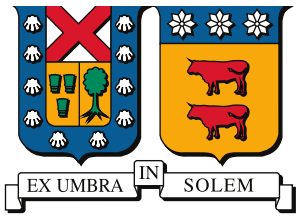
2022-01

SIMULACIÓN EN TIEMPO REAL DE UN CONVERTIDOR DC-DC FLYING CAPACITOR DE TRES NIVELES

OSORIO VILLALOBOS, FELIPE ANDRES



UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Memoria de Título

Simulación en tiempo real de un convertidor DC-DC Flying Capacitor de tres niveles

Tesis para optar al título de
Ingeniero Civil Electrónico Mención Electrónica Industrial

Alumno
Felipe Andrés Osorio Villalobos

Profesor Supervisor
Dr. Christian Rojas

Comisión Co-Referente
Dr. Marcelo Pérez

Enero 20, 2022, Valparaíso, Chile

La ciencia, muchacho, está hecha de errores, pero de errores que conviene cometer, porque conducen poco a poco a la verdad
-Jules Verne

AGRADECIMIENTOS

ESTE trabajo de memoria de título es posible gracias al apoyo del profesor Christian Rojas, quién confío en mi autoría, brindándome académicamente los conocimientos necesarios, siempre con buena disposición. Estoy seguro que es una inspiración y referente para muchos estudiantes, haciendo crecer el interés por electrónica de potencia y la electromovilidad.

Estaré por siempre en deuda con mi madre y padre, hermana, abuelos y abuelas, tíos y tías, por su apoyo, motivación, consejos, paciencia y cariño durante mi vida y especialmente en mi época universitaria, en que más que nunca reforzaron el valor de la superación personal y la responsabilidad. Gracias Sofía, por el inmenso cariño, que a pesar de la distancia, estuvo en todo momento presente, recordándome que sé es humano aunque el estrés haga que se olvide.

Es innevitable recordar a mi excelente profesora de matemáticas Ada Cam. Gracias a sus enseñanzas y motivación en mis tiempos escolares, pude continuar mis estudios en la UTFSM, lugar que afortunadamente me permitió tener buena una vida universitaria.

De seguro habría sido aun más arduo y complejo mi ciclo de estudios sin la camaradería de mis compañeros y compañeras que me apoyaron en los momentos de mayores dificultades académicas. Sin el equipo de Sysmic Robotics (Ex- AIS RoboCup) no hubiese descubierto lo divertido que puede ser trabajar en equipo, incluso en largas y estresantes jornadas de trabajo. Finalmente, gracias a las Academias de Tecnología, por hacerme ver como una niña o niño puede hacer realidad sus propios proyectos electrónicos con empeño y dedicación, mientras se le brinden los recursos necesarios.

Finalmente, extiendo mis agradecimientos al proyecto de investigación Fondecyt Regular 1210757 por el financiamiento a la presente memoria de título.

Felipe Andrés Osorio Villalobos

RESUMEN

EL laboratorio PowerLab del departamento de Electrónica de la UTFSM, lleva a cabo distintos proyectos de investigación en áreas de conversión eléctrica, accionamiento AC, energías renovables, electromovilidad y sistemas de control.

Para desarrollar prototipos y testear tecnologías de innovación, es necesario comenzar por una etapa de simulación de convertidores eléctricos. Ya se han desarrollado modelos de máquina de inducción (IM) y tarjetas de adquisición, así como la construcción de un Flying Capacitor. La simulación y control de un modelo de FCBC en la plataforma de simulación en tiempo real MLBX, consiste en un avance que permite generar un mayor banco de pruebas para integrar los trabajos anteriores y potenciar el desarrollo de las tecnologías en estudio actualmente en la academia y para se transferidas a la industria.

La MLBX posee dos módulos, el DS1302 corresponde a la FPGA Xilinx Kintex-7 XC7K325T y el módulo DS1202 corresponde a un procesador en tiempo real de 64 bits Freescale QorlQ P5020. Estos módulos se utilizan bajo el esquema FPGA-in-the-loop para simular el convertidor en tiempo real, siendo esta una etapa de simulación para un prototipado rápido de control (RCP). En el marco teórico se compara la MLBX con otras plataformas del mercado, que pueden poseer entornos de software para su programación distintos de Simulink, pero siguen siendo entornos de alto nivel, al mismo tiempo que su hardware corresponde a un System on Chip (SoC) que incluye FPGA, ya que esta última es necesaria para simular las dinámicas eléctricas.

