Universidad Zaragoza
1542

Trabajo Fin de Grado

Control digital con FPGA de un convertidor de potencia *Buck* síncrono

Digital control with an FPGA of a synchronous *Buck* power converter

Autor

Pablo Escario Paraíso

Director

José Ignacio Artigas Maestre

Escuela de Ingeniería y Arquitectura

2019

Agradecimientos

A mi familia.

Resumen

En el presente trabajo fin de grado se busca desarrollar un diseño el cual pueda ser utilizado los próximos años en las prácticas impartidas en la rama de la Electrónica Digital y de Potencia. Se va a realizar un reductor síncrono de potencia, *Buck*, siendo este controlado mediante una FPGA, el proyecto aborda desde el diseño de la placa de circuito impreso, hasta todo el diseño del control digital con el fin de ser capaces de controlar la tensión de salida de este convertidor.

Abstract

The aim of this final project is to develop a design which can be used in the next few years in the laboratory exercises of subjects related to Digital and Power Electronics. A *Buck* synchronous power converter will be developed controlled by an FPGA. The project involves from the design of the printed circuit board, to the entire design of the digital control in order to be able to control the output voltage of this converter.

Contenido

1. Introducción

En este apartado introductorio se comentan las motivaciones que han llevado a la realización de este Trabajo de Fin de Grado (TFG) así como el contexto en el que se realiza y los conocimientos previos desde de los que se parte. Todo ello con la intención de destacar las competencias adquiridas a lo largo de los años de estudio de grado, relacionándolo con las distintas asignaturas que de manera especial tienen que ver con la temática de este TFG en particular.

Se detallan los objetivos que pretenden lograrse a la finalización de este TFG, así como el alcance del trabajo. Se presentan de forma breve y ordenada los contenidos que se desarrollarán a lo largo de esta memoria técnica, así como las metodologías y herramientas usadas en la misma.

1.1 Antecedentes

El gran desarrollo tecnológico alcanzado hoy en día en el amplio campo de la electrónica, tanto industrial como doméstica, sería del todo inconcebible sin la capacidad de transformar eficientemente niveles de tensión para adaptarlos a las muy variadas necesidades finales. Es por ello por lo que el uso de etapas convertidoras de electrónica de potencia con este fin esté hoy por hoy muy extendido, encontrándose ejemplos de su utilización como fuentes de alimentación conmutadas (en equipos industriales, automóviles, electrodomésticos...), accionamientos de motores de corriente continua (CC), industria aeroespacial, etc. [1]

La idea inicial que motiva la realización de este trabajo es el aprovechamiento del resultado del proyecto para su posible uso en las prácticas que los futuros estudiantes del grado puedan realizar. Especialmente de aquellas asignaturas relacionadas con la materia como Electrónica Digital, impartida en el segundo curso del grado, o Diseño Digital y control con FPGA, optativa del último curso. Por ello se ha elegido como elemento de control una placa Basys 3 de Digilent que incluye una FPGA Artix-7 de Xilinx, [2], placa con la cual el estudiante pueda estar familiarizado ya que se utiliza en las asignaturas de la rama de Electrónica Digital. Se parte de un esquema previo realizado por el alumno Adrián Marco Artigas en el ámbito de su trabajo fin de master [10], aunque no llegó a ser probado.

Los convertidores CC/CC son etapas de electrónica de potencia capaces de transformar una tensión continua a la entrada en otra distinta a la salida con alta eficiencia. La tensión de salida de forma habitual será regulada para independizarla de posibles variaciones de la carga o de la tensión de entrada. Estas etapas, basadas en la conmutación de transistores y diodos, pueden conformar diferentes topologías: reductoras (tipo Buck), elevadoras (tipo Boost) o reductoras-elevadoras (tipo Buck-Boost, flyback, Cuk…). [3]

En el caso concreto de este trabajo trataremos sobre el análisis, modelado y control de un reductor *Buck* síncrono. Estos sistemas son no lineales y variantes en el tiempo, por lo que las técnicas clásicas de control lineal vistas en asignaturas como Señales y Sistemas, Sistemas Automáticos o Ingeniería de control no son de inmediata aplicación. Para resolverlo, se desarrollarán modelos promediados sobre el periodo de conmutación (*Tsw*) en pequeña señal, sobre los dos circuitos equivalentes que se dan en el sistema en función del estado (conducción o corte) del transistor. Debido a tratarse de un Buck síncrono, se

considerará que trabaja en Modo de Conducción Continuo (MCC), es decir, la intensidad que circula por la bobina del convertidor será no nula en cualquier instante de tiempo.

Puesto que en la placa de circuito impreso somos capaces de medir tanto la tensión de salida del convertidor, como la corriente que circula por la bobina, caben destacar dos posibles modos para realizar el control, siendo estos el de tensión y el de corriente. Para el de tensión, la acción de entrada al sistema será la relación de servicio o *duty* ratio del cual depende de forma ideal la relación entre las tensiones de entrada y de salida. El algoritmo de control se implementará en coma fija y en lenguaje VHDL sobre la FPGA mencionada anteriormente. En el caso del control de la corriente, este se realizaría para mejorar la dinámica del sistema.

La construcción del *Buck* se realizará de forma que su conexión con la placa de FPGA se realice de forma sencilla mediante un conector, de forma que el conjunto final sea robusto y compacto.

En la Figura 1, se puede observar cual es el esquema, sin entrar mucho en detalle, del diseño que se va a realizar en el presente trabajo fin de grado.

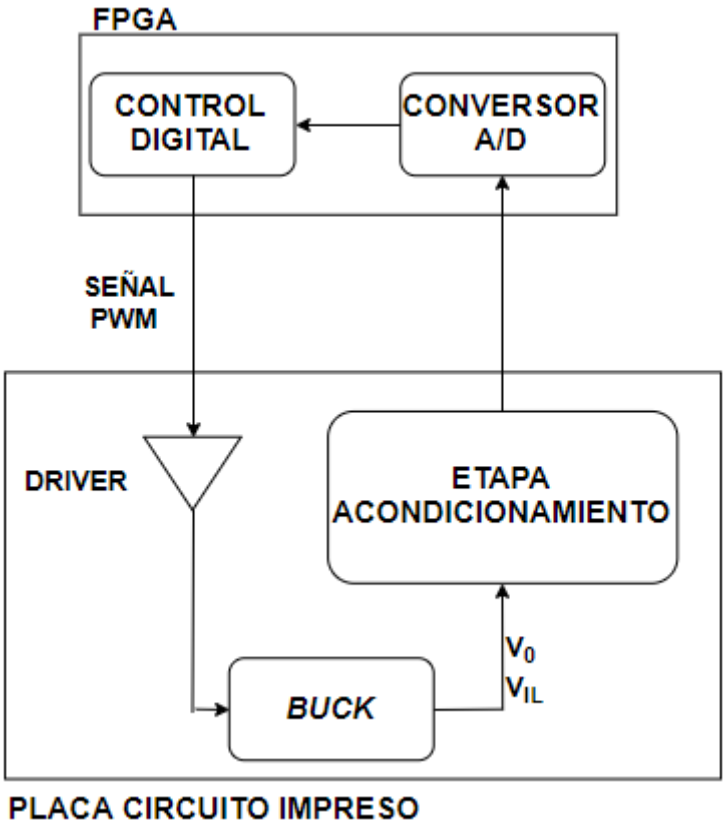


Figura 1

1.2 Objetivos y alcance

Los objetivos que se plantean conseguir a la finalización de este TFG son:

- Obtener un modelo matemático adecuado del convertidor reductor *Buck* sobre el que se trabajará.
- Diseñar un controlador que asegure un error de posición cero a la salida del sistema y que permita obtener una tensión regulada y constante ante cambios en la carga o la tensión de entrada.
- Realizar una correcta implementación del controlador digital en lenguaje VHDL y en coma fija sobre la FPGA.
- Configurar los conversores analógicos/digitales internos de la FPGA con el fin de no utilizar módulos adicionales, ya que en la placa utilizada en prácticas requiere de conversores externos.
- Diseñar y montar la placa de circuito impreso con el Buck para su fácil conexión con la FPGA.
- Verificar el correcto funcionamiento del sistema mediante pruebas reales en el laboratorio.

Para alcanzar dichos objetivos se realizan las siguientes tareas durante el desarrollo del presente trabajo:

- Se obtendrá el modelo dinámico del reductor mediante el promediado de los distintos estados del sistema a lo largo del periodo de conmutación *Tsw*. Posteriormente, se usará esa planta para el cálculo del regulador mediante la aplicación Sisotool de Matlab. Se realizará la discretización del regulador a través de la función c2d de Matlab mediante transformación bilineal.
- Se adaptarán los coeficientes al formato numérico en coma fija más adecuado para su implementación en lenguaje VHDL.
- Se realizará el estudio y posterior configuración de los conversores analógico/digitales de la FPGA. Posteriormente se procederá a la simulación y prueba experimental del correcto funcionamiento de estos.
- Se realizará la descripción en VHDL a través del entorno de desarrollo Vivado de Xilinx para implementar el regulador y todas las funciones auxiliares en la FPGA.
- Se realizará la simulación en la herramienta Orcad Pspice de los circuitos de acondicionamiento de la señal que llega a los conversores analógico/digitales para comprobar su correcto diseño.
- Se diseñará mediante la aplicación Eagle la placa de circuito impreso sobre la que se montará definitivamente el reductor *Buck*.
- Se montará un prototipo y se analizará y comprobará el correcto funcionamiento del sistema mediante las pertinentes pruebas en el banco del laboratorio.

1.3 Especificaciones del diseño a desarrollar.

El sistema a desarrollar consta de dos partes bien diferenciadas, por un lado, la FPGA y por otro lado la placa de circuito impreso que contendrá el convertidor *Buck*.

Esta FPGA ha sido elegida por su disponibilidad e idoneidad, además esta había sido utilizada en anteriores asignaturas, como las comentadas anteriormente. Otra razón por lo que esta resulta idónea es debido a que en la palca sobre la que esta va montada, ofrece unas prestaciones excepcionales, ya esta cuenta con conversores analógico/digitales internos, displays para poder visualizar las diferentes tensiones de interés, interruptores varios los cuales nos serán de utilidad. Además, presenta una forma robusta para poder hacer la conexión con la placa de circuito impreso, lo cual resulta idóneo para no perder información.

En cuanto a las especificaciones del reductor se han buscado características similares a las del reductor usado actualmente en las prácticas de Diseño Digital y control con FPGA.

Se ha elegido tensión de entrada 12Vdc y como tensión de salida 5Vdc, lo cual genera una relación de servicio nominal de 0.4166.

El resto de los valores fundamentales para el diseño del Buck se muestran en la siguiente Figura 2.

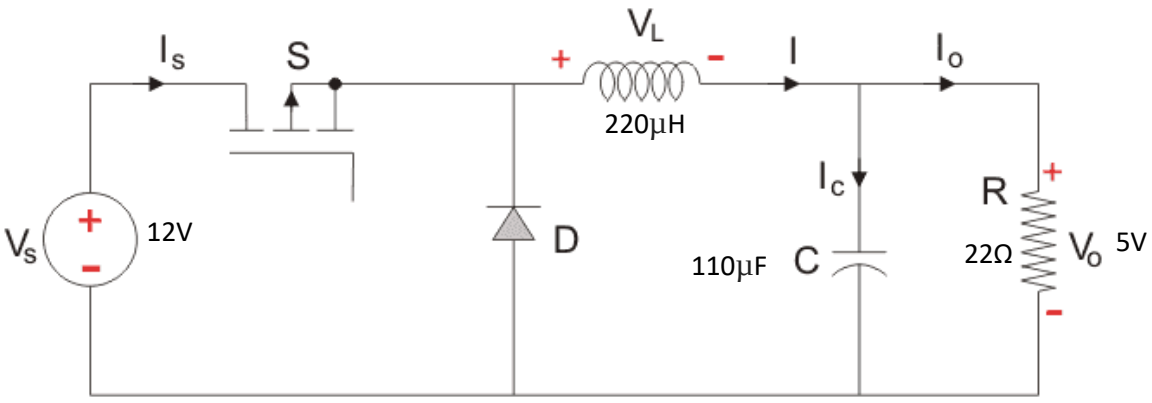


Figura 2

2. Convertidor Buck síncrono

2.1 Introducción

En el presente apartado se obtiene un modelo dinámico del comportamiento del reductor *Buck*. En primer lugar, se realiza una breve introducción general sobre los principios básicos y parámetros más importantes en el funcionamiento de este tipo de reductores. Se trabajará con el modelo en pequeña señal del *Buck* síncrono, la obtención de este modelo se encuentra explícitamente explicada en Anexo I. Obtención de la función de transferencia en pequeña señal del *Buck*.

2.2 El reductor Buck ideal

Este convertidor de CC/CC es capaz de transformar niveles de voltaje en otros usando elementos como bobinas y capacitores, almacenando temporalmente energía en ellos y descargándola de tal forma que los niveles de voltaje final son los buscados. Para la explicación este apartado nos hemos basado en textos de referencia como [4].

La forma en cómo se convierte el voltaje es forzando a que se almacene la suficiente energía en la bobina o capacitor y después, en otro tiempo se cambie la polaridad o la disposición de dicho elemento para descargar esa misma energía acumulada en la salida.

El principio de funcionamiento de un conversor reductor tipo *Buck* se basa en la aplicación de una tensión continua a una carga durante un tiempo, tiempo en *ON*, dentro de un periodo de conmutación *SW*.

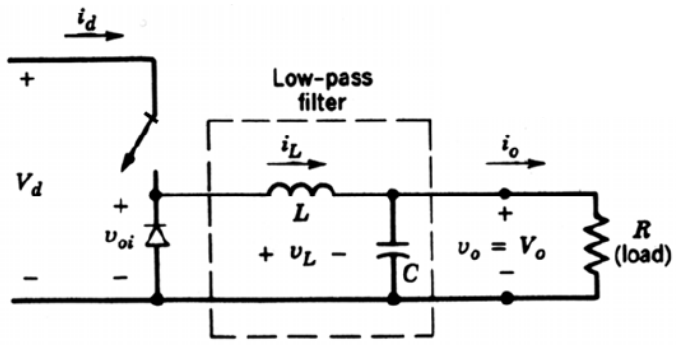


Figura 3

En un reductor *Buck* ideal, como el mostrado en la Figura 3, extraída de [4], la tensión de salida queda directamente relacionada con la tensión de entrada a través de la relación de servicio o *duty ratio*, definida como:

$$D = \frac{t_{ON}}{T_S}$$

(Ec. 2.2.1)

Donde t_{ON} es el tiempo de conducción (*ON*) del dispositivo de potencia principal y T_S es el periodo de conmutación.

$$V_0 = \frac{1}{T_S}\int_0^{T_S} v_0 dt = \frac{V_d t_{ON}}{T_S} = DV_d$$

(Ec. 2.2.2)

Para el análisis del convertidor *Buck* asumiremos que se encuentra en Modo de Conducción Continua (MCC), es decir que la corriente circulando por la bobina $I_L(t)$ es no nula en cualquier instante de tiempo.

Esto a su vez genera dos topologías distintas en función del estado de *SW* durante t_{ON} (conducción) y t_{OFF} (corte), como se muestra en la Figura 4.

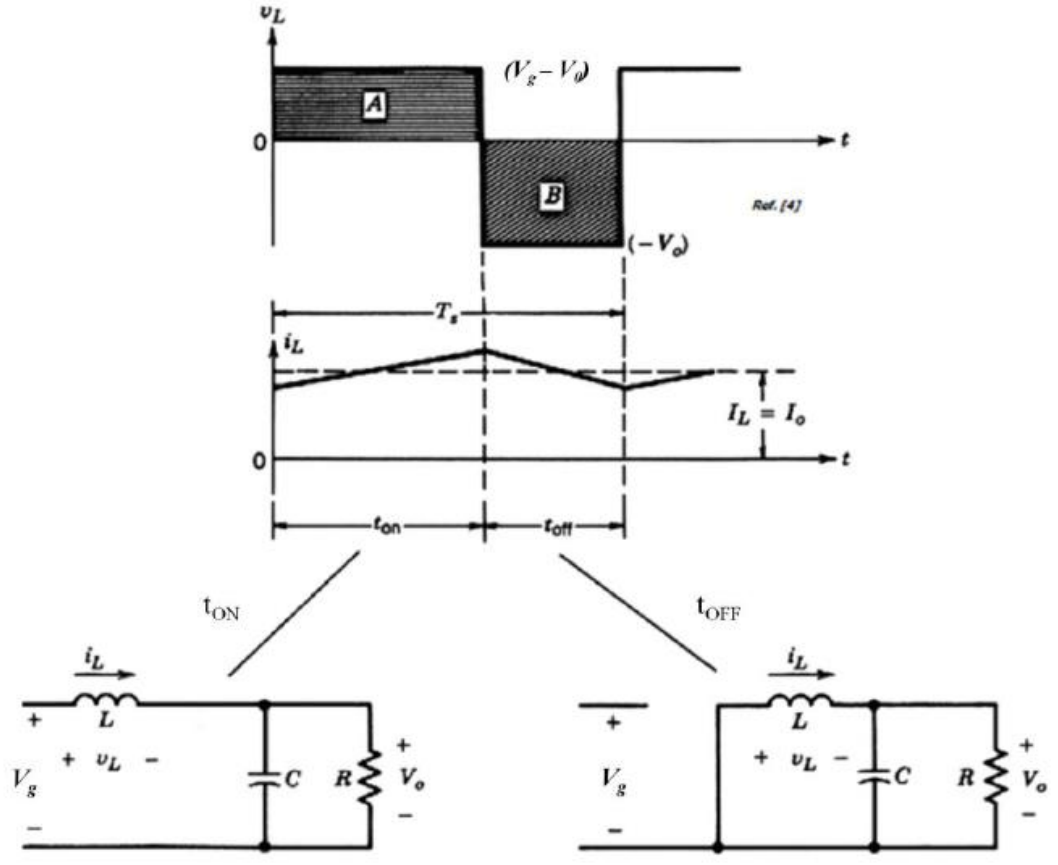


Figura 4

El circuito que podemos observar en la parte inferior izquierda de la Figura anterior, extraída de [4], se corresponde con el estado de *ON*: es este estado, la corriente va desde la fuente de entrada hasta el capacitor, cargando a su paso la bobina. El diodo, como se puede apreciar, no conduce, ya que en ese momento esta polarizado inversamente. La función del circuito es cargar la bobina, nuestro principal elemento de almacenamiento de energía, además de alimentar el circuito de carga con el voltaje suficiente por medio del capacitor.

Cuando el transistor se pone en un estado de corte, es decir, no hay conducción, la fuente principal de energía no alimenta el circuito, este circuito se corresponde con el mostrado en la parte inferior derecha de la Figura anterior, correspondiente al estado *OFF*. En ese momento se aprovecha la energía del inductor, almacenada en forma de campo magnético, para hacer circular una corriente por el circuito. Esta corriente sigue alimentando al capacitor y mantiene el nivel de voltaje a la salida.

Entre estos dos estados, de conducción y no conducción es como se transforma el voltaje de entrada al de salida. Al conmutar entre estados, a una frecuencia fija, la

conversión dependerá de cuánto dura cada estado con respecto a la frecuencia. Por convención usaremos el estado de conducción como base, será nuestro ciclo de trabajo.

La relación entre la tensión de entrada y la de salida viene determinada por el *duty* (D), como se puede observar en la ecuación 2.2.1, a continuación, vamos a justificar dicha relación a través de la tensión presenta la bobina a lo largo del periodo de conmutación.

Puesto que la tensión media en una bobina es nula:

$$V_L = L\frac{dI_L}{dt} \rightarrow V_L dt = L dI_L$$

(Ec. 2.2.3)

$$I_L(0) = I_L(t)$$

(Ec. 2.2.4)

Como se puede observar en la ecuación anterior, el estado es estable, por lo tanto:

$$\int_0^t V_L dt = \int_0^t L dI_L = 0$$

(Ec. 2.2.5)

Las áreas A y B de la gráfica superior de la Figura 4, correspondiente a la tensión en la bobina, han de ser iguales, por lo que:

$$(V_d - V_0)t_{ON} - V_0 t_{OFF} = 0$$

(Ec. 2.2.6)

$$\frac{V_0}{V_d} = D, \qquad 0 < D < 1$$

(Ec. 2.2.7)

En la Figura siguiente, extraída de [4], vamos a poder observar más claramente la forma de onda del rizado de la tensión de salida, así como el de la corriente por la bobina:

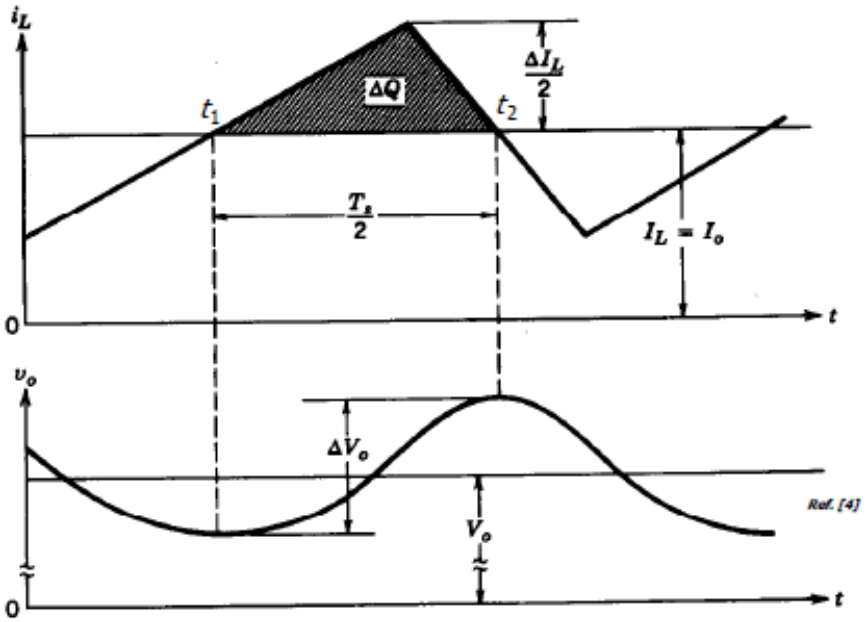


Figura 5

Donde el valor de pico de la corriente viene dado por la suma de la corriente media, más el rizado tal y como se puede observar en la siguiente fórmula:

$$\hat{I}_L = i_L(0) + \Delta I_L = i_L(0) + \frac{V_d - V_0}{L} t_{ON}$$

(Ec. 2.2.8)

Por otra parte, el rizado de la tensión de salida depende de la corriente que circula por el condensador, ya que es este es el encargado de proporcionar una tensión de salida estable.

$$i_C = C\frac{dV_0}{dt} = i_L - i_0 \cong i_L - I_L$$

(Ec. 2.2.9)

$$\Delta V_0 = \frac{1}{C}\int_{t1}^{t2} i_C dt \cong \frac{1}{C}\frac{1}{2}\frac{T_S}{2}\frac{\Delta I_L}{2} = \frac{T_S \Delta I_L}{8C}$$

(Ec. 2.2.10)

2.3 El reductor Buck síncrono

En esta etapa de potencia, en comparación con la explicada anteriormente, la principal diferencia se traduce en la forma de controlar el circuito. El diodo de circulación libre es sustituido por un transistor MOSFET de potencia. El MOSFET se selecciona de modo que sus pérdidas en conducción sean menores que las pérdidas del diodo rectificador, para así aumentar la eficiencia. Aunque esto complica el diseño de circuito de activación de la puerta, el mayor rendimiento hace de esta una opción atractiva. Otras consideraciones de la etapa síncrona de potencia del *Buck* previenen la conducción simultánea de ambos interruptores y la recuperación en inversa del diodo parásito de la unión PN del MOSFET.

El circuito *driver* debe asegurar que ambos MOSFET no estén encendidos simultáneamente; esto colocaría una resistencia parásita muy pequeña entre la entrada y tierra y podrían fluir corrientes destructivas a través de los MOSFET's. Es necesario un pequeño tiempo con los dos interruptores apagados.

Otra característica de la etapa de potencia del *Buck* síncrono es que funciona siempre en modo de conducción continua (MCC). Así la relación de tensión (ciclo de trabajo) y la función de transferencia del voltaje para la etapa de potencia del Buck síncrono son iguales que para la etapa de potencia de un *Buck* en MCC

La Figura 6 muestra un diagrama esquemático simplificado del Buck síncrono con un bloque driver incluido. Ambos interruptores son MOSFET de canal N

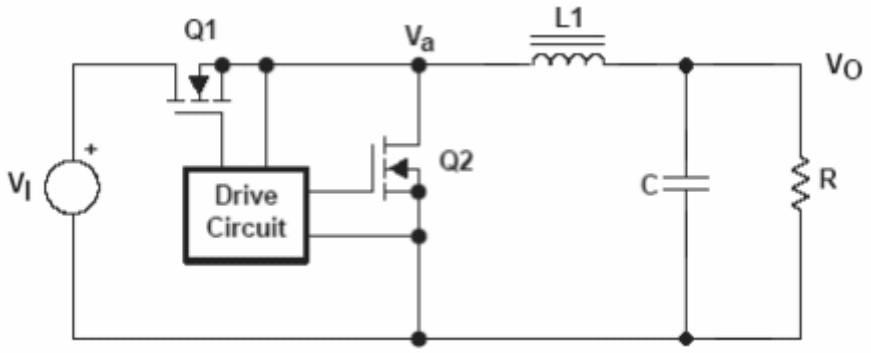


Figura 6

Al igual que en caso anterior, este presenta dos configuraciones distintas dependiendo del estado de Q1 y Q2:

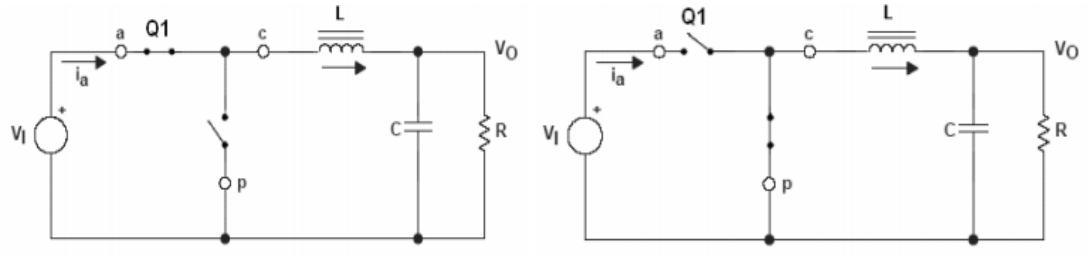


Figura 7

Tanto las ecuaciones que gobiernan el sistema como las gráficas resultantes son las mismas que se han explicado explícitamente en el apartado anterior.

El Slew-Rate de la corriente de salida requerido es alto. Esto requiere que el inductor de salida sea tan pequeño como sea posible.

Desafortunadamente, usar un pequeño inductor para alcanzar una respuesta transitoria más rápida hará que el rizado de la tensión de salida incremente como muestra la Ec. 2.2.8. Esto requiere que la frecuencia de la conmutación sea aumentada. Una frecuencia de conmutación más alta producirá más pérdidas de conmutación, por lo tanto, el rendimiento disminuirá.

Otra manera de disminuir el rizado de la tensión de salida es aumentar el condensador de salida, según lo mostrado en la Ec. 2.2.10.

Así pues, cuanto más alta es la frecuencia de conmutación, más rápida es la respuesta del lazo cerrado, y más pequeño es el valor requerido del condensador de salida para limitar el rizado de la tensión de salida dentro de los márgenes requeridos, y se requerirá un inductor de salida más grande para funcionar en MCC para alcanzar menos rizado de la corriente de salida.

3. Diseño del soporte hardware

En este capítulo se muestran los pasos seguidos para la construcción del modelo físico del reductor de tensión, partiendo del esquema inicial y la selección de componentes comerciales. Adicionalmente, se muestra el diseño y construcción del prototipo final con la ayuda del software *Eagle* para el desarrollo de placas PCB.

3.1 Selección de componentes

El primer paso que se ha realizado para la construcción del prototipo es el diseño de un esquemático donde poder reflejar todos los elementos necesarios. En las siguientes Figuras de este apartado, se muestra el esquema utilizado que incluye tanto la topografía del reductor, como los *drivers* asociados a los disparos de los MOSFETs, los circuitos de acondicionamiento de tanto de la tensión de salida como de la tensión en la bobina, los diferentes puntos de test, así como el integrado para poder obtener la V_{IL} o el conector para la unión con la FPGA.

Se han intentado aprovechar algunos de los componentes disponibles en los laboratorios de la escuela y utilizados en algunas de las prácticas del grado de Ingeniería Electrónica y Automática.

