UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Método de selección de longitud de palabra para un modelo HIL de convertidor Flyback

Máster Universitario en Ingeniería de Telecomunicación

Autor: Peces Martín, Sandra
Tutor: Martínez García, Mª Sofía
Ponente: Ángel de Castro Martín

FECHA: Septiembre, 2021

Método de selección de longitud de palabra para un modelo HIL de convertidor Flyback

AUTOR: Sandra Peces Martín
TUTOR: Mª Sofía Martínez García
PONENTE: Ángel de Castro Martín

Dpto. Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2021

Resumen

El desarrollo de la tecnología electrónica, concretamente de los convertidores conmutados avanza rápidamente, al igual que su complejidad y la dificultad de su simulación sin conllevar grandes costes. Es por esto que son necesarias nuevas técnicas para el proceso de pruebas y depuración, tales como la técnica Hardware In The Loop, HIL, la cual permite la simulación del sistema electrónico de forma digital en tiempo real.

Basándonos en la técnica HIL, se va a llevar a cabo un modelo digital de un convertidor Flyback, a partir de su modelo matemático para, a continuación, poder ser simulado en tiempo real desde un ordenador o FPGA. Es por esto que en este Trabajo de Fin de Máster se ha realizado la digitalización de un convertidor Flyback, en primera instancia en un modelo ideal, sin pérdidas ni problemas de resolución, y en la aritmética de coma fija. A continuación, se han comparado ambos modelos a partir del error relativo absoluto medio. Para obtener la longitud de palabra de las señales del modelo en coma fija, se han divido las señales del diseño en acumulativas, no acumulativas y constantes, y calculado su tamaño en bits en formato QX.Y.

La finalidad de dicho análisis es comprobar el número de bits de resolución o bits extras, que son necesarios añadir a las señales que intervienen en el convertidor Flyback de tal manera, que se consiga una distribución de bits entre señales eficiente, alcanzando un modelo de simulación óptimo, en el que se haya llegado a un equilibrio entre una resolución de señal fiable sin que se desperdicien bits.

Finalmente, como resultados de este proyecto se observa como la cantidad de bits requeridos para la representación de cada señal, depende de si son señales de corriente o de voltaje, y que, para alcanzar un modelo optimizado, hemos de tener en consideración el reparto de bits extra en función del tipo de señal, acumulativa, no acumulativa o constante.

Abstract

The development of the technology advances rapidly, as well as its complexity and the difficulty of its simulation without entailing great costs. This is why new techniques are necessary for the testing and debugging process, such as the Hardware In The Loop, HIL technique, which allows the simulation of the electronic system digitally in real time.

Based on the HIL technique, a digital model of a Flyback converter will be carried out, starting from its mathematical model so that it can then be simulated in real time from a computer or FPGA. This is why in this Master's Final Project the digitization of a Flyback converter has been carried out, in the first instance in an ideal model, without losses or resolution problems, and in fixed point arithmetic. Next, both models have been compared based on the mean absolute relative error. To obtain the word length of the fixed point model signals, the design signals have been divided into cumulative, non-cumulative and constant, and their size in bits has been calculated in QX.Y format.

The purpose of this analysis is to check the number of resolution bits or extra bits that need to be added to the signals that intervene in the Flyback converter in such a way that an efficient bit distribution between signals is achieved, reaching a simulation model. optimal, where a balance has been struck between reliable signal resolution without wasting bits.

Finally, as results of this project, it is observed how the amount of bits required for the representation of each signal depends on whether they are current or voltage signals, and that, to achieve an optimized model, we must take into account the distribution of extra bits depending on the type of signal, cumulative, non-cumulative or constant.

Palabras clave

Tamaño de palabra, convertidor Flyback, error absoluto medio, resolución de señal, Field programmable gate arrays, coma fija.

Keywords

Word Length, Flyback converter, mean absolute error, signal resolution, Field programmable gate arrays, fixed point.

Agradecimientos

Agradecer en primer lugar a mi familia todo el apoyo que me han brindado a lo largo del Grado y ahora del Máster. A mis hermanos, que me hacen sonreír en mis peores momentos donde mi estado de ánimo se ve afectado por el estrés y el cansancio debido a las entregas de prácticas, los exámenes y el trabajo. A mi pareja, mi compañero y mi apoyo a lo largo de todos estos años. Y en especial a mis abuelos, a los presentes y a los que ya no están, que los llevaré siempre conmigo allá donde me depare la vida.

Por último, agradecer a mi tutora y a todos los profesores y personal de la Escuela Politécnica Superior de la UAM, con quiénes he tenido la oportunidad de colaborar y de los que me llevo un sinfín de lecciones.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

En el ámbito de la electrónica, los reguladores conmutados son especialmente conocidos, son un tipo de convertidor de potencia DC-DC que convierte la corriente en diferentes niveles de tensión [1]. Si a estos convertidores se le suma que cada vez es más habitual el uso de controladores digitales, es necesario la comprobación de estos sistemas constituidos por el conversor de potencia y el controlador digital, mediante realización de una serie de pruebas de depuración, cuyo costo en tiempo y en lógica es elevado [2]. Actualmente, existen numerosas técnicas de testeo como simuladores de señal mixta o simuladores VHDL-AMS, pero son procesos lentos y por ello, para la realización de simulaciones, se emplea *Hardware In the Loop (HIL)*. Esta técnica consiste en la implementación de una parte del sistema en hardware digital, en este trabajo un convertidor de potencia Flyback, para emular todo el sistema de circuito cerrado hardware. Esta emulación evita daños graves y ahorra esfuerzo durante el proceso de diseño [3], y permite simular el convertidor emulando el rendimiento del sistema final. En las pruebas HIL, las señales reales son conectadas a un sistema de simulación, de tal forma que su controlador trabaja sobre un modelo digitalizado y no sobre el producto ensamblado al mismo.

A partir del incremento del uso de *Field-Programmable Gate Array (FPGA)* en los sistemas HIL, en los que los pasos de integración pueden llegar a ser del orden de cientos de nanosegundos se pueden realizar simulaciones en tiempo real [4]. En estas simulaciones en tiempo real con FPGA, es crucial optimizar la velocidad y el área. Por un lado, las mejoras en electrónica digital y FPGAs han permitido incrementar la frecuencia de conmutación del convertidor de potencia y disminuir el paso de integración de las variables de estado, aumentando así la velocidad. Sin embargo, si el paso de integración es muy pequeño, podrían surgir problemas de resolución [5]. Por otro lado, reducir el área utilizando el número mínimo de bloques *Look Up Table* (LUT) y *Digital Signal Processing* (DSP) sin perder precisión es otro factor importante en las FPGAs [6]. Es por estas razones que la determinación de la longitud de palabra (WL) de las señales involucradas debe considerarse cuidadosamente.

Este trabajo fin de máster surge de esta necesidad, en la que probar los convertidores de potencia se está convirtiendo en un cuello de botella. Existen numerosos métodos para optimizar la WL de las señales que ayudan a reducir el tiempo y el área de cálculo manteniendo una precisión suficiente [6]. Para la síntesis de VLSI (Integración a gran escala) con análisis de precisión, los métodos pueden clasificarse aproximadamente en cuatro categorías según el método para analizar los efectos de WL finitos: análisis, simulación, análisis-síntesis y simulación-síntesis [7]. Aquellos métodos que implica simulación, tienen el inconveniente de ser métodos lentos ya que requieren mucho tiempo de procesado y simulación.

El método propuesto en este proyecto, es un método más rápido y simple, además de tener un coste más bajo, en el que solo se requiere un estudio analítico y clasificación, según el tipo de señal. En este método, no es necesaria la realización de simulaciones para cada combinación de WL y, por lo tanto, se reduce el tiempo de diseño [8]. En esta referencia, se realizó el estudio para convertidores Boost y Buck. Con este trabajo fin de máster, se

pretende realizar una nueva aplicación de este método analítico para calcular la longitud de palabra necesarias para las distintas señales que intervienen en un modelo HIL del convertidor Flyback sin desperdiciar bits [8].

Se llevará a cabo el proceso de desarrollo, simulación y emulación de técnica HIL particularizada para un convertidor Flyback y posteriormente se realizará un estudio del error obtenido según el número de bits de las señales y la aplicación del método de optimización de palabra.

Para la realización del trabajo se empleará el lenguaje de programación VHDL, la herramienta ModelSim y la herramienta Simulink de Matlab.

1.2 Objetivos

Los objetivos parciales que se desarrollarán a lo largo de este TFM se listan a continuación:

- Documentación previa al trabajo. Estudio y comprensión de los conceptos a tratar en este trabajo, tales como los convertidores de potencia, en especial el convertidor Flyback, así como las herramientas de trabajo.

- Desarrollo convertidor Flyback real sin pérdidas. Realizar el modelo de planta de un convertidor Flyback real sin pérdidas en lenguaje VHDL y llevar a cabo simulaciones, utilizando el programa ModelSim y Simulink para validar el modelo diseñado.

- Desarrollo convertidor Flyback sin pérdidas en código sintetizable. Realizar el modelo de planta de un convertidor Flyback sin pérdidas en código sintetizable y llevar a cabo simulaciones empleando el programa ModelSim.

- Identificación de los grupos de señales presentes en el modelo: Se estudiarán todas las señales presentes en el modelo y se seleccionará el grupo al que pertenecen.

- Estudio del error: Se analizará el error introducido en el modelo por cada una de las señales con respecto a su WL o el número extra de bits añadidos a cada señal (n).

- Aplicación del método de optimización global: Se aplicará el método de optimización global para el cálculo del WL de las señales del modelo del convertidor y estudio del área ocupada.

1.3 Organización de la memoria

Esta memoria consta de los siguientes capítulos:

- Capítulo 1: En el primer capítulo de este Trabajo de Fin de Máster se abordan la Motivación y los Objetivos que han conducido a la realización de este proyecto.

- Capítulo 2: Se aborda el estado del arte de la tecnología a tratar, es decir, en qué consiste un convertidor Flyback, así como su de funcionamiento y aplicaciones en el mercado actual. Así mismo se abordan los conceptos y principios de la aritmética en coma fija.

- Capítulo 3: Este capítulo trata del modelado de un convertidor Flyback sin pérdidas. En primer lugar, se obtendrá su modelo matemático para que, a partir de él, podamos implementar un código de simulación en lenguaje VHDL, al que se denominará Modelo Real. Posteriormente, se llevará a cabo cálculo de la longitud de las señales que intervienen en el circuito en el formato QX.Y, para así implementar el código de simulación en VHDL del Modelo en Coma Fija.

- Capítulo 4: A lo largo de este capítulo, se realizarán simulaciones repetitivas del Modelo en Coma Fija con variaciones en los bits extras añadidos a cada señal que interviene en el Flyback. A partir de dichas simulaciones, se calculará el error relativo existente entre el Modelo Real, el cual será nuestro modelo ideal, y el Modelo en coma fija con respecto a WL y con respecto a los bits extra (n). Así mismo, se hará el análisis de dicho error y se llevará a cabo el cálculo de un Modelo Optimizado a partir de los resultados.

- Capítulo 5: En este último capítulo, se expondrán las conclusiones obtenidas tras la realización del Trabajo.

2 Estado del arte

En este proyecto se va a trabajar con convertidores de energía, es decir, el conjunto de componentes electrónicos o sistema electrónico cuya finalidad es convertir la energía eléctrica entre dos formatos distintos. Estos equipos proporcionan fiabilidad, volumen y eficiencia, que son capaces de llevar a cabo la transformación de corriente alterna a partir de corriente continua y viceversa [9].

El criterio de clasificación más común para los convertidores es su agrupación en función del formato de la energía entrante y saliente, obteniendo cuatro grupos fundamentales:

- Convertidores AC/DC, también conocidos como rectificadores, consisten en convertidores que transforman la corriente alterna en continua, ya sea monofásica o trifásica.

- Convertidores AC/AC, se trata de convertidores que tienen la capacidad de modificar arbitrariamente la frecuencia y amplitud de la intensidad, suelen ser empleados en motores de inducción para reducir la intensidad demandada durante el arranque.

- Convertidores DC/AC, denominados a su vez inversores, pueden transformar la corriente continua en corriente alterna. Este tipo de conversores permite el control del valor eficaz de la tensión o corriente de salida, así como la frecuencia.

- Convertidores DC/DC, este tipo de convertidores modifican el valor de entrada de corriente continua en otro diferente de salida. Pueden por tanto incrementar (Boost) o reducir (Buck) su valor. Además, pueden añadir aislamiento galvánico entre entrada y salida, como es el caso del Flyback.

En este Trabajo de Fin de Máster, nos centraremos en el estudio, diseño y desarrollo de un modelo HIL para el convertidor Flyback, el cual se encuentra dentro del grupo de convertidores DC/DC.

2.1 Convertidor Flyback

Se trata de un tipo de convertidor de potencia capaz de convertir AC a DC o DC a DC, con un tipo de aislamiento denominado galvánico entre la entrada y la salida. El Flyback almacena energía a partir del flujo de corriente que circula por su circuito y que posteriormente libera dicha energía, permitiendo así alcanzar múltiples valores de voltaje de salida a partir de un amplio abanico de voltajes de entrada. Esto lo consigue a partir de su transformador, consistente en un inductor acoplado múltiple que permite así elevar o reducir el voltaje a la salida [10].

La topología de un convertidor Flyback es la más conocida para fuentes de alimentación aisladas. Tiene la capacidad de suministrar múltiples salidas aisladas con un único transistor y una limitada cantidad de componentes externos. Ha sido tradicionalmente la solución del

diseñador para convertidores de potencia aislados cuya salida no superara los 100 W. Esta solución solo necesita un componente magnético y un rectificador de salida, por lo que tiene un bajo coste y una gran simplicidad.

Sin embargo, el problema de este tipo de convertidor es la necesidad de componentes de gran tamaño, como es el caso del condensador de salida, debido al estrés por corrientes altas en el transistor, y en la salida del diodo, un núcleo del transformador grande, o problemas con interferencias electromagnéticas.

La estructura de un convertidor Flyback es similar a la de un convertidor Buck-Boost, la diferencia entre ellos es que el Flyback cuenta con un transformador en lugar de un inductor permitiendo así alcanzar altos ratios de conversión. Su diseño consta de un transformador, un interruptor MOSFET, un diodo, un condensador y una resistencia a la salida.

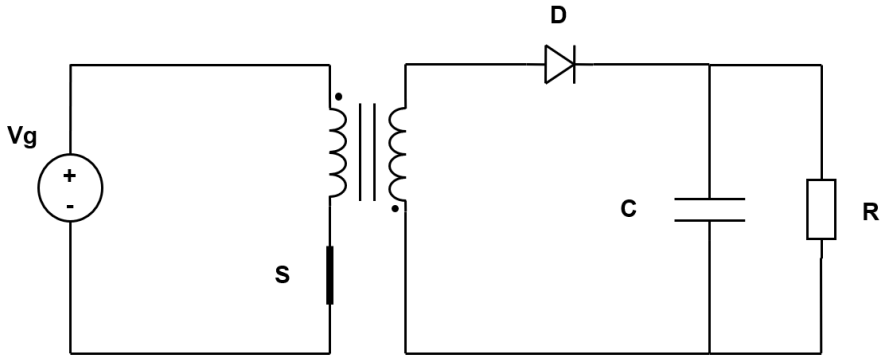

Figura 2-1: Topología Convertidor Flyback

Este convertidor, funciona de un modo distinto al resto de convertidores de potencia, donde la corriente primaria y la secundaria fluyen al mismo tiempo, y solo una pequeña parte de esta energía es almacenada en el transformador (corriente de magnetización). Un transformador flyback se parece más a múltiples inductores en el mismo núcleo.

A continuación, vamos a entrar en más detalle acerca del modo de funcionamiento del convertidor Flyback, en el cual, en función del estado en el que se encuentre MOSFET, representado como el interruptor S, es decir, activo o inactivo, el circuito actuará en consecuencia dando lugar a tres posibles estados [10]:

1. El interruptor está activo.
2. El interruptor está inactivo, siguiendo el Modo de Conducción Continua (CCM).
3. El interruptor está inactivo, dando lugar al Modo de conducción Discontinua (DCM).

2.1.1 Modo de funcionamiento con el interruptor activo

Durante el intervalo de tiempo en el que S está activo o en modo ON, la corriente en la malla del transformador primario circula provocando el almacenamiento de energía en la primera inductancia del transformador.

Esta energía es generada a partir del incremento lineal de la corriente producida por el voltaje aplicado por el generador Vg. La energía, no es transferida a la malla del transformador secundario provocando que el diodo D esté polarizado en inversa.
Por la carga de salida R, únicamente circula la corriente residual generada a partir de la tensión almacenada previamente sobre el condensador [10].

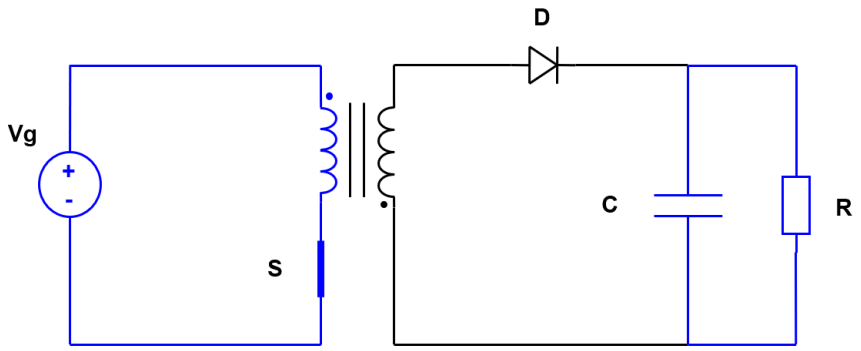

Figura 2-2: Convertidor Flyback, S = ON

2.1.2 Modo de Conducción Continua (CCM)

Cuando el interruptor se encuentra inactivo o modo OFF, en función del estado del diodo, se da lugar al Modo de Conducción Continua (CCM) o el Modo de conducción Discontinua (DCM)

Durante el Modo de Conducción Continua, la energía deja de almacenarse sobre la inductancia primaria y es transferida a la inductancia secundaria del transformador, el diodo se encuentra polarizado en directa y permite el traspaso de corriente y su llegada al condensador y a la resistencia de carga. El condensador comienza a almacenar la energía a partir de dicha corriente produciéndose así la transferencia de potencia de un lado a otro del transformador.

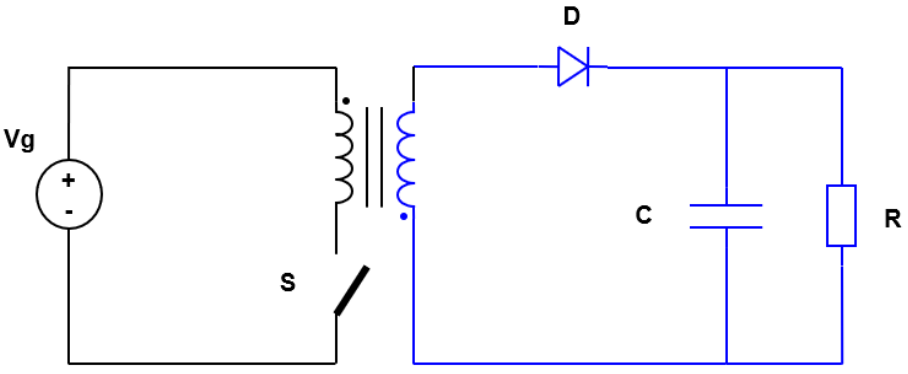

Figura 2-3: Convertidor Flyback, CCM

2.1.3 Modo de conducción Discontinua (DCM)

La segunda casuística tiene lugar cuando el interruptor está en modo OFF y se da el modo de conducción discontinua. Sigue sin producirse almacenamiento de energía en el inductor primario. La corriente en la inductancia secundaria no es suficiente para que el diodo conduzca en directa por lo que no permite el traspaso de corriente y el condensador deja de almacenar energía. Deja de circular corriente por la malla de la inductancia secundaria, dejando únicamente el paso de corriente por la malla de carga y la energía almacenada en el condensador es liberada sobre la resistencia de salida [10].

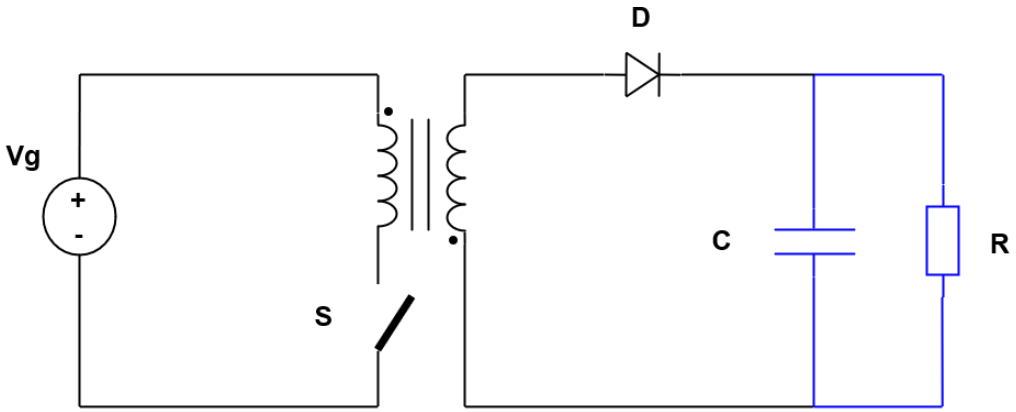

Figura 2-4: Convertidor Flyback, DCM

2.2 Aplicaciones de un convertidor Flyback

Una vez conocido su modo de funcionamiento vamos a abordar los usos de este tipo de convertidor de potencia. Desde su inicio, el convertidor Flyback fue empleado en aplicaciones cuyas señales de voltaje eran de alta frecuencia, y han ido evolucionando hasta el día de hoy, en donde su uso más conocido es en conversiones DC y DC y AC-DC [11].

Podemos encontrar este tipo de transformadores en los ámbitos que se listan a continuación:

- Dispositivos en el ámbito de la telecomunicación, en el que se los sistemas dependen en gran medida de los transformadores Flyback para usos energéticos y conversión de suministro eléctrico.

- Equipamiento en el ámbito de la aeronáutica: estos equipos deben funcionar a una frecuencia muy precisa y estos transformadores se utilizan para obtener resultados sólidos y consistentes para lograr un alto voltaje de fuente eléctrica.

- También podemos encontrar este tipo de convertidores en equipos industriales como motores, generadores o bombas eléctricas, que necesitan tales transformadores especializados para un mejor rendimiento.

