UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Máster en Ingeniería de Telecomunicación

TRABAJO FIN DE MÁSTER

EMULACIÓN HIL DE CONVERTIDORES CONMUTADOS UTILIZANDO MODELOS PARAMETRIZABLES

Sandra Jurado Jabonero

Junio 2016

EMULACIÓN HIL DE CONVERTIDORES CONMUTADOS UTILIZANDO MODELOS PARAMETRIZABLES

AUTORA: Sandra Jurado Jabonero

TUTOR: Alberto Sánchez González

PONENTE: Ángel de Castro Martín

Trabajo realizado en el grupo

HCTLab

Hardware and Control Technology Laboratory

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Junio 2016

UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

Agradecimientos

En primer lugar agradecer a mi familia: mis padres y mi hermana, su apoyo y comprensión. También agradecer a mi tutor hacer esto posible, gracias por esas jornadas adaptadas a mi horario sin las que no podría estar hoy aquí.

Gracias.

Sandra Jurado Jabonero

Junio 2016

RESUMEN

Las ventajas del uso de controladores digitales frente a analógicos están fuera de toda duda. Debido a ello y gracias a la reducción de precio de los primeros, los controladores digitales cada vez están más presentes en convertidores de potencia. Como cualquier otro producto de ingeniería, un controlador debe ser probado antes de pasar a la etapa de producción. Sin embargo, debido a la gran cantidad de energía que se maneja, no es trivial hacer pruebas con un convertidor real, por lo que es obligado realizar pruebas de simulación antes de probar el controlador con el sistema final.

Una de las opciones que se utiliza habitualmente son las simulaciones mixtas; llamadas así debido a que hay una componente analógica (convertidor de potencia) y una componente digital (controlador). Sin embargo, estas simulaciones suelen ser muy costosas en términos de tiempo.

Para aumentar la rapidez de la etapa de simulación, se suelen emplear técnicas HIL (*Hardware In the Loop*). Esta tecnología se basa en la creación de un modelo digitalizado del circuito analógico y ejecutar dicho modelo en hardware. Combinando esta metodología con el uso de FPGAs, se consiguen realizar simulaciones en tiempo real con tiempos de integración en torno a las decenas de nanosegundos.

En este caso se ha digitalizado un convertidor conmutado Flyback con el fin de poder aplicar la metodología HIL sobre este modelo. Se han analizado las diferentes aritméticas que pueden usarse y se ha realizado una comparativa entre coma flotante y coma fija. Debido a que la coma fija es mucho más rápida y conlleva menos recursos, se ha propuesto utilizar esta aritmética, pero de forma parametrizable. De esa forma, el modelo se adaptará a las condiciones de simulación que el usuario requiera y se obtendrá un modelo generalizado manteniendo el requisito de tiempo real con bajo tiempo de integración.

La veracidad del empleo de la tecnología HIL para el testeo de controladores sobre convertidores conmutados esta abalada en muchos estudios. En este caso se comprueba el modelo concreto parametrizable que se ha diseñado, frente al mismo realizado con Simulink (herramienta de diseño de Matlab) con el fin de demostrar que este diseño del Flyback es válido para diversos valores del mismo. Por lo tanto, este resultado podría formar parte de una librería de convertidores conmutados con el fin de acceder a esta simulación de forma rápida.

ABSTRACT

The advantages of digital control over analog are taken as a true theory. For this reason and because of the price of the first one, which are cheaper, digital controllers are taking place in switching power converters. As another engineering products, some verification is needed to check the correct work of that particular controller before its production. However, because of the high energy which is managed in this kind of devices, it is not safe to do real verifications over the real converter. For this reason, it is necessary to test the control before its implementation with the model.

One of the typical option is mixed simulations, which name comes from the two different components: analog (switching converters) and digital (control). Nevertheless, this kind of simulations use to be so expensive.

In this Project, Flyback switching converter model is digitalized in order to apply HIL methodology over this model. Different arithmetics have been analyzed providing a comparison between fixed and floating point. Fixed point uses less resources and it is faster than floating point, so we have decided to use this arithmetic to implement a parametrical model. This design let the users custom the model to apply their own conditions to reach their own simulations.

HIL option to test controls over switching converters is widely accepted in lots of studies. In this project, the design model is tested versus a Simulink model, in order to prove that this design is veracious. Furthermore, the model proposed in this project may be one model of a switching converter models library.

PALABRAS CLAVE

Convertidor Flyback, fuente conmutada, planta analógica, controlador digital, emulación, simulación, hardware en lazo cerrado (HIL), digitalización, convertidor CC/CC, coma fija, simulador mixto, Simulink.

KEYWORDS

Flyback converter, switching source, analog plant, digital controller, emulation, simulation, Hardware-In-The-Loop (HIL), digitizing, DC/DC converter, fixed point, mixed simulator, Simulink.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE ABREVIATURAS

AC (Alternating current)

ADC (Analog-to-Digital Converter)

ADMS (Analog-Digital Mixed Signal)

CC (Corriente Continua) o DC (Direct Current)

CCM (Continuous Conduction Mode)

DAC (*Digital to Analog Converter*)

DCM (Discontinuous Conduction Mode)

DSP (Digital Signal Processor)

FPGA (Field-Programmable Gate Array)

HDL (Hardware Description Language)

HIL (Hardware In-the-Loop)

IGBT (Insulated Gate Bipolar Transistor)

MOSFET (Metal–oxide–semiconductor field-effect transistor)

PID (Proportional Integral Derivative)

PWM (Pulse-Width Modulation)

VHDL (VHSIC Hardware Description Language)

1 INTRODUCCIÓN

La electrónica de potencia es un área de la electrónica de gran importancia debido a la presencia que tiene en el resto de las áreas. Con electrónica de potencia hacemos referencia a los dispositivos, que permiten adecuar unos flujos de tensión y corriente a otros flujos necesarios para el correcto funcionamiento de los elementos que los precisen. Por lo tanto, esta parte de la electrónica afecta al resto de dispositivos eléctricos, ya que todos ellos requieren de unas condiciones de alimentación concretas.

Históricamente el control de la electricidad se ha tratado de forma *ruda* disipando la parte sobrante de la energía que no se emplea en forma de calor como ocurre con los reguladores lineales, Figura 1. Actualmente existen otras maneras de manejar esa energía y conseguir los mismos resultados con menos pérdidas. En el grupo de dispositivos que permiten esta mejora se encuentran los convertidores conmutados, que están formados por elementos no disipativos con lo que se permite conseguir una alta eficiencia. Estos convertidores se basan en la conmutación de un elemento interruptor (MOSFET, IGBT, Tiristores, etc.).

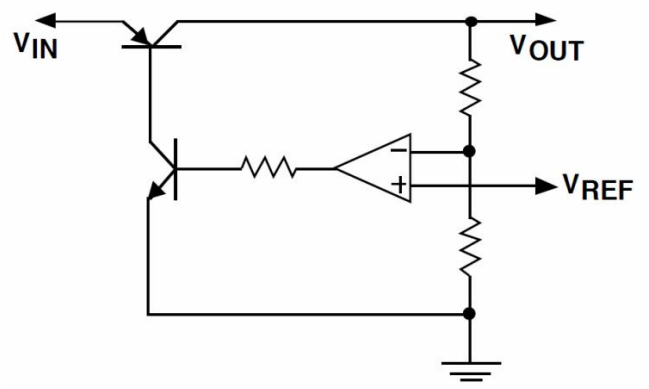


Figura 1. Esquema de un regulador lineal

Los convertidores conmutados son básicamente diseños circuitales en los que elementos típicos como bobinas y condensadores, que permiten almacenar y liberar energía, se acompañan de conmutadores, que varían su estado para así cambiar la forma del circuito y por lo tanto su funcionamiento, con el fin de regular el valor de la tensión y de la corriente a la salida del diseño. Este modelo debe ser acompañado de un conjunto de señales que manipulen el estado de los interruptores. La apertura y cierre de los conmutadores puede ser fija, es decir, que se mantengan abiertos un periodo de tiempo y cerrados durante el resto del ciclo de conmutación. Esto nos da una salida del circuito poco estable y no controlada. Para mantener una salida apropiada del circuito,

estas señales de control dependerán de los valores de tensión y corriente del circuito en cada instante. Con estos datos varían el estado de cierre o de apertura de los conmutadores de una forma variable con el fin de obtener los flujos de potencia adecuados y controlados. Es decir, en lugar de mantener el interruptor abierto la mitad del ciclo de conmutación y cerrado la otra mitad, se mantendrá cerrado durante más o menos tiempo y abierto el resto del ciclo de conmutación. Este tiempo de apertura será variable en función de si la salida medida en un instante está por encima o por debajo de la esperada.

El controlador encargado de dar valores a estas señales de control puede ser analógico. Este tipo de controlador se implementa con elementos circuitales como resistencias, amplificadores o condensadores que realizan filtros para modelar la respuesta en frecuencia del sistema. Estos controladores pese a tener gran ancho de banda y elevada resolución (ya que se trata de elementos analógicos que pueden tomar infinitos valores) tienen desventajas muy importantes como el envejecimiento de sus componentes o la dificultad de réplica.

El controlador también puede ser digital. Esta es la tendencia de los últimos años ya que los controles digitales se pueden programar en microprocesadores con lo que se permite variar este control dependiendo de los requisitos. Estos controladores digitales, pese a requerir un convertidor DAC (*Digital to Analog Converter*) o ADC (*Analog to Digital Converter*) para convertir las señales analógicas de la planta en digitales, legibles para el control y viceversa, ocupan menos espacio que sus homólogos analógicos. Por otro lado aunque su dificultad de diseño es mayor, su comportamiento es más preciso y permiten implementar algoritmos complejos.

Un problema importante de emplear los controladores digitales es simular este sistema analógico-digital (también conocido como sistema mixto) que forma junto a la planta analógica que controla. Especialmente en la etapa de diseño del control, donde se deben realizar numerosas pruebas hasta asegurar que el control a emplear es el apropiado para una planta dada.

En el caso de los convertidores conmutados, debido a que se trata de sistemas de potencia, es peligroso manipularlos directamente, especialmente en las primeras pruebas o cuando se está probando los límites de estabilidad del sistema en lazo cerrado. Esto es debido a que un fallo en el control puede generar corrientes y tensiones muy elevadas indeseadas. Para evitar daños en el circuito que puedan acarrear su reconstrucción o que puedan causar daños físicos a las personas que lo manipulen, se está empezando a emplear métodos de verificación HIL (Hardware In the Loop). Estos métodos consisten en la digitalización del modelo de planta a controlar y simularlo conjuntamente con el control digital. Otra de las ventajas de la verificación HIL es que permite separar las etapas de control y construcción del propio convertidor, por lo que un diseñador del regulador de control puede empezar a realizar pruebas de su regulador incluso sin tener finalizado el convertidor final.

Existen otros métodos de simulación mixta de distintos precios y prestaciones. En general, respecto a otros métodos de simulación mixta, las ventajas de un sistema HIL son principalmente: la facilidad de probar directamente el control final que se empleará junto al modelo del circuito y la rapidez de simulación que se permite con FPGAs (*Field-Programmable Gate Array*).

Por lo tanto, esta técnica se emplea en la fase de verificación de controles digitales de sistemas analógicos de todo tipo ya sean eléctricos, hidráulicos o mecánicos e independientemente, en sistemas más o menos complejos. Con este sistema se evitan daños físicos que en el caso de la electricidad puede ser una descarga indeseada o en el caso de sistemas mecánicos puede ser la apertura de una trampilla antes de tiempo. Ambos errores pueden causar daños materiales y/o personales. Además, con esta tecnología HIL la verificación del control junto a la planta digitalizada se puede hacer tantas veces como sea necesario para conseguir sistemas más robustos, de mayor calidad y sin invertir más recursos.

Para este trabajo en concreto se realizará el modelo de planta de un convertidor Flyback. Este convertidor es capaz tanto de aumentar como de disminuir la tensión de la salida respecto a la de la entrada, por lo que se conoce como un sistema reductor/elevador, pero con la ventaja de que este sistema tiene aisladas la entrada y la salida. Como se ve en la Figura 2, el transformador actúa no solo para proporcionar energía, sino también, como aislamiento eléctrico entre la entrada y la salida. En definitiva el convertidor Flyback no es más que un convertidor reductor/elevador (*buck/boost*) en el que se añade una etapa de aislamiento.

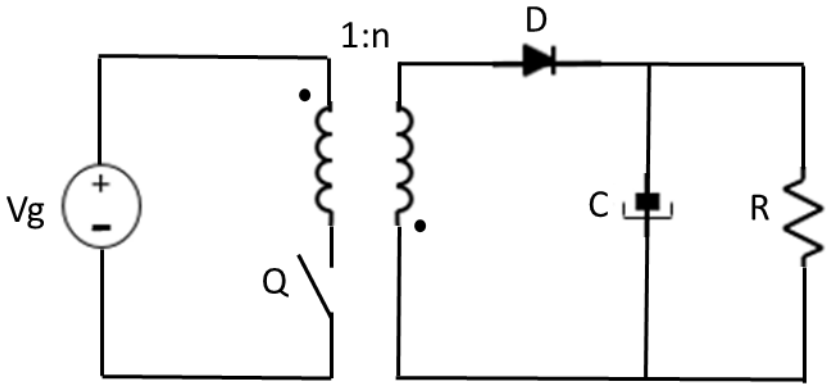


Figura 2. Esquema eléctrico del convertidor flyback

Siguiendo el planteamiento HIL, después de hacer el modelo matemático del Flyback y digitalizar el circuito en VHDL, nos encontramos con una réplica de la planta. Esta réplica tiene un beneficio muy importante respecto a su patrón analógico para la etapa de verificación: mientras que un fallo en el control sobre el circuito tradicional puede acarrear daños tan graves que hasta pueden implicar la reconstrucción del mismo, un

fallo en el modelo digitalizado solo implica la salida de un flujo de bits distinto de lo esperado.

Para representar fielmente el modelo que estamos digitalizando se puede implementar el diseño en ordenadores [2]. Para reducir el tiempo de paso de simulación y, por tanto, aumentar la precisión de los modelos, en la literatura se encuentran sistemas HIL basados en FPGA, que permiten implementar en hardware el modelo aprovechando las ventajas del uso de hardware [2]. De esa forma, se implementa el diseño en una FPGA y con convertidores DAC obtenemos unas salidas idénticas a las que tiene nuestro circuito analógico. Por lo tanto el modelo de planta sobre el que se prueba el control es una réplica del original con la ventaja de evitar daños en el circuito inicial. Gracias a este paso se imitaría fielmente el modelo de planta, al que se puede enviar desde el control las mismas señales que se enviarían inicialmente al modelo analógico.

Para avanzar más en la creación de modelos digitalizados, no solo se va a crear un modelo HIL de un convertidor Flyback con unos valores constantes de tensiones/ y corrientes, sino que se va a extender a un patrón generalizado, en el que pueda parametrizarse el modelo de forma que cualquier rango de tensión o corriente pueda ser simulado. Esto quiere decir que se diseñará el circuito digitalizado con la posibilidad de variar los valores de sus elementos y características de una forma dinámica, hasta asemejarlo al modelo analógico que se quiere imitar, sin necesidad de rehacer el código que lo define, ni modificar el *bitstream* en el caso de que se haya implementado en FPGA.

Si realizamos esto para los convertidores conmutados con topologías más conocidas, conseguimos una librería de elementos digitalizados que se pueden simular junto a cualquier control digital gracias a la tecnología HIL en tiempo real y con un paso de simulación muy bajo, lo que permitirá una precisión mayor y soportará reguladores que tienen una frecuencia de conmutación elevada.

La propuesta presentada se someterá a pruebas tanto de precisión como de velocidad y recursos utilizados. En cuanto a la precisión, la veracidad de estos modelos ha sido demostrada en estudios anteriores [1] pero en este caso se va a comparar nuevamente con el modelo del circuito implementado con la herramienta Simulink de Matlab que goza de confianza en el mundo de la electrónica.

