UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Emulación HIL de un Convertidor Flyback con pérdidas

Máster Universitario en Ingeniería de Telecomunicación

Autor: Domínguez Araujo, Marta
Tutor: Martínez García, María Sofía
Ponente: Ángel de Castro Martín

FECHA: Diciembre, 2019

Emulación HIL de un Convertidor Flyback con pérdidas

AUTOR: Marta Domínguez Araujo
TUTOR: María Sofía Martínez García
PONENTE: Ángel de Castro Martín

Trabajo realizado en el grupo

Human Computer Technology Laboratory

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Diciembre de 2019

Resumen

Debido a que los sistemas son cada vez más complejos, es cada vez más complicado realizar simulaciones en tiempo real del sistema final. Nuevas técnicas para el proceso de pruebas y depuración han surgido en los últimos años, como es el caso de la técnica Hardware In-The-Loop, también conocida como HIL, que nos permiten simular en tiempo real. Esta técnica consiste en digitalizar la planta mediante un modelo matemático para, a continuación, poder implementarla sobre un ordenador o FPGA y llevar a cabo la simulación del sistema real.

En este proyecto, para poder emplear la técnica comentada, se ha llevado a cabo la digitalización de un convertidor Flyback. Se ha realizado un estudio de las aritméticas que se han empleado para desarrollar este modelo en lenguaje VHDL, y se han comparado entre sí los modelos: real, en coma fija y en coma flotante. Además, se realiza una comparativa de éstos con el modelo obtenido en la herramienta Simulink.

Por otro lado, se desarrollan estos modelos en lenguaje VHDL incluyendo pérdidas eléctricas para obtener resultados lo más reales posibles, ya que, al fin y al cabo, un modelo nunca va a ser ideal al 100%. Así, se comprueba cómo dichas pérdidas influyen en los resultados de los modelos.

Además, se sintetizan los modelos en coma fija y en coma flotante sin y con pérdidas en una FPGA Xilinx Zynq para obtener el área y el tiempo total de ejecución.

Por tanto, en los resultados mostrados en este proyecto se observa la precisión de cada uno de los modelos con las diferentes aritméticas comentadas. Además, con los estudios realizados se verifica que la implementación del modelo en coma fija permite que la velocidad de emulación sea mayor y que las simulaciones se puedan ejecutar en tiempo real y con mayor precisión.

Abstract

Because of the systems are increasingly complex, it's increasingly complicated to perform real-time simulations of the final system. New techniques for the process of testing and debugging have emerged in recent years, as the case of the *Hardware In-The-Loop* technique, also known as HIL, which allow us to simulate in real time. This technique consists on digitizing the plant using a mathematical model to, then, be able to implement it on a computer or FPGA and carry out the simulation of the real system.

In this project, in order to use the previous technique, the digitalization of a Flyback converter has been carried out. It has realised a study of the arithmetic that has been used to develop this model in VHDL language, and the models have been compared to each other: real, fixed point and floating point. Besides, it realises a comparison of these with the model obtained in the Simulink tool.

On the other hand, these models develop in VHDL language including electrical losses to obtain the most real results possible, due to, after all, a model will never be 100% ideal. Thus, it is verified how these losses influence the results of the models.

In addition, the fixed point and floating point models are synthesized without and with losses in a Xilinx Zynq FPGA to obtain the total area and execution time.

Therefore, the results shown in this project show the accuracy of each of the models with the different arithmetic mentioned. Besides, with the studies carried out, it verifies that the implementation of the fixed point model allows the emulation speed to be greater and that the simulations can be executed in real time and with greater precision.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

La electrónica de potencia se trata de una de las ramas más importantes de la electrónica. Es capaz de manejar y transformar energía de forma eficiente, permitiendo transformar y controlar voltajes y corrientes de niveles significativos. Gracias a la gran cantidad de dispositivos que solicitan unas condiciones de alimentación específicas, esta parte de la electrónica se ha convertido en una de las más prestigiosas.

Antes, el control de la electricidad se llevaba a cabo mediante reguladores lineales, pero tenían un inconveniente: baja eficiencia por disipadores en su interior. Hoy en día, existen otras formas de transformar energía obteniendo los mismos resultados y con menos pérdidas. Uno de los dispositivos que permite esta mejora son los llamados convertidores conmutados, los cuales consiguen una alta eficiencia gracias a que están formados por elementos no disipativos.

Estos convertidores se basan en circuitos que están formados, entre los elementos más característicos, por bobinas y condensadores, los cuales tienen la capacidad de almacenar y descargar energía. Además de éstos, se encuentran los conmutadores, que son los que se encargan de modificar el estado de estos elementos para cambiar el funcionamiento del circuito y así regularizar el valor de las señales de tensión y corriente a la salida del diseño. Asimismo, este modelo debe tener señales de control que modifiquen el estado de los interruptores.

Para conseguir una salida del circuito estable y controlado, estas señales de control deben depender de los valores de las señales de corriente y tensión del circuito en cada instante. Con los valores obtenidos se varían los estados de apertura y cierre de los conmutadores.

Los controladores que dan valor a estas señales de control pueden ser: analógico o digital.
El uso de los controladores digitales es cada vez mayor en el mercado debido a ciertas ventajas con respecto a los controladores analógicos:

- Implementación de algoritmos de control más complejos.
- Modificación del algoritmo de control sin necesidad de realizar ningún cambio sobre el hardware.
- Disminución del tiempo de diseño.
- Aumento de la fiabilidad del sistema.
- Disminución coste.

Sin embargo, existe un gran inconveniente en controladores digitales: trabajan con señales analógicas que deben ser digitalizadas para que puedan ser usadas. Para ello, se deben emplear convertidores DC/AC (convierte corriente continua en corriente alterna) y AC/DC (convierte corriente alterna en corriente continua).

Como en todos los sistemas, después de ser implementado debe ser probado. Sin embargo, no es una tarea trivial llevar a cabo la verificación de un control digital junto con una planta analógica (convertidor de potencia) debido al sistema mixto analógica-digital.

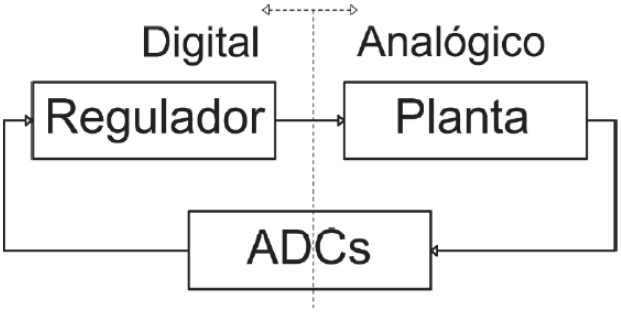

Figura 1. Esquema analógico-digital de convertidor de potencia

Como el convertidor conmutado se trata de un sistema de potencia, y por tanto, tiene la capacidad de manejar gran cantidad de energía, cualquier fallo que pueda haber puede generar daños en el circuito o en el personal por aumento no deseado de señales de tensión y corriente. Por esta razón, es necesario realizar simulaciones previas del controlador digital para comprobar que funciona correctamente.

Para ello, se debe emplear una técnica llamada *Hardware In the Loop* (HIL), la cual crea un modelo digitalizado del circuito analógico y lo ejecuta en hardware para mejorar el rendimiento de la simulación. Así, se consiguen emulaciones en tiempo real con tiempos de integración de hasta decenas de nanosegundos [1,2].

Las ventajas principales de un sistema HIL frente a otras técnicas son:

- Simular previamente y en paralelo con el desarrollo de la planta. Así, se van sustituyendo a medida que se vayan encontrando disponibles las partes simuladas por las que ya se encuentren implementadas físicamente.
- Evitar dañar la planta real cuando se somete a situaciones extremas a la planta basada en HIL.
- Reducir el coste de desarrollo.
- Evitar el coste de verificación y averías de un sistema real.

Esta técnica se emplea en la etapa de verificación de cualquier tipo de controlador digital junto con el modelo de planta correspondiente.

En este Trabajo de Fin de Máster se va a desarrollar el modelo de planta de un convertidor Flyback. Este convertidor, que es un derivado de la topología *buck-boost*, es de tipo DC a DC y el cual aísla eléctricamente la entrada y la salida gracias al uso de un transformador (Figura 2).

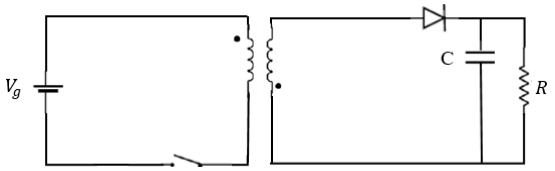

Figura 2. Topología del convertidor Flyback

Este transformador, además de ser usado para almacenar energía, permite que esta topología tenga capacidad tanto de elevador (convertidor *Boost*) como de reductor de tensión (convertidor *Buck*), es decir, que puede aumentar y disminuir la tensión de salida del circuito respecto a la de la entrada.

Una vez desarrollado el modelo matemático y digitalizado en lenguaje VHDL de la topología Flyback, se consigue una copia de la planta. La gran ventaja de esta réplica es que si ocurre algún fallo simplemente se obtendrá un número de bits distintos a los esperados a la salida del circuito.

En este proyecto se realiza el proceso de digitalización de un convertidor Flyback [3] con pérdidas para que así se pueda aplicar la técnica HIL sobre éste. Se implementa con pérdidas, ya que en cualquier circuito eléctrico real existen pérdidas. Por esta razón, se debe realizar un estudio para observar cómo afectan estas pérdidas a la simulación.

1.2 Objetivos

Este Trabajo de Fin de Máster tiene como objetivo desarrollar, simular y emular un modelo HIL para un convertidor Flyback considerando resistencias parásitas y pérdidas eléctricas de los elementos que forman parte del convertidor.

Para ello, se van a desarrollar en lenguaje VHDL y simular (empleando programa *ModelSim*) los modelos de planta del convertidor Flyback:

- *Simulink* de *Matlab* sin y con pérdidas.
- Real sin y con pérdidas.
- Coma fija sin y con pérdidas.
- Float32 sin y con pérdidas.

Una vez desarrollados y probados todos los modelos, se realizan las siguientes comparativas:

- Comparativa de los resultados obtenidos de forma teórica, en Simulink y en ModelSim con formato tipo real del modelo desarrollado del convertidor Flyback sin pérdidas. En esta comparativa validaremos nuestro modelo de referencia, que será el modelo con formato real.
- Comparativa del modelo del convertidor Flyback con pérdidas eléctricas cuando se obtienen los resultados de forma teórica, en Simulink y en ModelSim con formato tipo real. Se validará también el modelo de referencia.
- Comparativa de los modelos en diferentes formatos sin y con pérdidas y también entre ellos para comprobar cuál es el efecto de los formatos.

1.3 Organización de la memoria

Este Trabajo de Fin de Máster consta de los siguientes capítulos:

- En el primer y segundo capítulo trata la motivación y objetivos que me han llevado a realizar este trabajo, y el estado del arte.

- El tercer capítulo consta del modelado matemático del convertidor Flyback sin pérdidas, las distintas aritméticas que se emplean para desarrollar los distintos modelos, y la implementación de éstos en lenguaje VHDL. Además, se explica cómo se implementa el circuito del convertidor Flyback en la herramienta Simulink cuando no hay pérdidas.

- En el cuarto capítulo se indica el estudio realizado para obtener las pérdidas eléctricas de los distintos elementos del circuito del convertidor Flyback, su modelado matemático, las distintas aritméticas que se emplean para desarrollar los distintos modelos, y la implementación de éstos en lenguaje VHDL. Además, se explica cómo se implementa el circuito del convertidor Flyback en la herramienta Simulink cuando hay pérdidas.

- En el quinto capítulo se explican las condiciones iniciales empleadas en las simulaciones de los modelos, las comparativas entre el modelo real sin y con pérdidas y Simulink sin y con pérdidas. Además, se realiza una comparativa del modelo real con el modelo en coma fija y en coma flotante y se obtienen los errores medios de cada uno de los cotejos realizados.

- En el sexto capítulo se presentan las conclusiones de este proyecto y las líneas futuras.

2 Estado del arte

En los últimos años se ha presenciado un salto bastante importante de los sistemas analógicos a los digitales. La industria ha ido adaptando el crecimiento tecnológico hacia un sistema digital cada vez más propagado. Este cambio proporciona mayor flexibilidad y mayores posibilidades. Aun así, en nuestro día a día, se siguen empleando sistemas analógicos importantes como los convertidores de potencia.

Los convertidores de potencia son los encargados de transformar y controlar las señales de tensión y corriente a la entrada para conseguir valores precisos a su salida. En electrónica los tipos más habituales de conversión son [4]:

- **DC a DC**
  Esta transformación se trata de circuitos que tienen la capacidad de modificar niveles de voltaje empleando elementos como bobinas y condensadores. Se almacena energía en ellos temporalmente y se descargan consiguiendo así los niveles de voltaje deseados a la salida del circuito.
- **AC a DC**
  En esta transformación se emplean rectificadores, los cuales permiten convertir corriente alterna en corriente continua. También puede emplear fuentes de alimentación conmutada.
- **DC a AC**
  En esta transformación se emplean inversores, los cuales transfieren potencia desde una fuente de corriente continua a una carga de corriente alterna.
- **AC a AC**
  Esta transformación modifica la amplitud y la frecuencia de la señal de corriente alterna de forma arbitraria.

Las transformaciones de potencia se pueden llevar a cabo por: reguladores lineales o convertidores conmutados. Mientras que los reguladores lineales se basan en disipar energía empleando una resistencia variable, los convertidores conmutados emplean interruptores para permitir o impedir que pase la energía y así conseguir que la salida del circuito varíe.

Se suelen usar más convertidores conmutados que reguladores lineales ya que los segundos tienen menor eficiencia al haber disipadores en su interior. Gracias a los convertidores conmutados se obtienen los mismo resultados con mayor eficacia y con menores pérdidas por estar formados por elementos no disipativos. Además, estos convertidores pueden tener el papel de elevador o de reductor de potencia.

Los convertidores según el tipo de conmutación de los dispositivos dentro del convertidor se pueden clasificar en [5]:

- Conmutación natural (Convertidores de frecuencia de línea). Las tensiones de la alimentación presentes a un lado del convertidor permiten la conexión o desconexión de los dispositivos de semiconductores de potencia.
- Conmutación forzada (Convertidores de conmutación). Este tipo de convertidor necesitan las señales de control para abrir o cerrar los interruptores.
- Convertidores resonantes y cuasi resonantes. Este tipo de convertidor abre o cierra los interruptores si la tensión y/o corriente es cero.

Por tanto, según las descripciones realizadas, el convertidor de conmutación necesita de un controlador que genere las señales necesarias para poder abrir o cerrar los interruptores y así obtener la salida deseada del circuito.

Este tipo de control puede ser analógico o digital. Antes, para regular los convertidores conmutados se utilizaban los controles analógicos.
Sin embargo, a lo largo de los años ha aumentado el uso de controladores digitales frente a los controladores analógicos por numerosas ventajas, aunque también tiene algunas desventajas [6].

Los controladores analógicos se implementan con elementos como amplificadores, resistencias o condensadores. Sus principales ventajas son: elevado ancho de banda, alta resolución y facilidad al diseñar y verificar. Sin embargo, tiene algunos inconvenientes como: envejecimiento de componentes y que varía con las condiciones del entorno [6].

Por otro lado, los controladores digitales se encuentran implementados por microprocesadores o microcontroladores, entre otros, y necesitan los conversores ADC (conversión analógica a digital) o DAC (conversión digital a analógica) para comunicarse con la planta. Este tipo de controlador es de diseño programable, es decir, permite que se modifique sin variar el hardware. Además, tiene la capacidad de implementar algoritmos complejos, son más eficaces, menos costosos, mayor fiabilidad del sistema y más estable cuando ocurren cambios en el entorno [7]. Sin embargo, necesita procesadores de altas prestaciones.

Aunque los beneficios de los controladores digitales son más atractivos que las de los analógicos, hay casos en los que es mejor utilizar los analógicos que los digitales.

Existen dos grupos de convertidores en el mercado: convertidor no aislado o convertidor aislado [8]. Los convertidores no aislados no contienen la etapa de aislamiento. En este grupo se encuentran: convertidor *Buck* (de reducción), convertidor *Boost* (de elevación), convertidor *Buck-Boost* (reducción-elevación) y convertidor *full-bridge* (convertidor de puente completo). Por otro lado, entre los convertidores aislados se encuentran: convertidor Flyback, convertidor directo y *Cuk* aislado. Este tipo de convertidores son derivados de los convertidores no aislados a los que se les añaden una etapa de aislamiento.
Por ejemplo, el convertidor Flyback es un derivado del convertidor *Buck-Boost* (reductor-elevador).

Estos convertidores después de ser implementados deben ser probados. Sin embargo, no es una tarea trivial ya que éstos están compuestos por un control digital (regulador) y una planta analógica (convertidor de potencia).

Como el convertidor conmutado se trata de un sistema de potencia, y por tanto, tiene la capacidad de manejar gran cantidad de energía, cualquier fallo que pueda haber puede generar daños en el circuito o en el personal por aumento no deseado de señales de tensión y corriente. Por esta razón, es necesario realizar simulaciones previas del controlador digital para comprobar que funciona correctamente.

Para que no ocurran este tipo de accidentes se han desarrollado varios estudios sobre diversas simulaciones que se pueden llevar a cabo en sistemas mixtos [9].

Por otro lado, en [10] se simulan tres modelos distintos: uno desarrollado en Simulink Matlab, otro en Verilog: lenguaje de descripción de hardware (*Hardware Description Language* (HDL)), el cual se emplea para modelar sistemas electrónicos, y el último en *Spice*.

Por un lado, si se realiza una prueba con el primero de ellos se pueden obtener resultados con gran precisión y se pueden simular a gran velocidad, pero tiene un problema: el modelo que se ha diseñado en esta herramienta no corresponde con el que se va a implementar en hardware. Por otro lado, se realiza una nueva prueba, pero en este caso con el segundo modelo comentado, con un simulador mixto: *Cadence Spectre Virtuoso*. Con esta simulación se quiere probar un controlador escrito en HDL junto al esquemático del convertidor, pero hay un inconveniente: tiempo de simulación bastante alto. Finalmente, se lleva a cabo la última simulación empleando modelos desarrollados en *Spice*. En este caso, también el tiempo de simulación es muy grande por cada ciclo de conmutación.

Como hoy en día existen pocos simuladores mixtos analógico-digital, una de las soluciones que se pueden emplear es integrar conjuntamente estos dos simuladores [11]: uno para la parte analógica (*PSIM*) y otro para la parte digital (*ModelSim*). Sin embargo, hay un inconveniente y es que enlazar estos dos programas no es tarea trivial por factores como flujo de re-alimentación entre aplicaciones, sincronización de datos, etc

Existe otro tipo de simulación cuyo objetivo es, a partir de las ecuaciones diferenciales que actúan sobre el comportamiento del circuito, transformar la planta en lenguaje HDL.
En [12] se realiza una comparación entre: VHDL-AMS y VHDL.
Mientras que VHDL-AMS se trata de un lenguaje que se emplea en circuitos de señal mixta, VHDL se emplea sólo en circuitos que son digitales. Cuando se realizan las simulaciones de dos modelos, uno desarrollado en VHDL-AMS y el otro en VHDL, se comprueba que el tiempo de ejecución del VHDL es mucho menor que el de VHDL-AMS.

Para evitar que ocurra lo de este último caso, es decir, que el tiempo de simulación sea tan alto, se debe emplear el sistema HIL (*Hardware In-the-Loop*), el cual consiste en digitalizar la planta analógica mediante un modelo matemático para, a continuación, llevar a cabo su ejecución en hardware y así, poder realizar pruebas junto al control digital. Gracias a este proceso, el rendimiento de la simulación mejora. Actualmente se está empleando mucho esta técnica en simulación de convertidores, en las ramas de la ingeniería y en el campo de la automatización. Además, ha sido probado con éxito en trabajos como en [13].

Hay empresas que invierten en esta técnica como son *dSPACE* o *Typhoon HIL*. La empresa *dSPACE* es el productor líder de herramientas de ingeniería para desarrollar y probar sistemas de control mecatrónico. Tienen una gran demanda como socios de desarrollo en la industria automotriz, aeroespacial y en automatización industrial.Son los encargados de proporcionar diversos productos de hardware y software para simulaciones HIL. Estas simulaciones implican la operación de sistemas mecatrónicos en un circuito cerrado con componentes que se simulan en tiempo real para probarlos de forma intensiva en este entorno virtual [14]. En particular, dentro de los sistemas mecatrónicos se emplean las unidades de control electrónico (ECU), las cuales están compuestas por sensores y actuadores donde los sensores son los encargados de informar a la unidad central y ésta envía la orden a los actuadores para transformar la información inicial.

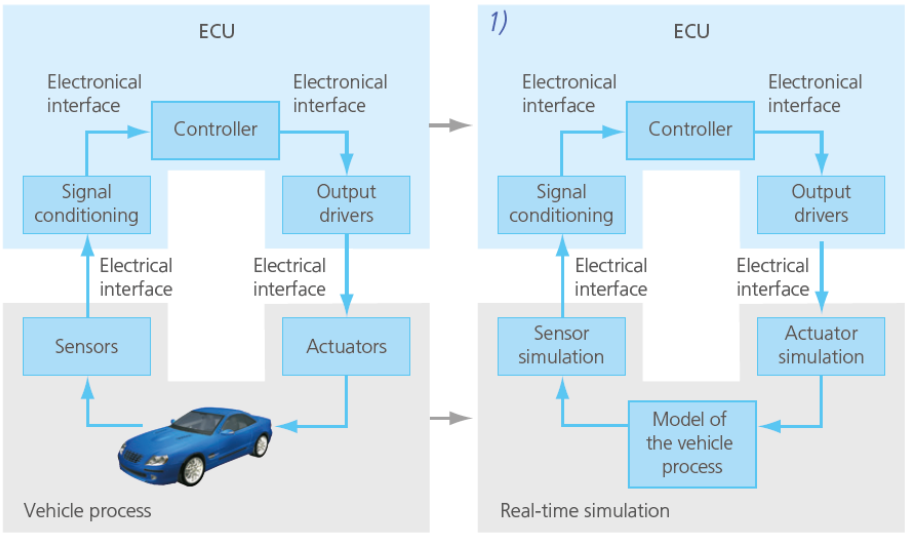

Figura 3. Flujos de señal en un sistema real (izq) y en simulación HIL (der) [14]

Por otro lado, *Typhoon HIL* es una empresa de electrónica de potencia que desarrolla herramientas de simulación en tiempo real, sistemas de pruebas de Hardware in the loop (HIL) y prototipos rápidos de control basados en DSP (*Digital Signal Processor*) para aplicaciones en sistemas de almacenamiento de energía solar, eólica y energética. Las herramientas de esta empresa ofrecen una experiencia de usuario única, libre de las complejidades de hardware y software de terceros: el software, con todas las bibliotecas, se instala con un solo clic y los modelos se compilan en segundos. [15].

En este proyecto se va a desarrollar un modelo de planta de un convertidor Flyback que pueda ser implementado en una FPGA. Para poder ejecutar el diseño del modelo en esta placa se va a utilizar el lenguaje VHDL. Se debe tener en cuenta la longitud de las señales ya que éstas influyen en los recursos que se consumen y en el tiempo de simulación. Una vez definido el modelo matemático se debe elegir la notación a emplear en las señales.

A continuación, además de explicar las posibles aritméticas que se pueden aplicar al modelo del convertidor Flyback, se va a describir la herramienta Simulink, la cual se emplea para comprobar que el modelo real sin y con pérdidas, el cual es el modelo de referencia, es correcto.

2.1 Posibles aritméticas

En las librerías del lenguaje VHDL se encuentran varios tipos de datos [2] como son: real (*real*), coma fija (*sfixed*) o coma flotante (*float*):

- Real
  Las señales de tipo *real* se suelen usar por lo fácil que es implementar un modelo en este estándar y por su gran resolución numérica, ya que usa coma flotante de 64 bits. Además, no es necesario hacer ningún tipo de modificación en los tamaños de las señales.
  Como los diseños con estas señales son bastante sencillos de desarrollar, éstos se pueden comparar con otros modelos que sí se pueden sintetizar en FPGA.

Éste es un tipo de dato que puede ser desarrollado y simulado con herramientas de simulación VHDL como ModelSim, pero que no puede ser implementado en una FPGA, es decir, que no puede ser sintetizado en hardware real.
En nuestro caso, esta aritmética va a ser el referente para el resto de modelos, pero no va a ser el definitivo ya que, como se ha comentado anteriormente, no se puede llevar a cabo su implementación en una FPGA.

- Coma fija

  Este tipo de aritmética se puede simular y puede ser sintetizada en una FPGA, es decir, que se pueden llevar a cabo emulaciones en tiempo real y no sólo de simulaciones software.
  La emulación es mucho más rápida que la simulación, consiguiendo reducir el tiempo de pruebas del convertidor de potencia.
  Por otro lado, con este tipo de dato, el diseñador debe definir los tamaños óptimos para cada una de las señales del modelo e indicar el formato que deben tener éstas.
  Estas señales se pueden describir con la siguiente notación: QX.Y. Como se observa en la Figura 4, se asigna 1 bit de signo para conocer si el número es positivo o negativo, X bits que se deben asignar a la parte entera e Y bits que se deben asignar a la parte decimal. Además, se encuentran escritos en complemento a 2. Por tanto, si una señal tiene un formato Q3.4, el número total de bits es de 8 (1+3+4).