Se ha buscado un driver (LM5104M de Texas Instruments), componente U3 de la Figura 8, específico para el control del disparo de los MOSFETs de un reductor *Buck* síncrono. Este dispositivo controla el disparo de los dos transistores con una única entrada y con un tiempo muerto ajustable a través de una resistencia externa. También incluye la técnica de *Bootstrap* para el disparo del transistor superior del *Buck*, simplificando considerablemente el montaje. El cálculo tanto de la resistencia como del condensador se pueden ver en Anexo III.5 Selección de la resistencia para el ajuste del tiempo muerto, Anexo III.4 Cálculo del valor del condensador Bootstrap respectivamente.

Para la conmutación de una segunda resistencia de carga en paralelo se ha optado por un driver mucho más sencillo y económico, pero también de pequeñas dimensiones (UCC27517DBVT de Texas Instruments), componente U2 de la Figura 8.

La resistencia de shunt utilizada para la medida de la corriente por la bobina es de carácter general, de Vishay, cuyo valor se calcula y justifica en Anexo III.3.2 Circuito de acondicionamiento de la medida de corriente y cálculo de Rshunt

El amplificador utilizado para esta medida de corriente es el INA214, componente U1 de la Figura 8, de Texas Instruments, ya que este es recomendado por el fabricante de la FPGA para esta aplicación de medida de corriente. [5]

Para las resistencias y condensadores de carácter general se ha utilizado el encapsulado 0805 para economizar el espacio del diseño. No obstante, para las resistencias de carga se han usado encapsulados cerámicos.

Los elementos de conmutación utilizados son los transistores MOSFET de canal N IRFZ234N (de Infineon), estos son unos MOSFET de potencia en encapsulado TO-220AB.

Como componentes críticos del Buck, cuyo cálculo es motivo de estudio, se consideran la bobina y el condensador de salida cuya explicación detallada y cálculos correspondientes se encuentran en Anexo III.1 Cálculo de los parámetros más relevantes del *Buck*

Como alimentación para los distintos integrados, U1, U2, U3, se ha elegido que esta sea la misma que la de entrada al convertidor, y esto es debido a que, basándonos en sus hojas de características, se observa que este valor de alimentación, 12 V, se encuentra dentro del rango permitido por estos tres integrados. El más restrictivo de los tres es integrado U1, amplificador de la medida de corriente, cuya alimentación va de 9 a 14 V, por lo tanto, la tensión elegida se adecua perfectamente.

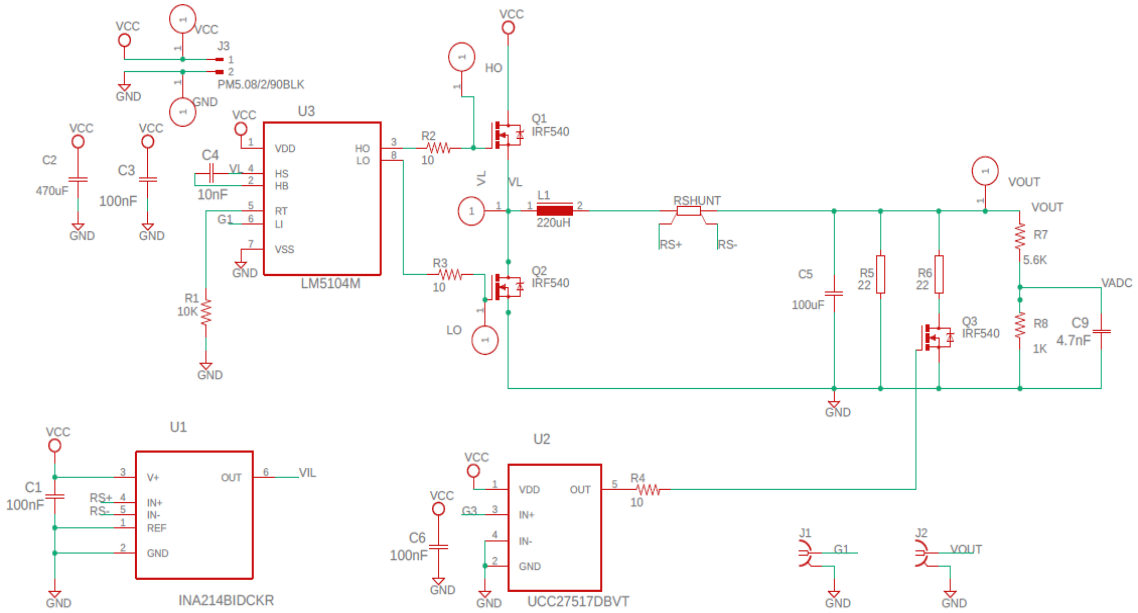


Figura 8

En la Figura 9 se pueden observar los circuitos de acondicionamiento de la señal comentados en este mismo apartado, cabe destacar que estos son los recomendados por el fabricante de la FPGA, [5]. El cálculo de los valores de los componentes de estos circuitos se justifica en Anexo III.3 Cálculo acerca de la etapa acondicionamiento del conversor analógico/digital.

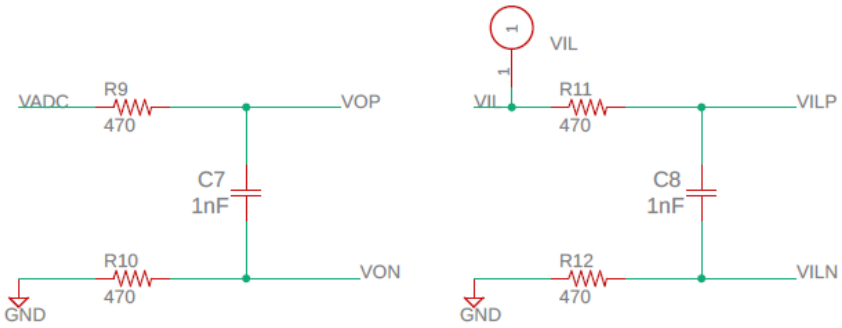


Figura 9

Por último, en la Figura 10 se observa el conector correspondiente para realizar la conexión con la FPGA. Se trata de un conector de 12 pines con un paso de 2.54mm. Debido a la distribución de estos marcados por el fabricante, se ha intentado distribuir estos de forma que el posterior trazado de las pistas se realice de la forma más eficaz posible.

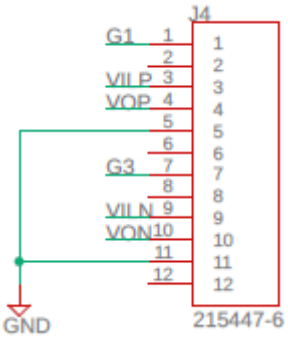


Figura 10

3.2 Etapa de acondicionamiento y sincronismo del sistema

Tanto para realimentar la tensión de salida del reductor *Buck*, como para la tensión en la bobina, el nivel de tensión ha de acondicionarse para su conexión con la entrada analógica de la FPGA. Ya que la tensión máxima de entrada que admite el conversor A/D de la FPGA es de 1V se ha calculado que la ganancia de esta etapa ha de ser de 0.1515 en el caso de la tensión de salida, para asegurar que de esta forma el valor de lectura del conversor A/D en el régimen estacionario se encuentra en torno al 70% del valor máximo del rango dinámico del conversor. En el caso de corriente que circula por la bobina, esta adaptación de niveles a la entrada del conversor se ha realizado eligiendo adecuadamente la resistencia de shunt y el amplificador conociendo la corriente máxima que pasa por ésta. Para consultar cálculos más detallados acerca de la etapa de acondicionamiento consultar Anexo III.3 Cálculo acerca de la etapa acondicionamiento del conversor analógico/digital.

Adicionalmente, y por recomendación del fabricante [5], resulta necesaria la inclusión de un filtro anti-aliasig a la entrada del conversor. Este filtro tendrá una incidencia directa en el cálculo del tiempo de establecimiento de la tensión. Estos cálculos se encuentran explicados detalladamente en Anexo III.3 Cálculo acerca de la etapa acondicionamiento del conversor analógico/digital.

3.3 Diseño de la placa de circuito impreso

Para la construcción de la placa de circuito impreso sobre la que va montada el prototipo final se ha usado el software comercial *Eagle* de *CadSoft*.

Partiendo del esquemático explicado en el apartado anterior, se procede a realizar la placa de circuito impreso. En el apartado anterior se ha separado el esquema en sus diferentes partes, con el fin de que estas se puedan observar claramente. No obstante, en el presente apartado se incluye la figura correspondiente al esquemático completo, Figura 11 la cual se puede consultar al final de este apartado.

En cuanto al diseño de la placa, el circuito realizado es de doble cara y se ha previsto que sea sencilla de soldar sin necesidad de disponer de taladros metalizados. El posicionamiento de los componentes se ha realizado de tal manera que las pistas queden lo más cortas, directas, ordenadas y sin cruces posibles y agrupando elementos según su función, este posicionamiento se puede observar en la Figura 12. Los nombres de

componentes posicionados junto a los componentes y en un mismo sentido para que te ayuden en la soldadura. La anchura de las pistas es de 0.5 mm en la mayoría de los casos. Hay situaciones en las que he utilizado otro ancho de pistas, como por ejemplo en las pistas de alimentación, tensión de salida y tensión en la bobina, que son de 1 mm. Por otra parte, he tenido que reducir el ancho de pistas a 0.3 mm en el caso de algún integrado como el componente U1, esto es debido a que he tenido que ajustar el ancho de pistas para que este no fuera mayor que la anchura de los *pads* del integrado para evitar problemas de que varios *pads* quedaran unidos, o que su soldadura fuera muy complicada. Cabe destacar que la mayoría de las pistas van por la cara TOP debido a que la mayoría de los componentes son de montaje superficial (SMD). Por otra parte, las pistas de los puntos de test se han trazado por la cara BOTTON, debido que su soldadura se simplifica notablemente. Como ya se ha comentado anteriormente, la distribución correcta de los pines del conector J4 ha resultado crucial para poder obtener un diseño compacto. Se ha hecho especial hincapié en los caminos de retorno de masa y en las señales criticas tales como G1, G3, o las pistas que salen de los transistores MOSFET, ya que se trata de señales de alta frecuencia, o las señales que llevan la tensión a los conversores de la FPGA. Se ha procurado evitar los cruces por las diferentes caras con objeto de facilitar los retornos de masa. Por último, se han añadido planos de masa por ambas caras. En las siguientes figuras se puede observar como ha quedado finalmente el diseño de la placa de circuito impreso. La Figura 13 se corresponde con el trazado de pistas por la cara TOP, mientras que la Figura 14 se corresponde con el trazado de pistas por la cara BOTTON.

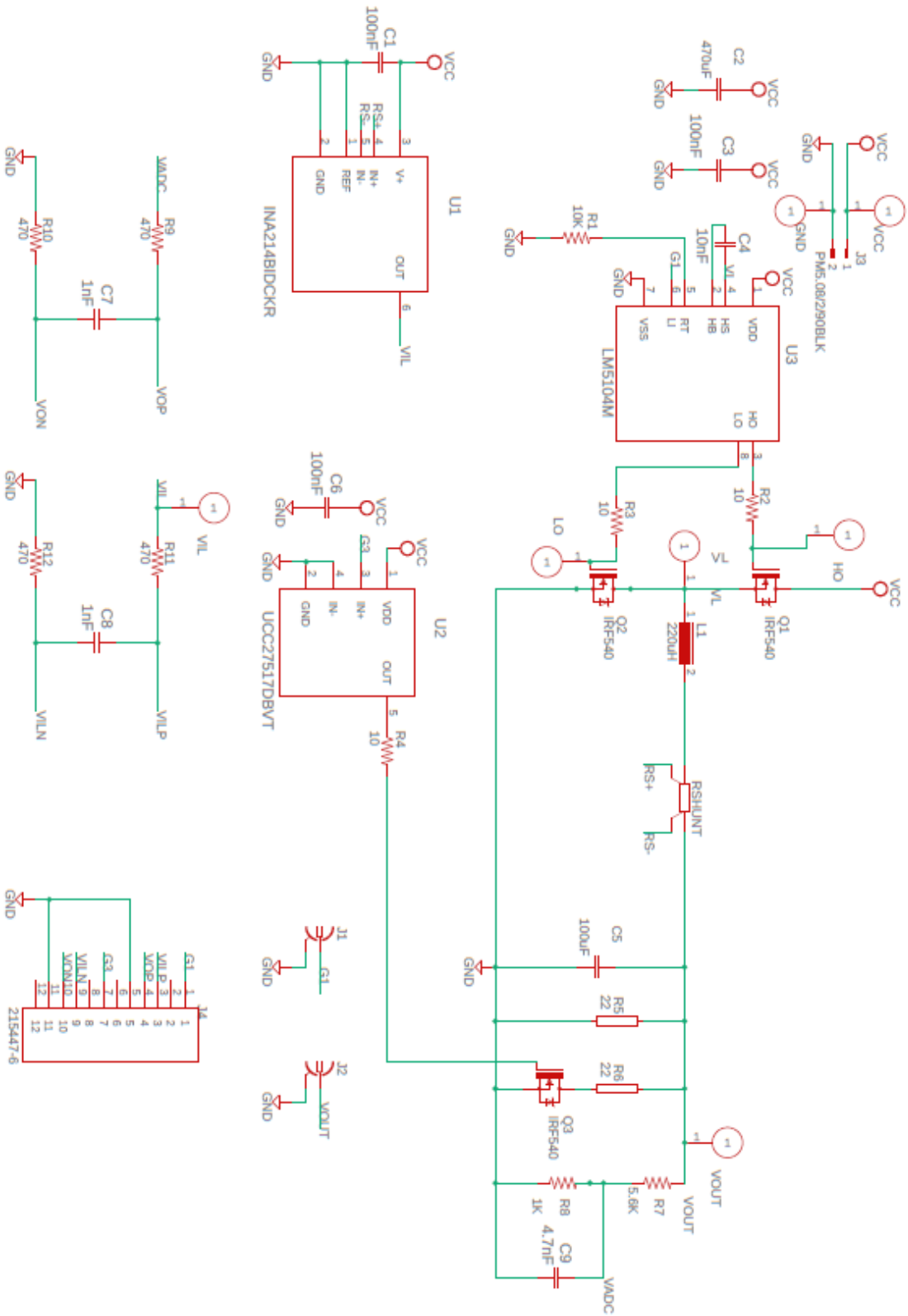

C1
100nF
VCC
GND
U1
INA214BIDCKR
VADC
R10
470
R9
470
C7
1nF
VON
VOP
VIL
R12
470
R11
470
C8
1nF
VILN
VILP
C6
100nF
U2
UCC27517DBVT
R4
10
J1
J2
VOUT
J4
215447-6
C2
470uF
C3
100nF
R1
10K
C4
10nF
U3
LM5104M
J3
PM5.08/2/90BLK
R3
10
R2
10
LO
HO
VL
Q2
IRF540
Q1
IRF540
L1
220uH
RSHUNT
RS+
RS-
C5
100uF
R5
22
R6
22
Q3
IRF540
R8
1K
R7
5.6K
C9
4.7nF

Figura 11

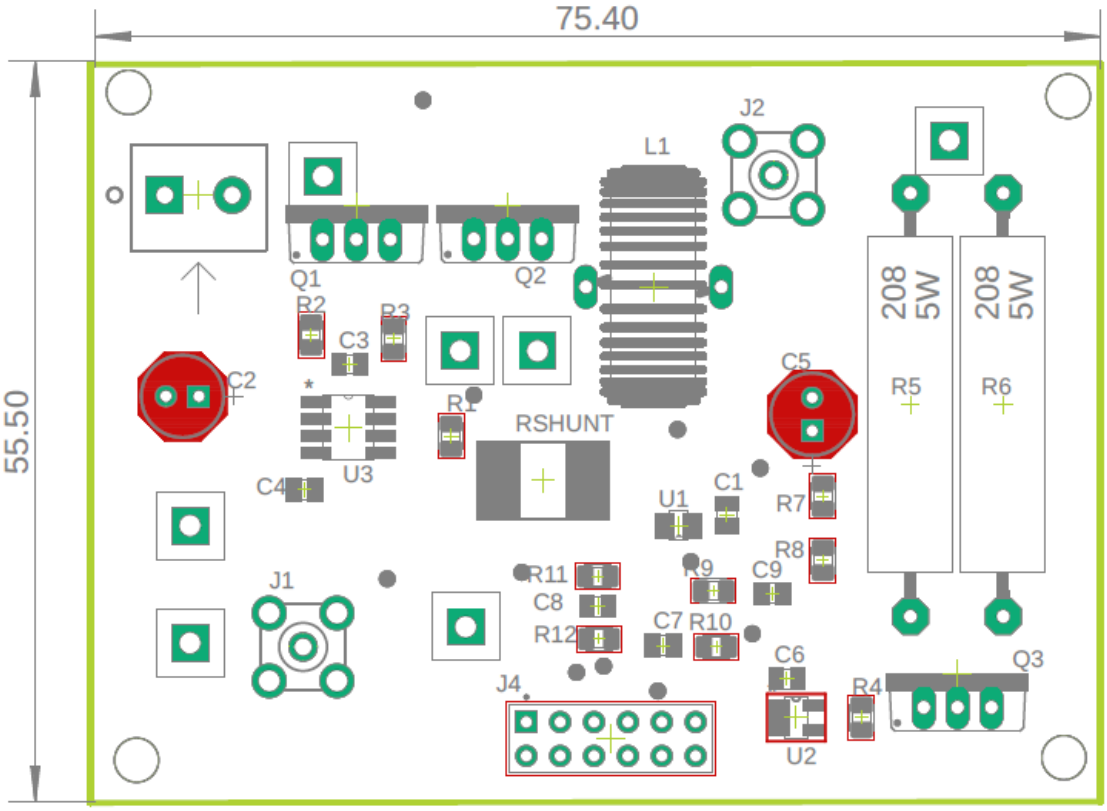


Figura 12

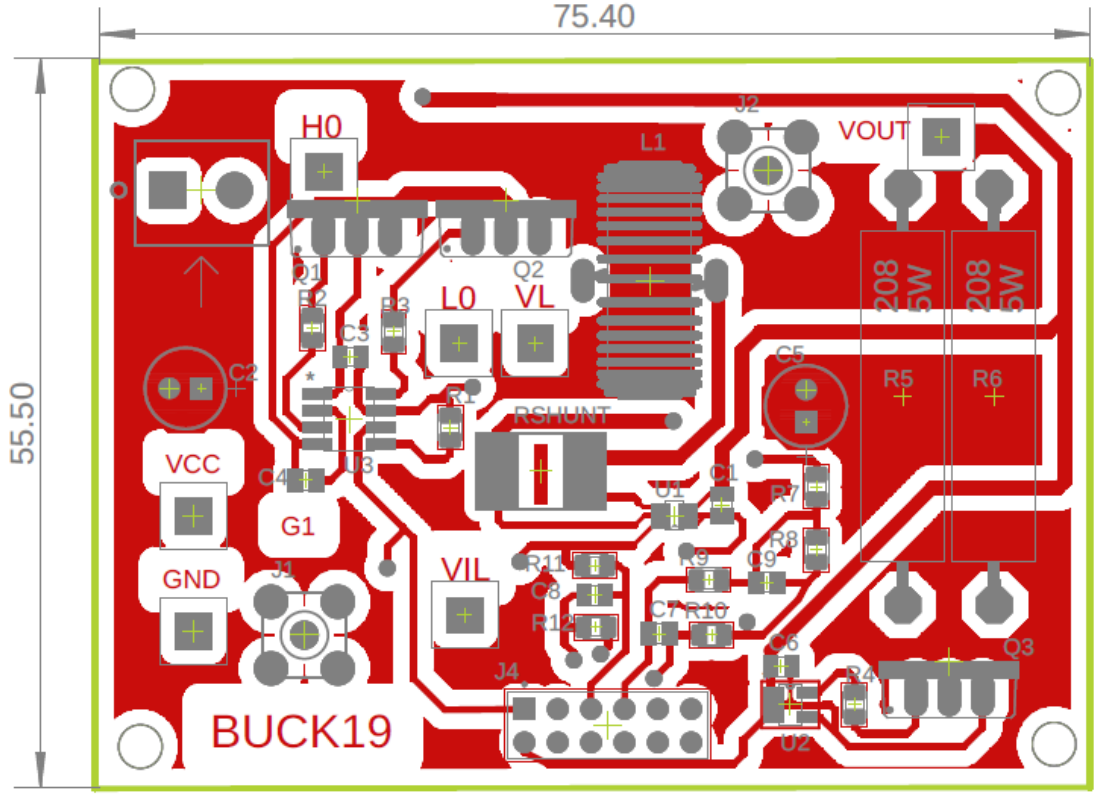


Figura 13

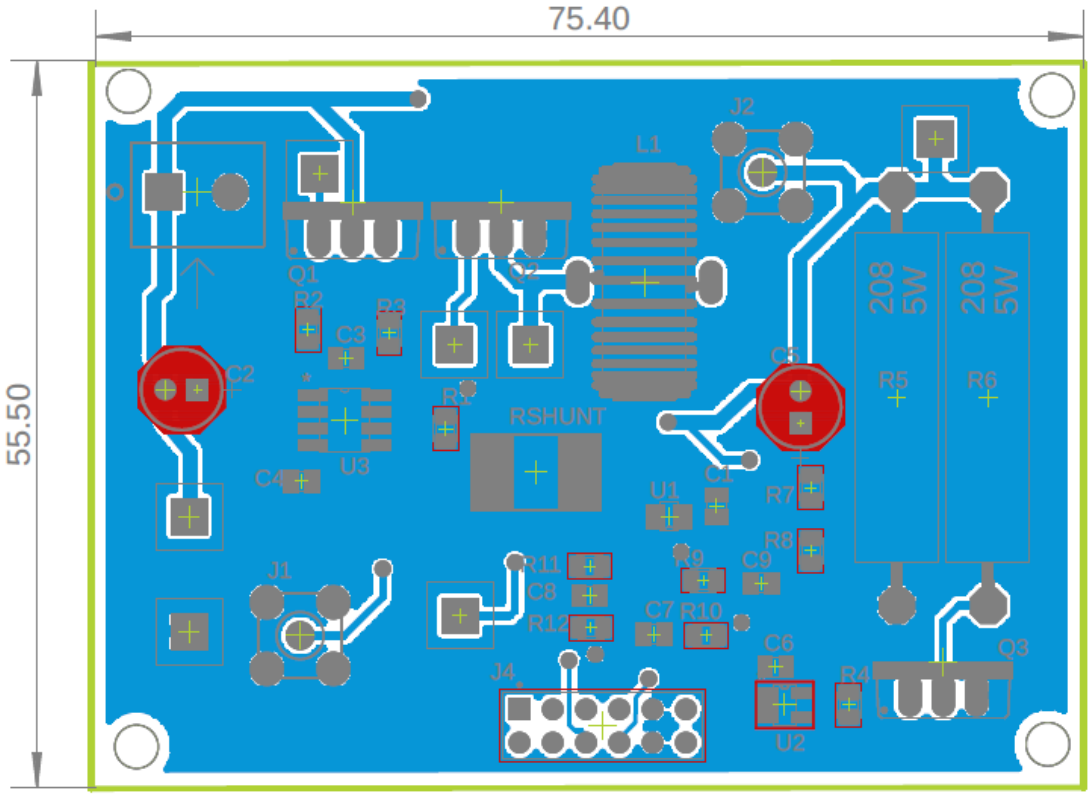


Figura 14

A continuación, se puede observar el resultado final de la placa de circuito impreso.

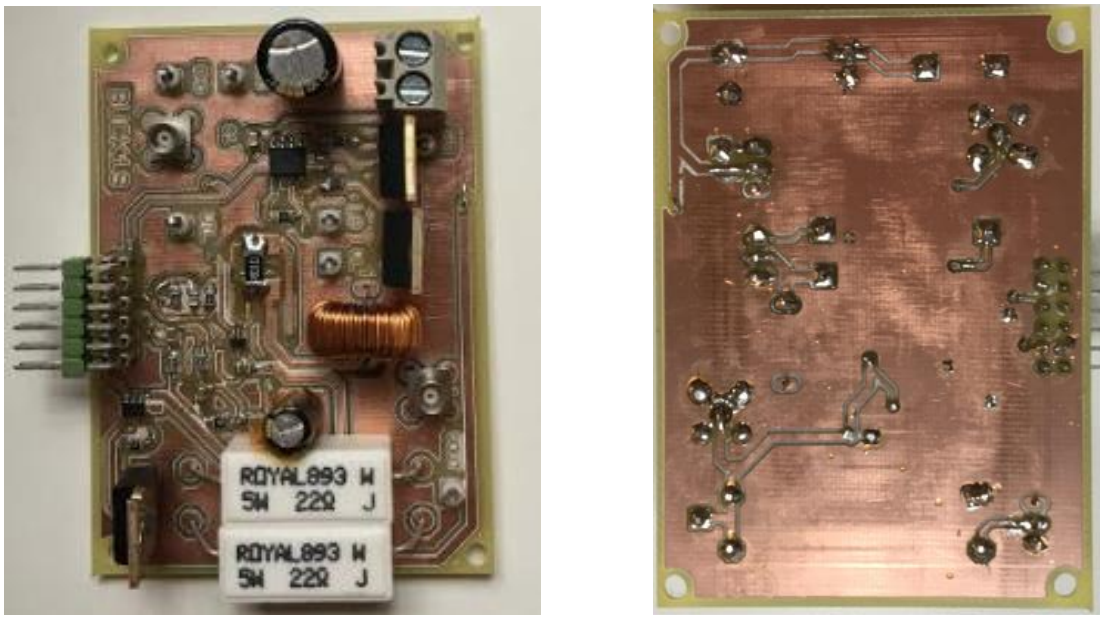


Figura 15

También se va a mostrar cómo queda finalmente la conexión de ésta con la placa Basys 3 de FPGA.

Figura 16

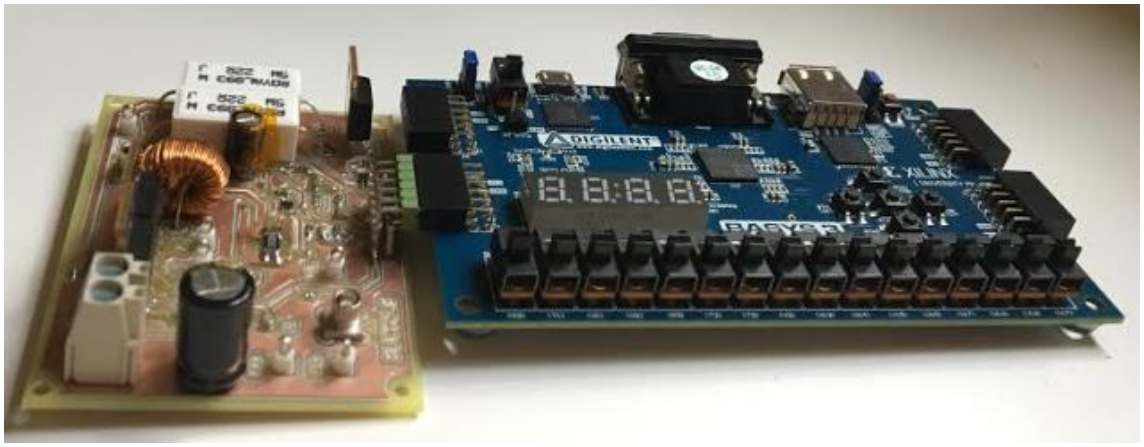


Figura 17

Adicionalmente se presenta el listado de componentes de la placa de circuito impreso.

Parts	Ref.Fab	Fab	Farnell Code
C5	EEUFM1E101	Panasonic	5261430
C1, C3, C6	C0805C104K5RALTU	KEMET	2679994
C4	08051C103KAT2A	AVX	1658874
C2	EEUFM1E471	Panasonic	5261547
C7,C8	C0805C102K4RACTU	KEMET	2855511
C9	MC0805B472K500CT	MULTICOMP	1759241
R1	CPF-A-0805B100KE1	TE CONNECTIVITY	2483949
R7	CPF-A-0805B5K6E1	TE CONNECTIVITY	2484007
R8	CPF-A-0805B1K0E1	TE CONNECTIVITY	2483964
R9,R10,R11,R12	CPF-A-0805B470RE1	TE CONNECTIVITY	2483998
R5, R6	SQP7S-22RJB15	Welwyn	1292536
R2, R3, R4	MC0805S8F100JT5E	MULTICOMP	2371697
Rshunt	WSL2010R0100FEA18	VISHAY	2483580
L1	7447020	Wurth	2082527
U3	LM5104M	National Semiconducto	8181390
U2	UCC27517DBVT	Texas Instruments	2099942
U1	INA214AQDCKRQ1	TEXAS INSTRUMENTS	2782424
X1	90122-0766	Molex	1248189
J1, J2	1-1337581-0	TE Connectivity	1909247
J3	PM5.08/2/90 BLK	Weidmuller	1131855
Q1, Q2, Q3	IRFZ34N L405	Infineon	8648298
	1421T5CL	Hammond	1876520
GND, HO, LO, VCC, VL, VOUT, VIL	5011	Keystone	8731225

4. Diseño del controlador digital

En el presente aparatado se parte de la función de transferencia obtenida en Anexo I. Obtención de la función de transferencia en pequeña señal del *Buck.*, para obtener un controlador en el espacio continuo a través de la aplicación software para *Matlab Sisotool* y apoyándonos en textos de referencia como [6]. Una vez obtenida la ecuación de dicho controlador, se procede a su discretización a través de un método adecuado.