- Convertidores e inversores eléctricos que funcionan a frecuencias muy fluctuantes. Este convertidor ayuda a cambiar inmediatamente la corriente de bajo voltaje a alto voltaje. También proporcionan un suministro constante de electricidad como en

fuentes de alimentación, electrodomésticos como frigoríficos, equipos como ordenadores o incluso sistemas de climatización como es el caso del aire acondicionado. Su uso también es frecuente en teléfonos móviles y en sus cargadores.

- Aunque ya han quedado prácticamente obsoletos, estos convertidores han sido utilizados en tubos CRT, los tubos eran empleados en las televisiones que se veían afectadas por interrupciones de energía del transformador de retorno. Era necesario, por tanto, un voltaje de suministro constante y fiable y los transformadores Flyback eran excelentes para conseguirlo. Hoy en día siguen siendo empleados en televisores que requieren un alto suministro de voltaje, láseres, linternas, entre otros [11].

2.3 Aritmética de coma fija

En este Trabajo de Fin de Máster, se va a emplear la notación QX.Y, basada en la aritmética de coma fija o *fixed point*, en la que el número de bits para las partes enteras y fraccionaria permanecen fijos, al valor asignado por el diseñador. A partir de este tipo de notación se representará el tamaño total en bits que son necesarios para almacenar una señal de nuestro modelo de simulación Flyback. Está notación está compuesta por una parte entera de bits 'X' y una parte fraccionaria de bits 'Y', a los cuales, se les sumará 1 bit adicional de signo de la notación complemento a 2. Es por tanto que la WL de cada señal, tendrá un total de bits X+Y+1.

Se hace uso de este tipo de aritmética debido a que con un diseño adecuado ocupa menor área y el paso de integración es menor que si se usa coma flotante. Además, gracias a los pequeños pasos de integración obtenidos, es posible la realización de emulaciones hardware en tiempo real y no sólo de simulaciones software, las cuales han demostrado ser más lentas que la emulación y que, por tanto, consiguen pasos de tiempos mucho más grandes. Queda de la mano del diseñador el cálculo del tamaño óptimo de las WL, encontrando un equilibrio con el error relativo obtenido, entre los valores exactos que se quieren representar y los valores que se es capaz de conseguir y el área ocupada [6].

Cuando se quiere representar una señal con X bits enteros e Y bits fraccionarios. Si el número de bits de la parte decimal no es muy elevado, se puede obtener un elevado error relativo. Por ejemplo, si los valores máximo y mínimo de una señal son 5.7 y 0.8 respectivamente, se puede representar con un formato Q3.2. Entonces, al usar este formato, 5.7 es equivalente a 0101.11 y 0.8 es igual a 0000.11. Redondeando al valor más cercano, el formato Q3.2 representa 5.75 y 0.75, respectivamente. Si calculamos el error absoluto, para ambos casos es 0.05. Sin embargo, los errores relativos son 0,877% y 6,25%, respectivamente. Es por esto que, cuando se han de representar señales con valores pequeños y se requiere de una alta precisión, se sumará a la parte fraccionaria (Ym) un número 'n' extra de bits. Siendo la nueva parte fraccionaria: Y = Ym + n. Más adelante se demostrará que el cálculo de bits fraccionarios no es siempre trivial cuando el valor mínimo de la señal es muy próximo a cero.

Más adelante, se aplicarán las reglas de la suma y del producto de esta aritmética, para la obtención de determinados tamaños de señales en notación QX.Y, a continuación, abordaremos estas operaciones:

Regla de la suma/resta: en esta operación, tanto para la parte fraccional como para la parte entera hay que escoger el máximo valor de ambos sumandos, además, a esta última se le suma un bit para evitar desbordamientos:

$$QX_1.Y_1 \pm QX_2.Y_2 = QX.Y \quad (1)$$

Donde X es el mayor valor de la parte entera + 1 bit.
Y es el mayor valor parte fraccional.

Regla del producto: en esta operación, la parte fraccional y la parte entera son el resultado de la suma de cada una de las partes respectivamente, sumando adicionalmente un bit a la parte entera para evitar desbordamientos:

$$QX_1.Y_1 * QX_2.Y_2 = QX.Y \quad (2)$$

Donde $X = X_1 + X_2 + 1$ bit.
$Y = Y_1 + Y_2$

Mediante esta aritmética, se llevarán a cabo las operaciones de suma y producto

2.4 Software

2.4.1 Matlab

Matlab es una potente herramienta especializada en la realización de cálculos numéricos con vectores y matrices, así como cadenas de caracteres, escalares y otras estructuras de mayor complejidad [12]. Matlab, se corresponde con la abreviatura de *Matriz Laboratory*, el cual consiste en un lenguaje de alto rendimiento para cálculos técnicos, por lo que Matlab es un entorno y a su vez, un lenguaje de programación. Matlab, a parte del cálculo matricial y álgebra lineal, también puede manejar polinomios, funciones, ecuaciones diferenciales ordinarias, gráficos, entre otros muchos. Combina entornos de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y *arrays* directamente.

Mediante esta herramienta, se pueden crear funciones y programas propios conocidos como archivos M, en código Matlab, y pueden ser agregadas en *Toolbox* o librerías, que son colecciones especializadas de M-archivos para trabajar diferentes problemáticas.

Este programa, está equipado con aplicaciones interactivas, que permiten al usuario obtener una perspectiva del funcionamiento de distintos algoritmos con sus datos. Una de sus principales características es si capacidad de escalación, el cual permite el análisis de datos desde la ejecución en *clusters*, GPUs y nubes únicamente con cambios menores en el código.

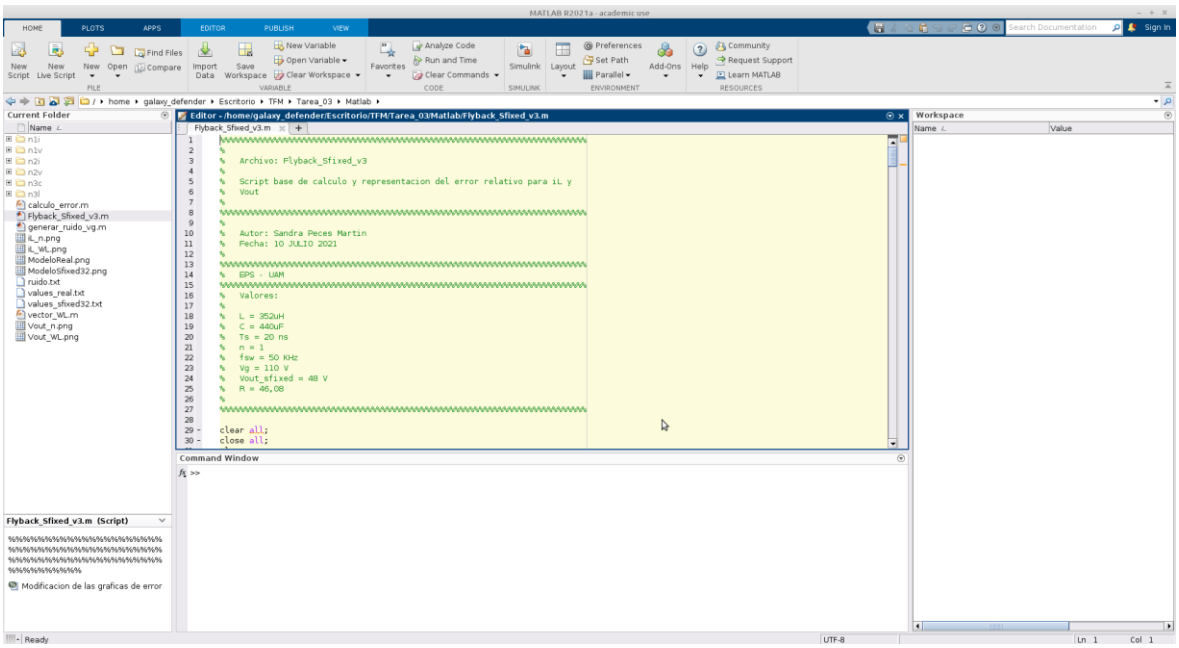

Figura 2-5: Interfaz de Matlab

Entre sus múltiples prestaciones destacan las siguientes:

- Cálculo en la nube.
- Despliegue en escritorio y web.
- Cálculo paralelo
- Uso de MATLAB con otros lenguajes
- Desarrollo de algoritmos
- Creación de apps
- Hardware
- Análisis de datos
- Gráficas

En este Trabajo de Fin de Máster, Matlab ha sido empleado para el cálculo y representación en gráficas de los errores a partir de los datos obtenidos en las simulaciones.

2.4.2 Simulink

Simulink, es una herramienta integrada con el lenguaje de programación Matlab. Así mismo, se encuentra en el entorno de programación Matlab, y emplea los algoritmos de resolución de Matlab, así como su potencia de procesamiento de datos [13]. Simulink consiste en un entorno de programación visual, con aplicaciones especialmente en electrónica debido a que permite llevar a cabo simulaciones sencillas y precisas. En él, se pueden encontrar toda clase de dispositivos eléctricos, desde cargas a bobinas, pasando por fuentes de alimentación, medidores, amplificadores, generadores de señal entro otros muchos.

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.

A lo largo de este Trabajo de Fin de Máster, esta herramienta será utilizada para verificar la correcta implementación del código en lenguaje VHDL de un convertidor Flyback mediante su simulación y realización de pruebas en este entorno.

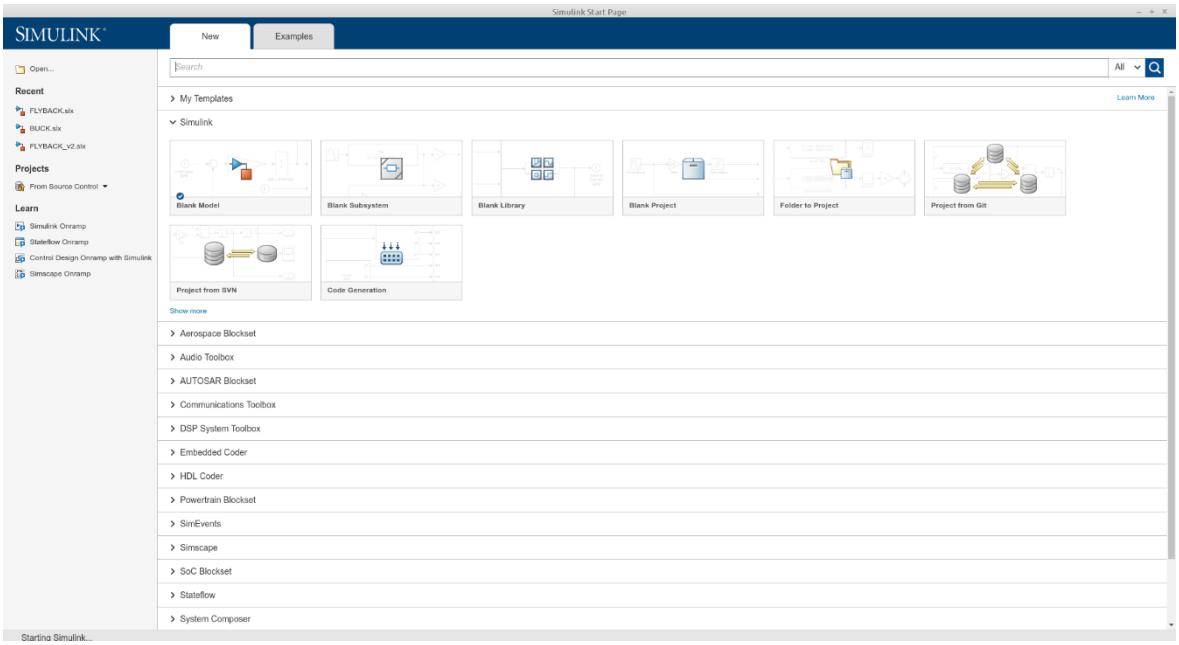

Figura 2-6: Interfaz de Simulink

2.4.3 Xilinx Vivado

Vivado Design Suite, es un paquete de software producido por Xilinx para la síntesis y análisis de diseños HDL, reemplazando a Xilinx ISE con características adicionales para el desarrollo de sistemas en un chip y síntesis de alto nivel. Vivado representa una reescritura y un replanteamiento de todo el flujo de diseño (en comparación con ISE).

Consiste en una herramienta que introduce síntesis de alto nivel, con una cadena de herramientas que convierte el código C en lógica programable.

Este programa ha sido empleado en este Trabajo para calcular el área en FFs, DSPs y LUTs de funcionamiento que abarcaría nuestro modelo en una FPGA.

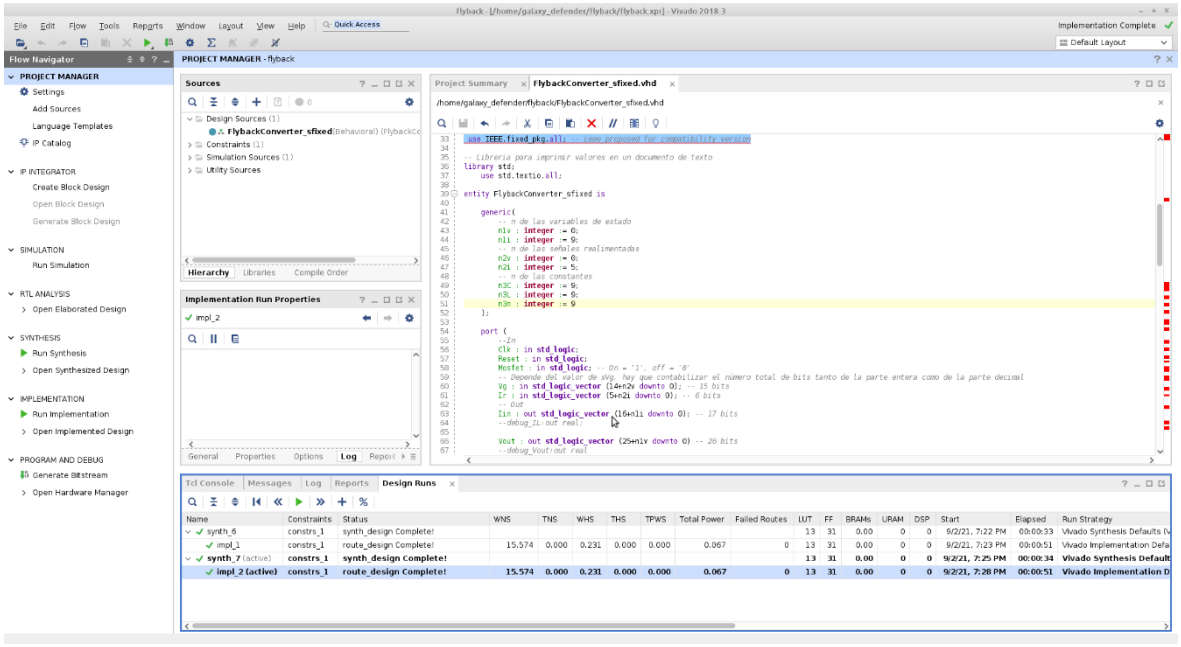

Figura 2-7: Interfaz de Vivado

3 Modelado de un Convertidor Flyback

A lo largo de esta sección, se va a abordar la implementación de un convertidor Flyback. Como ya se ha comentado con anterioridad, este tipo de convertidor de potencia permite obtener a su salida valores de tensión de mayores o menores que en su entrada, además, cuenta con un transformador permitiendo así el aislamiento galvánico entre dicha entrada y su salida. Los valores de diseño que han sido considerados a lo largo de este trabajo para este convertidor Flyback son:

Tabla 3-1: Parámetros Convertidor Flyback

Parámetros	Valor
Bobina, L	352 µH
Condensador, C	440 µF
Resistencia, R	46,08 Ω
Relación de transformación, n	1
Voltaje de entrada, Vg	110 V
Voltaje de salida, Vout	48 V
Frecuencia de connmutación, fsw	50 kHz
Ciclo de reloj, T	20 ns
Intervalo de tiempo de simulación, Δt	20 ns

Para el modelaje de este convertidor, se llevará a cabo su desarrollo mediante fórmulas matemáticas, su posterior implementación en lenguaje de VHDL, la validación del modelo y su simulación mediante la herramienta Modelsim.

3.1 Modelo Matemático

El primer paso será la obtención de las ecuaciones matemáticas para el convertidor Flyback, de tal forma que se pueda ser implementado sobre código VHDL y ser simulado mediante la herramienta ModelSim.

Como se ha comentado anteriormente, el Flyback pasa por el modo DCM y CCM en función del estado del diodo. Durante este último, el modo de conducción discontinua, el duty cycle o ciclo de trabajo el convertidor, cambiará tanto con el voltaje de entrada como con la corriente de carga de salida. Sin embargo, en modo de conducción continua, el duty cycle puede ser considerado como una constante, será casi el mismo a un voltaje de entrada dado, es decir, cambiará con el voltaje de entrada y será casi constante con la carga [14]. Entre el cambio de modo discontinuo a modo continuo, existe un límite o punto de transición en el que el ciclo de trabajo dependerá del voltaje de entrada y el voltaje de salida:

$$Duty\ cycle\ (D) = \frac{Vout}{Vout + Vg \cdot n} \tag{3}$$

Dado que se conocen los valores de las tensiones de entrada (110 V) y de salida (48 V) así como el factor de transformación n (1) podemos calcular el valor del *duty cycle* que es 30,4%. A partir de él, se pueden calcular los intervalos de tiempo de encendido y apagado del interruptor, es decir, el del MOSFET, lo cual es necesario para poder generar la señal del MOSFET para la simulación:

$$Ton = \frac{D}{Fsw} \tag{4}$$

$$Toff = \frac{1}{Fsw} - Ton \tag{5}$$

Como resultado, obtenemos el tiempo de apertura, Ton, del MOSFET es de 6,08 μs y el tiempo de apagado, Toff, es de 13,92 μs.

Para la simulación, también es necesario conocer el valor de la corriente que atraviesa el condensador C, y el voltaje que se acumula/descarga en las bobinas del transformador, para ello, se establecen las siguientes relaciones:

$$Vl = L \cdot iL \cdot \frac{\Delta t}{L} \tag{6}$$

$$iC = C \cdot vout \cdot \frac{\Delta t}{C} \tag{7}$$

Se llevará a cabo de la monitorización de la corriente, iL, que atraviesa la bobina L y la tensión del condensador C, también considerada la tensión de salida Vout del transformador. Estas se denominarán de ahora en adelante como variables de estado. Por cada paso de simulación, el valor de estas variables no es constante y depende de su estado anterior t, por lo que son definidas como:

$$iL\,(t) = iL(t - 1) + Vl(t) \cdot \frac{\Delta t}{L} \tag{8}$$

$$vout\,(t) = vout(t - 1) + iC(t) \cdot \frac{\Delta t}{C} \tag{9}$$

El comportamiento del circuito dependerá del estado del MOSFET y del diodo, por lo que para la simulación se van a considerar las diferentes casuísticas y obtener las ecuaciones que definen el comportamiento del circuito para cada instante, considerando además el aislamiento entre la entrada y la salida.

Cuando el MOSFET está activo o en modo ON, se produce el almacenamiento de energía en la primera inductancia del transformador. La energía es generada a partir del incremento lineal de la corriente producida por el voltaje aplicado por el generador Vg, y no es transferida a la segunda malla del circuito, por lo que, el incremento de la tensión del condensador depende de la corriente de la carga iR:

$$iL\ auxiliar\ (\Delta iL) = Vg \cdot \frac{\Delta t}{L} \tag{10}$$

$$Vout\ auxiliar\ (\Delta Vout) = -iR \cdot \frac{\Delta t}{C} \tag{11}$$

Cuando el MOSFET se encuentra inactivo o en modo OFF, el comportamiento del circuito depende del diodo, dando lugar a dos casos:

Si el diodo conduce, la energía almacenada en el transformador es liberada al resto del circuito, las ecuaciones de estado de la bobina y el condensador son las siguientes:

$$iL\ auxiliar\ (\Delta \mathrm{iL}) = -V\mathrm{out} \cdot \mathrm{n} \cdot \frac{\Delta \mathrm{t}}{L} \tag{12}$$

$$Vout\ auxiliar\ (\Delta \mathrm{Vout}) = \left(\frac{\mathrm{iL}}{n} - iR\right) \cdot \frac{\Delta \mathrm{t}}{C} \tag{13}$$

Si la corriente que atraviesa el diodo no es suficiente y provoca que el diodo se encuentre polarizado en inversa, las ecuaciones de las variables de estado son:

$$iL\ auxiliar\ (\Delta \mathrm{iL}) = 0 \tag{14}$$

$$Vout\ auxiliar\ (\Delta \mathrm{Vout}) = -iR \cdot \frac{\Delta \mathrm{t}}{C} \tag{15}$$

3.2 Implementación del Modelo Real

Una vez obtenidas las ecuaciones anteriores, el siguiente paso es la implementación del código VHDL. Como primera instancia, el código es elaborado con valores constantes reales, en el que no se contempla el tamaño en bits que puedan llegar a abarcar las señales. A partir de la simulación de este modelo, al que denominaremos el modelo "Real", se obtendrán los valores máximos y mínimos que se necesitarán más adelante para el modelo en sfixed, en el que las señales serán representadas en coma fija. El modelo real en VHDL se encuentra en el *Anexo A - Código VHDL de un Convertidor Flyback, modelo Real*

Para la implementación del código se ha empleado la herramienta ModelSim-Intel® FPGA, [15] la cual consiste en un entorno multilingüe de Mentor Graphics, empleado para la simulación de lenguajes de descripción de hardware como VHDL, Verilog y SystemC, e incluye un depurador C integrado. Modelsim, realiza la compilación del código y permite su posterior simulación, a continuación, se muestra una captura de la herramienta, en la que se muestran las señales obtenidas durante un tiempo de simulación de 90 ms.