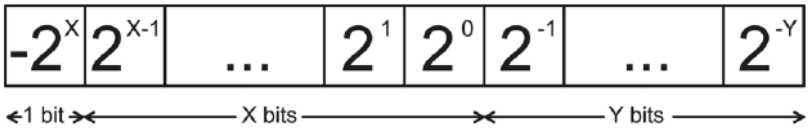

Figura 4. Formato Q X.Y

Este tipo de aritmética debe tener una resolución que permita que se puedan obtener los resultados deseados. Con este formato los tamaños de cada señal deben ser decididos para cada caso concreto.

- Coma flotante

  Este tipo de aritmética no es soportada por todos los sintetizadores y se utilizan mucho más recursos hardware para llevar a cabo las implementaciones de las operaciones en coma flotante que en coma fija.
  Por otro lado, este tipo de dato no requiere ninguna definición de los tamaños de las señales del modelo, lo que significa que el diseñador no debe hacer un esfuerzo extra.
  Este tipo de datos emplea señales de 32 bits, de los cuales el primer bit es para el signo (s), los siguientes 8 bits corresponden al exponente (e) y los 23 bits restantes forman la mantisa (m) [16]:

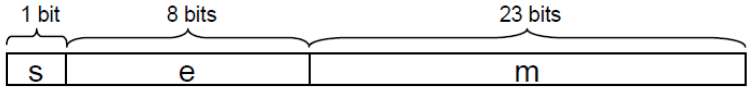

Figura 5. Formato coma flotante [16]

El bit del signo (s) se encuentra en el bit 31, el exponente empieza en el bit 23 y acaba en el bit 30, y la mantisa comienza en el bit 0 hasta el bit 22.

Esta aritmética tiene una ventaja y es que la posición de la coma en cada señal del modelo se adapta automáticamente. Sin embargo, está formado por un hardware bastante complejo, el área utilizada es muy grande y su notación es más lenta de procesar y menos precisa que la de coma fija.

2.2 Simulink

Simulink, anteriormente llamado *Simulab*, se integra con el lenguaje de programación Matlab. Esta herramienta emplea los algoritmos de resolución de Matlab y su potencia de cálculo para ofrecer soluciones numéricas a todos los modelos que se hayan implementado. Actualmente, las librerías de *Simulink* presentan numerosas aplicaciones en el procesamiento de señales y en los componentes eléctricos [17].

Simulink es un entorno de programación visual, el cual funciona sobre el entorno de programación *Matlab*. Esta herramienta se suele emplear en los dispositivos electrónicos, ya que permite llevar a cabo simulaciones sencillas y precisas. Además, contiene, entre otras cosas, librerías con las definiciones de elementos simples como resistencias, bobinas y condensador y elementos más complejos como transformadores o generadores de señal.

Por otro lado, una de las características de esta herramienta es que permite que se puedan exportar las señales obtenidas en un formato que pueda abrir *Matlab* para así poder realizar comparativas entre los valores obtenidos con esta herramienta y los valores que se han adquirido de los modelos que se han diseñado directamente con el lenguaje VHDL.

Para poder simular y realizar las pruebas oportunas en sistemas *Hardware In The Loop* (HIL) en *Simulink*, en primer lugar se debe simular empleando *Matlab* y *Simulink*, y a continuación, diseñar, construir, y probar la aplicación en tiempo real. Además, con *Simulink Real-Time* se pueden crear aplicaciones en tiempo real a partir de los modelos implementados en *Simulink*.

En este Trabajo de Fin de Máster se va a desarrollar, simular y emular un modelo HIL para un convertidor Flyback considerando resistencias parásitas y pérdidas eléctricas de los elementos que forman parte del convertidor. Se van a diseñar tres modelos con las siguientes notaciones: real, coma fija y coma flotante.

3 Modelado del convertidor Flyback

En este capítulo, se va a diseñar el modelo de planta de un convertidor de retroceso o convertidor Flyback. Éste es un derivado de la topología *buck-boost*, es de tipo DC a DC y se encarga de aislar eléctricamente la entrada y la salida gracias al uso de un transformador. Para llevar a cabo este diseño, en primer lugar, se va a desarrollar el modelo matemático de dicho convertidor y después, se va a realizar su implementación. Para realizar esto último, se ha empleado el lenguaje de programación VHDL.

En la Tabla 1 se muestran los valores que se van a emplear en este modelo sin pérdidas:

Componente	**Valor**
Bobina (L)	352 µH
Condensador (C)	440 µF
Ciclo de reloj (dt)	20 ns
Relación de transformación (n)	1
Ciclo de reloj	20 ns
Frecuencia de conmutación (f_{sw})	50 KHz
Tensión de entrada (v_g)	110V
Tensión de salida (v_{out})	48V
Potencia de salida (P_{out})	50W
Resistencia de carga (R)	46,08 Ω

Tabla 1. Constantes y señales del circuito

La resistencia de carga se obtiene de la siguiente forma:

$$P_{out} = v_{out} * I \quad (1)$$

$$P_{out} = v_{out} * \left(\frac{v_{out}}{R}\right) \quad (2)$$

$$R = \frac{{v_{out}}^2}{P_{out}} = \frac{48^2}{50} = 46{,}08\ \Omega \quad (3)$$

3.1 Modelo matemático

Para llevar a cabo el modelado matemático de un convertidor Flyback es necesario obtener aquellas ecuaciones que definen los elementos de dicho convertidor.

Las ecuaciones que relacionan la tensión que cae en los bornes de dichos elementos (4) y la corriente que pasa por ellos (5) son las que se definen a continuación:

- Definición de la bobina

$$v_L = L * \frac{di_L}{dt} \quad (4)$$

- Definición del condensador

$$i_c = C * \frac{dv_c}{dt} \quad (5)$$

A partir de éstas, se obtienen las variables de estado: la corriente en la bobina (i_L) y la tensión del condensador (v_c). Ambos elementos se encuentran definidos en el circuito del convertidor Flyback de la siguiente forma (Figura 6): el condensador se observa en la parte derecha del circuito junto al diodo y el transformador y la bobina forma parte del transformador empleado:

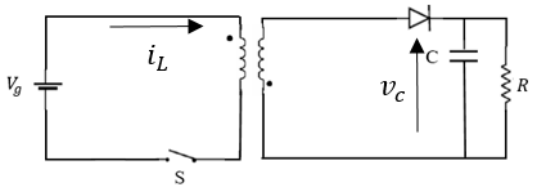

Figura 6. $\boldsymbol{i_L}$ y $\boldsymbol{v_c}$ de la topología Flyback

Para llevar a cabo la digitalización del modelado matemático, se deben convertir las ecuaciones de las variables de estado en ecuaciones en diferencias:

- Bobina

$$i_L(k) = i_L(k-1) + v_L(k) * \frac{\Delta t}{L} \quad (6)$$

- Condensador

$$v_c(k) = v_c(k-1) + i_c(k) * \frac{\Delta t}{C} \quad (7)$$

Una vez que se han definido las variables de estado, se va a analizar el comportamiento que puede tener el circuito cuando varía el estado del interruptor y del diodo.
Mientras que el interruptor puede estar abierto o cerrado, el diodo puede:

- conducir (CCM, *Continuous Conduction Mode*): ocurre cuando la corriente atraviesa al diodo desde su ánodo (+) hasta su cátodo (-).
- no conducir (DCM, *Discontinuous Conduction Mode*): ocurre cuando la corriente que atraviesa al diodo es nula.

Con las ecuaciones en diferencias definidas y con los distintos estados que puede tomar el circuito, se pueden obtener las ecuaciones que definen el comportamiento del circuito para cada instante. Además, se debe tener en cuenta que el transformador, el cual se encuentra en el circuito del convertidor Flyback, permite que la entrada y salida de éste se encuentren aisladas.

Por tanto, cuando el interruptor se encuentra cerrado (encendido), se produce el almacenamiento de energía del transformador a través de su inductancia magnetizante (L_m). Además, la carga almacenada en el condensador se libera hacia la carga y el diodo se encuentra polarizado inversamente:

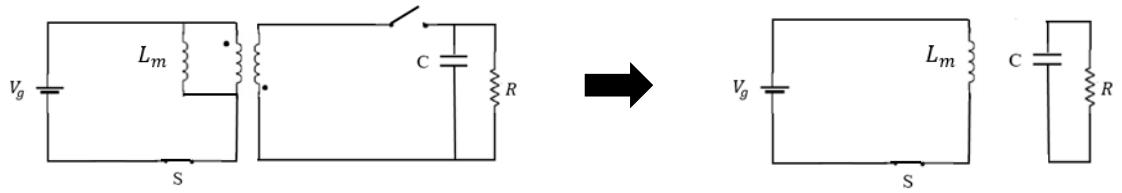

Figura 7. Circuito Flyback con Q= ON

En este caso, se observa que la corriente que atraviesa la inductancia magnetizante (i_{Lm}) es igual a la corriente de entrada (i_{in}) y por tanto se obtiene que:

$$\Delta i_L = v_g * \frac{\Delta_t}{L} \quad (8)$$

Además, como la tensión del condensador (v_c) es igual a la tensión de salida del circuito (v_{out}), se obtiene la siguiente ecuación:

$$\Delta v_c = -i_R * \frac{\Delta_t}{C} \quad (9)$$

Por otro lado, si el interruptor se encuentra abierto (apagado), el diodo entra en conducción y permite que la energía almacenada anteriormente en el transformador se pueda descargar:

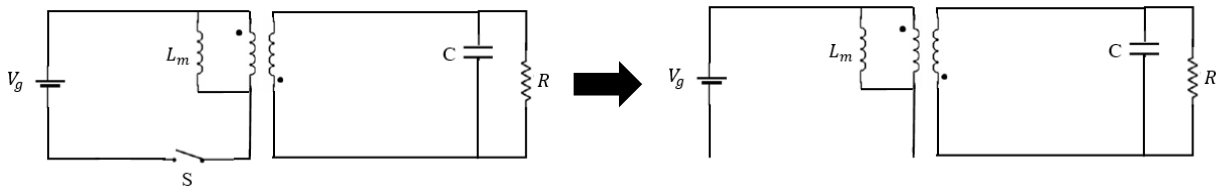

Figura 8. Circuito Flyback con Q=OFF

En este caso, las ecuaciones de estado de la bobina y el condensador son las siguientes:

$$\Delta i_L = \frac{-v_{out}}{n} * \frac{\Delta_t}{L} \quad (10)$$

$$\Delta v_c = (\frac{i_L}{n} - i_R) * \frac{\Delta_t}{C} \quad (11)$$

También, puede ocurrir el caso en el que el interruptor esté abierto y el diodo se encuentre polarizado inversamente:

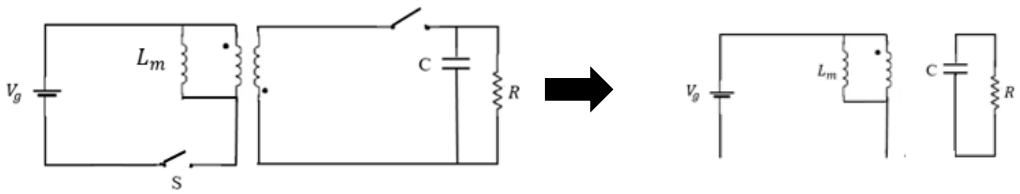

Figura 9. Circuito Flyback con Q=OFF y diodo en DCM

Las ecuaciones de las variables de estado en este caso son las siguientes:

$$\Delta i_L = 0 \tag{12}$$

$$\Delta v_c = -i_R * \frac{\Delta_t}{C} \tag{13}$$

Se ha definido el modelo matemático del circuito de un convertidor Flyback, tras deducir las ecuaciones que definen las variables de estado (i_L; v_c) y las ecuaciones de sus incrementos (Δi_L; Δv_c), las cuales se obtienen dependiendo del estado del circuito.

A modo resumen, en la Tabla 2 se muestran todas las ecuaciones obtenidas anteriormente:

Estados circuito		$i_L(k)$	$v_c(k)$
Q	**Diodo**		
ON	CCM	$i_L(k-1) + v_g * \frac{\Delta_t}{L}$	$v_c(k-1) - i_R * \frac{\Delta_t}{C}$
OFF	CCM	$i_L(k-1) - \frac{v_{out}}{n} * \frac{\Delta_t}{L}$	$v_c(k-1) + (\frac{i_L}{n} - i_R) * \frac{\Delta_t}{C}$
	DCM	$i_L(k-1) + 0 * \frac{\Delta_t}{L}$	$v_c(k-1) - i_R * \frac{\Delta_t}{C}$

Tabla 2. Ecuaciones de $\boldsymbol{i_L}$ (k) y $\boldsymbol{v_c}$ (k) en función del estado del circuito

3.2 Régimen permanente

Si el convertidor opera en condiciones estacionarias, la cantidad de energía que se encuentra almacenada en la bobina debe ser la misma al inicio y al final del ciclo completo de conmutación. Esto es lo que se llama régimen permanente.

Para conocer como es el comportamiento del circuito del convertidor Flyback cuando se encuentra en equilibrio, se van a emplear las ecuaciones de estado que se han obtenido en el apartado 3.1.

Para obtener las ecuaciones en equilibrio de este convertidor, se debe tener en cuenta que debe operar en régimen permanente y que el diodo debe conducir en modo de conducción continua (CCM).

Así, se define en forma de la siguiente ecuación, donde d_{ON} indica el tiempo en el que el transistor se encuentra encendido y d_{OFF} el tiempo en el que éste se encuentra apagado:

$$\Delta I_{ON} + \Delta I_{OFF} = 0 \quad (14)$$

$$v_g * \frac{\Delta_t}{L} * d_{ON} - \frac{v_{out}}{n} * \frac{\Delta_t}{L} * d_{OFF} = 0 \quad (15)$$

$$v_g * d_{ON} - \frac{v_{out}}{n} * d_{OFF} = 0 \quad (16)$$

Si tiene en cuenta que: $d_{OFF} = 1 - d_{ON}$, ya que se considera que el ciclo de trabajo se encuentra normalizado a la unidad. A partir de ahora, en vez de d_{ON} se indicará d:

$$v_g * d - \frac{v_{out}}{n} * (1 - d) = 0 \quad (17)$$

$$v_g * d = \frac{v_{out}}{n} * (1 - d) \quad (18)$$

$$\frac{v_{out}}{v_g} = \frac{d*n}{(1-d)} \quad (19)$$

Con este proceso se ha obtenido el comportamiento del convertidor Flyback en régimen permanente, el cual depende del ciclo de trabajo (d).

3.2.1 Tiempo de encendido y apagado transistor

Para saber cuánto tiempo debe estar el interruptor (Q) abierto y cuanto cerrado se deben realizar los siguientes cálculos.

Si se sustituyen los valores según la Tabla 1, se obtiene el ciclo de trabajo:

$$\frac{48}{110} = \frac{d*1}{(1-d)} \quad \rightarrow \quad (20)$$

$$0{,}43636363 = \frac{d*1}{(1-d)} \quad \rightarrow \quad (21)$$

$$d = 0{,}43636363 * (1 - d) \rightarrow \quad (22)$$

$$d = 0{,}303797 = 30{,}37\% \approx 30{,}4\ \%$$

Una vez obtenido este valor, se debe calcular el tiempo que el interruptor (Q) debe estar cerrado (estado ON):

$$tiempoON = \frac{d}{f_{sw}} = \frac{0{,}304}{50e3} = 6{,}0759\ \mu s \rightarrow 6{,}08\ \mu s \quad (23)$$

Este valor se redondea a 6,08 µs ya que $tiempoON\ y\ tiempoOff$ deben ser múltiplo del periodo de conmutación, que es de 20 µs.

Por otro lado, el tiempo que el interruptor (Q) debe estar abierto (estado OFF):

$$tiempoOFF = t_{sw} - tiempoON = \frac{1}{50e3} - 6{,}08 = 13{,}92\ \mu s \quad (24)$$

3.3 Implementación del modelo

Tras llevar a cabo la elaboración del modelo matemático, éste se debe implementar en lenguaje VHDL. Antes de nada, se debe elegir la aritmética que se va a emplear.
En este caso se van a diseñar los siguientes modelos: uno que utilice señales reales, otro con señales en coma fija (QX.Y), empleando la biblioteca *sfixed* y otro con señales en coma flotante, empleando la biblioteca *float*.

En primer lugar se va a explicar el modelo del convertidor Flyback en la herramienta *Simulink* cuando no hay pérdidas, el cual se emplea para poder corroborar que el modelo real desarrollado es correcto. A continuación, se va a llevar a cabo la implementación del modelo real, el modelo en coma fija y el modelo en coma flotante para, finalmente, poder realizar una comparativa entre los 3 modelos.

3.3.1 Modelo Simulink

En la Figura 10, se muestra como se ha diseñado el convertidor Flyback sin pérdidas con los elementos propios de la herramienta *Simulink*, siguiendo el esquema que se muestra en Figura 2. Por tanto, se han empleado los siguientes elementos: una fuente de alimentación, la cual corresponde con la tensión de entrada del circuito, un mosfet que corresponde al interruptor indicado anteriormente, un transformador inductivo, un diodo, un condensador y una resistencia, que corresponde con la resistencia de salida del circuito.

Además de estos elementos, también se observa un sensor de corriente (*Current Measurement),* el cual representa un amperímetro que mide la corriente sólo en la parte del circuito donde se encuentra, un sensor de tensión (*Voltage Measurement*), el cuál es un voltímetro que mide la tensión que hay a la salida del circuito. Otros elementos que se han empleado son *scope,* que se emplea para representar las señales, en este caso, de la corriente de entrada del circuito y la tensión de salida de éste y *Pulse Generator*, el cual es una señal modulada como una amplitud de 1, un periodo de conmutación de 20 µs y un ancho de pulso de 30,4%. Este ancho indica que el tiempo de apertura del mosfet es de 6,08 µs y el tiempo de apagado es de 13,92 µs.

Para poder resolver este circuito, se ha empleado una resolución *Backward Euler* con paso temporal de $1 * 10^{-7}$. Esta es la configuración que suele venir por defecto.

Por otro lado, los valores que se han empleado en este diseño son los que se muestran en la Tabla 1.

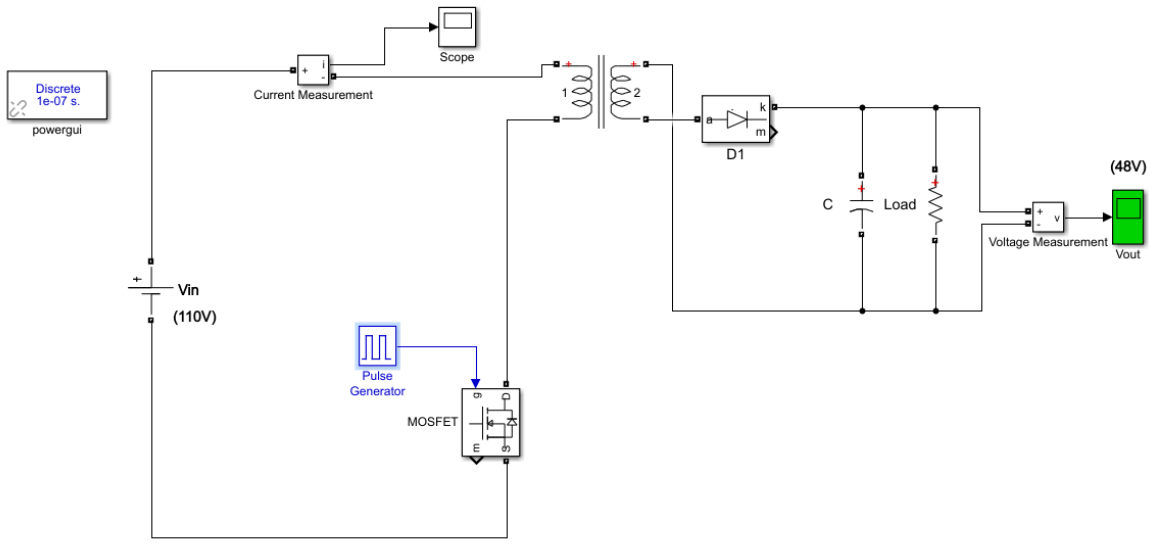

Figura 10. Modelo Flyback sin pérdidas en Simulink

3.3.2 Modelo real

En este modelo se implementan las ecuaciones del convertidor Flyback en el apartado 3.1 con el tipo de datos *real*. Su diseño es sencillo ya que solo es necesario declarar las señales, pero cabe destacar que no puede ser sintetizado en una FPGA.
El código desarrollado se divide en dos procesos: uno combinacional y el otro secuencial.

Por un lado, el proceso combinacional ocurre cuando se modifica alguna señal que se encuentra dentro de su lista de sensibilidad. Es conveniente recordar que si hay una señal que no se encuentra dentro de esta lista y cambia de valor, no va a variar ningún valor que dependa de ella. En este proceso, se hace referencia tanto a la apertura como al cierre del interruptor (Q) como al estado del diodo cuando el interruptor se encuentra abierto (CCM o DCM) a través del valor de i_L.

Por otro lado, el proceso secuencial tiene en su lista de sensibilidad tanto la señal del reloj como la señal de reset. Por tanto, este proceso ocurre cuando se activa la señal de reloj de forma síncrona o la señal de reset de forma asíncrona. Este proceso es el encargado de asignar valores a las ecuaciones de estado.

En la Figura 11 se muestra el esquema que sigue el modelo *real*. Como se puede observar, es un diagrama sencillo, con dos flips-flops que permiten que se registren las variables de estado en cada ciclo de reloj. Además, se emplean 3 multiplexores, 3 sumadores y 4 multiplicadores.

En la parte superior del diagrama se muestra el cálculo de la corriente que atraviesa la bobina (i_L), siguiendo las ecuaciones que se han obtenido en el apartado 3.1. En el primer multiplexor se obtiene el valor de la señal i_{LAdd} dependiendo del valor que se seleccione, y junto con la señal dt/L se obtiene el incremento de la variable de estado i_L. Este incremento junto con los valores de la señal de estado i_L en el instante anterior genera el valor de la ecuación en el instante actual.

Por otro lado, en la parte inferior del esquema se muestra el cálculo de la tensión que hay en el condensador (v_c), siguiendo las ecuaciones que se han obtenido en el apartado 3.1. En el

tercer multiplexor se obtiene el valor de la señal $v_{outAuxAdd}$ dependiendo del valor que se seleccione, y junto con la señal dt/C se obtiene el incremento de la variable de estado v_{outAux}. Este incremento junto con los valores de la señal de estado v_{outAux} en el instante anterior genera el valor de la ecuación en el instante actual.

Finalmente, la tensión del condensador que coincide con la tensión de salida del circuito se registra en el instante actual mediante un proceso secuencial, y la corriente que atraviesa la bobina se registra para, a continuación, pasar por un multiplexor, el cual selecciona el valor de la corriente de entrada (I_{in}). El desarrollo de este modelo aparece en el Anexo I.

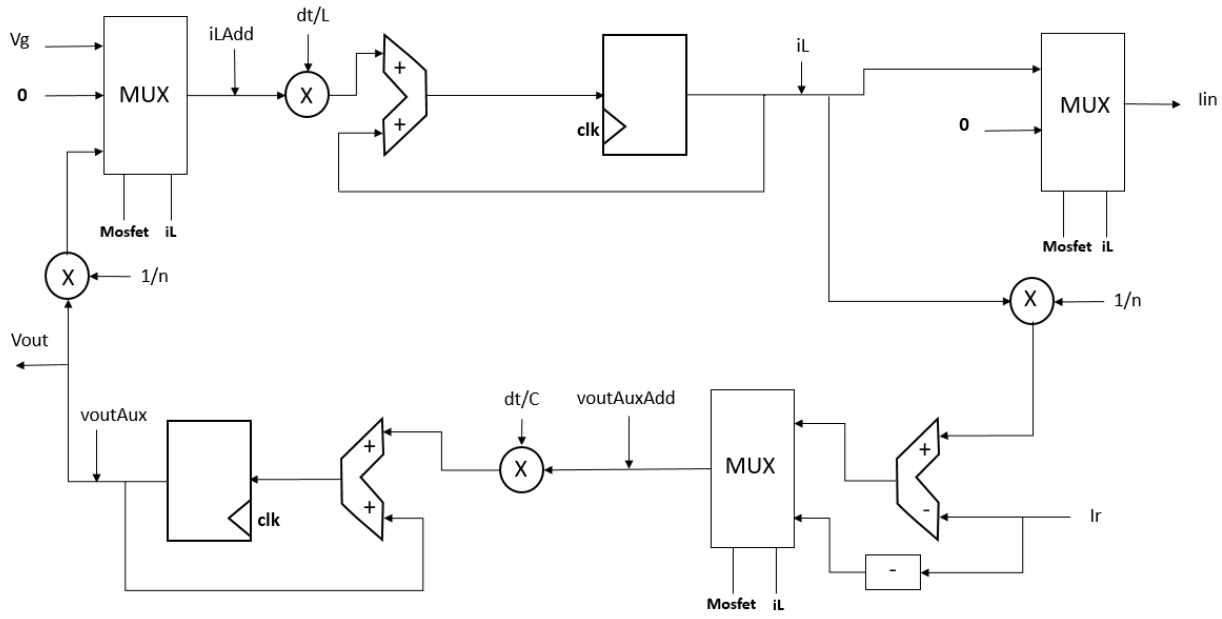

Figura 11. Esquema del convertidor Flyback en modelo real

3.3.3 Modelo coma fija

Coma Fija es un tipo de anotación que indica un número de bits fijo para la parte de entera y otro para la parte decimal.