Finalmente, se realiza la transformación del formato de los coeficientes del regulador a coma fija para su implementación en la FPGA.

En cuanto a las simulaciones pertinentes para comprobar el correcto funcionamiento del sistema y por tanto el correcto cálculo del controlador, estas se encuentran en el capítulo 6. Resultados.

4.1 Selección de la frecuencia de muestreo

En los sistemas electrónicos de potencia conmutados, una alta frecuencia favorece la obtención de poco rizado en la tensión de salida del convertidor. Por otro lado, el *temporizador* encargado de la generación de la señal PWM debe contar con la definición suficiente para evitar problemas relacionados con la aparición del fenómeno de los ciclos límite.

Por ello se ha elegido implementar un contador con una definición de 1000 valores a una frecuencia de 100 MHz (frecuencia de trabajo del reloj de la FPGA), lo cual genera una frecuencia de conmutación para el *Buck* de 100 KHz.

Finalmente, cabe destacar que en los sistemas eléctricos controlados mediante modulación PWM aparecen corrientes con componentes frecuenciales de valores múltiplos de la frecuencia de conmutación f_{SW}. Para cumplir el teorema de muestreo de Nyquist la frecuencia de muestreo f_S debería ser muy inferior a f_{SW}. Siguiendo las técnicas de control clásicas se elige una frecuencia de muestreo 20 veces inferior a la frecuencia de conmutación, por lo que esta será de 5 KHz. Adicionalmente esta tiene que ser mayor que la frecuencia de corte deseada de la planta para atenuar el pico de resonancia y con ello rechazar perturbaciones.

4.2 Consideración acerca de los ciclos límite

La no linealidad introducida por la cuantificación del conversor A/D y la señal digital de control PWM puede genera oscilaciones de baja frecuencia en la salida conocidas como ciclos límite. Para evitar la aparición de ciclos límite relacionados con estos elementos, ha de cumplirse la siguiente relación [7]:

$$\mathrm{q_{ADC}} > \mathrm{H} \cdot V_g \cdot q_{PWM}$$

(Ec. 4.2.1)

Siendo *H* la ganancia de la etapa de acomodación, V_g la tensión aguas arriba del reductor, q_{ADC} el paso de cuantificación del A/D y q_{PWM} el paso de cuantificación del PWM. Configurando la definición del conversor A/D que integra la FPGA a 9 bits podemos comprobar que la desigualdad anterior se cumple garantizando la no aparición de ciclos límite por estos motivos.

$$\frac{1}{2^9} > 0.1515 \cdot 12 \cdot \frac{1}{1000}$$

$$0.00195 > 0.001818$$

4.3 Obtención del regulador

Para una información más detallada sobre el procedimiento seguido para el cálculo del regulador consultar Anexo III.2 Cálculo del Regulador

La ecuación del regulador obtenida es la siguiente:

$$C(s) = 4.04\frac{(s+3142)(s+12531)}{s(s+78762)}$$

(Ec. 4.3.1)

A continuación, a través de la función *c2d* se ha discretizado el regulador para un periodo de muestreo con una frecuencia de muestreo igual a la frecuencia de conmutación, 100 KHz. Para llevar a cabo dicha discretización se ha forzado al programa a usar el método de la transformada bilineal a través de la opción *'tustin'*.

Con todo ello el regulador obtenido, esta vez en el espacio discreto, tiene la siguiente forma:

$$C(z) = \frac{D(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

(Ec. 4.3.2)

Donde $(b_0, b_1, b_2) = (3.129, -5.791, 2.674)$ y $(a_1, a_2) = (-1.435, 0.435)$. El polo en z=1 se corresponde al integrador que anula el error de posición, para no perderlo hay que asegurarse que se cumple que $1 + a_1 + a_2 = 0$.

$$C(z) = \frac{D(z)}{E(z)} = \frac{3.129 - 5.791 z^{-1} + 2.674 z^{-2}}{1 - 1.435 z^{-1} + 0.1435 z^{-2}}$$

(Ec. 4.3.3)

Una vez obtenida la ecuación del controlador discreteado se procede a plantear una ecuación en diferencias para disponer de un modelo de simulación en lazo cerrado de todo el sistema para poder comprobar su funcionamiento en las diferentes fases del diseño, ver Figura 15.

No es necesario que el modelo sea exacto a nivel de ciclo de reloj, nos basta con que represente el comportamiento a nivel de periodo de muestreo T_S del controlador. Por tanto, obtendremos un modelo discreto con periodo T_S en coma flotante de todos los elementos del lazo de control. La ecuación en diferencias resultante una vez incluidas las ganancias $K_{ADC}=2^9/1.0$ y $K_{PWM}=1/1000$:

$$dcf(k) = b_0'e(k) + b_1'e(k-1) + b_2'e(k-2) - a_1 dcf(k-1) - a_2 dcf(k-2)$$

(Ec. 4.3.4)

Donde:

$$(b_0', b_1', b_2') = (b_0, b_1, b_2)/(K_{PWM}K_{ADC})$$

La salida dfc(k) debe ser redondeada a un entero y limitada entre 100 y 900, esta limitación del duty es para evitar que el interruptor este en *ON* o en *OFF* durante todo un periodo de conmutación y mantener la frecuencia constante.

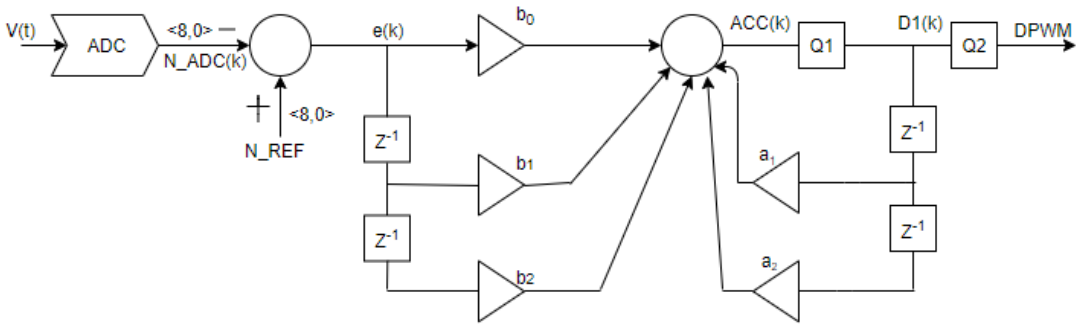


Figura 18

4.4 Codificación en formato de coma fija

El muestreo del conversor A/D estará sincronizado con el periodo de conmutación. En la Figura 19, sacada de [7], se muestra la planificación temporal donde CNT_TS representa el contador del PWM digital, N1 es el instante en que se inicia la conversión A/D, y N2 el instante en que el resultado de la conversión está disponible y comienza el cálculo del controlador.

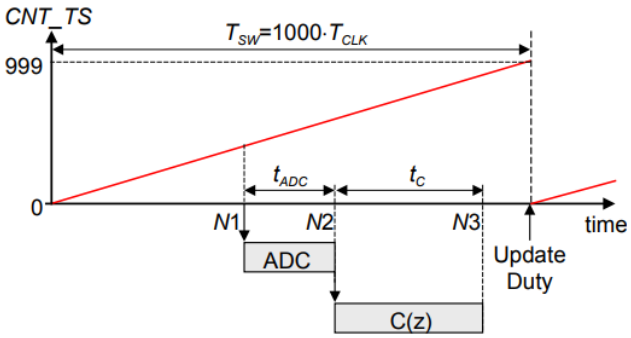
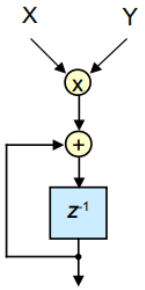


Figura 19

4.4.1 Selección de la estructura de realización del controlador.

Existen diferentes estructuras para implementar las ecuaciones en diferencias del controlador: directas, en cascada, paralelo… Se sugiere utilizar una forma directa que realice la ecuación en diferencias para obtener dcf(k). La ecuación se implementará con un bloque de multiplicación y acumulación (MAC). La Figura 20, sacada de [7], muestra la estructura y operación de la unidad MAC, de forma que en cada ciclo de reloj se realiza una operación.



#ciclo	X	Y	ACC op.
N2	b_0	$e(k)$	ACC←X×Y
N2+1	b_1	$e(k-1)$	ACC←ACC + X×Y
N2+2	b_2	$e(k-2)$	ACC←ACC + X×Y
N2+3	$-a_1$	$dcf(k-1)$	ACC←ACC + X×Y
N2+4	$-a_2$	$dcf(k-2)$	ACC←ACC + X×Y

Figura 20

Este es el planteamiento inicial, posteriormente durante la implementación se ha tenido que realizar el bloque de multiplicación y acumulación en ciclos diferentes, debido a que si estos se hacían en el mismo nos surgía un problema con los tiempos en la implementación. Además, cada término de la multiplicación se ha tenido que ir acumulando ciclo a ciclo en una variable, la cual le pasamos luego al multiplicador, esto se ha realizado por la misma razón que antes. Cabe destacar que el hecho de acumular valores por separado para su posterior multiplicación nos retrasa un ciclo de reloj cada una de las operaciones posteriores, multiplicación y posterior acumulación, esto se ha debido tener en cuenta a la hora de establecer los valores de N1 y N2, ver Figura 19, ya que hay que asegurar que estas operaciones se realizan dentro de cada periodo.

4.4.2 Codificación de los coeficientes en coma fija

Para la posterior realización del regulador en la FPGA, se ha decidido que tanto el formato de los coeficientes del controlador, como de los distintos parámetros necesarios para poder implementar la ecuación en diferencias, tales como errores, acumuladores, etc., se van a implementar en un formato de coma fija.

Se ha buscado implementar dichos coeficientes en la estructura de datos mínima que fuera posible, pero conservando la precisión necesaria. Se ha optado por el tipo de dato *sfixed disponible en el empaquetamiento fixed_std* de VHDL-2008. Los multiplicadores de la FPGA admiten un tamaño máximo de 25x18, por lo tanto, son estos los que nos va a marcar la restricción a la hora de asignar tamaño a las distintas señales.

Es necesario garantizar el rango dinámico en todos los nodos intermedios del controlador para que la respuesta en lazo cerrado del sistema sea similar al caso en coma flotante. La parte entera debe ser lo bastante grande para incluir la excursión completa de la señal sin saturaciones en el caso más desfavorable (transitorio de arranque) y la parte fraccionaria debe ser suficiente para que no aparezcan efectos adversos de cuantificación como, por ejemplo, ciclos límite.

A continuación, voy a justificar el cálculo y dimensiones de los principales nodos para implementar la ecuación en diferencias:

- Para el error se ha a usar un formato de 10 bits, todos ellos de parte entera, <10,0>, ya que este es el resultado de restar el dato proveniente del ADC y la referencia, siendo estos dos últimos datos de 9 bits. Se da un bit más para evitar pérdida de información con posible acarreo, ver Figura 18.

- En cuanto al acumulador que vaya almacenado el valor de la ecuación en diferencias, se realizaron simulaciones en Matlab para determinar a qué valor máximo podría llegar, con el fin de darle los bits suficientes para no perder información. Se determinó que este acumulador podía llegar hasta un valor de 14.000 por lo que se le asigna una parte entera de 15 bits, los bits necesarios para que quepa este dato en complemento a 2. Para la parte fraccionaria le queda una dimensión de 10 bits, los máximos que nos quedan restantes para legar a completar el rango de este tipo de datos, la cual consideramos suficiente. Este formato <14,10>, se corresponde con el acumulador cuantificado, ya que para ir acumulando el resultado de la ecuación en diferencias se utiliza un formato de <21,-20>, con el fin de no perder nada de información por el camino, y es al final cuando se reduce el número de bits al formato comentado anteriormente.

- Para los coeficientes del controlador se ha utilizado un formato de 7 bits de parte entera y 10 de parte fraccionaria para b_0, b_1, b_2, <7,-10>, y un formato de 6 bits de parte entera y 10 de parte fraccionaria para a_1, a_2, <3-10>. Para llegar definitivamente a estos valores, se fueron probando distintos valores con el fin de encontrar aquel que nos permitiera aprovechar el máximo rango de bits para su precisión, pero sin sobrepasar cierto límite, puesto que luego estos coeficientes van a ir multiplicados por el error y el acumulador final no puede exceder del número de bits máximo para no perder información.

Para la señal que posteriormente servirá para comparar con el *temporizador* para generar la señal de control del PWM, partimos de la señal del duty, que tiene el mismo formato que el acumulador cuantificado, <14,-10>, esta sufre también una cuantificación al formato, <10,0>, para que este sea igual que el formato del *temporizador*, y mismo tipo, unsigned, con el cual se va a comparar.

5. Implementación del control en VHDL

En el presente capitulo se expone el trabajo realizado en el programa Vivado en el lenguaje VHDL. Se procede a explicar cuáles son los bloques principales que gobiernan el sistema explicando cual es la función de cada uno de ellos, quedando el código al completo recogido en Anexo IV. Diseño en lenguaje VHDL, para su consulta. La idea principal es realizar un diseño mediante el cual seamos capaces de leer las variables medidas en placa de circuito impreso, V_0, V_{IL}, estando ya estas adecuadas a los niveles de tensión que acepta la FPGA a su entrada, convertir estas variables a un valor digital mediante los conversores analógico/digitales presentes en la FPGA. Posteriormente, se realizará el control de dichas variables para obtener las especificaciones requeridas, 5V a la salida de nuestro convertidor *Buck*, mandándole las señales de control necesarias a la placa de circuito impreso.

5.1 Diseño del diagrama de bloques

Vivado presenta una interesante forma de representar e interconexionar los diferentes bloques necesarios para llevar a cabo el control del sistema. Se ha escogido esta forma de realizarlo ya que se ha considerado que es la más clara frente a realizarlo mediante un nuevo fichero VHDL. En la Figura 28 al final del presente apartado se puede observar dicho diagrama.

Como entradas al sistema podemos observar el clock, de 100MHz, y el reset, el cual se ha determinado que sea activo en alto. Además, podemos observar tres señales adicionales de entrada las cuales se corresponde con tres interruptores, mediante los cuales somos capaces de variar la carga a la salida, SW_HI_LOAD_IN, o de desactivar el funcionamiento del controlador, SW_CLOSED_IN. La función del tercero, SW_CHANNEL, es seleccionar cual es el canal que queremos visualizar a la salida.

Como salidas del sistema tenemos las dos señales de control del Buck, G1 para generar el PWM y G3 para realizar los cambios de carga. Adicionalmente, se observar otros 3 grupos de señales, AN [3:0], DPSEG y SSEG [6:0], las cuales son utilizadas para la visualización en los displays propios de la placa de FPGA.

El primer bloque con el que nos encontramos es el Syncro_ctl, Figura 21, mediante el cual somos capaces de sincronizar las entradas comentadas anteriormente, por lo que a la salida de éste se encuentran las diferentes señales sincronizadas.

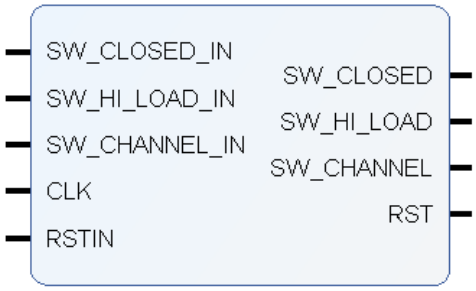


Figura 21

El bloque xdac_wiz, Figura 22, correspondiente al conversor analógico/digital, cuyo funcionamiento y configuración se encuentran explícitamente detallados en Anexo II. Configuración del conversor analógico/digital interno de la FPGA, es el encargado de recibir tanto la tensión de salida del Buck, como la corriente de la bobina, con los niveles de tensión adecuados, y de convertir estos en valores digitales con el fin de poder utilizar estos para el posterior control del sistema y visualización.

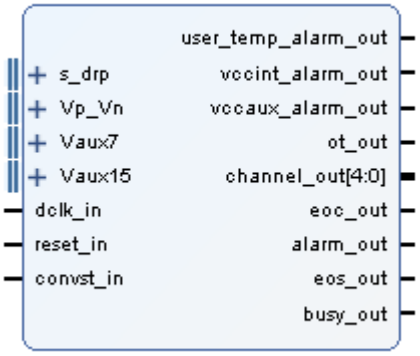


Figura 22

El bloque ctl_adc, Figura 23, es el encargado una vez haya terminado la conversión, mandarle una señal al conversor, DEN, especificándole la dirección de la señal que queremos mediante la salida ADDR [6:0], para obtener un nuevo dato a la salida, una vez esté ya disponible a la salida, DO_VALID, almacenamos el dato y se realiza el mismo proceso para almacenar el dato del otro conversor. Adicionalmente los acondiciona al formato requerido. Una vez haya almacenado los dos valores, es el encargado de generar la señal DA_VALID, que sirve de habilitación para algunos de los bloques.

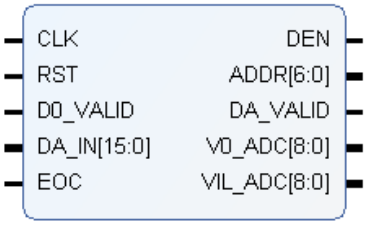


Figura 23

En cuanto al bloque Buck_ctl, Figura 24, este es el bloque en cual se lleva a cabo la implementación del controlador. En cuanto a sus entradas podemos observar el resultado de la conversión correspondiente a la tensión de salida, el cual proviene del bloque ctl_adc, DA_IN [8:0], el cual será capturado cuando el dato sea válido, DA_VALID. La señal G1 es la encargada de dar la señal correspondiente al PWM, la cual se genera a través del cálculo del duty. Además, podemos observar como por entradas también se encuentran las señales sincronizadas correspondientes a los interruptores de cambio de carga y de desconectar el controlador comentados anteriormente, y mediante los cuales se genera la salida G3 o establecemos un *duty* que será de 417, con el fin de obtener una salida de 5V. Este bloque es el encargado de mandarle la señal de control al bloque del conversor CONVST, mediante la cual se le indica al conversor que inicie una nueva conversión.

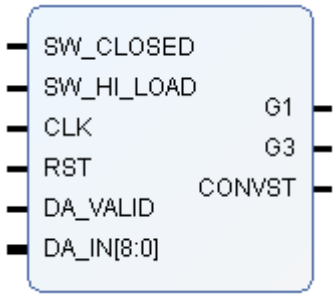


Figura 24

Los tres bloques restantes se utilizan para realizar la visualización de los resultados. En primer lugar, el bloque ui, Figura 25, es el encargado de recibir los datos de salida del conversor ya escalados procedentes del bloque ctl_adc. En este bloque se realizan las operaciones necesarias para adecuar ambos datos de entrada, para que su visualización sea correcta. La entrada SW_CHANNEL, es la encargada de seleccionar cual es el canal que se quiere visualizar a la salida.

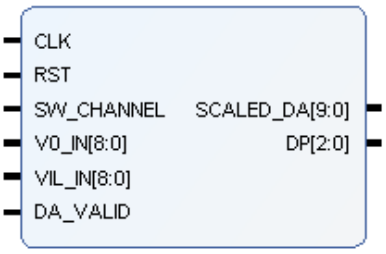


Figura 25

El siguiente bloque es el bin2bcd, Figura 26, el cual como su propio nombre indica es el encargado de convertir el resultado de binario a BCD.

Figura 26

Por último, queda el bloque display4x7, Figura 27, el cual es el encargado de realizar el control multiplexado de los visualizadores para representar SCALED_DA en tres visualizadores.

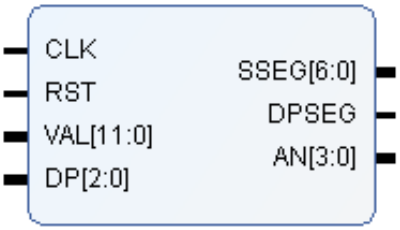


Figura 27

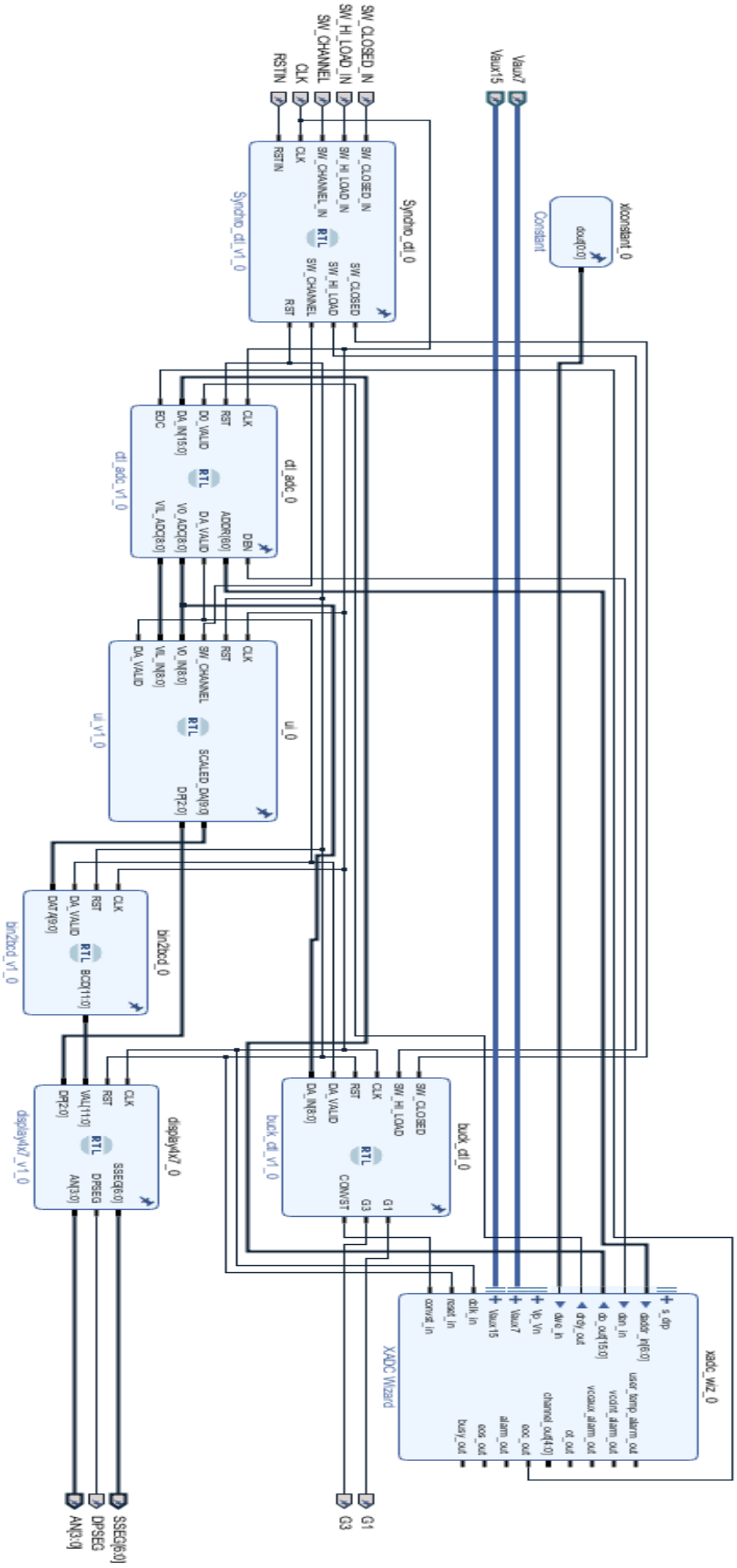

SW_CLOSED_IN
SW_HI_LOAD_IN
SW_CHANNEL
CLK
RSTIN
Vaux7
Vaux15
xlconstant_0
Constant
Synchro_ctl_0
Synchro_ctl_v1_0
ctl_adc_0
ctl_adc_v1_0
ui_0
ui_v1_0
bin2bcd_0
bin2bcd_v1_0
buck_ctl_0
buck_ctl_v1_0
displaybyNx7_0
displaybyNx7_v1_0
xlabc_wiz_0
XADC Wizard
AN[3:0]
DPSEG
SSEG[6:0]
G3
G1

Figura 28

5.2 Implementación de la simulación

En el presente apartado se presentan las bases de como se ha implementado en Vivado la simulación correspondiente para verificar el correcto funcionamiento del sistema.

Debido a la configuración interna del conversor analógico/digital, para la simulación de este solo admite entradas mediante un fichero de texto, lo cual nos supone un pequeño inconveniente. Como solución se ha decidido que, para la simulación y verificación del correcto funcionamiento del diseño, se van a simular dos situaciones.

Por una parte, se va a verificar el correcto funcionamiento y configuración del conversor analógico/digital mediante la simulación a través del fichero de texto. Por otra parte, se va a simular el correcto funcionamiento en lazo cerrado del sistema, para verificar el correcto diseño del controlador con el fin de obtener la tensión deseada a la salida. Esto se va a realizar mediante un fichero de entorno de test que realice la función de *Buck*, que vaya conectado al diagrama de bloques explicado en el apartado anterior, de tal manera que las salidas de éste sean las entradas del entorno de test y viceversa. En él se definen los valores de los parámetros del Buck, así como de las ecuaciones que gobiernan el sistema, estando estas condicionadas por la señal G1 proveniente del diagrama de bloques. En este diseño también se implementará un modelo del funcionamiento del conversor analógico/digital mediante VHDL no sintetizable.

Para esto, vamos a tener que modificar ligeramente nuestro diagrama de bloques. Las entradas Vaux15 y Vaux7 se utilizan para inicializar las entradas del conversor en el entorno de simulación, ya que esto es requerido por el fabricante. Como última entrada, DA_IN [8:0], tenemos el dato correspondiente a V_0 proveniente del fichero del entorno de test. En cuanto a las salidas, CONVST, es la encargada de mandar al modelo del *Buck* que calcule un nuevo valor para V_0 en función del valor de G. Este diagrama de bloques completo se puede observar en la Figura 29.

Para poder realizar la simulación se precisa realizar un fichero específico para ello. En él se definen los dos “componentes” presentes en el diseño. Por un lado, el componente correspondiente al diagrama de bloques comentado anteriormente y por otro lado el correspondiente al fichero que voy a utilizar como entorno de test para emular el comportamiento del Buck. En este fichero se realiza tanto la inicialización de los componentes como la declaración de señales las cuales servirán para “conectar “dichos componentes. También se les da valores a las entradas tales como el reloj, el reset y los interruptores. Para una información más detallada se recomienda consultar el código correspondiente estando esté presente en Anexo IV. Diseño en lenguaje VHDL.