La plataforma MLBX puede ser programada en C con Simulink Coder™ o una IDE externa,también en VHDL con Verilog, o por bloques con la librería de Xilinx Generator Systema (XSG) para DSP, la cual es universal para distintos equipos RTI que utilizan chips Xilinx. Las versiones de software utilizadas en este estudio son: Windows 10 Ultimate, Matlab&Simulink 2015b, la librería XSG for DSP 2014 y RTI library(dSpace), lo cual permite emplear un entorno de alto nivel para utilizar FPGA.

Para obtener buenos resultados de simulación, es necesario realizar un buen modelado del convertidor considerando las limitaciones de hardware y software en la MLBX, lo cual conlleva consideraciones de diseño para los controladores del lazo cerrado que actúan para regular las variables eléctricas simuladas de la planta.

Los datos obtenidos en las simulaciones en tiempo real a partir de la HMI ControlDesk, son mostrados con el objetivo de validar la plataforma para simulaciones FPGA-in-the-loop, dando paso a pruebas sobre hardware real a futuro, en donde se esperan ver el mismo comportamiento eléctrico que en este estudio, tomando en cuenta las simplificaciones de modelado.

Palabras Claves

Plataforma RTI, FPGA, FPGA-in-the-loop, Convertidor de Capacitor Flotante Reductor, RCP, PS-PWM, EV

ABSTRACT

THE PowerLab laboratory of Electronic's department of UTFSM, manage research proyects about eletric conversion, AC electric drives, renowable energies, EVs and control systems.

In order to develop prototypes and to test innovating technologies, it's necessary to start in a simulation stage. The previus developments includes Induction Machine (IM) models, interface boards, and a flying capacitor converter hardware. The simulation and control of a FCBC model on the RT MLBX plataform, means an advance that allows the lab to increase his testing capability to integrate the previous works and develop trending technologies that are need to be transfer to the industry market.

The MLBX has thow modules, one of them is DS1302 that consist in a FPGA Xilinx Kintex-7XC7K325T and the other is module DS1202 that correspond to a real time processor of 64 bitsFreescale QorlQ P5020. This modules are to simulate the system in a scheme called FPGA-in-the-loop, that put the controller in DS1202 module and the FCBC model on DS1302 along with the PWM. This simulation scheme is a step to achieve a functional prototype. In the first pages, MLBX is compared to others real time simulators platforms avalaible on the market that match the facts that allow high level programming and has System on Chip (SoC) hardware with an FPGA, that is necessary to simulate fast electric dymanics correctly in real time.

The platform MLBX can be programmed in C with Simulink Coder™ or with an extern IDE, also with VHDL using Verilog enviroment, or with block programming through library XGS for DSP that is compatible with others Xilinx Chips. The current versions used in this work are: Windows 10 Ultimate, Matlab&Simulink 2015b, XSG for DSP 2014 and RTI block library (dSpace), what enable to programm in a high level enviroment with a FPGA.

To obtain good simulation results, the FCBC model and the control system should deal with hardware limitations of a discrete machine like MLBX, in consequence, there are desing considerations to take on count like the internal clock of DS1202 and DS1302 module. The obtain data of simulations in real time using ControlDesk HMI are shown to validate the platform to this kind of simulations, and give a step towards real hardware test in the future that should have similar behaviors considerating the simplifications of the model.