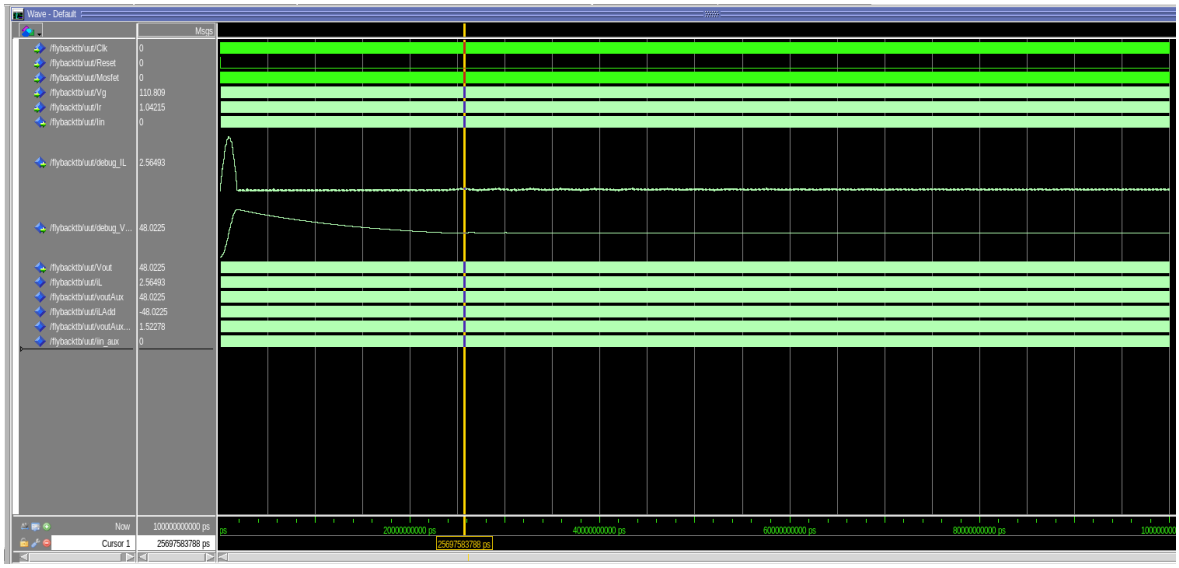

Figura 3-1: Captura simulación de un Flyback, Modelo Real

Se puede considerar que la simulación es correcta, ya como se verá en la sección siguiente coinciden con los resultados obtenidos a partir de Simulink, un entorno de programación visual, que funciona sobre el entorno de programación Matlab.

A partir de la simulación anterior, se han obtenido los valores máximos de las señales en régimen transitorio y los valores mínimos de las mismas en régimen permanente. Esto es necesario para poder llevar a cabo el cálculo de la notación QX.Y de las señales y obtener así el número de bits requeridos para poder representarlas mediante librerías *sfixed* en lenguaje VHDL, implementando así el modelo del convertidor Flyback en coma fija.

Tabla 3-2: Valores máximos y mínimos de las señales

Señal	Valor máx. transitorio	Valor mín. estacionario
Vg	110	110
Ir	2,0395	1,03752
Iin *	54,3131	0
Vout	94,41	47,5955
iL	53,2015	0,270299
vL (iL auxiliar)	110	-48,1046
iC (Vout auxiliar)	52,5873	-1,03919

(*) La señal Iin, es una señal auxiliar implementada en el código que representa la corriente de la malla del transformador primario.

3.2.1 Implementación en Simulink

Es necesario validar si el código implementado en VHDL del Modelo Real. Para ello, se va a emplear la herramienta Simulink, en la cual se ha diseñado un circuito convertidor Flyback con los mismos elementos que en la Figura 2-1, y cuyos valores son iguales a los de la Tabla 3-1. Este convertidor es sin pérdidas con los elementos propios de la herramienta. Tal y como se muestra a continuación, el circuito está integrado por los siguientes componentes:

- Fuente de alimentación (Vg)
- Mosfet o interruptor (S)
- Transformador lineal inductivo de 2 bobinas (T)
- Diodo (D)
- Condensador (C)
- Carga o resistencia (R)

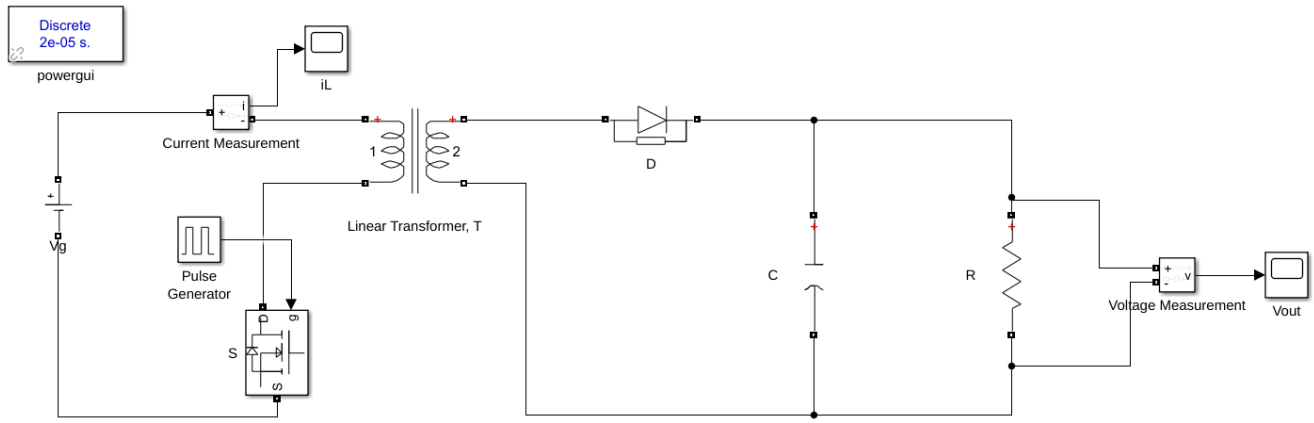

Figura 3-2: Circuito Convertidor Flyback, Simulink

Para poder realizar la simulación, además de los anteriores componentes, también son necesarios sensores de corriente o amperímetros, *Current Measurement*; sensores de tensión o voltímetros, *Voltage Measurement*; un *Scope*, opción necesaria para la representación de gráfica dentro del programa Simulink; y por último, un generador de pulso, *Pulse Generator*, que introduce al circuito una señal modulada con la configuración que el usuario considere para su diseño, en nuestro caso, introduciremos en el circuito una señal modulada de amplitud 1, 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. Otro elemento importante es el *Powergui*, en el cual seleccionamos en qué tiempo queremos simular y el modo de simulación, se ha configurado para realizar una simulación en tiempo discreto en modo Forward Euler con un tiempo de muestreo de 20µs e inicializando las señales a cero.

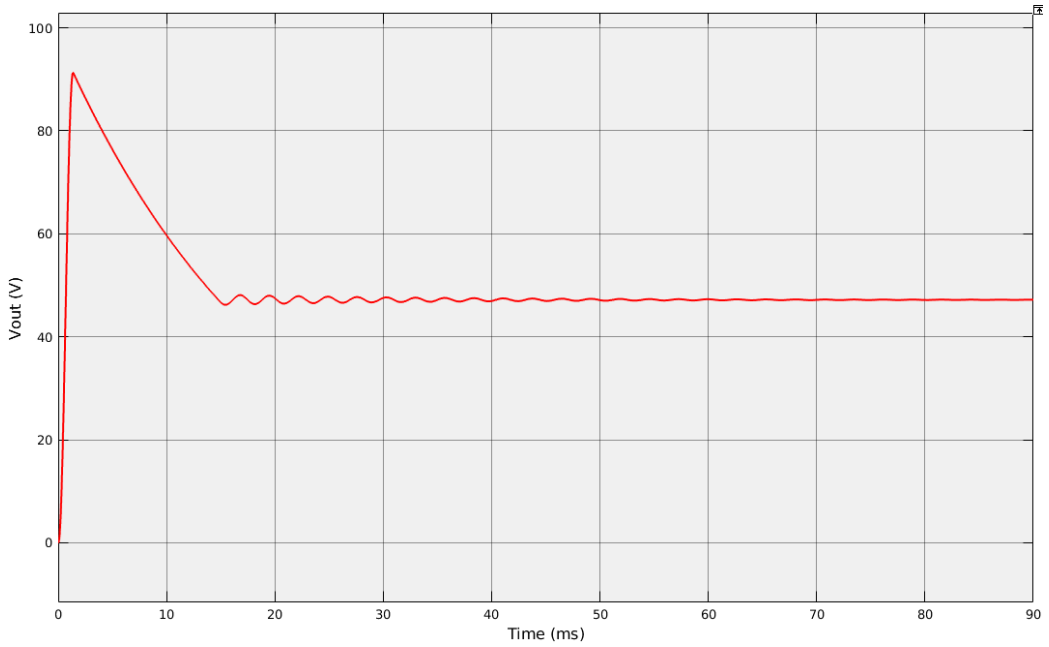

Figura 3-3: Simulación Vout, Simulink

Tal y como se puede observar en la figura anterior, en la simulación de Simulink, se han alcanzado en régimen transitorio los 92 V y los 47,5872 V en régimen estacionario. Son valores muy similares a los obtenidos con el Modelo Real en la herramienta Modelsim, en donde Vout alcanzaba un valor de 93,47 V en transitorio y 47,94 V en estacionario.

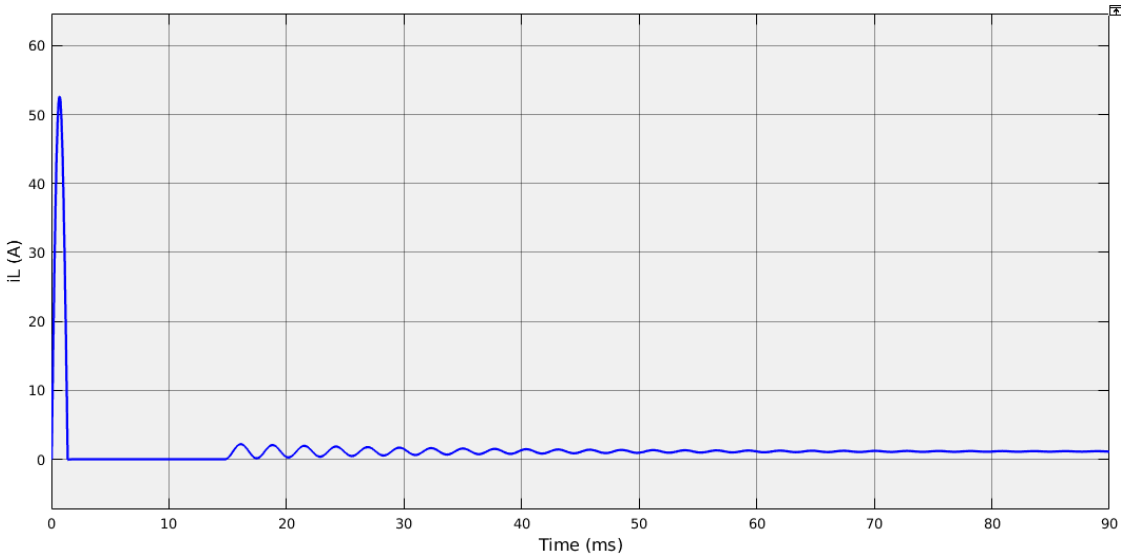

Figura 3-4: Simulación iL, Simulink

Lo mismo ocurre para la corriente iL, en Simulink, se ha obtenido un valor de corriente de 52.1 A en régimen transitorio y de 2,09 A en régimen estacionario. Valores muy próximos al Modelo Real de Modelsim, en donde iL alcanza un valor de 53,2 A en transitorio y 2,39 A en estacionario. Podemos concluir, por tanto, que la digitalización del convertidor Flyback sin pérdidas es correcta, y puede ser empleado más adelante como modelo de referencia.

3.3 Implementación del modelo en coma fija

Para poder representar las señales en notación QX.Y, necesitamos conocer el tamaño total en bits de las señales, es decir, su *Word Lenght* (WL). Existen varios métodos de cálculo de WL, algunos ya se han introducido anteriormente, pero pocos estudios han aplicado métodos analíticos puros a la WL completa, tanto en partes enteras como fraccionarias. No fue hasta 2004, cuando se presentó otro método analítico puro: un tratamiento unificado del análisis de ancho de bits para diferentes representaciones de números, tanto en coma fija como en coma flotante. La agrupación de señales es un punto importante en los estudios para minimizar el tiempo y el área de computación.

En este trabajo se aplica el método propuesto en la referencia [8], el cual es un método más simple, rápido y de bajo coste, en el que solo un estudio analítico y de clasificación, según el tipo de señal, es suficiente. No se requieren simulaciones para cada combinación de WL y, por lo tanto, se reduce el tiempo de diseño. Este método es válido en aplicaciones HIL para emulación de convertidores de potencia, resultando en una disminución de los recursos de hardware para un error específico. Esta emulación se realiza optimizando la WL completa y diciéndole al usuario qué señales necesitan más o menos bits, sin desperdiciarlos.

A continuación, se muestra el esquemático de un convertidor Flyback sin pérdidas. A partir de esta representación se pueden reconocer más fácilmente las señales que intervienen en el diseño y la relación que existe entre ellas, así como la influencia de sus componentes, multiplicadores, sumadores y multiplicadores en dichas señales.

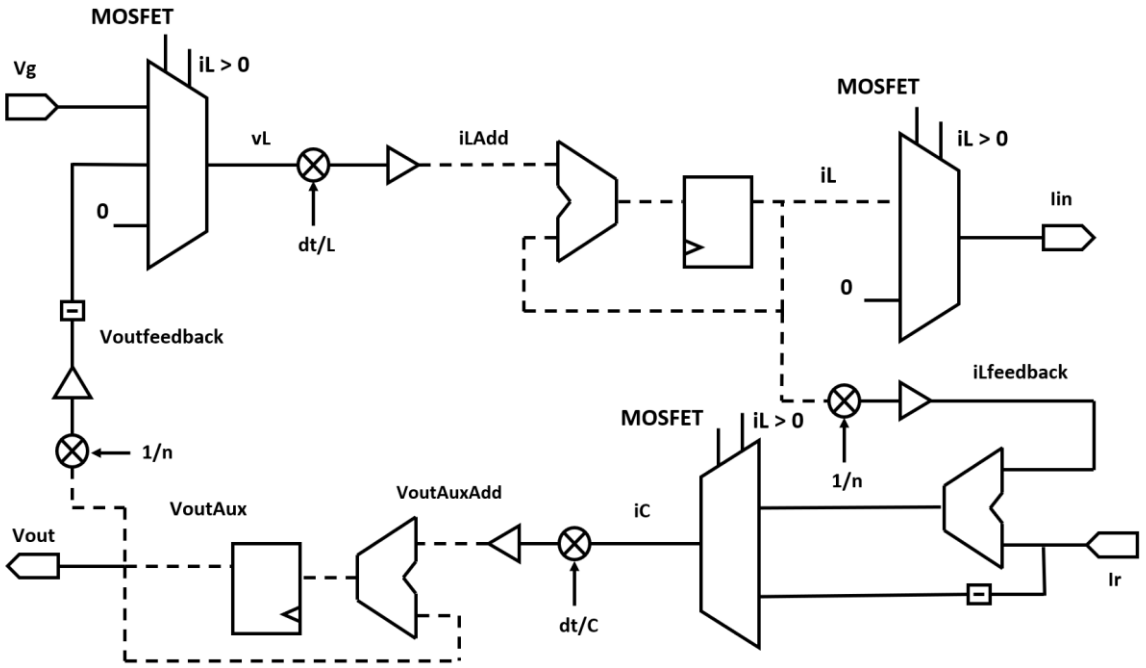

Figura 3-5: Esquema del circuito Flyback implementado.

El siguiente paso es la agrupación de las señales en función de si son acumulativas, es decir, si dependen de sí mismas en un instante anterior, las constantes cuyo valor no sufre variaciones a lo largo de la simulación, y en no acumulativas, en las que se incluyen aquellas

señales que no comparten las características de las anteriores[8][16]. La agrupación de señales es un punto importante, ya que el cálculo de la WL será distinto para cada grupo.

3.3.1 Agrupación de señales

3.3.1.1 Señales Constantes

Es caso de las señales las constantes Δt/L, Δt/C y la relación de transformación n [8]:

$$\frac{\Delta t}{C} = \frac{20\text{ns}}{440\mu F} = 45{,}4546\ [\frac{\mu s}{F}] \tag{16}$$

$$\frac{\Delta t}{L} = \frac{20\text{ns}}{352\mu H} = 56{,}8182\ [\frac{\mu s}{H}] \tag{17}$$

Se puede observar que las señales $\frac{\Delta t}{C}$ y $\frac{\Delta t}{L}$ presentan un valor realmente pequeño, por lo tanto, el número de bits para la parte entera (X) es - 14.

$$log2\left(\frac{\Delta t}{C}\right) = log2\left(45\ [\frac{\mu s}{F}]\right) = -14 \tag{18}$$

$$log2\left(\frac{\Delta t}{L}\right) = log2\left(56\ [\frac{\mu s}{H}]\right) = -14 \tag{19}$$

Esto significa que este número es tan pequeño que 14 bits de la parte entera (X) se da a la parte fraccionaria (Ym). Esta transferencia está representada por un signo negativo en notación QX.Y. Por otro lado, el número mínimo de bits para la parte fraccionaria (Ym) es 15.

Para las constantes, el valor máximo y mínimo es siempre el mismo, X = (-Ym) + 1, y, por lo tanto, ambas constantes podrían representarse con un solo bit para ajustar el valor más el bit de signo, pero en ambos casos, el error relativo es alto, por lo que se agregan dos números adicionales de bits (n3L y n3C) a las partes fraccionarias, para evitar el error y aumentar la resolución.

Notación: **Q -14.(15 + n3C) y Q -14.(15 + n3L)**

Para el caso de n, como tan solo vale 1, vamos a considerar una notación: **Q 1.(0 + n3n)**

3.3.1.2 Señales Acumulativas

Estas señales se corresponden con aquellas cuyo valor depende de ella misma en un instante anterior. Este es el caso de las variables de estado, iL y Vout, así como las señales

correspondientes con su incremento en el tiempo ΔiL o iL auxiliar e ΔVout o Vout Auxiliar, estas señales no son consideradas como acumulativas, pero son incluidas en este grupo porque son necesarias para el cálculo de iL y Vout. Dado que su valor puede variar realmente poco de un instante a otro de la simulación, se necesita una alta resolución para su representación. Por lo tanto, para el calcular su WL, se tomará para la parte entera el valor máximo de cada señal en régimen transitorio más un bit de seguridad, y el valor mínimo de su señal incremental Δx en régimen permanente. Adicionalmente, se deben agregar bits en la parte fraccionaria tal y como se ha mencionado con anterioridad para tener suficiente resolución.

Señal iL:

- Parte entera (X):

$$log2(iLmax) = log2(53{,}2015) = 5{,}73 \approx 6\ bits + 1\ bit \tag{20}$$

- Parte fraccional (Y):

$$\Delta \mathrm{vL} = |vLmin| \cdot \frac{\Delta \mathrm{t}}{L} = |-48{,}1046| \cdot (56{,}8182 \cdot e - 6) = 0{,}0027331 \tag{21}$$

$$-log2(\Delta \mathrm{vL}) = log2(0{,}0027331) = 8{,}5152 \approx 9\ bits + n1i \tag{22}$$

- Notación: **Q 7.(9+n1i)**

Señal iLAdd:

Esta señal se corresponde con el incremento temporal de la señal iL.

- Parte entera (X): depende del producto de las señales vL y $\frac{\Delta \mathrm{t}}{L}$, que, de acuerdo con la aritmética en coma fija, donde la parte entera es la suma de los factores más 1 bit de seguridad. La notación QX.Y de la señal vL es calculada en el siguiente apartado con las señales no acumulativas y tiene un tamaño de 8 bits, la parte entera de la señal dt/L es de -14 bits, por lo tanto, la parte entera de esta señal auxiliar es de -5 bits.

- Parte fraccional (Y): es la misma que la parte fraccional de iL, 9 + n1i.

- Notación: **Q -5.(9 + n1i)**

Señal Vout:

- Parte entera (X):

$$log2(Vout) = log2(94{,}41) = 6{,}54 \approx 7\ bits + 1\ bit \tag{23}$$

- Parte fraccional (Y): la parte fraccional de la señal Vout depende de la señal incremental de iC, es decir, depende de ΔiC. Es necesario comprobar si el periodo de dicha señal en régimen permanente alcanza valores negativos, es decir, si pasa por cero y la señal no es

cuadrada [17]. En la siguiente figura, se representa un periodo de ΔvL, las unidades del eje x están en ps y en el eje y en V.

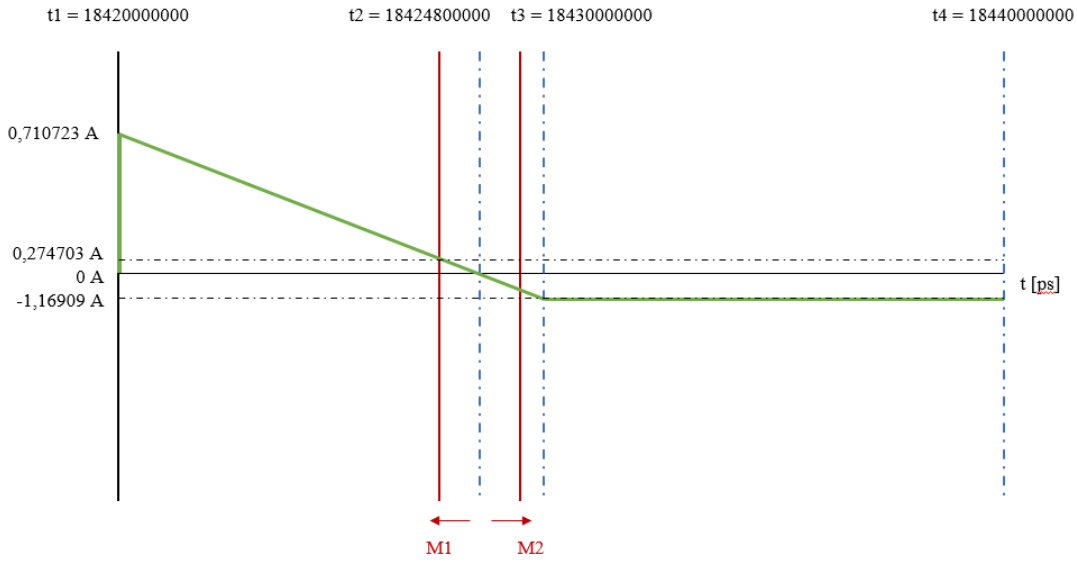

Figura 3-6: Periodo de la señal iC

En la figura, se observan en color rojo los marcadores M1 y M2, que se encuentran situados a ± 2.5% de t2. Para obtener el número de bits de la parte fraccional, nos quedaremos con el menor de ellos en valor absoluto.