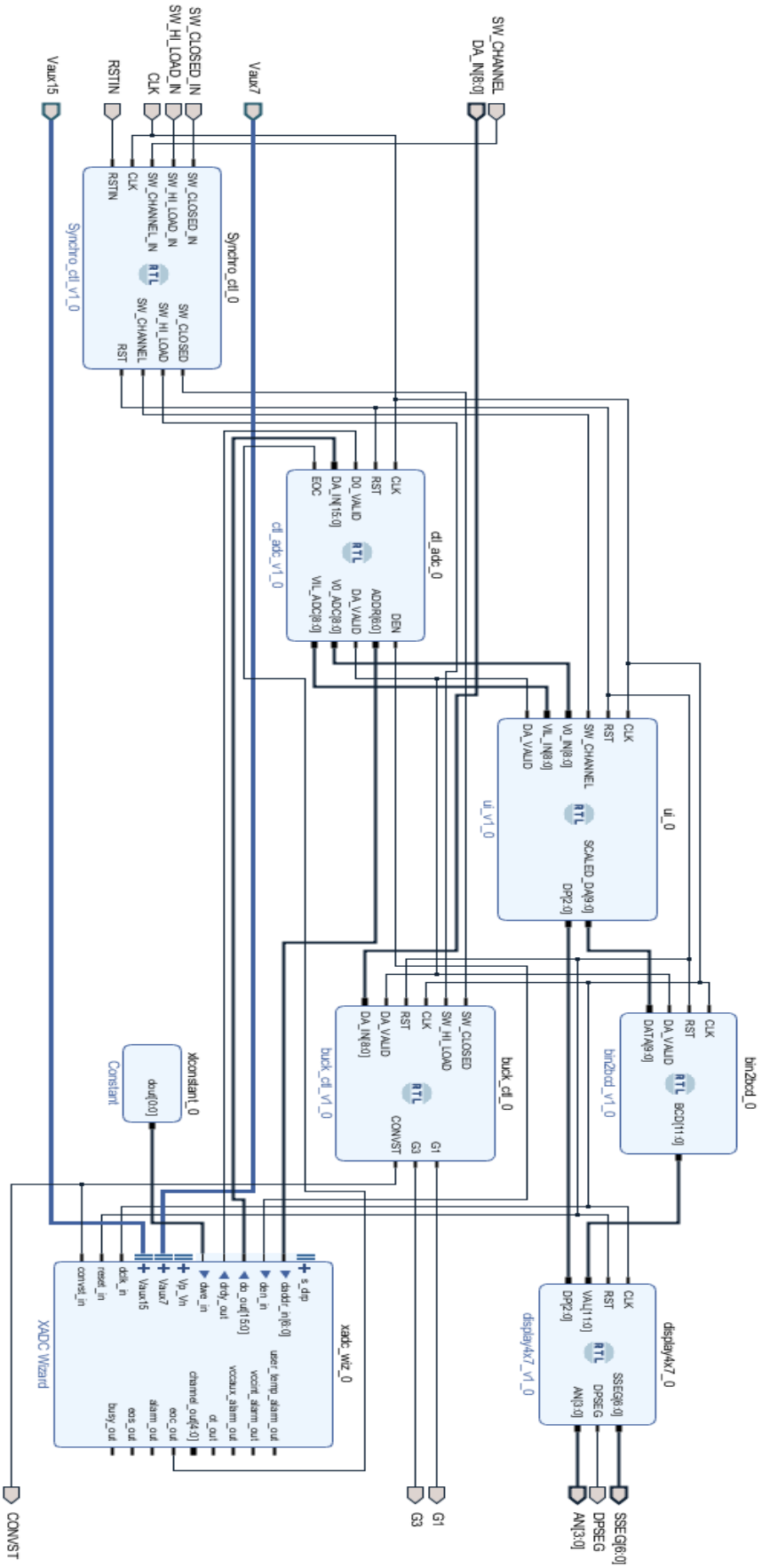

Vaux15
RSTIN
CLK
SW_HI_LOAD_IN
SW_CLOSED_IN
Vaux7
SW_CHANNEL
DA_IN[8:0]
Synchro_ctl_v1_0
Synchro_ctl_0
RSTIN
CLK
SW_CHANNEL_IN
SW_HI_LOAD_IN
SW_CLOSED_IN
RTL
SW_CHANNEL
SW_HI_LOAD
SW_CLOSED
RST
ctl_adc_v1_0
ctl_adc_0
EOC
DA_IN[15:0]
D0_VALID
RST
CLK
RTL
VIL_ADC[8:0]
V0_ADC[8:0]
DA_VALID
ADDR[6:0]
DEN
ui_v1_0
ui_0
DA_VALID
VIL_IN[8:0]
V0_IN[8:0]
SW_CHANNEL
RST
CLK
RTL
DP[2:0]
SCALED_DA[9:0]
buck_ctl_v1_0
buck_ctl_0
DA_VALID
DA_IN[8:0]
RST
CLK
SW_HI_LOAD
SW_CLOSED
RTL
CONVST
G3
G1
bin2bcd_v1_0
bin2bcd_0
DATA[9:0]
DA_VALID
RST
CLK
RTL
BCD[11:0]
Constant
xlconstant_0
dout[0:0]
XADC Wizard
xadc_wiz_0
Vaux15
Vaux7
Vp_Vn
dclk_in
reset_in
convst_in
s_dclk
daddr_in[6:0]
den_in
do_out[15:0]
drdy_out
dwe_in
vccaux_alarm_out
vccint_alarm_out
user_temp_alarm_out
ot_out
channel_out[4:0]
eoc_out
alarm_out
eos_out
busy_out
display4x7_v1_0
display4x7_0
DP[2:0]
VAL[11:0]
RST
CLK
RTL
DPSEG
AN[3:0]
SSEG[6:0]
CONVST
G3
G1
AN[3:0]
DPSEG
SSEG[6:0]

Figura 29

5.3 Utilización de recursos

En el resumen del proyecto, Project Summary, que la herramienta Vivado nos facilita una vez hemos realizado la síntesis y la implementación de nuestro proyecto, podemos observar los recursos que se han utilizado. En las dos siguientes imágenes se va a poder observar claramente.

Resource	Utilization	Available	Utilization %
LUT	185	20800	0.89
FF	261	41600	0.63
DSP	2	90	2.22
IO	34	106	32.08
BUFG	1	32	3.13

Figura 30

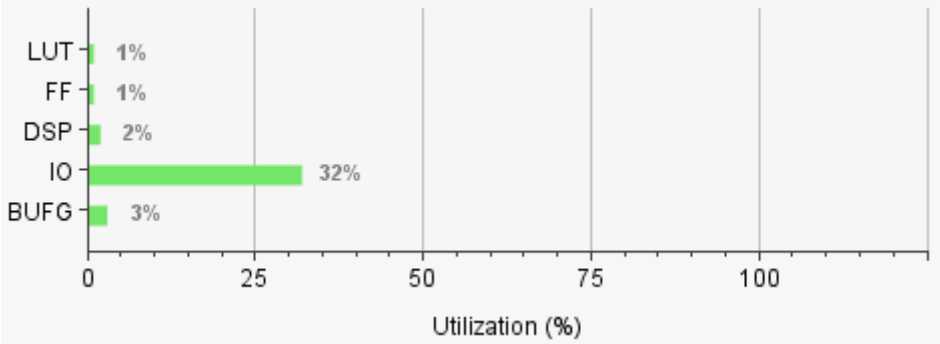


Figura 31

6. Resultados

En el presente apartado se van a adjuntar los diferentes resultados obtenidos tanto en simulación como experimentalmente, comparando estos entre sí, y comprobando si se cumplen los requerimientos especificados en este trabajo fin de grado.

En primer lugar, vamos a ver como es el transitorio de la tensión de entrada hasta llegar al régimen permanente, llegando esta tensión al valor especificado de 5V. La Figura 32 se corresponde con la simulación en VHDL, mientras que la Figura 33 se corresponde con la medida experimental medida con el osciloscopio.

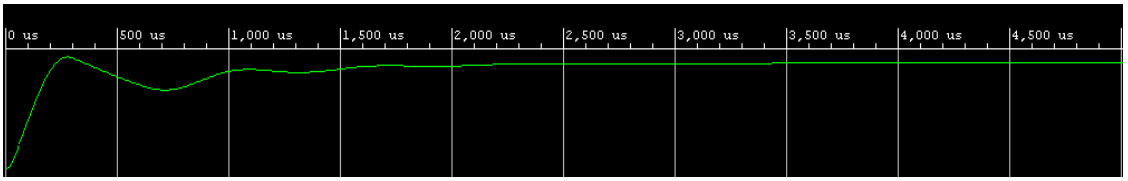


Figura 32

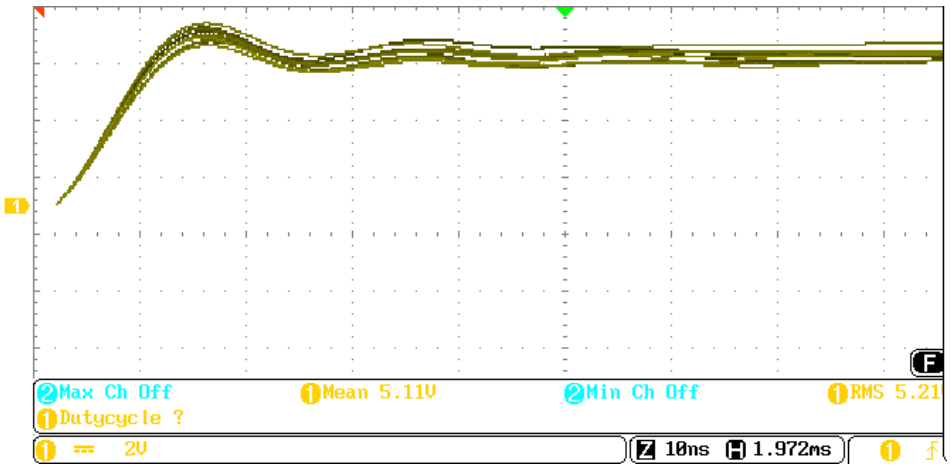


Figura 33

Se observa un tiempo de respuesta en torno a 2.25ms, siendo ambas Figuras muy similares entre sí. A continuación, vamos a ver el duty correspondiente cuando la salida es de 5V.

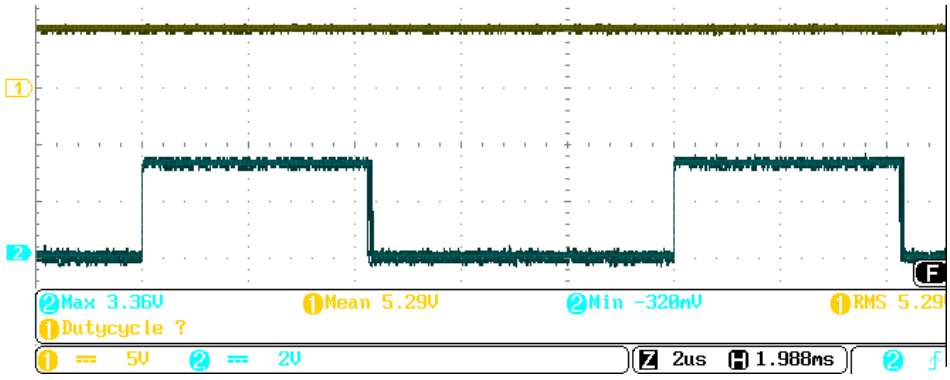


Figura 34

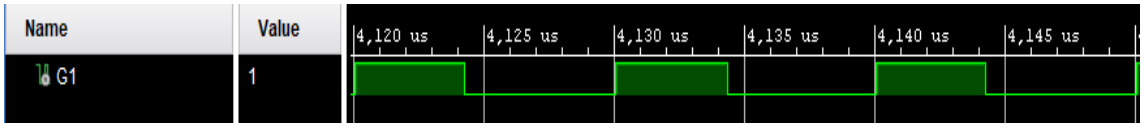


Figura 35

Se puede ver cómo, para 5V a la salida, señal color verdoso, de obtiene un *duty* en torno al 42%, señal azul, del tiempo de ciclo, el cual corresponde con lo calculado anteriormente de forma analítica.

También se han realizado pruebas cuando se realizan cambios de carga a la salida del convertidor En las próximas cuatro Figuras la señal morada es la señal de control G3, mientras que la señal de color verdoso es la tensión de salida.

Para un cambio de carga de 22 a 11 Ω:

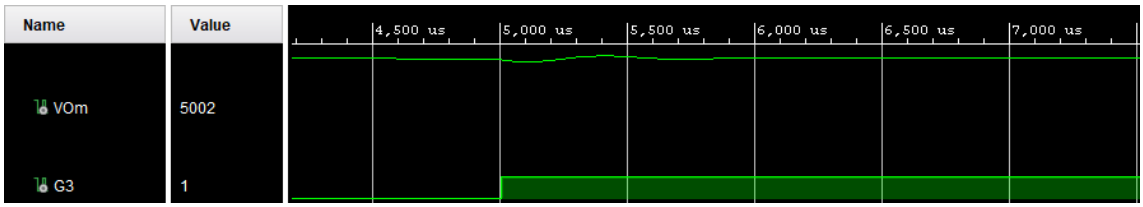


Figura 36

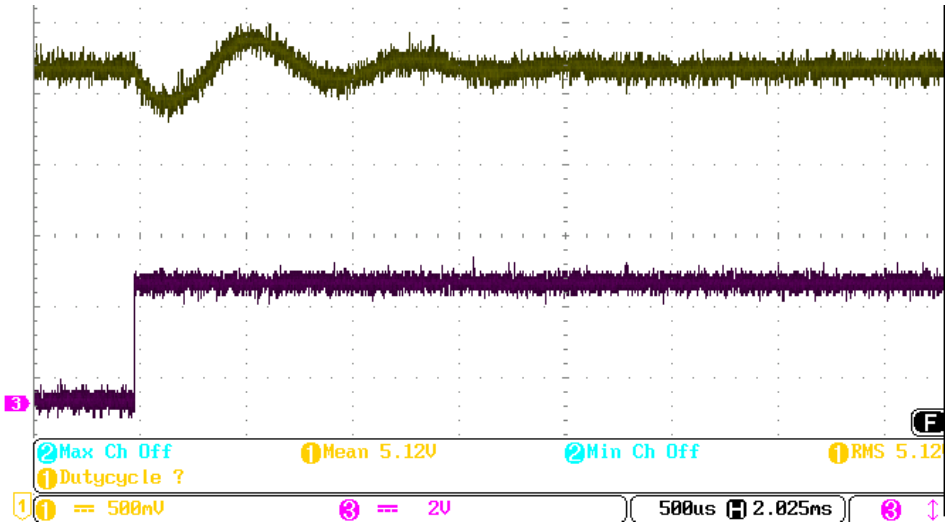


Figura 37

Para un cambio de carga de 11 a 22 Ω:

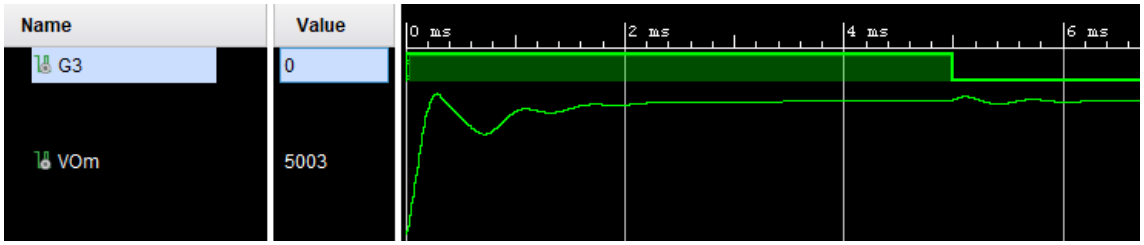


Figura 38

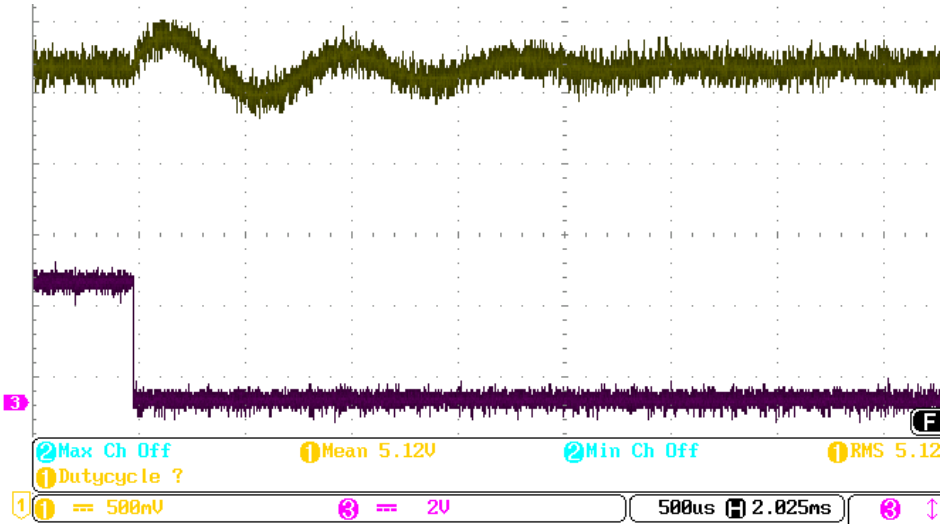


Figura 39

En ambos cambios de carga se puede observar un pequeño transitorio hasta alcanzar de nuevo el régimen permanente, siendo este transitorio en torno a 1ms.

Con el fin de comprobar el correcto disparo de los transistores Q1 y Q2, ambos controlados por el driver LM5104M, se han realizado la medida experimental de las tensiones de puerta de ambos transistores para verificarlo.

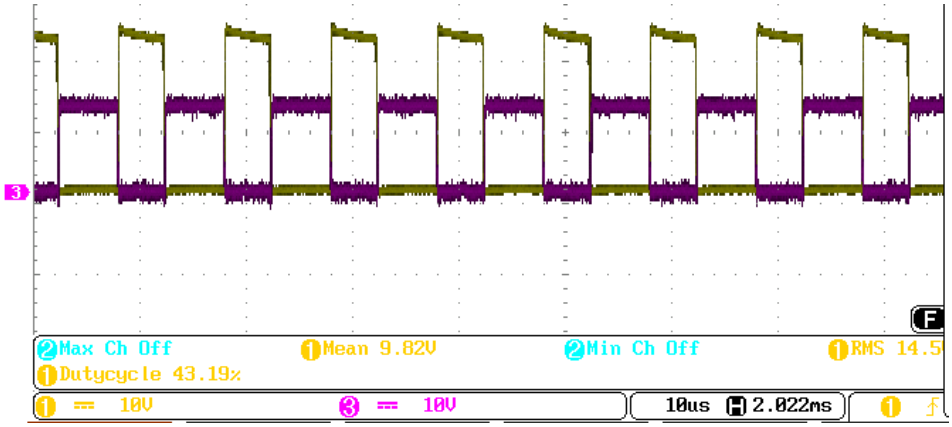


Figura 40

Donde podemos observar que al disparar el transistor Q1, señal de color verdoso, este llega a una tensión el doble que Q2, señal de color morado, y esto es debido al condensador de *Bootstrap*. También se puede ver como ambas señales son complementarias. Además de ser complementarias, hay que asegurar que ambos transistores no conduzcan al mismo tiempo, sino que hay que asegurar un tiempo muerto tal y como se explica en 3.1 Selección de componentes. Para ello se va a hacer un zoom de la Figura anterior, cabe destacar que se ha modificado la escala de la señal de Q1, con el fin de que este tiempo muerto se observe claramente.

Figura 41

Este tiempo muerto es de 0.2µs, lo cual coincide con lo calculado analíticamente en Anexo III.5 Selección de la resistencia para el ajuste del tiempo muerto.

Además, en los displays presentes en la FPGA, somos capaces de visualizar tanto la tensión de salida como la corriente que circula por la bobina.

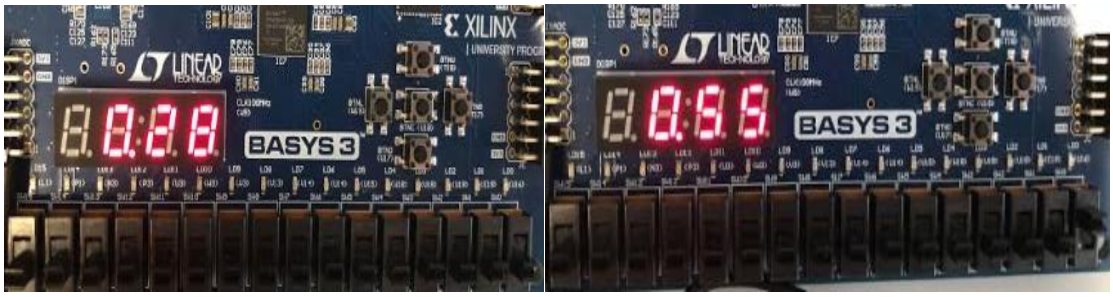


Figura 42

En la Figura anterior se puede observar la tensión que circula por la bobina, el caso de la derecha con una carga de 22 Ω, y en el caso de la izquierda con una carga de 11Ω. Una es doble de la otra como cabía esperar.

En cuanto al caso de la tensión de salida, cuando el controlador esta activo, la tensión de salida no sufre ninguna modificación al variar la carga, mientras que cuando el control no esta activado y le establecemos un *duty* predeterminado, al variar la carga esta tensión se ve modificada en 0.2V.

Por último, se simuló el comportamiento del conversor mediante un fichero de texto, para comprobar su correcta configuración:

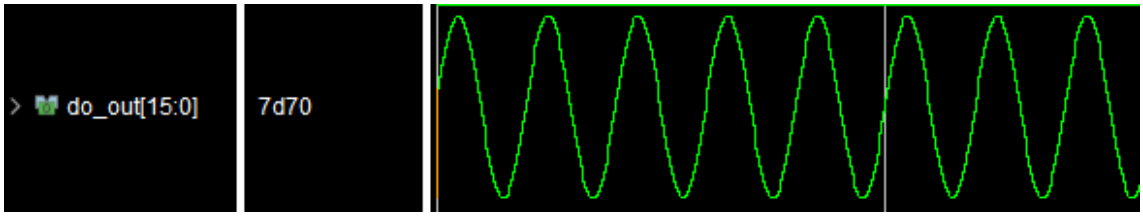


Figura 43

7. Conclusiones

A la finalización de este TFG se puede afirmar que:

- Se ha conseguido configurar correctamente los conversores analógicos/digitales presentes en la FPGA y con ello somos capaces de ahorrarnos módulos externos.

- Se ha conseguido obtener un modelo matemático que refleja el comportamiento dinámico del reductor de tensión *Buck* con la suficiente calidad como para desarrollar sobre él un control en modo tensión que se ha demostrado efectivo.

- Se ha calculado un regulador capaz de conseguir un error de posición nulo en la tensión de salida y que es capaz de ofrecer un sistema en bucle cerrado estable para el rango de operación del prototipo construido para el objeto de este TFG.

- Se han realizado las pruebas correspondientes tanto en simulación como experimentalmente, para corroborar el correcto funcionamiento del sistema.

- Se ha realizado una placa de circuito impreso con un Buck síncrono, la cual se ha demostrado que funciona correctamente a través de las pruebas experimentales realizadas.

A la vista de las conclusiones obtenidas, se puede afirmar que los objetivos planteados en 1.2 Objetivos y alcance, se han cumplido satisfactoriamente.

Referencias

[1] Erickson, R. W., & Maksimovic, D. (2007). *Fundamentals of power electronics*. Springer Science & Business Media.

[2] Digilent. Basys 3. Artix-7 FPGA Trainer Board. https://reference.digilentinc.com/reference/programmable-logic/basys-3/start, última vez visitada el 26/06/2019

[3] Hart, D. W. (2001). *Electrónica de potencia*. Pearson Educación, SA.

[4] Mohan, N., & Undeland, T. M. (2007). *Power electronics: converters, applications, and design*. John Wiley & Sons.

[5] Cathal Murphy. (2014). Driving the Xilinx Analog-to-Digital Converter. Xilinx, XAPP795 (v1.1). https://www.xilinx.com/support/documentation/application_notes/xapp795-driving-xadc.pdf, última vez visitada el 26/06/2019

[6] Franklin, P., & Powell, J. D. (2006). Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson Prentice Hall, New Jersey, 4, 2.

[7] Barragán, L. A. y Artigas, J. I. (2018). TEMA 3: Implementación sistemas LTI discretos en FPGA. *Grado en Ingeniería Electrónica y Automática*. Universidad de Zaragoza, IEC.

[8] Digilent. (2018). 7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter. *User Guide*. Xilix. https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf , última vez visitada el 26/06/2019

[9] Digilent. (2017). Basys 3™ FPGA Board Reference Manual. Xilinx. https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual, última vez visitada el 26/06/2019

[10] Adrián Marco Artigas (2016). Trabajo Fin de Máster. Alternativas de implementación de un controlador digital en FPGA para un convertidor Buck en prácticas de laboratorio https://zaguan.unizar.es/record/60492/files/?ln=es, última vez visitada el 26/06/2019

Anexo I. Obtención de la función de transferencia en pequeña señal del *Buck*

En el presente anexo se presenta la obtención de la función de transferencia en pequeña señal del convertidor *Buck*. Para ello voy a realizar un análisis del presente circuito:

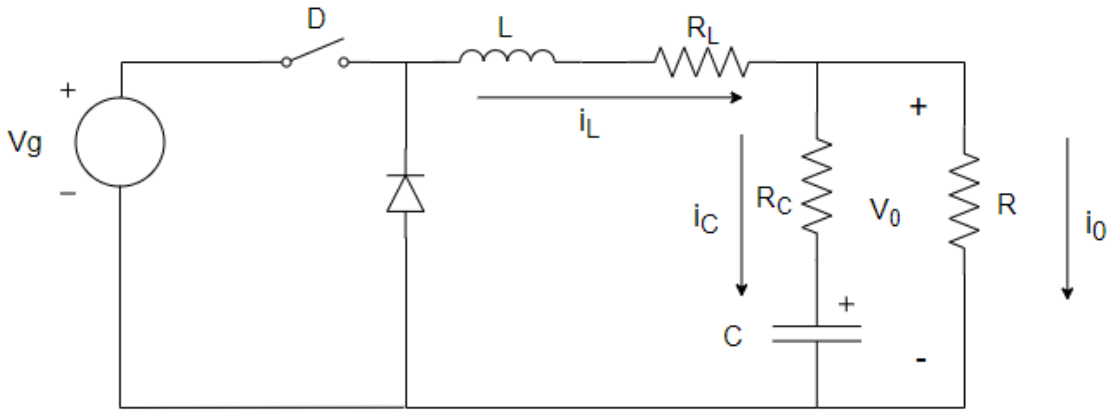


Figura 44

Nos disponemos a realizar un análisis en corriente:

$$i_L = i_C + i_0$$

$$\frac{V_g D(w) - V_0(w)}{jwL + R_L} = \frac{V_0(w)}{\frac{1}{jwC} + R_C} + \frac{V_0(w)}{R}$$

Donde V_g D(w) representa la excitación senoidal de pequeña señal correspondiente a la tensión de entrada reducida por el ciclo de servicio.

$$\frac{V_g D(w)}{jwL + R_L} = \frac{V_0(w)}{jwL + R_L} + \frac{V_0(w)}{\frac{1}{jwC} + R_C} + \frac{V_0(w)}{R}$$

$$\frac{V_g D(w)}{jwL + R_L} = V_0(w)\left(\frac{1}{jwL + R_L} + \frac{1}{\frac{1}{jwC} + R_C} + \frac{1}{R}\right)$$

$$\frac{V_g D(w)}{jwL + R_L} = V_0(w)\,\frac{\left(\frac{1}{jwC} + R_C\right)R + (jwL + R_L)R + (jwL + R_L)\left(\frac{1}{jwC} + R_C\right)}{(jwL + R_L)\left(\frac{1}{jwC} + R_C\right)R}$$

$$V_g D(w) = V_0(w)\,\frac{\left(\frac{1}{jwC} + R_C\right)R + (jwL + R_L)R + (jwL + R_L)\left(\frac{1}{jwC} + R_C\right)}{\left(\frac{1}{jwC} + R_C\right)R}$$

$$\frac{V_0(w)}{D(w)} = V_g \frac{\left(\frac{1}{jwC} + R_C\right)R}{\left(\frac{1}{jwC} + R_C + jwL + R_L\right)R + (jwL + R_L)\left(\frac{1}{jwC} + R_C\right)}$$

$$\frac{V_0(w)}{D(w)} = V_g \frac{\left(\frac{1 + R_C\, jwC}{jwC}\right)R}{\left(\frac{1 + R_C jwC + jwLjwC + R_L jwC}{jwC}\right)R + (jwL + R_L)\left(\frac{1 + R_C jwC}{jwC}\right)}$$

$$\frac{V_0(w)}{D(w)} = V_g \frac{(1 + R_C\, jwC)R}{(1 + R_C jwC + jwLjwC + R_L jwC)R + (jwL + R_L)(1 + R_C jwC)}$$

Pasando al campo transformado de Laplace:

$$\frac{V_0(s)}{D(s)} = V_g \frac{(1 + R_C\, Cs)R}{(1 + R_C Cs + LCs^2 + R_L C)R + (Ls + R_L)(1 + R_C Cs)}$$

Reagrupando:

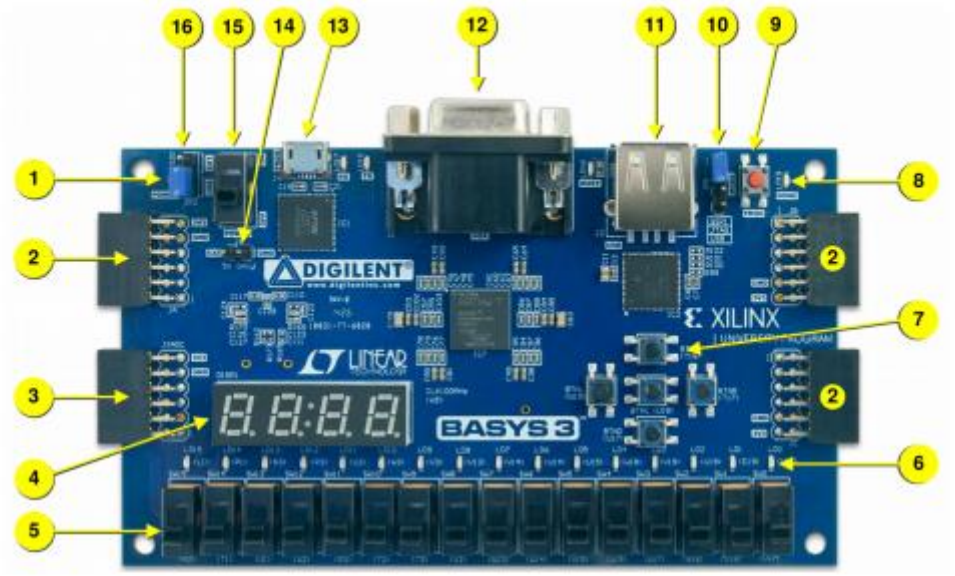
$$\frac{V_0(s)}{D(s)} = V_g \frac{(1 + R_C\, Cs)R}{(R + R_L) + (R\, R_C C + R\, R_L C + R_L\, R_C C + L)s + (R\, L\, C + L\, R_C\, C)s^2}$$

Operando:

$$\frac{V_0(s)}{D(s)} = V_g \frac{\frac{R}{(R + R_L)}(1 + R_C\, Cs)}{1 + \left(R_C C + \frac{R\, R_L}{(R + R_L)}C + \frac{L}{(R + R_L)}\right)s + \left(L\, C + \frac{R + R_C}{(R + R_L)}\right)s^2}$$

Anexo II. Configuración del conversor analógico/digital interno de la FPGA

Se procede a explicar como se ha realizado la configuración detallada de los conversores analógicos/digitales internos de la FPGA, adicionalmente al final del presente anexo se presenta los resultados obtenidos tanto en simulación como experimentales con el fin de justificar la correcta configuración y funcionamiento de estos. El uso de estos conversores se realiza con la finalidad de no tener que usar otros módulos externos. Además, debido a su disposición y modo de conexión mediante un conector, nos permite realizar una conexión robusta y estable con la placa de circuito impreso. Las bases de la explicación del funcionamiento del conversor han sido sacadas de [8].