Keywords

RTI platform, FPGA,FPGA-in-the-loop, Flying Capacitor Buck Converter, RCP, PS-PWM, EV

ÍNDICE

Índice de figuras

Índice de tablas

ABREVIACIONES

Mayúsculas

PS-PWM	: phase shifted pulsewidth modulation.
PI	: proportional-integral coefficients of linear controllers
DSP	: digital signal processor.
IM	: induction machine
FPGA	: field programmable gate array
HMI	: human machine interface
CPU	: Central Processing Unit
EV	: Electric Vehicle
FCBC	: Flying Capacitor Buck Converter
HIL	: Hardware-in-the-loop
PIL	: Power-in-the-loop
RCP	: Rapid Control Prototyping
RTI	: Real-Time Interface
RT	: Real-Time
MLBX	: MicroLabBox
VHDL	: Verilog Hardware Description Language
3L-FCBC	: three-level Flying Capacitor Buck Converter
3L-NPC	: three-level neutral-point-clamped
TLBC	: three level buck converter
2L-VSI	: two-level voltage source inverter
IoT	: Internet of Things
IA	: Inteligencia Artificial
IDE	: Integrated Development Environment
XSG	: Xilinx System Generator
SoC	: State of Charge

Minúsculas

dc	: direct current
ac	: alternate current

SIMBOLOGÍA

Escalares

V_{dc}	: DC Voltage Source
C_f	: Floating Capacitor Capacitance
L_o	: Inductance value
R_o	: Load Resistance
u_i	: i_o and v_o actuation
u_v	: v_f actuation
d	: duty cycle for transistor
c_1	: PWM carrier
s_1	: fire pulse
BW	: Ancho de banda
T_s	: Sampling Period
f_s	: Sampling Frequency

Subíndices

fpga	: Variable de módulo DS1302
cpu	: Variable de módulo DS1202
s1202	: muestreo módulo DS1202
s1302	: muestreo módulo DS1202
1	: Señal utilizada para transistor 1
2	: Señal utilizada para transistor 2

Superíndices

$*$	: references
k	: k-th sampling time, $\mathbf{x}(kT_s) \equiv \mathbf{x}(k) \equiv \mathbf{x}^k$
$\tilde{}$	: average value, e.g., $\tilde{f}$
o	: nominal value

Capítulo 1

INTRODUCCIÓN

1.1. Introducción

LAS simulaciones componen una etapa importante en el desarrollo de un equipo que pretende ser utilizado sin dañar componentes internos, externos o producir accidentes en las personas. En particular, el laboratorio PowerLab, posee proyectos de electro-movilidad y energías renovables que vienen en auge desde los últimos años en respuesta a la necesidad de prescindir del uso de combustibles convencionales. Los automóviles deben ser sujetos a pruebas de desarrollo que permitan tener una confiabilidad debido a las peligrosas consecuencias de una falla en su funcionamiento.

El control de convertidores estáticos en EV requiere de una alta frecuencia de conmutación en los semiconductores del convertidor, lo que pone exigencias en el controlador utilizado . En este caso, se usa PS-PWM (Phase Shifted Pulse Width Modulation) para controlar un convertidor reductor de capacitor flotante de tres niveles (3L-FCBC), con una señal carrier de 100k[Hz], por lo que se requiere un control con una velocidad de procesamiento de al menos 1[Mhz] para tener un buen muestreo de la carrier. Si se agrega además la latencia de procesamiento del algoritmo de control, los requerimientos de velocidad de procesamiento alcanzan fácilmente el orden de 10[Mhz].

Hay diversos tipos de plataformas de simulación en tiempo real, que además de su módulo CPU, incluyen un módulo FPGA que les permiten tener la latencia y el throughput que requieren las dinámicas eléctricas de conmutación antes mencionas, que son comunes en convertidores estáticos. En este estudio, se utiliza la MLBX de dSpace que posee dos módulos: el módulo DS1302 que contiene una Xilinx® Kintex®-7 XC7K325T FPGA ,y por otro lado, el módulo DS1202 posee una CPU QorlQ P5020 dual-core 1.6[Ghz].

Existen distintas estructuras de simulación en el proceso de prototipado de equipos y de sus sistemas de control. En este caso, se simula un modelo del FCBC en el módulo FPGA DS1302, mientras que el controlador es un híbrido entre .C y Simulink que está programado en el módulo CPU DS1202. Este concepto de simulación es llamado FPGA-In-The-Loop, lo cual permite cumplir con la alta tasa de procesamiento del modelo en tiempo real. En estudios posteriores de este sistema, se puede utilizar esquemas HIL o PIL para testear el hardware y el controlador.