Esta aritmética se caracteriza porque el diseñador debe elegir qué tamaño es más apropiado para cada señal del circuito y la posición de la coma. Este diseño es bastante más complejo que el modelo real ya que es necesario tener en cuenta cada operación y cada tamaño, pero es un modelo que se puede sintetizar en una FPGA y que permite que se consigan mejores resultados en cuanto a la frecuencia y al área.

Este modelo emplea la biblioteca *sfixed* del estándar VHDL2008, la cual realiza de forma automática el redondeo para evitar cualquier desbordamiento en los resultados de las operaciones aritméticas. Además, permite la traducción de un número en coma fija a VHDL. Aunque esta librería permita facilidades en cuanto a la etapa de implementación, sigue siendo necesario el diseño de las resoluciones y calcular el tamaño de todas las señales del circuito.

Para obtener cuantos bits se deben asignar a la parte entera (X), se debe calcular:

$$log_2(valor\_maximo) \tag{25}$$

Por otro lado, el número total de bits de la señal se calcula de la siguiente forma [18]:

$$width = log_2\left(\frac{X}{\Delta_X}\right) + n \tag{26}$$

, donde:

- X: máximo valor de la señal
- Δ_X: incremento de la señal
- n: número de bits usados para almacenar el incremento de la señal.

Así, si se conoce el número total de bits y el número de bits de la parte entera, se puede obtener el número de bits que se pueden asignar a la parte fraccionaria. Un problema principal de este modelo es que se deben calcular tanto los tamaños de las señales de las variables de estado como de cualquier señal auxiliar del modelo. Por esta razón, el diseño es mucho más complejo.

Además, para evitar desbordamientos se debe tener en cuenta que:

- En el resultado de las sumas y las restas se debe añadir un bit más en la parte entera por si hay acarreo:

$$\begin{array}{r} Q\ A.B \\ +\ Q\ C.D \\ \hline Q\ \underbrace{(mayor\ parte\ entera + 1)}_{\text{X}}.\underbrace{(mayor\ parte\ decimal)}_{\text{Y}} \end{array}$$

Figura 12. Suma en coma fija

  Además, como se observa en la Figura 11, para obtener el resultado de una suma o una resta, se debe escoger de las partes enteras de ambos formatos la mayor, y lo mismo para la parte decimal.

- En las multiplicaciones, el tamaño del resultado es la suma del número de bits de las partes enteras y el número de bits de las partes decimales, y además se añade un bit a la parte entera por si hay algún desbordamiento (Figura 12):

$$\begin{array}{r} Q\ A.B \\ \text{x}\ \ Q\ C.D \\ \hline Q\ \underbrace{(A + C + 1)}_{\text{X}}.\underbrace{(B + D)}_{\text{Y}} \end{array}$$

Figura 13. Multiplicación en coma fija

En este modelo las señales de entrada son: la tensión de entrada (v_g) y la corriente de la resistencia de carga (I_r) y para las señales de salida son: la corriente de entrada del circuito

(I_{in}) y la tensión de salida del circuito (v_{outAux}). Para estas señales, se ha elegido un tamaño de 13 bits. Este tamaño normalmente vendría impuesto por los DACs o ADCs que se utilicen.

A continuación, se explica uno a uno los diferentes formatos elegidos en coma fija para cada una de las señales del circuito:

- dtL
  Esta constante se obtiene de la operación:

  $$\frac{dt}{L} = \frac{0{,}00000002}{0{,}000352} = 5{,}6812182 * 10^{-5}\ s/H \quad (27)$$

  Estos valores aparecen en la Tabla 1.

  El número de bits que se debe asignar a la parte entera es (25):
  $log_2(5{,}6812182 * 10^{-5}) = -14{,}1033$ → Se redondea el valor a infinito: -14

  Por otro lado, como esta señal es una constante, el número de bits que se deben asignar a la parte fraccionaria se calcula de la siguiente forma:

  $$-log_2(\Delta_x) + n \quad (28)$$

  Si se sustituyen los valores se obtiene la siguiente ecuación:

  $$-log_2(5{,}6812182 * 10^{-5}) + n = 14{,}1033 + n$$

  Primero se redondea el valor 14,1033 a 15 y luego se suma n=8 como se recomienda en [16]:

  15 + n = 15 +8= 23

  Por tanto, el tamaño calculado es: Q-14.23.
  El problema es que con este número de bits el resultado no sale exacto. Para solucionarlo, se aumentan el número de bits de la parte decimal a 26.
  Por tanto, el tamaño final de la constante dtL es: Q -14.26

- dtC
  Esta constante se obtiene de la operación:

  $$\frac{dt}{L} = \frac{0{,}00000002}{0{,}000440} = 4{,}545454545 * 10^{-5}\ s/F \quad (29)$$

  Estos valores aparecen en la Tabla 1.

  El número de bits que se debe asignar a la parte entera es (25):

  $log_2(4{,}545454545 * 10^{-5}) = -14{,}4252$ → Se redondea el valor a infinito: -14

  Por otro lado, como esta señal es una constante, el número de bits que se deben asignar a la parte fraccionaria se calcula empleando (28) y se obtiene lo siguiente:

$$-log_2(4{,}545454545 * 10^{-5}) + n = \ 14{,}4252 + n$$

Primero se redondea el valor 14,4252 a 15 y luego se suma n=8 (siempre el mismo):

15 + n = 15 +8= 23

Por tanto, el tamaño calculado es: Q-14.23.
El problema es que con este número de bits el resultado no sale exacto. Para solucionarlo, se aumentan el número de bits de la parte decimal a 34.
Por tanto, el tamaño final de la constante dtL es: Q -14.34

- i_L

Para calcular que números se deben asignar a la parte entera y a la parte decimal de esta señal, primero se debe obtener el número total de bits de la señal con la fórmula (26):

$$width = log_2\left(\frac{X}{\Delta_X}\right) + n$$

, donde:

- X= 54,892 V

  X es el máximo valor que se obtiene cuando se realiza la simulación real del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión de salida (v_{out}) es 0.

- $\Delta_X = i_{LAdd} * \frac{d_t}{L}$ (30)

  Los posibles valores que puede tomar la señal i_{LAdd} son:

  1. $v_g = 110\ V$
  2. $\frac{-v_{outAux}}{n} = \ -\frac{48}{1} = -48$ (31)

  Entre ambos valores, se debe elegir el menor de ellos en valor absoluto: 48.
  Por tanto, se sustituyen los valores en (30):

  $$\Delta_X = i_{LAdd} * \frac{d_t}{L} = 48 * \frac{0{,}00000002}{0{,}000352} = 2{,}727273 * 10^{-5}$$

- n=8

Sustituyendo los valores obtenidos:

$width = log_2\left(\frac{54{,}892}{2{,}727273e-5}\right) + 8 = 22{,}29$ → Se redondea el valor a infinito: 23

Ahora se calcula el número de bits que se deben asignar a la parte entera de la señal (25):

$log_2(54{,}892) = \ 5{,}77$ → Se redondea el valor a infinito: 6

Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 22-6=16

Así, el tamaño de la señal i_L es: Q6.16

Este tamaño no es el definitivo, ya que realizando pruebas se obtienen valores más parecidos a los del modelo real sin pérdidas si el tamaño final es: Q6.22.
Por tanto, el tamaño final de la señal i_L es Q6.22.

- v_{outAux}

Para calcular que números se deben asignar a la parte entera y a la parte decimal de esta señal, primero se debe obtener el número total de bits de la señal con la fórmula (26):

$$width = log_2\left(\frac{X}{\Delta_X}\right) + n$$

, donde:

- X= 94,0437 V

  X es el máximo valor que se obtiene cuando se realiza la simulación real del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión de salida (v_{out}) es 0.

- $\Delta_X = v_{outAuxAdd} * \frac{d_t}{C}$ (32)

  Los posibles valores que puede tomar la señal $v_{outAuxAdd}$ son:

  3. $-I_r = \frac{-v_{outAux}}{R} = \frac{-48}{46{,}088} = -1{,}04166667A$ (33)
  4. $\frac{i_L}{n} - I_r = -0{,}494465$ A (34)

  Posibles valores de la señal i_L:

  - Máximo valor
    $\frac{i_L}{n} - I_r = 54{,}892 - 1{,}0416667 = 53{,}8503333$

  - Mínimo valor
    $\frac{i_L}{n} - I_r = 0{,}54720 - 1{,}0416667 = -0{,}494465$

  Entre ambos valores, se debe elegir el menor de ellos en valor absoluto: 0,494465.
  Por tanto, se sustituyen los valores en (32) y se obtiene lo siguiente:

  $\Delta_X = v_{outAuxAdd} * \frac{d_t}{C} 0{,}494465 * \frac{0{,}00000002}{0{,}000440} = 2{,}247571214 * 10^{-5}$

- n=8

Sustituyendo los valores obtenidos:

$width = log_2\left(\frac{94{,}0437}{2{,}247571214e-5}\right) + 8 = 29{,}99$ → Se redondea el valor a infinito: 30

Ahora se calcula el número de bits que se deben asignar a la parte entera de la señal (25):

$log_2(94{,}0437) = 6{,}55$ → Se redondea el valor a infinito: 7

Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 29-7=22
Así, el tamaño de la señal v_{outAux} es: Q7.22
Este tamaño no es el definitivo, ya que realizando pruebas se obtienen valores más parecidos a los del modelo real sin pérdidas si el tamaño final es: Q7.25.
Por tanto, el tamaño final de la señal v_{outAux} es Q7.25.

- i_{LAdd}
  En el apartado donde se calcula el tamaño de la señal i_L, se indica que el valor máximo de la señal i_{LAdd} es 110 V.
  Por tanto, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(110) = 6{,}78$ → Se redondea el valor a infinito: 7

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28) empleando el incremento Δ_x calculado en el apartado de i_L:

  $-log_2(2{,}727273e-5) + n = 15{,}1621 + n$

  Primero se redondea el valor 15,1621 a 16 y luego se suma n=8:
  16+n=16+8=24

  Por tanto, el tamaño final de la señal i_{LAdd} es: Q7.24

- $v_{outAuxAdd}$
  En el apartado donde se calcula el tamaño de la señal v_{outAux}, se indica que el valor máximo de la señal $v_{outAuxAdd}$ es 53,8503333 V.
  Por tanto, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(53{,}8503333) = 5{,}75$ → Se redondea el valor a infinito: 6

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28) empleando el incremento Δ_x calculado en el apartado de v_{outAux}:

  $-log_2(2{,}247571214e-5) + n = 15{,}4412 + n$

  Primero se redondea el valor 15,4412 a 16 y luego se suma n=8:
  16+n=16+8=24

  Por tanto, el tamaño de la señal $v_{outAuxAdd}$ es: Q7.24

  Este tamaño no es el definitivo ya que esta señal depende del número de bits del resultado de i_L/n.
  Por tanto, el tamaño final es: Q9.23.

- v_g

  Esta señal es de entrada y se ha especificado que todas las señales de entrada del circuito deben tener un número total de13 bits.
  Como el valor máximo de esta señal es 110 V, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(110) = 6{,}78$ → Se redondea el valor a infinito: 7

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-7 = 5
  Por tanto, el tamaño final de v_g es: Q7.5

- I_{in}

  Esta señal es de salida y se ha especificado que todas las señales de salida del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 54,892A, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(54{,}892) = 5{,}77$ → Se redondea el valor a infinito: 6

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-6 = 6
  Por tanto, el tamaño final de I_{in} es: Q6.6

- I_r

  Esta señal es de entrada y se ha especificado que todas las señales de entrada del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 1,04166667 A, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(1{,}04166667) = 0{,}058$ → Se redondea el valor a infinito: 1

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-1 = 11
  Por tanto, el tamaño final de I_r es: Q1.11

- v_{out}

  Esta señal es de salida y se ha especificado que todas las señales de salida del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 94,0437 V, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(94{,}0437) = 6{,}55$ → Se redondea el valor a infinito: 7

Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-7 = 5
Por tanto, el tamaño final de I_r es: Q7.5

A continuación, se muestra en la Tabla 3 un resumen de los formatos elegidos para cada señal empleada en el circuito, el tamaño y su resolución:

Señal	Número de bits	Formato
dtL	13 bits	Q-14.26
dtC	21 bits	Q-14.34
i_L	29 bits	Q6.22
v_{outAux}	33 bits	Q7.25
i_{LAdd}	20 bits	Q7.12
$v_{outAuxAdd}$	33 bits	Q9.23
I_{in}	13 bits	Q6.6
v_{out}	13 bits	Q7.5
I_r	13 bits	Q1.11
v_g	13 bits	Q7.5

Tabla 3. Formato de las señales QX.Y

El código desarrollado en este modelo se divide en dos procesos, igual que en el modelo real: uno combinacional y el otro secuencial.

Mientras que en el proceso combinacional se hace referencia tanto a la apertura como al cierre del interruptor (Q) como al estado del diodo cuando el interruptor se encuentra abierto (CCM o DCM) a través de la corriente i_L, en el proceso secuencial se emplean la señal de reloj de forma síncrona o la señal de reset de forma asíncrona.

En la Figura 14 se muestra el esquema que sigue el modelo en *coma fija*. Como se puede observar, es un diagrama en el que se emplean 3 multiplexores, 3 sumadores, 4 multiplicadores y dos flips-flops que permiten que se registren las variables de estado en cada ciclo de reloj. Además, se emplea la función *resize* para redimensionar al tamaño deseado las señales, y *sfixed* para convertir las señales de entrada en formato de coma fija QX.Y.

En la parte superior del diagrama se muestra el cálculo de la corriente que atraviesa la bobina (i_L), siguiendo las ecuaciones que se han obtenido en el apartado 3.1. En el primer multiplexor se obtiene el valor de la señal i_{LAdd} dependiendo del valor que se seleccione, y junto con la señal dt/L se obtiene el incremento de la variable de estado i_L. Este incremento junto con los valores de la variable de estado i_L en el instante anterior genera el valor de la ecuación en el instante actual.

Por otro lado, en la parte inferior del esquema se muestra el cálculo de la tensión que hay en el condensador (v_c), siguiendo las ecuaciones que se han obtenido en el apartado 3.1. En el tercer multiplexor se obtiene el valor de la señal $v_{outAuxAdd}$ dependiendo del valor que se seleccione, y junto con la señal dt/C se obtiene el incremento de la variable de estado

v_{outAux}. Este incremento junto con los valores de la señal de estado v_{outAux} en el instante anterior genera el valor de la ecuación en el instante actual.

Finalmente, la tensión del condensador que coincide con la tensión de salida del circuito se registra en el instante actual mediante un proceso secuencial, y la corriente que atraviesa la bobina se registra para, a continuación, pasar por un multiplexor, el cual selecciona el valor de la corriente de entrada (I_{in}). El desarrollo de este modelo aparece en el Anexo II.

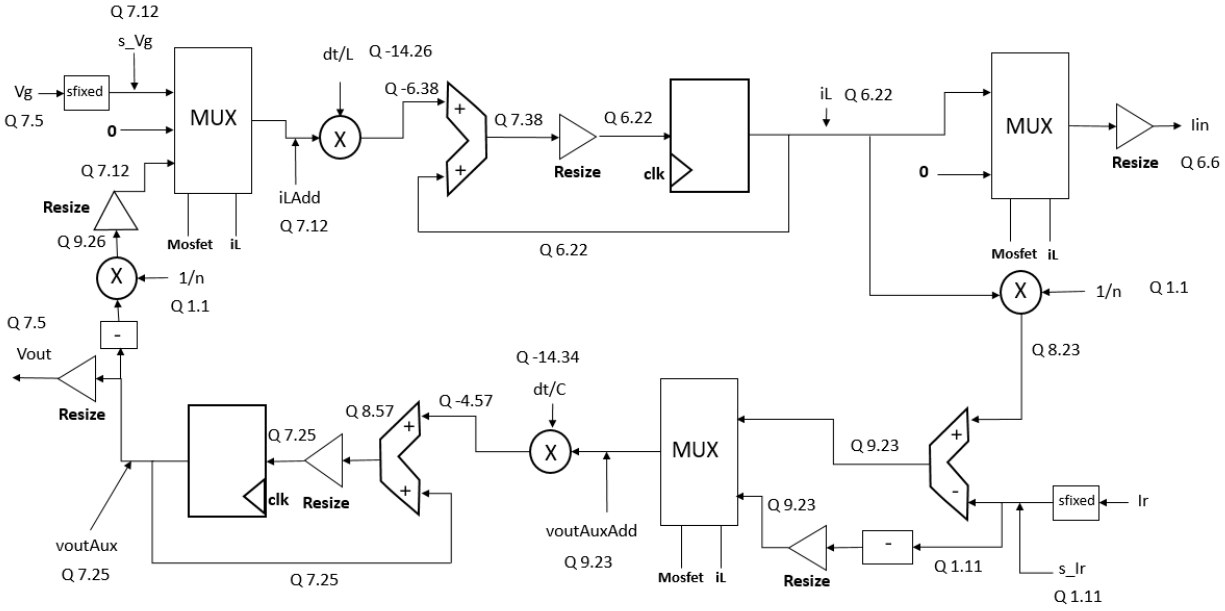

Figura 14. Esquema del convertidor Flyback en modelo coma fija

3.3.4 Modelo coma flotante

El modelo en coma flotante emplea la librería *float_pkg de VHDL-2008*, la cual describe los tipos de datos *float32* y *float64*. Sin embargo, en este modelo se ha empleado únicamente *float32*. Este modelo es bastante similar al modelo real descrito en el apartado 3.3.1.

El código desarrollado se divide en dos procesos, igual que en el modelo real pero la diferencia es la función que se emplea para definir las señales (Anexo III):

- Modelo coma flotante
  constant C : float32 := to_float(0.000440);
  constant L : float32 := to_float(0.000352);
  constant dt : float32 := to_float(0.000000020);

Además, las asignaciones de ceros, 0.0, se deben convertir también a *float32*: *to_float(0.0)*.

Por otro lado, este modelo, como ya se ha comentado en el apartado del estado del arte, emplea señales de 32 bits de los cuales 24 bits son para la mantisa: hay un "1" fijo y 23 bits restantes.

4 Incorporación de pérdidas eléctricas

En el modelo que se ha explicado en el capítulo anterior se emplea un convertidor Flyback en el cual todos sus elementos son ideales, es decir, no tienen pérdidas. El problema es que en realidad es poco común que este escenario ocurra. Para que la simulación del modelo sea realista, se deben tener en cuenta varios factores. Si se incluyen pérdidas al circuito, el modelo va a ser mucho más complejo tanto en el diseño como en la implementación.

En este capítulo se va a llevar a cabo el diseño del modelo con pérdidas eléctricas y el estudio de éstas.

4.1 Estudio pérdidas eléctricas

Para realizar el diseño del convertidor Flyback con pérdidas eléctricas es importante saber qué elementos con pérdidas se añaden al diseño y con qué valor.

La bobina de la primera espiral del transformador (R_{l1}) es el primer componente que añade pérdidas eléctricas al circuito:

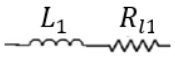

Figura 15. Diseño pérdidas bobina transformador

La bobina de la segunda espiral del transformador (R_{l2}) es el primer componente que añade pérdidas eléctricas al circuito:

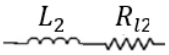

Figura 16. Diseño pérdidas bobina transformador

Ambas resistencias tienen el mismo valor.

Para saber qué valor deben tener, se ha buscado una bobina comercial que tenga un valor lo más próximo posible a 352 µH y que soporte como mínimo 1A de corriente. El modelo elegido es: IHB6BV331K-ND, en el cual la resistencia es de 40 mΩ y la máxima corriente que soporta es de 9.7A.
Por tanto, el valor de las resistencias es: R_{l1}= 40 mΩ= 0,004 Ω y R_{l2}= 40 mΩ= 0,004 Ω.

Por otro lado, el transistor (interruptor) es otro elemento que añade pérdidas eléctricas al circuito (R_T):

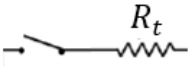

Figura 17. Diseño pérdidas transistor

El modelo del transistor es: IRPF360 en encapsulado TO-247C.
Para conocer el valor de la resistencia del transistor, se ha accedido a su datasheet y se ha buscado el valor *On Resistance* (R_{DS}), ya que es la resistencia total entre la fuente (*source*) y drenador (*drain*) cuando el transistor se encuentra encendido. Por tanto, el valor de la resistencia R_t es: 0,18 Ω.

Otro elemento que añade pérdidas eléctricas al circuito es el diodo:

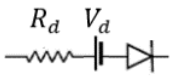

Figura 18. Diseño pérdidas diodo

El modelo del diodo es: MUR840 en encapsulado TO-220AC.En este caso, no solo se debe obtener el valor de la resistencia de este si no también la tensión directa o *forward voltage*, el cual se define como la cantidad de voltaje necesaria para que la corriente fluya a través del diodo.

En el datasheet, se obtiene que dicha tensión (V_d o V_F) tiene un valor de 1,3 V. Por otro lado, para obtener la resistencia del diodo, ya que no es un valor directo que obtengamos de la hoja de datos, debemos estimarlo. Para ello se debe indicar el punto inicial y el punto final de la tangente de la curva con temperatura 25ºC de la siguiente gráfica:

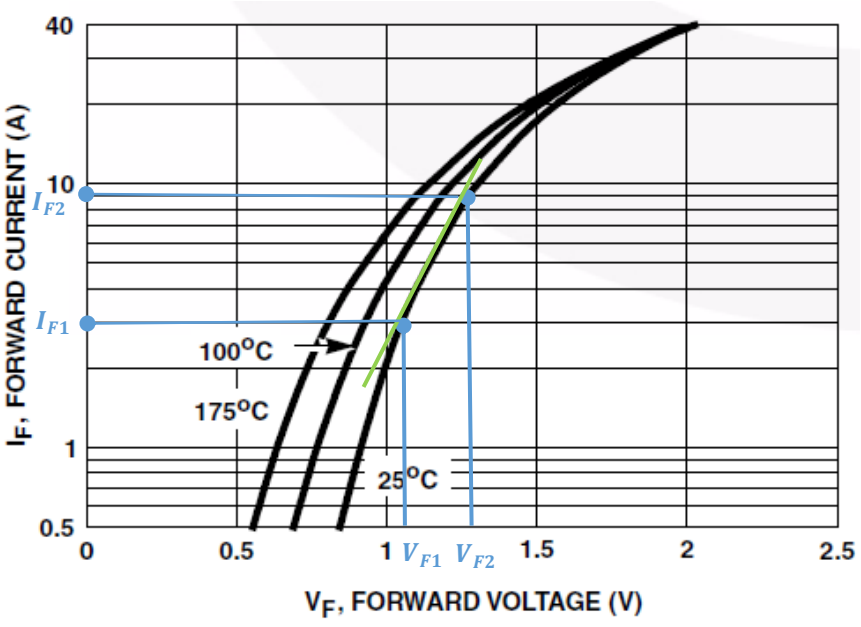

Figura 19. Forward Current VS Forward Voltage

Los valores obtenidos en la gráfica son: $V_{F1} = 1{,}05\,V$ y $V_{F2} = 1{,}28\,V$.

Estos valores de tensión corresponden con los siguientes valores de I_F:

$V_{F1} = 1{,}05\ V \rightarrow I_{F1} = 3\ A$

$V_{F2} = 1{,}28\ V \rightarrow I_{F1} = 9\ A$

Una vez obtenidos estos valores, se realiza la siguiente operación para calcular el valor de R_d:

$$Rd = \frac{\Delta V_F}{\Delta I_F} = \frac{1{,}28-1{,}05}{9-3} = 0{,}0383\ \Omega \qquad (35)$$

El último componente del circuito que añade pérdidas eléctricas al circuito es el condensador:

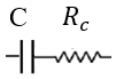

Figura 20. Diseño pérdidas condensador

El condensador empleado en el modelo, el cual tiene un valor de 440 µF, está hecho con dos condensadores de 220 µF en paralelo de 63 V. Se ha buscado un condensador con estas características y el modelo elegido es: EEU-FR1J221LB. Por tanto, su ESR (resistencia serie equivalente) tiene un valor de: 33 mΩ= 0,033 Ω.

Debido a los componentes y sus pérdidas eléctricas correspondientes, el resultado del circuito del convertidor Flyback es el que se muestra en la Figura 21. Se observa que es el circuito con los componentes ideales al que se le han añadido las pérdidas eléctricas de cada elemento.

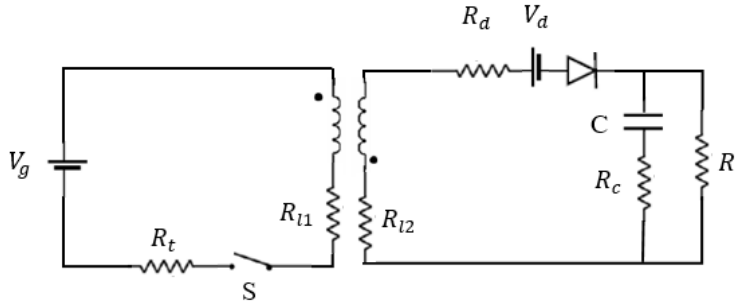

Figura 21. Topología convertidor Flyback con pérdidas

4.2 Modelo matemático

Para desarrollar el modelo matemático del convertidor Flyback con pérdidas eléctricas se va a emplear una metodología similar a la del apartado 3.1.