2 ESTADO DEL ARTE

En la no tan reciente historia de la electrónica se ha sufrido un cambio de la maquinaria analógica a la digital. La facilidad de uso, de cambio y de duplicidad, entre otras características, hace de esta evolución un punto y aparte en la fabricación de dispositivos electrónicos. Pese a los beneficios indudables de estos sistemas hay modelos analógicos importantes en nuestro día a día como motores o convertidores de potencia entre muchos otros que se encuentran en su modelo analógico.

Para el correcto funcionamiento de algunos sistemas es necesario tener un control que efectúe los cambios oportunos con el fin de conseguir el funcionamiento buscado. Un ejemplo de sistema popularmente conocido que requiere de un control es la calefacción de una estancia. Esta planta, temperatura de la estancia, se regula mediante un termostato que permite parar la caldera cuando se alcanza la temperatura adecuada y la enciende nuevamente cuando la temperatura desciende. Al igual que ocurre con la temperatura, existen otros sistemas como los convertidores de potencia que necesitan de un control para mantener unos requisitos de tensión y corriente adecuados en cada momento.

Los convertidores de potencia transforman y controlan unos flujos de entrada de corriente y tensión para ofrecer unos valores oportunos a su salida. Con este nombre es como típicamente se conocen a los encargados de la transformación CC a CC. Esta transformación puede darse como un sistema en sí mismo aislado que mantenga unas condiciones. Un ejemplo de este tipo sería un cargador de teléfono móvil en un coche, que transforma la alimentación del mechero del coche en el flujo adecuado para el dispositivo a cargar. Pero los convertidores también se pueden encontrar como parte de los sistemas de transformación CC a AC o AC a CC [4].

La transformación CC/AC ha cobrado importancia en los últimos años debido a la energía fotovoltaica, ya que sus células generan CC que es necesario transformar en CA para conectarse a la red de transporte de energía. En esta transformación el paso CC/CA se realiza con un inversor. Por otro lado, en la transformación AC/CC se emplean rectificadores, este tipo de transformación está presente en todos los cargadores de dispositivos electrónicos actuales que se conectan a la red eléctrica de los hogares. En ambos casos, los convertidores son necesarios para mantener los niveles deseados a la salida de ambos sistemas. En la Figura 3 se puede ver el esquema típico de una fuente de alimentación AC/CC que transforma la corriente alterna gracias a un transformador, cuya salida pasa por un rectificador que elimina los ciclos negativos. Por último, estos pasan por un filtro para alcanzar una corriente relativamente continua. El proceso podría acabar aquí, pero las condiciones resultantes no serían óptimas por lo que es necesario

el conversor CC/CC, que es el encargado de que el flujo que llega al usuario final sea el requerido por el mismo.

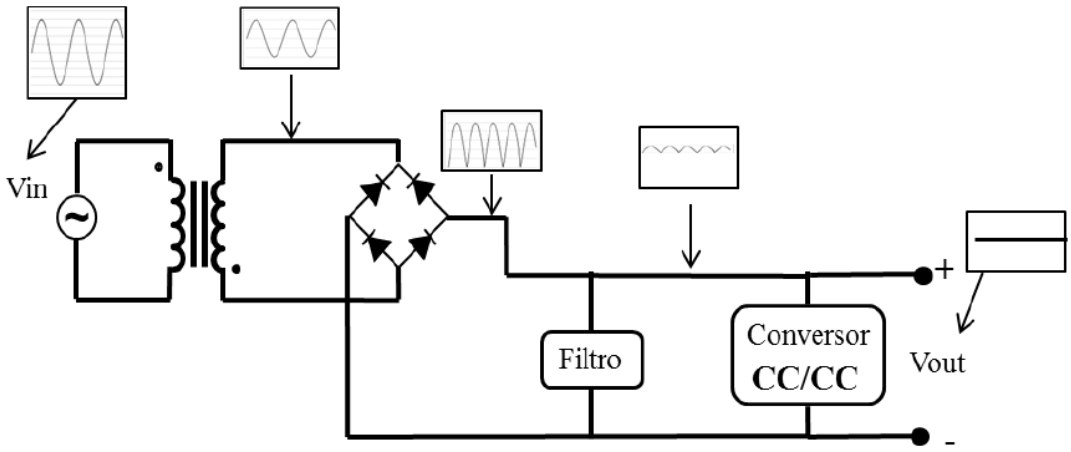


Figura 3: Modelo completo de una fuente de alimentación

La transformación de potencia se puede llevar a cabo, tanto por reguladores lineales como por convertidores conmutados. Los reguladores lineales basan su funcionamiento en la disipación de energía por medio de una resistencia variable, mientras que los convertidores conmutados se basan en el uso de interruptores para permitir o negar el paso de energía y así hacer que varíe a su salida.

El uso de convertidores conmutados frente a reguladores lineales se debe a que los primeros tienen mayor eficiencia debido a que los segundos basan su comportamiento en la disipación de la energía sobrante en forma de calor, por lo que para salidas mayores a 40 vatios no se suele contemplar el uso de estos reguladores. Como característica adicional, los convertidores conmutados pueden funcionar tanto de elevadores de potencia como de reductores, lo que los hace aún más útiles.

Los convertidores pueden conmutar de diferentes maneras [4]: de forma natural (convertidores de frecuencia de línea), donde la propia alimentación presente a un lado del convertidor desconecta o conecta un dispositivo semiconductor; y de forma forzada (convertidores de conmutación) que requieren unas señales de control para la apertura y cierre de sus interruptores. Así, estos convertidores conmutados de potencia son ejemplo de sistema controlado, que necesita de un controlador que genere unas señales para manejar sus interruptores y lograr la salida requerida por el usuario.

Este control puede ser analógico o digital. Con el paulatino abaratamiento de costes, cada vez se encuentran más controladores digitales debido que sus ventajas pesan más que sus inconvenientes, a diferencia de lo que ocurre en el caso de los analógicos [3].

Los controladores analógicos pueden implementarse con elementos circuitales, que posicionados de forma adecuada, pueden generar las señales apropiadas para manipular el sistema que controlan. Como ya se ha comentado en la introducción, existen algunas ventajas de este modelo frente a los controladores digitales. Estos controladores son de alta resolución, elevado ancho de banda y son muy sensible a cambios temporales. Además son típicamente sencillos de diseñar y verificar. Entre sus desventajas cabe destacar el envejecimiento de los componentes, las variaciones con las condiciones del entorno, y además su tamaño es más elevado debido a que se trata de un montaje con elementos discretos.

Por su parte los controladores digitales suelen implementarse con microcontroladores o microprocesadores que necesitan de conversores DAC y ADC para comunicarse con la planta, esto puede añadir retardos que deben ser contemplados [14]. También, aunque puede ser suficiente, su resolución no es tan amplia como en los modelos analógicos. Su dificultad de diseño es mayor y requieren de procesadores de altas prestaciones para su implementación. Entre sus ventajas se puede destacar que son de diseño reprogramable que permite su modificación sin variar el hardware. Además permiten implementar algoritmos complejos como controles no lineales, predictivos o adaptativos. Su tiempo de diseño es menor, en especial cuando hablamos de reproducibilidad del modelo. Este sistema aumenta la fiabilidad [14], es más estable frente a cambios en el entorno y atenúa la sensibilidad al ruido. Como en casi todo, aunque las características nombradas del controlador digital son muy atractivas frente a las del analógico, existen casos en los que es más conveniente el uso de un controlador analógico.

Existen distintos convertidores en el mercado que se pueden clasificar en dos tipos: convertidores no aislados y aislados. Los convertidores no aislados no disponen de la etapa de aislamiento eléctrico, los más conocidos son: reductor (*buck*), elevador (*boost*), convertidor reductor/elevador (*buck/boost*), convertidor *Cúk* y convertidor de puente completo (*full-bridge*). Los convertidores elevador y reductor poseen las topologías básicas en las que se apoyan el resto de modelos. Por otro lado están los convertidores aislados entre los que se encuentran el convertidor Flyback, el convertidor directo, y el *Ćuk* aislado. Estos convertidores provienen de los convertidores no aislados a los que se añade una etapa de aislamiento. Así el Flyback se deriva del convertidor elevador/reductor, mientras que el convertidor directo se deriva del reductor.

Los convertidores conmutados varían su comportamiento en relación con el control que se emplee en su funcionamiento. Por lo tanto, en la verificación del diseño de los convertidores se puede tomar una señal modulada por ancho de pulso fijo para verificar el funcionamiento del sistema en sí mismo, que se puede comparar con el resultado de resolver el circuito matemáticamente. Si se emplea un control, esta salida podría ser corregida hacia los flujos requeridos independientemente de si estos son solución o no del circuito. Este tipo de verificación del sistema se denomina en lazo abierto. Cuando un control está variando la señal modulada por ancho de pulso (PWM o *Pulse Width*

Modulation) con la intención de modelar la señal a la salida en función de sus valores actuales se dice que está funcionando en lazo cerrado.

Cuando se realiza un regulador debe ser verificado antes de su implementación. En el caso del regulador clásico, por ejemplo un PID (controlador de acción proporcional integral y derivativa) se puede hacer en Sisotool (herramienta de Matlab), donde se puede implementar el regulador para una planta definida matemáticamente, sobre la que se puede hacer pruebas de estabilidad. Una vez que funciona correctamente en esta herramienta se implementaría el control que se ha diseñado. Este control se quiere probar junto al convertidor real. En este paso si el modelo verificado en Sisotool muestra errores tras su implementación podría estropear el convertidor. Para esto se emplea HIL, para realizar un paso intermedio entre la implementación del control y la prueba real.

Extendiendo esta idea, una vez que el control ha sido correctamente diseñado, una forma intuitiva de probarlo es realizando el montaje natural tal y como se va a utilizar tras esta fase de pruebas. Esta forma de verificarlo trae consigo varios problemas: al tratarse de convertidores de potencia un error en el control podría estropear el convertidor si la corriente se dispara fuera de los límites soportados por los elementos que lo forman. Esto podría suponer la reconstrucción del convertidor desde cero debido a los daños ocasionados por emplear un control inadecuado. Por otro lado, una subida de corriente podría causar, no solo daños materiales y por lo tanto económicos, sino también daños personales durante su manipulación.

Para evitar que este tipo de desastres ocurran existen numerosos estudios sobre las distintas posibilidades de simulación que se ofertan para sistemas mixtos [1]. En [5] se realizan varias simulaciones, en la primera se emplea un sistema con Matlab/Simulink que permite una simulación rápida y un sistema de verificación para resoluciones y retardos, pero no se corresponde completamente con el modelo del sistema a verificar, sino una aproximación con Simulink. En la segunda simulación se emplea un simulador mixto, Cadence Spectre Virtuoso, para un convertidor analógico definido en Spice y un controlador descrito en HDL (Hardware Description Language), para este caso se requiere mucho tiempo de simulación. La última propuesta es igual que la anterior pero empleando modelos genéricos de Spice, lo que permite mayor aceleración en las simulaciones. De igual modo estas simulaciones mixtas necesitan varias horas por ciclo de conmutación. Posteriormente aparece la herramienta de mentor ADMS (Analog-Digital Mixed Signal) que permite la simulación de modelos analógicos y digitales de forma conjunta. El problema de esta herramienta recae como ha ocurrido en opciones anteriores en el elevado tiempo requerido para la simulación.

Ya que existen pocos simuladores mixtos o siendo muy costosos los que se comercializan, existen otras propuestas que emplean dos simuladores [6], uno analógico, PSIM, y otro digital, ModelSim. El problema es que para llevar a cabo estas simulaciones conjuntamente, es necesario que el diseñador cree un una interfaz para

enlazar ambos programas. Para esto puede emplear un lenguaje como C o C++ y se debe tener en cuenta la sincronización de datos, los flujos de realimentación, etc. Hay simuladores mixtos más actuales como Gecko [11]. Este simulador permite realizar el diseño esquemático que será convertido a código Java y simulado junto al control en el mismo lenguaje. En el caso de tratarse de un control embebido en un microprocesador estará diseñado en un lenguaje HDL por lo que el control simulado no será el mismo.

Otra de las modalidades de simulación consiste en modelar la planta en HDL a partir de las ecuaciones en diferencias que modelan el circuito. Así en [7] se compara el uso de Spice, VHDL y VHDL-AMS (consiste en una extensión de VHDL para definir señales digitales y analógicas). Donde se concreta que empleando Spice como referencia, VHDL-AMS es 2,18 veces más rápido y la simulación con VHDL es 3,18 veces más rápida. Esta comparación entre VHDL-AMS y VHDL también se comprueba en [8] donde se consiguen tiempos de simulación 10 veces mayores para VHDL que para VHDL-AMS.

Similar a este último caso aparece el sistema HIL (Hardware In-The-Loop), este sistema consiste en digitalizar la planta analógica para poder implementarlo sobre un ordenador, microprocesador o FPGA, y realizar las pruebas en lazo cerrado junto al control digital real. El controlador que esta ya implementado de manera digital se simula conjuntamente con la planta, ya sea en un sistema único (todo digital sin convertidores) o con dos dispositivos diferenciados que emulen el funcionamiento de la planta analógica (para lo que se incluirían convertidores DAC) y el control digital junto a los convertidores necesarios en la puesta a punto real.

Esta metodología HIL está abalada por numerosos estudios [9] y se emplea no solo en el campo de la automatización y del control sino también en otras áreas de la ingeniería. Así en empresas como Controllab [10] vemos el uso de esta tecnología en otras áreas. Por ejemplo, la herramienta WinMOD tiene varios modelos HIL que permiten la construcción de modelos de planta a través de componentes que se basan en bloques funcionales. Tras un periodo de diseño con la herramienta, se pueden llegar a modelar las distintas plantas de una industria de forma digital. En la Figura 4 se puede ver claramente el empleo del sistema HIL en la industria real.

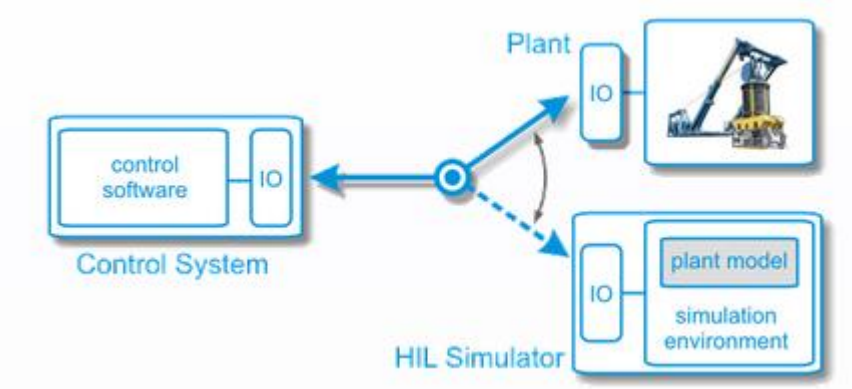


Figura 4: Imagen extraida de [10]

Otro ejemplo de empresa que invierte en esta tecnología es Typhoon HIL [12], esta empresa se ha especializado en la verificación de convertidores con sus respectivos controles. La herramienta que comercializan es capaz de simular un conjunto de sistemas complejos gracias a sus numerosas y completas bibliotecas que permiten realizar este trabajo de una forma rápida y sencilla. Por ejemplo, se puede usar para simular un *grid* (red eléctrica) que consta de generadores de energía, sistemas de transporte, etc. Este tipo de sistemas se caracterizan por su complejidad pero, a la vez, por sus bajos requerimientos en cuanto al paso de integración necesario para obtener simulaciones precisas. Esto es debido a que la conmutación y dinámica de este tipo de sistemas está en torno a las decenas, centenas o kilohertzios, mientras que la conmutación de un convertidor de potencia sencillo, como puede ser un Flyback, puede realizarse a centenas de kilohertzios o incluso mega hertzios. Por tanto, aunque resulte sorprendente, herramientas tan potentes de HIL no pueden ofrecer resultados óptimos para simular convertidores sencillos regulados a altas frecuencias. Además, el gran inconveniente de este sistema es su elevado precio que alcanza las decenas de miles de euros.