1.2. Estado del Arte

1.2.1. Hardware

Sea que un convertidor eléctrico se utilice en aplicaciones aeroespaciales, fotovoltaicas, vehículos eléctricos o minería, la simulación en tiempo real del sistema de conversión energética implica una alta demanda de procesamiento de datos, debido a que las dinámicas eléctricas son rápidas, poniendo una exigencia a a la latencia de los controladores.

La modulación PS-PWM que controla la conmutación de los transistores es utilizado usualmente en frecuencias del orden de los Khz [1], inclusive con los transistores GaN en auge se pueden generar pulsos de Mhz. Si se quiere una buena resolución de simulación, entonces se aumenta el requerimiento para representar las dinámicas generadas por el controlador PWM, en donde se suele explotar el paralelismo de una FPGA para computar múltiples señales.

La unidad de procesamiento central debe tener una velocidad de procesamiento suficientemente rápida y tener suficientes núcleos para satisfacer los anchos de banda de los algoritmos de control a utilizar.

Los puertos de la plataforma de RCP, deben ser suficientes para medir todas las señales que se requieran en el modelo o controlador a implementar. Además, se tiene que tener en cuenta la velocidad de muestreo de estos puertos, que puede ser más baja que la frecuencia de conmutación de un semiconductor en algunas aplicaciones.

(a) MicroLabBox (b) OP4510 (c) RT BOX 1 (d) RT BOX 2

Figura 1.1: Vista frontal de los equipos

En base a la información de técnica de Plexim [2], dSpace [3] y Opal RT [4], disponible en sus sitios web, se ha recopilado información de los equipos de la figura 1.1, para construir la tabla de la figura 1.2 en donde se comparan las características de las cuatro plataformas de simulación, todas ellas poseen un módulo FPGA y un módulo CPU que interactúan entre ellas.

La OP4510 es la tarjeta de mayor precio. Posee el procesador más rápido con cuatro núcleos de hasta 3.5 [Ghz] y una resolución más alta, hasta 250[ns] de simulación RTI. El punto débil de este modelo en cuanto a la competencia, es la poca resolución de sus canales análogos, en especial de entrada, siendo la más baja con 0.4 Mega muestras por segundo (Msps).

La RT BOX-2 de PLexim y la MLBX de dSpace poseen precios similares. Con sus dos procesadores, la RT BOX-2 alcanza una mayor velocidad de cómputo que la MLBX, además la FPGA ZU7CG tiene la capacidad de simular un sistema más grande que la Kintex 7. Por otro lado, la MLBX tiene una mejor resolución en sus canales análogos que la RT BOX-2, lo que permitiría medir señales a una alta tasa de muestreo. Otro punto en considerar es que la MLBX tiene solo 48 puertos I/O digitales, pero estos pueden usarse tanto para entrada

como salida, lo que brinda versatilidad en conjunto con los otros 16 canales análogos extra en relación a la competencia.

En caso de querer ahorrar aun más costos, se presenta la RT BOX-1, la cual posee una FPGA y CPU de menor rendimiento, pero con muestreo en sus canales análogos al nivel de la competencia.

		Plataforma de simulación en tiempo real				
Nombre		OP4510	MicroLabBox	RT BOX-1	RT BOX-2	
Marca		OPAL-RT Technologies	dSpace	Plexim		
Procesador	modelo	Intel Xeon E3 v5	QorIQ P5020	Xilinx Zynq-Dual-core ARM Cortex-A9 MPCore	Application Processing Unit: Dual-core Arm® Cortex®-A53 MPCore™	Real-Time Processing Unit: Processor: Dual-core Arm Cortex-R5F MPCore
	velocidad	2.1-3.5 Ghz	1.6 to 2.0 GHz	Up to 1GHz	Up to 1.3 Ghz	up to 533MHz
	núcleos	cuatro	dos	dos	dos	dos
Fpga	modelo	Xilinx ® Kintex ® -7 XC7K325T	Xilinx ® Kintex ® -7 XC7K325T	Xilinx ® Zynq-7030	Xilinx ® ZU7CG	
	logic cells(K)	326	326	125	504	
	Block RAM	16	16	9.3	38.0	
	DSP slices	840	840	250	1,728	
Analog	input	16 canales de 16bit (0.4Msps)	8 canales de 14 bit (10Msps)	16 canales de 16bit (2Msps)	16 canales de 16bit (2Msps)	
			24 canales de 16bit (1Msps)			
	output	16 canales de 16bit (1Msps)	16 canales de 16bit (1Msps)	16 canales de 16bit (2Msps)	16 canales de 16bit	
Digital	input	32	48 compartidos	32	32	
	output	32		32	32	

