993	2033,664	2035	0.07 %
8	1.155	2365,440	2364	0.06 %

Tabla 5.3: Relación de amplitudes respecto a la fase 1 de las señales ensayadas

Caracterización de la corriente total en estado estacionario

A partir de las mediciones de las amplitudes calculadas anteriormente por la transformada de Goertzel, el algoritmo implementado obtiene la secuencia de los picos positivos y negativos del ripple total (Δi_T), es decir, P_x^+ y P_x^-, para un determinado ordenamiento de las fases. Para verificar que dichos valores se correspondan al Δi_T real se normalizan a la unidad esta última señal y la obtenida de la implementación con el ordenamiento de sus fases inicial y se las grafica, como se observa en la Fig. 5.11. Es importante destacar que la normalización se hace a efectos comparativos ya que cada una tiene un factor de escalamiento distinto y, en el caso de la señal real, la amplitud pico a pico debe medirse mediante rectas de interpolación, como se hizo anteriormente. Se puede ver que las dos señales coinciden con leves diferencia debido al ruido y efectos no deseados en los picos de la señal real por la conmutación de las llaves. Por ello, se concluye que la caracterización del ripple total ha sido correcta.

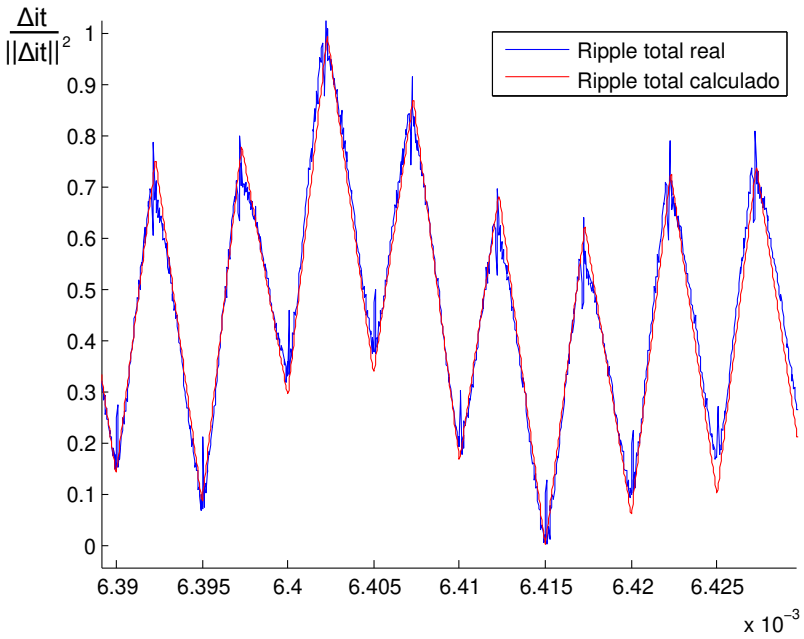
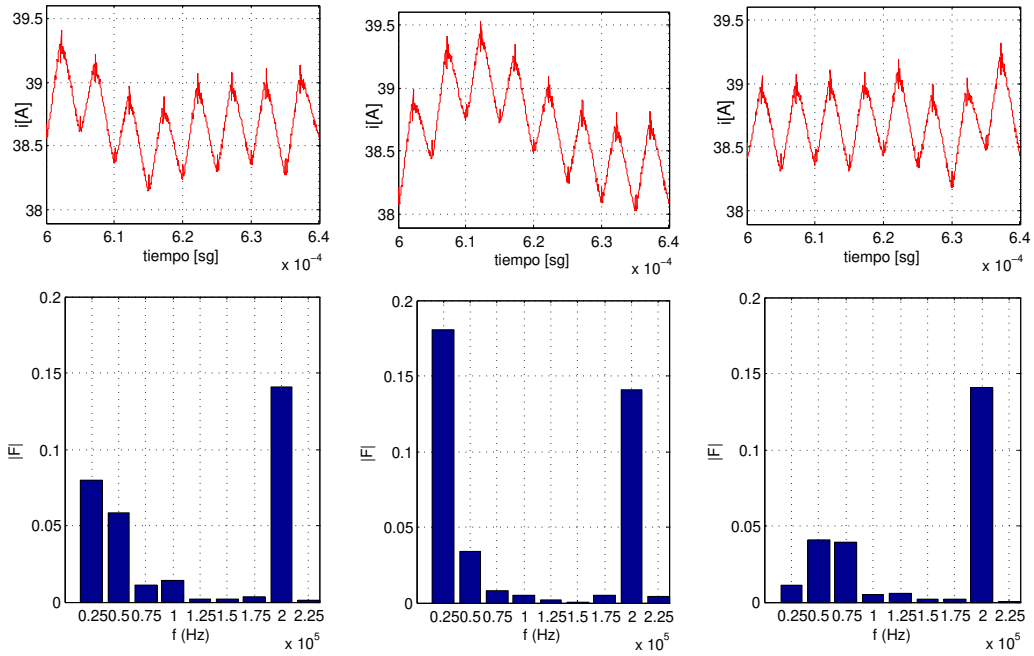