	Component Description		Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG micro USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Figura 45

En primer lugar, como podemos observar la Figura anterior, extraída de [9], el número 3 se corresponde con el puerto Pmod correspondiente al XADC. En la Figura siguiente, también extraída de [9], se ve más detalladamente la distribución de sus pines:

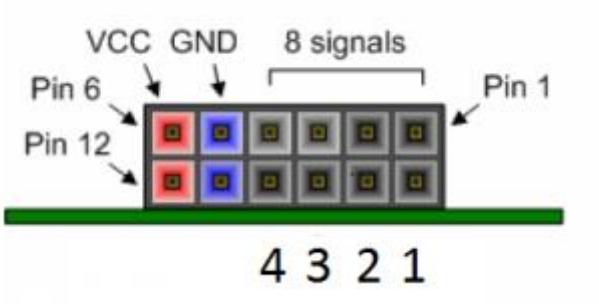


Figura 46

Este cuenta con 12 pines, de los cuales 4 son para alimentación, en nuestro caso los pines de VCC no se van a utilizar, pero los pines de GND van a ser utilizados para conectar la masa de la FPGA con la masa de la placa de circuito impreso.

Los 8 pines restantes se pueden utilizar configurar como entradas analógicas o como salidas digitales. En este proyecto se van a utilizar 4 de ellos como entradas analógicas, para monitorizar las tensiones V_0 e V_{IL}, de los 4 restantes 2 los voy a utilizar para generar las señales digitales de control G1 y G3. Los dos restantes quedarán inutilizados.

La FPGA consta de dos conversores analógico/digital internos de 12 bits. El diagrama de bloques es el siguiente, Figura extraída de [8]:

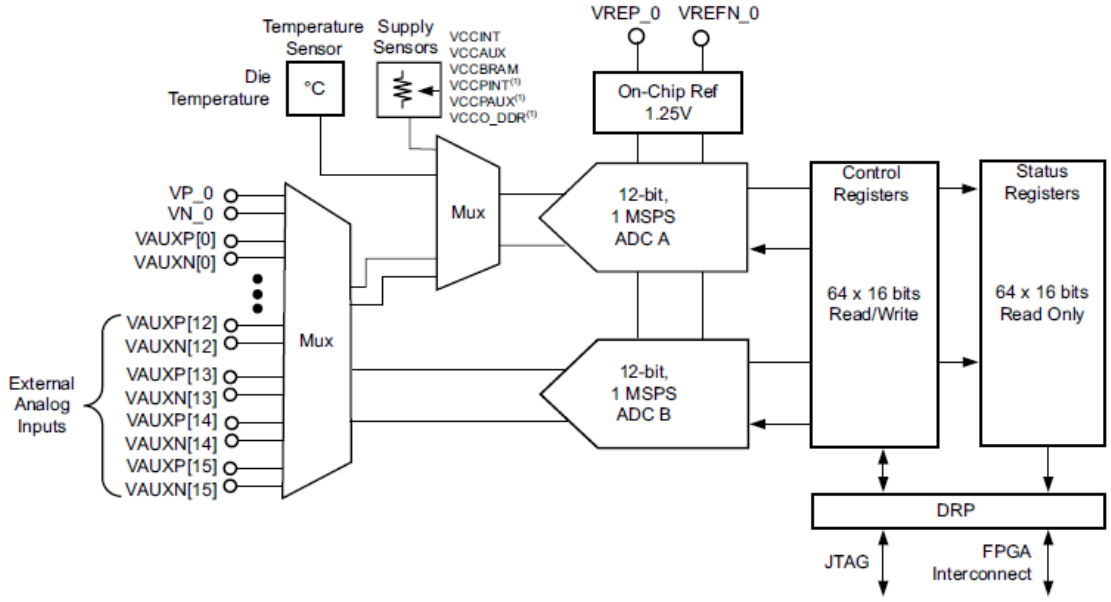


Figura 47

Consta de hasta 16 entradas analógicas, las cuales pueden ser configuradas también como entada/salida digital.

La conversión queda almacenada en unos registros llamados registros de estado, a los cuales se puede acceder a través de la FPGA usando el puerto DRP (Dynamic Reconfiguration Port).

Estos conversores tienen un rango de tensiones de entrada analógicas de 0 a 1V. Los vamos a configurar de modo que trabajen en modo unipolar, aunque se podrían configurar

en modo bipolar también. En este modo, las entradas analógicas del conversor producen un código máximo de escala de FFFh (12 bits) cuando la entrada es de 1V.

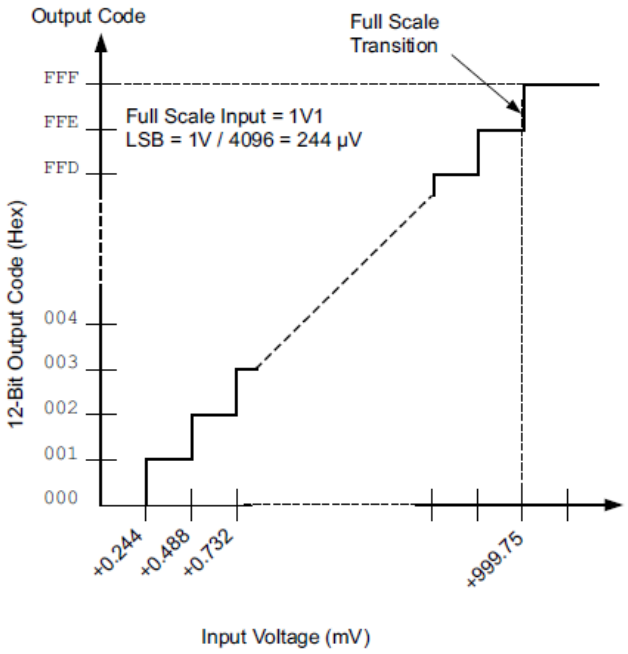


Figura 48

Como se puede observar en la Figura anterior, sacada de [8], el valor de 1 LSB es el siguiente:

$$LSB = \frac{1V}{2^{12}} = 244\mu V$$

Las entradas analógicas usan un sistema diferencial de muestreo para reducir los efectos en modo común de ruido de señal. Estas señales de ruido pueden de 100mV o más, lo que se corresponde con cientos de LSBs, por lo que la medida se vería distorsionada considerablemente.

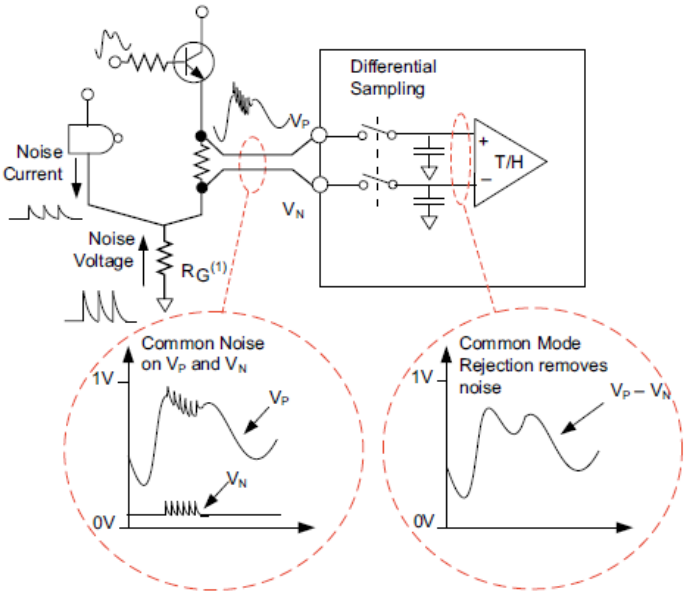


Figura 49

Estas entradas, además cuentan con un interruptor y un condensador de muestreo los cuales se utilizan para almacenar la tensión analógica requerida para la conversión como se puede observar en las Figuras 49 y 50 extraídas de [8]. Durante la adquisición el interruptor permanece cerrado y el condensador se carga a la tensión a la tensión de la entrada analógica, tal y como se muestra en la Figura. El tiempo necesario para la adquisición depende de la capacitancia del condensador de muestreo, la resistencia del multiplexor y de otras impedancias externas.

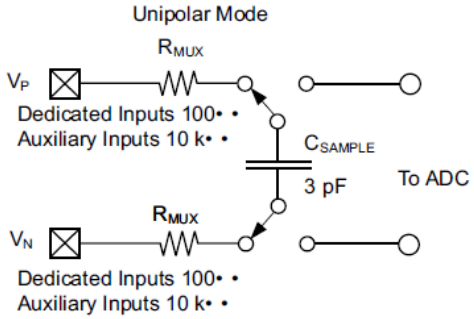


Figura 50

El tiempo de establecimiento de esta tensión viene dado por la siguiente ecuación:

$$t_{settle} = 9\, R_{MUX}\, C_{SAMPLE}$$

Esto es cierto para que la señal este dentro del LSB en el conversor de 12 bits

Hay 4 modos diferentes en los que podemos configurar estos conversores:

- Simoultaneos Selection
- Independent ADC
- Single Channel
- Channel Sequencer

En este proyecto puesto que se desea convertir ambos canales a la vez, el modo en el que vamos a configurar estos conversores es el Simoultaneos Selection. En este modo de operación las 8 primeras entradas analógicas van asignadas al ADC A y las 8 siguientes al ADC B.

A continuación, voy a mostrar cómo se realiza la configuración del módulo XADC en Vivado:

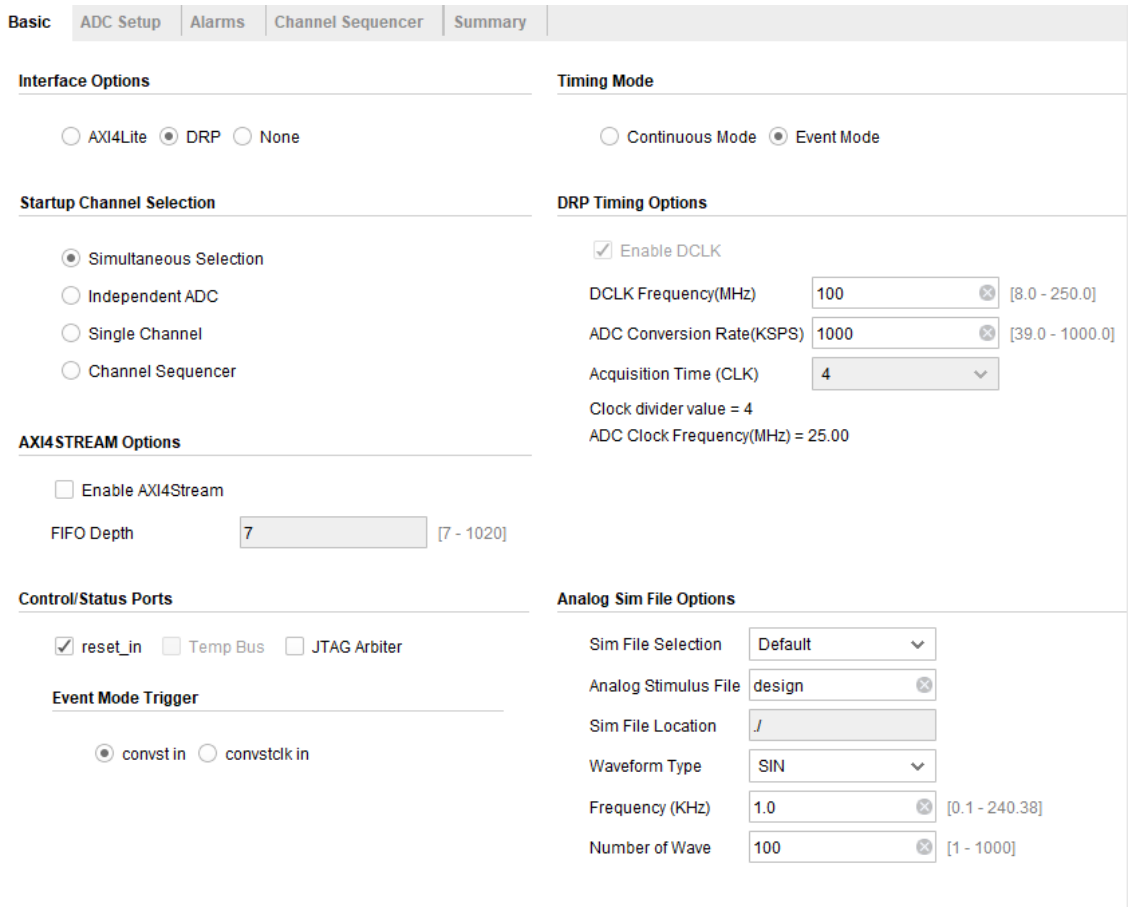


Figura 51

Este es el menú de configuración del XADC. La Figura anterior se corresponde con la pestaña Basic, tal y como se puede observar. Dentro de ella podemos seleccionar distintas formas de configuración:

- Interface Options: como ya había comentado anteriormente, el modo de hacer la interfaz va a ser mediante el DRP.
- Timing Mode: puesto que debido a la aplicación que se va a implementar requiere que la conversión se lance en un momento determinado y no continuamente, el modo seleccionado es Event Mode, mediante el cual controlaremos el disparo de inicio de conversión. Como se puede observar en la parte inferior izquierda de la Figura 51 este Event Mode Trigger será controlado mediante la señal convst in.
- DRP Timing Options: en este apartado voy a configurar el reloj a una frecuencia de 100MHz, para que sea la misma que a la que trabaja la FPGA.
- Analog Sim File Options: en este apartado se genera un fichero de texto para la simulación. He elegido que se trate de una señal analógica de frecuencia 1KHz.

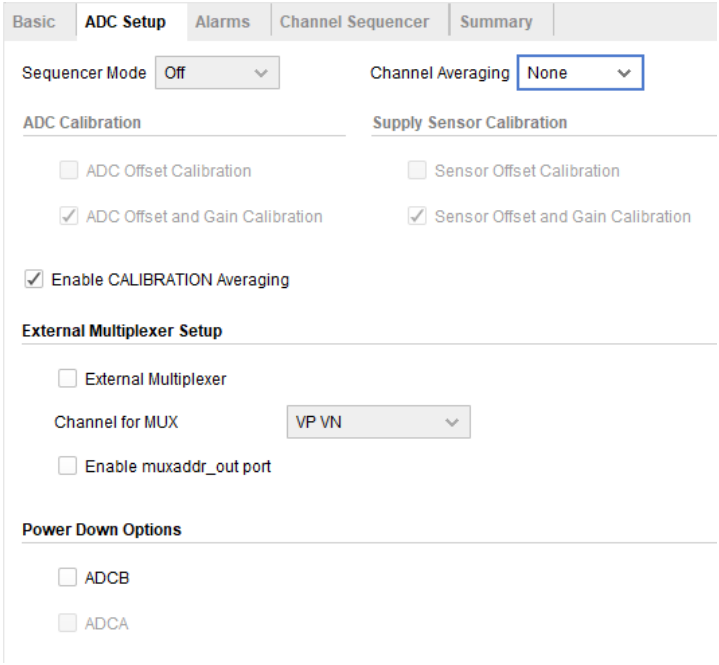


Figura 52

En la segunda pestaña del menú de configuración, mostrada en la Figura anterior, la única modificación que he realizado respecto a la configuración por defecto ha sido habilitar el Enable CALIBRATION Averaging, el cual se usa para realizar la calibración de la media, como su propio nombre indica.

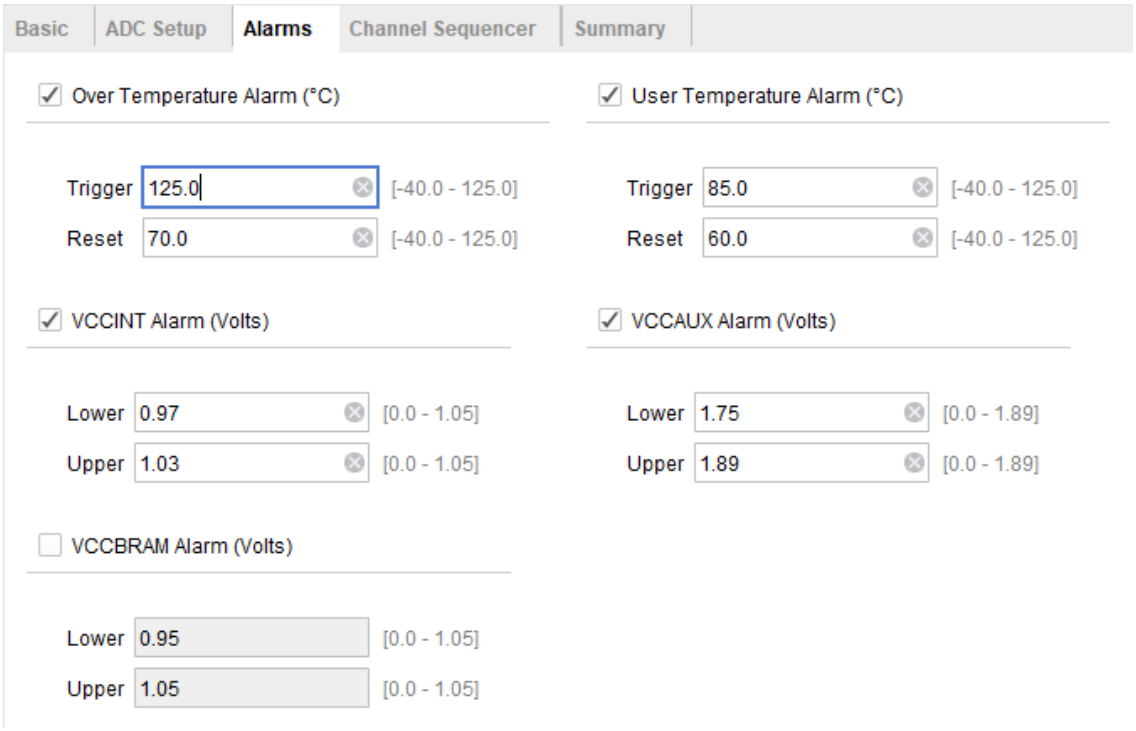


Figura 53

En la tercera pestaña dedicada a las alarmas, Figura 53, no se ha modificado ningún valor, aunque estos podrían ser modificados dependiendo de la aplicación a implementar.

En la cuarta pestaña, Channel Sequencer, Figura 54, se seleccionan aquellos canales que se quieran utilizar. En nuestro caso se van a utilizar dos canales, uno para V_0 y otro para V_{IL}. De acuerdo con la documentación del fabricante de la placa Basys3, para el modo de operación en el que estamos configurando los conversores, los canales que se pueden utilizar son 6, 7, 14 y 15. En este caso he elegido el 6 y el 14 para realizar la demostración, nótese que cuando se selecciona el canal 6, el canal 14 se marca por defecto y esto es debido a lo comentado anteriormente, que los 8 primeros canales van al ADC A y los 8 últimos al ADC B, por eso se selecciona automáticamente el 14

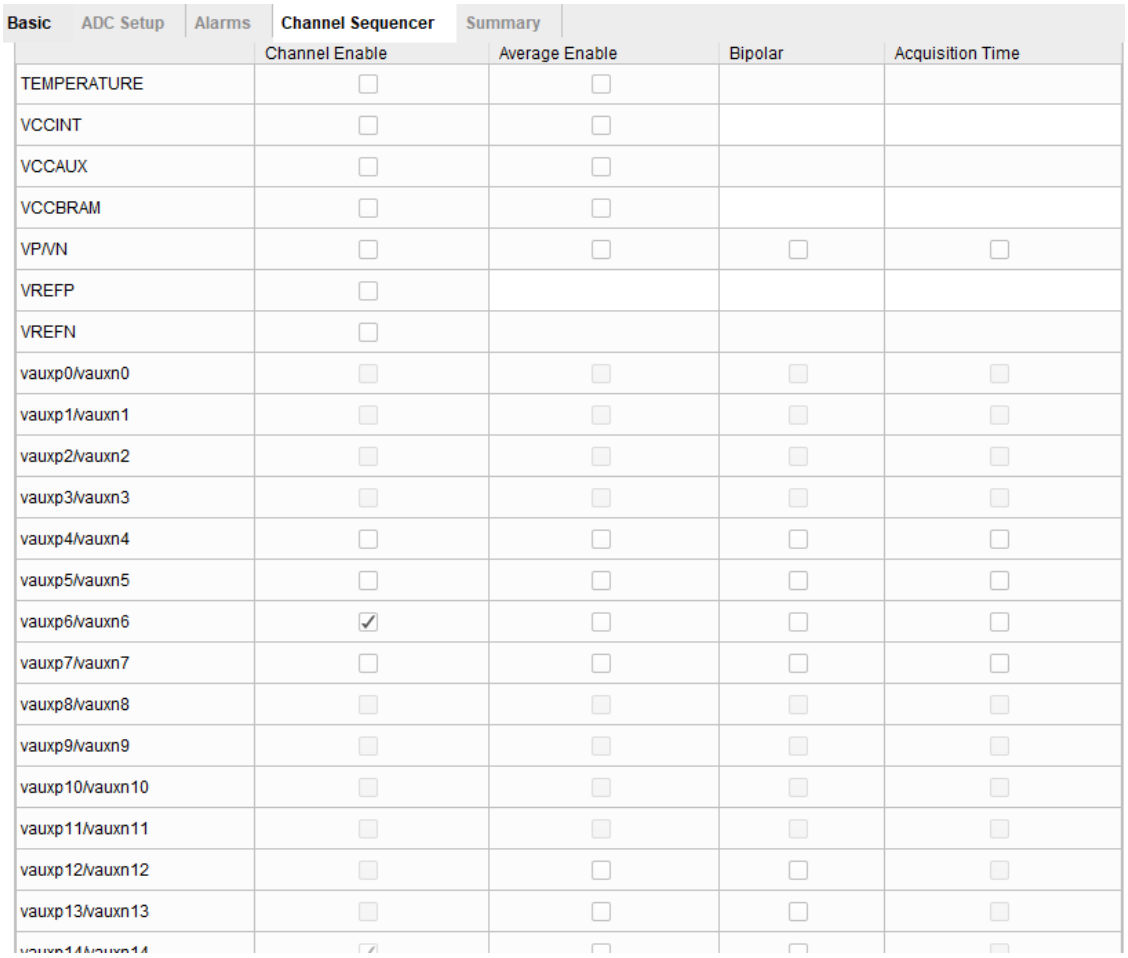


Figura 54

.

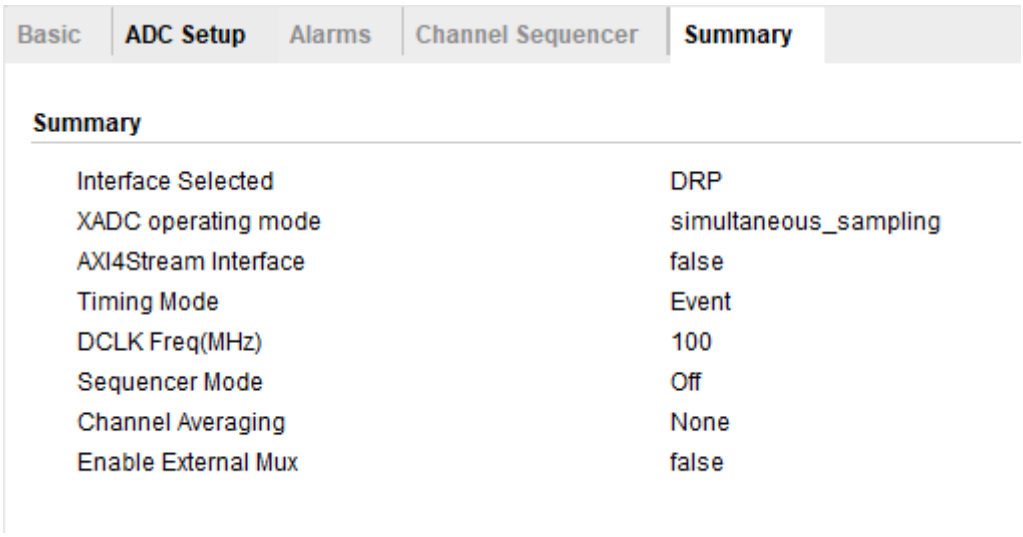


Figura 55

En la última pestaña del menú sale el resumen de la configuración seleccionada.

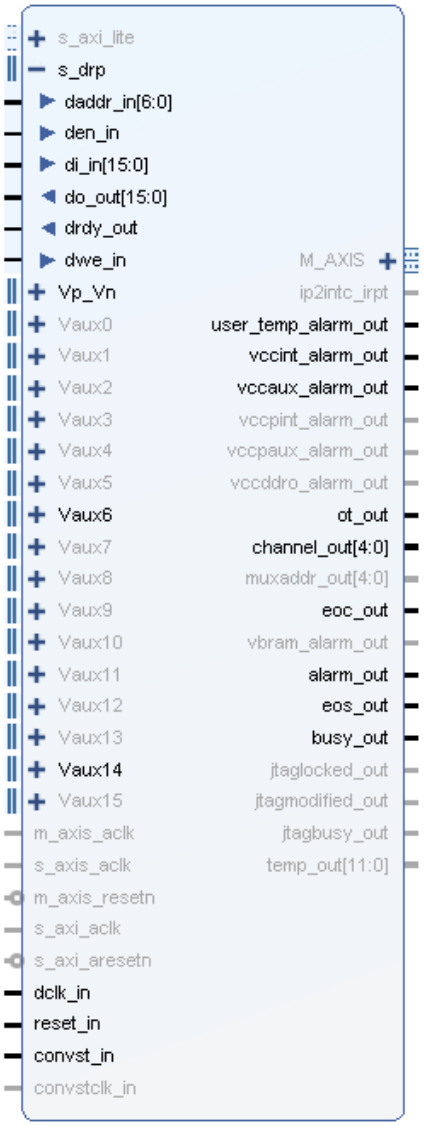


Figura 56

En la anterior Figura vemos como ha quedado nuestro bloque XADC después de la configuración. Los puertos más significativos son los siguientes:

- daddr_in [6:0]: mediante este puerto de entrada seleccionaremos el canal del cual se quiera obtener la salida.
- do_out [15:0]: puerto de salida en el cual se obtiene el resultado de la conversión del canal seleccionado.
- drdy_out: puerto que nos indica cuando el dato está disponible a la salida.
- den_in: puerto mediante el cual somos capaces de decirle al conversor, una vez haya terminado la conversión, que este nos saque un nuevo dato a la salida.
- dclk_in: puerto correspondiente al clock, el cual conectaré con el clock del sistema.
- reset_in: puerto correspondiente al reset, el cual conectaré con el reset del sistema.
- eoc_out: nos indica cuando el ciclo de la conversión ha finalizado, por lo que nos indica cuando es posible lanzar una nueva conversión.