Figura 1.2: Simuladores RTI

1.2.2. Software

Incluso si los requerimientos de hardware son suficientes para simular en tiempo real el modelo del sistema eléctrico, existen aspectos del software que se deben tomar en cuenta en base a las alternativas presentes para la tarjeta a utilizar, como por ejemplo: lenguaje de programación de la CPU de alto y/o bajo nivel, programación visual y/o textual, interfaces gráficas HMI, el lenguaje de descripción de hardware, las librerías disponibles, el precio de las respectivas licencias y la documentación disponible.

La OP4510 emplea eMEGASIM que permite construir sistemas eléctricos a partir de su representación circuital, compatible con Sps, Spice y Plecs. EL software eMEGASIM está basado en Matlab&Simulink, se puede interactuar con entornos Python, TestStand, labView, C++ y Java. Se puede utilizar el software eFPGASIM para programar la FPGA específicamente y eEPHASORIM para integrar las renovables a redes de transmisión y revisar temas de protección de la red. eMEGASIM, eFPGASIM y ePHASORIM conforman un ecosistema para funcionar en conjunto en simulaciones RT.

La MLBX puede ser programada en Simulink utilizando la librería Xilinx System Generator for DSP [5] necesaria para su módulo DS1302. A más bajo nivel, se puede usar

C con la Real-Time library [6], que permite transformar código C en VHDL. A más bajo nivel aun, se puede programar la FPGA usando Verilog.

Las simulaciones de las RT BOX-1 y 2 se llevan a cabo en el software Plecs Standalone o el Blockset vinculable con Simulink, también es programable en C utilizando el bloque Plecs coder .Existe el detalle de que Plexim no indica que la FPGA de la RT BOX pueda ser programada directamente con un lenguaje de descripción de hardware como Verilog o VHDL.

Se puede entonces observar que las tres plataformas de simulación presentan alternativas de programación visual de alto nivel, programación en texto de bajo nivel y descripción de hardware de bajo nivel.

1.2.3. Simulaciones in-the-loop

El objetivo de una simulación en un sistema discreto es generar un modelo aproximado al real para su estudio, control y posterior prototipado. Hay distintas pruebas que se pueden realizar, las cuales de forma progresiva van asemejándose más al sistema final deseado. Estas pruebas son parte del diseño Model-Based [7] [8] que consiste en por medio de distintos modelos concretar un dispositivo, en este caso, electrónico.

A modo de definir la forma en que se incluyen los distintos módulos de una plataforma de simulación para interactuar en tiempo real con el hardware a modelar y/o controlar, se presentan los principales métodos de simulación existentes en el sitio de Mathworks, desarrollador del entorno Simulink utilizado.

1. Model-in-the-Loop: La simulación M-I-L es la primera y más básica etapa de simulación, la cual consiste en crear un modelo de la planta y un modelo del controlador, el cual se prueba por completo en un entorno como Simulink o Plecs. El objetivo es verificar que el controlador y la planta hayan sido modelados exitosamente.

   En esta etapa no se utiliza ni FPGA, ni sistemas embebidos, ni C.

2. Software-in-the-Loop: La simulación S-I-L surge al usar un bloque de código en C. o en el lenguaje que se quiera realizar el control. En esta simulación se valida que el modelo del controlador puede ser convertido a código en C y si es implementable en hardware.

   En esta etapa no se utiliza ni FPGA, ni un sistema embebido.

3. Processor-in-the-loop y FPGA-in-the-Loop: En esta etapa se emplea un sistema embebido para cargar en archivo C, reemplazando el subsistema del controlador. De esta forma se valida completamente el controlador.