En este punto la intención de este trabajo toma forma, se va tratar de realizar un modelo de planta de un convertidor que pueda ser implementado en una FPGA. Este modelo será genérico y por lo tanto se podrán modificar los valores de los elementos discretos que lo componen. Esta planta podrá ser entonces simulada junto a un control que puede estar en otro microprocesador o en el mismo para así verificar el diseño del sistema conjunto.

Para realizar los modelos y poder simularlos conjuntamente de forma rápida es necesario controlar las dimensiones de las señales para poder ejecutarlo sobre un microprocesador. En este caso se va a emplear el lenguaje hardware VHDL para poder ejecutar el diseño sobre una FPGA. La longitud de las señales que se empleen es importante tanto para los recursos que se consumen como para el tiempo de simulación que se requiere [2].

En la librería VHDL encontramos varias opciones de datos como se ve en [15] como coma fija (‘*sfixed’*), real (‘*real’*), coma flotante (‘*float’*) o vector binario (‘*Standard Logic Vector’*). Las señales *real* no se pueden ejecutar sobre FPGA pero es habitualmente usado por la facilidad de implementar un modelo en este estándar y debido a su gran resolución numérica, ya que se trata de un modelo en coma flotante de 64 bits. Como se ha comentado, los diseños con estas señales son más sencillos de llevar a cabo ya que no hay que tener en cuenta tamaños de las señales al operar ni desbordamientos. Por estos motivos, un diseño elaborado con señales *real* puede servir como referencia para comparar otros modelos sintetizables en la FPGA y comprobar su resolución frente al modelo *real*.

La ventaja de la coma flotante es que la posición de la coma de las variables se adapta de forma automática, mientras que las señales en coma fija no pueden realizar dicha adaptación. Por tanto, en los modelos realizados en coma fija es necesario definir los tamaños de las señales dependiendo de los valores que tomarán las mismas. Esto imposibilita la generalización de un modelo genérico que puede tomar diferentes datos y en el que los valores de las señales serán muy diferentes dependiendo de estos valores. Sin embargo, la coma fija presenta otra ventaja que es la simplicidad del hardware generado cuando se realiza una operación, en comparación con la complejidad del mismo en el caso de la coma flotante. Esto permitirá que su frecuencia de implementación sea mayor y, por tanto, se pueda conseguir un paso de simulación menor manteniendo los requisitos de simulación en tiempo real.

En este Trabajo Fin de Máster se propone diseñar un modelo de un convertidor conmutado, apto para emulaciones HIL, basado en coma fija parametrizable. Esta nueva técnica permite aprovechar las ventajas de velocidad de la coma fija, pero permite adaptarse para almacenar diferentes rangos de valores como lo hace la coma flotante. En la literatura no se ha encontrado ninguna propuesta similar por lo que se trata de una aportación original de este Trabajo Fin de Máster.

3 DISEÑO DE MODELOS VHDL DE CONVERTIDORES CONMUTADOS

El primer paso de la digitalización consiste en comprender el funcionamiento del circuito analógico para así conseguir esa funcionalidad mediante el modelo digital. En este caso se va a estudiar el esquema del circuito Flyback (Figura 1). En primer lugar se va a realizar el diseño matemático del mismo y posteriormente se va a proceder a su implementación. Para la implementación se ha elegido utilizar una FPGA para poder obtener pasos de integración reducidos, y se ha elegido VHDL como lenguaje de descripción de hardware.

3.1. Modelo matemático de un Flyback

En primer lugar es necesario conocer las ecuaciones que definen los elementos que forman el convertidor. La función que define el condensador (1) y la ecuación de la bobina (2) relacionan la corriente que atraviesa estos elementos con la tensión que cae en sus bornes. Estas ecuaciones dan lugar a las ecuaciones de estado: la tensión en el condensador (v_c) y la corriente en la bobina (i_L). Ambos elementos se encuentran presentes en el circuito del Flyback (Figura 2), el condensador (C) se puede ver físicamente en la figura, mientras la bobina (L) se encuentra como parte del circuito equivalente del transformador (Figura 6). La tensión de salida (igual a la tensión del condensador) y la corriente de la bobina son variables de estado ya que no pueden variar de manera instantánea, sino que su valor en un instante depende del valor en el instante anterior, de forma que describen el estado del convertidor en un instante de tiempo.

$$i_c = C \cdot \frac{dv_c}{dt} \tag{1}$$

$$v_L = L \cdot \frac{di_L}{dt} \tag{2}$$

Para poder realizar la digitalización de este modelo es necesario aproximar las ecuaciones de estado a ecuaciones en diferencias. Estas ecuaciones en diferencias se comportan igual que las ecuaciones derivativas cuando el paso de integración es mínimo, es decir, cuando Δt (paso de integración o de simulación) se comporta como dt y tiende a 0:

$$v_c(k) = v_c(k-1) + i_c(k) \cdot \frac{C}{\Delta t} \tag{3}$$

$$i_L(k) = i_L(k-1) + v_L(k) \cdot \frac{L}{\Delta t} \tag{4}$$

Una vez que tenemos las variables de estado vamos a analizar el circuito en función de la actuación del conmutador. En el caso del Flyback existe un único interruptor que puede estar abierto o cerrado, pero también influye el estado del diodo que puede estar conduciendo (CCM, *Continuous Conduction Mode*) cuando la corriente atraviesa el diodo desde su ánodo hasta su cátodo o no conduciendo (DCM, *Discontinuous Conduction Mode*) cuando la corriente que atraviesa el diodo es nula. Estos son los estados del diodo ideal que se pueden ver en la Figura 5.

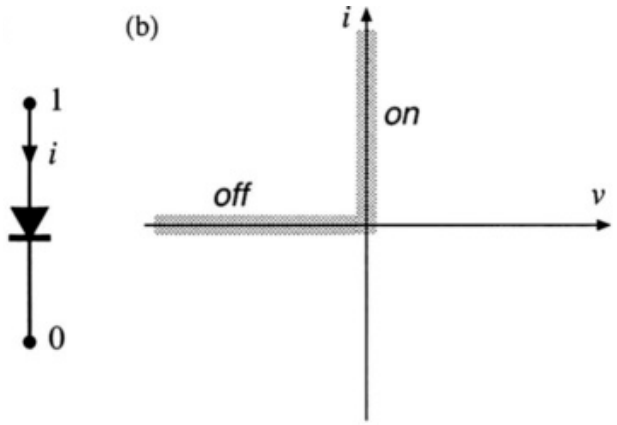


Figura 5: Diodo ideal

Con las ecuaciones en diferencias y los estados del circuito podemos definir las ecuaciones que rigen el comportamiento del circito para cada instante. Además de esto, en este circuito hay que tener en cuenta el funcionamiento del transformador inductivo que permite que el Flyback funcione como un circuito aislado. Para esto basta con contemplar la energía almacenada en el núcleo magnético del transformador como una inductancia de magnetización (L) como se ve en la Figura 7Figura 6. Así el funcionamiento del circuito se puede resumir como que la energía se almacena en L cuando el interruptor está cerrado, la corriente sube linealmente y el diodo de salida está polarizado inversamente. Por otro lado, cuando el interruptor se abre la energía se transfiere a la parte de salida del convertidor y, por tanto, a la carga ya que el diodo estará polarizado directamente.

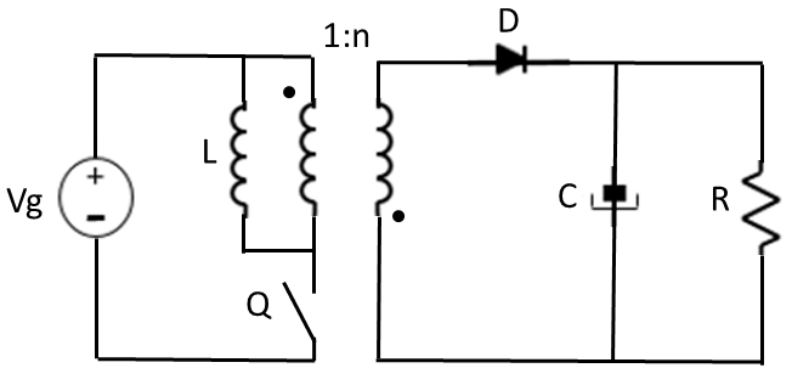


Figura 6: Circuito flyback contemplando la inductancia de magnetización.

Desarrollando la idea del apartado anterior, en el caso en el que el interruptor se encuentra cerrado, es decir conduciendo, el diodo queda polarizado en inversa y el circuito resultante se puede observar en la Figura 7. En este caso la energía generada por la alimentación se almacena en la bobina mientras que la almacenada en el condensador se libera hacia la carga.

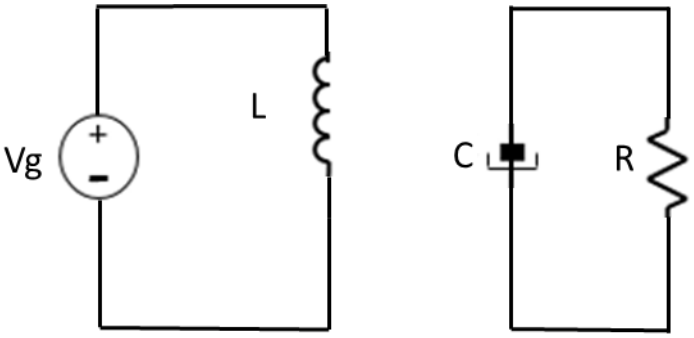


Figura 7: Circuito equivalente Flyback con Q=ON.

En este caso, la corriente que atraviesa la bobina (i_L) es igual a la corriente de entrada (i_i) y por lo tanto se puede deducir la ecuación (5). En cuanto a la tensión del condensador (v_c) se puede ver que es igual a la tensión de salida (v_o) y, por lo tanto, se cumple la ecuación (6), donde i_R es la corriente que atraviesa la carga de salida.

$$\Delta i_L = v_g \cdot \frac{\Delta t}{L} \tag{5}$$

$$\Delta v_c = -i_R \cdot \frac{\Delta t}{C} \tag{6}$$

En el caso en el que el interruptor esté apagado, el circuito que ilustra este estado se puede ver en la

Figura 8 en la que se observa cómo se descarga la bobina, lo que permite la carga del condensador y la transmisión de energía a la carga. La peculiaridad de este estado está en las opciones de trabajo del diodo que puede estar conduciendo o inversamente polarizado.

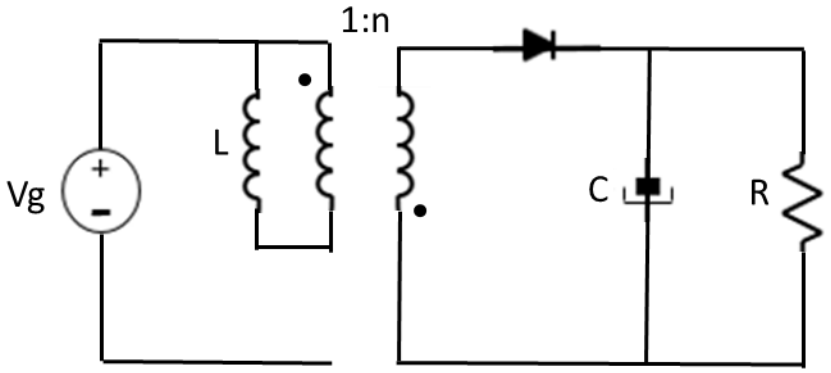


Figura 8: Circuito equivalente Flyback con el interruptor abierto y CCM.

En el caso en el que el diodo este conduciendo, las ecuaciones de estado de la bobina y el condensador son las mostradas en (7) y (8). En este caso la corriente almacenada en la bobina debe circular y atravesar el diodo polarizado en directa.

$$\Delta i_L = \frac{-v_o}{n} \cdot \frac{\Delta t}{L} \tag{7}$$

$$\Delta v_c = (\frac{i_i}{n} - i_R) \cdot \frac{\Delta t}{C} \tag{8}$$

Como un estado más del circuito vamos a contemplar la opción de que el interruptor esté cerrado y el diodo esté polarizado en inversa, por lo que nos encontramos con las ecuaciones (9) y (10). En este caso el circuito resultante es similar a Figura 7, pero sin que circule corriente por la bobina.

$$\Delta i_L = 0 \tag{9}$$

$$\Delta v_c = -i_R \cdot \frac{\Delta t}{C} \tag{10}$$

Tras deducir las ecuaciones que definen las de las variables de estado y las ecuaciones que definen los incrementos en ellas dependiendo del estado del circuito: el incremento de tensión en el condensador y el incremento de corriente de la bobina, podemos concluir que disponemos del modelo matemático del circuito de un convertidor Flyback. El resumen de este proceso se puede ver en la Tabla 1.

Q	ON	OFF	
Diodo	CCM	CCM	DCM
$v_c(k)$	$v_c(k-1) + \ -i_R \cdot \frac{C}{\Delta t}$	$v_c(k-1) + \ (\frac{i_i}{n} - i_R) \cdot \frac{C}{\Delta t}$	$v_c(k-1) + \ -i_R \cdot \frac{C}{\Delta t}$
$i_L(k)$	$i_L(k-1) + \ v_g \cdot \frac{L}{\Delta t}$	$i_L(k-1) + \ \frac{-v_o}{n} \cdot \frac{L}{\Delta t}$	$i_L(k-1) + \ 0 \cdot \frac{L}{\Delta t}$

Tabla 1: Resumen de ecuaciones que definen el Flyback

3.2. Condición de equilibrio.

Para tener una idea aproximada del funcionamiento del circuito en equilibrio, podemos utilizar las ecuaciones de estado teniendo en cuenta que el valor de las variables de estado durante un ciclo de conmutación es invariable. Esto es debido a que, si la corriente por ejemplo, subiera durante un ciclo de conmutación, entonces la tensión de salida también subiría y, por tanto, el sistema no estaría en equilibrio.

Así cuando el interruptor está cerrado, el valor de la tensión en la bobina es v_g, y la corriente sobre la misma aumenta. Cuando el interruptor se abre, el valor de la tensión en la bobina es la tensión a la salida en valor negativo y con el coeficiente *n*, por lo que la corriente disminuye. Esto se puede ver en la Figura 9.