Figura 5.11: Ripple total real y estimado a partir de la implementación

Método de ordenamiento de los ripples de fase

Para poder evaluar que el ordenamiento al que se arriba mediante la implementación del algoritmo genético es válido, se compara en la Fig. 5.12 el ripple total con el ordenamiento de las fases hallado, el inicial y el peor caso. Este último se puede calcular de la misma forma en que se halló el óptimo pero seleccionando a los individuos con mayor valor de su función costo (amplitud pico a pico). Si bien se muestra en esta figura la corriente del ripple total para cada caso, con el objetivo de poder apreciar más el efecto de minimizar la amplitud pico a pico del ripple se grafica parte del espectro de dichas señales. Como ya se explicó anteriormente, minimizar la amplitud del ripple total equivale a minimizar su contenido armónico para las componentes entre f_{sw} y $(N-1).f_{sw}$, es decir, para este caso entre 25KHz y 175KHz. Se observa que el ordenamiento óptimo hallado de la Fig. 5.12c mejora de forma notoria respecto a los otros casos, en especial con el peor de ellos, la amplitud de la componente fundamental y la segunda armónica de 50KHz, sin empeorar significativamente las demás componentes. Estas dos componentes frecuenciales son las más difíciles de filtrar por ser las de menor frecuencia. Se demuestra así que el resultado del ordenamiento encontrado es válido y, por lo tanto, el método de ordenamiento de los ripples de fase funciona correctamente, así como el código implementado en su totalidad.



(a) Ordenamiento inicial (b) Peor ordenamiento (c) Ordenamiento óptimo

Figura 5.12: Ripple total para distintos ordenamientos de los ripples de fase

Capítulo 6

Conclusiones

En este trabajo se ha abordado la implementación de un método para hallar el ordenamiento óptimo de las fases de un convertidor multifásico, que minimiza la amplitud del ripple total de corriente de salida.

En primer lugar, se diseñó una placa de adquisición de dimensiones acordes al gabinete y compatible con la etapa de potencia y sensores del prototipo existente, para el posterior procesamiento digital de los 8 ripples de corriente del convertidor. Ésta cuenta con la precisión adecuada para distinguir la amplitud entre las fases según la tolerancia especificada para los inductores y la frecuencia de muestreo con margen suficiente para permitir un procesamiento de la señal y contemplar que las señales adquiridas puedan ser utilizadas con propósitos de control y monitoreo. Asimismo, se diseñó una etapa de acondicionamiento de tal forma que permitiese utilizar todo el rango dinámico del ADC, y con los elementos de protección contra sobretensión (diodos Schottky y ESD) necesarios para evitar el daño de los circuitos integrados de mayor sensibilidad, como los amplificadores de instrumentación y operacionales, y los ADC. Cada uno de estos componentes empleados en el diseño de la placa fueron seleccionados teniendo en cuenta sus

prestaciones para la función a desempeñar. Por ejemplo, el amplificador de instrumentación fue elegido debido a su bajo ruido y distorsión en la salida y su buen rechazo a tensiones de entrada de modo común. Estas características lo hacen apropiado para trabajar con conversores analógicos-digitales de altas frecuencias de muestreo.

Para el posterior procesamiento de las señales, se seleccionó una plataforma digital en función de los requisitos de la aplicación, tales como la modularidad y la velocidad de procesamiento. La plataforma elegida fue una FPGA ya que reúne estas características y procesa el código escrito en paralelo, facilitando así la operación modular requerida en el proyecto. A su vez, se contempló la cantidad de bloques lógicos configurables y componentes necesarios para la implementación de cada bloque del proyecto en la FPGA, y la cantidad suficiente de pines de entrada y salida para conectar las señales de adquisición y control.

Una vez seleccionada la plataforma, se identificaron los distintos procesos que se llevan a cabo en el algoritmo genético para la obtención del ordenamiento óptimo de las fases del convertidor. Cada uno de estos procesos fue escrito en lenguaje VHDL, a partir de máquinas de estados y circuitos lógicos combinacionales. Para ello, se tuvo en cuenta el uso de estructuras lógicas apropiadas para la optimización de los recursos y consideraciones de sincronización entre cada una de las señales participantes a partir de los flancos de activación de los procesos, tal como se vió en el bloque SPI . Para optimizar también el espacio utilizado en la FPGA se emplearon Pblocks, herramienta que permite seleccionar el espacio ocupado de cada bloque lógico.