Tanto el modelo sin pérdidas como el modelo con pérdidas emplean las mismas ecuaciones en diferencias de las variables de estado: la corriente que atraviesa la bobina (i_L) y la tensión que cae en bornes del condensador (v_c):

- Bobina

$$i_L(k) = i_L(k-1) + v_L(k) * \frac{\Delta t}{L} \quad (36)$$

- Condensador

$$v_c(k) = v_c(k-1) + i_c(k) * \frac{\Delta t}{C} \quad (37)$$

Una vez definidas las variables de estado, se deben analizar los posibles estados del circuito, tal y como se han explicado en el apartado 3.1.

Si el interruptor se encuentra cerrado (encedido), se almacena la energía del transformador a través de su inductancia magnetizante (L_m), se libera la carga almacenada en el condensador hasta la carga (R) y el diodo se encuentra inversamente polarizado:

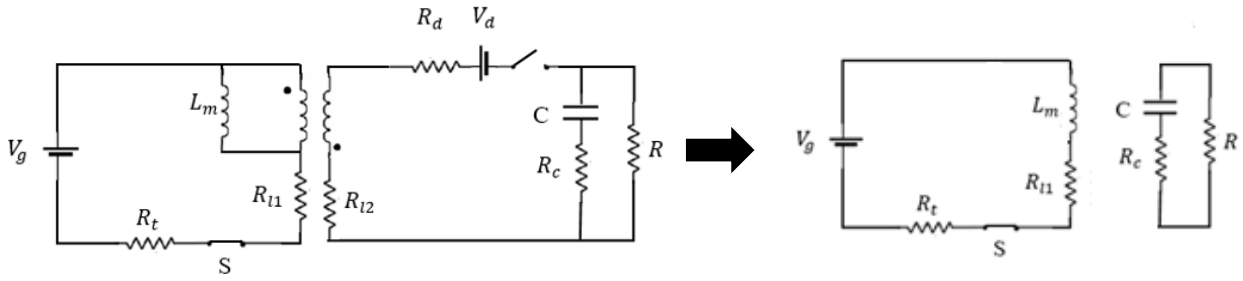

Figura 22. Circuito Flyback con pérdidas con Q=ON

Por un lado, se observa que la corriente de entrada del circuito se calcula de la siguiente forma:

$$\Delta i_L = (v_g - (R_{l1} + R_t) * i_L) * \frac{\Delta_t}{L} \quad (38)$$

En este caso, se deben tener en cuenta, además de la tensión de entrada (v_g), la resistencia de la primera espiral del transformador (R_{l1}) y la resistencia del interruptor (R_t).

Por otro lado, para calcular la tensión que cae en bornes del condensador se debe tener en cuenta la corriente que atraviesa la resistencia de carga:

$$\Delta v_c = -I_r * \frac{\Delta_t}{C} \quad (39)$$

Por otro lado, la tensión que cae en bornes de la resistencia de carga (R) es:

$$\Delta v_{out} = (v_{CAux} - (R_c * I_r)) \quad (40)$$

Se observa que dicha tensión se obtiene a partir de la tensión del condensador (v_{CAux}) , de la resistencia del condensador (R_c) y de la corriente que atraviesa la resistencia de carga (I_r).

Si por el contrario, el interruptor se encuentra apagado (abierto), se descarga la energía almacenada anteriormente en el transformador. Además, el diodo entra en conducción (CCM):

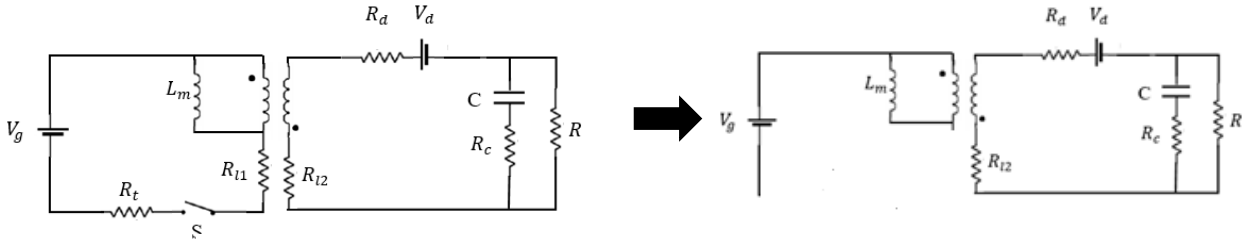

Figura 23. Circuito Flyback con pérdidas con Q=OFF

En este caso, las ecuaciones de estado de la bobina y el condensador son las siguientes:

$$\Delta i_L = \frac{-1}{n} * (\frac{i_L}{n} * (R_{l2} + R_d) + V_{out} + V_d) * \frac{\Delta_t}{L} \quad (41)$$

$$\Delta v_c = (\frac{i_L}{n} - i_R) * \frac{\Delta_t}{C} \quad (42)$$

Se observa que la tensión del condensador del circuito es la misma que en el modelo sin pérdidas, pero en el cálculo de la corriente se debe tener en cuenta: la resistencia de la segunda espiral del transformador (R_{l2}), la resistencia del diodo (R_d), la tensión de salida del circuito (V_{out}) y la tensión directa del diodo (V_d).

Por otro lado, se calcula la tensión de salida del circuito teniendo en cuenta la tensión del condensador, la corriente que atraviesa la resistencia de carga y la resistencia del condensador:

$$\Delta v_{out} = v_c + \left(\frac{i_L}{n} - i_R\right) * R_c \quad (43)$$

El último estado del circuito ocurre cuando el interruptor se encuentra apagado (abierto) y el diodo inversamente polarizado (DCM):

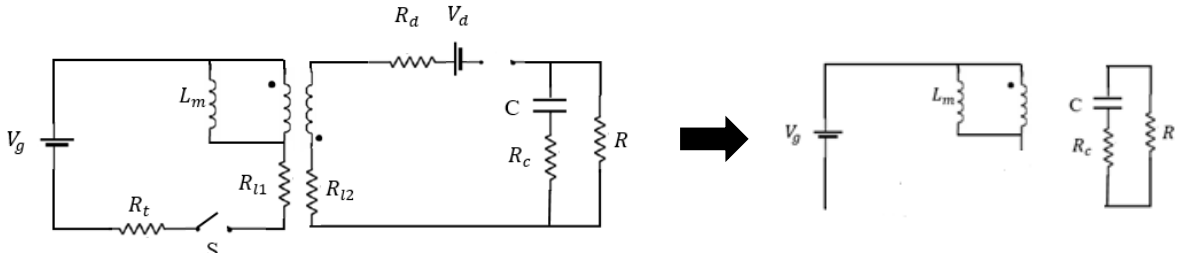

Figura 24. Circuito Flyback con pérdidas con Q=OFF y diodo DCM

En este caso, llas ecuaciones de estado de la bobina y el condensador y la tensión de salida del circuito son las que se muestran a continuación:

$$\Delta i_L = 0 * \frac{\Delta_t}{L} \quad (44)$$

$$\Delta v_c = -I_r * \frac{\Delta_t}{C} \quad (45)$$

$$\Delta v_{out} = (v_{CAux} - (R_c * I_r)) \quad (46)$$

Se observa que no circula corriente por la bobina y que la tensión del condensador y la tensión de salida del circuito se calculan de la misma forma que cuando el interruptor se encuentra encendido (cerrado).

Tras deducir las ecuaciones que definen las variables de estado (i_L; v_c) y las ecuaciones de sus incrementos (Δi_L; Δv_c), las cuales se obtienen dependiendo del estado del circuito, Se ha definido el modelo matemático del circuito de un convertidor Flyback con pérdidas eléctricas.

A modo resumen, en la Tabla 4, como se ha hecho en el capítulo 3, se muestran todas las ecuaciones obtenidas anteriormente:

Estados circuito		$i_L(k)$	$v_c(k)$
Q	**Diodo**		
ON	CCM	$i_L(k-1) + (v_g - (R_{l1} + R_t) * i_{L)*} \frac{\Delta_t}{L}$	$v_c(k-1) - i_R * \frac{\Delta_t}{C}$
OFF	CCM	$i_L(k-1) - \frac{1}{n} * (\frac{i_L}{n} * (R_{l2} + R_d) + V_{out} + V_d) * \frac{\Delta_t}{L}$	$v_c(k-1) + (\frac{i_L}{n} - i_R) * \frac{\Delta_t}{C}$
	DCM	$i_L(k-1) + 0 * \frac{\Delta_t}{L}$	$v_c(k-1) - i_R * \frac{\Delta_t}{C}$

Tabla 4. Ecuaciones de $\boldsymbol{i_L}$ (k) y $\boldsymbol{v_c}$ (k) en función del estado del circuito

4.3 Implementación del modelo

Tras llevar a cabo la elaboración del modelo matemático de un convertidor Flyback con pérdidas, éste se debe implementar en lenguaje VHDL.

En este caso se va a implementar, en primer lugar, el modelo del convertidor Flyback con pérdidas en la herramienta *Simulink* por las mismas razones que se han comentado anteriormente, además de los mismos modelos que en el capítulo 3, pero añadiendo pérdidas eléctricas: uno que utilice señales reales, otro con señales en coma fija y otro con señales en coma flotante.

Por otro lado, los valores de las constantes y señales que se emplean en este modelo son los mismos que aparecen en la Tabla 5 pero se deben añadir los siguientes valores:

Componente	Valor
Resistencia primera espiral transformador (R_{l1})	0,040 Ω
Resistencia primera espiral transformador (R_{l2})	0,040 Ω
Resistencia transistor (R_t)	0,18 Ω
Resistencia diodo (R_d)	0,0383 Ω
Resistencia condensador (R_c)	0,075 Ω
Tensión del diodo (V_d)	1,3 V

Tabla 5. Valores pérdidas eléctricas

4.3.1 Modelo simulink

En la Figura 25, se muestra como se ha llevado a cabo el diseño del convertidor Flyback con pérdidas empleando elementos de la propia herramienta *Simulink*. En este caso, el esquema que se ha seguido es el que se muestra en la Figura 21.

En este modelo se han empleado, además de los elementos comentados en el modelo del Flyback sin pérdidas, las pérdidas en modo de resistencia de: Mosfet, transformador, diodo, condensador, y la fuente de tensión del diodo.

Para la resolución del circuito, se ha empleado la misma configuración que en el modelo sin pérdidas: una resolución *Backward Euler* con paso temporal de 1e-7.

Finalmente, los valores que se han empleado en este diseño son los que se muestran en la Tabla 5. Se debe comentar que el condensador es el único elemento en el cual se deben añadir las pérdidas de forma manual en el circuito.

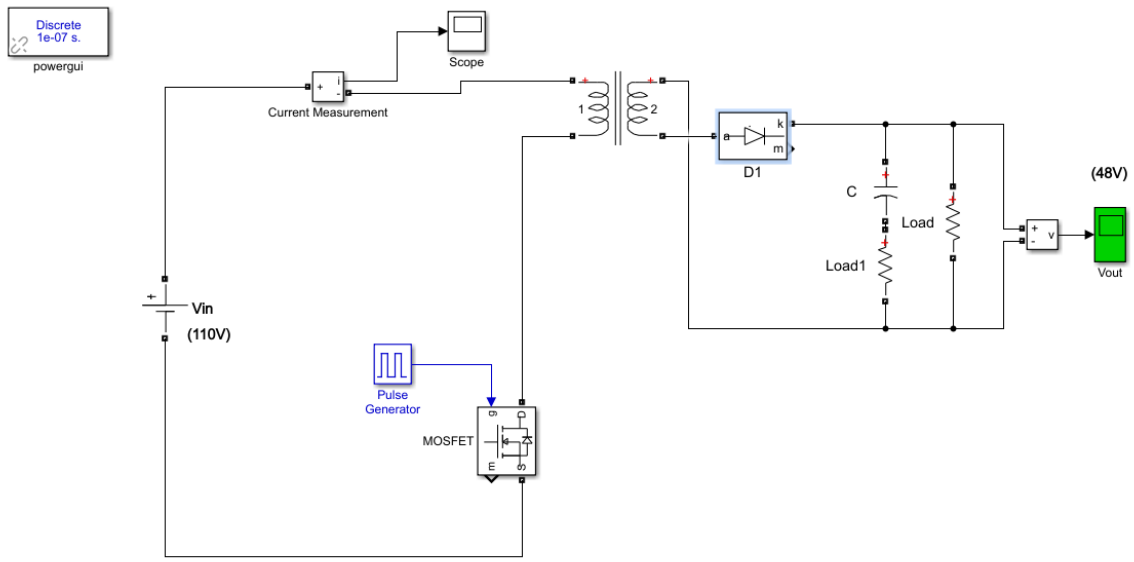

Figura 25. Modelo Flyback con pérdidas en Simulink

4.3.2 Modelo real

En este modelo se implementan las ecuaciones del convertidor Flyback del apartado 4.1 con el tipo de datos *real*. Su diseño es algo más complejo que el modelo real sin pérdidas, ya que se deben añadir elementos nuevos.
Como se ha explicado en el apartado 3.3.1, el código desarrollado se divide en dos procesos: uno combinacional y el otro secuencial.

Por un lado, en el proceso combinacional se hace referencia tanto al cierre del interruptor (Q) como al estado del diodo cuando el interruptor se encuentra abierto (CCM o DCM) y por tanto el valor de la corriente i_L.

Por otro lado, en el proceso secuencial se emplean la señal de reloj síncrona y la señal de reset asíncrona. Cabe destacar que en este modelo y en este proceso, se debe crear un nuevo registro para v_{outAux}, es decir, emplear un biestable ($v_{outAuxTemp}$), ya que si no se genera un bucle infinito debido a que en el cálculo de la corriente de la resistencia de carga (I_r) se emplea la tensión de salida del circuito (v_{outAux}) y viceversa. Por tanto, la tensión de salida del circuito se corresponde con el valor de $v_{outAuxTemp}$, y no v_{outAux} como en el modelo real sin pérdidas.

En la Figura 26 se muestra el esquema que sigue el modelo real con pérdidas. Como se puede observar, es un diagrama algo más complejo que el modelo real sin pérdidas, con 3 flips-flops que permiten el registro tanto de las variables de estado como de la tensión de salida del circuito en cada ciclo de reloj. Además, se emplean 4 multiplexores, 8 sumadores y 9 multiplicadores.

En la parte superior del diagrama se muestra cómo se calcula la corriente que atraviesa la bobina (i_L), siguiendo las ecuaciones que se han obtenido en el apartado 4.1. En el primer multiplexor se obtiene el valor de la señal i_{LAdd} dependiendo del valor que se seleccione. En este caso, además de emplear la tensión de entrada del circuito y la corriente que atraviesa la bobina, como en el modelo explicado en el capítulo 3, se tiene en cuenta la resistencia de transistor y la resistencia de la primera espiral del transformador. Además, la señal i_{LAdd} junto con la constante dt/L genera el incremento de la variable de estado i_L. Este incremento junto con los valores de la señal de estado i_L en el instante anterior genera el valor de la ecuación en el instante actual.

Por otro lado, en la parte inferior del esquema se muestra el cálculo de la tensión que cae en bornes en el condensador (v_{cAux}), siguiendo las ecuaciones que se han obtenido en el apartado 4.1. En el tercer multiplexor se obtiene el valor de la señal v_{CAdd} dependiendo del valor que se seleccione, y junto con la señal dt/C se obtiene el incremento de la variable de estado v_{CAux}. Este incremento junto con los valores de la señal de estado v_{CAux} en el instante anterior genera el valor de la ecuación en el instante actual.

Finalmente, teniendo en cuenta la resistencia del condensador, se obtiene la señal v_{outAux} dependiendo del valor que se elija en el multiplexor. Ésta se registra en cada ciclo de reloj mediante uno de los flips-flops comentados anteriormente, generando la señal: $v_{outAuxTemp}$, la cual se almacena en la tensión de salida del circuito v_{out}. Por otro lado, la corriente que

atraviesa la bobina se registra para, a continuación, pasar por un multiplexor, el cual selecciona el valor de la corriente de entrada (I_{in}). El desarrollo de este modelo aparece en el Anexo IV.

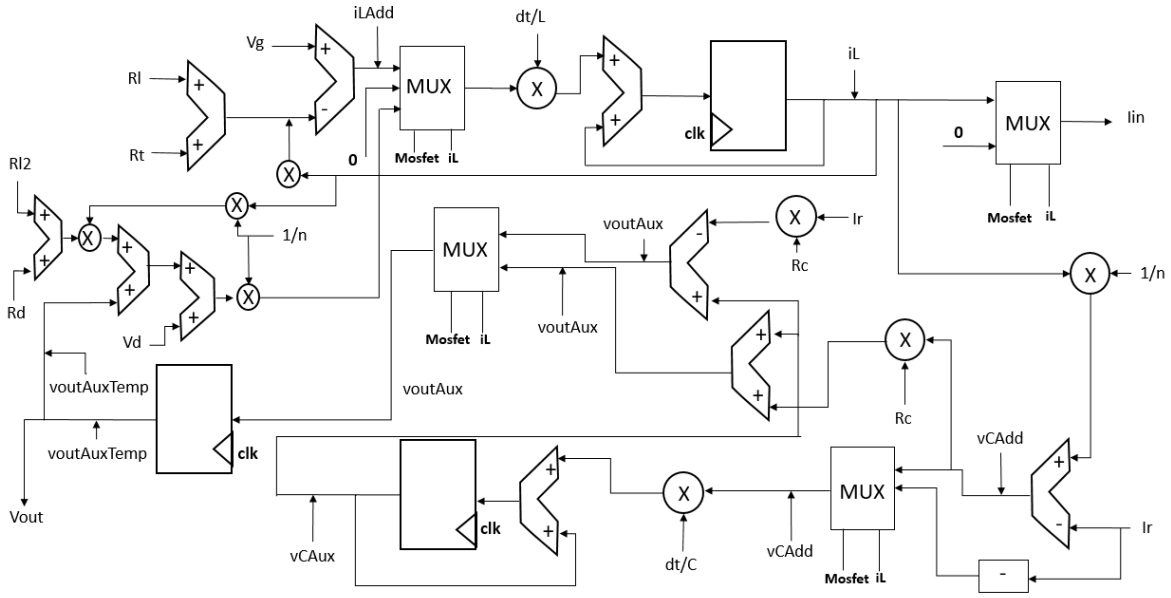

Figura 26. Esquema del convertidor Flyback con pérdidas eléctricas en modelo real

4.3.3 Modelo coma fija

Como ya se ha explicado en el capítulo anterior, la notación en coma fija permite adaptar la posición de la coma de cada valor y el propio diseñador es el encargado de definir cada uno de los tamaños de las constantes y señales empleadas en el circuito.

En este modelo las señales de entrada son: la tensión de entrada (v_g) y la corriente de la resistencia de carga (I_r) y para las señales de salida son: la corriente de entrada del circuito (I_{in}) y la tensión de salida del circuito ($v_{outAuxTemp}$). Para estas señales, se ha elegido un tamaño de 13 bits al igual que el modelo sin pérdidas.

Para obtener cuantos bits se deben asignar a la parte entera (X), se debe calcular de la misma forma que en el capítulo 3 (25):

$$log_2(valor\_maximo)$$

Además, el número total de bits de la señal se aplica de la misma forma [18] que en el capítulo anterior.

A continuación, se explica uno a uno los diferentes formatos elegidos en coma fija para cada una de las señales del circuito:

- dtL
  Esta constante se obtiene de la operación (27):

  $$\frac{dt}{L} = \frac{0{,}00000002}{0{,}000352} = 5{,}6812182 * 10^{-5}\ s/H$$

  Estos valores aparecen en la Tabla 1.

  El número de bits que se debe asignar a la parte entera es (25):
  $log_2(5{,}6812182 * 10^{-5}) = -14{,}1033$ → Se redondea el valor a infinito: -14

  Por otro lado, como esta señal es una constante, el número de bits que se deben asignar a la parte fraccionaria se calcula de la siguiente forma (28):

  $$-log_2(5{,}6812182 * 10^{-5}) + n = 14{,}1033 + n$$

  Primero se redondea el valor 14,1033 a 15 y luego se suma n=8:

  15 + n = 15 +8= 23

  Por tanto, el tamaño calculado es: Q-14.23. El problema es que con este número de bits el resultado no sale exacto. Para solucionarlo, se aumentan el número de bits de la parte decimal a 26.
  Por tanto, el tamaño final de la constante dtL es: Q -14.26

- dtC
  Esta constante se obtiene de la operación (29):

  $$\frac{dt}{L} = \frac{0{,}00000002}{0{,}000440} = 4{,}545454545 * 10^{-5}\ s/F$$

  Estos valores aparecen en la Tabla 2.

  El número de bits que se debe asignar a la parte entera es (25):
  $log_2(4{,}545454545 * 10^{-5}) = -14{,}4252$ → Se redondea el valor a infinito: -14

  Por otro lado, como esta señal es una constante, el número de bits que se deben asignar a la parte fraccionaria se calcula de la siguiente forma (28):

  $$-log_2(4{,}545454545 * 10^{-5}) + n = 14{,}4252 + n$$

  Primero se redondea el valor 14,4252 a 15 y luego se suma n=8:

  15 + n = 15 +8= 23

  Por tanto, el tamaño calculado es: Q-14.23.
  El problema es que con este número de bits el resultado no sale exacto. Para solucionarlo, se aumentan el número de bits de la parte decimal a 34.
  Por tanto, el tamaño final de la constante dtC es: Q -14.34

- v_{CAux}

  Para calcular que números se deben asignar a la parte entera y a la parte decimal de esta señal, primero se debe obtener el número total de bits de la señal con la fórmula (20), donde:

  - X= 75,0319 V

    X es el máximo valor que se obtiene cuando se realiza la simulación real del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión del condensador (v_{cAux}) son 0.

  - $\Delta_X = v_{outAuxAdd} * \frac{d_t}{C}$

    Los posibles valores que puede tomar la señal $v_{outAuxAdd}$ son:

    1. $-I_r = \frac{-v_{outAux}}{R} = \frac{-48}{46{,}08} = -1{,}04166667A$

       En este caso, tanto la tensión de salida del circuito (v_{outAux}) como la resistencia de carga (R) aparecen en la Tabla 1. La señal $v_{outAuxAdd}$ toma este valor cuando el transistor está cerrado (Q=ON).

    2. $\frac{i_L}{n} - I_r$ = -0,494465 A

       Posibles valores de la señal i_L:

       - Máximo valor

         $$\frac{i_{Lmax}}{n} - I_r = 43{,}3725 - 1{,}00694 = 42{,}36556 \qquad (47)$$

         La corriente de entrada máxima (i_{Lmax}) y la corriente de la resistencia de carga (I_r) se obtienen a partir de la simulación real con pérdidas del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión del condensador (v_{cAux}) son 0. En el caso de I_r se calcula a partir del valor medio: $\frac{I_{rmax} + I_{rmin}}{2}$.

       - Mínimo valor

         $$\frac{i_{Lmin}}{n} - I_r = 0{,}519893 - 1{,}00694 = -0{,}487047 \qquad (48)$$

         En este caso, la corriente de entrada mínima (i_{Lmin}) y la corriente de la resistencia de carga se obtienen de la misma forma que en el caso anterior.

    La señal $v_{outAuxAdd}$ toma este valor cuando el transistor está abierto (Q=OFF).

    Entre ambos valores, se debe elegir el menor de ellos en valor absoluto: 0,487047.

Por tanto, se sustituyen los valores en (32) y se obtiene lo siguiente:
$\Delta_X = v_{outAuxAdd} * \frac{d_t}{C} = 0{,}487047 * \frac{0{,}00000002}{0{,}000440} = 2{,}21385 * 10^{-5}$

- n=8

Sustituyendo los valores obtenidos (26):

$width = log_2\left(\frac{75{,}0319}{2{,}21385*10^{-5}}\right) + 8 = 29{,}6925$ → Se redondea el valor a infinito: 30

Ahora se calcula el número de bits que se deben asignar a la parte entera de la señal (25):

$log_2(75{,}0319) = 6{,}23$ → Se redondea el valor a infinito: 7

Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 29-7=22
Así, el tamaño de la señal v_{CAux} es: Q7.22

- i_L

Para calcular que números se deben asignar a la parte entera y a la parte decimal de esta señal, primero se debe obtener el número total de bits de la señal con la fórmula (20), donde:

- X= 43,3725 A
  X es el máximo valor que se obtiene cuando se realiza la simulación real del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión del condensador (v_{cAux}) son 0.

- $\Delta_X = i_{LAdd} * \frac{d_t}{L}$

  En este caso, la señal i_{LAdd} se calcula empleando las siguientes fórmulas:

  1. $v_g - (R_{l1} + R_t) * i_L$ (49)

     La señal i_{LAdd} toma este valor cuando el transistor está cerrado (Q=ON). En este caso, esta señal puede tomar los siguientes valores dependiendo de si el valor de la corriente de entrada del circuito es máximo o mínimo:
     - Si la corriente es máxima, i_{LAdd} tiene un valor de 100,45 A.
     - Si la corriente es mínima, i_{LAdd} tiene un valor de 109,88 A.