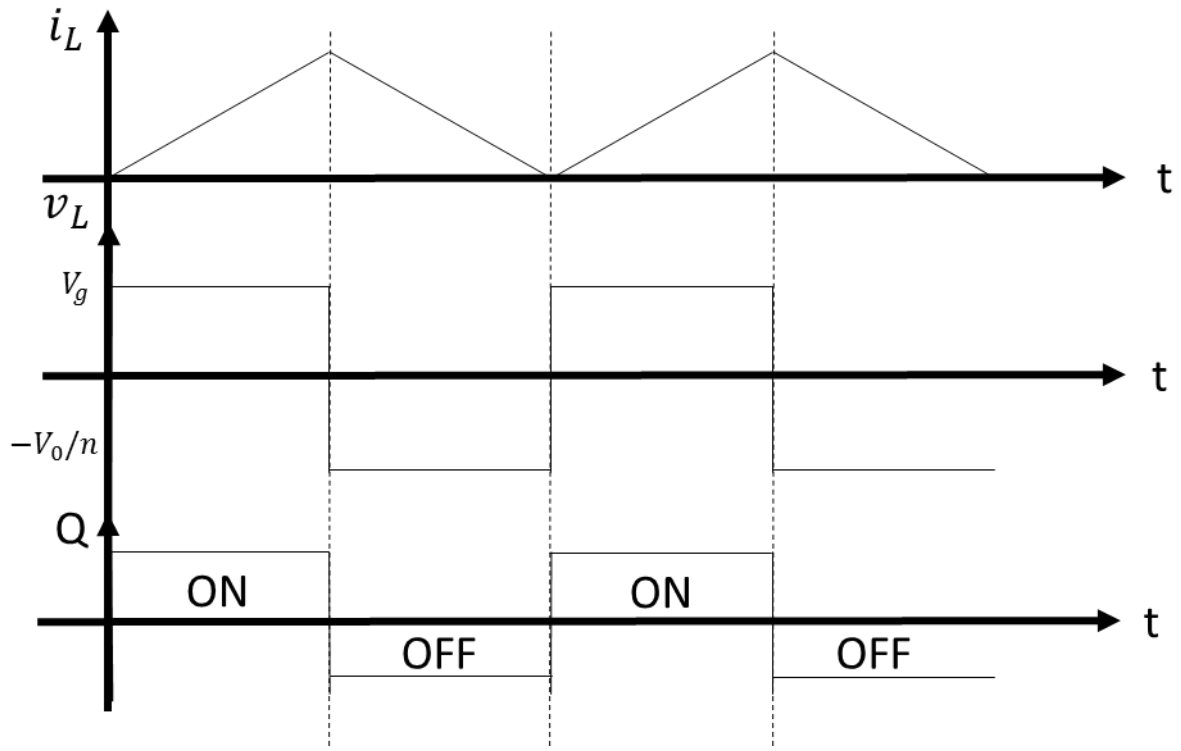


Figura 9: Formas de onda del Flyback

Así podemos realizar las operaciones oportunas para el estado de la bobina en el punto de equilibrio como se muestra en la ecuación (11), y en (12) siendo D el tiempo en el que permanece cerrado (‘*ON*’) el interruptor y D-1 el tiempo en el que se encuentra abierto (‘*OFF*’). Así en el punto de equilibrio el comportamiento del circuito lo rige la ecuación (13).

$$i_L(T = ON) + i_L(T = OFF) = 0 \tag{11}$$

$$0 = v_g \cdot \frac{\Delta t}{L} \cdot D + \frac{(-v_o)}{n} \cdot \frac{\Delta t}{L} \cdot (1 - D) \tag{12}$$

$$\frac{v_o}{v_g} = \frac{D}{n \cdot (1 - D)} \tag{13}$$

Con esta aproximación del circuito hemos extraído el comportamiento en régimen permanente de la topología Flyback dependiendo del ciclo de trabajo aplicado al interruptor del convertidor.

3.3. Aritmética de implementación.

Tras la elaboración de un modelo matemático viene el paso de implementarlo en VHDL. Lo primero que debemos hacer es elegir la aritmética que emplearemos para llevarlo a cabo. En VHDL tenemos varias opciones con distintos pros y contras: *real* (coma flotante de 64 bits no sintetizable), coma flotante sintetizable, coma fija (*fixed point*) y vectorial (*standard logic vector*).

- *Real* es un tipo de datos que se puede simular mediante una herramienta de simulación VHDL, como puede ser Modelsim, pero no es posible implementarlo en una FPGA. Entre las ventajas de esta aritmética destaca que se trata de un tipo de datos con mucha resolución, ya que se trata de un modelo en coma flotante de 64 bits. Además es de fácil manejo debido a que no es necesario asignar tamaños ni formatos espciales a las señales. En nuestro caso esta aritmética se empleará como patrón del resto, pero no será nuestro modelo final debido a la imposibilidad de implementación en FPGA.

- La coma flotante también puede ser sintetizada utilizando las bibliotecas necesarias, como puede ser *float* del estándar VHDL2008. Sin embargo, no todos los sintetizadores la soportan y, en cualquier caso, los recursos hardware necesarios para implementar operaciones en coma flotante son enormes en comparación con las operaciones análogas en coma fija. La ventaja de la coma flotante es que no requiere un esfuerzo extra por el diseñador para seleccionar los tamaños oportunos para las señales a definir. Pero este modelo no se baraja entre las opciones de diseño debido a que con una buena elección de tamaños en coma fija no aporta mejoras en resolución y además requiere tiempos de simulación mayores [15].

- Coma fija es no sólo simulable, sino que puede ser sintetizable en una FPGA, permitiendo emulaciones. Este tipo de datos requiere un trabajo por parte del diseñador para definir los tamaños oportunos para cada señal y su formato adecuado para la parte entera y decimal siguiendo las indicaciones de la biblioteca que rige esta aritmética [13]. Como parte positiva, mejora los tiempos necesarios para la verificación del convertidor junto al control.

- El vector lógico es una estructura de datos que consta de un conjunto de bits en forma de vector. Es el formato básico para definir un conjunto de bits y su interpretación directa sería un número entero, no dando soporte a operaciones con coma. Aunque no tener soporte para manejar el lugar de la coma puede parecer un problema, precisamente esta limitación se va a usar como ventaja ya que el modelo que se propone no va a conocer la posición de la coma, sino que trabajará con números como si fueran enteros, permitiendo aún mayor velocidad de síntesis. Sin embargo, como matemáticamente se necesita operar

con decimales, cada número en este modelo tendrá una escala o «número de decimales» virtuales. Para conocer el valor de una variable de estado o de cualquier otra señal del modelo, sólo hará falta interpretar el vector lógico junto a su escala realizando una sencilla operación matemática.

Tras este repaso a las distintas aritméticas, en este caso se implementarán cuatro modelos: un modelo *real* inicial que servirá para verificar el modelo matemático, a continuación se realizará el modelo en coma fija capaz de trabajar más rápido que el modelo real. Tras esto se desarrollará un modelo empleando vectores lógicos para unos valores concretos del circuito (modelo no parametrizable) y por último se ampliará este último modelo a un modelo parametrizable capaz de soportar diversos valores del circuito.

3.3.1 Modelo *Real*

En este modelo se implementan las ecuaciones del Flyback con aritmética en coma flotante usando el tipo de datos *real*. En este caso el diseño es sencillo pues no es necesario definir los tamaños de las señales, sino que es suficiente con declararlas. Para este modelo se ha realizado el diseño con dos procesos: uno combinacional y otro secuencial.

Un proceso combinacional es aquel que se inicia cuando se produce algún cambio en las señales de su lista de sensibilidad. Éste es el motivo por el que la lista de sensibilidad es muy importante en este caso ya que si alguna señal cambia de valor y no se encuentra en la lista de sensibilidad, no variará el valor que dependa de ella. Este proceso hace referencia a la apertura y cierre del interruptor que se representa como Q en el código que se puede ver en el Anexo 1. El proceso secuencial es aquel que incluye en su lista de sensibilidad la señal de reloj y la señal de reset asíncrona. Por lo tanto este proceso se activa de forma síncrona con la señal de reloj o con la señal de reset que actúa de forma asíncrona. Este proceso será el encargado de asignar los valores a las ecuaciones de estado.

En la Figura 10 se puede observar el diagrama que sigue el modelo *real*. En los multiplexores se selecciona el valor que tomarán la tensión de la bobina (v_l) y la corriente en el condensador (i_c) dependiendo del estado del interruptor Q (proceso combinacional). Estos valores son escalados por dt/L y dt/C respectivamente dando lugar a los incrementos de las señales que representan las variables de estado. Estos incrementos se añaden a los valores de las señales de estado en el instante anterior para formar el valor de la ecuación en ese instante. Por último la tensión en el condensador, que coincide con la tensión a la salida, pasa por un registro (proceso secuencial) para registrar su valor en ese instante. Mientras que la corriente en la bobina (i_l) es registrada y se emplea para hallar la corriente de entrada (i_i). Ambas corrientes solo coinciden cuando el interruptor esta encendido y para otro caso el valor de i_i es cero.

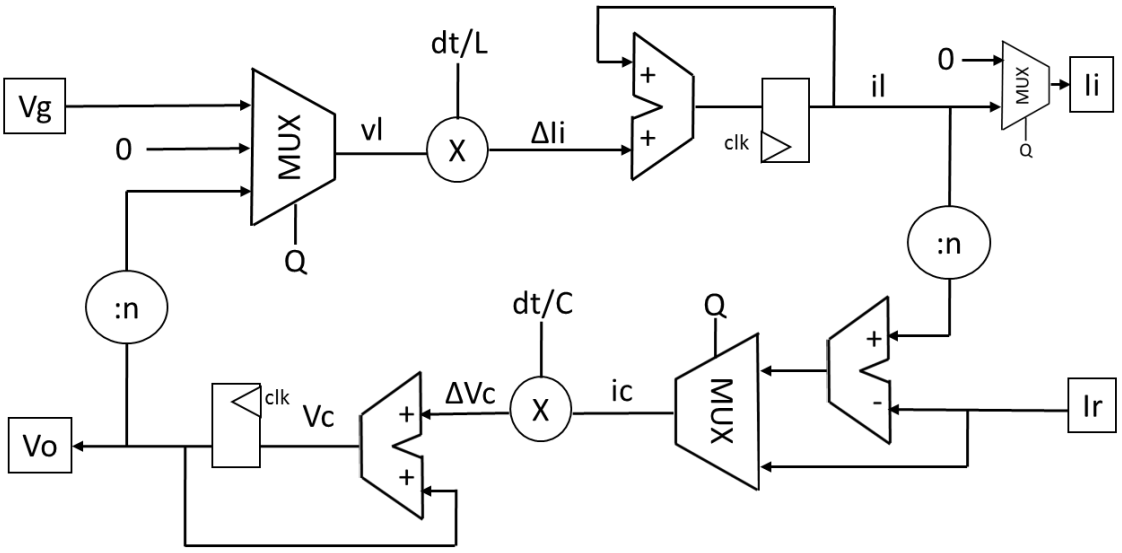


Figura 10: Diagrama del modelo del flyback en tipología real

3.3.2 Modelo coma fija

Este modelo sigue la estructura del anterior pero en este caso la peculiaridad reside en la asignación de los tamaños de las señales. El modelo en coma fija utiliza el estándar VHDL2008, donde se define el tipo de datos *sfixed*. Para declarar una señal de este tipo hay que definir el número de bits que se van a destinar para la parte entera y el número de bits que se asignarán para la parte decimal. El desarrollo completo de este modelo se puede ver en Anexo 2.

En este caso es necesario contemplar los desbordamientos en sumas y multiplicaciones para que las señales mantengan el valor que les corresponden sin saturar. Es conveniente recordar que la coma fija no permite adaptar la posición de la coma si un valor no cabe en el número de bits actual. Para calcular el número de bits enteros necesarios, se toma el entero superior a $log_2(valor\_máximo)$. Conociendo el número total de bits de la señal, y el número de bits enteros, el resto de bits se asignarán a la parte decimal. El principal problema de la coma fija, como se ha comentado, es que este reparto de bits para las partes entera y decimal hay que realizarlo no sólo para las variables de estado, sino para cualquier señal auxiliar del modelo, por lo que su diseño es mucho más complejo.

Además de esto es necesario seguir las pautas de [13] que son sencillas normas para evitar desbordamientos y que se deben aplicar en esta aritmética. Por ejemplo, en el caso de las sumas y restas, es necesario contemplar en el tamaño de la suma un bit extra para la parte entera por si la operación conlleva un acarreo. Por otra parte, el tamaño de una señal resultante de una multiplicación es la suma de las partes enteras y decimales de los sumandos, pero añadiendo un bit extra a la parte entera también para evitar posibles desbordamientos provocados por un acarreo.

En el siguiente código se puede ver el proceso de asignación con esta aritmética donde es necesario redimensionar las señales para que no sea necesario variar el tamaño de las señales en cada iteración. Para esto es necesario haber seleccionado un tamaño de señal lo suficientemente grande como para contemplar la suma total de todas las iteraciones. Esta parte del dimensionado es particular para cada circuito dependiendo de los valores que soporte, por lo que empleando esta aritmética es necesario modificar las señales para cada caso.

```
Asignacion : process (Clk, Reset)
begin
	if Reset = '1' then
		Voo <= Vini;
		Iii <= Iini;
	elsif rising_edge(Clk) then
		Voo <= resize(Voo + Vo_aux*dtC, Voo);
		Iii <= resize(Iii + Ii_aux*dtL, Iii);
	end if;
end process Asignacion;
```

3.3.3 Modelo vector lógico

Para este modelo se define la misma estructura en cuanto a procesos que para los anteriores, aunque es necesario definir el tamaño de las señales de una forma diferente. En este caso se define un único vector que se interpretará en complemento a 2 y para el que no es necesario diferenciar tamaños para la parte entera y decimal. En este caso, siempre se representarán números enteros a los que aplicando una cierta escala, es decir, dividiéndolos entre la potencia de 2 adecuada, se pueden interpretar con su valor real.

Así, para tomar el valor definido en esta aritmética es necesario definir, además del tamaño, una escala que será el número por el que dividir el valor (entero) de la señal fruto de su transformación directa de binario a decimal. Así si la señal es 0110010_2 su valor en decimal sería 50_{10}, si la escala es 2 el valor de la señal sería 25_{10} y si la escala fuera 4, el valor de la señal sería $12{,}5_{10}$.

Este modelo se ha diseñado para unos valores concretos como una primera aproximación al modelo parametrizable, se ha diseñado para los mismos valores que se han realizado los modelos anteriores. La diferencia con el modelo siguiente es que en este caso los valores de las señales se interpretan en el interior del modelo mediante su tamaño y su escala y en el siguiente se interpretarán el un modelo superior.

3.3.4 Modelo parametrizable

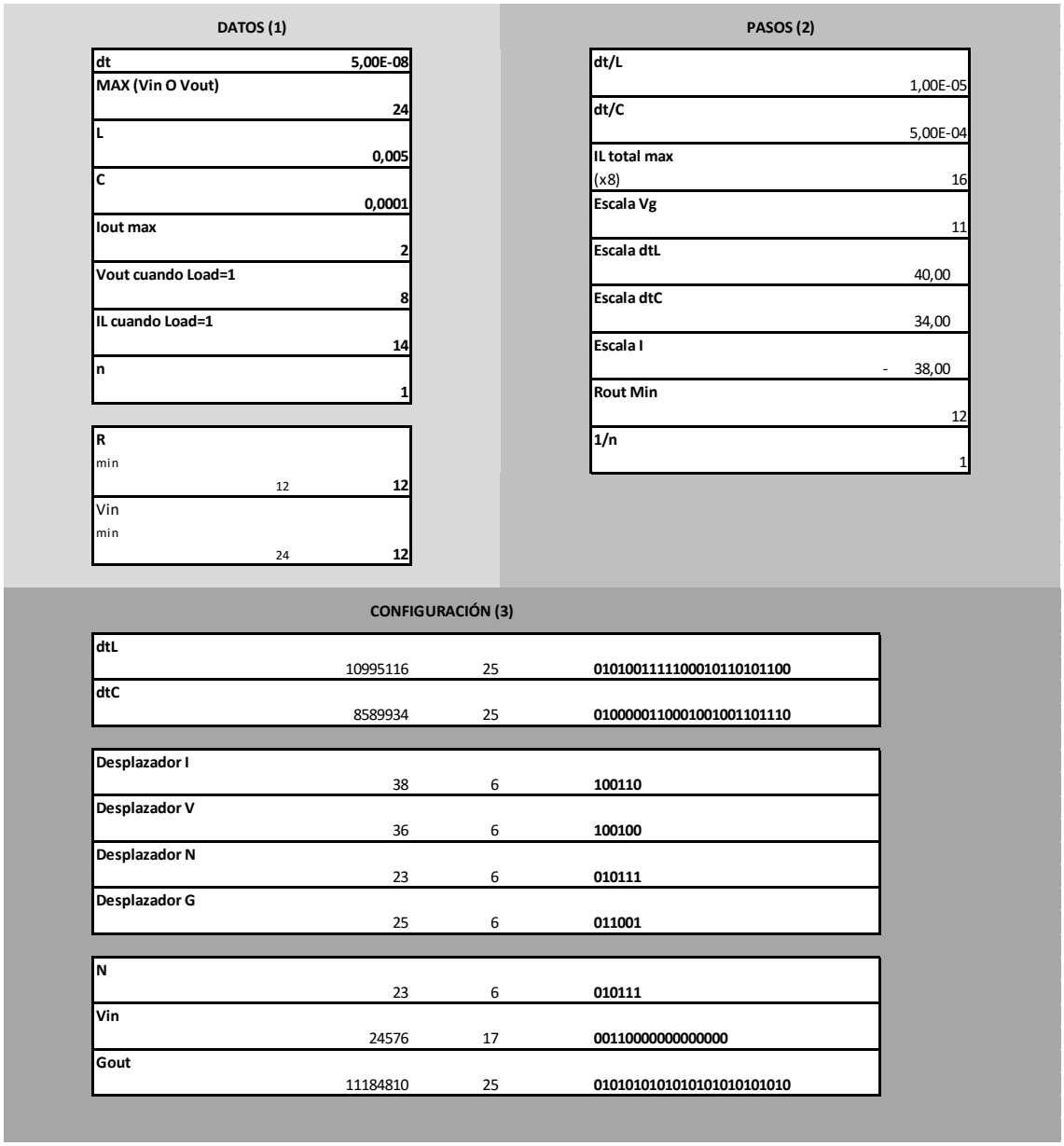
Este modelo se basa en el modelo de vector lógico, es decir, se emplea un tipo de datos *std_logic_vector*. En este caso se sigue la implementación anterior con algunas variaciones.