- Un periodo completo de la señal en régimen permanente es de: t4 - t1 = 20000000 ps
- El 5% de dicho periodo es 1000000 ps y el 2.5% de 500000 ps.
- Por tanto:
  - M1 = t2 – t2.5% = 1842300000 ps, el cual se corresponde con 0,274703 A
  - M2 = t2 + t2.5% = 1843300000 ps, el cual se corresponde con -1,16909 A

$$\Delta iC = |iCmin| \cdot \frac{\Delta t}{C} = |0{,}274703| \cdot (45{,}4546 \cdot e - 6) = 12{,}4865\ \mu s/F \tag{25}$$

$$-log2(\Delta iC) = log2(12{,}4865\ \mu) = 16{,}289 \approx 17\ bits + n1i \tag{26}$$

- Notación: **Q 8.(17 + n1v)**

Señal VoutAuxAdd:

Esta señal se corresponde con el incremento temporal de la señal Vout.

- Parte entera (X): depende del producto de las señales iC y $\frac{\Delta t}{C}$, acorde con la aritmética en coma fija, donde la parte entera es la suma de los factores más 1 bit de seguridad. La notación QX.Y de la señal iC se calcula en el siguiente apartado y se corresponde con 7 bits, la parte entera de la señal dt/C es de -14 bits, por lo tanto, la parte entera de esta señal auxiliar es de -6 bits.

- Parte fraccional (Y): es la misma que la parte fraccional de Vout, 17 + n1i.

- Notación: **Q -6.(17 + n1v)**

3.3.1.3 Señales no Acumulativas

Las señales no acumulativas no dependen de sí mismas en un estado anterior, su principal diferencia respecto las señales acumulativas es la obtención de los bits de la parte fraccional, que mientras que estas últimas dependen de una señal auxiliar incremental, en este caso únicamente es necesario utilizar su valor mínimo en régimen permanente. Para la parte entera se emplea su valor máximo en régimen transitorio.

Estas señales se corresponden con las salidas, señales de realimentación, entradas y salidas de un multiplexor. Otra diferencia respecto a las señales acumulativas es que sus errores no son constantes, pueden tener diferente signo, y llegar a compensarse, por lo que pertenecen a un grupo diferente. Hay que tener presente que las señales del multiplexor, tanto de entrada como de salida deben compartir el mismo número de bits fraccionarios, por lo que están alineadas.

Señal vL:

- Parte entera (X):

$$log2(\Delta\text{iLmax}) = log2(110) = 6{,}78 \approx 7\ bits + 1\ bit \tag{27}$$

- Parte fraccional (Y):

$$log2(\Delta\text{iLmin}) = log2(|-48{,}1046|) = 5{,}58 \approx 6\ bits + n2v \tag{28}$$

- Notación: **Q 8.(6 + n2v)**

Señal Vg:

- Parte entera (X):

$$log2(\text{Vg}) = log2(110) = 6{,}78 \approx 7\ bits + 1\ bit \tag{29}$$

- Parte fraccional (Y): al ser una de las entradas del multiplexor 1, ha de compartir el mismo número de bits fraccionales que vL: 6 + n2v.

- Notación: **Q 8.(6 + n2v)**

Señal iIn:

- Parte entera (X):

$$log2(\text{Iinmax}) = log2(54{,}3131) = 5{,}76 \approx 6\ bits + 1\ bit \tag{30}$$

- Parte fraccional (Y): al ser salida del multiplexor 2, debe tener la misma parte fraccional que iL: 9 + n1i.

- Notación: **Q 7.(9 + n1i)**

Señal iC:

- Parte entera (X):

$$log2(\Delta\text{Voutmax}) = log2(52{,}5873) = 5{,}71 \approx 6\ bits + 1\ bit \tag{31}$$

- Parte fraccional (Y): se calcula a partir del valor obtenido en el apartado anterior, en el cálculo de la señal de estado Vout, al descartar el 5% del valor de tiempo.

$$log2(\text{iCmin}) = log2(0{,}274703) = 1{,}86 \approx 2\ bits + n2i \tag{32}$$

- Notación: **Q 7.(2 + n2i)**

Señal iR:

- Parte entera (X):

$$log2(\text{iRmax}) = log2(2{,}0935) = 1{,}066 \approx 2\ bits + 1\ bit \tag{33}$$

- Parte fraccional (Y): al ser entrada del multiplexor 3, debe tener la misma parte fraccional que iC: 2 + n2i.

- Notación: **Q 3.(2 + n2i)**

Señal Voutfeedback:

- Parte entera: esta señal auxiliar es el resultado del producto en coma fija debido al multiplicador que hay entre la señal 1/n, con parte entera de 1 bit, y la señal Vout, cuya parte entera es de 8 bits. Por tanto, el resultado es de 10 bits.

- Parte fraccional (Y): al ser entrada del multiplexor 1, debe tener la misma parte fraccional que vL: 6 + n2v.

- Notación: **Q 10.(6 + n2v)**

Señal iLfeedback:

- Parte entera: esta señal auxiliar es el resultado del producto en coma fija debido al multiplicador que hay entre la señal 1/n, con parte entera de 1 bit, y la señal iL, cuya parte entera es de 7 bits. Por tanto, el resultado es de 9 bits.

- Parte fraccional (Y): al ser entrada del multiplexor 3, debe tener la misma parte fraccional que iC: 2 + n2i.

- Notación: **Q 9.(2 + n2i)**

A modo de conclusión y como resultado de la agrupación de señales, obtenemos los siguientes valores para cada señal. Así mismo, en el esquemático se muestra la clasificación de las señales: en color azul se representan las señales acumulativas, en color verde las señales constantes y en color rojo las señales no acumulativas.

Tabla 3-3: Notación QX.Y de las señales

Señal	Notación QX.Y
Vg	Q 8.(6 + n2v)
Ir	Q 3.(2 + n2i)
Iin	Q 7.(9 + n1i)
Vout	Q 8.(17 + n1v)
iL	Q 7.(9+n1i)
vL (iL auxiliar)	Q 8.(6 + n2v)
iC (Vout auxiliar)	Q 7.(2 + n2i)
iLAdd	Q -5.(9 + n1i)
VoutAuxAdd	Q -6.(17 + n1v)
iLfeedback	Q 9.(2 + n2i)
Voutfeedback	Q 10.(6 + n2v)
dt/C	Q -14.(15 + n3C)
dt/L	Q -14.(15 + n3L)
n	Q 1.(0 + n3n)

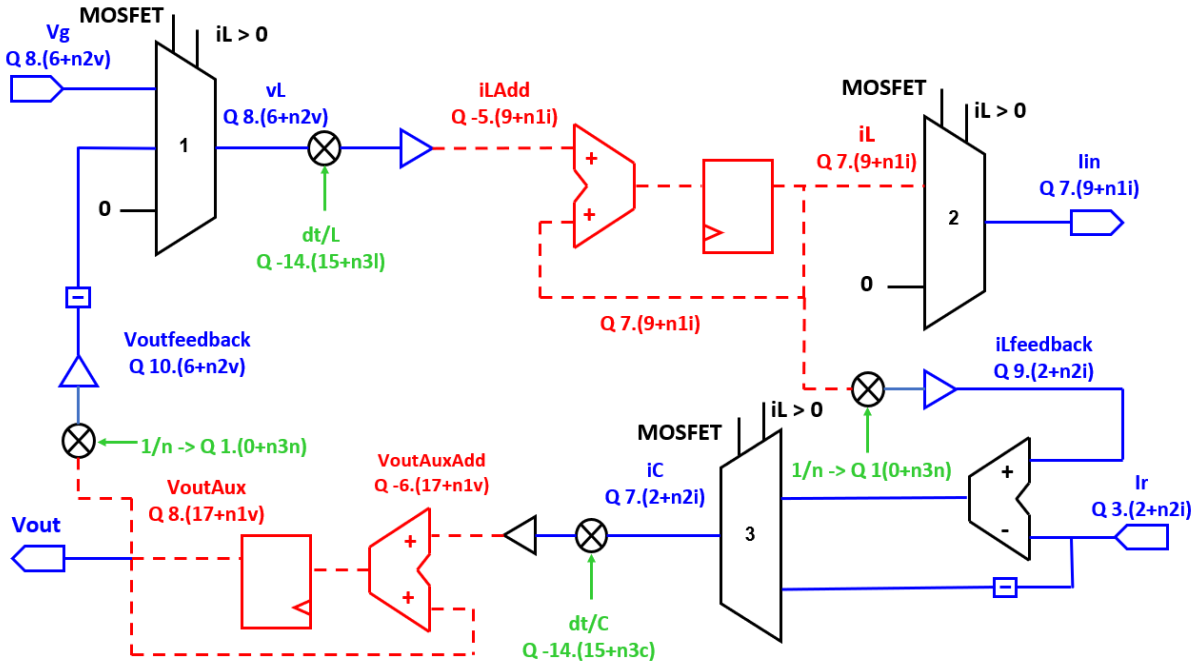

Figura 3-7: Esquema Flyback con agrupación de señales

3.3.2 Código VHDL

Una vez que se ha calculado la notación QX.Y para cada señal, se implementa el código VHDL del Modelo en Coma Fija del Flyback, el cual se adjunta en el *Anexo B - Código VHDL de un Convertidor Flyback, coma fija*. La principal diferencia de este modelo respecto al Real, es que ahora se asigna un número determinado de bits para cada una de las señales que intervienen en el convertidor. La instanciación de las señales se realiza mediante la función *sfixed* de la librería sfixed, en la cual, además de las funciones de asignación de bits, se pueden encontrar las funciones *to_sfixed* y *to_real*, las cuales permiten llevar a cabo la conversión entre la señal en formato real y una señal en formato coma fija y viceversa. En este modelo ya se tienen en consideración los bits de resolución n1v, n1i, n2v, n2i, n3c, n3L, los cuales, como primera instancia, se han igualado a 32 bits.

Así mismo, en el *testbench* del modelo en coma fija se ha insertado ruido blanco a la señal Vg para conseguir una simulación más fiel al modelo físico. Por lo tanto, a la tensión de entrada cuyo valor en el Modelo Real es estático de 110 V, para el Modelo en coma fija se ha implementado una señal Vg de 250 valores que oscila entre los 109 V y los 110 V.

A continuación, se muestran unas gráficas comparativas, entre el Modelo Real y el Modelo en Coma Fija del Flyback. Dichas gráficas representan las señales iL y Vout para una simulación de 90 ms. Estos datos han sido obtenidos gracias a la librería de texto *std.textio.all*, que permite el volcado de valores de la simulación desde el ModelSim a un fichero de texto que puede ser leído mediante la herramienta Matlab, para ser posteriormente procesados.

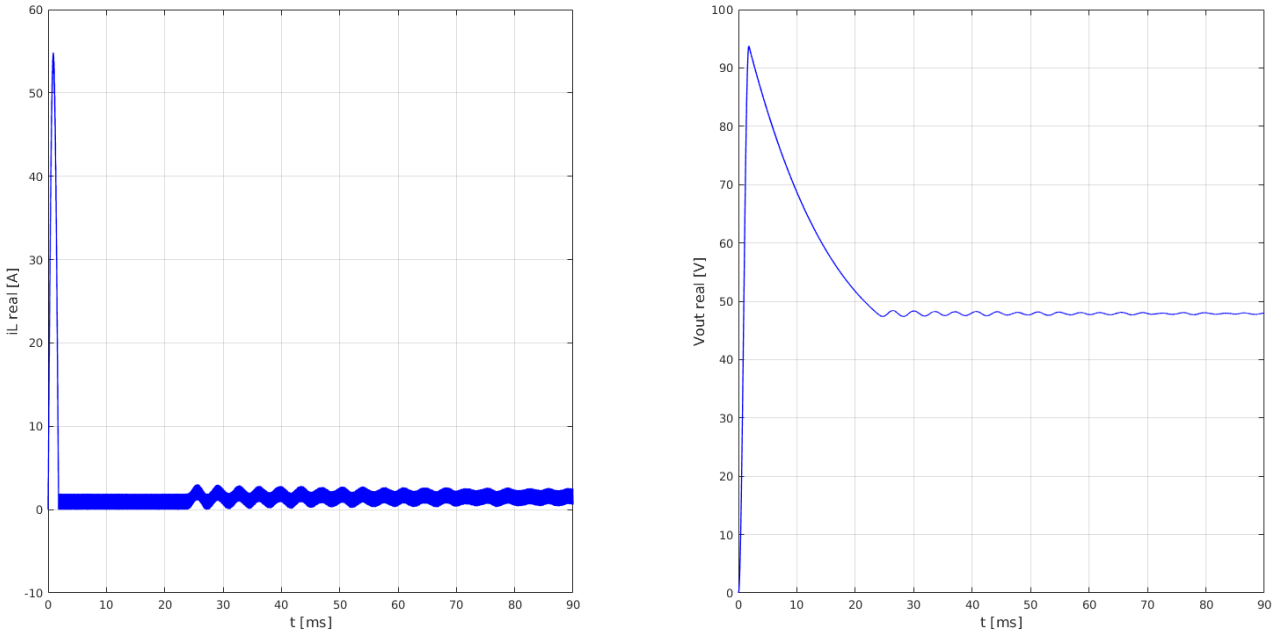

Figura 3-8: Variables iL y Vout para el Modelo Real

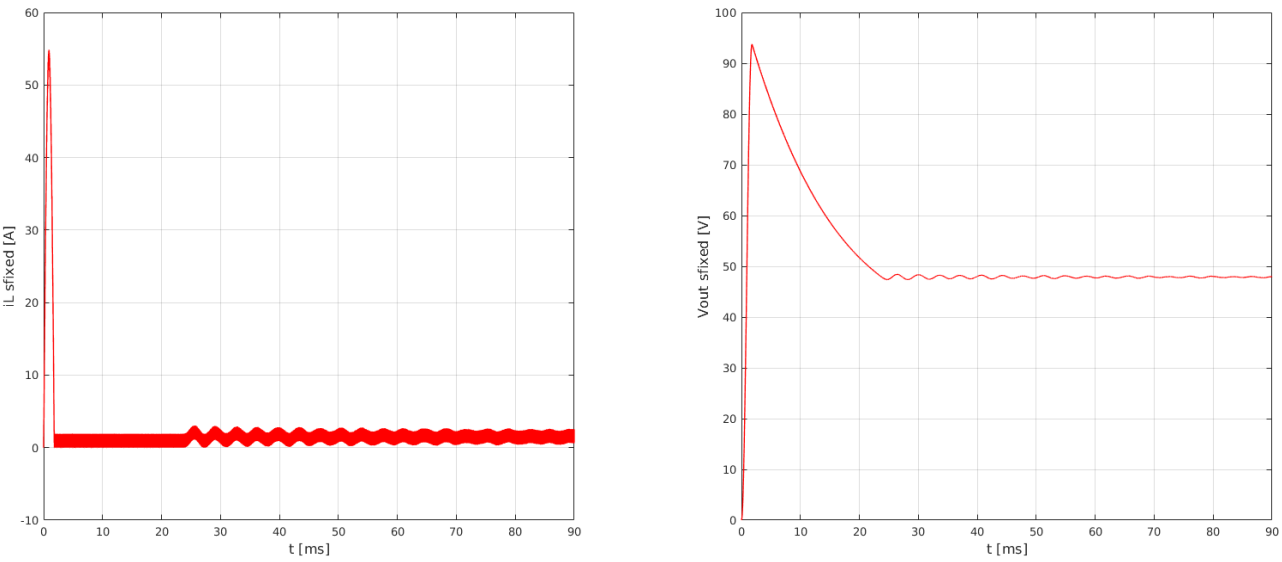

Figura 3-9: Variables iL y Vout para el Modelo en Coma Fija

En las gráficas de nuestro Modelo Real, podemos observar el comportamiento de la corriente, iL, que atraviesa el transformador del Flyback y la tensión que se obtiene en la carga del mismo, cuando las condiciones son ideales. En las gráficas restantes, se representan las mismas variables, pero para el Modelo en Coma Fija. Este es un modo de verificar la correcta implementación de este último modelo, ya que el comportamiento de ambas variables es muy similar al Modelo Real.

Para verificar la similitud entre ambos códigos se ha optado por su representación en una misma gráfica, tanto para iL como para Vout, se puede comprobar, que las gráficas se

solapan prácticamente, lo que confirma lo similares que son los resultados para ambas simulaciones.

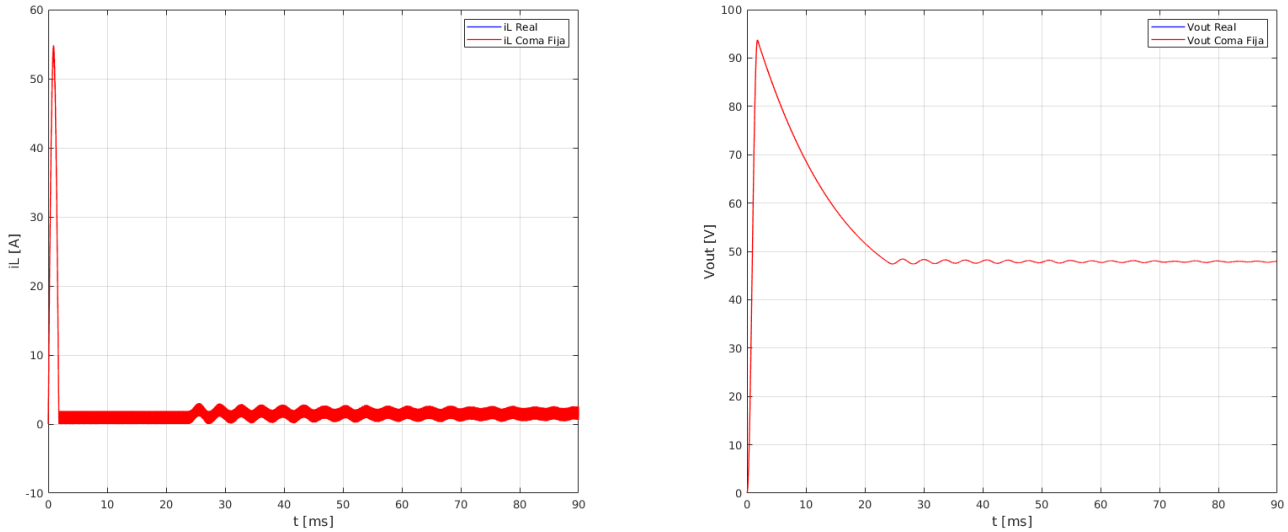

Figura 3-10: iL y Vout del Modelo Real y el Modelo en Coma Fija

4 Resultados

En este apartado, se muestra el procedimiento de validación de los conceptos y principios abordados en las secciones anteriores, que, a partir de los mismos, se obtendrá el método completo para optimizar el WL, cuya resolución se verá más adelante.

En primer lugar, se comprobará la validez de nuestro sfixed del convertidor Flyback mediante el cálculo del error entre el Modelo Real y el Modelo en Coma Fija. Posteriormente, se analizará este error con respecto a la WL de cada señal por separado, antes de proponer cómo combinar la WL de todas las señales para un resultado global óptimo. Seguido del análisis de la relación del error relativo con el valor de los bits extra, n, para cada grupo de señales y la relación con el número total de bits (X + Ym + n) por grupos.

4.1 Cálculo del error relativo

Para comprobar la validez de nuestro Modelo en Coma Fija, se realiza el cálculo del error relativo mediante el Modelo Real, un modelo que como ya se ha comentado se trata de un modelo ideal, sin pérdidas ni problemas de resolución ya que utiliza una representación de números reales para cada señal, es decir, de punto flotante de 64 bits.

La obtención del error relativo, se realiza mediante el Error Absoluto Medio, MAE, que consiste en una medida de la diferencia entre dos variables continuas, en nuestro caso, las señales de corriente del transformador, iL, y la tensión obtenida a la salida del convertidor Vout. Mediante el MAE, podemos cuantificar la precisión de nuestra técnica comparando los valores de nuestro Modelo en Coma Fija frente a nuestro Modelo Real [18].

$$MAE = \frac{\sum_{i=1}^{N} |x_i - x'_i|}{N} \tag{34}$$

Donde:

- x_i son los valores del Modelo en Coma Fija
- x'_i son los valores del Modelo en Real
- N es el número total de muestras

Los valores obtenidos de MAE se dividen posteriormente por el valor típico de las señales iL y Vout, para obtener finalmente el error relativo. Estos valores típicos se corresponden con el valor que alcanzan dichas señales en su régimen permanente, con un valor de 48 V para Vout y de 1.15 A para iL, este último valor se ha obtenido calculando el valor medio de su valor máximo y mínimo de su régimen permanente, ya que no llega a alcanzar un valor constante, sino que oscila entre 0.3 A y 2 A, por lo que finalmente se ha decidido que su valor típico sea de 1.15 A.

$$MRE = \frac{MAE}{Valor\ típico} \tag{35}$$

En la siguiente tabla, se muestra el MRE, entre el Modelo Real y el Modelo en Coma Fija asignando un total de 32 a los bits de resolución, n1v, n1i, n2v, n2i, n3c y n3l, para las señales acumulativas, iL y Vout.

Tabla 4-1: Error Absoluto Medio Relativo

iL	Vout
-3,4508	-5,1813

4.2 Análisis del error con respecto a n

Se procede al análisis del error relativo en función de los diferentes valores que se asignan a n. Es decir, si la WL de las señales depende de X + Ym + n, se va a realizar un barrido de los bits de resolución n de {-10 a 30} bits con un intervalo de dos en dos. Este barrido se realiza cambiando un tipo de n en cada simulación, manteniendo el resto a 32 bits. La finalidad de esto es comprobar cómo afectan estos bits extra a cada tipo de señal, las acumulativas, las no acumulativas y las constantes.

Se realiza, por tanto, un total de 21 simulaciones por cada tipo de n, n1v, n1i, n2v, n2i, n3c y n3l, para cada una de las cuales se obtiene un fichero de texto en el que se vuelcan los valores de las variables de iL y Vout. A continuación, estos archivos de texto con abiertos mediante la herramienta Matlab para procesar los datos y calcular el error relativo para cada tipo n para ambas variables. En el *Anexo D - Función de cálculo del error relativo en Matlab* se encuentra la función implementada para la realización de este cálculo.