  2. $-(i_L * (R_{l2} + R_d) + v_{outAux} + V_d)$ (50)

     La señal i_{LAdd} toma este valor cuando el transistor está abierto (Q=OFF).

     En este caso, esta señal puede tomar los siguientes valores dependiendo de si el valor de la corriente de entrada del circuito y la tensión de salida del circuito son máximos o mínimos:

- Si la corriente y la tensión son máximos, i_{LAdd} tiene un valor de -79,7711 A.
- Si la corriente es máxima y la tensión es mínima, i_{LAdd} tiene un valor de -51,1458 A.
- Si la corriente y la tensión son mínimos, i_{LAdd} tiene un valor de -47,79071 A.
- Si la corriente es mínima y la tensión es máxima, i_{LAdd} tiene un valor de -76,0877 A.

Entre estos valores, se debe elegir el menor de ellos en valor absoluto: 47,79071, y se calcula de la siguiente forma:

$$-(i_{LMin} * (R_{l2} + R_d) + v_{outAuxMin} + V_d) \quad \rightarrow \quad (51)$$

$$-(0{,}519 * (0{,}040 + 0{,}0383) + 46{,}45 + 1{,}3) = -47{,}79071$$

La corriente de entrada mínima (i_{Lmin}) y la tensión mínima de salida del circuito se obtienen a partir de la simulación real con pérdidas del modelo en ModelSim cuando las condiciones iniciales de la corriente de entrada (i_L) y la tensión del condensador (v_{cAux}) son 0.

Por tanto, se sustituyen los valores en (30) y se obtiene lo siguiente:

$$\Delta_X = i_{LAdd} * \frac{d_t}{L} = 47{,}79071 * \frac{0{,}00000002}{0{,}000352} = 2{,}71538 * 10^{-3}$$

- n=8

Sustituyendo los valores obtenidos, se calcula el número total de bits (26):

$$width = log_2\left(\frac{43{,}3725}{2{,}71538e{-}3}\right) + 8 = 21{,}963 \quad \rightarrow \text{Se redondea el valor a infinito: 22}$$

Ahora se calcula el número de bits que se deben asignar a la parte entera de la señal (25):

$log_2(43{,}3725) = 5{,}438 \rightarrow$ Se redondea el valor a infinito: 6

Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 21-6=15
Así, el tamaño de la señal i_L es: Q6.15
Este tamaño no es el definitivo, ya que realizando pruebas se obtienen valores más parecidos a los del modelo real sin pérdidas si el tamaño final es: Q6.19.
Por tanto, el tamaño final de la señal i_L es Q6.19.

- i_{LAdd}
  En el apartado donde se calcula el tamaño de la señal i_L, se indica que el valor máximo de la señal i_{LAdd} es 109,88 V.
  Por tanto, el número de bits que se deben asignar a la parte entera es (25):

$log_2(109{,}88) = 6{,}77 \rightarrow$ Se redondea el valor a infinito: 7

Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28) empleando el incremento Δ_x calculado en el apartado de i_L:

$-log_2(2{,}71538e-3)+n=8{,}52+n$

Primero se redondea el valor 8,52 a 9 y luego se suma n=8:
9+n=9+8=17

Por tanto, el tamaño final de la señal i_{LAdd} es: Q7.17

- v_{CAdd}
  En el apartado donde se calcula el tamaño de la señal v_{CAux}, se indica que el valor máximo de la señal v_{CAdd} es 42,36556 V.
  Por tanto, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(42{,}36556) = 5{,}43$ → Se redondea el valor a infinito: 6

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28) empleando el incremento Δ_x calculado en el apartado de v_{CAux}:

  $-log_2(2{,}21385e-5)+n=15{,}46+n$

  Primero se redondea el valor 15,46 a 16 y luego se suma n=8:
  16+n=16+8=24

  Por tanto, el tamaño de la señal v_{CAdd} es: Q6.24

  Este tamaño no es el definitivo ya que esta señal depende del número de bits del resultado de i_L/n.
  Por tanto, el tamaño final es: Q9.20.

- v_{outAux}
  El máximo valor de esta señal es 74,7875.
  Por tanto, el número de bits que se deben asignar a la parte entera son (25):

  $log_2(74{,}7875) = 6{,}22$ → Se redondea el valor a infinito: 7

  Por otro lado, el número de bits que se deben asignar a la parte fraccionaria depende del número de decimales que llegan a ésta.
  Por tanto, el tamaño final de la señal v_{outAux} es: Q7.22

- R_{l1} y R_{l2}
  Se sabe que el valor de ambas resistencias es 0,040 Ω por lo explicado en el apartado 4.1.
  Por un lado, se calcula el número de bits que se deben asignar a la parte entera (25):

  $log_2(0{,}040) = -4{,}64$ → Se redondea el valor a infinito: -4

Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28):

$$-log_2(0{,}040) + n = 4{,}64 + n$$

Primero se redondea el valor 4,64 a 5 y luego se suma n=8:
5+n=5+8=13

Realizando pruebas se observa que ambas constantes deben tener 20 bits en la parte decimal ya que así el número que se obtiene es exacto.
Por tanto, el tamaño final de R_{l1} / R_{l2} es Q-4.20.

- R_t
  Se sabe que el valor de esta resistencia es 0,18 Ω por lo explicado en el apartado 4.1.
  Por un lado, se calcula el número de bits que se deben asignar a la parte entera (25):

  $log_2(0{,}18) = -2{,}47$ → Se redondea el valor a infinito: -2

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28):

  $$-log_2(0{,}18) + n = 2{,}47 + n$$

  Primero se redondea el valor 2,47 a 3 y luego se suma n=8:
  3+n=3+8=11

  Realizando pruebas se observa que ambas constantes deben tener 18 bits en la parte decimal ya que así el número que se obtiene es exacto.
  Por tanto, el tamaño final de R_t es Q-2.18.

- R_d
  Se sabe que el valor de esta resistencia es 0,0383 Ω por lo explicado en el apartado 4.1.
  Por un lado, se calcula el número de bits que se deben asignar a la parte entera (25):

  $log_2(0{,}0383) = -4{,}7$ → Se redondea el valor a infinito: -4

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28):

  $$-log_2(0{,}0383) + n = 4{,}7 + n$$

  Primero se redondea el valor 4,7 a 5 y luego se suma n=8:
  5+n=5+8=13

  Realizando pruebas se observa que ambas constantes deben tener 21 bits en la parte decimal ya que así el valor de la constante R_d que se obtiene es exacto.

Por tanto, el tamaño final de R_t es Q-4.21.

- R_c
  Se sabe que el valor de esta resistencia es 0,075 Ω por lo explicado en el apartado 4.1.
  Por un lado, se calcula el número de bits que se deben asignar a la parte entera (25):

  $log_2(0{,}075) = -3{,}73$ → Se redondea el valor a infinito: -3

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28):

  $-log_2(0{,}075) + n = 3{,}73 + n$

  Primero se redondea el valor 3,73 a 4 y luego se suma n=8:
  4+n=4+8=12

  Realizando pruebas se observa que ambas constantes deben tener 25 bits en la parte decimal ya que así el valor de la constante R_c es exacto. Por tanto, el tamaño final de R_t es Q-3.25.

- V_d
  Se sabe que el valor de esta tensión es 1,3 V por lo explicado en el apartado 4.1.
  Por un lado, se calcula el número de bits que se deben asignar a la parte entera (25):

  $log_2(1{,}3) = 0{,}37$ → Se redondea el valor a infinito: 1

  Por otro lado, se calcula el número de bits que se deben asignar a la parte fraccionaria (28):

  $-log_2(1{,}3) + n = -0{,}37 + n$

  Primero se redondea el valor -0,37 a 0 y luego se suma n=8:
  0+n=0+8=8

  Realizando pruebas se observa que ambas constantes deben tener 20 bits en la parte decimal ya que así el número que se obtiene es exacto.
  Por tanto, el tamaño final de R_t es Q1.20.

- v_g
  Esta señal es de entrada y se ha especificado que todas las señales de entrada del circuito deben tener un número total de13 bits.
  Como el valor máximo de esta señal es 110 V, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(110) = 6{,}78$ → Se redondea el valor a infinito: 7

Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-7 = 5
Por tanto, el tamaño final de v_g es: Q7.5

- I_{in}
  Esta señal es de salida y se ha especificado que todas las señales de salida del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 43,3725A, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(43{,}3725) = 5{,}43$ → Se redondea el valor a infinito: 6

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-6 = 6
  Por tanto, el tamaño final de I_{in} es: Q6.6

- I_r
  Esta señal es de entrada y se ha especificado que todas las señales de entrada del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 1,04166667 A, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(1{,}04166667) = 0{,}058$ → Se redondea el valor a infinito: 1

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-1 = 11
  Por tanto, el tamaño final de I_r es: Q1.11

- v_{out}
  Esta señal es de salida y se ha especificado que todas las señales de salida del circuito deben tener un número total de 13 bits.
  Como el valor máximo de esta señal es 75,0319 V, el número de bits que se deben asignar a la parte entera es (25):

  $log_2(75{,}0319) = 6{,}22$ → Se redondea el valor a infinito: 7

  Por otro lado, se sabe que el número total de bits de esta señal es de 13 bits, pero un bit es para el signo. Por tanto, el número de bits que se deben asignar a la parte fraccionaria es: 12-7 = 5. Por tanto, el tamaño final de I_r es: Q7.5

A continuación, se muestra en la Tabla 6 un resumen de los formatos elegidos para cada señal empleada en el circuito, el tamaño y su resolución:

Señal	Número de bits	Formato
dtL	13 bits	Q-14.26
dtC	21 bits	Q-14.34
$\boldsymbol{i_L}$	26 bits	Q6.19
$\boldsymbol{v_{CAux}}$	30 bits	Q7.22
$\boldsymbol{i_{LAdd}}$	25 bits	Q7.17
$\boldsymbol{v_{CAdd}}$	30 bits	Q9.20
$\boldsymbol{v_{outAux}}$	30 bits	Q7.22
$\boldsymbol{v_{outAuxTemp}}$	30 bits	Q7.22
$\boldsymbol{I_{in}}$	13 bits	Q6.6
$\boldsymbol{v_{out}}$	13 bits	Q7.5
$\boldsymbol{I_r}$	13 bits	Q1.11
$\boldsymbol{v_g}$	13 bits	Q7.5
$\boldsymbol{R_{l1}}$	17 bits	Q-4.20
$\boldsymbol{R_t}$	17 bits	Q-2.18
$\boldsymbol{R_{l2}}$	17 bits	Q-4.20
$\boldsymbol{R_d}$	18 bits	Q-4.21
$\boldsymbol{R_c}$	23 bits	Q-3.25
$\boldsymbol{V_d}$	22 bits	Q1.20

Tabla 6. Formato de las señales QX.Y con pérdidas

El código desarrollado en este modelo se divide en dos procesos, igual que en el modelo real: uno combinacional y el otro secuencial.

Mientras que en el proceso combinacional se hace referencia tanto a la apertura como al cierre del interruptor (Q) como al estado del diodo cuando el interruptor se encuentra abierto (CCM o DCM), en el proceso secuencial se emplean la señal de reloj de forma síncrona o la señal de reset de forma asíncrona.

En la Figura 27 se muestra el esquema que sigue el modelo en coma fija con pérdidas. Como se puede observar, es un diagrama algo más complejo que el modelo en coma fija sin pérdidas, con 3 flips-flops que permiten el registro tanto de las variables de estado como de la tensión de salida del circuito en cada ciclo de reloj. Además, se emplean 4 multiplexores, 8 sumadores y 9 multiplicadores. Además, se emplea la función *resize* para redimensionar al tamaño deseado las señales, y *sfixed* para convertir las señales de entrada en formato de coma fija QX.Y.

En la parte superior del diagrama se muestra cómo se calcula la corriente que atraviesa la bobina (i_L), siguiendo las ecuaciones que se han obtenido en el apartado 4.1. En el primer multiplexor se obtiene el valor de la señal i_{LAdd} dependiendo del valor que se seleccione. En este caso, además de emplear la tensión de entrada del circuito y la corriente que atraviesa la bobina, como en el modelo explicado en el capítulo 3, se tiene en cuenta la resistencia de transistor y la resistencia de la primera espiral del transformador. Además, la señal i_{LAdd} junto con la constante dt/L genera el incremento de la variable de estado i_L. Este incremento junto con los valores de la señal de estado i_L en el instante anterior genera el valor de la ecuación en el instante actual.

Por otro lado, en la parte inferior del esquema se muestra el cálculo de la tensión que cae en bornes en el condensador (v_{cAux}), siguiendo las ecuaciones que se han obtenido en el apartado 4.1. En el tercer multiplexor se obtiene el valor de la señal v_{CAdd} dependiendo del valor que se seleccione, y junto con la señal dt/C se obtiene el incremento de la variable de estado v_{CAux}. Este incremento junto con los valores de la señal de estado v_{CAux} en el instante anterior genera el valor de la ecuación en el instante actual.

Finalmente, teniendo en cuenta la resistencia del condensador, se obtiene la señal v_{outAux} dependiendo del valor que se elija en el multiplexor. Ésta se registra en cada ciclo de reloj mediante uno de los flips-flops comentados anteriormente, generando la señal: $v_{outAuxTemp}$, la cual se almacena en la tensión de salida del circuito v_{out}. Por otro lado, la corriente que atraviesa la bobina se registra para, a continuación, pasar por un multiplexor, el cual selecciona el valor de la corriente de entrada (I_{in}). El desarrollo de este modelo aparece en el Anexo V.

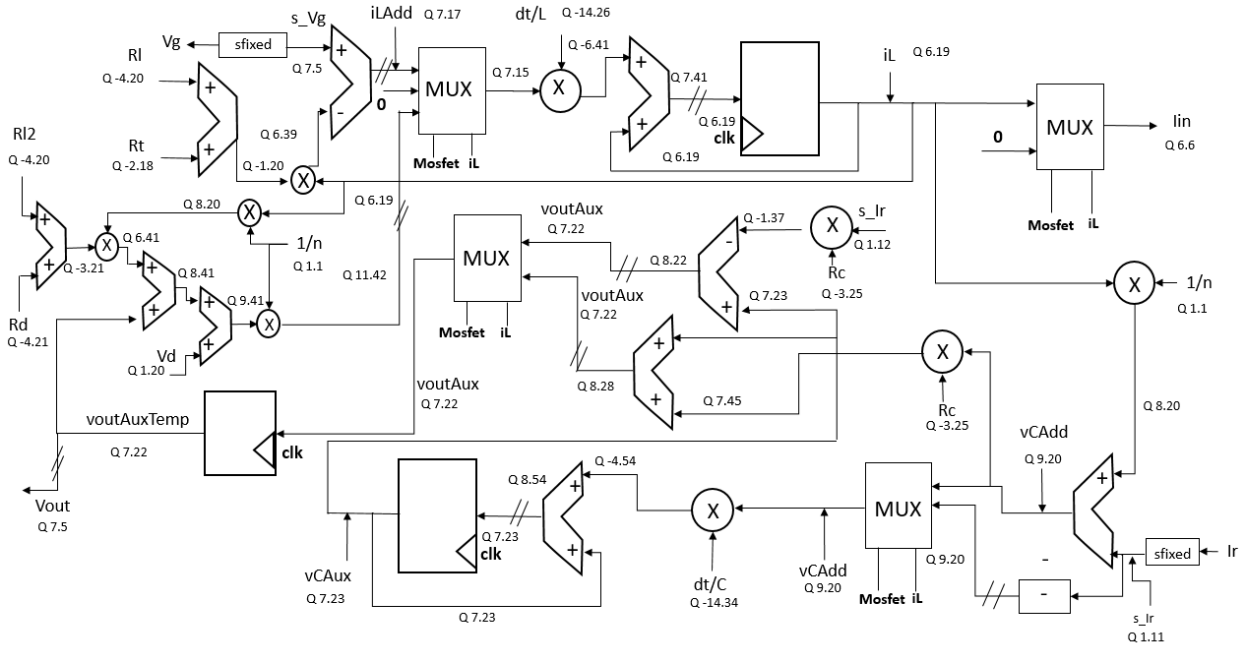

Figura 27. Esquema del convertidor Flyback con pérdidas en modelo coma fija

4.3.4 Modelo coma flotante

El modelo en coma flotante emplea la librería *float_pkg de VHDL-2008*, como ya se ha comentado en el apartado 3.3.3. En este modelo se emplea únicamente *float32* (32 bits). Este modelo es bastante similar al modelo real descrito en el apartado 4.3.1.

El código desarrollado se divide en dos procesos, igual que en el modelo real pero la diferencia es la función que se emplea para definir las señales (Anexo VI):

- Modelo coma flotante
  constant C : float32 := to_float(0.000440);
  constant L : float32 := to_float(0.000352);
  signal iL : float32 := to_float(0.0);
  signal voutAux,vCAux : float32 := to_float(48.0);

Además, las asignaciones de ceros, 0.0, se deben convertir también a *float32*: *to_float(0.0)*.

Por otro lado, este modelo, como ya se ha comentado en el capítulo 3, emplea señales de 32 bits de los cuales 24 bits son para la mantisa: hay un “1” fijo y 23 bits restantes.

5 Resultados

Una vez que se ha llevado a cabo la implementación de un convertidor Flyback, se han obtenido los siguientes modelos sin pérdidas y con pérdidas: modelo real, modelo en coma fija y modelo en coma flotante.

Cabe destacar que el modelo real, como se ha comentado anteriormente, se emplea como el modelo de referencia, no se puede sintetizar en una FPGA pero sirve para comprobar el resto de modelos estudiados.

Se realiza una comparativa de precisión de este modelo con los modelos de coma fija y coma flotante. Los resultados para llevar a cabo estas comparativas se obtienen ejecutando y simulando los modelos comentados en la herramienta *ModelSim*.

Además, como los modelos en coma fija y en coma flotante pueden implementarse en una FPGA, se realiza una comparativa de éstos en cuanto al área ocupada y el tiempo total de ejecución. Estos resultados se obtienen empleando la herramienta *Xilinx ISE*.

5.1 Lazo abierto

Las pruebas que se han llevado a cabo en este Trabajo de Fin de Máster de los modelos han sido en lazo abierto, es decir, se han ejecutado y simulado cada uno de los circuitos indicados con señales de control fijas, que, en este caso, son el ciclo de trabajo y el período de conmutación. En este tipo de circuito no hay retroalimentación hacia el control, es decir, la salida no vuelve a la entrada, por lo que hace que el sistema no pueda llevar a cabo ninguna función de ajuste de los valores obtenidos. Gracias a este tipo de comportamiento, se puede comprobar, sin ninguna alteración externa, la precisión de cada uno de los modelos explicados.

En todas las simulaciones de resultados que se van a mostrar a partir de ahora, se va a emplear un período de conmutación de 20 µs y un ciclo de trabajo de 30,4 %, en el cual el tiempo de encendido del transistor es de 6,08 µs y el tiempo de apagado 13,92 µs (cálculos realizados en el apartado 3.2).

5.2 Condiciones iniciales

En este apartado se va a explicar cómo se han obtenido las condiciones iniciales con la mayor exactitud posible, para entrar en régimen permanente de la forma más rápida y evitar largas simulaciones. Estas son las condiciones iniciales que se van a emplear en todas las pruebas comentadas a partir de ahora.
Por otro lado, para obtener los valores mínimos y máximos de las señales obtenidas en el modelo en coma fija, las condiciones iniciales de la corriente de entrada y la tensión de salida eran de 0.0.

5.2.1 Corriente entrada inicial

La corriente de entrada, como ya se ha comentado anteriormente, comienza en 0.0, tal cual se muestra a continuación:

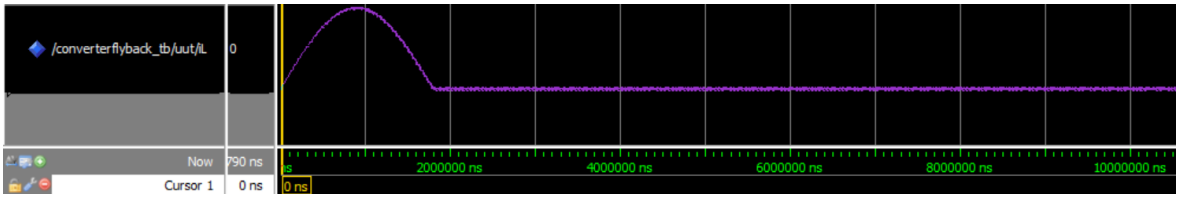

Figura 28. Simulación corriente entrada modelo real

Para calcular las condiciones iniciales que se deben emplear para que dicha corriente entre en régimen permanente de una forma mucho más rápida se deben realizar los cálculos que se explican a continuación.

El valor medio de la corriente debe ser el que se muestra a continuación, donde la subida la proporciona la bobina de la primera espiral del transformador (i_{L1}) y la bajada la aporta la bobina de la segunda espiral del transformador (i_{L2}):

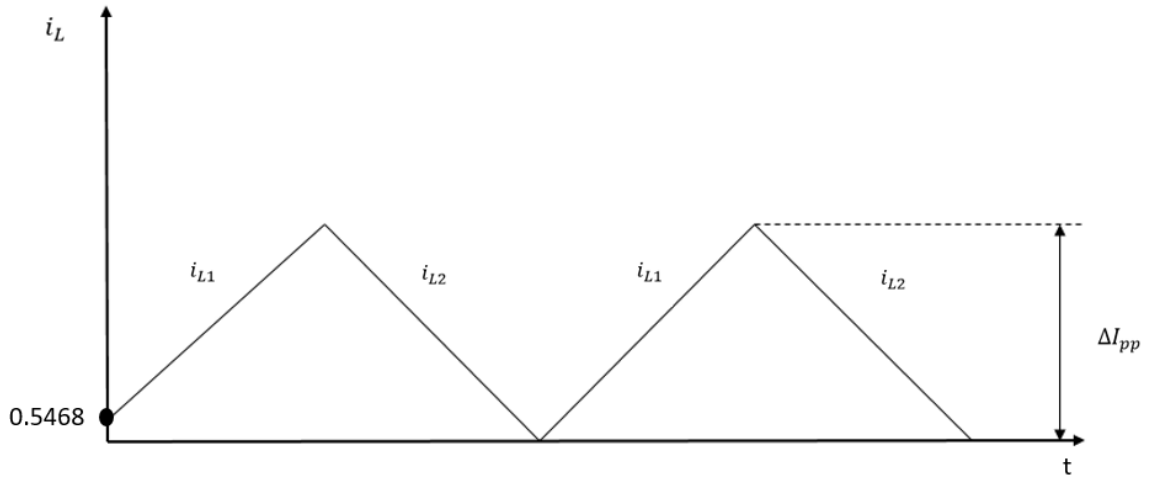

Figura 29. Valor medio corriente entrada

La corriente en la bobina de la primera espiral de transformador es la siguiente:

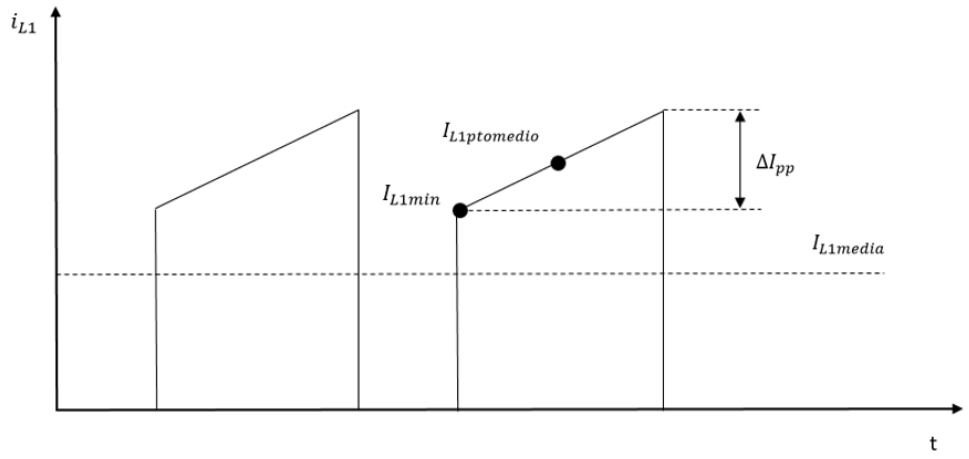

Figura 30. Corriente bobina primera espiral transformador

Si se considera que $I_{L1media}$ corresponde a la potencia de entrada P_{in} y que ésta tiene el mismo valor que la potencia de salida del circuito:

$$I_{in} = I_{L1media} = \frac{P_{in}}{V_{in}} = \frac{50}{110} = 0{,}4545454\widehat{5}\ A \quad (52)$$

Por otro lado, se sabe del apartado 3.2.1, que el ciclo de trabajo (d) es de 0,3037.