Como se ha explicado anteriormente, el objetivo es realizar un modelo parametrizable que sirva para simular para cualquier Flyback, independientemente de sus condiciones de simulación (tensiones, corrientes, ciclo de trabajo, etc.). La idea del modelo en coma fija parametrizable es que, una vez sintetizado, el código no cambiará, sino que únicamente habrá que configurar una serie de parámetros para que el modelo se adapte a las condiciones necesarias de la simulación. Esta parametrización consiste en configurar los cambios de escala necesarios para que el modelo funcione correctamente con los valores actuales de tensiones y corrientes. Dicha parametrización vendrá desde el exterior y se puede recibir mediante un puerto de comunicación o desde un procesador, suponiendo que el modelo esté integrado como periférico de un sistema procesador.

Además, el usuario del sistema no tiene, necesariamente, que calcular a mano los parámetros de configuración, sino que se ha realizado una hoja de cálculo para el cálculo automático de dichos parámetros. Se puede ver esta hoja en la Figura 11, donde se muestra la recogida de valores, los pasos intermedios para hallar las escalas y los tamaños y los resultados finales que se introducirán en el modelo parametrizable.

Para implementar este modelo genérico se han tenido en cuenta varias limitaciones iniciales. En primer lugar el hecho de emplear una FPGA, en este caso una Zynq-7000 XC7Z020-1, que posee multiplicadores de 25x17 bits nos guía para el uso de señales de dichos tamaños. En particular, las constantes, como puede ser dt/L y dt/C se escribirán con 25 bits, mientras que las entradas, las salidas, y las señales de realimentación utilizarán únicamente 17 bits. Con estas pautas iniciales podemos diseñar el modelo del Flyback generalizado.

Para poder seguir la realización de este modelo de una forma más sencilla, se va a seguir la implementación de un diseño concreto. Para esto lo primero es solicitar algunos datos sobre el circuito a digitalizar: el máximo entre v_g y v_o, el valor de la bobina de magnetización del transformador, el valor del condensador, la tensión máxima a la salida y la escala de transformación del transformador. Además, también se puede modificar el valor de la carga a la salida y el valor de la tensión a la entrada. Esto se puede ver en la Figura 11 en el cuadro “DATOS”.



DATOS (1)

dt	5,00E-08
MAX (Vin O Vout)	24
L	0,005
C	0,0001
Iout max	2
Vout cuando Load=1	8
IL cuando Load=1	14
n	1

R min	12	12
Vin min	24	12

PASOS (2)

dt/L	1,00E-05
dt/C	5,00E-04
IL total max (x8)	16
Escala Vg	11
Escala dtL	40,00
Escala dtC	34,00
Escala I	- 38,00
Rout Min	12
1/n	1

CONFIGURACIÓN (3)

dtL	10995116	25	01010011111000101101011 00
dtC	8589934	25	01000001100010010011011 10

Desplazador I	38	6	100110
Desplazador V	36	6	100100
Desplazador N	23	6	010111
Desplazador G	25	6	011001

N	23	6	010111
Vin	24576	17	00110000000000000
Gout	11184810	25	01010101010101010101010

Figura 11: Hoja de cálculo para elaborar el modelo parametrizable

Con estos valores se realizan las operaciones necesarias para obtener los valores que se usan en las implementaciones en binario y el tamaño necesario de estas señales lógicas. Para saber cuál es el tamaño apropiado de las señales se usa $log_2(Valor\_máximo)$ que nos devuelve el número de bits necesarios en los que se pueden representar la parte entera de esa señal. Así, si volvemos al ejemplo del apartado anterior (cómo representar el valor 50), $log_2(50) = 5{,}6 \approx 6$, vemos que efectivamente 6 bits son los necesarios para representarlo: 110010_2. Gracias a esta forma de determinar el número de bits necesarios para representar la parte entera, obtenemos el máximo número de bits para la parte decimal en un tamaño de señal específico. Dicho número de bits en la parte decimal es lo que llamaremos escala.

En el caso de complemento a 2 de enteros tanto positivos como negativos, que es la aritmética que se empleará en este caso, es necesario añadir un bit que hace referencia al signo de la señal. Este bit es un 0 si es un número positivo y será 1 si se trata de un número negativo. En caso de valor igual a 1, su representación consiste en interpretar como $-2^{posicion}$ el bit más significativo si está a 1 y el resto como $2^{posicion}$ de manera habitual.

Adaptando las escalas y teniendo en cuenta las operaciones que se realizan entre las señales, somos capaces de crear este modelo parametrizable. Este modelo no es interpretado dentro del modelo equivalente al modelo de planta sino que sus escalas serán empleadas por un modelo superior, que en este caso se designa como *top* para interpretar los valores que se están manejando (Figura 12).

Para comprender mejor este modelo vamos a repasar los pasos seguidos para diseñar un modelo concreto. En primer lugar se han seleccionado como valores del condensador 100 µF, para la bobina 5 mH, una tensión de entrada de 12 V. Se espera una salida de 12 V, pero para contemplar posibles transitorios se estima un máximo de tensión de 24 V y una corriente máxima de 2 A. La resistencia de salida por lo tanto debe ser mayor a 24/2=12 Ω, así se toman estos 12 Ω como carga a la salida.

Como estos modelos se hacen a través de aproximaciones temporales es importante seleccionar el paso de integración que se va a emplear. Este paso debe ser lo menor posible para que las ecuaciones en diferencias se aproximen a las ecuaciones diferenciales de la bobina y del condensador. Sin embargo, el periodo de reloj de la FPGA debe ser igual que el paso de integración si deseamos que el modelo se ejecute en tiempo real. Por tanto, existe una limitación del valor del paso de integración debido al proceso de síntesis en la FPGA. El modelo que se va a utilizar en la propuesta tiene un paso de integración igual a 50 ns, aunque en la sección 4.3 se hará un estudio detallado de cómo afecta este paso al diseño.

Para entender mejor el modelo parametrizable vamos a estimar las escalas para los tamaños de señales antes mencionados. Se puede estimar la escala de una señal como Bits - log_2(Valor) que hace referencia al número de bits designados para la parte decimal. Con esta escala se puede hallar el valor de la señal escalado como el entero resultante de multiplicar Valor x 2^{Escala}. Este valor es el que se va a representar en binario y que se asignará a las señales en *std_logic_vector*, para posteriormente poder interpretarlo correctamente es necesario conocer su escala.

Señal	Valor	Bits	Escala	Valor escalado en decimal	Valor binario
dtL	$1 \cdot 10^{-5}$	25	40	10995116	01010011111000101101011 00
dtC	$5 \cdot 10^{-4}$	25	34	8589934	01000001100010010011011 10
v_g	24	17	11	24576	00110000000000000
n	1	25	23	8388608	01000000000000000000000 0
I_{MAX}	16	17	13	-	-

Tabla 2: Valores del modelo parametrizable del Flyback.

En la Tabla 2 se pueden ver los valores y las escalas que van a tomar para esas señales en el modelo parametrizable del Flyback siguiendo el planteamiento del párrafo anterior, excepto en el caso de la corriente que su valor viene dado por el circuito. En el modelo parametrizable del Flyback, los valores directos no se emplean en este diseño sino que se pasan como señales de entrada de un modelo superior. En este caso de diseño se ha empleado una jerarquía que se puede ver en Figura 12. El modelo del Flyback emplea otro elemento (cambio de escala) para realizar los cambios de escala desde una escala de una señal inicial hasta una escala deseada. Para realizar las pruebas del modelo, se ha creado un fichero llamado *Top*, el cual se encarga de configurar el modelo. Por otra parte, este fichero está, a su vez, instanciado por un *testbench* encargado de dar valores a las señales de reset, reloj, interruptor y al que se reporta el valor de la tensión a la salida. En una versión implementada en FPGA, el *testbench* no se utiliza, y el fichero *Top* tiene pre-guardada la configuración del modelo o, en un caso definitivo, el fichero *Top* implementará un sistema de comunicación para que se pueda configurar el modelo de forma remota.

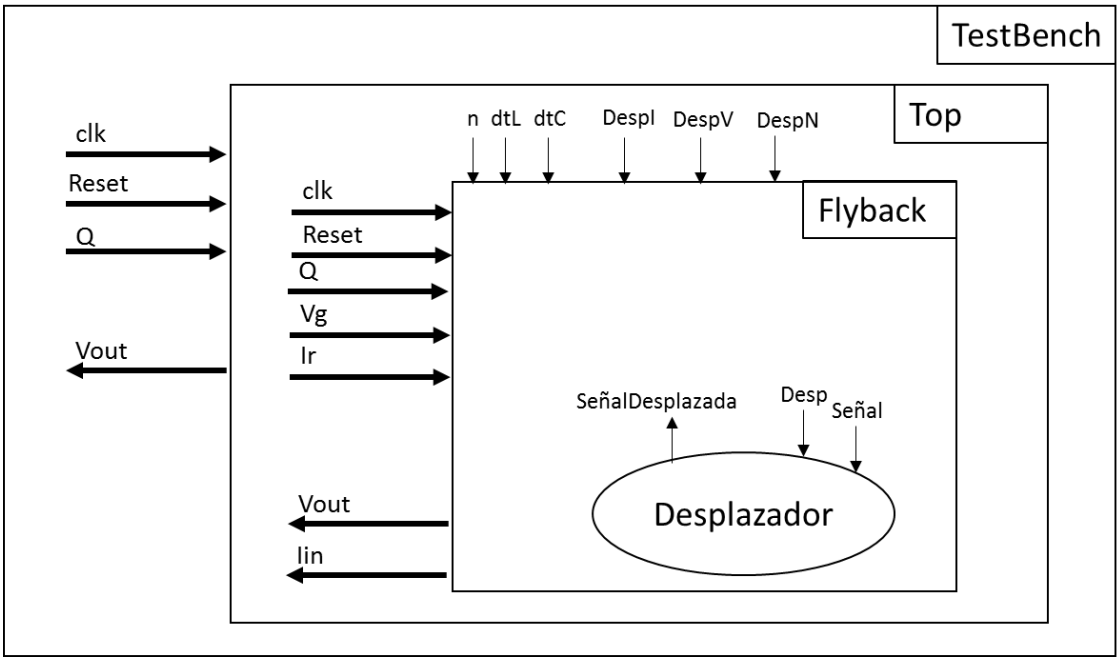


Figura 12: Estructura modelo parametrizable

Para entender mejor este planteamiento parametrizable del Flyback vamos a apoyarnos en la Figura 13. Definida la escala de v_g y dtL, 11 y 40 respectivamente, podemos definir la escala del Δi_i como 51 por ser el producto de v_g y dtL. Para optimizar este modelo se implementa un cambio de escala, que se representa como una flecha. En el caso de i_i se cambia de una escala 51 a una escala 13, ya que el usuario ha configurado que no espera corrientes mayores que 4 A y, por tanto, $log_2(I_{MAX}) = 4$, dando lugar a que su escala sea 17-4=13 bits.

A continuación, dado que el transformador (Figura 6) varía la tensión a su entrada, esta variación se representa con una señal de 25 bits con una escala de 23 para este ejemplo. Por lo tanto es necesario cambiar la escala del resultado de esta división y así recuperar la escala inicial de 13 bits. La escala de i_c sigue siendo 13 bits que al multiplicarlo por dtC, de escala 34, se obtiene ΔVc que tiene una escala de 47 bits. Como la tensión a la salida tiene que tener una escala de 11 se vuelve a modificar esta la escala de 47 para conseguir la entrada deseada.

El motivo por el que se seleccionan señales de 25 bits en su mayoría es porque la FPGA que se piensa emplear tiene DSPc de 25x17 bits que implementan, entre otras operaciones, la multiplicación. Por lo que se puede maximizar la eficiencia de la placa con señales de este tamaño.

El sistema final que se emplea para representar este sistema consta de 3 partes. Un testbench que engloba todo y en el que se dan los valores de reset, de reloj, del interruptor Q y se obtiene el valor de la tensión a la salida. El siguiente elemento es un top donde se dan valores a las señales Vg, dtL, dtC, n, desplazadores y se obtiene el

valor de Ii y de Vo. En este nivel somos capaces de interpretar todas las señales. El último nivel es el equivalente al Flyback en sí que se comporta de la misma manera que el convertidor con las mismas entradas (Clk, Reset, Q, Vg) y salidas (Iin, Vout), además de unas señales de configuración del mismo (dtL, dtC, n, Ir y los escaladores). Así en este modelo se puede configurar todo por el nivel superior (*top*) y así digitalizar distintos modelos.

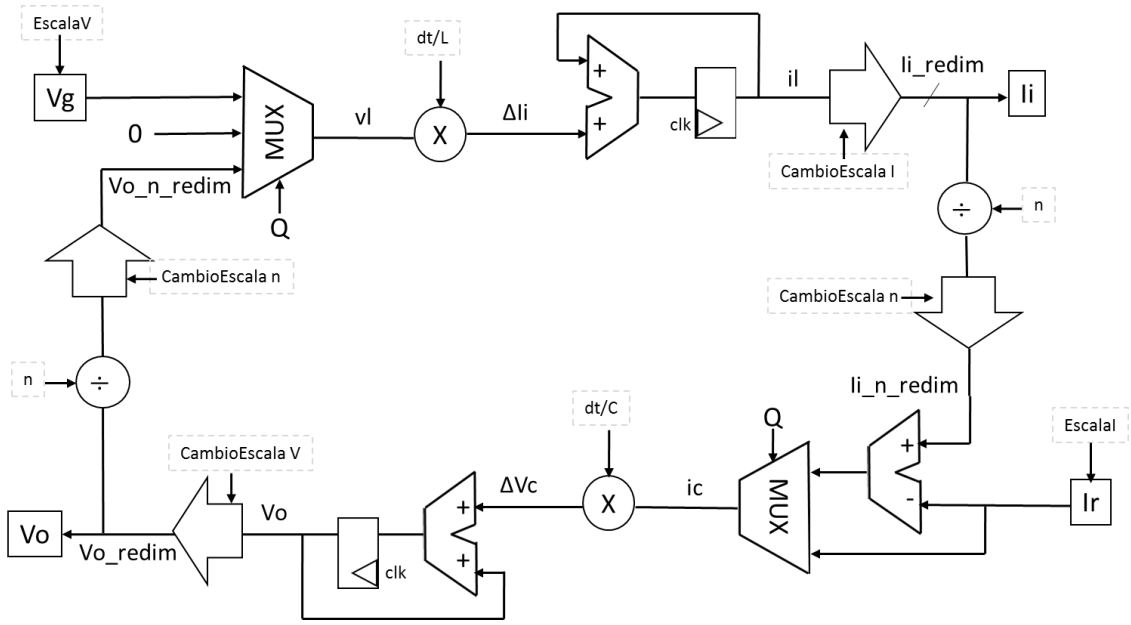


Figura 13: Diagrama del modelo de un flyback parametrizable.