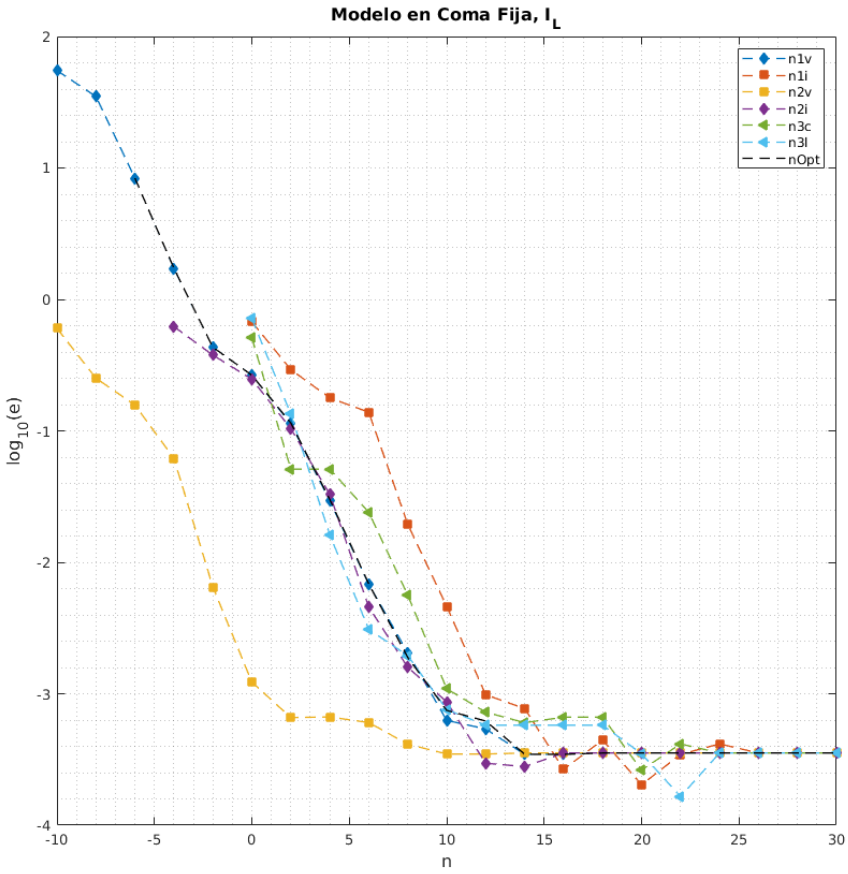

Figura 4-1: Error Relativo Promedio en función de n de iL

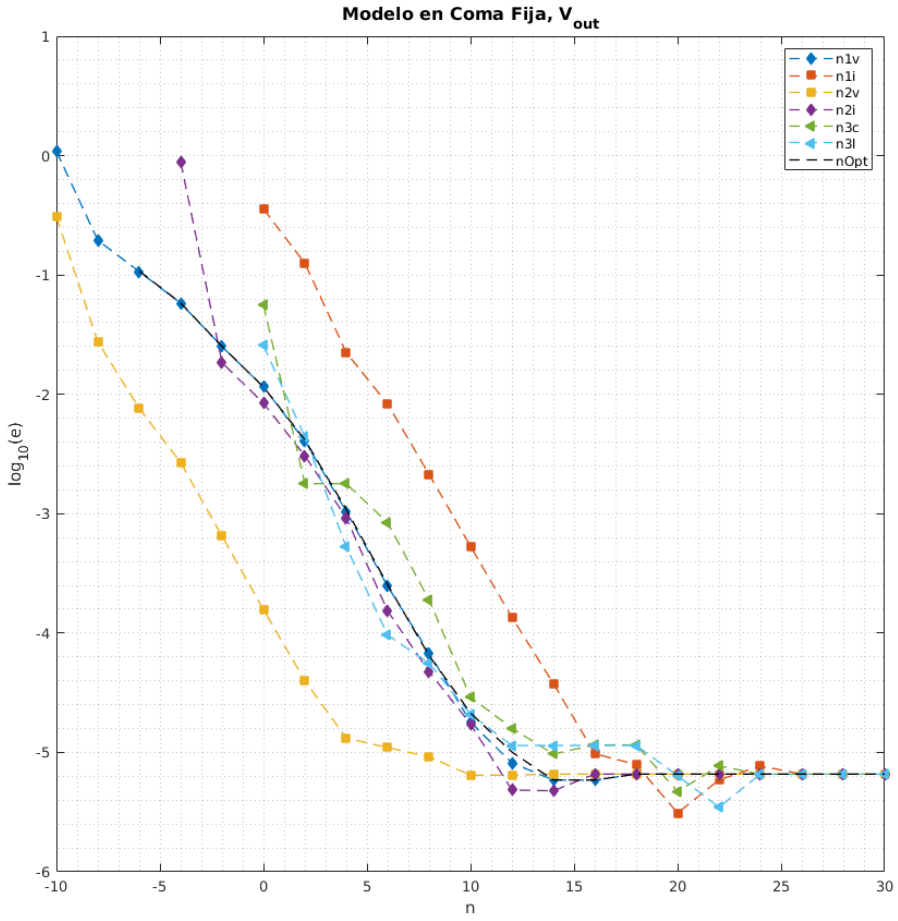

Figura 4-2: Error Relativo Promedio en función de n de Vout

Las figuras anteriores muestran los errores relativos en escala logarítmica de iL y Vout con respecto a los diferentes valores de n. De ambas gráficas se puede concluir que el error para cada tipo de n tiende a agruparse por ramas. Es decir, puede apreciarse la agrupación de los errores para n1v y n2i, que son las señales que se utilizan para calcular Vout, la rama de voltaje, que al tener una mayor proximidad a valores de n negativos se puede decir que son menos exigentes que el resto de señales en términos de bits adicionales. Otra agrupación que se puede identificar es la de las constantes, n3L y n3C, las cuales necesitan una mayor cantidad de bits de resolución en comparación al grupo anterior. Sin embargo, las señales de error para n1i y n2v, las cuales se emplean para calcular iL, la rama de corriente, no dependen tanto la una de la otra, requiriendo menos bits de resolución n2v que n1i.

En una primera aproximación, podemos concluir que algunas señales demandan más bits extra que otras. También puede deducirse que conforme aumenta el valor de las n, cuando son más positivas, el error se ve reducido y de ahí que las pendientes sean negativas. Así mismo, con la disminución de n, incluidos los valores negativos, los errores relativos aumentan.

Además, se puede observar que los errores relativos tienden a estabilizarse a partir n = 20 bits, cuyos errores son siempre menores que 10^{-3}, por lo que también se puede concluir que el error se reduce de forma proporcional al aumento de número de bits de resolución.

En primera instancia, se representó también el error relativo a la señal 1/n, cuyos bits de resolución se han denominado n3n, sin embargo, tal y como se muestra a continuación, se aprecia que el añadir o disminuir sus bits extras a esta señal no afecta al resto ya que siempre se está representando el valor 1, que es la relación de transformación, es por esto que ha sido descartada para el resto de análisis y resultados.

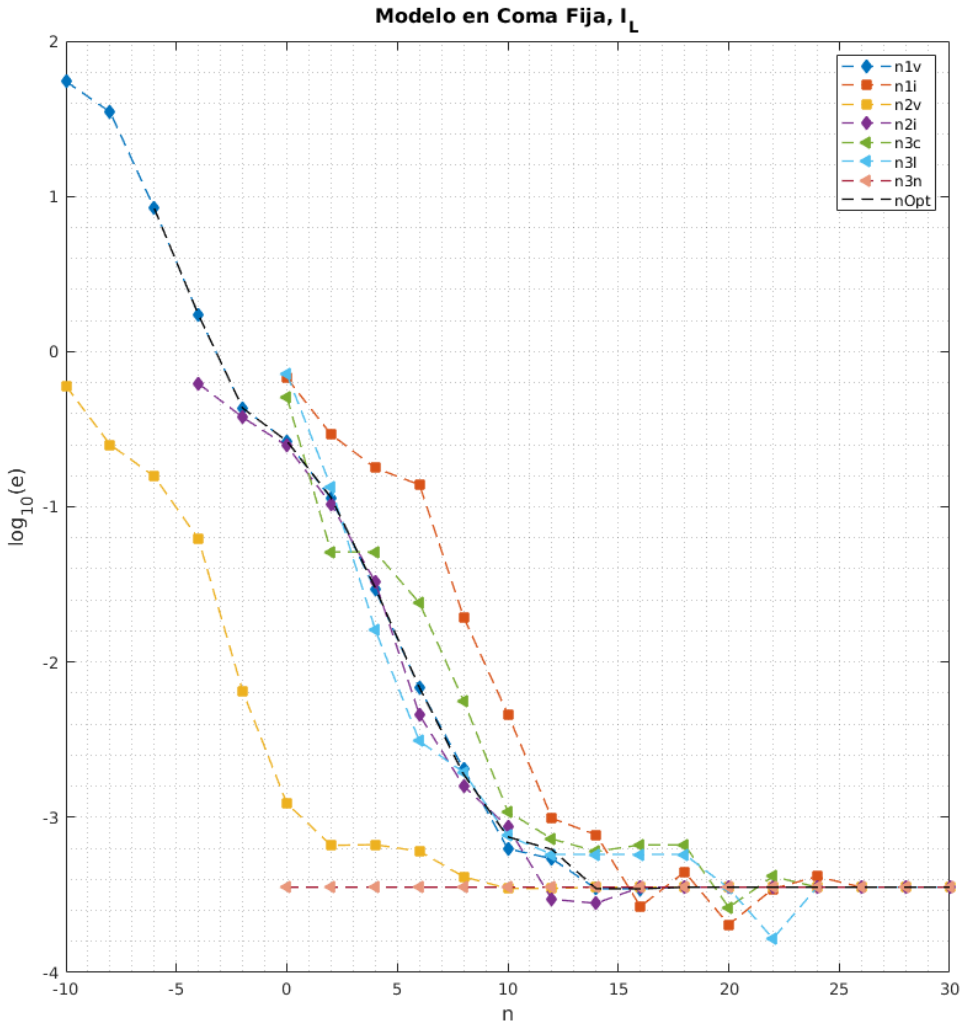

Figura 4-3: Error Relativo Promedio en función de n de iL junto con n3n

4.3 Análisis del error con respecto a la longitud de palabra

Para este caso, se va a realizar el análisis del error relativo a partir de las simulaciones obtenidas en el apartado anterior, es decir, a partir de la variación de los valores de n, pero con la diferencia de que, para la representación de las gráficas, en vez de que el eje x sean los valores de los bits de resolución n, el eje se corresponde con las diferentes WL de cada señal, X + Ym + n. Dado que conocemos los bits de parte entera y parte fraccional para cada señal porque ya han sido calculadas con anterioridad, en el cálculo de la WL se procede de la siguiente forma:

Como ejemplo, emplearemos la señal Vg. Se ha obtenido en la sección *3.3.1 Agrupación de señales*, que el formato QX.Y de esta señal es Q 8.(6 + n2v), donde los bits de la parte entera, X, son 8, y los bits de la parte fraccional, Ym, son 6, a los que hay que sumar los n2v bits extra de resolución en un barrido de {-10 a 30} bits. Desde la herramienta Matlab, se ha implementado una función de la cual se obtiene el vector de WL para cada tipo de n, esta función se adjunta en el *Anexo E - Función de cálculo de la WL en Matlab*, que consiste en un bucle que va rellenando un vector en el que, en cada posición, introduce el resultado de la suma X + Ym + n, que para este ejemplo sería 8 + 6 + (-10) = 4, 8 + 6 + (-8) = 6, 8 + 6 + (-6) = 8, etc.

Cómo resultado, el error relativo en función de la WL para cada n, es el siguiente:

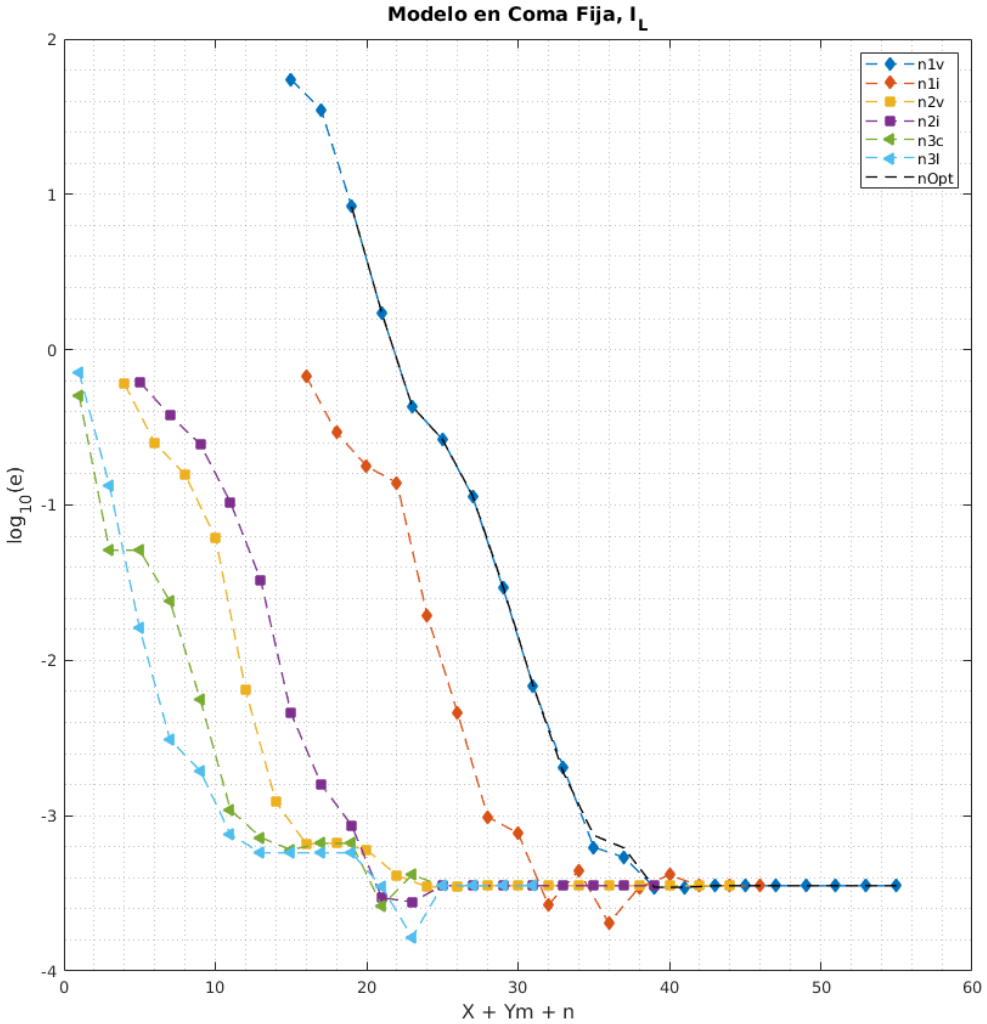

Figura 4-4: Error Relativo Promedio en función de WL de iL

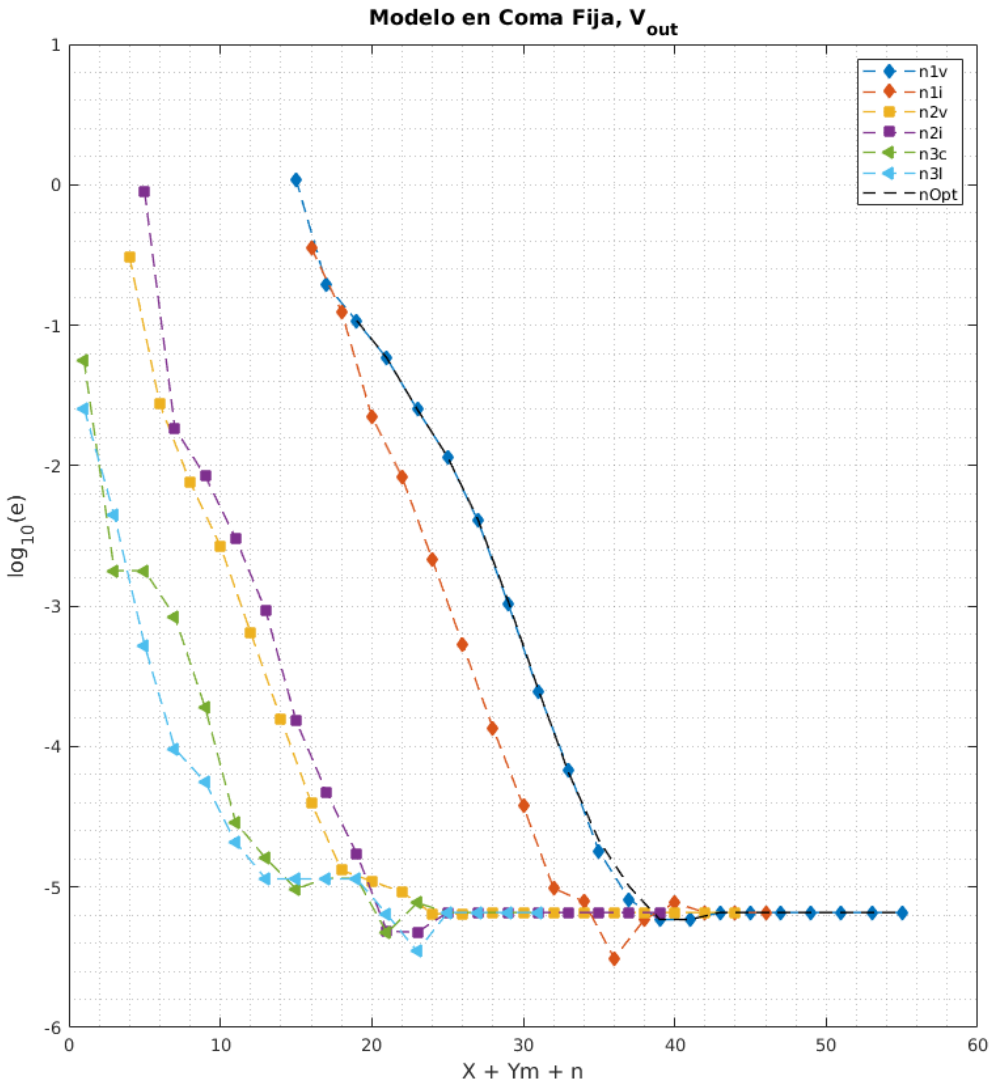

Figura 4-5: Error Relativo Promedio en función de WL de Vout

Al igual que en la representación del error en función de los bits de resolución n, se puede observar la agrupación de las señales, pero en función de agrupación en voltaje o constantes, se da a partir del tipo de señal, acumulativas, no acumulativas y constantes, es decir, el grupo de señales acumulativas con n1i y n1v; el grupo de las señales no acumulativas, con n2v y n2i; y el grupo de las constantes, con n3L y n3C, que al necesitar una WL menor, son las señales menos exigentes en términos error.

Como primera aproximación para este caso, es que la WL final de señales del mismo grupo tiende a ser el mismo para un error específico. Para cada grupo de señales, la suma del número máximo de bits para sus partes enteras X, los bits para sus partes fraccionarias Ym y los bits adicionales de resolución para alcanzar un error específico es casi la misma. Así mismo, se puede deducir no solo qué señales necesitan más bits adicionales, n, sino también cuáles necesitan una WL más larga para el mismo error relativo específico. Si todas las señales tienen el mismo WL, muchos bits en las señales menos serán desperdiciados.

Por ejemplo, para un error relativo de 10^{-4} en la figura 4-5, las variables de estado, que son las señales acumulativas, necesitan alrededor de 30 bits. Sin embargo, se ha demostrado que, para el mismo error, las señales no acumulativas necesitan 15 bits y las constantes

necesitan alrededor de 10. Es por esto que si para un tipo de señal específico como es el caso de las señales acumulativas requieren de un mayor número de bits y el resto de señales no, no será necesario utilizar el mismo número de bits en todas ellas.

4.4 Modelo optimizado

En una aplicación real, mantener todos los bits de resolución a 32 bits mientras que se va variando solo una n, no tiene sentido si lo que andamos buscando es encontrar un modelo optimizado en el que se busca conseguir el número mínimo de bits en cada señal para obtener un mismo error general. Para los análisis de error relativo que hemos realizado con anterioridad, se ha considerado una única causa de error y por lo que se desperdiciarían bits en otras señales [8]. A continuación, se va a calcular el modelo óptimo para el convertidor Flyback, el cual consiste en cuatro pasos fundamentales:

1. Determinar la X y la Ym.
2. Obtención del máximo error, y la selección en consecuencia de una n.
3. Calcular la Word Length.
4. Optimizar n (por grupos), incrementando las n de las señales del mismo grupo.

El primer paso ya se ha realizado en el apartado *3.3.1 Agrupación de señales*, en el cual se ha calculado la parte entera y fraccional de todas las señales que interfieren en el diseño.

En cuanto al segundo punto, a partir de las figuras 4-2 y 4-3, se ha podido concluir que a partir de n = 6 bits el error relativo máximo que estamos dispuestos a asumir es algo mayor del 1%. De esta forma, se garantiza un error máximo con un valor específico de n, en este caso de 6 bits. Sin embargo, para algunas señales, esto significa desperdiciar bits, porque n no está completamente optimizado. Para conocer las señales donde se utilizan bits innecesarios, el método propone analizarlas por grupos. Tal y como se puede deducir de las gráficas 4-4 y 4-5, donde las señales del mismo grupo tienen niveles de error similares para el mismo valor de X + Ym + n.

En este tercer paso, se va a llevar a cabo el análisis del error por grupos, para que todas las señales tengan la misma contribución al nivel de error. Se va a mantener el tamaño de la señal que tenga mayor WL de cada grupo, pero a su vez, se incrementará el tamaño del resto de las señales. De esta forma, se reduce el error máximo y no se desperdician bits, esto se debe porque al aumentar el número de bits el error siempre disminuye. En la siguiente tabla, se muestra el cálculo de la WL (X + Ym + n + 1) para una 6 de bits y la diferencia existente en bits del tamaño de señales pertenecientes al mismo grupo.

Tabla 4-2: Bits de las señales para n = 6

Tipo de Señal	Señales	n	X	Ym	WL	Diferencia
Acumulativas	Vout	n1v	8	17	32	9
	iL	n1i	7	9	23	
No Acumulativas	vL	n2v	8	6	21	5
	iC	n2i	7	2	16	
Constantes	dt/C	n3L	-14	15	8	0
	dt/L	n3C	-14	15	8	

Por último, se va a realizar el cálculo de la longitud óptima para cada tipo de señal: acumulativa, no acumulativa y constante. Se toma como punto de partida 6 bits de resolución, tal y como se ha comentado con anterioridad.