Como se observa en la Figura 30, $I_{L1media}$ es el área bajo la curva y, por tanto:

$$I_{L1media} * t_{sw} = I_{L1ptomedio} * tiempoON \quad (53)$$

$$I_{L1ptomedio} = \frac{I_{L1media} * t_{sw}}{tiempoON} = \frac{0{,}45}{d} = \frac{0{,}4545454545}{0{,}3037} = 1{,}49669231\ A \quad (54)$$

Por otro lado, la corriente ΔI_{pp} se calcula de la siguiente forma:

$I_{min} = 0{,}546843\ A$

$I_{max} = 2{,}446537\ A$

$$\Delta I_{pp} = I_{max} - I_{min} = 2{,}446537 - 0{,}546843 = 1{,}899694\ A \quad (55)$$

Por tanto, el mínimo valor que va a tomar la corriente de entrada para entrar antes en régimen permanente es:

$$I_{L\_init} = I_{L1ptomedio} - \frac{\Delta I_{pp}}{2} = 1{,}496692 - \frac{1{,}899694}{2} = 0{,}546845\ A \quad (56)$$

5.2.2 Tensión salida inicial

La tensión de salida del circuito, como ya se ha comentado en el inicio de este capítulo, comienza en 0.0, tal cual se muestra a continuación:

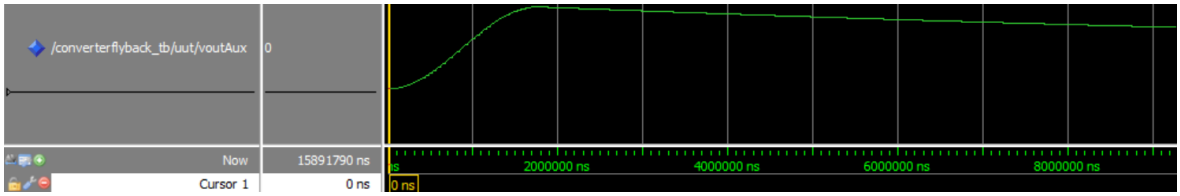

Figura 31. Simulación corriente de salida modelo real

Para calcular las condiciones iniciales que se deben emplear para que dicha tensión entre en régimen permanente de una forma mucho más rápida se deben realizar los cálculos que se explican a continuación.

El valor inicial en régimen permanente de la tensión de salida del circuito se obtiene empleando la siguiente fórmula:

$$V_{outmin} = V_{outmedio} + \frac{\Delta V_{pp}}{2} \quad (57)$$

Por un lado, la tensión de salida media es:

$$V_{outmedio} = \frac{d*V_{in}}{1-d} = \frac{0{,}304*110}{1-0{,}304} = 48{,}00459374\ V \quad (58)$$

Por otro lado, se calcula la mínima variación de la señal de la tensión de salida:

$$V_{min} = 48{,}00679\ V$$

$$V_{max} = 48{,}012\ V$$

$$\Delta V_{pp} = V_{max} - V_{min} = 48{,}012 - 48{,}00679 = 5{,}21252 * 10^{-3}\ V \quad (59)$$

Por tanto, el mínimo valor que va a tomar la tensión de salida para entrar antes en régimen permanente:

$$V_{out\_init} = V_{outmedio} + \frac{\Delta V_{pp}}{2} = 48{,}00459374 + \frac{5{,}21252 * 10^{-3}}{2} = 48{,}0072\ V$$

5.3 Comparaciones

En este apartado se va a llevar a cabo en primer lugar una comparativa de las variables de estado del modelo real sin y con pérdidas con el circuito sin y con pérdidas implementado en la herramienta *Simulink* para comprobar que los modelos real desarrollados, los cuales se van a emplear como referencia del resto, son correctos.

Una vez obtenidos estos valores, se va a realizar una comparativa de éstos con las variables de estado obtenidas de forma teórica. Además, se va a realizar un cotejo del diseño *real* sin y con pérdidas frente a los modelos en coma fija y en coma flotante sin y con pérdidas.

Para algunos casos se ha escogido un tiempo de simulación de 100 ms y para otros 240 ms, para conseguir gráficas que representen el régimen transitorio hasta que llega al régimen permanente.

Se van a mostrar en las pruebas de cada uno de los modelos las gráficas de las variables de estado: corriente de entrada del circuito (i_L) y tensión de salida del circuito (v_c) y los errores medios obtenidos para cada una de las comparaciones que se van a realizar a continuación.

Todos los valores de las gráficas se han obtenido mediante las simulaciones realizadas en *ModelSim* y se han representado éstas empleando el programa *Matlab*.
Las gráficas en color rojo corresponden al modelo real y las gráficas en color azul al modelo en coma flotante o en coma fija dependiendo del apartado.

5.3.1 Modelos: resultados teóricos, Simulink y modelo real

En este apartado se va a llevar a cabo una comparativa de las representaciones obtenidas de los modelos implementados en la herramienta *Simulink* sin y con pérdidas con el modelo real sin pérdidas y con pérdidas por la razón que se ha comentado anteriormente.

Además, se va a realizar una comparativa de los resultados en régimen permanente de los valores de las variables de estado obtenidas en el modelo real simulado en la herramienta de simulación *ModelSim*, de forma teórica y en el modelo *Simulink* sin y con pérdidas.

5.3.1.1 Sin pérdidas: Simulink VS Real

En este apartado se lleva a cabo la comparativa del modelo real con el modelo en Simulink cuando no hay pérdidas. En este caso se ha simulado hasta 240 ms, ya que así se observa mejor cómo llega tanto la tensión de salida como la corriente de entrada a régimen permanente.
En la Figura 32 se compara la tensión de salida de ambos modelos:

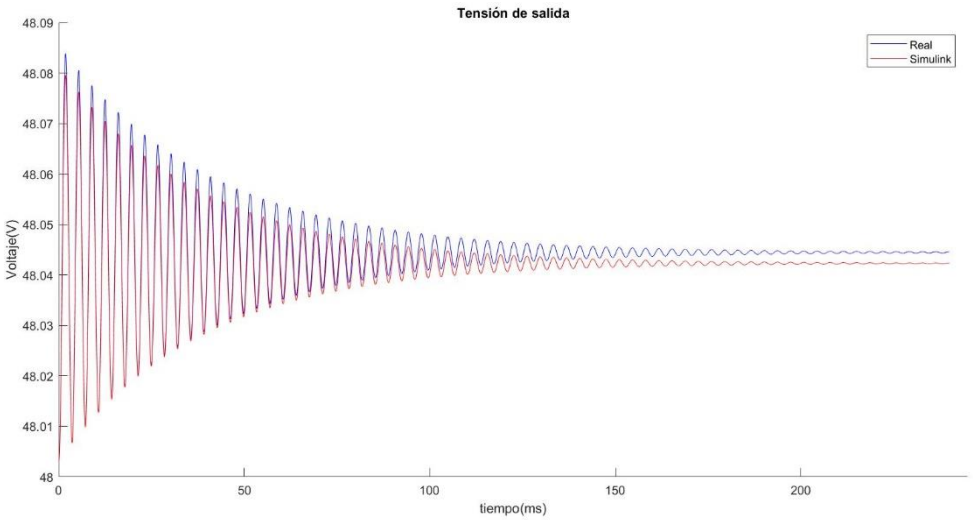

Figura 32. Comparativa tensión salida modelo real y modelo en Simulink

Se observa en esta gráfica que el error entre ambas tensiones es mínimo, del orden de $2 * 10^{-3}$. Además, a partir de 150 ms, se observa que la variación de la tensión de salida de ambos modelos es mínima.

En la Figura 33 se observa la corriente de entrada del circuito de ambos modelos:

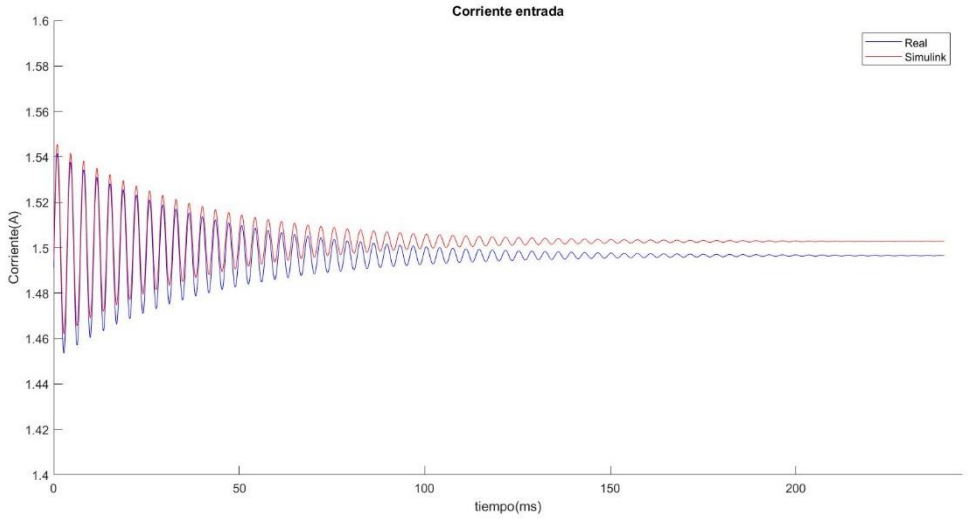

Figura 33. Comparativa corriente entrada modelo real y modelo en Simulink

En esta gráfica se observa que el error de la corriente obtenida de ambos modelos es mínimo, del orden de $6 * 10^{-3}$, y que a partir de 130 ms, la variación es mínima entre ambos modelos.

5.3.1.2 Con pérdidas: Simulink VS Real

En este apartado se lleva a cabo la comparativa del modelo real con el modelo en Simulink cuando hay pérdidas. En este caso se ha ejecutado hasta 100 ms ya que este tiempo de simulación es suficiente para observar cómo llega tanto la tensión de salida como la corriente de entrada a régimen permanente.
En la Figura 34 se compara la tensión de salida de ambos modelos:

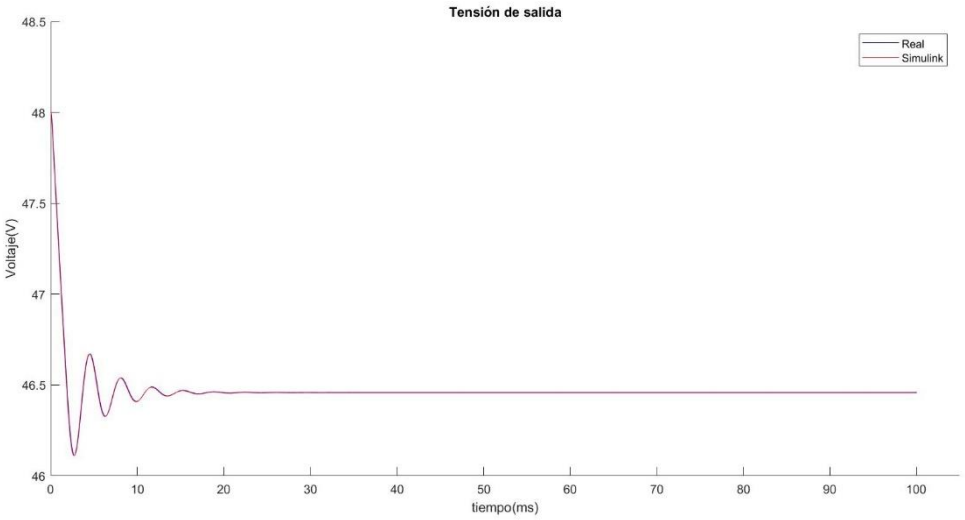

Figura 34. Comparativa tensión modelo real y modelo en Simulink

Se observa en esta gráfica que el error medio es mínimo entre ambos modelos, del orden de $5{,}2 * 10^{-2}$. Además, a partir de 20 ms, ambas representaciones se encuentran prácticamente superpuestas.

En la Figura 35 se observa la corriente de entrada de ambos modelos:

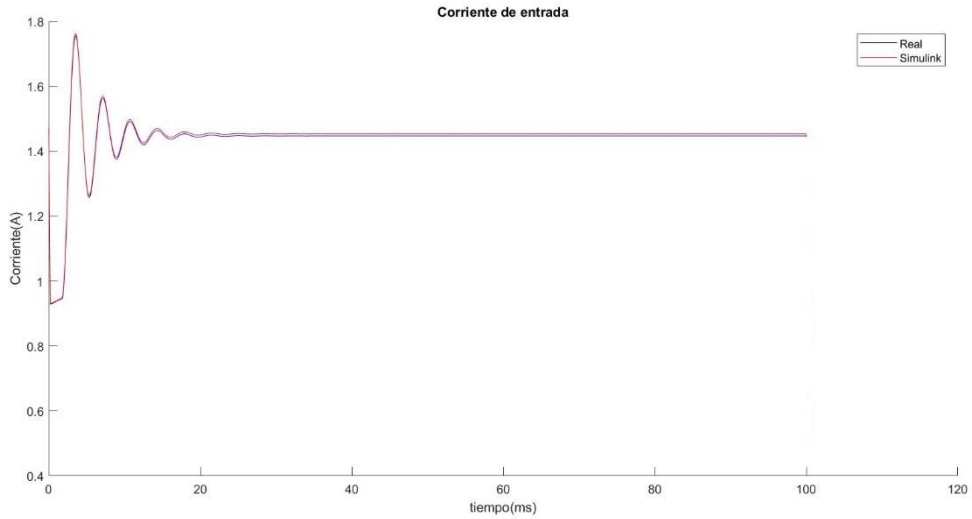

Figura 35. Comparativa corriente entrada modelo real y modelo en Simulink

Se observa en esta gráfica que el error de la corriente de entrada es mínimo entre ambos modelos, del orden de $3{,}8 * 10^{-2}$. Además, a partir de 30 ms, las corrientes obtenidas de ambos modelos se encuentran prácticamente superpuestas.

5.3.1.3 Resultados

En este apartado se realiza una comparativa de los resultados obtenidos de forma teórica, en ModelSim y en la herramienta Simulink.

En la Tabla 7 se muestran los valores de las variables de estado cuando no hay pérdidas:

Tensión de salida (v_{out})		
Téorico	**ModelSim**	**Simulink**
48	48,041	48,039
Corriente de entrada (i_L)		
Téorico	**ModelSim**	**Simulink**
1,497	1,4966	1,490

Tabla 7. Comparativa variables de estado sin pérdidas: teórico, ModelSim y Simulink

En esta comparativa se observa que los valores obtenidos en los tres modelos de ambas variables de estado (v_{out} e i_L) son muy similares. Así, se concluye que se ha llevado a cabo de forma correcta la transformación en VHDL de las variables de estado en el modelo real.

En la Tabla 8 se muestran los valores de las variables de estado cuando hay pérdidas:

Tensión de salida (v_{out})		
Téorico	**ModelSim**	**Simulink**
46,414	46,455	46,405
Corriente de entrada (i_L)		
Téorico	**ModelSim**	**Simulink**
1,448	1,448	1,41

Tabla 8. Comparativa variables de estado con pérdidas: teórico, ModelSim y Simulink

En esta comparativa se observa que los valores obtenidos en los tres modelos de ambas variables de estado (v_{out} e i_L) se diferencian algo más que en el caso anterior, pero siguen siendo valores similares, por ello podemos establecer el modelo real de ModelSim como modelo de referencia para las comparaciones que se realicen de ahora en adelante.

5.3.2 Modelo real y modelo en coma fija

5.3.2.1 Sin pérdidas

En este apartado se va a llevar a cabo la comparación del modelo real sin pérdidas con el modelo en coma fija sin pérdidas.

El tiempo de simulación en este modelo sin pérdidas es de 240 ms, ya que así se observa mejor cómo llegan las señales de tensión de salida y corriente de entrada a régimen permanente.

Por un lado, para el caso de la tensión de salida del circuito, se obtiene en un principio su representación sin filtrar. Esta gráfica se debe filtrar para obtener unos valores más específicos y más visuales. Por tanto, la gráfica final que se obtiene es la que se muestra en la Figura 36.

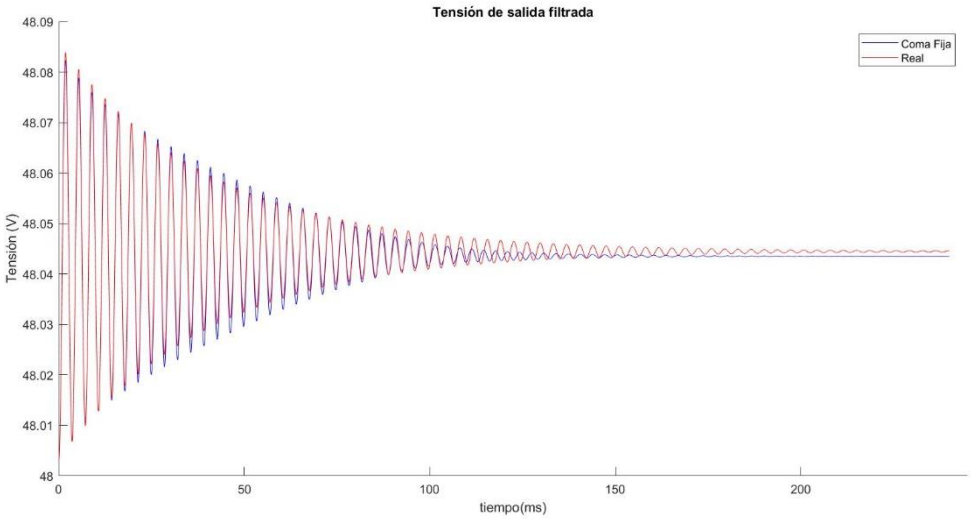

Figura 36. Comparación tensión salida filtrada modelos real y coma fija sin pérdidas

Una vez que se ha filtrado la gráfica, se observa que su oscilación a medida que van aumentando el tiempo es menor hasta que llega a régimen permanente. Su mínimo valor corresponde con la condición inicial de la tensión de salida, calculada para empezar en régimen permanente

Por otro lado, en la Figura 37 se muestra la corriente de entrada del circuito de ambos modelos frente al tiempo de simulación. Su representación se debe filtrar para obtener unos valores de señal muchos más claros:

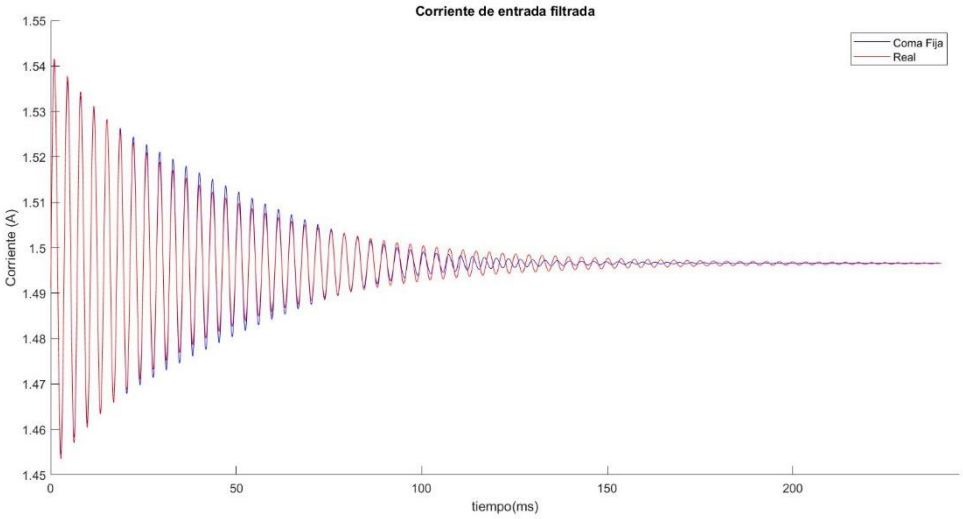

Figura 37. Comparación corriente entrada filtrada modelos real y coma fija sin pérdidas

En la gráfica de la corriente de entrada filtrada, se observa que a partir de 150 ms la corriente de ambos modelos es prácticamente la misma.

En las Figuras 36 y 37 se observa que el error es casi despreciable, hasta tal punto que prácticamente se superponen.

En la Tabla 9 se muestra el error calculado de i_L y v_c entre el modelo real y el modelo en coma fija sin pérdidas

Errores medios	
$\boldsymbol{i_L}$	$\boldsymbol{v_c}$
$8{,}88 * 10^{-4}$	$9{,}84 * 10^{-4}$

Tabla 9. Error medio entre modelo real y modelo en coma fija sin pérdidas

Estos errores se han calculado a partir de un script desarrollado en *Matlab*.
Tanto para la corriente de entrada como para la tensión de salida, se restan los valores obtenidos del modelo real y del modelo en coma fija. A continuación, mediante un bucle *for*, se calcula el valor absoluto de cada uno de los componentes del resultado de la resta de ambos modelos en cada iteración y se va sumando el valor calculado en la iteración anterior con la actual. Así, una vez finalizado el bucle se obtiene un único valor. Éste se divide entre el número de muestras del resultado de la resta entre ambos modelos.

5.3.2.2 Con pérdidas

En este apartado se va a llevar a cabo la comparación del modelo real con pérdidas con el modelo en coma fija con pérdidas.

Este modelo se ejecuta durante 100 ms ya que este tiempo de simulación es suficiente para observar cómo llegan tanto la tensión de salida como la corriente de entrada a régimen permanente.

Para el caso de la tensión de salida del circuito, primero se obtiene su representación sin filtrar. Esta gráfica se debe filtrar por la razón que se ha comentado anteriormente, y se obtiene la representación que se muestra a continuación:

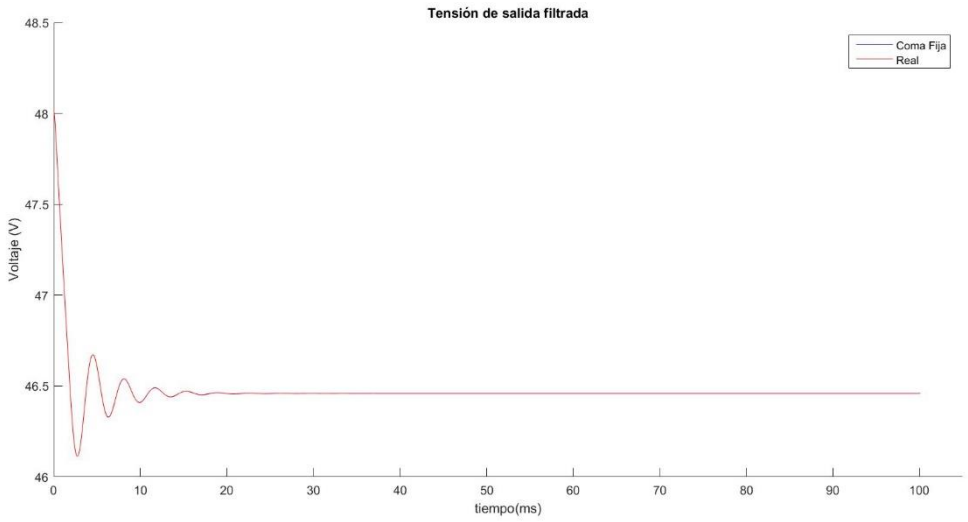

Figura 38. Comparación tensión salida filtrada modelos real y coma fija con pérdidas

En esta gráfica se observa que a partir de los 20 ms la tensión de salida se encuentra en régimen permanente, y en el resto de muestras la señal se encuentra en régimen transitorio. El valor máximo de esta señal corresponde con el valor inicial de la tensión de salida.

Para el caso de la corriente de entrada, ocurre lo mismo que con la tensión de salida: se obtiene en primer lugar una gráfica sin filtrar con el tiempo de simulación, el tiempo de muestreo y el número de muestras igual que en los casos anteriores. Una vez obtenida ésta, se debe filtrar por la misma razón que en los casos anteriores:

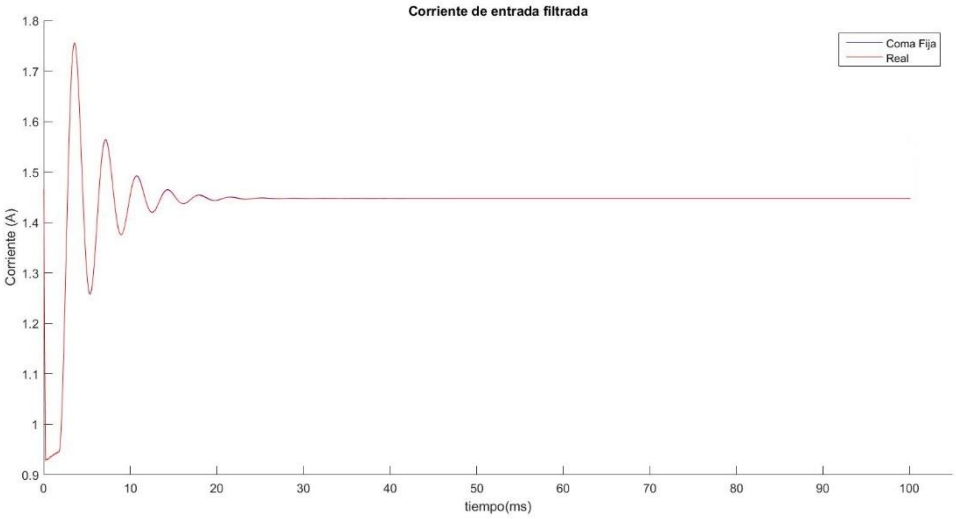

Figura 39. Comparación corriente entrada filtrada modelos real y coma fija con pérdidas

En esta gráfica se observa que a partir de los 20 ms la corriente de entrada se encuentra en régimen permanente.

En las representaciones de las Figuras 38 y 39 se observa que el error es nuevamente muy pequeño.