4 RESULTADOS

Finalmente, tras el proceso de implementación hemos obtenido varios modelos: un modelo *real*, un modelo en coma fija, un modelo en *std_logic_vector* y un modelo en coma fija parametrizable. El modelo en vectores lógicos representa un paso intermedio para el entendimiento y diseño del modelo parametrizable pero sus valores resultantes son idénticos a este último por lo que no se empleará en los resultados.

4.1. Lazo abierto y lazo cerrado

Como el objeto de este trabajo es la verificación del circuito digitalizado del Flyback, inicialmente vamos a realizar las pruebas en lazo abierto. Esto significa que se van a simular los circuitos junto a una señal de control fija, es decir, con un periodo de conmutación fijo. En los casos que se muestran a continuación, a menos que se especifique lo contrario, se va a simular con un PWM de 50 µs de periodo de conmutación y un ciclo de trabajo de 50 % lo que equivale a que el tiempo de encendido y de apagado del interruptor es el mismo (25 µs).

El hecho de trabajar en lazo abierto se debe a que, en el caso en el que se simulara el modelo junto un control en lazo cerrado, el propio regulador podría enmascarar los posibles errores en el modelo desarrollado. En caso de existir estos errores y utilizar un lazo cerrado, tendríamos dos posibilidades principales: el modelo puede ser tan inestable que incluso con el control no alcanza el valor esperado, o el modelo puede contener errores, pero el hecho de emplear un control hace que alcance el valor adecuado pese a que la planta no es la apropiada. Sin embargo, en lazo cerrado, el control genera la salida deseada pero no la modifica si los resultados no son los esperados. Por tanto, la precisión del modelo puede ser analizada sin ningún problema.

Estos conceptos son importantes a la hora de verificar que el modelo HIL de una planta, ya que probar un modelo HIL erróneo junto a un regulador en lazo cerrado, puede hacer que el regulador corrija los errores del modelo HIL, y se obtengan las tensiones y corrientes esperadas. Por tanto, para verificar un modelo HIL, se deben utilizar unas señales de control fijas o en lazo abierto, de forma que un error en el modelo HIL sea fácilmente detectable. Una vez que el modelo HIL esté verificado, obviamente éste podrá ser usado en cualquier esquema de conexión y permitirá ser simulado junto a un control en lazo cerrado. Por tanto, en este Trabajo Fin de Máster, se realizarán estudios del modelo propuesto junto a un control en lazo abierto.

4.2. Simulink

Matlab Simulink es una herramienta muy extendida en el diseño de dispositivos electrónicos. Esta herramienta contiene, entre muchas otras, bibliotecas con las definiciones de varios elementos simples como bobinas, resistencias y condensadores, o más complejos como transformadores, actuadores, generadores de señal…

Esta herramienta además permite exportar las señales resultantes en un formato que se puede abrir con Matlab. Esta característica ha sido muy importante para elegir esta herramienta como patrón para nuestros modelos, ya que podemos comparar las muestras obtenidas con Simulink con las muestras obtenidas por los modelos diseñados en VHDL de una forma directa.

Con este programa hemos podido realizar el diseño del Flyback con los elementos facilitados en la herramienta que se puede ver en Figura 14, siguiendo el esquemático de la Figura 2. Así hemos empleado: una fuente de alimentación, un interruptor que en este caso se representa con un Mosfet, un transformador inductivo, un diodo, un condensador y una resistencia de carga. Además de estos elementos, podemos ver un sensor de corriente (*current sensor)* que se trata de un amperímetro que permite medir la corriente en esa parte del circuito y un sensor de tensión (*voltage sensor*) que representa el voltímetro que nos permite medir a la salida del circuito. Otros elementos que podemos ver en este diseño son el *scope*, que consiste en un visualizador donde se representarán las señales, el elemento *PS-Simulink converter*, que es el elemento que permite convertir su entrada en una señal visible en pantalla y, por último, el s*olver configuration*, que es el encargado de aplicar métodos numéricos para la resolución del circuito. Para resolver el circuito hemos seleccionado un una resolución de tipo *Backward Euler* con muestra temporal $1 \cdot 10^{-7}$, esta definición viene por defecto y se puede emplear en la mayoría de simulaciones.

Los valores empleados en este diseño son los que se pueden ver en el apartado de diseño del Flyback con aritmética parametrizable. Como se van a comparar los diferentes modelos en lazo abierto, la apertura y cierre del Mosfet se controla con una señal modulada en pulso que tiene un periodo de 50 µs y un ancho de pulso de la mitad del ciclo de conmutación, es decir 25 µs.

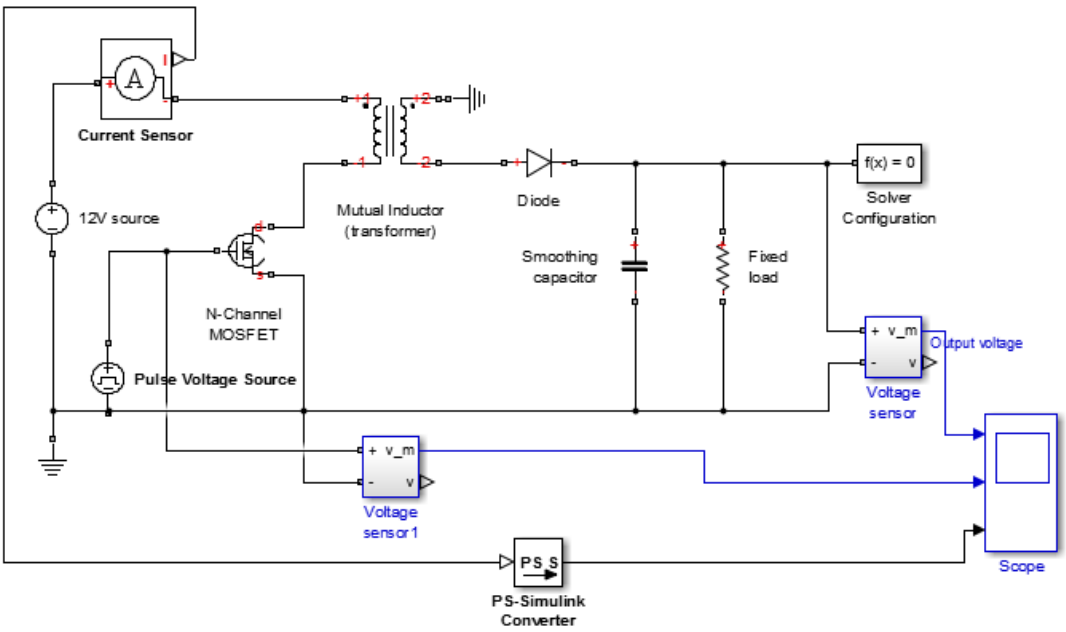


Figura 14: Modelo del Flyback en Simulink

4.3. Elección el paso de integración

Teóricamente, el paso de integración se deriva de las ecuaciones diferenciales de las variables de estado. Para que nuestras ecuaciones en diferencias se parezcan a la realidad es necesario que el paso de integración sea lo más pequeño posible. Para esto se han realizado varias simulaciones en modelo *real* con distintos pasos de integración. Sin embargo, como ya se ha adelantado en apartados anteriores, el tiempo de integración viene limitado por la frecuencia máxima de síntesis del modelo en la FPGA. Por ello, es interesante realizar un estudio teórico para observar cómo afecta el paso de integración a la precisión del modelo.

Para estas pruebas se va a comparar el modelo obtenido con Simulink y el modelo diseñado con aritmética *real* debido a su enorme resolución. Gracias a esto no tendremos en cuenta los posibles problemas que puedan aparecer por falta de resolución y así nos centramos en el objeto de este apartado que es la variación a causa del paso de integración.

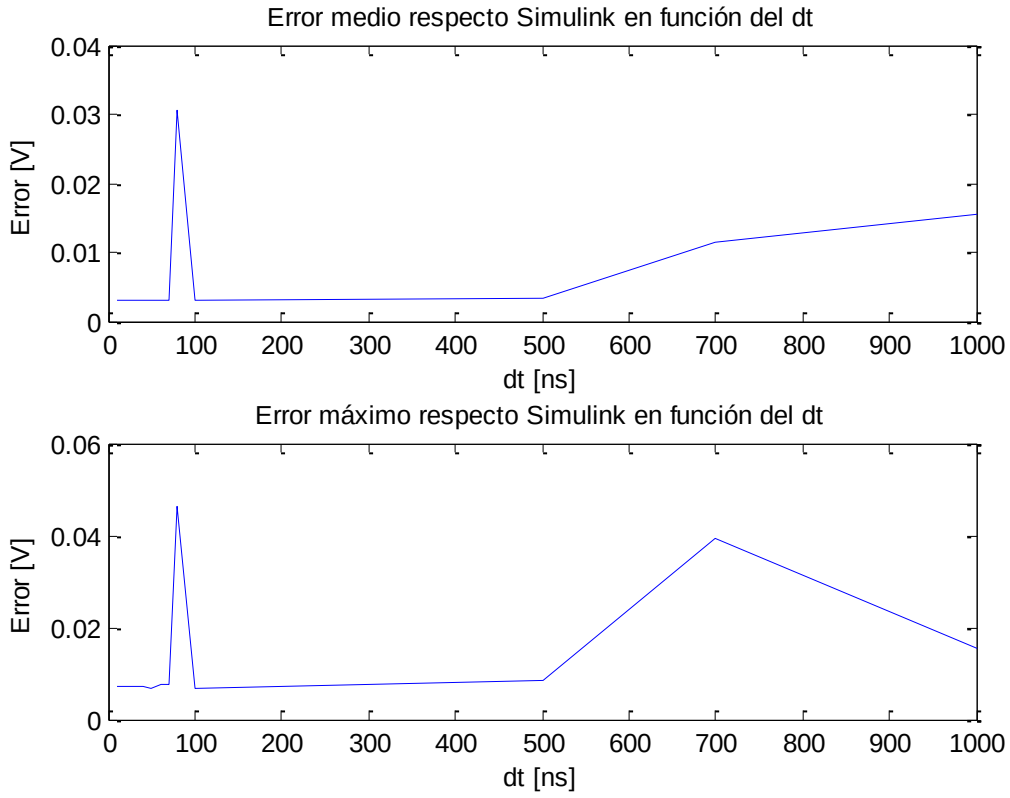


Figura 15: Error entre el modelo real y simulink para distintos dt

En la Figura 15 se pueden observar los errores medios y máximos en la tensión de salida para distintos pasos de integración. En general se puede ver que, como se ha mencionado anteriormente, que el error aumenta al aumentar el paso de integración. Pero esto no es lo único que llama la atención de la figura. Existen varios puntos en los que el error aumenta. Esto se debe a que, la elección del paso de integración puede hacer que el ciclo de trabajo se modifique ligeramente de cara al modelo. Esto ocurre cuando la frecuencia del reloj de la FPGA no es un múltiplo de la frecuencia de conmutación. Si esto ocurre, no todos los ciclos de conmutación tendrán el mismo número de pasos de integración, y los resultados de precisión pueden variar ligeramente. En la realidad este suceso puede ocurrir y las frecuencias de simulación y conmutación pueden acoplarse, pero esto no tiene relación con la *precisión* del paso de integración.

Para tener una visión más amplia de este apartado se han extraído algunos resultados numéricos en la Tabla 3. En esta tabla se pueden ver con mayor precisión los datos obtenidos en la gráfica anterior. Así la dinámica del error en términos generales es aumentar cuando el paso de integración aumenta. Para el caso de un paso de 80 ns estamos en la situación de que el periodo de conmutación, 50 µs, no es múltiplo del mismo y por lo tanto los pulsos de conmutación son ligeramente distintos para este caso.

dt	Error medio	Error máximo	Error medio respecto a 1V
10	0.003070473500644	0.006961111539495	0.31%
50	0.003031018323936	0.006778862922244	0.30%
80	0.030589378385524	0.046156077851860	3.06%
100	0.002999513639753	0.006564457346744	3%
500	0.003456495175397	0.008330371272853	0.35%
1000	0.015600394593770	0.015600394593770	1.56%

Tabla 3: Error en V_o respecto al paso de integración

Por otra parte, en la Figura 15 puede parecer que es suficiente con conseguir pasos de integración en torno a microsegundos, ya que el error es muy bajo. Sin embargo, cabe destacar que esa gráfica se ha realizado para probar un control con un periodo de conmutación de 50 μs. Si la frecuencia de conmutación es mayor, la frecuencia de simulación (que es el inverso del paso de integración) debe aumentar de la misma forma. Actualmente se encuentran controles cuya frecuencia de conmutación está en torno al megahertzio, por lo que un paso de integración de 1 μs obviamente no es viable. Por tanto, la búsqueda del modelo más optimizado para obtener el paso de integración menor está justificada.

4.4. Camino crítico

Para seleccionar el paso de integración es necesario saber las restricciones de tiempo que tiene la placa sobre la que queremos implementarlo, lo que nos permitirá una simulación en tiempo real. Este ciclo de trabajo mínimo necesario para la implementación estará ligado al camino crítico del circuito. Para hallar este camino, tomamos el modelo digitalizado del Flyback en su versión parametrizable y lo ejecutamos mediante la herramienta Xilinx ISE. Esta herramienta permite la simulación del circuito diseñado en una placa y nos devuelve el tiempo que tarda en ejecutarse entre registros.

Para este caso en concreto se va a emplear la FPGA Zynq-7000 XC7Z020-1. A la hora de realizar la etapa de *place & route*, se pueden fijar el periodo de reloj máximo deseado, por ejemplo 20 ns, viendo que la FPGA puede ejecutarlo correctamente:

```
Slack (setup path):     0.107ns (requirement - (data path - clock path skew +
uncertainty))
   Source:               uut/Vo_56 (FF)
   Destination:          uut/Ii_57 (FF)
   Requirement:          20.000ns
   Data Path Delay:      19.695ns (Levels of Logic = 25)
   Clock Path Skew:      -0.163ns (1.491 - 1.654)
   Source Clock:         Clk_BUFGP rising at 0.000ns
   Destination Clock:    Clk_BUFGP rising at 20.000ns
   Clock Uncertainty:    0.035ns
```

Si hacemos este requerimiento más restrictivo, por ejemplo 10, 15, 18 ns, los objetivos de tiempo no se cumplen, por lo que se estima que 20 ns es mínimo periodo de reloj utilizable en nuestro modelo.

Analizando los reportes de la herramienta, se puede observar que el camino crítico de este circuito es la rama que va desde el registro Vo hasta el registro Ii (ver Figura 13).

En el apartado anterior, hemos visto que modelos con pasos de integración mayor a 20 ns funcionan con un error mínimo. Sin embargo, como frecuencias de conmutación mayores, requieren pasos de integración menores, se va a añadir una etapa de segmentación (*pipeline*) en el modelo para reducir el tamaño del camino crítico, como se puede ver en la Figura 16. Estos registros de segmentación se han añadido aproximadamente en el punto medio del camino crítico, pero como se puede ver es casi tan lento el camino de v_o a i_i como de i_i a v_o, por lo que se decide añadir los registros a mitad de camino entre ambas ramas.