- Señales Acumulativas: de acuerdo a la tabla 4-1, se ha obtenido una WL de 30 bits para Vout y de 21 bits para iL. Se mantendrá la longitud de la señal con mayor WL, en este caso Vout, y se incrementa la señal iL en 9 bits, que es la diferencia que existe entre ambas:
    - n1v = 6 bits.
    - n1i = 6 + 9 bits = 15 bits

- Señales No-Acumulativas: al igual que en el caso anterior, de acuerdo a la tabla 4-1, vL tiene una WL de 13 bits e iC un tamaño de 14. La señal vL permanece igual, pero se incrementa la señal iC 5 bits:

    - n2v = 6 bits
    - n2i = 6 + 5 = 11 bits

- Señales Constantes: la diferencia entre ambas señales es de 0 bits, esto significaría que no habría que añadir bits adicionales en ninguna, sin embargo, si queremos que el error relativo sea proporcional para todas las señales, se va a añadir el mayor número de bits obtenidos como diferencia, en este caso 9 bits (entre las señales acumulativas):

    - n3L = n3C = n1i = 15 bits

Por tanto, el tamaño final para las señales con n óptima es:

Tabla 4-3: WL óptima

Tipo de Señal	Señales	WL
Acumulativas	Vout	Q 8.23
	iL	Q 7.24
No Acumulativas	vL	Q 8.12
	iC	Q 7.13
Constantes	dt/C	Q -14.30
	dt/L	Q -14.30

En las gráficas de análisis del error en función del barrido de n, también viene representado el error relativo para nOpt, este error se ha obtenido a partir de la simulación del Modelo en Coma Fija haciendo un barrido, pero esta vez, en vez de variar los bits de resolución de un solo tipo y manteniendo el resto a 32, se ha realizado el barrido modificando todos los bits a partir de los valores óptimos calculados anteriormente. Dado que esta simulación ha afectado al tamaño de todos los bits n1v, n1i, n2v, n2i, n2l, y n2c, para poder ser representada se ha tomado como referencia la señal n1v ya que es la más exigente, y que como se puede apreciar, es la que mayor error tiene.

En el caso de la representación del error para nOpt en función de WL, se ha tomado igualmente el tipo n1v, pero esta vez con un desplazamiento en el eje X, al que se le han sumado, estos bits de resolución para que nuestra simulación se óptima.

Por ejemplo, se eligió nOpt = 6 para garantizar un error menor a 10^{-1}, lo cual se logró debido a que se incrementaron las señales con menos bits, reduciendo así el error. En resumen, sin realizar simulaciones para cada WL y solo cambiando n, el método propuesto puede obtener un nivel máximo de error relativo y optimizar X + Ym + n para no desperdiciar bits, teniendo todas las señales una contribución similar a la del error global.

Las figuras que se muestran a continuación, se corresponden con las Figuras 4-1 y 4-5, pero con una ampliación sobre la rama n1v y nOpt, a partir de la cual, se puede observar que el error nOpt es levemente superior al alcanzado en el barrido de n1v.

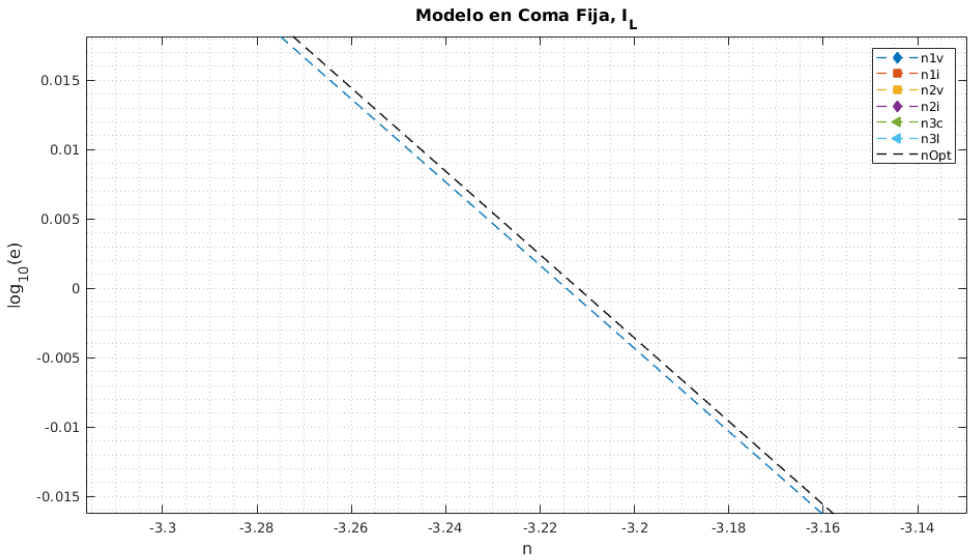

Figura 4-6: Ampliación gráfica nOpt para iL en función de n

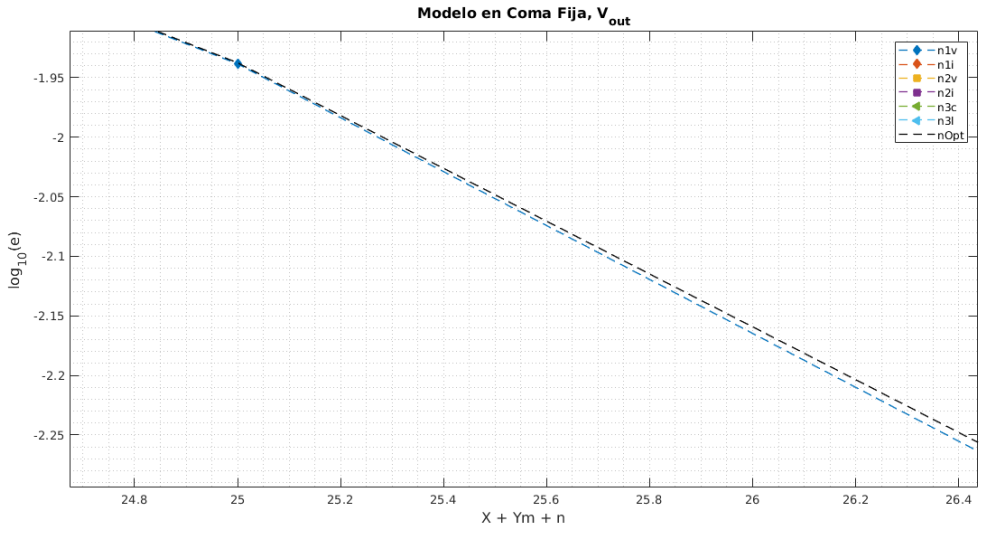

Figura 4-7: Ampliación gráfica nOpt para Vout en función de WL

Finalmente, los resultados de síntesis para ambos ejemplos de aplicación, el convertidor Flyback, con valores optimizados de nOpt es estudiada. La síntesis ha sido realizada

mediante la herramienta Vivado 2018.3, empleando para las simulaciones una FPGA Xilinx Artix 7, referencia, xc7A35-ticsg324-iL.

Tabla 4-4: Resultados de la síntesis en FPGA

n	LUTs	FFs	DSPs	$T_{clk,min}$ (ns)
0	13	31	0	4,426
4	40	37	0	7,036
6	53	39	0	8,897
8	72	48	2	11,536
20	156	72	2	12,582
32	261	91	6	15,928
0 (sin DSP)	13	31	0	4,426
4 (sin DSP)	40	37	0	7,036
6 (sin DSP)	53	39	0	8,897
8 (sin DSP)	89	48	0	10,254
20 (sin DSP)	397	72	0	15,202
32 (sin DSP)	817	81	0	19,137

Tal y como se muestra en la tabla anterior, conforme el número de bits n aumenta, el número de flip-flops lo hace también. De hecho, el número de flip flops es igual al número de bits utilizados en las variables de estado, iL y vout, que se registran. Por ejemplo, el formato QX.Y de iL es Q7.9 + n1i y Vout Q8.17 + n1v. Para el caso de nOpt = 20, n1i y n1v tienen los valores 29 y 20, respectivamente. Por tanto, el número total de bits de cada uno es 46, por lo que la suma de ambos es 90, que queda por encima de 72, los Flip Flops requeridos para este caso.

Sin embargo, para el área combinacional, el número de LUT aumenta exponencialmente. De hecho, el área combinacional incluye tanto bloques LUT como DSP integrados en la FPGA. Para facilitar la comparación, también se incluyen los resultados de síntesis sin usar bloques DSP, por lo que el área combinatoria total es solo el número de LUT. Como aumentar el número de bits adicionales tiene una influencia importante en el área total, que crece exponencialmente.

Para concluir, el área utilizada en nOpt = 6 bits es considerablemente menor que el área combinatoria necesaria para coma fija de 32 bits, así como el intervalo de tiempo mínimo, que también es menor. Por lo tanto, el método propuesto puede mantener el mismo nivel de error al tiempo que disminuye tanto el área como el tiempo.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

A lo largo de este Trabajo de Fin de Máster, se ha abordado desde el diseño de un Modelo Matemático de un convertidor tipo Flyback hasta un Modelo en Coma Fija optimizado. La optimización del modelo se ha llevado a cabo a partir de un método simple y analítico para calcular la WL necesaria (X + Ym + n) para cada señal que interviene en el modelo HIL del convertidor de potencia sin desperdiciar bits siguiendo el método propuesto en [8].

En primer lugar, se ha realizado el análisis de un convertidor de potencia tipo Flyback, desde su modo de funcionamiento hasta sus aplicaciones en la industria actual. Una vez obtenido su modelo matemático, se ha procedido a la implementación del Modelo Real del Flyback en lenguaje VHDL, este modelo es sin pérdidas y libre de errores de resolución, por lo que se ha convertido en nuestro modelo ideal, el cual se ha tomado como referencia en el cálculo del error relativo. A continuación, se ha comprobado su correcta implementación mediante la simulación de un convertidor de este tipo mediante la herramienta Simulink de Matlab, una vez demostrada que la implementación el código ha sido válida, el siguiente paso ha sido la implementación del modelo analítico.

El Modelo Coma Fija o sfixed, se ha obtenido a partir del cálculo del tamaño de las señales en notación QX.Y. Para ello, se ha llevado a cabo la agrupación previa de las señales que intervienen en el diseño en tres categorías: señales Acumulativas, No Acumulativas y Constantes. Una vez obtenida la parte entera, X, y la parte fraccional, Ym, se ha procedido al estudio del valor óptimo de bits de resolución n, también denominados bits extra, a partir del análisis del error relativo frente a distintos valores de n, y frente a la WL (X + Ym + n) para cada grupo de señales. Se ha realizado el cálculo del Error relativo, MRE, a partir del MAE, que permite la cuantificación de la precisión de nuestra técnica comparando los valores del Modelo en Coma Fija frente al Modelo Real.

El análisis del error se ha implementado en código Matlab y partir de esta herramienta se han construido gráficas comparativas de las cuales, se puede concluir que cuando las señales se representan con respecto a los bits de resolución n, los errores relativos tienen cierta tendencia a la agrupación en ramas de voltaje, o de variables constantes. Y en la representación de los errores frente a WL, se agrupan por tipo de señales: señales Acumulativas, No Acumulativas y Constantes. A partir de las simulaciones frente a n, se puede concluir que las señales menos restrictivas en cuanto a número de bits para su representación se refieren son para la rama de voltaje asociadas a n1v, y n2i.

Por otro lado, del análisis del error frente a la WL, se deduce que las señales constantes son las que menor WL necesitan para ser representadas, mientras que las señales Acumulativas son las que más requieren. Así mismo, de estas gráficas se puede deducir que la estimación de la WL para que todas las señales tengan la misma contribución al nivel de error y obtener así un Modelo Óptimo, conviene realizarla en función del tipo de señal.

Finalmente, se han obtenido las longitudes de palabra, WL, para las señales a partir del método de n óptima, el cual consiste en determinar la parte entera y fraccional de cada señal;

la decisión de qué error máximo se permitirá en el diseño y establecer un valor preliminar de n, en este trabajo se ha escogido una n= 6 bits, el propósito principal de este paso es asegurarse de que todas las señales hagan la misma contribución al error global para que no se desperdicien bits. De esta forma, se puede garantizar un nivel de error sin necesidad de largas simulaciones configurando la WL de cada señal; por último, el ajuste de las señales en función del grupo al que pertenecen.

Tanto el método analítico como las conclusiones a las que se llegaron en el artículo de la referencia [8], en cuyo caso se trabajó con un convertidor Buck y un convertidor Boost, son similares a las obtenidas en este TFM con un Flyback

5.2 Trabajo Futuro

En este Trabajo de Fin de Máster se han realizado simulaciones para un convertidor Flyback sin pérdidas, como trabajo futuro, se podría comprobar cómo se ve afectado el método de cálculo de la n óptima en el convertidor Flyback con pérdidas. Así mismo, este método podría aplicarse en otra clase de convertidores y analizar si funciona para otros casos.

Todos los scripts de código implementados a lo largo de este proyecto están disponibles en el siguiente enlace, el cual conduce a un directorio de la plataforma de almacenamiento Google Drive:
https://drive.google.com/drive/folders/1jKyLbIEx9y8RnN8gK8bGz-tyHTgdEqoP?usp=sharing

Referencias

[1] Ghosh S, Giambiasi N. Modeling and simulation of mixed-signal electronic designs - Enabling analog and discrete subsystems to be represented uniformly within a single framework. IEEE Circuits Devices Mag 2006;22(6):47–52.

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

[3] Jia J, Yang G, Nielsen AH, et al. Hardware-in-the-loop tests on distance protection considering VSC fault-ride-through control strategies. J Eng 2018;2018(15):824–9. https://doi.org/10.1049/joe.2018.0248.

[4] Typhoon HIL, "Typhoon HIL402 Brochure", Manual del Sistema HIL402

[5] Sanchez A, Castro AD, Garrido J. Parametrizable Fixed-Point Arithmetic for HIL With Small Simulation Steps. IEEE J Emerg Sel Top Power Electron 2019;7(4): 2467–75. https://doi.org/10.1109/JESTPE.2018.2886908.

[6] Sanchez A, Castro AD, Garrido J. A Comparison of Simulation and Hardware-in-the-Loop Alternatives for Digital Control of Power Converters. IEEE Trans Ind Inf 2012;8(3):491–500. https://doi.org/10.1109/TII.2012.2192281.

[7] María Murillo Moya, "Control digital de convertidores conmutados mediante LabVIEW", Trabajo Fin de Máster, Universidad Autónoma de Madrid (UAM), Escuela Politécnica Superior (EPS), 2017

[8] Martínez-García MS, Castro AD, Sanchez A, Garrido J. Word length selection method for HIL power converter models. 2020. https://doi.org/10.1016/j.ijepes.2020.106721

[9] Ned Mohan, Tore M. Undeland, William P. Robbins (2002). Power Electronics: Converters, Applications, and Design. Wiley.

[10] BASSO, C: Switch mode power supplies and practical designs. McGraw Hill, 2008.

[11] T.-M. Chen, C.-L. Chen, "Analysis and design of asymmetrical half bridge flyback converter", IEE Pr0c.-Electr. Power Appl., VoL 149, No. 6, November, 2002.

[12] *MATLAB - El lenguaje del cálculo técnico. (s. f.). MATLAB & Simulink.*,de https://es.mathworks.com/products/matlab.html

[13] *Simulink - Simulación y diseño basado en modelos*. (2021). MATLAB & Simulink. https://es.mathworks.com/products/simulink.html

[14] Keogh Bernard, Cohen Isaac, "Flyback transformer design considerations for efficiency and EMI", September 27, 2016.

[15] Intel® FPGA Simulation - ModelSim*- Intel® FPGA. (s. f.). Intel. Recuperado 19 de julio de 2021, de

https://www.intel.es/content/www/es/es/software/programmable/quartus-prime/model-sim.html

[16] Martínez-García SM, de Castro A, Sanchez et al. Analysis of Resolution in Feedback signals for Hardware-in-the-Loop Models of Power Converters. Electronics, vol. 8, no. 12, 2019, htttps://doi.org/10.3390/electronics8121527

[17] Goñi O, Sanchez A, Todorovich E, et al. Resolution Analysis of Switching Converter Models for Hardware-in-the-Loop. IEEE Trans Ind Inf 2014;10(2):1162–70. https://doi.org/10.1109/TII.2013.2294327

[18] Data, S. B. (2020, 20 enero). Aprendizaje automatico y las Metricas de regresión. sitiobigdata.com. https://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/#

Glosario

AC	Alternating Current
CCM	Continuous Conduction Mode
DC	Direct Current
DCM	Discontinuous Conduction Mode
DSP	Digital Signal Processor
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
HIL	Hardware-In-the-Loop
LUT	Look Up Table
MAE	Mean Average Error
MOSFET	Metal-oxide-semiconductor Field-effect transistor
VHDL	VHSIC Hardware Description Language
WL	Word Length

Anexos

A Código VHDL de un Convertidor Flyback, Modelo Real

```
----------------------------------------------------------------------
--
-- Archivo: FlybackConverterReal
--
-- Simulacion de un convertidor Flyback en lazo abierto sin perdidas
----------------------------------------------------------------------
-- Autor: Sandra Peces Martin
-- Fecha: 01 MARZO 2021
----------------------------------------------------------------------
-- EPS - UAM
----------------------------------------------------------------------
-- Valores:
--
-- L = 352u
-- C = 440u
-- Ts = 20ns
-- n = 1
-- fsw = 50KHz
-- Vg = 110 V
-- vout = 48 V
-- R = 46,08
--
----------------------------------------------------------------------

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 FlybackConverterReal 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
	);
end FlybackConverterReal;
```

```
architecture Behavioral of FlybackConverterReal is

	constant C : real := 0.00044;
	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;

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

	constant dtL : real := 0.00005681818;
	constant dtC : real := 0.00004545454;

	signal iin_aux : real:=0.0; -- Corriente del transformador primario, auxiliar
	constant n : real := 1.0; -- Relacion de transformacion


begin

	Iin <= iin_aux;
	Vout <= voutAux;

	SWITCHMUX: process(Mosfet, Vg, Ir, iL, voutAux)
	begin
		if Mosfet = '1' then -- Switch Cerrado
			iLAdd <= Vg;
			voutAuxAdd <= -(Ir);
			iin_aux <= iL;
		else -- Switch Abierto
			if iL > 0.0 then -- CCM
				iLAdd <= -(voutAux/n);
				voutAuxAdd <= (iL/n) - Ir;
			else -- DCM
				iLAdd <= 0.0;
				voutAuxAdd <= -(Ir);
			end if;
		iin_aux <= 0.0;
		end if;
	end process SWITCHMUX;


	DIFFEQ: process(Clk, Reset)
	-- Variables de estado
	begin
		if Reset = '1' then
			voutAux <= VOINIT;
			iL <= ILINIT;
```

```
		elsif rising_edge(Clk) then
			iL <= iL + iLAdd*dtL;
			voutAux <= voutAux + voutAuxAdd*dtC;
		end if;
	end process DIFFEQ;

end Behavioral;


----------------------------------------------------------------------
--
-- Archivo: FlybackTb
--
-- Simulacion de un convertidor Flyback en lazo abierto sin perdidas
----------------------------------------------------------------------
-- Autor: Sandra Peces Martin
-- Fecha: 01 JUNIO 2021
----------------------------------------------------------------------
-- EPS - UAM
----------------------------------------------------------------------
-- Valores:
--
-- L = 352u
-- C = 440u
-- Ts = 20ns
-- n = 1
-- fsw = 50KHz
-- Vg = 110 V
-- vout = 48 V
-- R = 46,08
--
----------------------------------------------------------------------


library IEEE, WORK;
use IEEE.std_logic_1164.all;
use IEEE.math_real.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;

library std;
	use std.textio.all;


entity FlybackTb is
	generic(
		-- Fichero de salida
		fichero_out : string := "values_real.txt"
	);
```

```
end FlybackTb;

architecture Behavioral of FlybackTb is

component FlybackConverterReal
  port(
    Clk : in std_logic;
    Reset : in std_logic;
    Mosfet : in std_logic;
    Vg : in real;
    Ir : in real;
    --Outputs of the ADCs
    Iin : out real;
    debug_IL:out real;
    Vout : out real;
    debug_Vout:out real
          );

end component;

          --Inputs
          signal clk : std_logic := '0';
          signal reset : std_logic := '0';
          signal mosfet : std_logic := '0';

          signal ir : real := 0.0;
          signal vg : real := 110.0;

          constant NUM : integer := 250;

          -- Array variable de entrada Vg con ruido
          type VgRuido is array (1 to NUM) of real;
          -- Orden de Vg a continuación:
          constant Array_Vg_Ruido : VgRuido :=
          (
                    110.926177,
                    110.093611,
                    110.042272,
                    109.463189,
                    109.977795,
                    110.248120,
                    110.358271,
                    109.791030,
                    109.734873,
                    110.975964,
                    109.075478,
                    110.770336,
                    110.826574,
                    110.592368,
                    109.197425,
```

109.523742,
109.670714,
110.359456,
109.273106,
110.442455,
109.213524,
110.307515,
109.988348,
110.558103,
110.430074,
110.807441,
110.781845,
109.668326,
110.397492,
109.395620,
109.061082,
110.488149,
110.000045,
109.959844,
110.809444,
110.219733,
110.235333,
110.718885,
110.610979,
110.153443,
109.365845,
109.479864,
110.773024,
109.057348,
109.979803,
109.335854,
110.957361,
110.425389,
110.000943,
109.942177,
109.119238,
110.363944,
109.084862,
109.142891,
110.043300,
109.193460,
110.636297,
110.635094,
110.444879,
109.299731,
110.319211,
110.037190,
110.945949,
110.297983,
110.600661,

109.907595,
109.864783,
110.650628,
109.166940,
109.266342,
109.346777,
109.781876,
110.662759,
110.606729,
109.120942,
109.798516,
110.053752,
109.833599,
110.313720,
110.255947,
109.583968,
109.863302,
109.030974,
110.968127,
109.334337,
109.212433,
109.744819,
109.396237,
109.979375,
109.678987,
110.903261,
110.840664,
109.105354,
110.475716,
109.538239,
109.845671,
110.095742,
110.885474,
109.835488,
110.966105,
109.602910,
110.402198,
110.332678,
110.078253,
110.396211,
110.333056,
109.356265,
109.256029,
110.998161,
109.342242,
109.065202,
110.122400,
110.763733,
110.338351,
109.380867,