En la Tabla 10 se muestra el error calculado de i_L y v_c entre el modelo real y el modelo en coma fija con pérdidas:

Errores medios	
$\boldsymbol{i_L}$	$\boldsymbol{v_c}$
$3{,}07 * 10^{-4}$	$1{,}96 * 10^{-4}$

Tabla 10. Error medio entre modelo real y modelo en coma fija con pérdidas

Estos errores se han calculado a partir de un script desarrollado en *Matlab*. Se han calculado los errores de la misma forma que se ha comentado anteriormente, pero en este caso, se comparan la corriente de entrada y la tensión de salida del modelo real con pérdidas con el modelo en coma fija con pérdidas.

5.3.3 Modelo real y modelo en coma flotante

5.3.3.1 Sin pérdidas

En este apartado se va a llevar a cabo la comparación del modelo real sin pérdidas con el modelo en coma flotante sin pérdidas.

El tiempo de simulación en este modelo es de 240 ms, ya que así se observa mejor cómo llegan las señales de tensión de salida y corriente de entrada a régimen permanente.

Para el caso de la tensión de salida, en primer lugar se obtiene una representación de ambos modelos sin filtrar. Esta gráfica se debe filtrar por la razón que se ha comentado anteriormente y se obtiene la representación que se muestra a continuación:

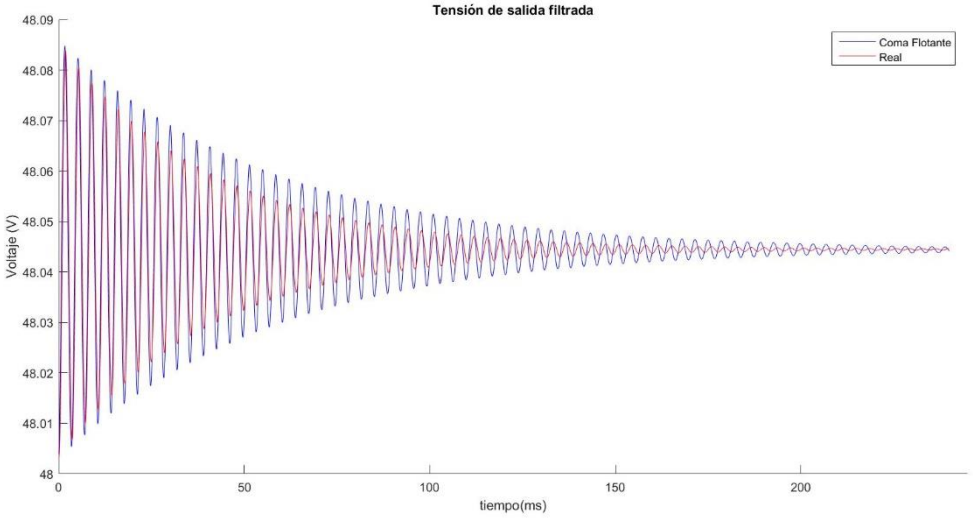

Figura 40. Comparación tensión salida filtrada modelos real y coma flotante sin pérdidas

En la gráfica de la tensión de salida filtrada se observa que sí que llega a régimen permanente cuando el tiempo de simulación es de 240 ms. Su mínimo valor corresponde con el valor inicial de la tensión de salida en el modelo.

Para el caso de la corriente de entrada, ocurre lo mismo que con la tensión de salida: se obtiene en primer lugar una gráfica sin filtrar con el tiempo de simulación, el tiempo de muestreo y el número de muestras igual que en los casos anteriores. Una vez obtenida ésta, se debe filtrar por la misma razón que en los casos anteriores:

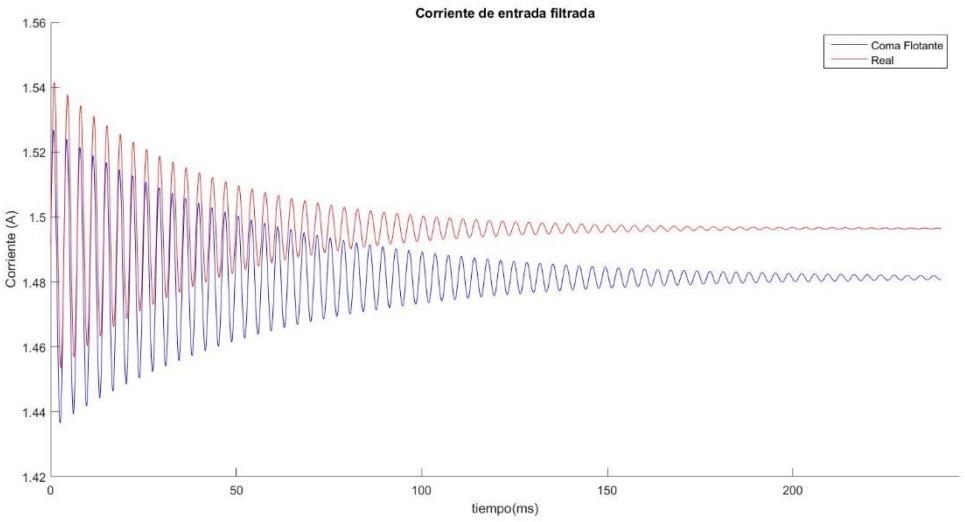

Figura 41. Comparación corriente entrada filtrada modelos real y coma flotante sin pérdidas

En esta gráfica se observa que la oscilación cada vez es menor hasta que llega a régimen permanente.

En las Figuras 40 y 41 se observa que el error es mayor que en la comparativa del modelo real con el de coma fija. Esta diferencia se debe a que este modelo tiene problemas de resolución, ya que emplea señales de 32 bits, de los cuales 24 bits son para la mantisa: hay un "1" fijo y 23 bits restantes. Por tanto, no se obtiene suficiente resolución debido al número fijo de bits en la parte fraccionaria. Con este problema lo que se consigue es que en cada ciclo de reloj, el error se vaya acumulando y por tanto, la precisión cada vez es menor.

En la Tabla 11 se muestra el error calculado de i_L y v_c entre el modelo real y el modelo en coma flotante sin pérdidas:

Errores medios	
$\boldsymbol{i_L}$	$\boldsymbol{v_c}$
0,0153	0,0048

Tabla 11. Error medio entre modelo real y modelo en coma flotante sin pérdidas

Estos errores se han calculado a partir de un script desarrollado en *Matlab*. Se han calculado los errores de la misma forma que en los casos anteriores, pero la diferencia con el resto es que se lleva a cabo una comparativa de la corriente de entrada y la tensión de salida del modelo real sin pérdidas con el modelo en coma flotante sin pérdidas.

5.3.3.2 Con pérdidas

En este apartado se va a llevar a cabo la comparación del modelo real con pérdidas con el modelo en coma flotante con pérdidas.

Para el caso de la tensión de salida, en primer lugar se obtiene una representación de ambos modelos sin filtrar. Esta gráfica se debe filtrar por la razón que se ha comentado anteriormente:

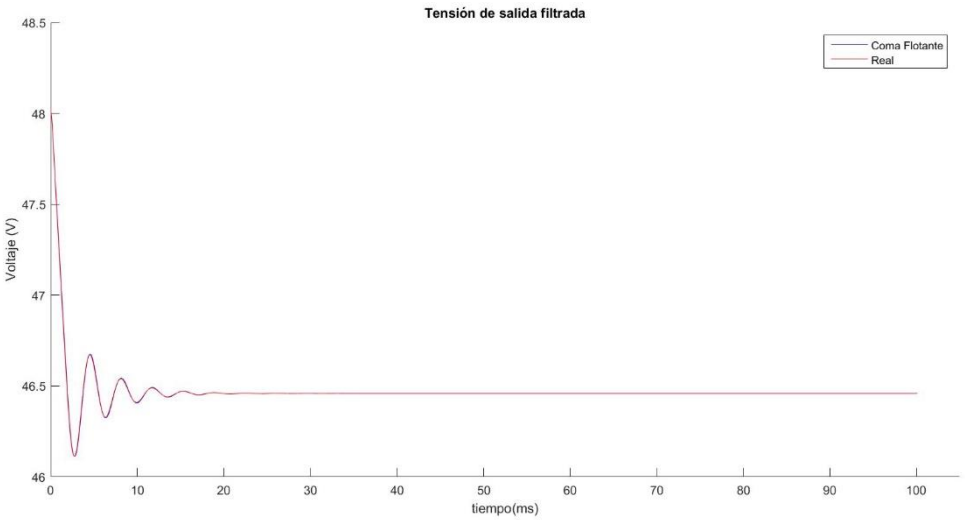

Figura 42. Comparación tensión salida filtrada modelos real y coma flotante sin pérdidas

En esta gráfica se observa que hasta los 20 ms la tensión de salida se encuentra en régimen transitorio, y a partir de ahí en régimen permanente. Su máximo valor corresponde con el valor inicial de la tensión de salida.

Para el caso de la corriente de entrada, ocurre lo mismo que con la tensión de salida: se obtiene en primer lugar una gráfica sin filtrar con el tiempo de simulación, el tiempo de muestreo y el número de muestras igual que en los casos anteriores. Una vez obtenida ésta, se debe filtrar por la misma razón que en los casos anteriores:

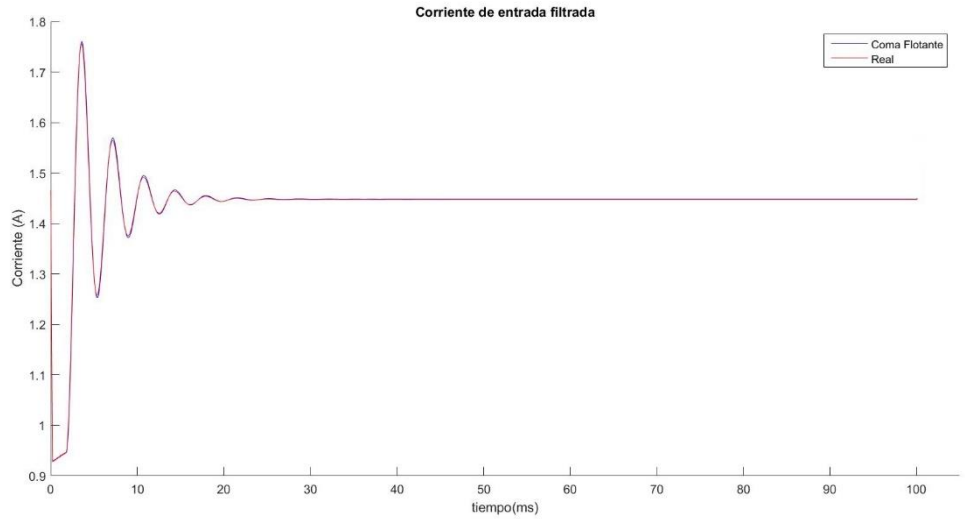

Figura 43. Comparación corriente entrada filtrada modelos real y coma flotante con pérdidas

En esta gráfica se observa que hasta los 20 ms se encuentra en régimen transitorio y que a partir de ahí, la corriente se encuentra en régimen permanente. Su mínimo valor es de 0,95 A y su máximo valor es de 1,77 A.

En las Figuras 42 y 43 se observa que el error es menor que en el caso en el que se compara el modelo real con el modelo en coma flotante sin pérdidas, pero el error sigue siendo mayor

si se compara con el modelo en coma fija por la misma razón que se ha comentado anteriormente.

En la Tabla 12 se muestra el error calculado de i_L y v_c entre el modelo real y el modelo en coma flotante sin pérdidas:

Errores medios	
$\boldsymbol{i_L}$	$\boldsymbol{v_c}$
$1{,}30 * 10^{-3}$	$1{,}10 * 10^{-3}$

Tabla 12. Error medio entre modelo real y modelo en coma flotante con pérdidas

Estos errores se han calculado a partir de un script desarrollado en *Matlab*. Se han calculado los errores de la misma forma que en los casos anteriores, pero la diferencia con el resto es que se lleva a cabo una comparativa de la corriente de entrada y la tensión de salida del modelo real con pérdidas con el modelo en coma flotante con pérdidas.

5.3.4 Errores modelos: real, coma fija y coma flotante

Por un lado, se observa en las Tablas 9, 10, 11 y 12 que el error calculado en régimen permanente en los modelos en coma flotante, sin pérdidas y con pérdidas, es mayor que en los modelos en coma fija, ya que se están empleando mayor número de bits en la parte fraccionaria en coma fija que en el modelo en coma flotante. Por tanto, el modelo en coma fija al usar más bits la precisión es mayor.

Por otro lado, se realiza una comparativa de los errores de las variables de estado obtenidos en los modelos en coma fija y en coma flotante con respecto al modelo real sin y con pérdidas.

En la Tabla 13 se muestra un resumen de los errores de las variables de estado cuando no hay pérdidas:

Tensión de salida (v_{out})	
Modelo en coma fija	**Modelo en coma flotante**
$9{,}84 * 10^{-4}$	0,0048
Corriente de entrada (i_L)	
Modelo en coma fija	**Modelo en coma flotante**
$8{,}88 * 10^{-4}$	0,0153

Tabla 13. Errores variables de estado: modelos en coma fija y en coma flotante con respecto al modelo real sin pérdidas

En esta comparativa se observa como el valor con mayor diferencia es el del modelo de coma flotante, por la misma razón que se ha mencionado anteriormente: no se pueden emplear suficientes bits en la mantisa de las señales del modelo en coma flotante para que la precisión sea la misma que en el modelo en coma fija. El valor del modelo en coma fija es prácticamente el mismo. Gracias a estos valores, se puede llegar a la conclusión que se ha llevado a cabo de forma correcta la transformación en VHDL de las variables de estado.

En la Tabla 14 se muestra un resumen de los errores de las variables de estado cuando hay pérdidas:

Tensión de salida (v_{out})	
Modelo en coma fija	**Modelo en coma flotante**
$1{,}96 * 10^{-4}$	$1{,}10 * 10^{-3}$
Corriente de entrada (i_L)	
Modelo en coma fija	**Modelo en coma flotante**
$3{,}07 * 10^{-4}$	$1{,}3 * 10^{-3}$

Tabla 14. Errores variables de estado: modelos en coma fija y en coma flotante con respecto al modelo real hay pérdidas

En este caso se observa que el modelo con mayor diferencia es el modelo en coma flotante por la misma razón que se acaba de comentar.

5.3.5 Influencia de las pérdidas eléctricas

Para comprobar el efecto que tienen las pérdidas eléctricas en el circuito del convertidor Flyback, se ha llevado a cabo un estudio donde se han analizado qué pérdidas son las que influyen más en el convertidor Flyback.

Para confirmar esta afirmación, se han realizado diversas pruebas modificando el valor de las pérdidas, es decir, empleando valores mayores de los que se indican en la Tabla 5. Se lleva a cabo esta modificación ya que así se observa mucho mejor cómo influyen estas pérdidas en los valores de las variables de estado obtenidas.

En la Tabla 15 se muestran los valores que se han empleado para las pérdidas eléctricas:

Componente	**Valor**
Resistencia primera espiral transformador (R_{l1})	2 Ω
Resistencia primera espiral transformador (R_{l2})	2 Ω
Resistencia transistor (R_t)	2 Ω
Resistencia diodo (R_d)	2 Ω
Resistencia condensador (R_c)	2 Ω
Tensión del diodo (V_d)	1,3 V

Tabla 15. Análisis pérdidas: aumento valores resistencias

A continuación, se muestran en distintas tablas, y por tanto, distintos casos, cómo influyen las pérdidas eléctricas (%) dependiendo de qué resistencias tomen valor en el cálculo de las variables de estado de forma analítica y en la simulación.

El porcentaje de la influencia de dichas pérdidas se calcula dividiendo los valores obtenidos de las variables de estado en los distintos casos presentados a continuación con las variables de estado calculadas en el modelo sin pérdidas.

Por un lado, el primer caso ocurre cuando no tienen valor las pérdidas eléctricas. Por otro lado, el segundo caso ocurre cuando solo la resistencia de la primera espiral del transformador toma valor.

Primer caso					Segundo caso				
Pérdidas		Var. estado	Analítico	Simulado	Pérdidas		Var. estado	Analítico	Simulado
R_l	0	i_L	0%	0%	R_l	2	i_L	3%	3%
R_{l2}	0				R_{l2}	0			
R_t	0				R_t	0			
R_d	0	v_c	0%	0%	R_d	0	v_c	3%	3%
R_c	0				R_c	0			
V_d	0				V_d	0			

Tabla 16. Efecto pérdidas: primer y segundo caso

Mientras que el tercer caso solo ocurre cuando la resistencia del transistor toma valor, el cuatro caso se obtiene cuando toma valor la resistencia de la segunda espiral del transformador.

Tercer caso					Cuarto caso				
Pérdidas		Var. estado	Analítico	Simulado	Pérdidas		Var. estado	Analítico	Simulado
R_l	0	i_L	3%	3%	R_l	0	i_L	6%	5%
R_{l2}	0				R_{l2}	2			
R_t	2				R_t	0			
R_d	0	v_c	3%	3%	R_d	0	v_c	6%	6%
R_c	0				R_c	0			
V_d	0				V_d	0			

Tabla 17. Efecto pérdidas: tercer y cuarto caso

Por un lado, el quinto caso ocurre cuando tanto la resistencia como la tensión del diodo toman valor. Por otro lado, el sexto ocurre cuando sólo toma valor la resistencia del condensador.

Quinto caso					Sexto caso				
Pérdidas		Var. estado	Analítico	Simulado	Pérdidas		Var. estado	Analítico	Simulado
R_l	0	i_L	8%	8%	R_l	0	i_L	2%	1%
R_{l2}	0				R_{l2}	0			
R_t	0				R_t	0			
R_d	2	v_c	8%	8%	R_d	0	v_c	2%	2%
R_c	0				R_c	2			
V_d	1,3				V_d	0			

Tabla 18. Efecto pérdidas: quinto y sexto caso

Finalmente, el séptimo caso ocurre cuando todas las pérdidas eléctricas toman valor.

Séptimo caso				
Pérdidas		Var. estado	Analítico	Simulado
R_l	2	i_L	18%	17%
R_{l2}	2			
R_t	2			
R_d	2	v_c	19%	19%
R_c	2			
V_d	1,3			

Tabla 19. Efecto pérdidas: séptimo caso

Una vez realizado el análisis, se observa que cuando la resistencia del condensador es la única que toma valor (sexto caso) la influencia de las pérdidas en el circuito del convertidor Flyback es mínima. Por otro lado, aumenta dicha influencia cuando tiene pérdidas la segunda espiral del transformador (cuarto caso) y el diodo (quinto caso). Finalmente, se observa que el circuito indicado tiene más perdidas cuando toman valor todas las pérdidas de los elementos comentados como era de esperar. Este efecto es el que se observa en la Tabla 19.

Además, gracias a que se ha desarrollado un modelo con estas pérdidas, se puede conseguir un sistema con mayor precisión y con un comportamiento muy similar al de un modelo real.

5.4 Área y tiempo

En esta sección se van a mostrar los resultados obtenidos del área y el tiempo total de ejecución cuando se sintetizan los modelos en coma fija y en coma flotante sin pérdidas y con pérdidas en la FPGA Xilinx Zynq-7000 (XC7A35TICSG324-1L). No se habla del modelo real ya que como se ha comentado varias veces anteriormente, no puede ser implementado en una FPGA.

Para obtener el número de recursos empleados por cada modelo y el tiempo que tarda en ejecutarse cada uno de éstos, se ha empleado la herramienta: *Vivado 2018.2*.

Por un lado, el número de recursos que emplea cada modelo se obtienen gracias al uso de este programa y son: el número *Look Up Tables* (LUT), el número de registros (*Flip Flop* (FF)) y el número de multiplicadores utilizados (DSP).

Por otro lado, para obtener el tiempo total de ejecución de cada modelo, se debe emplear el parámetro *Worst Negative Slack* (WNS), el cual indica el tiempo del peor camino del modelo. El tiempo que tarda en ejecutarse el modelo se calcula realizando la siguiente operación: $tiempo = t_{ciclo} - WNS$, donde t_{ciclo} es el periodo del ciclo de reloj (20 ns).

En la Tabla 20, se muestran los recursos empleados y el tiempo total de ejecución de los dos modelos comentados sin pérdidas:

Sin Pérdidas	**Área**			**Tiempo total**	
Modelos	**Número LUT**	**Número Registros**	**Número DSP**	**WNS (ns)**	**Tiempo (ns)**
Modelo coma fija	1143	62	0	-3.181	23,181
Modelo coma flotante	1418	86	0	0,996	19,004

Tabla 20. Área y tiempo modelos sin pérdidas

En los modelos sin pérdidas se observa que el tiempo de ejecución en el modelo en coma flotante es menor que en el modelo en coma fija. Esto se debe a que hay algunas señales de éste último que tienen un tamaño superior a 32 bits.

En la Tabla 21, se muestran los recursos empleados y el tiempo total de ejecución de los dos modelos comentados con pérdidas:

Con Pérdidas	**Área**			**Tiempo total**	
Modelos	**Número LUT**	**Número Registros**	**Número DSP**	**WNS (ns)**	**Tiempo (ns)**
Modelo coma fija	889	86	10	-16,959	36,959
Modelo coma flotante	1747	110	8	-39,649	59,649

Tabla 21. Área y tiempo modelos con pérdidas

Sin embargo, en los modelos sin pérdidas se observa que el tiempo de ejecución es menor en el modelo en coma fija, ya que todas sus señales están compuestas por muchos menos bits que en el modelo en coma flotante.

Además, se observa claramente cómo afecta la incorporación de las pérdidas a ambos modelos, ya que hay un aumento tanto de recursos como tiempo de ejecución. Esto se debe a que los modelos con pérdidas son bastante más complejos que los modelos sin pérdidas y esto implica que se ejecuten un número mayor de operaciones en el sistema. Por ejemplo, mientras que el modelo sin pérdidas emplea 4 multiplicadores, el modelo con pérdidas usa 8 multiplicadores.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Debido a la necesidad de simulación de sistemas en tiempo real en los últimos años han surgido algunas técnicas, entre las que destaca HIL (*Hardware In The Loop*), que mejora la velocidad de simulación. En este Trabajo de Fin de Master se lleva a cabo un modelo HIL de un convertidor Flyback mediante el lenguaje VHDL. Se define el modelo de planta de este convertidor obteniendo las ecuaciones diferenciales correspondientes y a partir de éstas, se desarrollan tres modelos distintos empleando aritméticas diferentes: modelo real, modelo en coma fija y modelo en coma flotante. Además, se han realizado estos mismos modelos con pérdidas para que el sistema tenga un comportamiento lo más real posible.

En primer lugar, se han calculado de forma teórica y con la herramienta Simulink las variables de estado sin y con pérdidas. Con estos resultados se ha comprobado que el modelo real sin y con pérdidas es correcto. Así, una vez que se ha corroborado lo anterior, se puede emplear este modelo como modelo de referencia para el resto. La aritmética real no pueden ser sintetizada en una FPGA, como se ha comentado durante todo este proyecto, y por tanto, se desarrollan dos modelos más que sí que pueden: modelo en coma fija y modelo en coma flotante.

Por un lado, se lleva a cabo la comparativa del modelo real con los modelos en coma fija y coma flotante para comprobar que efectivamente el modelo en coma fija es el que mayor precisión y menor error tiene de los modelos comentados. Esto es así porque los bits del modelo en coma fija están más optimizados. Cuando se añaden pérdidas a estos modelos, el comportamiento es muy similar. Se observa que las variables de estado del modelo en coma fija sin y con pérdidas han mejorado con respecto al modelo en coma flotante sin y con pérdidas. En cuanto a los modelos sin pérdidas, el error de la tensión de salida y la corriente de entrada del modelo en coma fija han mejorado un 18% y un 5,22% con respecto al modelo en coma flotante. Por otro lado, en los modelos con pérdidas, se observa una mejora del 17% y del 23%. Por tanto, se concluye que el modelo en coma flotante en ambos casos es el que mayor error tiene.

Finalmente, se sintetizan los modelos en coma fija y en coma flotante sin pérdidas y con pérdidas en la FPGA Xilinx Zynq-7000 y se verifica que se pueden ejecutar en tiempo real. Se observa que aumentan el número de recursos en los modelos con pérdidas al ser más complejos, y por tanto, esto hace que el número de operaciones a ejecutar en el sistema sea mayor. Por tanto, el modelo en coma flotante con pérdidas es el que más tarda en finalizar su ejecución (59,649 ns).

Sin embargo, se observa que en los modelos sin pérdidas el tiempo de ejecución es mayor en coma fija, y esto se debe a que algunas de las señales del modelo en coma fijan exceden los 32 bits, que son el número de bits de float. En este caso, hay 1143 LUTs, 62 registro y 0 DSPs. Aunque este comportamiento no es el esperado y es debido a que el modelo desarrollado no se encuentra optimizado al 100%, son los resultados que se han obtenido.

Por lo tanto, en este Trabajo de Fin de Máster, se ha verificado que los modelos que se han desarrollado se pueden emplear para llevar a cabo la simulación en tiempo real de convertidores de potencia, en concreto, para un convertidor Flyback.

6.2 Trabajo futuro

En este Trabajo de Fin de Máster se ha llevado a cabo el modelo con pérdidas de un convertidor Flyback.