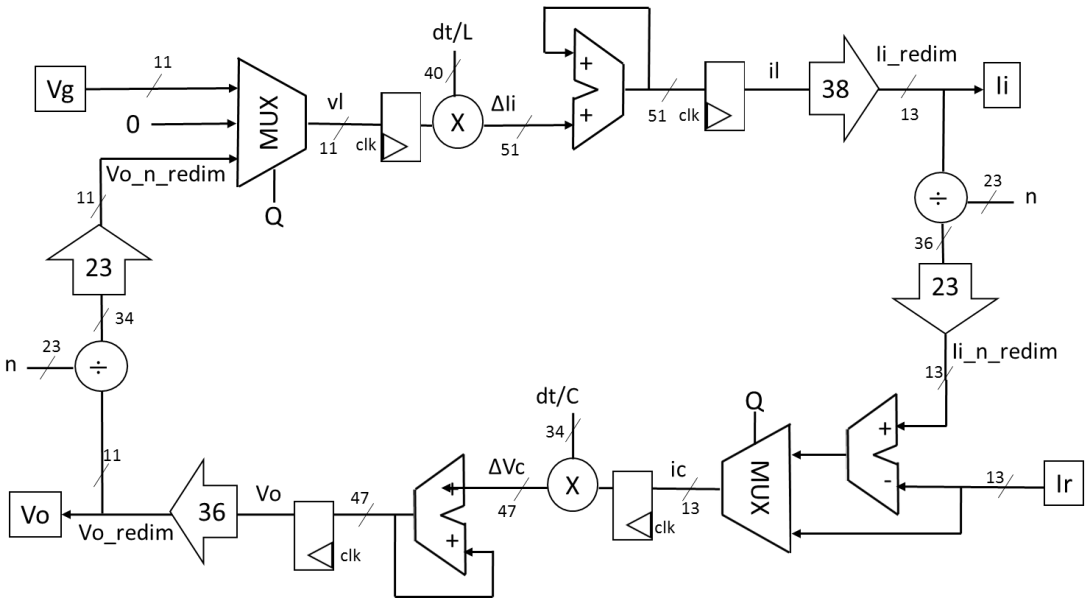


Figura 16: Diagrama del modelo de un flyback parametrizable con pipeline

Añadiendo los nuevos registros se consigue bajar el tiempo de implementación obteniendo unos resultados de tiempo mejores en ISE de 10.708 ns:

```
Slack (setup path):     0.184ns (requirement - (data path - clock path skew +
uncertainty))
   Source:               uut/Ii_56 (FF)
   Destination:          uut/Vo_aux_R_13 (FF)
   Requirement:          11.000ns
   Data Path Delay:      10.708ns (Levels of Logic = 7)
   Clock Path Skew:      -0.073ns (0.839 - 0.912)
   Source Clock:         Clk_BUFGP rising at 0.000ns
   Destination Clock:    Clk_BUFGP rising at 11.000ns
   Clock Uncertainty:    0.035ns
```

Como es de esperar, añadir una etapa de segmentación en el punto intermedio del camino crítico nos ofrece una mejora ligeramente menor que 2x. Este tiempo de simulación es bastante bueno, pero debemos comprobar si la precisión del modelo sigue siendo la misma tras añadir los nuevos registros. Al tratarse de un modelo realimentado, la tensión de salida depende de la corriente del transformador y viceversa. Al añadir una etapa de segmentación en cada rama del circuito, la tensión de salida ahora obtendrá el valor de la corriente del transformador pero en el ciclo de simulación anterior, por lo que se está cometiendo un error matemático.

Para ver cómo afecta este cambio a nuestro modelo, hemos hecho una prueba empleando el paso de integración obtenido en Xilinx de 11 ns. Empleando este paso en el modelo *real* lo ejecutamos añadiendo la pipeline y sin ella y comparamos ambos casos con el modelo de Simulink. Los resultados obtenidos se pueden ver en Tabla 4. Aquí se puede ver que los valores de error son idénticos y por lo tanto, con el fin de ejecutar el circuito a mayor velocidad, podemos añadir pipelines.

	Error medio	Error Máximo
Sin Pipeline	0.0035 V	0.1016 V
Con Pipeline	0.0035 V	0.1016 V

Tabla 4: Error del modelo real con y sin pipeline frente a Simulink

En este caso, se emplea un paso de integración muy pequeño, por lo que las variaciones entre dos instantes son despreciables y el modelo sigue cumpliendo con una buena resolución aunque teóricamente el hecho de añadir un registro adicional varía la definición del modelo.

4.5. Comparativas

Para tomar los resultados de las comparativas vamos a emplear un paso de integración de 50 ns. Este paso es lo suficientemente pequeño comparado con el ciclo de conmutación del Mosfet de 50 µs, que es el ciclo que se va a emplear para las comparativas. Además este paso ha sido verificado en el apartado 4.3 y se ha visto que

su error es prácticamente igual que en los casos de paso de integración menor para dicha frecuencia de conmutación.

Dado que no se va a implementar sobre FPGA para realizar las pruebas, la simulación sobre ordenador de nuestro Flyback irá más rápido cuantas menos muestras tenga que evaluar. Por lo tanto no emplearemos los modelos que incluyen pipeline para las comparativas.

Los datos que se van a emplear para estas comparativas son los que se han especificado anteriormente en el apartado 3.3.4. Estos se van a usar para todos los modelos que se comparen en este apartado.

En este apartado se va a evaluar la precisión del diseño parametrizable frente al resto de diseños en aritmética *real* y en coma fija. Además se compararán estos con el modelo del Flyback en Simulink. Todas estas comprobaciones de precisión se realizan en lazo abierto para poder observar la dinámica del circuito.

4.5.1 Simulink y modelo *real*

En este apartado se va a comparar el modelo *real* con el modelo de Simulink. Idealmente este error debería ser muy pequeño puesto que es la aritmética con mayor resolución. En la

Figura 17 se observan los resultados de la comparación de ambas salidas.

En la primera gráfica se puede ver el error de la tensión de salida en voltios entre el modelo en *real* y el Simulink. Este error es casi despreciable y como cabe esperar mayor en el transitorio que durante la etapa de régimen permanente. La segunda gráfica muestra en azul la corriente de salida de Simulink, podemos ver que esta señal varía su valor entre 0 y el valor que toma la corriente de la bobina. Esta corriente es la que se ha representado conjuntamente para su comparación. Por último, en la última gráfica se representan las tensiones de salida de ambos modelos. Se puede observar como prácticamente se superponen tal y como podríamos aventurar gracias a la primera gráfica.

Visto este resumen se puede estimar que el parecido entre el modelo *real* y el modelo de Simulink es muy fuerte como para poder emplear cualquiera de los dos en la verificación de controles.

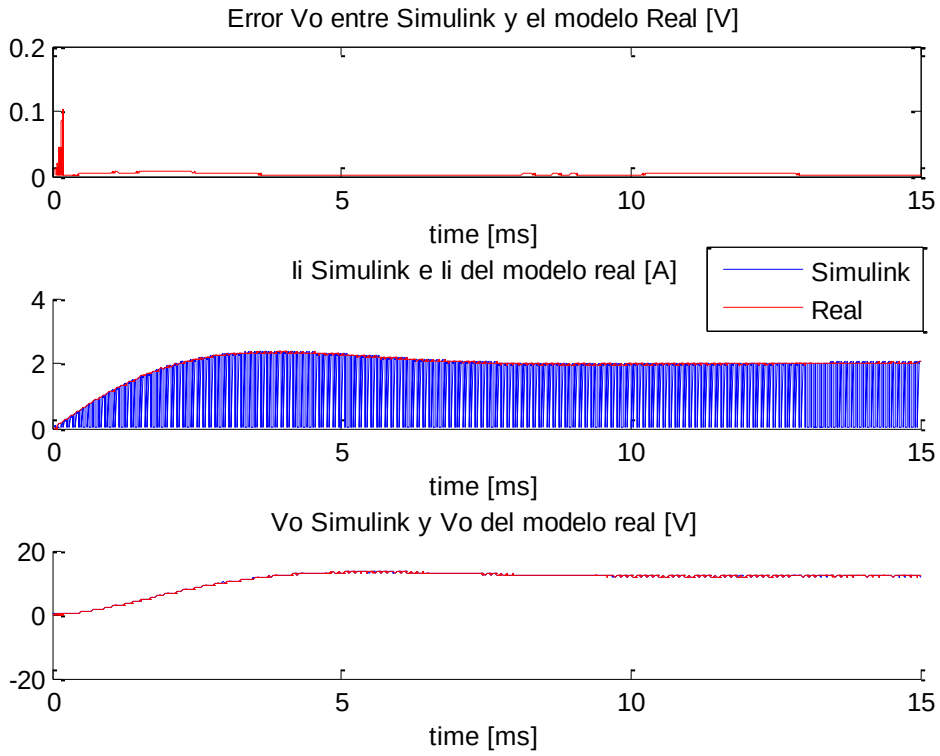


Figura 17: Comparativa modelo *real* y Simulink

4.5.2 Simulink y modelo en coma fija.

En este caso se van a comparar el modelo de Simulink junto al modelo en coma fija. Como se ha comentado en secciones anteriores, en el modelo en coma fija deben seleccionarse las escalas de cada señal. En la versión no parametrizable estas escalas estarán fijas y no podrán cambiarse sin resintetizar. Como se han elegido las escalas óptimas para esta simulación, se espera que el error no sea grande con este modelo.

Este resultado se puede observar en la

Figura 18, donde se representan las mismas gráficas que para el apartado anterior. Efectivamente vemos como ambas formas de onda son muy parecidas como se puede ver en las ondas superpuestas del modelo en coma fija y del modelo de Simulink. Además el error entre ambas es nuevamente muy pequeño respecto a los valores que toma la señal en tensión de salida.

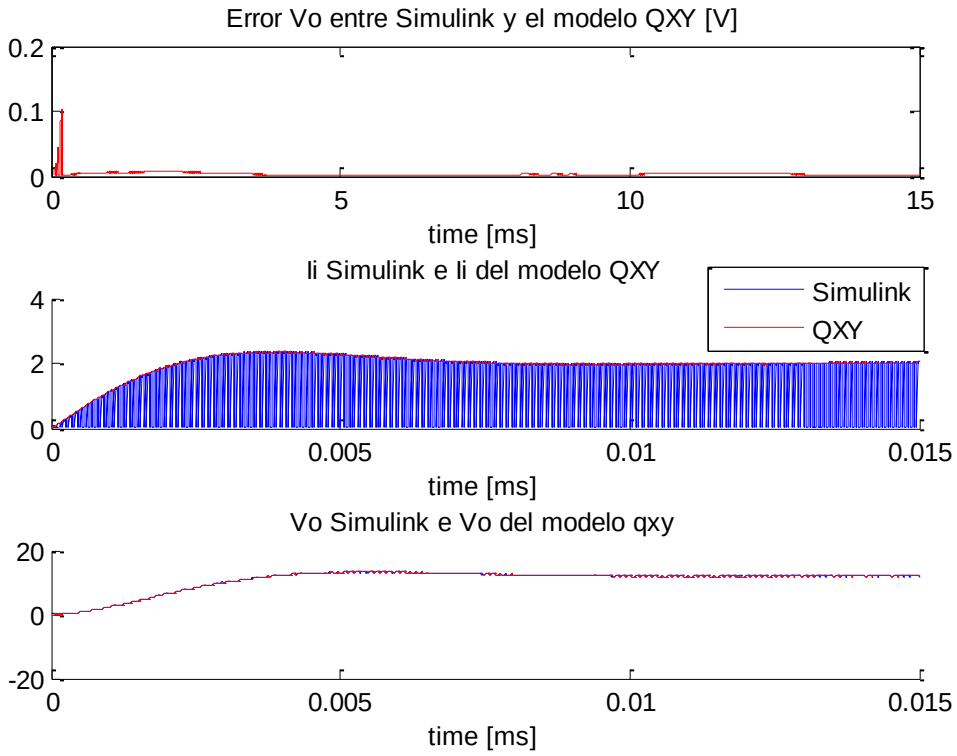


Figura 18: Comparativa modelo en coma fija y Simulink

4.5.1 Simulink y modelo parametrizable.

Este caso se realiza la comparativa del modelo parametrizable junto al modelo de Simulink. Al igual que en los casos anteriores se sacan las mismas gráficas que se pueden observar en Figura 19. Nuevamente vemos que el modelo es muy similar al modelo resultado de Simulink. Por esta similitud, determinamos que el diseño del Flyback con esta aritmética se comporta de la misma manera que el modelo en Simulink frente a la misma una señal PWM.

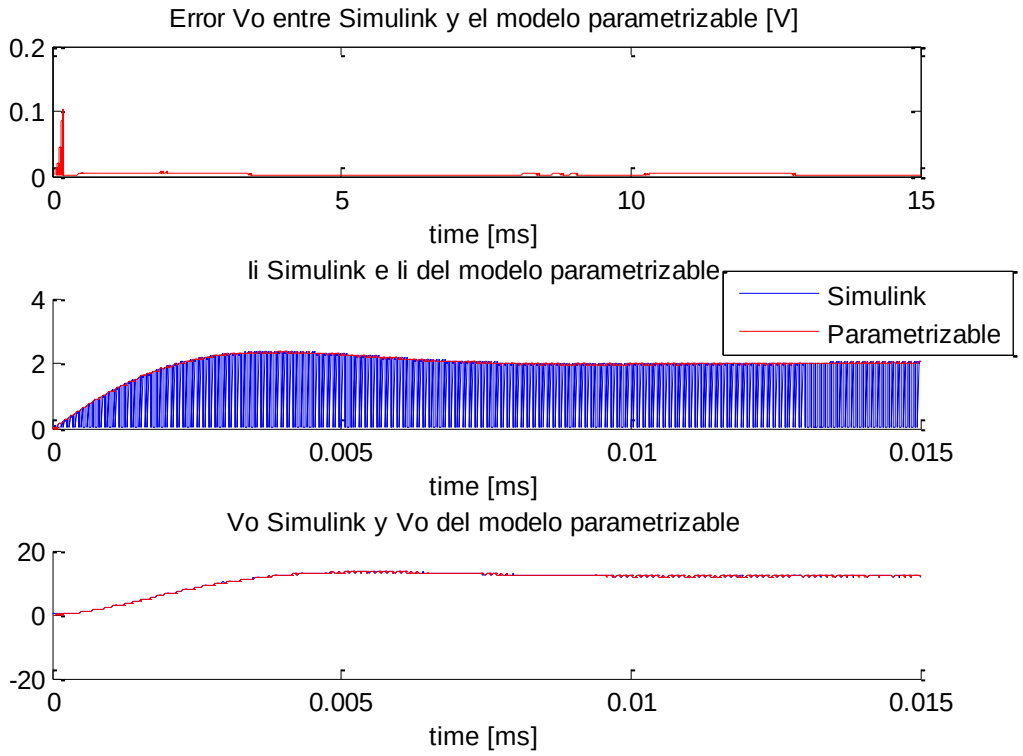


Figura 19: Comparativa modelo parametrizable y Simulink

4.5.1 Comparativa numérica de los cuatro diseños.

Por último para corroborar que los modelos diseñados son válidos en estas condiciones de simulación, vamos a ver con más detalle los errores frente al modelo de Simulink. Podemos ver que efectivamente cualquiera de los modelos diseñados tiene una buena resolución frente a la solución que nos da Simulink.

Modelo	Error V_o medio [V]	Error V_o máx [V]
Real	0.0034	0.1016
QXY	0.0034	0.1016
Parametrizable	0.0033	0.1016

Tabla 5: Errores medio y máximo frente a Simulink

Pese a que la propuesta de Simulink se toma como un modelo fiable y extendido, no funciona a tiempo real. Las simulaciones HIL pretenden ir a tiempo real. En el caso del modelo *real* se pueden conseguir buenos tiempos de simulación, pero solo se puede simular en ordenador ya que esta solución no es sintetizable sobre FPGA. Aun así gracias a su gran resolución se ha empleado para hacer los estudios del paso de integración. El modelo en coma fija QXY no es parametrizable aunque si se puede

emular sobre FPGA y sus resultados para las señales marcadas son muy precisod. El modelo parametrizable, se considera el modelo final del trabajo. Para este modelo se ha demostrado que se pueden obtener resultados precisos con un paso de integración de 50 ns. Además de buena resolución, este modelo es posible implementarlo con este paso de integración en tiempo real, ya que, la FPGA sobre la que se integra es capaz de emplear este paso de integración para realizar las operaciones oportunas hasta resolver las salidas del Flyback.