109.737833,
109.921452,
110.963276,
109.312810,
110.711046,
110.289529,
109.752544,
109.381847,
109.856506,
109.964044,
109.241223,
110.179015,
109.452375,
109.769238,
110.165973,
109.503612,
109.580881,
110.234182,
109.530562,
110.648753,
110.965327,
110.460498,
109.687754,
110.168139,
109.215538,
110.812616,
110.759307,
110.635521,
109.521456,
110.188713,
109.045025,
109.850519,
109.625438,
109.322969,
109.357532,
109.845771,
109.188459,
110.197047,
109.941849,
110.391899,
110.399776,
110.277062,
109.067208,
109.137612,
109.639199,
110.061729,
110.308891,
109.815238,
110.639962,
110.436718,

110.937299,
110.062668,
109.650291,
109.211258,
110.221917,
110.557604,
109.846906,
109.181647,
109.532943,
109.307313,
109.562011,
109.880170,
110.054285,
109.914849,
110.750743,
110.036104,
110.887245,
110.275418,
110.915388,
109.481414,
110.352245,
109.578129,
110.343616,
110.390281,
109.135986,
109.509580,
109.448080,
110.335665,
110.688784,
109.688925,
110.561039,
110.350664,
109.013431,
110.204341,
109.773542,
110.831982,
109.002302,
109.924898,
109.848698,
109.921833,
110.540319,
109.644944,
110.569479,
109.942714,
109.071525,
109.351749,
110.443516,
109.946972,
109.305442,
109.682249,

```
        110.214778,
        109.383491,
        110.476854,
        109.485699,
        110.834849,
        109.538123,
        110.531000,
        109.377324,
        109.574996,
        109.182227,
        110.152419,
        110.366726,
        110.093186,
        109.851458,
        110.288886,
        110.295235,
        110.358034,
        110.271573,
        110.890348,
        109.417870,
        110.418563,
        109.472461,
        109.238792,
        110.214608,
        109.900275,
        109.917451,
        110.323890,
        110.540571,
        109.700436,
        110.324019,
        109.832317,
        110.683858,
        110.665834,
        109.512882,
        110.226921
);

-- Fin simulación, por si queremos matar la simulación por falta de eventos
signal finSimu : boolean := false;

--Outputs
signal vout :real := 0.0;
signal iin : real := 0.0;

signal Debug_ILTb : real := 0.0;
signal Debug_VoutTb : real := 0.0;

-- CONSTANTEs
constant R_out : real := 46.08;
-- Periodo de reloj 20 ns (Fclk = 50 MHz)
```

```
        constant CLKPERIOD : time := 20 ns;
        -- Periodo de conmutacion del switch:
        constant Switch_PERIOD : time := 20000 ns;

        ---------------------------------------------------------------
        -- CALCULO DEL DUTY CYCLE
        --Constant V_out : real := 48.0;
        --Constant V_in : real := 110.0;
        --constant n : real := 1.0; -- Relacion de transformacion
        --constant D : real := V_out/(V_out+V_in*n); -- 24/79
        ---------------------------------------------------------------

        -- Tiempo MOSFET ON
        constant T_ON : time := 6060 ns; -- D*SWITCH_PERIOD = 6075.95 ns que pasan
a ser 6060 ns para que sea múltiplo exácto de Tclk.
        --Tiempo MOSFET OFF
        constant T_OFF: time := 13920 ns; -- SWITCH_PERIOD-T_ON = 13920 ns
(multiplo de Tclk)

        -- Resistencia de salida constante:
        signal Rout : real := 46.08;

begin
                uut: FlybackConverterReal  port map (
                        Clk => clk,
                        Reset => reset,
                        Mosfet => mosfet,
                        Vg => vg,
                        Ir => ir,
                        Iin => iin,
                        Vout => vout,
                        -- Debug
                        debug_IL => Debug_ILTb,
                        debug_Vout => Debug_VoutTb
                        );

                --vg <= 110.0;

                clkGen: process                                         -- Proceso de reloj
                begin
                        clk <= '0';
                        wait for CLKPERIOD/2;
                        clk <= '1';
                        wait for CLKPERIOD/2;
                end process clkGen;

                -- Proceso del reset
                general: process
                begin
                        reset <= '1';
```

```
                wait for CLKPERIOD;
                reset <= '0';
                wait for CLKPERIOD*100000000;
                finSimu <= true;
                wait; -- No se vuelve a hacer nada con el reset
        end process;

        MsftProcess: process                                    -- Proceso de Mosfet
        begin
                mosfet <= '0';
                wait for T_OFF;
                mosfet <= '1';
                wait for T_ON;
        end process MsftProcess;

--Mantenemos resistencia constante, en lugar de corriente constante:

        RoutProcess: process            -- Proceso de resistencia, por si queremos hacer un
barrido de resistencias
        begin
          -- Valor fijo de Rout --
          Rout <= R_out;
          wait;

        end process RoutProcess;

ir <= vout/Rout;

-- Proceso para asignar valores de Vg con ruido.
VGRUIDOSA: process
        begin
        while (not finSimu) loop
                -- Repetir para todos los casos de Vg
                for i in 1 to NUM loop
                        vg <= Array_Vg_Ruido(i);--to_slv (to_sfixed (,sVg_sfixed));
                        wait for SWITCH_PERIOD;
                end loop;
        end loop;
        wait;
end process;

-- PROCESO PARA EXPORTAR LOS VALORES A UN FICHERO
Ficheros: process
-- Save results
file      outfile  : text is out fichero_out;
variable  outline  : line;
variable s : string (1 to 1) := " ";
variable w_iL, w_vout: real;
begin
```

```
                for i in 1 to 4700000 loop -- Número de iteraciones (puntos). En este caso, 100000 * 10 ns = 1 ms. (Eso es el tiempo total que necesito de simulación. Lo lógico es hacerlo hasta el 5% o 2% del settling time)
                        wait for 20 ns; -- Muestreo cada 10 ns. El filtrado para quitar el ruido de conmutación lo hago en Matlab. Lo único a tener en cuenta es que estos 10 ns deberían preferiblemente ser submúltiplo del Tsw, para evitar aliasing
                        w_iL := Debug_ILTb; -- IL_debug es mi variable de estado para la corriente
                        write (outline, w_iL, right,25, 17); -- 20, 17
                        w_vout := Debug_VoutTb; -- Vout_debug  es mi variable de estado para la tensión
                        write (outline, w_vout, right, 25, 17); --20, 17
                        writeline (outfile, outline);
                end loop;
                wait;
        end process Ficheros;


        end Behavioral;
```

B Código VHDL de un Convertidor Flyback, Modelo Coma Fija

```
----------------------------------------------------------------------
--
-- Archivo: FlybackConverter_sfixed
--
-- Simulacion de un convertidor Flyback en lazo abierto sin perdidas en formato sfixed
----------------------------------------------------------------------
-- Autor: Sandra Peces Martin
-- Fecha: 25 JUNIO 2021
----------------------------------------------------------------------
-- EPS - UAM
----------------------------------------------------------------------
-- Valores:
--
-- L = 352u
-- C = 440u
-- Ts = 20ns
-- n = 1
-- fsw = 50KHz
-- Vg = 110 V
-- vout = 48 V
-- R = 46,08
--
----------------------------------------------------------------------

-- LIBRERIAS
library IEEE;
 use IEEE.std_logic_1164.all;
 use IEEE.numeric_std.all;
 use IEEE.std_logic_unsigned.all;
 use IEEE.std_logic_arith.all;
 use IEEE.math_real.all;
 use IEEE.fixed_float_types.all; -- ieee_proposed for VHDL-93 version
 use IEEE.fixed_pkg.all; -- ieee_proposed for compatibility version

-- Libreria para imprimir valores en un documento de texto
library std;
	use std.textio.all;

entity FlybackConverter_sfixed is

	generic(
		-- n de las variables de estado
		n1v : integer := 32;
		n1i : integer := 32;
		-- n de las señales realimentadas
		n2v : integer := 32;
		n2i : integer := 32;
```

```
		-- n de las constantes
		n3C : integer := 32;
		n3L : integer := 32;
		n3n : integer := 32
	);

	port (
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic; -- On = '1', off = '0'
		-- Depende del valor de sVg, hay que contabilizar el número total de bits tanto de la parte entera como de la parte decimal
		Vg : in std_logic_vector (14+n2v downto 0); -- 15 bits
		Ir : in std_logic_vector (5+n2i downto 0); -- 6 bits
		-- Out
		Iin : out std_logic_vector (16+n1i downto 0); -- 17 bits
		debug_IL:out real;

		Vout : out std_logic_vector (25+n1v downto 0); -- 26 bits
		debug_Vout:out real
	);

end FlybackConverter_sfixed;


architecture Behavioral of FlybackConverter_sfixed is


	-- CONSTANTES
	constant dtL : sfixed (-14 downto -(15+n3L)):= to_sfixed(0.0000568181818181818, -14, -(15+n3L));
	constant dtC : sfixed (-14 downto -(15+n3C)):= to_sfixed(0.0000454545454545454, -14, -(15+n3C));
	constant n : sfixed (1 downto -(0+n3n)):= to_sfixed(1, 1, -(0+n3n)); -- Relacion de transformacion
	constant VOINIT : sfixed(8 downto -(17+n1v)):= (others=>'0');
	constant ILINIT : sfixed(7 downto -(9+n1i)):= (others=>'0');

	-- SENALES DE ESTADO
	signal iL : sfixed(7 downto -(9+n1i)):= (others=>'0');
	signal voutAux : sfixed(8 downto -(17+n1v)):= to_sfixed(48,8,-(17+n1v));

	-- SENALES DE ENTRADA
	signal sVg: sfixed (8 downto -(6+n2v));
	signal sIr: sfixed (3 downto -(2+n2i));

	-- SEÑALES REALIMENTADAS
	signal vL :  sfixed (8 downto -(6+n2v));
	signal iC : sfixed (7 downto -(2+n2i));
```

```
	-- SENALES DEPENDIENTES
	signal iLAdd : sfixed (-5 downto -(9+n1i));
	signal voutAuxAdd : sfixed (-6 downto -(17+n1v));
	signal iin_aux : sfixed (7 downto -(9+n1i));

	signal iLFeedBack : sfixed (9 downto -(2+n2i));
	signal voutFeedBack : sfixed (10 downto -(6+n2v));


begin

	-- Convertimos las entradas en sfixed
	sVg <= to_sfixed (Vg, sVg);
	sIr <= to_sfixed (Ir, sIr);

	-- Convertimos las salidas en std_logic_vector
	Iin <= to_slv (iin_aux);
	Vout <= to_slv (voutAux);

		--Variables realimentadas que dependen de las variables de estado.
	iLFeedBack <= resize (iL, iLFeedBack, fixed_wrap, fixed_truncate);
	voutFeedBack <= resize (voutAux, voutFeedBack, fixed_wrap, fixed_truncate);

	SWITCHMUX: process(Mosfet, sVg, sIr, iL, iLFeedBack,voutFeedBack)
	begin
		if Mosfet = '1' then -- Switch Cerrado
			vL <= resize(sVg,vL);
			iC <= resize(-sIr,iC);
			iin_aux <= resize (iL, iin_aux, fixed_wrap, fixed_truncate);

		else -- Switch Abierto
			if iL > 0.0 then -- CCM
				vL <= resize(-(voutFeedBack/n),vL);
				iC <= resize((iLFeedBack/n) - sIr,iC);
				--iC <= (iLFeedBack/n) - sIr;

			else -- DCM
				vL <= (others => '0');
				iC <= resize(-(sIr),iC);

			end if;
		iin_aux <= (others => '0');

		end if;
	end process SWITCHMUX;

	-- Obtención del producto fuera del process (es combinacional)
	iLAdd <= resize (vL*dtL, iLAdd, fixed_wrap, fixed_truncate);
```

```
	voutAuxAdd <= resize (iC*dtC, voutAuxAdd, fixed_wrap, fixed_truncate);


	DIFFEQ: process(Clk, Reset)
	-- Variables de estado
	begin
		if Reset = '1' then
			iL <= ILINIT;
			voutAux <= VOINIT;
		elsif rising_edge(Clk) then
			iL <= resize (iL + iLAdd, iL, fixed_wrap, fixed_truncate);
			voutAux <= resize (voutAux + voutAuxAdd, voutAux, fixed_wrap,
fixed_truncate);
		end if;
	end process DIFFEQ;

	-- SALIDAS DE COMPROBACION DE LAS SEÑALES
	debug_Vout <= to_real(voutAux);
	debug_IL <= to_real(iL);

end Behavioral;
```

```
--------------------------------------------------------------------
--
-- Archivo: FlybackTb_sfixed
--
-- Simulacion de un convertidor Flyback en lazo abierto sin perdidas
--------------------------------------------------------------------
-- Autor: Sandra Peces Martin
-- Fecha: 25 JUNIO 2021
--------------------------------------------------------------------
-- EPS - UAM
--------------------------------------------------------------------
-- Valores:
--
-- L = 352uH
-- C = 440uF
-- Ts = 20 ns
-- n = 1
-- fsw = 50 KHz
-- Vg = 110 V
-- vout = 48 V
-- R = 46,08
--
--------------------------------------------------------------------

-- LIBRERIAS
library WORK, IEEE;
	use IEEE.std_logic_1164.all;
	use IEEE.numeric_std.all;
	use IEEE.std_logic_unsigned.all;
	use IEEE.std_logic_arith.all;
	use IEEE.math_real.all;
	use IEEE.fixed_pkg.all;

library std;
	use std.textio.all;

--------------------------------------------------------------------

entity FlybackTb_sfixed is
	generic(
		-- Valores por defecto,los que realmente coge son los del portmap en el TB
		-- n de las variables de estado
		n1v : integer := 32;
		n1i : integer := 32;
		-- n de las senales realimentadas y las entradas
		n2v : integer := 32;
		n2i : integer := 32;
		-- n de las constantes
		n3C : integer := 32;
		n3L : integer := 32;
```

```
		n3n : integer := 32;
		-- Fichero de salida
		fichero_out : string := "values_sfixed32.txt"
	);

end FlybackTb_sfixed;

------------------------------------------------------------

architecture Behavioral of FlybackTb_sfixed is

------------------------------------------------------------
	component FlybackConverter_sfixed
		port (
			--Inputs
			Clk : in std_logic;
			Reset : in std_logic;
			Mosfet : in std_logic; -- On = '1', off = '0'
			Vg : in std_logic_vector (14+n2v downto 0);
			Ir : in std_logic_vector (5+n2i downto 0);
			--Outputs of the ADCs
			Iin : out std_logic_vector (16+n1i downto 0);
			Vout : out std_logic_vector (25+n1v downto 0);
			-- Debug
			debug_IL : out real;
			debug_Vout : out real
		);
	end component;


	-- Senales de entrada:
	signal clk : std_logic := '0';
	signal rst : std_logic := '0';
	signal mosfet : std_logic := '0';

	signal sVg : std_logic_vector (14+n2v downto 0); -- Le asignaremos valores con
ruido blanco
	signal sVg_sfixed : sfixed (8 downto -(6+n2v));

	signal sIr : std_logic_vector (5+n2i downto 0):= (others=>'0');
	signal sIr_sfixed: sfixed (3 downto -(2+n2i));

	-- Senales de salida:
	signal sIin : std_logic_vector (16+n1i downto 0);
	signal sVout : std_logic_vector (25+n1v downto 0);

	-- Senales auxiliares:
	--signal sIr_real : real := 0.0; -- Y en versión real
	signal sVout_real : real; -- Y en versión real
	signal Debug_ILTb : real := 0.0;
```

```
signal Debug_VoutTb : real := 0.0;

-- CONSTANTES

constant Rout : real := 46.08;
-- Periodo de reloj 20 ns (Fclk = 50 MHz)
constant CLKPERIOD : time := 20 ns;
-- Periodo de conmutacion del switch:
constant SWITCH_PERIOD : time := 20000 ns;

-----------------------------------------------------------------
-- CALCULO DEL DUTY CYCLE
--Constant V_out : real := 48.0;
--Constant V_in : real := 110.0;
--constant n : real := 1.0; -- Relacion de transformacion
--constant D : real := V_out/(V_out+V_in*n); -- 24/79
-----------------------------------------------------------------

-- Tiempo MOSFET ON
constant T_ON : time := 6060 ns; -- D*SWITCH_PERIOD = 6075.95 ns que pasan a ser 6060 ns para que sea múltiplo exácto de Tclk.
--Tiempo MOSFET OFF
constant T_OFF: time := 13920 ns; -- SWITCH_PERIOD-T_ON = 13920 ns (multiplo de Tclk)

constant NUM : integer := 250;

-- Array variable de entrada Vg con ruido
type VgRuido is array (1 to NUM) of real;
-- Orden de Vg a continuación:
constant Array_Vg_Ruido : VgRuido :=
(
        110.926177,
        110.093611,
        110.042272,
        109.463189,
        109.977795,
        110.248120,
        110.358271,
        109.791030,
        109.734873,
        110.975964,
        109.075478,
        110.770336,
        110.826574,
        110.592368,
        109.197425,
        109.523742,
        109.670714,
        110.359456,
```

109.273106,
110.442455,
109.213524,
110.307515,
109.988348,
110.558103,
110.430074,
110.807441,
110.781845,
109.668326,
110.397492,
109.395620,
109.061082,
110.488149,
110.000045,
109.959844,
110.809444,
110.219733,
110.235333,
110.718885,
110.610979,
110.153443,
109.365845,
109.479864,
110.773024,
109.057348,
109.979803,
109.335854,
110.957361,
110.425389,
110.000943,
109.942177,
109.119238,
110.363944,
109.084862,
109.142891,
110.043300,
109.193460,
110.636297,
110.635094,
110.444879,
109.299731,
110.319211,
110.037190,
110.945949,
110.297983,
110.600661,
109.907595,
109.864783,
110.650628,

109.166940,
109.266342,
109.346777,
109.781876,
110.662759,
110.606729,
109.120942,
109.798516,
110.053752,
109.833599,
110.313720,
110.255947,
109.583968,
109.863302,
109.030974,
110.968127,
109.334337,
109.212433,
109.744819,
109.396237,
109.979375,
109.678987,
110.903261,
110.840664,
109.105354,
110.475716,
109.538239,
109.845671,
110.095742,
110.885474,
109.835488,
110.966105,
109.602910,
110.402198,
110.332678,
110.078253,
110.396211,
110.333056,
109.356265,
109.256029,
110.998161,
109.342242,
109.065202,
110.122400,
110.763733,
110.338351,
109.380867,
109.737833,
109.921452,
110.963276,

109.312810,
110.711046,
110.289529,
109.752544,
109.381847,
109.856506,
109.964044,
109.241223,
110.179015,
109.452375,
109.769238,
110.165973,
109.503612,
109.580881,
110.234182,
109.530562,
110.648753,
110.965327,
110.460498,
109.687754,
110.168139,
109.215538,
110.812616,
110.759307,
110.635521,
109.521456,
110.188713,
109.045025,
109.850519,
109.625438,
109.322969,
109.357532,
109.845771,
109.188459,
110.197047,
109.941849,
110.391899,
110.399776,
110.277062,
109.067208,
109.137612,
109.639199,
110.061729,
110.308891,
109.815238,
110.639962,
110.436718,
110.937299,
110.062668,
109.650291,

109.211258,
110.221917,
110.557604,
109.846906,
109.181647,
109.532943,
109.307313,
109.562011,
109.880170,
110.054285,
109.914849,
110.750743,
110.036104,
110.887245,
110.275418,
110.915388,
109.481414,
110.352245,
109.578129,
110.343616,
110.390281,
109.135986,
109.509580,
109.448080,
110.335665,
110.688784,
109.688925,
110.561039,
110.350664,
109.013431,
110.204341,
109.773542,
110.831982,
109.002302,
109.924898,
109.848698,
109.921833,
110.540319,
109.644944,
110.569479,
109.942714,
109.071525,
109.351749,
110.443516,
109.946972,
109.305442,
109.682249,
110.214778,
109.383491,
110.476854,

```
            109.485699,
            110.834849,
            109.538123,
            110.531000,
            109.377324,
            109.574996,
            109.182227,
            110.152419,
            110.366726,
            110.093186,
            109.851458,
            110.288886,
            110.295235,
            110.358034,
            110.271573,
            110.890348,
            109.417870,
            110.418563,
            109.472461,
            109.238792,
            110.214608,
            109.900275,
            109.917451,
            110.323890,
            110.540571,
            109.700436,
            110.324019,
            109.832317,
            110.683858,
            110.665834,
            109.512882,
            110.226921
    );

    -- Fin simulación, por si queremos matar la simulación por falta de eventos
    signal finSimu : boolean := false;

begin
    -- Instanciación del componente BuckConverterRealSincrono
    uut: FlybackConverter_sfixed
    generic map(
        -- n de las Variables de Estado
        n1v => n1v,
        n1i => n1i,
        -- n de las realimentadas
        n2i => n2i,
        n2v => n2v,
        -- n de las constantes
        n3C => n3C,
        n3L => n3L,
```

```
        n3n => n3n
)
port map (
        Clk => clk,
        Reset => rst,
        Mosfet => mosfet,
        Vg => sVg,
        Ir => sIr,
        -- Out
        Iin => sIin,
        Vout => sVout,
        -- Debug
        debug_IL => Debug_ILTb,
        debug_Vout => Debug_VoutTb
);

-- Proceso señal de reloj
CLKPROCESS: process
begin
        clk <= '0';
        wait for CLKPERIOD/2;
        clk <= '1';
        wait for CLKPERIOD/2;
end process CLKPROCESS;

-- Proceso de Mosfet
MsftProcess: process
begin
        mosfet <= '0';
        wait for T_OFF;
        mosfet <= '1';
        wait for T_ON;
end process MsftProcess;

-- Calculo de Ir
sVout_real <= to_real (to_sfixed (sVout, 8, -(17+n1v)));
--sIr_real <= sVout_real/Rout;
sIr_sfixed <= to_sfixed (sVout_real/Rout, 3, -(2+n2i));
sIr <= to_slv (sIr_sfixed);

-- Proceso del reset
general: process
begin
        rst <= '1';
        wait for CLKPERIOD;
        rst <= '0';
        wait for CLKPERIOD*100000000;
        finSimu <= true;
        wait; -- No se vuelve a hacer nada con el reset
end process;
```

```
-- Proceso para asignar valores de Vg con ruido.
VGRUIDOSA: process
        begin
        while (not finSimu) loop
                -- Repetir para todos los casos de Vg
                for i in 1 to NUM loop
                        sVg <= to_slv (to_sfixed (Array_Vg_Ruido(i),sVg_sfixed));
                        wait for SWITCH_PERIOD;
                end loop;
        end loop;
        wait;
end process;


-- PROCESO PARA EXPORTAR LOS VALORES A UN FICHERO
Ficheros: process
-- Save results
file     outfile  : text is out fichero_out;
variable  outline  : line;
variable s : string (1 to 1) := " ";
variable w_iL, w_vout: real;
begin
        for i in 1 to 4700000 loop -- Número de iteraciones (puntos). En este caso, 100000 * 10 ns = 1 ms. (Eso es el tiempo total que necesito de simulación. Lo lógico es hacerlo hasta el 5% o 2% del settling time)
                wait for 20 ns; -- Muestreo cada 10 ns. El filtrado para quitar el ruido de conmutación lo hago en Matlab. Lo único a tener en cuenta es que estos 10 ns deberían preferiblemente ser submúltiplo del Tsw, para evitar aliasing
                w_iL := Debug_ILTb; -- IL_debug es mi variable de estado para la corriente
                write (outline, w_iL, right,25, 17); -- 20, 17
                w_vout := Debug_VoutTb; -- Vout_debug  es mi variable de estado para la tensión
                write (outline, w_vout, right, 25, 17); --20, 17
                writeline (outfile, outline);
        end loop;
        wait;
end process Ficheros;

end Behavioral;
```