Como se ha comentado anteriormente, el modelo desarrollado en coma fija sin pérdidas no está optimizado porque tarda más en ejecutarse que el modelo en coma flotante. Por tanto, como trabajo futuro se podría realizar un nuevo diseño para conseguir una mayor optimización y así aprovecharlas ventajas que da la coma fija.

Por otro lado, se ha observado que el modelo en coma fija no es muy eficiente, ya que en el momento en el que se quieran modificar las condiciones de simulación, se debe volver a diseñar y realizar los cálculos de nuevos. Por tanto, una mejora futura sería desarrollar el modelo en coma fija parametrizable, aplicando las directrices de [19], el cual permite que el modelo se adapte al modificar la posición de la coma de una señal sin tener que sintetizar de nuevo el diseño, y en diferentes situaciones de simulación.

Además, como en este proyecto se ha llevado a cabo el estudio de cómo afectan las pérdidas eléctricas de primer orden a los modelos, se podría elaborar este mismo estudio, pero con pérdidas de segundo orden, para comprobar qué efectos son los que producen y cómo es su nivel de complejidad.

Referencias

[1] F. López-Colino, A. Sánchez, A. de Castro and J. Garrido, *Modeling of power converters for debugging digital controllers through FPGA emulation*, in Power Electronics and Applications (EPE) 15th European Conference on, pp. 1-8, Sept. 2013.

[2] A. Sanchez, A. de Castro and J. Garrido, *A Comparison of Simulation and Hardware-in-the- Loop Alternatives for Digital Control of Power Converters*, in IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 491-500, Aug. 2012.

[3] R.B. Erickson, D.Maksimovic, *Fundamentals of Power Electronics*, in Kluwer Academic Publishers, no. 2, 2000.

[4] Mario Mañana Canteli, *Regulación, control y protección de máquinas eléctricas*, Departamento de ingeniería eléctrica y energética. Universidad de Cantabria.

[5] Ned Mohan, Tore M. Undeland, William P.Robbins, *Electrónica de Potencia. Convertidores, aplicaciones y diseño*, 3º Edición – 2009.

[6] Rafael Ramos Lara, *Sistemas digitales de control en tiempo discreto*, Departamento de Ingeniería electrónica. Universidad Politécnica de Cataluña. Febrero 2007.

[7]A. de Castro, *Aplicación del Control Digital Basado en Hardware Específico para Convertidores de Potencia Conmutados*, PhD thesis, Universidad Politécnica de Madrid, 2003.

[8] Miguel Gudino, *Tipos de convertidores interruptores de CC a CC*, Arrow Electronics, 12 julio de 2017.

[9] Alberto Sánchez, *Aportaciones mediante implementación basada en sistemas embebidos al control digital de convertidores conmutados*, Tesis Doctoral, Universidad Autónoma de Madrid

[10] A. Prodic and D. Maksimovic, *Mixed-signal simulation of digitally controlled switching converters*, in Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop on, Jun. 2002.

[11] P. Zumel, M. García-Valderas, A. Lázaro, C. Loópez-Ongil, and A. Barrado, *Co-simulation psim-modelsim oriented to digitally controlled switching power Converters*, in Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop, Junio 2010.

[12] A. de Castro, T. Riesgo, O. Garcia, and R. Prieto, *Comparing vhdl and vhdl-ams for modelling and simulation of power converters with digital control*, in XVIII Conference on Design of Circuits and Integrated Systems (DCIS), Nov. 2003.

[13] Juan Carlos Martínez Quintero y Jaime Eduardo Andrade Ramírez, *Implementación de controladores en sistemas retroalimentados usando electrónica embebida y simulación hardware in the loop*, Universidad Tecnológica de Pereira (Colombia), 2013.

[14] Alfonso Lago Ferreiro, Andrés Nogueiras Meléndez, Carlos Martínez-Peñalver Freire, Jorge Marcos Acevedo, *Metodología de Diseño de un Convertidor Flyback Orientado a la Docencia*, Departamento de Tecnología Electrónica, Universidad de Vigo.

[15] Dr. Peter Waeltermann, *Hardware-in-the-Loop: The Technology for Testing Electronic Controls in Vehicle Engineering*, dSpace Inc, Marzo 2016.

[16] Daniel Mozos Muñoz, *Aritmética en punto flotante*, Facultad de Informática.

[17] Balduino Blanqué Molina, *Simulación interactiva de motores de reluctancia autoconmutados*, Departamento de ingeniería eléctrica, Universidad politécnica de Cataluña, noviembre 2017.

[18] Oscar Goñi, Alberto Sanchez, Elías Todorovich, and Angel de Castro, Member, *Resolution Analysis of Switching Converter Models for Hardware-in-the-Loop*, IEEE Transactions on industrial informatics, vol.10, no.2 mayo 2014.

[19] Irene Villar, Alberto Sanchez, Angel de Castro, Fernando López-Colino y Javier Garrido, *Emulación de convertidores conmutados utilizando modelos en coma fija parametrizable*, Hardware and Control Technology Laboratory, Dpto. TEC, Escuela Politecnica Superior, Universidad Autónoma de Madrid

Glosario

ADC	Analog-to-Digital Converter
AC	Alternating Current
ADC	Analog-to-Digital Converter
CCM	Continuous Conduction Mode
DAC	Digital to Analog Converter
DC	Direct Current
DCM	Discontinuous Conduction Mode
DSP	Digital Signal Processor
FPGA	Field-Programmable Gate Array
HIL	Hardware-In-the-Loop
MOSFET	Metal-oxide-semiconductor Field-effect transistor
PSIM	Physical security information management
VHDL	VHSIC Hardware Description Language

Anexos

Anexo I: Código convertidor Flyback real sin pérdidas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;

entity ConverterFlyback is
  port (
        --In
        Clk : in std_logic;
        Reset : in std_logic;
        Mosfet : in std_logic; -- On = '1', off = '0'
        Vg : in real;
        Ir : in real;
        -- Out
        Iin : out real;
        Vout : out real;
        Vout_debug:out real;
        Il_debug:out real
  );
end ConverterFlyback;

architecture Behavioral of ConverterFlyback is

  constant C : real := 0.000440;
  constant L : real := 0.000352;
  constant dt : real := 0.000000020;

  signal iL : real := 0.0;
  signal voutAux : real := 48.0;
  signal iLAdd, voutAuxAdd : real := 0.0;
  signal Iinaux:real:=0.0;

  constant VOINIT : real := 0.0;
  constant ILINIT : real := 0.0;

  constant dtL : real := dt/L;
  constant dtC : real := dt/C;

  constant N1 : real := 1.0;
  constant N2 : real := 1.0;
  constant n : real := N1/N2;

begin

  Vout <= voutAux;
  Iin <= Iinaux;

  SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
  begin
        if Mosfet = '1' then -- Closed switch
                voutAuxAdd <= -Ir;
                iLAdd <= Vg;
                Iinaux <= iL;
```

```
        else -- Open switch
                if iL > 0.0 then -- CCM (MODO CONTINUO)
                        voutAuxAdd <= iL/n-Ir;
                        iLAdd <= -voutAux/n;
                else -- DCM (MODO DISCONTINUO)
                        voutAuxAdd <= -Ir;
                        iLAdd<= 0.0;
                end if;
                Iinaux <= 0.0;
        end if;
  end process SWITCHMUX;

  DIFFEQ: process(Clk, Reset)
  begin
        if Reset = '1' then -- Se resetean las señales al valor inicial
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
                iL <= iL + iLAdd*dtL;
                voutAux <= voutAux + voutAuxAdd*dtC;
        end if;
  end process DIFFEQ;
  Vout_debug<=voutAux;
  Il_debug<=iL;

end Behavioral;
```

Anexo II: Código convertidor Flyback coma fija sin pérdidas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.math_real.all;
use IEEE.fixed_float_types.all;
use IEEE.fixed_pkg.all;

entity ConverterFlybackComaFijaNV is
	port (
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic; -- On = '1', off = '0'
		Vg : in std_logic_vector(12 downto 0); -- 13 bits
		Ir : in std_logic_vector(12 downto 0); -- 13 bits
		-- Out
		Iin : out std_logic_vector(12 downto 0); -- 13 bits
		Vout : out std_logic_vector(12 downto 0); -- 13 bits
		Vout_debug: out real;
		Il_debug: out real
	);
end ConverterFlybackComaFijaNV;

architecture Behavioral of ConverterFlybackComaFijaNV is

	constant dtL : sfixed(-14 downto -26) := to_sfixed(0.00005681818182,-14,-26);
	constant dtC : sfixed(-14 downto -34) := to_sfixed(0.00004545454545,-14,-34);

	signal iL : sfixed(6 downto -22):= (others=>'0');
	signal voutAux : sfixed(7 downto -25):= to_sfixed(48,7,-25);
	signal voutAuxAdd: sfixed(9 downto -23):= (others=>'0');
	signal iLAdd: sfixed(7 downto -24):= (others=>'0');

	signal voutAuxAdd_aux: sfixed(7 downto -5):= (others=>'0');
	signal iLAdd_aux: sfixed(6 downto -6):= (others=>'0');
	signal Iinaux:sfixed(6 downto -22):= (others=>'0');

	signal s_Ir:sfixed(1 downto -11):= (others=>'0');
	signal s_Vg:sfixed(7 downto -5):= (others=>'0');
	signal s_R : sfixed(6 downto -6):= to_sfixed(46.08,6,-6);

	constant VOINIT :  sfixed(7 downto -25):= (others=>'0');
	constant ILINIT : sfixed(6 downto -22):= (others=>'0');

	constant n : sfixed(1 downto -1) := to_sfixed(1,1,-1);

begin
	s_Ir <= to_sfixed(Ir,s_Ir);
	s_Vg <= to_sfixed(Vg,s_Vg);

	voutAuxAdd_aux <= resize(voutAux, voutAuxAdd_aux);
	iLAdd_aux <= resize(Iinaux,iLAdd_aux);
	Vout <= to_slv(voutAuxAdd_aux);
	Iin <= to_slv(iLAdd_aux);

	SWITCHMUX: process(Mosfet, iL, voutAux, s_Vg,s_Ir,iLAdd,voutAuxAdd)
		if Mosfet = '1' then -- Closed switch
			voutAuxAdd <= resize(-s_Ir,voutAuxAdd);
			iLAdd <= resize(s_Vg,iLAdd);
```

```
                Iinaux <= iL;
        else -- Open switch
                if iL > 0.0 then -- CCM (MODO CONTINUO)
                        --voutAuxAdd <= resize((iL/n),s_temp)-s_Ir;
                        --voutAuxAdd <= resize((iL/n)-s_Ir,voutAuxAdd);
                        voutAuxAdd <= (iL/n)-s_Ir;
                        iLAdd <= resize(-(voutAux/n),iLAdd);
                else -- DCM (MODO DISCONTINUO)
                        voutAuxAdd <= resize(-s_Ir,voutAuxAdd);
                        iLAdd<= (others=>'0');
                end if;
                Iinaux <= (others=>'0');
        end if;
    end process SWITCHMUX;

    DIFFEQ: process(Clk, Reset)
    begin
        if Reset = '1' then -- Se resetean las señales al valor inicial
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
                iL <= resize(iL + iLAdd*dtL,iL);
                voutAux <= resize(voutAux + voutAuxAdd*dtC,voutAux);
        end if;
    end process DIFFEQ;
        Vout_debug<=to_real(voutAux);
    Il_debug<=to_real(iL);
end Behavioral;
```

Anexo III: Código convertidor Flyback coma flotante sin pérdidas

```
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.fixed_float_types.all;
use IEEE.fixed_pkg.all;
use IEEE.math_real.all;
Library floatfixlib;
use floatfixlib.float_pkg.all;

entity ConverterFlyback_Float32 is
  port (
        --In
        Clk : in std_logic;
        Reset : in std_logic;
        Mosfet : in std_logic; -- On = '1', off = '0'
        Vg : in float32;
        Ir : in float32;
        -- Out
        Iin : out float32;
        Vout : out float32;
        Vout_debug:out real;
        Il_debug:out real
  );
end ConverterFlyback_Float32;

architecture Behavioral of ConverterFlyback_Float32 is

  constant C : float32 := to_float(0.000440);
  constant L : float32 := to_float(0.000352);
  constant dt : float32 := to_float(0.000000020);

  signal iL : float32 := to_float(0.0);
  signal voutAux : float32 := to_float(48.0);
  signal iLAdd, voutAuxAdd : float32 := to_float(0.0);
  signal Iinaux:float32:=to_float(0.0);

  constant VOINIT : float32 := to_float(48.0072);
  constant ILINIT : float32 := to_float(0.546845);

  constant dtL : float32 := dt/L;
  constant dtC : float32 := dt/C;

  constant N1 : float32 := to_float(1.0);
  constant N2 : float32 := to_float(1.0);
  constant n : float32 := N1/N2;

begin
  Vout <= voutAux;
  Iin <= Iinaux;

  SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
        begin
        if Mosfet = '1' then -- Closed switch
                voutAuxAdd <= -Ir;
                iLAdd <= Vg;
                Iinaux <= iL;
        else -- Open switch
                if iL > 0.0 then -- CCM (MODO CONTINUO)
```

```
                        voutAuxAdd <= iL/n-Ir;
                        iLAdd <= -voutAux/n;
                else -- DCM (MODO DISCONTINUO)
                        voutAuxAdd <= -Ir;
                        iLAdd<= to_float(0.0);
                end if;
                Iinaux <= to_float(0.0);
        end if;
  end process SWITCHMUX;


  DIFFEQ: process(Clk, Reset)
  begin
        if Reset = '1' then -- Se resetean las señales al valor inicial
                voutAux <= VOINIT;
                iL <= ILINIT;
        elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
                iL <= iL + iLAdd*dtL;
                voutAux <= voutAux + voutAuxAdd*dtC;
        end if;
  end process DIFFEQ;
  Vout_debug<=to_real(voutAux);
  Il_debug<=to_real(iL);

end Behavioral;
```

Anexo IV: Código convertidor Flyback real con pérdidas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.math_real.all;
use IEEE.fixed_float_types.all;
use IEEE.fixed_pkg.all;

entity ConverterFlybackRealConPerdidas is
	port (
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic; -- On = '1', off = '0'
		Vg : in real;
		Ir : in real;
		-- Out
		Iin : out real;
		Vout : out real;
		Vout_debug:out real;
		Il_debug:out real
	);
end ConverterFlybackRealConPerdidas;

architecture Behavioral of ConverterFlybackRealConPerdidas is

	constant C : real := 0.000440;
	constant L : real := 0.000352;
	constant dt : real := 0.000000020;

	signal iL,voutAuxTempInit : real := 0.0;
	signal voutAux,vCAux : real := 48.0;
	signal iLAdd, vCAdd,voutAuxTemp : real := 0.0;
	signal Iinaux:real:=0.0;

	constant VOINIT : real := 0.0;
	constant ILINIT : real := 0.0;

	constant dtL : real := dt/L;
	constant dtC : real := dt/C;

	constant N1 : real := 1.0;
	constant N2 : real := 1.0;
	constant n : real := N1/N2;

	constant Rl : real := 0.040; -- bobina transformador 1
	constant Rt : real := 0.18;-- transistor (fijo)
	constant Rl2 : real := 0.040; -- bobina transformador 2
	constant Rd : real := 0.0383; -- diodo (fijo)
	constant Rc : real := 0.075; -- condensador
	constant Vd : real := 1.3; -- tension diodo (fijo)
	constant Rout : real := 46.08;

begin

	Vout <= voutAuxTemp;
	Iin <= iL;

	SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux, vCAux)
```

```
	begin
		if Mosfet = '1' then -- Closed switch
			vCAdd <= -Ir;
			iLAdd <= Vg - (Rl + Rt)* iL;
			Iinaux <= iL;
			voutAux <= vCAux -(Ir*Rc);
		else -- Open switch
			if iL > 0.0 then -- CCM (MODO CONTINUO)
				vCAdd <= iL/n-Ir;
				iLAdd <= -(iL/n*(Rl2 + Rd) + voutAux + Vd)/n;
				voutAux <= vCAux + ((iL/n)-Ir)*Rc;
				--voutAux<=Ir*Rout;
			else -- DCM (MODO DISCONTINUO)
				vCAdd <= -Ir;
				iLAdd<= 0.0;
				voutAux <= vCAux -(Ir*Rc);
			end if;
			Iinaux <= 0.0;
		end if;
	end process SWITCHMUX;

	DIFFEQ: process(Clk, Reset)
	-- Update of Vout and Iin each clock cycle
	-- A la izq de la igualdad es k, y a la derecha es siempre k-1
	begin
		if Reset = '1' then -- Se resetean las señales al valor inicial
			vCAux <= VOINIT;
			iL <= ILINIT;
			voutAuxTemp<=voutAuxTempInit;
		elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
			iL <= iL + iLAdd*dtL;
			vCAux <= vCAux + vCAdd*dtC;
			voutAuxTemp<=voutAux;
		end if;

	end process DIFFEQ;
	Il_debug<=iL;
	Vout_debug<=vCAux;


end Behavioral;
```

Anexo V: Código convertidor Flyback coma fija con pérdidas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.math_real.all;
use IEEE.fixed_float_types.all;
use IEEE.fixed_pkg.all;

entity ConverterFlybackRealConPerdidasNV is
	port (
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic; -- On = '1', off = '0'
		Vg : in std_logic_vector(12 downto 0);
		Ir : in std_logic_vector(12 downto 0);
		-- Out
		Iin : out std_logic_vector(12 downto 0);
		Vout : out std_logic_vector(12 downto 0);
		Vout_debug: out real;
		Il_debug: out real
	);
end ConverterFlybackRealConPerdidasNV;

architecture Behavioral of ConverterFlybackRealConPerdidasNV is

	constant dtL : sfixed(-14 downto -26) := to_sfixed(0.00005681818182,-14,-26);
	constant dtC : sfixed(-14 downto -34) := to_sfixed(0.00004545454545,-14,-34);

	signal iLAdd: sfixed(7 downto -17):= (others=>'0'); --
	signal vCAdd: sfixed(9 downto -20):= (others=>'0'); ---

	signal iL : sfixed(6 downto -19):= (others=>'0');
	signal voutAux : sfixed(7 downto -22):= to_sfixed(48,7,-22);

	signal vCAux : sfixed(7 downto -22):= (others=>'0'); --
	signal Iinaux: sfixed(6 downto -19):= (others=>'0');

	signal voutAuxTemp,voutAuxTempInit: sfixed(7 downto -22):= (others=>'0');

	constant VOINIT :  sfixed(7 downto -22):= (others=>'0');
	constant ILINIT : sfixed(6 downto -19):= (others=>'0');

	signal s_Ir:sfixed(1 downto -11):= (others=>'0');
	signal s_Vg:sfixed(7 downto -5):= (others=>'0');

	signal voutAuxAdd_aux: sfixed(7 downto -5):= (others=>'0');
	signal iLAdd_aux: sfixed(6 downto -6):= (others=>'0');

	constant Rl : sfixed(-4 downto -20) := to_sfixed(0.040,-4,-20); -- bobina transformador 1
	constant Rt : sfixed(-2 downto -18) := to_sfixed(0.18,-2,-18);-- transistor (fijo)
	constant Rl2 : sfixed(-4 downto -20) := to_sfixed(0.040,-4,-20); -- bobina transformador 2
	constant Rd : sfixed(-4 downto -21) := to_sfixed(0.0383,-4,-21); -- diodo (fijo)
	constant Rc : sfixed(-3 downto -25) := to_sfixed(0.075,-3,-25); -- condensador
	constant Vd : sfixed(1 downto -20) := to_sfixed(1.3,1,-20); -- tension diodo (fijo)
	constant Rout : sfixed(6 downto -6):= to_sfixed(46.08,6,-6);

	constant n : sfixed(1 downto -1) := to_sfixed(1,1,-1);
```

```
begin

	s_Ir <= to_sfixed(Ir,s_Ir);
	s_Vg <= to_sfixed(Vg,s_Vg);

	voutAuxAdd_aux <= resize(voutAuxTemp, voutAuxAdd_aux);
	iLAdd_aux <= resize(Iinaux,iLAdd_aux);
	-- Se debe hacer esta conversion porque Vout e Iin son std_logic_vector
	Vout <= to_slv(voutAuxAdd_aux);
	Iin <= to_slv(iLAdd_aux);

	SWITCHMUX: process(Mosfet, s_Vg, s_Ir, iL, voutAux, vCAux,iLAdd,vCAdd)
		if Mosfet = '1' then -- Closed switch
			vCAdd <= resize(-s_Ir,vCAdd);
			iLAdd <= resize(s_Vg - (Rl + Rt)* iL,iLAdd);
			Iinaux <= iL;
			voutAux <= resize(vCAux -(s_Ir*Rc),voutAux);
		else -- Open switch
			if iL > 0.0 then -- CCM (MODO CONTINUO)
				vCAdd <= (iL/n)-s_Ir;
				iLAdd <= resize(-(iL/n*(Rl2 + Rd) + voutAux + Vd)/n,iLAdd);
				voutAux <= resize(vCAux + ((iL/n)-s_Ir)*Rc,voutAux);
				--voutAux<=Ir*Rout;
			else -- DCM (MODO DISCONTINUO)
				vCAdd <= resize(-s_Ir,vCAdd);
				iLAdd<= (others=>'0');
				voutAux <= resize(vCAux -(s_Ir*Rc),voutAux);
			end if;
			Iinaux <= (others=>'0');
		end if;
	end process SWITCHMUX;


	DIFFEQ: process(Clk, Reset)
	begin
		if Reset = '1' then -- Se resetean las señales al valor inicial
			vCAux <= VOINIT;
			iL <= ILINIT;
			voutAuxTemp<=voutAuxTempInit;
		elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
			iL <= resize(iL + iLAdd*dtL,iL);
			vCAux <= resize(vCAux + vCAdd*dtC,vCAux);
			voutAuxTemp<=voutAux;
		end if;
	end process DIFFEQ;

	Vout_debug<=to_real(vCAux);
	Il_debug<=to_real(iL);
end Behavioral;
```

Anexo VI: Código convertidor Flyback coma flotante con pérdidas

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.fixed_float_types.all;
use IEEE.fixed_pkg.all;
use IEEE.math_real.all;
Library floatfixlib;
use floatfixlib.float_pkg.all;

entity ConverterFlybackPerd_Float32 is
	port (
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic; -- On = '1', off = '0'
		Vg : in float32;
		Ir : in float32;
		-- Out
		Iin : out float32;
		Vout : out float32;
		Vout_debug:out real;
		Il_debug:out real
	);
end ConverterFlybackPerd_Float32;

architecture Behavioral of ConverterFlybackPerd_Float32 is

	constant C : float32 := to_float(0.000440);
	constant L : float32 := to_float(0.000352);
	constant dt : float32 := to_float(0.000000020);

	signal iL : float32 := to_float(0.0);
	signal voutAux,vCAux : float32 := to_float(48.0);
	signal iLAdd, vCAdd,voutAuxTemp,voutAuxTempInit : float32 := to_float(0.0);
	signal Iinaux:float32:=to_float(0.0);

	constant VOINIT : float32 := 0.0;
	constant ILINIT : float32 := 0.0;

	constant dtL : float32 := dt/L;
	constant dtC : float32 := dt/C;

	constant N1 : float32 := to_float(1.0);
	constant N2 : float32 := to_float(1.0);
	constant n : float32 := N1/N2;

	constant Rl : float32 := to_float(0.040); -- bobina transformador 1
	constant Rt : float32 := to_float(0.18);-- transistor (fijo)
	constant Rl2 : float32 := to_float(0.040); -- bobina transformador 2
	constant Rd : float32 := to_float(0.0383); -- diodo (fijo)
	constant Rc : float32 := to_float(0.075); -- condensador
	constant Vd : float32 := to_float(1.3); -- tension diodo (fijo)
	constant Rout : float32 := to_float(46.08);

begin

	Vout <= voutAuxTemp;
```

```
	Iin <= iL;

	SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux, vCAux)
	begin
		if Mosfet = '1' then -- Closed switch
			vCAdd <= -Ir;
			iLAdd <= Vg - (Rl + Rt)* iL;
			Iinaux <= iL;
			voutAux <= vCAux -(Ir*Rc);
		else -- Open switch
			if iL > to_float(0.0) then -- CCM (MODO CONTINUO)
				vCAdd <= iL/n-Ir;
				iLAdd <= -(iL/n*(Rl2 + Rd) + voutAux + Vd)/n;
				voutAux <= vCAux + ((iL/n)-Ir)*Rc;
				--voutAux<=Ir*Rout;
			else -- DCM (MODO DISCONTINUO)
				vCAdd <= -Ir;
				iLAdd<= to_float(0.0);
				voutAux <= vCAux -(Ir*Rc);
			end if;
			Iinaux <= to_float(0.0);
		end if;
	end process SWITCHMUX;


	DIFFEQ: process(Clk, Reset)
	begin
		if Reset = '1' then -- Se resetean las señales al valor inicial
			vCAux <= VOINIT;
			iL <= ILINIT;
			voutAuxTemp<=voutAuxTempInit;
		elsif rising_edge(Clk) then -- Si el ciclo de reloj está HIGH
			iL <= iL + iLAdd*dtL;
			vCAux <= vCAux + vCAdd*dtC;
			voutAuxTemp<=voutAux;
		end if;
	end process DIFFEQ;
	Il_debug<=to_real(iL);
	Vout_debug<=to_real(vCAux);


end Behavioral;
```