- busy_out: nos indica cuando la conversión se está llevando a cabo.

Para comprender mejor el funcionamiento del conversor en éste modo de operación vamos a explicar el cronograma de este:

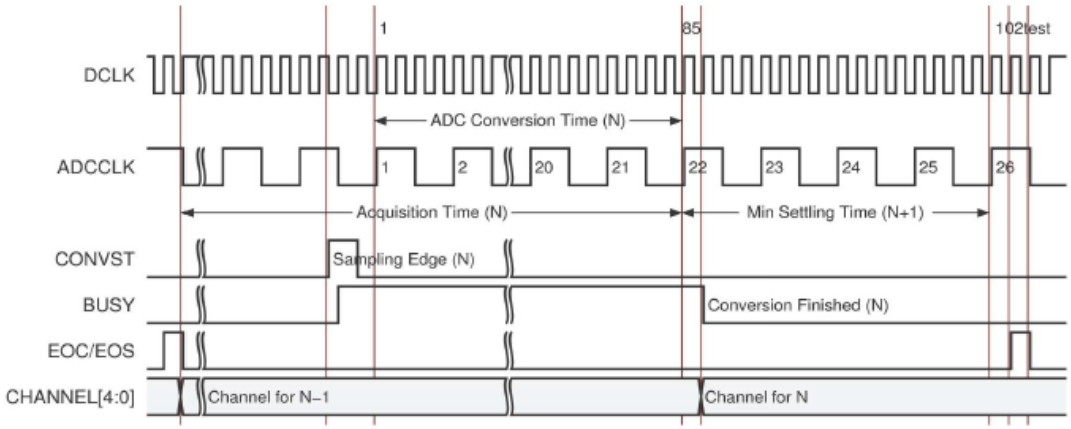


Figura 57

Como se puede observar en la Figura anterior extraída de [8], la conversión es lanzada con el flanco ascendente de CONVST, luego es conversor pasa a estar en el estado BUSY hasta que la conversión finaliza. En el cronograma no se observa, pero la simulación mostrada más adelante sí que se puede observar, se produciría un flanco ascendente en DRDY el cual nos indica que el dato ya está disponible a la salida pocos ciclos de reloj mas tarde del flanco descendente de EOC. Cabe destacar la diferencia entre DRDY y EOC. EOC nos indica cuando el ciclo de la conversión ha terminado y es posible lanzar una nueva. La señal DRDY la utilizaré como enable para los diferentes bloques de mi proyecto. La diferencia entre el fin de la conversión, flanco de bajada de BUSY, y el fin del ciclo de la conversión, flaco de subida de EOC, son 16 ciclos de DCLK o 4 de ADCCLK.

Para comprobar el correcto funcionamiento del bloque XADC, realicé un pequeño proyecto para verificar la correcta configuración y operación de éste.

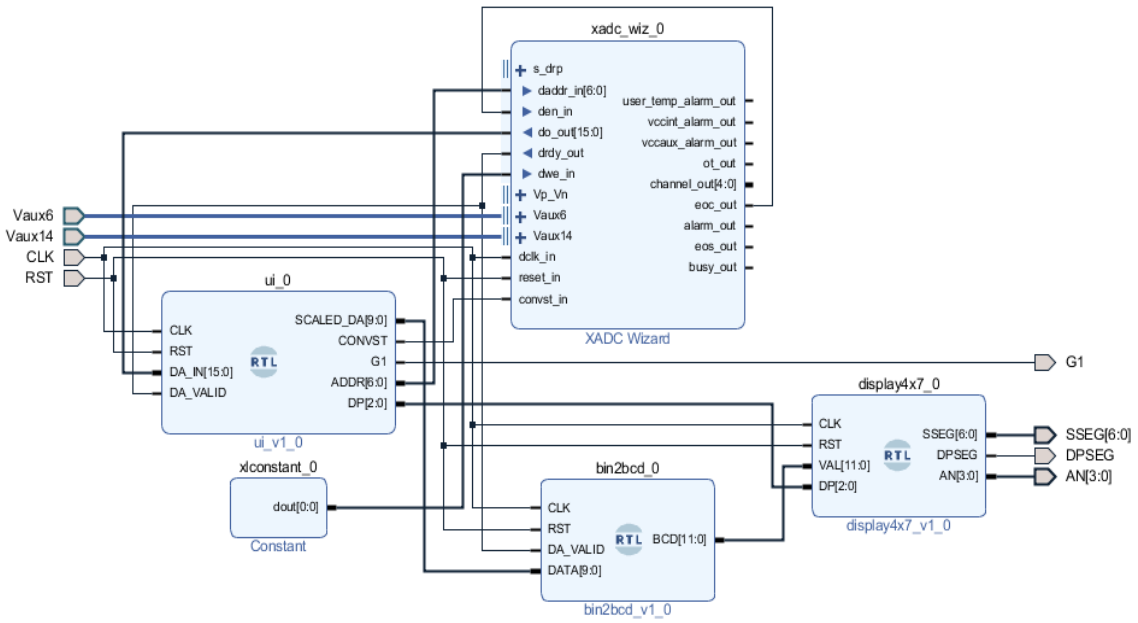


Figura 58

En la imagen anterior podemos observar el diagrama de bloques de este pequeño proyecto. A continuación, vamos a explicar por encima el funcionamiento de estos bloques.

- xadc_wiz_0: bloque correspondiente al XADC explicado explícitamente en apartados anteriores.
- display4x7_0: este bloque se encarga de convertir el dato en BCD para poder visualizarlo en los displays de 7 segmentos.
- bin2bcd: bloque encargado de realizar la conversión de binario a BCD.
- ui_0: este bloque es el encargado de recibir el resultado de la conversión procedente del bloque xadc_wiz_0 y realizar las operaciones necesarias para adecuar el dato al formato correspondiente para luego poderlo visualizar correctamente. También este bloque es el encargado de generar la señal CONVST, la cual determina el inicio de la conversión, en este caso se ha configurado para que sea lanzada cada 0.9ms. Por otro lado, también es el responsable de elegir cual es el canal que se visualiza. Adicionalmente genere una señal G1 con el fin de comprobar que podía usar los pines libres como salidas digitales.

Como he comentado anteriormente la entrada es una señal sinusoidal de frecuencia 1KHz.

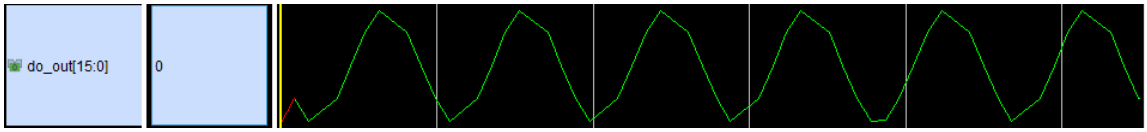


Figura 59

La salida correspondiente es la mostrada en la Figura anterior. Cabe destacar que, aunque el canal de salida sea conmutado entre el 6 y el 14, la salida no se ve afectada debido a que el fichero de texto que se genera por defecto para la simulación da el mismo valor para todos los canales.

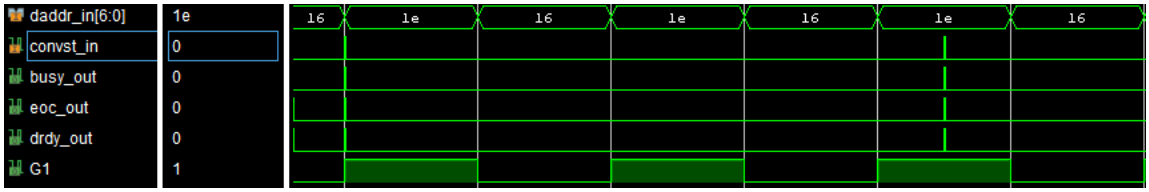


Figura 60

En la Figura anterior se pueden observar 4 señales. En primer lugar, la primera señal, daddr_in [6:0], nos muestra la conmutación entre los canales de salida. Las 4 señales siguientes las he comentado anteriormente, en esta imagen apenas se aprecia diferencia entre ellas por lo que vamos a verlas más detenidamente haciendo zoom. En cuanto a la última señal, es la señal G1 comentada anteriormente para verificar que podemos usar los pines restantes del XDAC para generar salidas digitales.

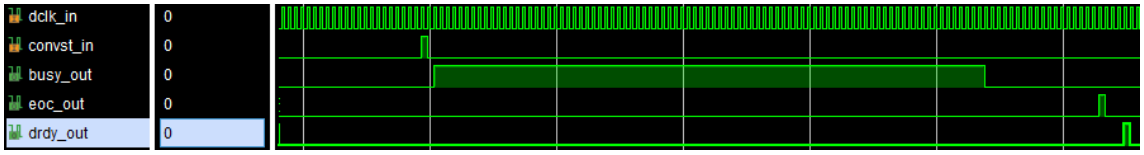


Figura 61

En esta Figura se corrobora lo explicado anteriormente haciendo referencia al cronograma proporcionado por el fabricante. Se observa claramente el inicio de la conversión con el flanco ascendente de convst_in. Posteriormente, se observa como la señal busy_out se pone a 1, determinando que la conversión se está llevando a cabo. Una vez esta señal pasa al estado bajo, significa que la conversión ha terminado, pero no con ello se puede lanzar una conversión, ya que esa es la función de eoc_out, la cual nos indica que una nueva conversión puede ser lanzada. Cabe destacar la diferencia de 16 ciclos de reloj entre el flanco descendente de busy_out y el flanco ascendente de eoc_out comentado anteriormente en este apartado. Por último, se observa la señal drdy_out, la cual es la encargada de avisarnos cuando el dato está disponible a la salida.

Se realizó una prueba experimental en la que a un canal le entraba una señal de 800mV, y al otro de 400mV, comprobando que en los displays nos mostraba estas tensiones en mV.

Anexo III. Cálculos

Anexo III.1 Cálculo de los parámetros más relevantes del *Buck*

En esta parte del anexo, vamos a justificar los valores de los principales parámetros de reductor *Buck* expuesto, para ello vamos a echar mano de las ecuaciones planteadas en el apartado de este trabajo fin de grado.

Definiendo la relación del servicio del *Buck* (*duty ratio*) como:

$$D = \frac{t_{ON}}{T_S}$$

Donde t_{ON} es el tiempo de conducción (*ON*) del dispositivo de potencia principal y T_S es el periodo de conmutación.

$$V_0 = \frac{1}{T_S}\int_0^{T_S} v_0 dt = \frac{V_d t_{ON}}{T_S} = DV_d$$

Sabiendo que la tensión de entrada del reductor V_g es de 12V y que la tensión de salida V_0 es de 5V:

$$D = \frac{V_0}{V_d} = \frac{5}{12} = 0.4166$$

$$\frac{\Delta I_L}{t_{ON}} = \frac{V_d - V_0}{L}$$

Siendo definido el rizado en la bobina como el porcentaje que representa la variación sobre la corriente total.

Puesto que este proyecto tiene un fin educativo y se realiza con el fin de que pueda ser utilizado en prácticas en años posteriores, se ha elegido un rizado de corriente por la bobina del 25% de la corriente media. Esto ha sido elegido con el fin de que este sea apreciable, para poder visualizarlo correctamente, pero quedando de esta manera muy alejado del límite en el que el sistema podría abandonar el MCC. De acuerdo con esta consideración, procedemos con el cálculo del valor de la bobina:

Tenemos dos situaciones posibles, y estas dependen de si MOSFET Q3 está activo o no, ya que tendremos una carga a la salida de 11Ω o de 22Ω. Para realizar los cálculos vamos a ponernos en el peor caso, que se da cuando la carga es pequeña, es decir 11Ω. Aunque la tensión de salida deseada es de 5V, hemos supuesto que la máxima será de 6V en un caso de que nuestro controlador no esté regulando correctamente. Esta tensión también habrá que tenerla en cuenta a la hora de realizar los cálculos para el caso más desfavorable. De acuerdo con el análisis realizado anteriormente:

$$I_0 = I_L = \frac{V_0}{R_0} = \frac{6V}{11\Omega} = 0.54\ A$$

$$\Delta I_L = 25\%\, I_0 = 0.13\ A$$

$$\Delta I_L = \frac{V_d - V_0}{L} t_{ON}$$

Teniendo en cuenta que nos tenemos que poner en el peor caso, V_0=6V, la relación de servicio pasa a ser 0.5 y sabiendo que T_S es 10µs:

$$D = \frac{t_{ON}}{T_S} \rightarrow t_{ON} = D\, T_S = 5\mu s$$

Por lo tanto:

$$\Delta I_L = \frac{V_d - V_0}{L} t_{ON}$$

$$0.13A = \frac{12V - 6V}{L} 5\mu s \rightarrow L \cong 230\mu H$$

Elegimos el valor comercial más cercano que es 220 μH

Para obtener una tensión continua de calidad aceptable aguas abajo del reductor hay que asegurar que el rizado que se produce en dicha tensión toma unos valores razonables.

Este rizado se puede observar en la Figura 62, extraída de [4]. Para simplificar el cálculo consideramos despreciable la resistencia parásita del condensador a colocar en la topología de reductor *Buck* y planteamos las siguientes ecuaciones:

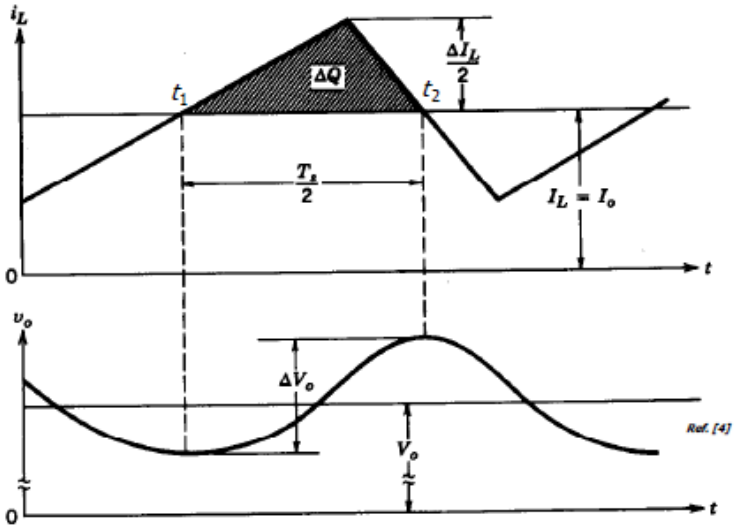


Figura 62

$$i_C = C\frac{dV_0}{dt} = i_L - i_0 \cong i_L - I_L$$

Siendo en la ecuación anterior i_C la corriente por la rama del condensador, i_0 la corriente que circula por la carga, i_L el valor absoluto del rizado de corriente en la bobina, e I_L el nivel de corriente continua que atraviesa la bobina

$$\Delta V_0 = \frac{1}{C}\int_{t1}^{t2} i_C dt \cong \frac{1}{C}\frac{1}{2}\frac{T_S}{2}\frac{\Delta I_L}{2} = \frac{T_S \Delta I_L}{8C}$$

Este incremento en la tensión de salida depende del tiempo de muestreo, ya prefijado, del rizado de la corriente por la bobina anteriormente comentado y del valor del condensador. Eligiendo un condensador de 100 µF Se obtiene un rizado de 1.625mV, el cual resulta despreciable con respecto a los 5V de salida deseados.

Anexo III.2 Cálculo del Regulador

Para el cálculo del regulador nos hemos apoyado en la herramienta Matlab. Lo primero que debemos de hacer es definir la frecuencia de corte adecuada, siguiendo las técnicas de control clásicas, se elige una frecuencia de corte 20 veces inferior a la frecuencia de conmutación. Adicionalmente, se ha elegido un margen de fase a la frecuencia de corte de 60º, el cual establece una sobre oscilación del 8%.

La función de transferencia $G_{vd}(s)$ en pequeña señal del *Buck* en modo continuo es la siguiente, donde R representa la resistencia de carga (R_0 o $R_0/2$) según el estado de G3, la obtención de esta función de transferencia se puede ver detalladamente en Anexo I. Obtención de la función de transferencia en pequeña señal del *Buck*.

$$G_{vd}(s) = \frac{V_0(s)}{D(s)} = \frac{V_g \frac{R}{R+R_L}(1+R_C C s)}{1 + \left(R_C C + \frac{R R_L}{R+R_L} C + \frac{L}{R+R_L}\right) s + LC\frac{R+R_C}{R+R_L} s^2}$$

Los parámetros del *Buck* son los siguientes:

- Tensión de entrada V_g=12V
- Tensión de salida V_0=5V
- L=220 µH
- C=110 µF
- R_c=0.13Ω
- R_L=0.16Ω
- R_0=22Ω
- f_{SW}=100KHz
- f_S=5KHz

Por lo tanto, la ecuación de planta queda:

$$G_{vd}(s) = \frac{0.0001538s + 11.83}{2.194e^{-8}s^2 + 4.848e^{-5}s + 1}$$

A continuación, se muestra el diagrama de bode correspondiente a la planta:

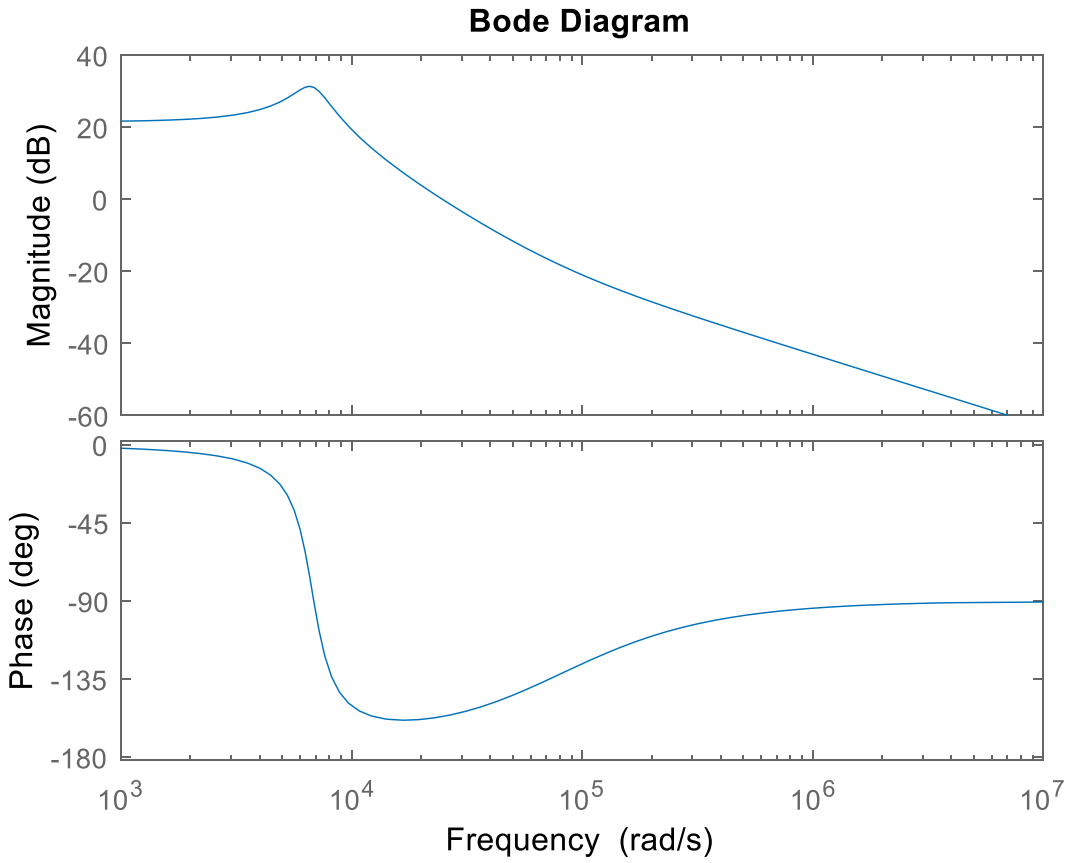


Figura 63

A esa frecuencia de corte podemos observar (ver Figura anterior) cómo el margen de fase (MF) es insuficiente para garantizar la estabilidad del sistema en bucle cerrado. Por tanto, el regulador que se plantea es el siguiente.

Para ello se parte de un controlador analógico C(s) con la siguiente función de transferencia (PI+PAF):

$$C(s) = K\frac{(s + z_1)(s + z_L)}{s(s + p_L)}$$

El controlador analógico se diseña para un margen de fase 67. 5º (7. 5º más que el deseado para incluir el efecto del retraso debido al PWM digital)

$$\emptyset_{DPWM} = DT_s f_c 360º = 7.5º$$

Primero calculo el PI:

$$PI(s) = K_{PI}\frac{(s+z_1)}{s}$$

$$w_C = 5000kHz = 31416\,rad/s$$

La frecuencia de corte del PI debe estar lo suficientemente alejada de manera que el compensador no afecte al MF a la frecuencia de corte. Se elige una frecuencia para el compensador PI diez veces menor que la frecuencia de corte:

El cero z_1 se coloca en $w_c/10$, es decir en 3142 rad/s. La K_{PI} la calcularé al final con el resto de K's del sistema para que el diagrama de Bode final corte en 0dB a la frecuencia de corte deseada.

Por lo tanto, el PI queda:

$$PI(s) = K_{PI}\frac{(s+3142)}{s}$$

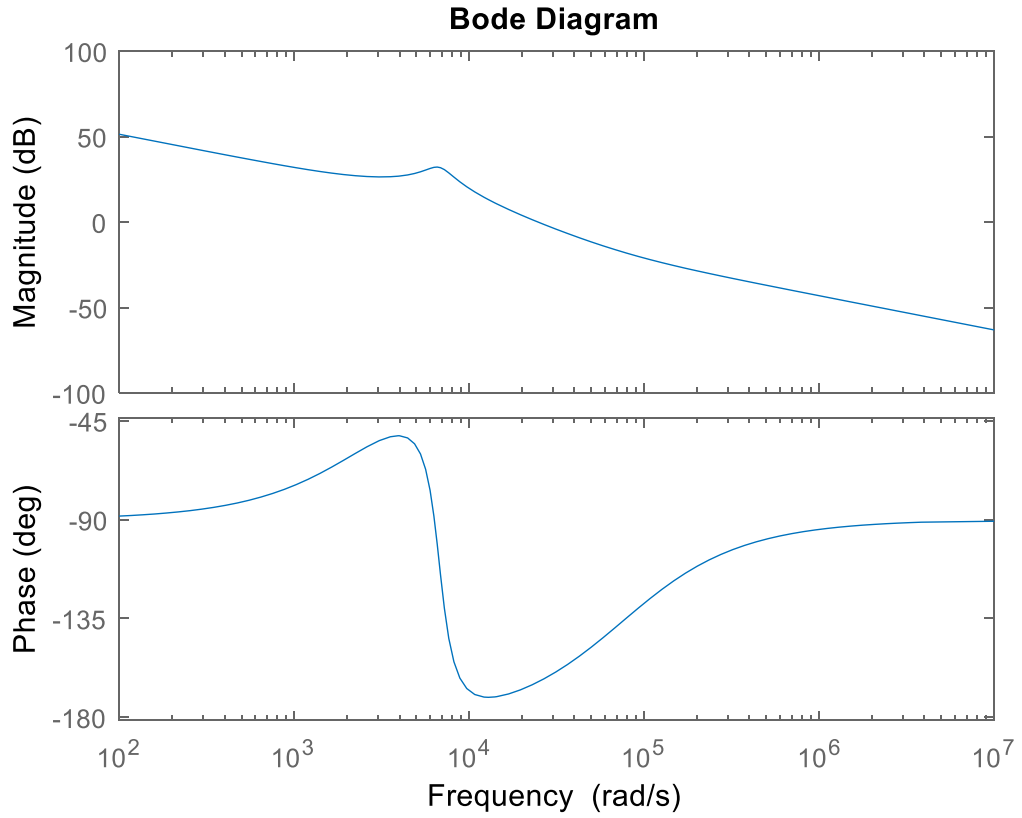


Figura 64

En la Figura anterior se muestra el diagrama de bode la planta añadiéndole el PI, se puede observar como el error de posición pasa a ser 0 debido a la entra del integrador del

PI (el diagrama de fases empieza en -90º y el de ganancia con una pendiente de -20dB por década).

Sin embargo, el margen de fase a la frecuencia de corte no es el deseado, ya que deseamos un margen de fase de 67. 5º y por ahora tenemos uno de 21º, fase del sistema a la frecuencia de corte deseada es de -159º, por ello la necesidad de añadir un regulador proporcional de avance de fase.

Cálculo de PAF:

$$PAF(s) = K_{PAF}\frac{(s+z_L)}{(s+p_L)}$$

Este se calcula de forma que su máximo desfase ocurra en w_C, es decir, que la media geométrica de z_L y p_L debe ser w_C. De esta manera y considerando ganancia unidad para el ADC y el PWM, se obtiene el siguiente controlador:

Dado el margen de fase actual y el margen de fase deseado hay que aportar 46. 5º

$$\alpha = \frac{1-sen\,\emptyset_{máx}}{1+sen\,\emptyset_{máx}} = 0.1591$$

Siendo $\emptyset_{máx} = 46.5º$

Sabiendo que $p_L = \alpha z_L$ y aplicando la condición de que la media geométrica debe ser w_C:

$$\sqrt{z_L p_L} = \sqrt{z_L \frac{z_L}{\alpha}} = 31416\ rad/s$$

$$z_L = 12531\ rad/s$$

$$p_L = \frac{z_L}{\alpha} = 78762\ rad/s$$

Por lo tanto, el PAF queda:

$$PAF(s) = K_{PAF}\frac{(s+12531)}{(s+78762)}$$

Para K_{PAF} sigo la misma estrategia que para K_{PI}, la calcularé al final.