C Script de simulación Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Archivo: Flyback_Sfixed_v4
%
%   Script base de calculo y representacion del error relativo para iL y
%   Vout
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Autor: Sandra Peces Martin
%   Fecha: 15 AGOSTO 2021
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   EPS - UAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Valores:
%
%   L = 352uH
%   C = 440uF
%   Ts = 20 ns
%   n = 1
%   fsw = 50 KHz
%   Vg = 110 V
%   Vout_sfixed = 48 V
%   R = 46,08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lectura de los datos del fichero del modelo REAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

aux=dlmread('values_real.txt');
IL_real = aux(:,1);
Vout_real = aux(:,2);

%%
time_vector = linspace(0,90,length(IL_real));
figure;
subplot(1,2,1);
plot(time_vector,IL_real,'b','LineWidth',1);
grid on;
```

```
xlabel('t [ms]','FontSize',12);
ylabel('iL real [A]','FontSize',12);

subplot(1,2,2);
plot(time_vector,Vout_real,'b','LineWidth',1);
grid on;
xlabel('t [ms]','FontSize',12);
ylabel('Vout real [V]','FontSize',12);
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lectura de los datos del fichero del modelo sfixed con n = 32 bits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

aux2=dlmread('values_sfixed32.txt');
IL_sfixed32 = aux2(:,1);
Vout_sfixed32 = aux2(:,2);
long= aux2(:,1);
%%
figure;
subplot(1,2,1);
plot(time_vector,IL_sfixed32,'r','LineWidth',1);
grid on;
xlabel('t [ms]','FontSize',12);
ylabel('iL sfixed [A]','FontSize',12);

subplot(1,2,2);
plot(time_vector,Vout_sfixed32,'r','LineWidth',1);
grid on;
xlabel('t [ms]','FontSize',12);
ylabel('Vout sfixed [V]','FontSize',12);
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Representacion de las senales iL y Vout, Modelo Real y sfixed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time_vector = linspace(0,90,length(IL_real)); %CAMBIO

figure;
subplot(1,2,1);
plot(time_vector,IL_real,'b','LineWidth',1);
hold on;
plot(time_vector,IL_sfixed32,'r','LineWidth',1);
grid on;
xlabel('t [ms]','FontSize',12);
ylabel('iL [A]','FontSize',12);
legend('iL Real', 'iL Coma Fija');
hold off;

subplot(1,2,2);
plot(time_vector,Vout_real,'b','LineWidth',1);
```

```
hold on;
plot(time_vector,Vout_sfixed32,'r','LineWidth',1);
grid on;
xlabel('t [ms]','FontSize',12);
ylabel('Vout [V]','FontSize',12);
legend('Vout Real', 'Vout Coma Fija');
hold off;

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculo del error relativo entre el Modelo Real y el Modelo en Coma Fija
% de 32 bits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IL_tipico = 1.15; %A
Vout_tipico = 48; %V
n_samples = length(IL_real);

MAE_iL = sum(abs(IL_sfixed32 - IL_real)); %MAE, Error Absoluto Medio
MAE_optimizado_iL = MAE_iL/n_samples;
errorRelativo_iL= MAE_optimizado_iL/IL_tipico

% Calculo del error relativo de Vout
MAE_Vout = sum(abs(Vout_sfixed32 - Vout_real)); %MAE, Error Absoluto Medio
MAE_optimizado_Vout = MAE_Vout/n_samples;
errorRelativo_Vout = MAE_optimizado_Vout/Vout_tipico

% Error relativo en escala logarítimica:
errorRelativo_iL_Log = log10(errorRelativo_iL)
errorRelativo_Vout_Log = log10(errorRelativo_Vout)


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lectura de los datos de los ficheros de simulacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cd n1v % Nos metemos en la carpeta n1v
numfiles = 21;

% Lectura del error n1v %

n1vdata = cell(1, numfiles);
for i = 1:numfiles
  file_n1v = sprintf('valor0%dn1v.txt', i);
  n1vdata{i} = importdata(file_n1v);
end
cd .. % Volvemos al directorio raiz
```

```
% Lectura del error n1i %

numfiles = 16;
cd n1i % Nos metemos en la carpeta n1i

n1idata = cell(1, numfiles);
for i = 1:numfiles
  file_n1i = sprintf('valor0%dn1i.txt', i);
  n1idata{i} = importdata(file_n1i);
end
cd .. % Volvemos al directorio raiz


% % Lectura del error n2v %

numfiles = 21;
cd n2v % Nos metemos en la carpeta n2v

n2vdata = cell(1, numfiles);
for i = 1:numfiles
  file_n2v = sprintf('valor0%dn2v.txt', i);
  n2vdata{i} = importdata(file_n2v);
end
cd .. % Volvemos al directorio raiz


% Lectura del error n2i %

numfiles = 18;
cd n2i % Nos metemos en la carpeta n2i

n2idata = cell(1, numfiles);
for i = 1:numfiles
  file_n2i = sprintf('valor0%dn2i.txt', i);
  n2idata{i} = importdata(file_n2i);
end
cd .. % Volvemos al directorio raiz


% Lectura del error n3c %

numfiles = 16;
cd n3c % Nos metemos en la carpeta n3c

n3cdata = cell(1, numfiles);
for i = 1:numfiles
  file_n3c = sprintf('valor0%dn3c.txt', i);
  n3cdata{i} = importdata(file_n3c);
end
cd .. % Volvemos al directorio raiz
```

```
% Lectura del error n3l %

numfiles = 16;
cd n3l % Nos metemos en la carpeta n3l

n3ldata = cell(1, numfiles);
for i = 1:numfiles
  file_n3l = sprintf('valor0%dn3l.txt', i);
  n3ldata{i} = importdata(file_n3l);
end
cd .. % Volvemos al directorio raiz

% Lectura del error n3n %

numfiles = 16;
cd n3n % Nos metemos en la carpeta n3l

n3ndata = cell(1, numfiles);
for i = 1:numfiles
  file_n3n = sprintf('valor0%dn3n.txt', i);
  n3ndata{i} = importdata(file_n3n);
end
cd .. % Volvemos al directorio raiz


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculo del error relativo para cada tipo de señal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

long_real = length(IL_real);

[ErrorRelativo_IL_n1v_Log, ErrorRelativo_Vout_n1v_Log] = calculo_error(n1vdata, 21,
IL_real,Vout_real);
[ErrorRelativo_IL_n1i_Log, ErrorRelativo_Vout_n1i_Log] = calculo_error(n1idata, 16,
IL_real,Vout_real);
[ErrorRelativo_IL_n2v_Log, ErrorRelativo_Vout_n2v_Log] = calculo_error(n2vdata, 21,
IL_real,Vout_real);
[ErrorRelativo_IL_n2i_Log, ErrorRelativo_Vout_n2i_Log] = calculo_error(n2idata, 18,
IL_real,Vout_real);
[ErrorRelativo_IL_n3c_Log, ErrorRelativo_Vout_n3c_Log] = calculo_error(n3cdata, 16,
IL_real,Vout_real);
[ErrorRelativo_IL_n3l_Log, ErrorRelativo_Vout_n3l_Log] = calculo_error(n3ldata, 16,
IL_real,Vout_real);
[ErrorRelativo_IL_n3n_Log, ErrorRelativo_Vout_n3n_Log] = calculo_error(n3ndata, 16,
IL_real,Vout_real);

%%
```

```
% Se ajustan la longitud de los vectores para que todas tengan la misma,
% esto se hace rellenando las primeras posiciones con NaN

% Ajuste n1i
vector_NaN_n1i = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_n1i_Log),1);
vector_NaN_n1i(:,1) = NaN;

ErrorRelativo_IL_n1i_Log = [vector_NaN_n1i;ErrorRelativo_IL_n1i_Log]; %
Concatenamos los vectores
ErrorRelativo_Vout_n1i_Log = [vector_NaN_n1i;ErrorRelativo_Vout_n1i_Log]; %
Concatenamos los vectores

% Ajuste n2i
vector_NaN_n2i = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_n2i_Log),1);
vector_NaN_n2i(:,1) = NaN;

ErrorRelativo_IL_n2i_Log = [vector_NaN_n2i;ErrorRelativo_IL_n2i_Log]; %
Concatenamos los vectores
ErrorRelativo_Vout_n2i_Log = [vector_NaN_n2i;ErrorRelativo_Vout_n2i_Log]; %
Concatenamos los vectores

% Ajuste n3c
vector_NaN_n3c = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_n3c_Log),1);
vector_NaN_n3c(:,1) = NaN;

ErrorRelativo_IL_n3c_Log = [vector_NaN_n3c;ErrorRelativo_IL_n3c_Log]; %
Concatenamos los vectores
ErrorRelativo_Vout_n3c_Log = [vector_NaN_n3c;ErrorRelativo_Vout_n3c_Log]; %
Concatenamos los vectores

% Ajuste n3l
vector_NaN_n3l = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_n3l_Log),1);
vector_NaN_n3l(:,1) = NaN;

ErrorRelativo_IL_n3l_Log = [vector_NaN_n3l;ErrorRelativo_IL_n3l_Log]; %
Concatenamos los vectores
ErrorRelativo_Vout_n3l_Log = [vector_NaN_n3l;ErrorRelativo_Vout_n3l_Log]; %
Concatenamos los vectores

% Ajuste n3n
vector_NaN_n3n = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_n3n_Log),1);
vector_NaN_n3n(:,1) = NaN;

ErrorRelativo_IL_n3n_Log = [vector_NaN_n3n;ErrorRelativo_IL_n3n_Log]; %
Concatenamos los vectores
```

```
ErrorRelativo_Vout_n3n_Log = [vector_NaN_n3n;ErrorRelativo_Vout_n3n_Log]; %
Concatenamos los vectores


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lectura de los datos de los ficheros de simulacion nOpt = 4 bits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

numfiles = 19;
cd nOpt % Nos metemos en la carpeta nOpt

nOptdata = cell(1, numfiles);
for i = 1:numfiles
  file_nOpt = sprintf('valor0%dnOpt.txt', i);
  nOptdata{i} = importdata(file_nOpt);
end
cd .. % Volvemos al directorio raiz

%%
% Calculo del error respecto nOpt

[ErrorRelativo_IL_nOpt_Log, ErrorRelativo_Vout_nOpt_Log] = calculo_error(nOptdata,
19, IL_real,Vout_real);

% Ajuste nOpt
vector_NaN_nOpt = zeros(length(ErrorRelativo_IL_n1v_Log)-
length(ErrorRelativo_IL_nOpt_Log),1);
vector_NaN_nOpt(:,1) = NaN;

ErrorRelativo_IL_nOpt_Log = [vector_NaN_nOpt;ErrorRelativo_IL_nOpt_Log]; %
Concatenamos los vectores
ErrorRelativo_Vout_nOpt_Log = [vector_NaN_nOpt;ErrorRelativo_Vout_nOpt_Log]; %
Concatenamos los vectores


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Representacion del error relativo de iL y Vout respecto a n
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n_vector = linspace(-10,30,21);

% iL

newcolors =
{'#0072BD','#D95319','#EDB120','#7E2F8E','#77AC30','#4DBEEE','#E9967A'};
colororder(newcolors)

figure;
```

```
plot(n_vector,ErrorRelativo_IL_n1v_Log, '--d',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#0072BD',...
    'MarkerFaceColor','#0072BD');
hold on
plot(n_vector,ErrorRelativo_IL_n1i_Log, '--s',...
    'LineWidth',1,...
    'MarkerSize',7,...
    'MarkerEdgeColor','#D95319',...
    'MarkerFaceColor','#D95319');
plot(n_vector,ErrorRelativo_IL_n2v_Log, '--s',...
    'LineWidth',1,...
    'MarkerSize',7,...
    'MarkerEdgeColor','#EDB120',...
    'MarkerFaceColor','#EDB120');
plot(n_vector,ErrorRelativo_IL_n2i_Log, '--d',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#7E2F8E',...
    'MarkerFaceColor','#7E2F8E');
plot(n_vector,ErrorRelativo_IL_n3c_Log, '--<',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#77AC30',...
    'MarkerFaceColor','#77AC30');
plot(n_vector,ErrorRelativo_IL_n3l_Log, '--<',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#4DBEEE',...
    'MarkerFaceColor','#4DBEEE');
% plot(n_vector,ErrorRelativo_IL_n3n_Log, '--<',...
%     'LineWidth',1,...
%     'MarkerSize',6,...
%     'MarkerEdgeColor','#E9967A',...
%     'MarkerFaceColor','#E9967A');
plot(n_vector,ErrorRelativo_IL_nOpt_Log, '--k',...
    'LineWidth',1);
hold off
grid minor
xlabel('n','FontSize',12);
ylabel('log_{10}(e)','FontSize',12);
title('Modelo en Coma Fija, I_L','FontSize',12,'FontWeight','bold');
legend('n1v','n1i','n2v','n2i','n3c','n3l','nOpt');
% legend('n1v','n1i','n2v','n2i','n3c','n3l','n3n','nOpt');


%% Vout

figure;
```

```
plot(n_vector,ErrorRelativo_Vout_n1v_Log, '--d',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#0072BD',...
    'MarkerFaceColor','#0072BD');
hold on
plot(n_vector,ErrorRelativo_Vout_n1i_Log, '--s',...
    'LineWidth',1,...
    'MarkerSize',7,...
    'MarkerEdgeColor','#D95319',...
    'MarkerFaceColor','#D95319');
plot(n_vector,ErrorRelativo_Vout_n2v_Log, '--s',...
    'LineWidth',1,...
    'MarkerSize',7,...
    'MarkerEdgeColor','#EDB120',...
    'MarkerFaceColor','#EDB120');
plot(n_vector,ErrorRelativo_Vout_n2i_Log, '--d',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#7E2F8E',...
    'MarkerFaceColor','#7E2F8E');
plot(n_vector,ErrorRelativo_Vout_n3c_Log, '--<',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#77AC30',...
    'MarkerFaceColor','#77AC30');
plot(n_vector,ErrorRelativo_Vout_n3l_Log, '--<',...
    'LineWidth',1,...
    'MarkerSize',6,...
    'MarkerEdgeColor','#4DBEEE',...
    'MarkerFaceColor','#4DBEEE');
plot(n_vector,ErrorRelativo_Vout_nOpt_Log, '--k',...
    'LineWidth',1);
hold off
grid minor
xlabel('n');
ylabel('log_{10}(e)');
title('Modelo en Coma Fija, V_{out}','FontSize',12,'FontWeight','bold');
legend('n1v','n1i','n2v','n2i','n3c','n3l','nOpt');


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Representacion del error relativo de iL y Vout respecto a WL: X+Ym+n
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Vout: Q 8.(17 + n1v)
% iL: Q 7.(9 + n1i)
% --
% vL: Q 8.(6 + n2v)
```

```
% iC: Q 7.(2 + n2i)
% --
% dt/L: Q -14.(15 + n3l)
% dt/C: Q -14.(15 + n3c)
%
% Donde cada nxx es un barrido de -10 a 30, de 2 en 2.

% Se calcula el eje X en funcion de los valores de n = [-10, ...,30]
vector_n1v_WL = vector_WL(8, 17);
vector_n1i_WL = vector_WL(7, 9);
vector_n2v_WL = vector_WL(8, 6);
vector_n2i_WL = vector_WL(7, 2);
vector_n3lc_WL = vector_WL(-14, 15);

% iL

figure;
plot(vector_n1v_WL,ErrorRelativo_IL_n1v_Log, '--d',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#0072BD',...
   'MarkerFaceColor','#0072BD');
hold on
plot(vector_n1i_WL,ErrorRelativo_IL_n1i_Log, '--d',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#D95319',...
   'MarkerFaceColor','#D95319');
plot(vector_n2v_WL,ErrorRelativo_IL_n2v_Log, '--s',...
   'LineWidth',1,...
   'MarkerSize',7,...
   'MarkerEdgeColor','#EDB120',...
   'MarkerFaceColor','#EDB120');
plot(vector_n2i_WL,ErrorRelativo_IL_n2i_Log, '--s',...
   'LineWidth',1,...
   'MarkerSize',7,...
   'MarkerEdgeColor','#7E2F8E',...
   'MarkerFaceColor','#7E2F8E');
plot(vector_n3lc_WL,ErrorRelativo_IL_n3c_Log, '--<',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#77AC30',...
   'MarkerFaceColor','#77AC30');
plot(vector_n3lc_WL,ErrorRelativo_IL_n3l_Log, '--<',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#4DBEEE',...
   'MarkerFaceColor','#4DBEEE');
plot(vector_n1v_WL,ErrorRelativo_IL_nOpt_Log, '--k',...
   'LineWidth',1);
```

```
hold on
hold off
grid minor
xlabel('X + Ym + n','FontSize',12);
ylabel('log_{10}(e)','FontSize',12);
title('Modelo en Coma Fija, I_L','FontSize',12,'FontWeight','bold');
legend('n1v','n1i','n2v','n2i','n3c','n3l','nOpt');

% Vout

figure;
plot(vector_n1v_WL,ErrorRelativo_Vout_n1v_Log, '--d',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#0072BD',...
   'MarkerFaceColor','#0072BD');
hold on
plot(vector_n1i_WL,ErrorRelativo_Vout_n1i_Log, '--d',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#D95319',...
   'MarkerFaceColor','#D95319');
plot(vector_n2v_WL,ErrorRelativo_Vout_n2v_Log, '--s',...
   'LineWidth',1,...
   'MarkerSize',7,...
   'MarkerEdgeColor','#EDB120',...
   'MarkerFaceColor','#EDB120');
plot(vector_n2i_WL,ErrorRelativo_Vout_n2i_Log, '--s',...
   'LineWidth',1,...
   'MarkerSize',7,...
   'MarkerEdgeColor','#7E2F8E',...
   'MarkerFaceColor','#7E2F8E');
plot(vector_n3lc_WL,ErrorRelativo_Vout_n3c_Log, '--<',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#77AC30',...
   'MarkerFaceColor','#77AC30');
plot(vector_n3lc_WL,ErrorRelativo_Vout_n3l_Log, '--<',...
   'LineWidth',1,...
   'MarkerSize',6,...
   'MarkerEdgeColor','#4DBEEE',...
   'MarkerFaceColor','#4DBEEE');
plot(vector_n1v_WL,ErrorRelativo_Vout_nOpt_Log, '--k',...
   'LineWidth',1);
hold off
grid minor
xlabel('X + Ym + n','FontSize',12);
ylabel('log_{10}(e)','FontSize',12);
title('Modelo en Coma Fija, V_{out}','FontSize',12,'FontWeight','bold');
legend('n1v','n1i','n2v','n2i','n3c','n3l','nOpt');
```

D Función de cálculo del error relativo en Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Funcion: calculo_error
%
%   Descripcion: Funcion para el error relativo de iL y Vout para cada tipo
%   de señal
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Autor: Sandra Peces Martin
%   Fecha: 20 JULIO 2021
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Entradas:
%       nXYdata : cell con los datos de iL y Vout para cada simulacion
%       num_files: numero de simulaciones, para este caso 21
%       iLReal: valores de iL del Modelo Real
%       VoutReal: valores de Vout del Modelo Real
%   Salidas:
%       errorRelativo_iL_Log: vector del error realtivo para iL para cada
%       tipo de señal
%       errorRelativo_Vout_Log: vector del error realtivo para Vout para cada
%       tipo de señal
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [errorRelativo_iL_Log, errorRelativo_Vout_Log] = calculo_error(nXYdata,
num_files, iLReal,VoutReal)

    n_samples = length(iLReal);
    IL_tipico = 1.15; %A
    Vout_tipico = 48; %V

    for i = 1:num_files
      % Calculo del error relativo de iL
      MAE_iL = sum(abs(nXYdata{1,i}(:,1) - iLReal)); %MAE, Error Absoluto Medio
      MAE_optimizado_iL(i,1) = MAE_iL/n_samples;
      errorRelativo_iL(i,1)= MAE_optimizado_iL(i)/IL_tipico;

      % Calculo del error relativo de Vout
      MAE_Vout = sum(abs(nXYdata{1,i}(:,2) - VoutReal)); %MAE, Error Absoluto Medio
      MAE_optimizado_Vout(i,1) = MAE_Vout/n_samples;
      errorRelativo_Vout(i,1)= MAE_optimizado_Vout(i)/Vout_tipico;

    end
    errorRelativo_iL_Log = flipud(log10(errorRelativo_iL));
    errorRelativo_Vout_Log = flipud(log10(errorRelativo_Vout));
```

end

E Función de cálculo de la WL en Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Funcion: vector_WL
%
%   Descripcion: Funcion para calcular la WL de cada tipo de señal en
%   funcion de la variacion de los bits extra n1i, n1v, etc
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Autor: Sandra Peces Martin
%   Fecha: 3 AGOSTO 2021
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Entradas:
%       vector_nxx_WL : vector con el cáculo de la WL para cada n1i, n1v...
%       X: bits de la parte entera de cada tipo de señal (previamente calculado)
%       Ym: bits de la parte fraccional de cada tipo de señal (previamente calculado)
%   Salidas:
%       vector : vector de WL para cada n1i...
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [vector] = vector_WL(X, Ym)

   nxx = -10:2:30; % Se han realizado las simulaciones para n = {-10 a 30} de 2 en 2

   for i = 1:length(nxx)
     % Calculo de la WL
     vector(i,1) = X+Ym+nxx(i);
   end

end
```