4.6. Adaptación a diferentes condiciones

En este apartado vamos a verificar el correcto funcionamiento para el modelo parametrizable en condiciones distintas. Como ya hemos dicho este diseño es capaz de adaptarse a otras condiciones mediante la información que se envía desde el *top* que engloba el modelo.

En este caso vamos a variar las condiciones de tensión y vamos a realizar variaciones en el ciclo de trabajo con la intención de conseguir más tensión a la salida. Si recordamos el apartado 3.2, en este hablábamos de una ecuación sencilla con la que modular la salida del convertidor a nuestro gusto:

$$\frac{v_o}{v_g} = \frac{D}{n \cdot (1 - D)}$$

Si mantenemos v_g a 12 V, n a 1 y modificamos el *duty cycle* al 80%, la ganancia del circuito pasaría a ser 4 en lugar de 1 (ganancia con la que hemos trabajado hasta este instante). Añadimos estos valores nuevos en el modelo *real*, en el modelo QXY y en el parametrizable. En este último no hay que olvidar que para realizar cambios es necesario emplear la hoja de cálculo intermedia, con la que tendremos que variar los tamaños y las escalas con las que interpretar el modelo.

En la Figura 20 se puede ver la representación de la tensión de salida del Flyback para las distintas aritméticas y para Simulink. Como se puede ver el modelo *real* y el modelo parametrizable se ajustan a la salida de Simulink, mientras que el modelo en coma fija no es capaz de variar el tamaño de sus señales de forma autónoma, por lo que se puede ver cómo satura al valor máximo.

A diferencia del modelo *real* para el que no es necesario realizar ninguna modificación en el código, para que el modelo en coma fija funcione correctamente es necesario modificar la definición de las señales en el código y resintetizar, esto no es viable. Pero en el modelo parametrizable solo es necesario cambiar las condiciones de la hoja de cálculo (que se podría automatizar para que enviara los cambios al modelo) y por lo tanto reconfigurarlo sin resintetizar.

Para apreciar de una forma cuantitativa los errores de los modelos en este caso podemos consultar la Tabla 6 donde se puede ver el error de cada uno de los modelos respecto a Simulink.

Modelo	Error Vo medio [V]	Error Vo máx [V]
Real	0.0129	0.1002
QXY	6.6589	17.8211
Parametrizable	0.0127	0.1007

Tabla 6: Errores medio y máximo frente a Simulink

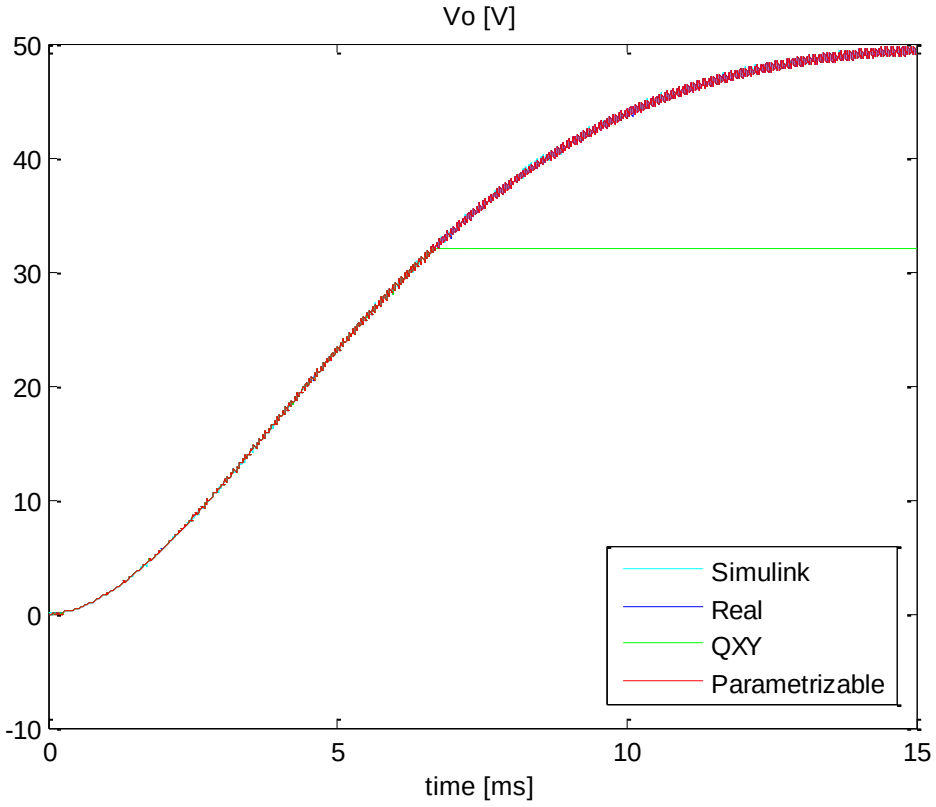


Figura 20: Representación de los distintos modelos con una salida de 48 V.

5 CONCLUSIONES

Debido a lo costosas que son las simulaciones de reguladores digitales junto a modelos analógicos de convertidores de potencia, es habitual utilizar alternativas. Una de ellas es la elaboración de modelos HIL debido a su aumento de velocidad de simulación. En la literatura se encuentran modelos HIL realizados con ordenadores y con FPGAs, entre otros dispositivos. En este caso se ha elaborado el modelo HIL de un convertidor tipo Flyback utilizando una FPGA mediante lenguaje VHDL. Sobre este modelo de planta inicial y se han realizado varias versiones: un modelo en VHDL empleando aritmética real, un modelo VHDL con aritmética en coma fija y por último un modelo en VHDL parametrizable. Todos estos modelos han sido verificados mediante un modelo en Simulink.

Se ha comprobado que cualquiera de los modelos VHDL anteriores se asemeja lo suficiente al modelo de Simulink como para afirmar que su evolución a lo largo del tiempo sigue la misma dinámica. Esto siempre se cumple para el modelo en *real*, comprobando que el método utilizado para modelar el circuito es correcto. Además, este modelo no presenta problemas de resolución numérica, dado que utiliza coma flotante de 64 bits. Debido que la aritmética en coma flotante utilizada no es sintetizable, se han realizado dos modelos en coma fija diferentes: no parametrizable y parametrizable. Ambos modelos también han sido verificados correctamente y se ha mostrado que utilizando un número muy reducido de recursos se obtienen resultados muy precisos y en tiempo real utilizando un tiempo de integración de decenas de nanosegundos. La diferencia entre el modelo no parametrizable y el parametrizable es que el primero sólo es preciso cuando las condiciones de simulación son las esperadas durante la etapa de diseño. Sin embargo, el modelo parametrizable se puede configurar sin necesidad de resintetizar para adaptarlo a dichas condiciones.

La mayoría de pruebas que se han realizado verifican la precisión del sistema. Como el modelo que interpreta este diseño está formado por ecuaciones diferenciales, es importante el paso de integración empleado en la digitalización del modelo. Hemos visto que este paso no depende únicamente del diseño del circuito, sino también de la frecuencia de conmutación del interruptor. Así determinamos que para pasos superiores a 500 ns sobre un periodo de conmutación de 50 µs, el error empieza a crecer de forma drástica. Se ha visto que las simulaciones cuyo paso de integración sea diez veces más pequeño que el periodo de conmutación, son suficientemente precisas para ser útiles.

Por otro lado, para acelerar la simulación del modelo sobre FPGA se ha estimado que el paso de integración máximo soportado en una FPGA Zynq 7000 es de 20 ns. Se ha visto que una alternativa es añadir segmentación al modelo. Aunque las ecuaciones del

modelo se modifican ligeramente para permitir segmentación, los resultados de precisión no se ven deteriorados, permitiendo mejorar la frecuencia de síntesis.

Durante este Trabajo se ha visto que esta metodología de modelado es completamente válida para ser empleado para la comprobación de controladores para convertidores de potencia y, en particular, para un convertidor Flyback.

6 LÍNEAS FUTURAS

En este trabajo se ha alcanzado la creación de un modelo parametrizable del convertidor tipo Flyback. Este resultado, pese a ser un modelo parametrizable que soporta diferentes variaciones del modelo del Flyback, puede continuarse en esta vertiente y añadir mejoras.

Una línea de continuación de este trabajo puede ser la implementación en FPGA de este modelo junto a convertidores DAC. Así obtendríamos un dispositivo que tiene las mismas entradas y salidas de un circuito Flyback, además de unas entradas de configuración. También, se podría realizar una interfaz que fusione la hoja de cálculo y la inserción de los datos al modelo. Así los valores de configuración del circuito que nos devuelve la hoja de cálculo podrían ser directamente introducidos en el Flyback para su puesta a punto.

Como estudio futuro se puede realizar un estudio más teórico sobre el efecto del pipeline en convertidores conmutados en general. En este caso se han realizado varias pruebas que determinan la viabilidad de su uso, debido a que los errores de modelo obtenidos son muy similares al modelo sin emplear pipelines.

Para ampliar las soluciones HIL de convertidores conmutados es interesante contemplar la creación de modelos parametrizables de cada uno de los convertidores tradicionales (*Buck, Boost, Buck/Boost*…), con el fin de crear una biblioteca de modelos que sirva para la verificación de cualquier controlador de convertidores conmutados.

BIBLIOGRAFÍA

[1] Alberto Sánchez, *Aportaciones mediante implementación basada en sistemas embebidos al control digital de convertidores conmutados.* Tesis Doctoral, Universidad Autónoma de Madrid, Junio 2013.

[2] Sandra Jurado Jabonero. *Emulación en FPGA del lazo cerrado de un controlador digital para convertidor conmutado.* Trabajo fin de grado, Universidad autónoma de Madrid. Septiembre 2014.

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

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

[5] A. Procdic and D. Maksimovic "*Mixed-signal simulation of digitally controlled switching converters". In Computers in Power Electronics*, 2002. Proceedings. 2002 IEEE Workshop on pp. Junio 2002.

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

[7] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdio, "*Comparing simulation alternatives of FPGA-based controllers for switching converters*" in Industrial Electronics, 2007. ISIE 2007.Junio 2007.

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

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

[10] Controllab Products, "*http://www.hil-simulation.com/*", Holanda, última visita 06/2016.

[11] Gecko Simulation *"http://www.gecko-simulations.com/geckocircuits.html"*, Última visita: 06/2016

[12] Typhoon HIL, "*https://www.typhoon-hil.com/applications/converter-testing"*, Última visita: 06/2016

[13] David Bishop. "*Fixed point package user's guide"*. Guía para el usuario de un tipo sfixed. Paquetes VHDL-2008.

[14] Ángel de Castro, "*Aplicación del control digital basado en hardware específico para convertidores de potencia conmutados", 2003.*

[15] A. Sanchez, A. de Castro & J. Garrido, "*A comparison of simulation and hardware-in-the-loop alternatives for digital control of power converters*", in IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 491-500, ago 2012.

ANEXO

Anexo 1: Convertidor Flyback Real

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

entity FlybackReal is
	port(
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Q : in std_logic; -- On = '1', off = '0'

		Vg : in real;			-- Senal de entrada
		Ir : in real;			-- Senal requerida por la salida
		-- Out
		Iin : out real;				-- Senal leida a la entrada
		Vout : out real				-- Tension leida a la salida
	);
end FlybackReal;

architecture Behavioral of FlybackReal is

	constant dt: real := 50.0e-9;	-- CLK
	constant C : real := 100.0e-6;	-- 100 uF
	constant L : real := 5.0e-3;	-- 5 mH
	constant n : real := 1.0;	 --Constante transformador

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

	constant Iini : real := 0.0;  -- Puntos de inicio de la simulacion
	constant Vini : real := 0.0;  --

	-- Senales de salida
	signal Il, Ii: real := 0.0;
	signal Vo : real := 0.0;

	-- Variables de entorno
	signal ic: real := 0.0;
	signal vl: real := 0.0;


begin
	Iin <= Il;
	Vout <= Vo;

	-- Proceso de conmutacion de los interruptores
	conmutacion : process (Q, Vg, Ir, Il, Vo)
	begin
		if Q = '1' then		-- Cerrado
			vl <= Vg;
			ic <= -Ir;
			Ii <= Il;
		else
```

```
                    if Il > 0.0 then
                           vl <= -Vo/n;
                           ic <= Il/n-Ir;
                    else
                           vl <= 0.0;
                           ic <= -Ir;
                    end if;
                    Ii <= 0.0;
             end if;
      end process conmutacion;

      -- Proceso de asignacion con el reloj y reset
      Asignacion : process (Clk, Reset)
      begin
             if Reset = '1' then
                    Vo <= Vini;
                    Il <= Iini;
             elsif rising_edge(Clk) then
                    Vo <= Vo + ic*dtC;
                    Il <= Il + vl*dtL;
             end if;
      end process Asignacion;


end Behavioral;
```

Anexo 2: Convertidor *Flyback QXY*

```
library IEEE, WORK,IEEE_PROPOSED;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;

entity FlybackQXY is
	port(
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Q : in std_logic; -- On = '1', off = '0'

		Vg : in std_logic_vector(16 downto 0);
		Ir : in std_logic_vector(16 downto 0);
		-- Out
		Iin : out std_logic_vector(16 downto 0);
		Vout : out std_logic_vector(16 downto 0)
	);
end FlybackQXY;

architecture Behavioral of FlybackQXY is

	constant n : sfixed(1 downto -23) := to_sfixed(1, 1, -23); -- 1/n
	constant dtL : sfixed(-16 downto -40) := to_sfixed(0.00001, -16,-40);
	constant dtC : sfixed(-10 downto -34) := to_sfixed(0.0005, -10,-34);

	constant Iini : sfixed(5 downto -51) := (others=>'0');
	constant Vini : sfixed(10 downto -47) := (others=>'0');

	-- Senales de salida
	signal Ii: sfixed(3 downto -13) := (others=>'0');
	signal Vo : sfixed(5 downto -11) := (others=>'0');
	signal Iii: sfixed(5 downto -51) := (others=>'0');
	signal Voo : sfixed(10 downto -47) := (others=>'0');

	-- Variables de entorno
	signal Vo_aux : sfixed(3 downto -13) := (others=>'0');
	signal Ii_aux : sfixed(5 downto -11) := (others=>'0');

	-- Solo para simulacion
	signal Vo_real : real := 0.0;
	signal Ii_real : real := 0.0;
	-- Solo para simulacion
	signal Ir_real, Vg_real : real := 0.0;

begin
	-- Senales reales solo para la simulacion ------------------------------
---------
	Vo_real <= to_real(Vo);
	Ii_real <= to_real(Ii);
	Ir_real <= to_real(to_sfixed(Ir,Vo_aux));
	Vg_real <= to_real(to_sfixed(Vg,Ii_aux));
	-------------------------------------------------------------------------
---------
	Vo <= resize(Voo, Vo);
	Ii <= resize(Iii,Ii);
	Vout <= to_slv(Vo);
	Iin <= to_slv(Ii);

	-- Proceso de conmutacion de los interruptores
	conmutacion : process (Q, Vg, Ir, Iii, Voo)
```

```
	begin
		if Q = '1' then
			Ii_aux <= resize(to_sfixed(Vg,Ii_aux), Ii_aux);
			Vo_aux <= resize(-to_sfixed(Ir,Vo_aux), Vo_aux);
		else
			if Ii > to_sfixed(0,Ii) then
				Ii_aux <= resize(-(Voo*n), Ii_aux);
				Vo_aux <= resize((Iii*n - to_sfixed(Ir,Vo_aux)),
Vo_aux);
			else
				Ii_aux <= (others=>'0');
				Vo_aux <= resize(-to_sfixed(Ir,Vo_aux),Vo_aux);
			end if;
		end if;
	end process conmutacion;

	-- Proceso de asignacion con el reloj y reset
	Asignacion : process (Clk, Reset)
	begin
		if Reset = '1' then
			Voo <= Vini;
			Iii <= Iini;
		elsif rising_edge(Clk) then
			Voo <= resize(Voo + Vo_aux*dtC, Voo);
			Iii <= resize(Iii + Ii_aux*dtL, Iii);
		end if;
	end process Asignacion;


end Behavioral;
```