El diagrama de bode resultante al añadir el regulador proporcional de avance de fases es el siguiente:

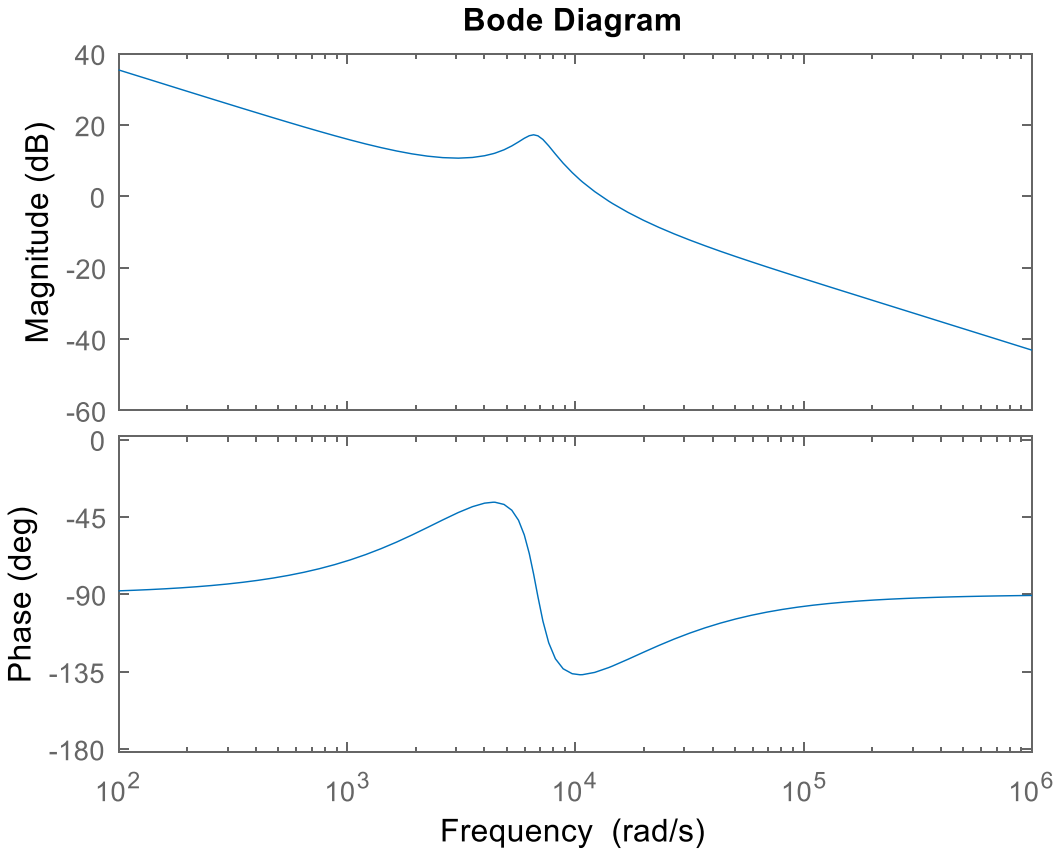


Figura 65

Donde podemos observar que ya hemos obtenido el margen de fase deseado, el último paso es ajustar la ganancia de los reguladores para que el diagrama de modulo corte en 0dB con la frecuencia de corte deseada. Observamos que a la frecuencia de corte deseada el diagrama de módulos de encuentra en -12dB, por lo que habrá que añadir 12dB para conseguir este cruce por cero.

$$20logK = 12.12dB$$

$$K = 4.04 = K_{PAF} * K_{PI}$$

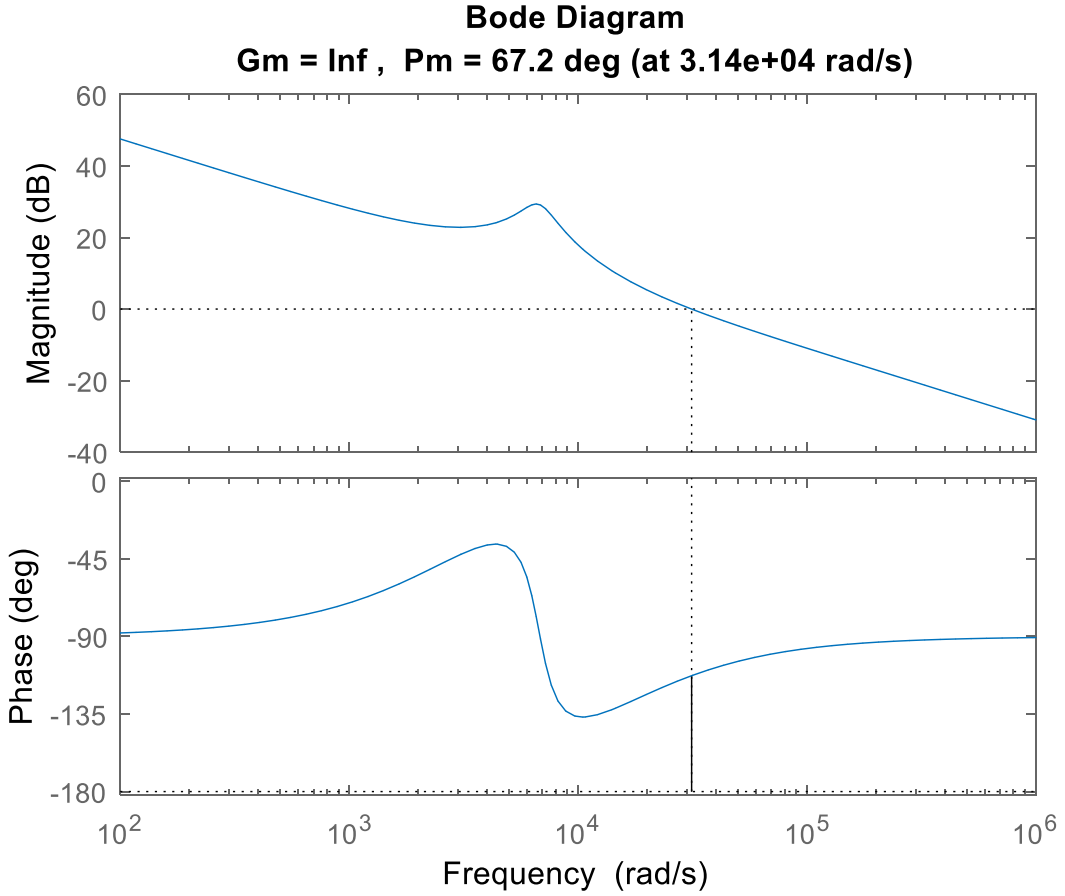


Figura 66

Donde podemos observar que se cumplen las especificaciones anteriores.

Por lo tanto, el regulador completo:

$$C(s) = K\frac{(s+z_1)(s+z_L)}{s(s+p_L)}$$

$$C(s) = 4.04\frac{(s+3142)(s+12531)}{s(s+78762)}$$

La frecuencia de muestreo del controlador digital f_S será igual a la de conmutación f_{SW}. Discretizando el controlador analógico con el método bilineal se obtiene el siguiente controlador digital C(z):

$$C(z) = \frac{D(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Donde $(b_0, b_1, b_2) = (3.129, -5.791, 2.674)$ y $(a_1, a_2) = (-1.435, 0.435)$. El polo en z=1 se corresponde al integrador que anula el error de posición, para no perderlo hay que asegurarse que se cumple que $1 + a_1 + a_2 = 0$.

Anexo III.3 Cálculo acerca de la etapa acondicionamiento del conversor analógico/digital

Para diseñar la ganancia de la etapa de acondicionamiento de la tensión de entrada al conversor A/D hemos determinado que la tensión en el punto de trabajo del convertidor represente aproximadamente el 70% del valor máximo del rango dinámico del conversor.

En el presente trabajo de fin de grado se presentan dos entradas a los conversores analógico/digitales, para la medida que se realiza en cada uno de ellos, el fabricante recomienda unos circuitos de acondicionamiento, los cuales he implementado en mi diseño[5].

Cabe destacar que todos los cálculos están condicionados por que se cumpla la siguiente condición:

$$t_{adq} + t_{settle} = T_S$$

$$t_{settle} = (Number\ of\ constants)RC$$

El tiempo de adquisición viene determinado por el conversor A/D interno de FPGA el cual es de 1.1 µs, y siendo el tiempo del ciclo de 10µs, nos queda un tiempo 8.9 µs para el tiempo de establecimiento.

Anexo III.3.1 Circuito de acondicionamiento de V_0

En el presente apartado se plantea la explicación del circuito de acondicionamiento de la tensión de salida, basado en las especificaciones del fabricante, en la siguiente Figura, extraída de [5], se observa el circuito planteado:

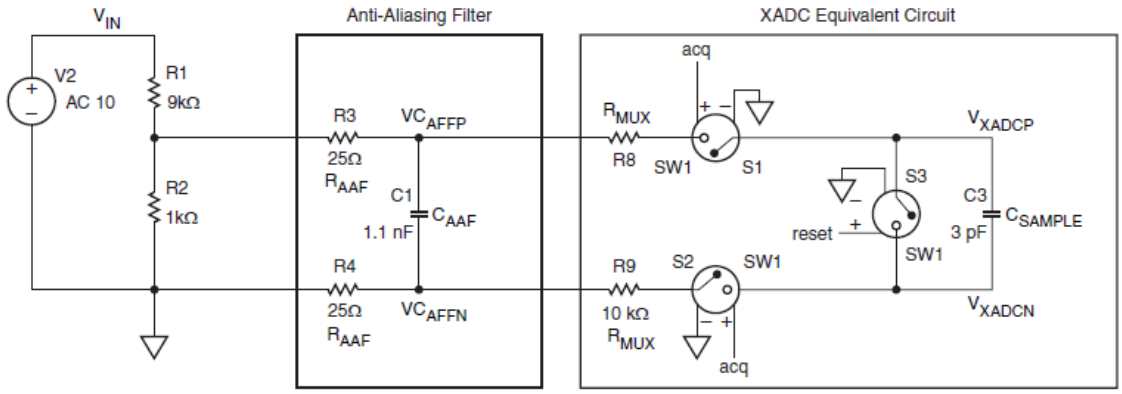


Figura 67

Por una parte, se observa un divisor resistivo el cual es el encargado de convertir la tensión de salida del Buck, en un valor en torno al 70 % del máximo valor del rango dinámico del conversor, siendo este de 1V.

Como ya se ha comentado en apartados anteriores, la tensión de salida deseada del Buck es de 5V, sin embargo, el peor caso que se puede presentar es de una tensión de salida de 6V, la cual hay que tener presente a la hora de realizar estos cálculos.

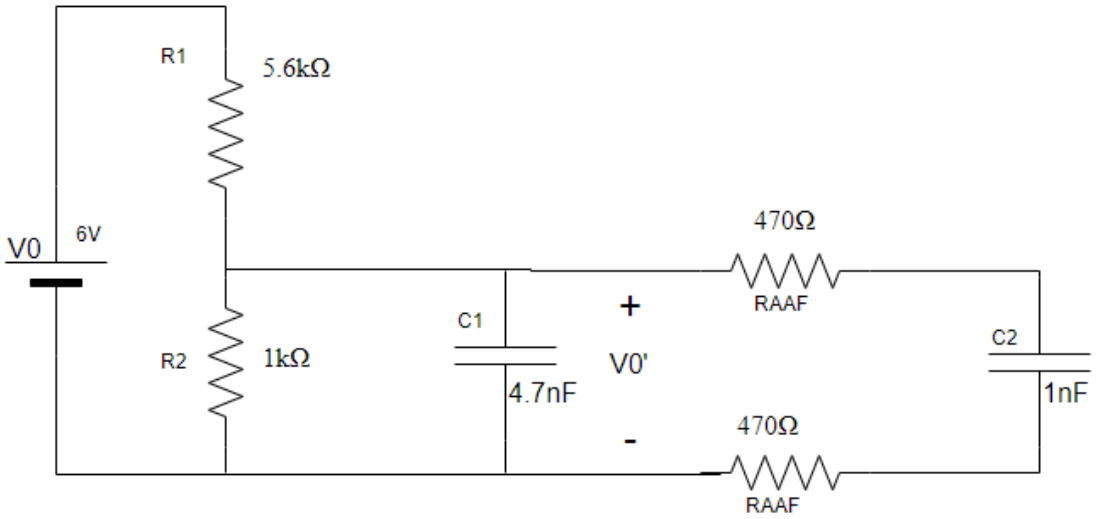


Figura 68

Por lo tanto, en nuestro caso se ha elegido un divisor resistivo cuyas resistencias R1 y R2 toman los valores de 5.6 y 1 kΩ respectivamente.

Para una tensión de salida de 5V a la salida del divisor resistivo se obtiene una tensión de:

$$V_0' = V_0 \frac{R_2}{R_{1+}R_2} = 0.75V$$

A continuación, se presenta el siguiente problema, y es que por un lado se pretende diseñar un filtro anti-aliasing de 40KHz y por otro lado hay que asegurar que el tiempo de settle sea inferior a 8.9 µs. Las ecuaciones que rigen ambas ecuaciones están gobernadas por los mismos parámetros por lo que cumplir ambas resulta imposible. Como solución se propone lo siguiente: añadir un condensador, C1 de la Figura anterior, de forma que por un lado tengamos un filtro cuya función sea de filtro anti-aliasing, y por otro lado otro filtro, con una frecuencia de corte mucho mayor, el cual sirva para asegurar el tiempo de settle.

Se ha elegido un ancho de banda para el primer filtro de 40 KHz, siendo este igual que el ancho de banda que proporciona el amplificador operacional en el caso de la medida de corriente.

$$f_{corte} = \frac{1}{2\Pi(R_1//R_2)C_1}$$

$$40KHz = \frac{1}{2\Pi\ 848.48\ C_1}$$

$$C_1 = 4.68\ nF$$

Por lo tanto, se elige un valor para C1 de 4.7 nF.

Para el segundo filtro se exige que el tiempo de settle sea inferior a 8.9 µs. Hay que tener en cuenta el filtro que ya tiene implementado la FPGA requiere un tiempo de settle, por lo tanto:

$$t_{settle\ interno} = 7.62 \cdot 2 \cdot 10K\Omega \cdot 3\text{pF} = 457.2\text{ns}$$

El número de tiempo de constantes de tiempo se ha elegido el correspondiente a una precisión de 10 bits, ya que en nuestro proyecto se trabaja con 9 bits, de allí el valor de 7.62, esto se puede observar en la siguiente Figura obtenida de [5].

Resolution (bits)	Error (%)	Number of Time Constants
6	0.3906	4.85
8	0.0977	6.24
10	0.0244	7.62
12	0.0061	9.01
14	0.0015	10.4

Figura 69

Como máximo tiempo de settle tenemos 8.442 µs. El condensador C2 se elige de 1 nF, debido a que este es el valor recomendado por el fabricante y además cumple la especificación de que sea mucho mayor que el condenador de 3pF interno de la FPGA

$$t_{settle} > 7.62 \cdot 2R_{AAF}C_2$$

Despejando R_{AAF}

$$R_{AAF} < 553.93\Omega$$

Por lo tanto, se eligen unas R_{AAF} de 470 Ω

Vamos a calcular la frecuencia de corte de este segundo filtro con objetivo de ver que es mucho mayor que la del anterior y que esta no tendrá ninguna incidencia en el filtrado.

$$f_{corte} = \frac{1}{2\Pi\ 2R_{AAF}\ C_2}$$

$$f_{corte} = 169.31\ KHz$$

Se utiliza la herramienta Orcad Pspice para simular el circuito de acondicionamiento con el fin de verificar el correcto funcionamiento y con ello el cálculo de los distintos componentes presentes en el circuito de acondicionamiento. El circuito en Orcad es el siguiente:

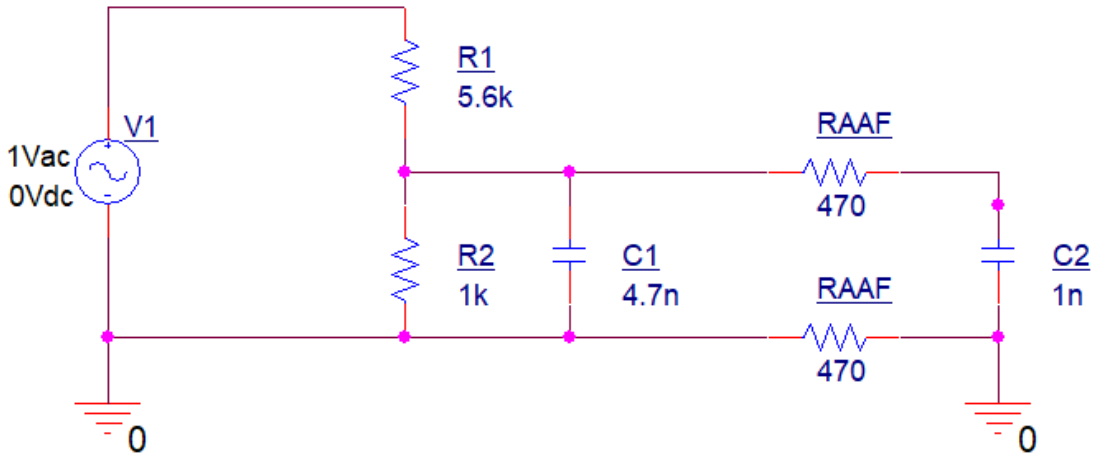


Figura 70

Realizando un análisis en frecuencia a la salida de ambos filtros se observa lo siguiente:

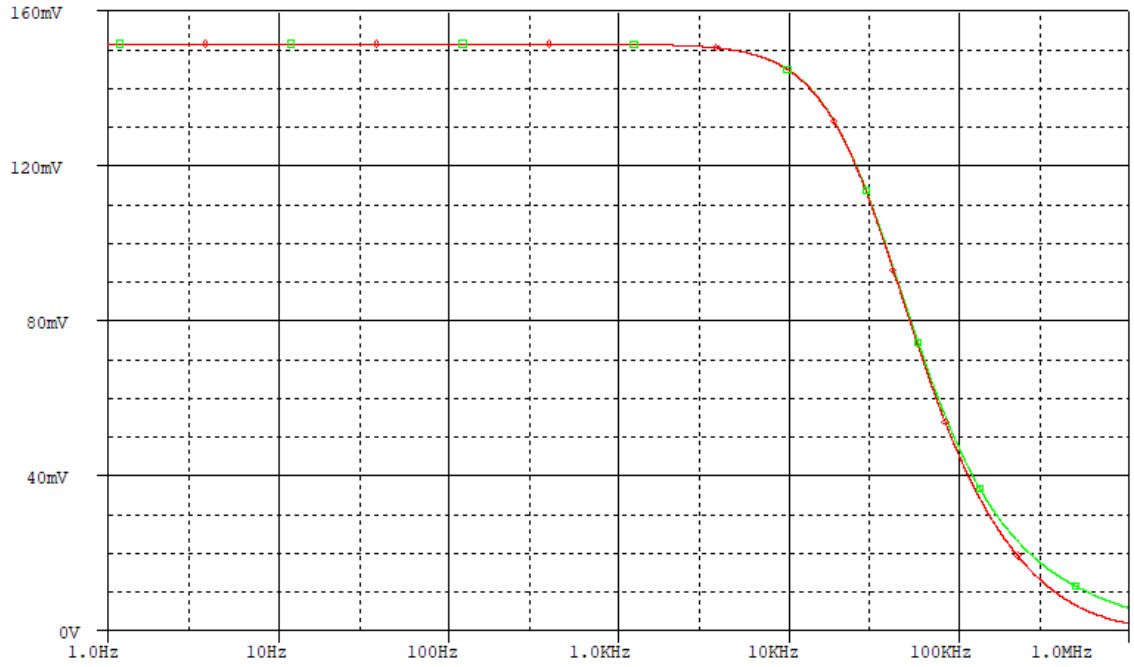


Figura 71

Donde la señal roja se corresponde con la salida del primer filtro, 40 KHz, tal y como se puede observar en la imagen. La señal verde se corresponde con la salida del segundo filtro, la cual como podemos observar no tiene casi influencia en el sistema y por lo tanto la hipótesis planteada con anterioridad de “separar” el circuito en dos resulta válida.

Anexo III.3.2 Circuito de acondicionamiento de la medida de corriente y cálculo de Rshunt

En este apartado se plantea el cálculo y justificación del circuito de acondicionamiento de la medida de corriente, así como el cálculo de Rshunt para cumplir con la especificación de que la tensión que le llegue al conversor sea aproximadamente de 70 % del máximo valor del rango dinámico del conversor. Al igual que en el caso anterior se va a seguir el circuito recomendado por el fabricante, el cual podemos observar en la siguiente figura, extraída de [5].

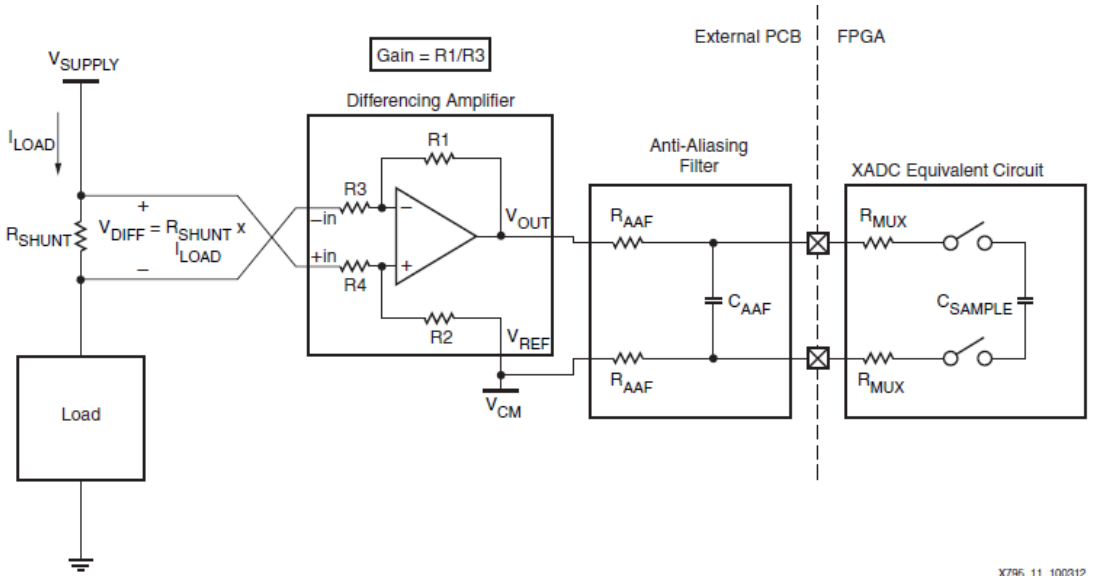


Figura 72

El amplificador escogido es el INA214, el cual presenta una ganancia de 100 V/V, este es uno de los propuestos por el fabricante. La corriente máxima que circula por la resistencia de shunt es la siguiente:

$$I_{Rshunt} = i_0 + \frac{\Delta I_L}{2} = 0.54 + 0.065 = 0.605A$$

Por lo tanto, la tensión que llegar a la entrada del conversor será la siguiente:

$$V_{IL} = G_{INA214} \cdot I_{Rshunt} \cdot R_{shunt}$$

La tensión a la entra puede ser como máximo de 1V por lo tanto la Rshunt:

$$R_{shunt} < 16.52m\Omega$$

Teniendo en cuenta lo comentado anteriormente acerca de que la tensión que le llegue al conversor sea aproximadamente de 70 % del máximo valor del rango dinámico del conversor, eligiendo una Rshunt de 10 mΩ se obtiene una V_{IL} de 0.605 V.

Para el caso del cálculo del filtro anti-aliasing, el cálculo es exactamente el mismo que el presentado en el caso anterior de la medida de la tensión.

Anexo III.4 Cálculo del valor del condensador Bootstrap

La utilización de la técnica *Bootstrap* es fundamental en este tipo de sistemas electrónicos de potencia para poder realizar correctamente el disparo de la puerta del MOSFET del lado *high* del semipuente.

Para el cálculo de la capacidad de almacenamiento de carga mínima del condensador a utilizar para poder realizar dicho disparo nos hemos apoyado en la muy completa documentación que facilita el fabricante sobre el driver LM5104 utilizado en la construcción de este prototipo.

Se define ΔV_{HB} como la variación de tensión en la puerta del lado *high* del sistema de la siguiente manera:

$$\Delta V_{HB} = V_{DD} - V_{DH} - V_{HBL}$$

Siendo V_{DD} la tensión a la que está alimentado el circuito, V_{DH} la caída de tensión en directa del diodo de *Bootstrap* que integra el *driver*. A su vez V_{HBL} puede ser calculado como:

$$V_{HBL} = V_{HBR} - V_{HBH}$$

Siendo V_{HBR} la tensión de subida umbral en el pin *HB* del *driver* para la carga del condensador de *Bootstrap* y V_{HBH} el umbral de histéresis del proceso.

El valor de la tensión, $V_{DD,}$ es de 12V, mientras que el resto de parámetro se pueden observar en la hoja de características del fabricante del driver LM5104M, siendo estos de valor V_{DH} = 1.1V, V_{HBR} = 7.1V, V_{HBH} = 0.4V, por lo tanto, ΔV_{HB}:

$$\Delta V_{HB} = 12V - 1.1V - (7.1V - 0.4V) = 4.2V$$

Con este dato podemos definir el valor de la capacidad del condensador de *Bootstrap* como:

$$C_{BOOT} = \frac{Q_{TOTAL}}{\Delta V_{HB}}$$

Siendo la carga energética total Q_{TOTAL} calculada a partir de la siguiente ecuación:

$$Q_{TOTAL} = Q_{qmax} + I_{HBO}\frac{D_{max}}{f_{SW}}$$

Con $I_{HBO} = 10\ \mu A$ la corriente del circuito de *Bootstrap* y $Q_{gmax} = 34\ nC$ la carga en la puerta del MOSFET para su disparo, características obtenidas de la información proporcionada por los fabricantes.

Además, el duty máximo, D_{max}, el cual es de 0.9 como se ha comentado anteriormente en el apartado 4.3 Obtención del regulador. Para la frecuencia de conmutación, f_{SW}, de 100KHz, también justificada en el apartado 4.4.1 Selección de la estructura de realización del controlador.

$$Q_{TOTAL} = 34nC + 10\mu A\frac{0.9}{100KHz}$$

$$Q_{TOTAL} = 34.09\ nC$$

Por lo tanto, el valor del condensador de *Bootstrap*:

$$C_{BOOT} \geq \frac{34.09\ nC}{4.2\ V} = 8.11\ nF$$

Inicialmente se eligió un valor de 10 nF, pero a la hora de hacer la prueba experimental se observó lo siguiente:

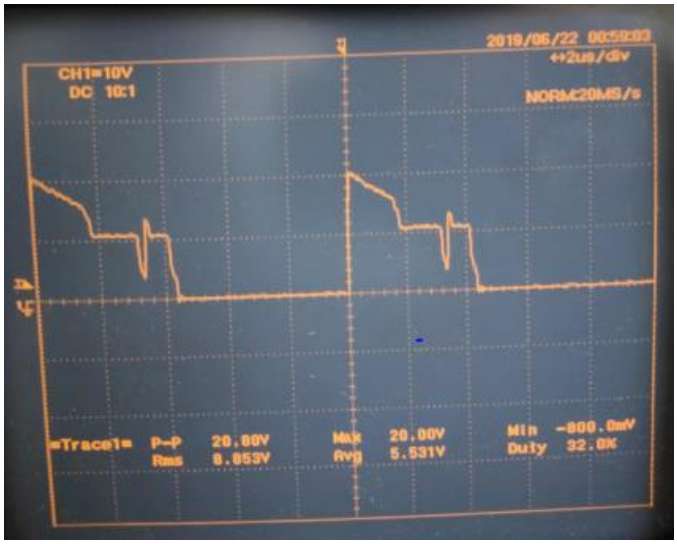


Figura 73

Y es que este condensador resultaba muy pequeño, se optó por soldar un condensador 10 veces más grande, este resulta adecuado tal y como se puede observar en 6. Resultados

Anexo III.5 Selección de la resistencia para el ajuste del tiempo muerto

Con el objetivo de que en ningún momento los dos MOSFET controlados por el driver LM5104M coincidan en un mismo estado de conducción, lo que daría lugar al cortocircuito del semipuente del reductor y por ende a su destrucción, el driver LM5104M incluye un tiempo muerto programable para asegurar un pequeño desfase entre ambas señales de disparo. Con ayuda de la siguiente figura extraída del *datasheet* del fabricante podemos obtener el valor de la resistencia a colocar en el diseño.

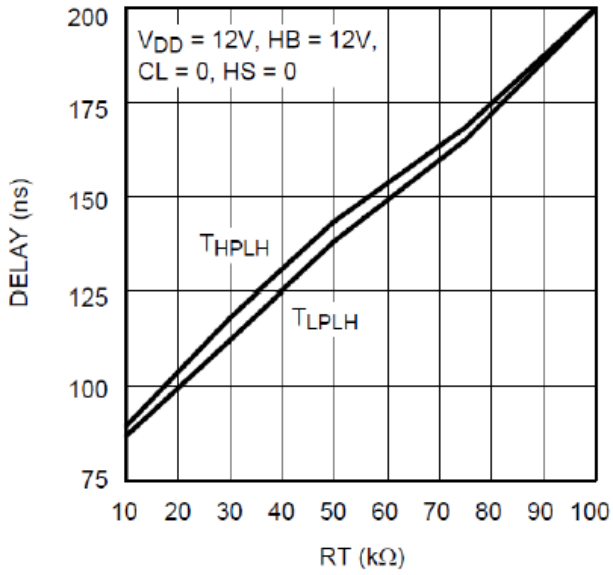


Figura 74

Teniendo en cuenta la temporización para la que se ha diseñado el reductor *Buck* en el presente trabajo fin de grado, se ha elegido una resistencia de 100kΩ que proporciona un tiempo muerto de 0.02µs.

Anexo III.6 Consideraciones acerca de la disipación del diodo de *Bootstrap*

En la hoja de características del componente LM5104 se explica cómo las fuentes de disipación de potencia para este dispositivo son dos: por un lado, las pérdidas relacionadas con el driver de la puerta y por otro las relacionadas con el diodo de *Bootstrap*.