Finalmente, se logró corroborar el correcto funcionamiento de las placas de adquisición mediante distintos valores de tensión continua a la entrada de las mismas dentro del rango de operación de los ADC. Aunque no se pudo ensayar el sistema completo, también se pudo verificar el correcto funcionamiento del

código implementado en la plataforma digital a través de simulaciones realizadas en Matlab, Simulink y ModelSim.

A lo largo del proyecto han sido adquiridas y desarrolladas distintas habilidades conceptuales y prácticas propias de la ingeniería electrónica. Entre ellas, consideraciones prácticas para la elección de los componentes de un circuito electrónico, lectura y selección de la información relevante de la hoja de datos de cada componente, utilización del software Altium Designer para el diseño de un PCB y buenas prácticas para la ubicación y ruteo entre los distintos componentes, montaje de componentes SMD y diagnóstico de fallas en un circuito físico. Además, fueron desarrolladas habilidades en el campo del procesamiento digital de las señales, como criterios de selección de una plataforma digital, utilización de una FPGA y buenas prácticas de escritura en código VHDL para la optimización de los recursos implementados por dicha plataforma. Ejemplos de esto último son el manejo de rutas dedicadas para las señales de reloj y el uso de los distintos flancos de una señal para la activación de los procesos y evitar problemas de sincronización, también la inicialización de las señales y el uso de Pblocks para optimizar el área de implementación, entre otros.

Bibliografía

[1] Pablo Antoszczuk, Rogelio Garcia Retegui, Marcos Funes, Sebastian Maestri, y Nicolas Wassinger. Método de ordenamiento de secuencia de disparo de llaves para convertidores interleaved. In *XXIV Congreso Argentino de Control Automático, AADECA 2014. ISBN: 978-950-99994-8-0*, 2014.

[2] P. Antoszczuk, P. Cervellini, R. G. Retegui, y M. Funes. Optimized switching sequence for multiphase power converters under inductance mismatch. *IEEE Transactions on Power Electronics*, 32(3):1697–1702, March 2017. ISSN 1941-0107. doi: 10.1109/TPEL.2016.2602810.

[3] Liqin Ni, D.J. Patterson, y J.L. Hudgins. High power current sensorless bidirectional 16-phase interleaved DC-DC converter for hybrid vehicle application. *IEEE Transactions on Power Electronics*, 27(3):1141–1151, March 2012. ISSN 0885-8993. doi: 10.1109/TPEL.2011.2165297.

[4] M. Gerber, J.A. Ferreira, I.W. Hofsajer, y N. Seliger. Interleaving optimization in synchronous rectified DC/DC converters. In *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, volume 6, pages 4655–4661 Vol.6, June 2004. doi: 10.1109/PESC.2004.1354823.

[5] O. Garcia, P. Zumel, A. De Castro, y J.A. Cobos. Automotive DC-DC bidirectional converter made with many interleaved buck stages. *IEEE Transactions on Power Electronics*, 21(3):578–586, May 2006. ISSN 0885-8993. doi: 10.1109/TPEL.2006.872379.

[6] Pablo Antoszczuk, Nicolas Wassinger, Marcos Funes, Rogelio Garcia Retegui, y Sebastian Maestri. Control de corriente para convertidores interleaved con cancelación de armónicos basada en el ajuste de las fases de los ripples. In *XXIV Congreso Argentino de Control Automático, AADECA 2014. ISBN: 978-950-99994-8-0*, 2014.

[7] P. Cervellini, P. Antoszczuk, R. G. Retegui, y M. Funes. Current ripple amplitude measurement in multiphase power converters. *IEEE Transactions on Power Electronics*, 32(9):6684–6688, September 2017. ISSN 1941-0107. doi: 10.1109/TPEL.2017.2686784.

[8] P.D. Antoszczuk, R.G. Retegui, N. Wassinger, S. Maestri, M. Funes, y M. Benedetti. Characterization of steady-state current ripple in interleaved power converters under inductance mismatches. *IEEE Transactions on Power Electronics*, 29(4):1840–1849, 2014. ISSN 0885-8993. doi: 10.1109/TPEL.2013.2270005.

[9] David E Goldberg. Genetic algorithms in search, optimization, and machine learning. *ISBN: 0-201-15767-5*, 1989.

[10] S. Sharma y K. Gupta. Solving the traveling salesmen problem through genetic algorithm with new variation order crossover. In *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, pages 274–276, April 2011. doi: 10.1109/ETNCC.2011.6255903.

[11] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, y Y. Yurramendi. Learning bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 26(4):487–493, Jul 1996. ISSN 1083-4427. doi: 10.1109/3468.508827.

[12] Lawrence Davis. Applying adaptive algorithms to epistatic domains. In *IJCAI*, volume 85, pages 162–164, 1985.

[13] Kusum Deep y Hadush Mebrahtu Adane. New variations of order crossover for travelling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(1):2–13, 2010.

[14] Cristian Sisterna. Generador de secuencia binaria pseudo aleatoria. *C7 Tecnología*, 2012.