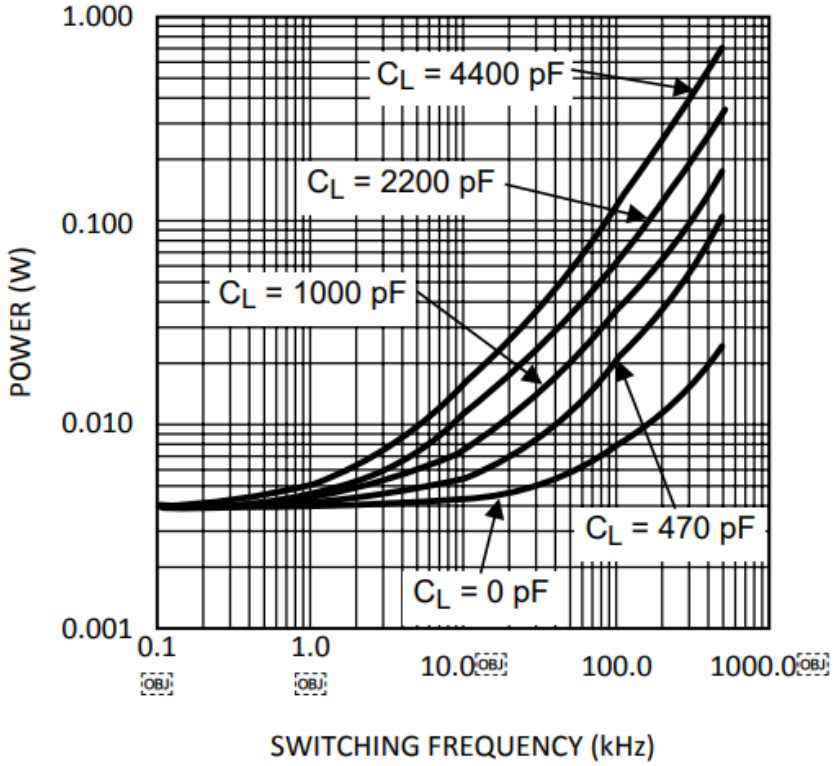


Figura 75

Como se puede observar la potencia disipada por este dispositivo es muy pequeña y por tanto podemos desechar la idea de utilizar diodos de *Bootstrap* de menor caída directa de tensión o modificar la frecuencia de conmutación.

Anexo IV. Diseño en lenguaje VHDL

-Código correspondiente al bloque Syncro_ctl

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Synchro_ctl is
   Port ( SW_CLOSED_IN : in STD_LOGIC;
        SW_HI_LOAD_IN : in STD_LOGIC;
        SW_CHANNEL_IN:in std_logic;
        CLK : in STD_LOGIC;
        RSTIN : in STD_LOGIC;
        SW_CLOSED : out STD_LOGIC;
        SW_HI_LOAD : out STD_LOGIC;
        SW_CHANNEL: out std_logic;
        RST : out STD_LOGIC);
end Synchro_ctl;
architecture Behavioral of Synchro_ctl is
signal RSTIN1,
RSTIN2,SW_CLOSED_IN1,SW_CLOSED_IN2,SW_HI_LOAD_IN1,SW_HI_LOAD
_IN2,SW_CHANNEL_IN1,SW_CHANNEL_IN2: std_logic;
begin
--proceso de F/F
        process(CLK)
        begin
           if(CLK' event and CLK='1') then
                RSTIN1<=RSTIN;
                RSTIN2<=RSTIN1;
                SW_CLOSED_IN1<=SW_CLOSED_IN;
                SW_CLOSED_IN2<=SW_CLOSED_IN1;
                SW_HI_LOAD_IN1<=SW_HI_LOAD_IN;
                SW_HI_LOAD_IN2<=SW_HI_LOAD_IN1;
                SW_CHANNEL_IN1<=SW_CHANNEL_IN;
                SW_CHANNEL_IN2<=SW_CHANNEL_IN1;
           end if;
```

```
        end process;

--salidas sincronizadas
   RST<=RSTIN2;
   SW_CLOSED<=SW_CLOSED_IN2;
   SW_HI_LOAD<=SW_HI_LOAD_IN2;
   SW_CHANNEL<=SW_CHANNEL_IN2;
end Behavioral;
```

-Código correspondiente al bloque bin2bcd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity bin2bcd is
   Port ( CLK : in  STD_LOGIC;
          RST : in  STD_LOGIC;
          DA_VALID: in std_logic;
          DATA : in  STD_LOGIC_VECTOR (9 downto 0);
          BCD : out  STD_LOGIC_VECTOR (11 downto 0));
end bin2bcd;

architecture Behavioral of bin2bcd is
   signal BIN_SH, nBIN_SH: unsigned(9 downto 0);
   signal BCD_SH, nBCD_SH, BCD_SH_ADD: unsigned(11 downto 0);
          -- output register to keep output constant during conversion
   signal BCD_OUT, nBCD_OUT: unsigned(11 downto 0);

   -- To keep track of shifts
   signal CN_SHIFTS, nCN_SHIFTS: natural range 0 to 10;
```

```
begin
   process(CLK)
   begin
                if (CLK'event and CLK='1') then
                   if (RST = '1') then
                      BIN_SH <= (others => '0');
                      BCD_SH <= (others => '0');
                      BCD_OUT <= (others => '0');
                      CN_SHIFTS <= 0;
                   else
                      BIN_SH <= nBIN_SH;
                      BCD_SH <= nBCD_SH;
                      BCD_OUT <= nBCD_OUT;
                      CN_SHIFTS <= nCN_SHIFTS;
                         end if;
      end if;
   end process;

   process (DA_VALID, DATA, BIN_SH, BCD_SH, CN_SHIFTS, BCD_SH_ADD)
   begin
                if (DA_VALID = '1') then -- register DATA
                         nBIN_SH <= unsigned(DATA);
                         nBCD_SH <= (others => '0');
                         nCN_SHIFTS <= 10;
                elsif (CN_SHIFTS /= 0)  then -- shifts
                         nBIN_SH <= BIN_SH(8 downto 0) & '0';
         nBCD_SH <= BCD_SH_ADD(10 downto 0) & BIN_SH(9);
         nCN_SHIFTS <= CN_SHIFTS - 1;
                else
                         nBIN_SH <= BIN_SH;
```

```
                    nBCD_SH <= BCD_SH;
                    nCN_SHIFTS <= CN_SHIFTS;
                end if;

        end process;

    BCD_SH_ADD(11 downto 8) <= BCD_SH(11 downto 8) + 3 when BCD_SH(11 downto 8) > 4 else
                    BCD_SH(11 downto 8);
    BCD_SH_ADD(7 downto 4) <= BCD_SH(7 downto 4) + 3 when BCD_SH(7 downto 4) > 4 else
                    BCD_SH(7 downto 4);
    BCD_SH_ADD(3 downto 0) <= BCD_SH(3 downto 0) + 3 when BCD_SH(3 downto 0) > 4 else
                    BCD_SH(3 downto 0);
    nBCD_OUT <= BCD_SH when (CN_SHIFTS = 0) else BCD_OUT;
    BCD <= std_logic_vector(BCD_OUT);

end Behavioral;
```

-Código correspondiente al bloque buck_ctl

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;
use work.fixed_pkg.all;



entity buck_ctl is
    Port ( SW_CLOSED : in STD_LOGIC;
           SW_HI_LOAD : in STD_LOGIC;
           CLK : in STD_LOGIC;
           RST : in STD_LOGIC;
           DA_VALID : in STD_LOGIC;
           DA_IN : in STD_LOGIC_VECTOR (8 downto 0);
           G1 : out STD_LOGIC;
           G3 : out STD_LOGIC;
           CONVST: out std_logic);

end buck_ctl;

architecture Behavioral of buck_ctl is
    signal EN: std_logic;
    signal TIMER, TIMER_sig:unsigned(9 downto 0);
    signal DUTY, nDUTY: unsigned(9 downto 0);
    constant cien: unsigned(9 downto 0):= to_unsigned(100,10);
    constant novecientos: unsigned(9 downto 0):= to_unsigned(900,10);
    constant DUTY_LAZO_ABIERTO: unsigned (8 downto 0):=to_unsigned(417,9);
    signal pDUTY_LAZO_ABIERTO:unsigned (8 downto 0);

    constant B0: sfixed(6 downto -10):= to_sfixed(6.1113,6,-10);
    constant B1: sfixed(6 downto -10):= to_sfixed(-11.3105,6,-10);
```

```
    constant B2: sfixed(6 downto -10):= to_sfixed(5.222,6,-10);
    constant A1: sfixed(2 downto -10):= to_sfixed(-1.435,2,-10);
    constant A2: sfixed(2 downto -10):= to_sfixed(0.435,2,-10);
    constant NREF: unsigned(8 downto 0):= to_unsigned(194,9);
    constant N1: unsigned(9 downto 0):= to_unsigned(860,10);
    constant N2: unsigned(9 downto 0):= to_unsigned(980,10);
    signal nX,pX: sfixed(6 downto -10);
    signal nY,pY: sfixed(14 downto -10);
    signal ERR0, ERR1, ERR2, ERR0_sig, ERR1_sig, ERR2_sig : sfixed(9 downto 0);
    signal D1, D2, D1_sig, D2_sig  : sfixed(14 downto -10);
    signal ACC,ACC_sig: sfixed(14 downto -10);
    signal nPRODUCTO1,pPRODUCTO1:sfixed(21 downto -20);
    signal nPRODUCTO2,pPRODUCTO2:sfixed(22 downto -20);
    signal pG1,nG1: std_logic;
    signal nDA_IN,pDA_IN: STD_LOGIC_VECTOR (8 downto 0);

begin
    process(RST,CLK)
    begin
        if(RST='1')then
            TIMER<=(others=>'0');
            ERR0<=(others=>'0');
            ERR1<=(others=>'0');
            ERR2<=(others=>'0');
            D1<=(others=>'0');
            D2<=(others=>'0');
            ACC<=(others=>'0');
            pX<=(others=>'0');
            pY<=(others=>'0');
            pPRODUCTO1<=(others=>'0');
```

```
            pPRODUCTO2<=(others=>'0');
            pDUTY_LAZO_ABIERTO<=(others=>'0');
            pG1<='0';
            pDA_IN<=(others=>'0');

        elsif(CLK'event and CLK='1')then
            TIMER<=TIMER_sig;
            ERR0<=ERR0_sig;
            ACC<=ACC_sig;
            pX<=nX;
            pY<=nY;
            pPRODUCTO1<=nPRODUCTO1;
            pPRODUCTO2<=nPRODUCTO2;
            pDUTY_LAZO_ABIERTO<=DUTY_LAZO_ABIERTO;
            pG1<=nG1;
            pDA_IN<=nDA_IN;

            if(EN='1') then
                ERR2<=ERR1;
                ERR1<=ERR0;
                D2<=D1;
                D1<=D1_sig;
            end if;
        end if;
    end process;



--combinacional para timer
    TIMER_sig<= TIMER + 1 when (TIMER<999) else (others=>'0');
```

```
--combinacional para EN
   EN<='1' when (TIMER=999) else '0';
--combinacional para ERR0_sig
   ERR0_sig<= to_sfixed(signed('0'&NREF)-signed('0'&pDA_IN),ERR0) when
(TIMER=N2-1)else
         ERR0;

--registros en los cuales voy almacenando los disitintos coeficientes para realizar el
producto
   nX<= B0 when (TIMER=N2)   else
       B1 when (TIMER=N2+1) else
       B2 when (TIMER=N2+2) else
       resize(-(A1),nX) when (TIMER=N2+3) else
       resize(-(A2),nX) when (TIMER=N2+4) else
      pX;

   nY<= resize(ERR0,nY) when (TIMER=N2)   else
       resize(ERR1,nY) when (TIMER=N2+1) else
       resize(ERR2,nY) when (TIMER=N2+2) else
       D1   when (TIMER=N2+3) else
       D2   when (TIMER=N2+4) else
       pY;
--realiazación de las correspondientes operaciones para calcular el Acumulador
   nPRODUCTO1<=pX*pY;

   ACC_sig<=resize(pPRODUCTO1 + ACC,ACC,fixed_saturate,fixed_truncate) when
(TIMER=N2+2 or TIMER=N2+6 or TIMER=N2+3 or TIMER=N2+4 or
TIMER=N2+5)else
         (others=>'0') when TIMER=N2 else
         ACC;

--combinscional para D1_sig
```

```
    D1_sig<= resize (ACC,D1,fixed_saturate,fixed_round);


--combinacional para D1
    process(D1)
    begin
        if(D1>900) then
            DUTY<=novecientos;
        elsif(D1<100) then
            DUTY<=cien;
        else
            DUTY<=unsigned(D1(9 downto 0));
        end if;
    end process;


--combinacional para G1
    nG1<='1' when (TIMER<DUTY and SW_CLOSED ='1') else
         '1' when (TIMER<417  and SW_CLOSED ='0') else
         '0';
    G1<=pG1;
--G3
    G3<='1' when(SW_HI_LOAD='1') else '0';
--combinacional para CONVST
    CONVST<='1' when (TIMER=N1) else '0';


--combinacional para DA_IN
    nDA_IN<=DA_IN when (DA_VALID='1') else pDA_IN;


end Behavioral;
```

-Código correspondiente al bloque ctl_adc

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ctl_adc is
   Port ( CLK : in STD_LOGIC;
          RST : in STD_LOGIC;
          D0_VALID:in STD_LOGIC;
          DA_IN : in STD_LOGIC_VECTOR (15 downto 0);
          EOC : in STD_LOGIC;
          DEN:out std_logic;
          ADDR : out STD_LOGIC_VECTOR (6 downto 0);
          DA_VALID: out STD_LOGIC;
          V0_ADC : out STD_LOGIC_VECTOR (8 downto 0);
          VIL_ADC : out STD_LOGIC_VECTOR (8 downto 0));
end ctl_adc;

architecture Behavioral of ctl_adc is

signal nCuenta,pCuenta:unsigned(9 downto 0);
signal pV0_ADC,nV0_ADC,pVIL_ADC,nVIL_ADC: std_logic_vector(8 downto 0);
type STATES is
(REPOSO,ESPERO1,LEOV0,ESPERO2,LEOVIL,GEN_DA_VALID);
signal nSTATE,pSTATE:STATES;




begin
process (CLK,RST)
    begin
```

```
      if (CLK'event and CLK='1') then
         if (RST = '1') then
            pCuenta <= (others=>'0');
            pV0_ADC <= (others=>'0');
            pVIL_ADC<= (others=>'0');
            pSTATE<=REPOSO;

         else
            pCuenta <= nCuenta;
            pV0_ADC<=nV0_ADC;
            pVIL_ADC<=nVIL_ADC;
            pSTATE<=nSTATE;
         end if;
      end if;
   end process;

--Actualizo salidas
    VIL_ADC<=pVIL_ADC;
    V0_ADC<=pV0_ADC;

--combinacional para DA_VALID
    DA_VALID<='1' when (pSTATE=GEN_DA_VALID) else '0';




--maquina de estados
    process(pSTATE,EOC,D0_VALID,DA_IN)
    begin
```

```
nSTATE<=pSTATE;
case pSTATE is
   when REPOSO=>
      if(EOC='1')then
         DEN<='1';
         ADDR<="0011111";
         nSTATE<=ESPERO1;
      end if;
   when ESPERO1=>
      DEN<='0';
      if(D0_VALID='1') then
         nSTATE<=LEOV0;
      end if;
   when LEOV0=>
      nV0_ADC<=DA_IN (15 downto 7);
      DEN<='1';
      ADDR<="0010111";
      nSTATE<=ESPERO2;
   when ESPERO2=>
      DEN<='0';
      if(D0_VALID='1') then
         nSTATE<=LEOVIL;
   end if;
   when LEOVIL=>
      nVIL_ADC<=DA_IN (15 downto 7);
      nSTATE<=GEN_DA_VALID;

   when GEN_DA_VALID=>
      nSTATE<=REPOSO;
end case;
```

```
    end process;


 end Behavioral;
```

-Código correspondiente al bloque display4x7

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


entity display4x7 is
    generic (
        CN_DISP_WIDTH : integer := 16;
                    -- Every digit will last 2**CN_DISP_WIDTH*Tclk = 2**16/100MHz =
0.65 ms
        DIGITS : integer := 3   -- 2 to 4
    );
    Port ( CLK : in STD_LOGIC;
           RST : in STD_LOGIC; -- Synchronous reset
           VAL : in STD_LOGIC_VECTOR (DIGITS*4-1 downto 0);
           DP : in STD_LOGIC_VECTOR (DIGITS-1 downto 0); -- Active high
           SSEG : out STD_LOGIC_VECTOR (6 downto 0);
           DPSEG : out STD_LOGIC;
           AN : out STD_LOGIC_VECTOR (3 downto 0));
end display4x7;




architecture Behavioral of display4x7 is
           -- Counter to multiplex digits
    signal CN_DISP, nCN_DISP: unsigned(CN_DISP_WIDTH-1 downto 0);
```

```
    constant CN_DISP_MAX: unsigned(CN_DISP_WIDTH-1 downto 0):=
(others=>'1');
    signal DIGIT_EN: std_logic;
    -- Multiplexed display
    signal VALGEN: unsigned (15 downto 0);
    signal DPGEN: std_logic_vector (3 downto 0);
    signal BCD: unsigned(3 downto 0);
    signal DIGIT_CTL, nDIGIT_CTL: std_logic_vector(3 downto 0);
begin
    process (CLK)
    begin
       if (CLK'event and CLK='1') then
          if (RST = '1') then
             CN_DISP <= (others=>'0');
             DIGIT_CTL <= "1110";
          else
             CN_DISP <= nCN_DISP;
             DIGIT_CTL <= nDIGIT_CTL;
          end if;
       end if;
    end process;

          --  Counter to multiplex digits
    nCN_DISP <= CN_DISP + 1;
    DIGIT_EN <= '1' when (CN_DISP = CN_DISP_MAX) else '0';



    -- Multiplexed display shift register
    process (DIGIT_CTL, DIGIT_EN)
    begin
```

```
    if (DIGIT_EN='1') then
        nDIGIT_CTL <= DIGIT_CTL(0) & DIGIT_CTL(3 downto 1);
    else
        nDIGIT_CTL <= DIGIT_CTL;
    end if;
end process;

    -- VAL and DP expansion from DIGITS
DIGIT2: if (DIGITS = 2) generate
    VALGEN <="00000000" & unsigned(VAL);
    DPGEN <= "11" & not DP;
end generate;

DIGIT3: if (DIGITS = 3) generate
    VALGEN <="0000" & unsigned(VAL);
    DPGEN <= '1' & not DP;
end generate;

DIGIT4: if (DIGITS = 4) generate
    VALGEN <= unsigned(VAL);
    DPGEN <= not DP;
end generate;

    -- Turn-off not used displays
    AN <=       DIGIT_CTL when (DIGITS = 4) else
                '1' & DIGIT_CTL(2 downto 0) when (DIGITS = 3) else
                "11" & DIGIT_CTL(1 downto 0);

BCD <=  VALGEN(15 downto 12) when (DIGIT_CTL(3)='0') else
        VALGEN(11 downto 8) when (DIGIT_CTL(2)='0') else
```

```
        VALGEN(7  downto 4) when (DIGIT_CTL(1)='0') else
        VALGEN(3  downto 0);


   SSEG <= "0000001" when (BCD = 0) else
        "1001111" when (BCD = 1) else
        "0010010" when (BCD = 2) else
        "0000110" when (BCD = 3) else
        "1001100" when (BCD = 4) else
        "0100100" when (BCD = 5) else
        "0100000" when (BCD = 6) else
        "0001111" when (BCD = 7) else
        "0000000" when (BCD = 8) else
        "0000100" when (BCD = 9) else
        "0001000" when (BCD = 10) else
        "1100000" when (BCD = 11) else
        "1110010" when (BCD = 12) else
        "1000010" when (BCD = 13) else
        "0110000" when (BCD = 14) else
        "0111000";


        DPSEG <= DPGEN(3) when (DIGIT_CTL(3)='0') else
         DPGEN(2) when (DIGIT_CTL(2)='0') else
         DPGEN(1) when (DIGIT_CTL(1)='0') else
         DPGEN(0);


end Behavioral;
```

-Código correspondiente al bloque ui

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.NUMERIC_STD.ALL;


entity ui is
   Port ( CLK : in STD_LOGIC;
          RST : in STD_LOGIC;
          SW_CHANNEL: in std_logic;
          V0_IN : in STD_LOGIC_VECTOR (8 downto 0);
          VIL_IN : in STD_LOGIC_VECTOR (8 downto 0);
          DA_VALID: in std_logic;
          SCALED_DA : out STD_LOGIC_VECTOR (9 downto 0);
          DP : out STD_LOGIC_VECTOR (2 downto 0));
end ui;


architecture Behavioral of ui is
          signal nCN, pCN: unsigned(24 downto 0);
          signal nMuestra,pMuestra:unsigned(6 downto 0);
          constant CN_MAX: unsigned(pCN'range) := (others=>'1');
          constant C660: unsigned (9 downto 0) := to_unsigned(660,10);
          constant C285: unsigned (7 downto 0) :=to_unsigned(285,8);
          signal nMULTV0, pMULTV0: unsigned (18 downto 0);
          signal nMULTV0Visualiza, pMULTV0Visualiza: unsigned (18 downto 0);
          signal nMULTVIL, pMULTVIL: unsigned (16 downto 0);
          signal nMULTVILVisualiza, pMULTVILVisualiza: unsigned (16 downto 0);
          signal pCuenta,nCuenta:unsigned (23 downto 0);
          constant Visualiza:unsigned (23 downto 0):=to_unsigned(10000000,24);



begin
    process (CLK)
    begin
```

```
      if (CLK'event and CLK='1') then
         if (RST = '1') then
            pCN <= (others=>'0');
            pMuestra<=(others=>'0');
            pMULTV0 <= (others=>'0');
            pMULTVIL <= (others=>'0');
            pCuenta <= (others=>'0');
            pMULTV0Visualiza <= (others=>'0');
            pMULTVILVisualiza <= (others=>'0');

         else
            pCN <= nCN;
            pMuestra<=nMuestra;
            pMULTV0 <= nMULTV0;
            pMULTVIL <= nMULTVIL;
            pCuenta<=nCuenta;
            pMULTV0Visualiza<=nMULTV0Visualiza;
            pMULTVILVisualiza<=nMULTVILVisualiza;

         end if;
      end if;
   end process;
--Acondiciono las señales para su correcta visualización
    nMULTV0 <= unsigned(V0_IN)*C660  when DA_VALID = '1' else pMULTV0;
    nMULTVIL<= unsigned(VIL_IN)*C285 when  DA_VALID = '1' else pMULTVIL;



-- Cojo una muestra cada 100ms para poder realizar una correcta visualizacion
    nMULTV0Visualiza<=(pMULTV0+pMULTV0)   when (pCuenta=Visualiza) else
pMULTV0Visualiza;
```

```
   nMULTVILVisualiza<=pMULTVIL when (pCuenta=Visualiza) else
pMULTVILVisualiza;


--Escojo la señales que se va a visualizar mediante el interruptor SW_CHANNEL
   SCALED_DA<=std_logic_vector(pMULTV0Visualiza(18 downto 9)) when
(SW_CHANNEL='1')
          else std_logic_vector(pMULTVILVisualiza(16 downto 7));


   DP <= "100";


--combinacional para pMuestra
   nMuestra<= (others=>'0') when (DA_VALID ='1') else
            pMuestra + 1 ;


--combinacional para pCuenta
   nCuenta<= (others=>'0') when (pCuenta = Visualiza) else
         pCuenta+1;
end Behavioral;
```

-Código correspondiente al fichero de interface usado para simular el sistema en bule cerrado, buck_sim_interface

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity buck_sim_interface is
   Port ( G1 : in STD_LOGIC;
         G3 : in STD_LOGIC;
         CONVST: in STD_LOGIC;
         DA_IN : out STD_LOGIC_VECTOR (8 downto 0));
end buck_sim_interface;


architecture Behavioral of buck_sim_interface is
```

```
-- ADC constants
constant VFS: real:= 1.0;       -- ADC Full Scale Voltage
constant B: integer:= 12;       -- ADC bits
constant LSB: real:= VFS/(2.0**B);
constant VFS1: real:= VFS-LSB;
   --constant C1000: real:=1000.0;
   -- Buck constants
   constant TS: real:= 10.0e-6;
   constant VG : real:= 12.0;
   constant L: real:= 220.0e-6;
   constant RL: real:= 0.16;
   constant C: real:= 100.0e-6;
   constant RC: real:= 0.13;
   constant RO: real:= 22.0;
   signal VC : real:= 0.0;
   signal IL : real:= 0.0;
   signal RR: real;
   signal VO, VOadc,V02adc: real:=0.0;
   signal VOm, ILm: integer := 0; -- mV and mA
   signal DATA: std_logic_vector (15 downto 0);

begin
-- Large signal averaged buck model
   RR <= RO when G3='0' else RO/2.0;
   VO <= (RC*RR/(RC+RR))*IL + (RR/(RC+RR))*VC;



        process (G1)
      variable IL_aux, d: real;
```

```
        begin
                if (G1'event and G1='0') then
                   d := real(G1'delayed'last_event / 10 ns)/1000.0; -- Duty cycle
                   if d > 0.9 then
                      d := 0.9; -- Ignore first G1 event from 'U' to '0'
        end if;
                   IL_aux := (1.0-TS*(RL/L + RC*RR/(L*(RC+RR)))) * IL -
              (TS*RR/(L*(RC+RR))) * VC + d*(TS*VG/L);
        if IL_aux < 0.0 then
           IL <= 0.0;
        else
           IL <= IL_aux;
        end if;
        VC <= (TS*RR/(C*(RC+RR)))*IL + (1.0-TS/(C*(RC+RR)))*VC;
      end if;
        end process;

-- To show analog style waveform
   VOm <= integer (VO*1000.0);
   ILm <= integer (IL*1000.0);

-- transform level voltage for input adc
   V02adc<=(VO/6.6);

-- ADC model
   VOadc <=0.0 when V02adc <= 0.0 else
        (VFS-LSB) when V02adc >= (VFS-LSB) else
         V02adc;

   DATA<= std_logic_vector(to_unsigned(integer(VOadc*((2.0**B)-1.0)/VFS1),16));-
- when (CONVST='1');
```

```
--refresh output
   DA_IN<=DATA(11 downto 3);


end Behavioral;
```

-Código correspondiente al test bench, buck_tb

```
entity buck_tb is
--  Port ( );
end buck_tb;

architecture Behavioral of buck_tb is
---defino ambos componentes
   component buck_sim_interface
       Port ( G1 : in  STD_LOGIC;
              G3 : in  STD_LOGIC;
              CONVST : in STD_LOGIC;
              DA_IN : out STD_LOGIC_VECTOR (8 downto 0));
    end component;
    component design_1_wrapper
        Port ( CLK : in STD_LOGIC;
               RSTIN : in STD_LOGIC;
               SW_CLOSED_IN: in STD_LOGIC;
               SW_HI_LOAD_IN: in STD_LOGIC;
               SW_CHANNEL: in STD_LOGIC;
               DA_IN: in STD_LOGIC_VECTOR(8 downto 0);
               Vaux7_v_n : in std_logic;
               Vaux7_v_p : in std_logic;
               Vaux15_v_n : in std_logic;
               Vaux15_v_p : in std_logic;
```

```
            CONVST : out STD_LOGIC;
            G1 : out STD_LOGIC;
            G3 : out STD_LOGIC;
            SSEG : out STD_LOGIC_VECTOR(6 downto 0);
            AN : out STD_LOGIC_VECTOR(3 downto 0);
            DPSEG : out STD_LOGIC);
        end component;

    signal CLK:            std_logic;
    signal RSTIN:          std_logic;
    signal SW_CLOSED_IN:   std_logic;
    signal SW_HI_LOAD_IN:  std_logic;
    signal SW_CHANNEL:     std_logic;
    signal DA_IN:          STD_LOGIC_VECTOR(8 downto 0);
    signal CONVST:         std_logic;
    signal G1:             std_logic;
    signal G3:             std_logic;

-- Clock period definitions
constant CLK_period : time := 10 ns;




begin
   uut0: design_1_wrapper port map (
      CLK=>CLK,
```

```
      RSTIN=>RSTIN,
      SW_CLOSED_IN=>SW_CLOSED_IN,
      SW_HI_LOAD_IN=>SW_HI_LOAD_IN,
      SW_CHANNEL=>SW_CHANNEL,
      DA_IN=>DA_IN,
      Vaux7_v_n=>'0',
      Vaux15_v_n=>'0',
      Vaux7_v_p=>'0',
      Vaux15_v_p=>'0',
      CONVST=>CONVST,
      G1=>G1,
      G3=>G3);

    uut1: buck_sim_interface port map (
      CONVST=>CONVST,
      G1=>G1,
      G3=>G3,
      DA_IN=>DA_IN);
-- le doy valor  a las distintas entradas al sistema
    CLK_process :
      process
      begin
         CLK <= '0';
         wait for CLK_period/2;
         CLK <= '1';
         wait for CLK_period/2;
      end process;

      RSTIN <=  '1', '0' after 50ns, '1' after 8ms, '0' after 9ms;--after 4 ms, '1' after 10 ms,
'0' after 15 ms;
```

```
    SW_CLOSED_IN<= '1';--, '1' after 7.5ms;--, '0' after 10 ms;
    SW_HI_LOAD_IN<= '1';--, '0' after 5 ms, '0' after 10 ms, '0' after 15 ms;
    SW_CHANNEL<='1', '0' after 5 ms, '1' after 10 ms, '0' after 15 ms;

end Behavioral;
```