   La planta puede ser simulada en un procesador o en una FPGA, lo cual conlleva a la distinción de simulación P-I-L o F-I-L.

   Si se quiere que la simulación suceda en tiempo real, resulta usualmente más factible realizar la simulación en la FPGA.

   Esta simulación no considera el uso de una planta real.

4. Hardware-in-the-loop: En esta etapa sí se utiliza la planta real en interacción con la planta simulada en tiempo real. Las señales de control de la planta simulada se conectan al hardware real, evaluando de esta forma los posibles delay existentes entre el envío de pulsos del controlador y la recepción en la planta real, así como la correcta implementación de sensores en la planta real para registrar sus parámetros.

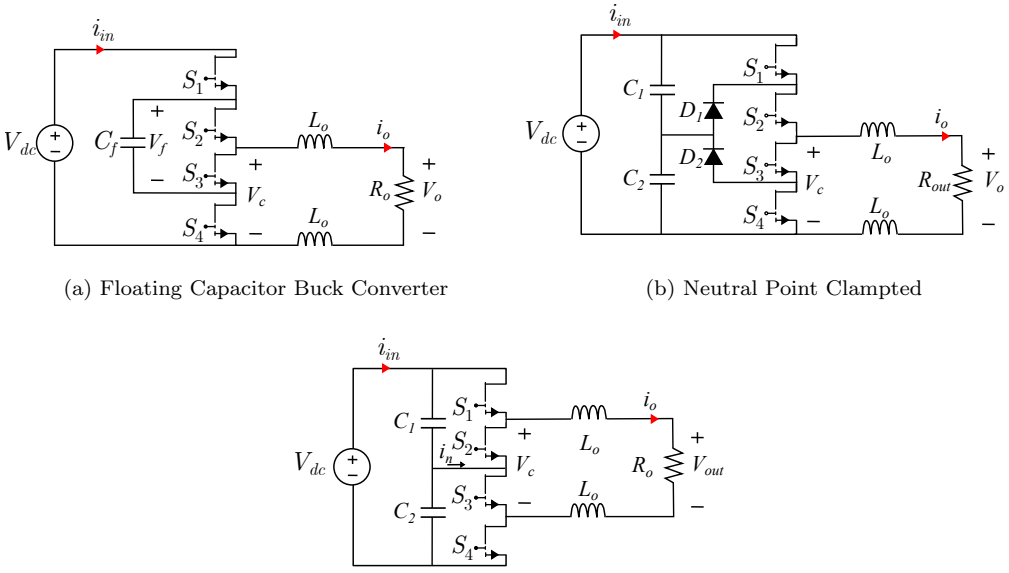
5. Hardware original: Finalmente, la prueba más importante consiste en utilizar el sistema embebido de control FPGA/procesador con la planta real.

Si bien, pueden existir discrepancias respecto de qué esquema de simulación se está realizando dependiendo del segmento de hardware o software a evaluar, este estudio se denomina simulación FPGA-in-the-loop en tiempo real, dado que no se emplea hardware real y se prueba el controlador en C. Aunque se utiliza una pequeña parte de Simulink en el controlador.

1.2.4. Topologías DC-DC

Dentro de los circuitos de conversión se tienen muchas topologías similares al 3L-FCBC que permiten reducir un voltaje de entrada y regularlo en la carga. Aunque, estos mismos convertidores pueden ser utilizados como boost [9] [1] [10] dado que son bidireccionales.

Ejemplo de convertidores de tres niveles son: Three Level Flying Capacitor Buck Converter (3L-FCBC), Three level Buck (TLBC) y Three level Neutral point Clampted (3L-NPC), cuyas topologías se indican en la figura 1.3. Los niveles del convertidor 0, $\frac{V_{dc}}{2}$ y V_{dc} deben verse claramente en el voltaje v_c, no en la carga por la característica filtrante del inductor.



(a) Floating Capacitor Buck Converter

(b) Neutral Point Clampted

(c) Three level Buck

Figura 1.3: Buck converter topologies

Cada convertidor contiene cuatro semiconductores controlados, lo que implica la misma cantidad de señales de disparo requeridas desde controlador.

El NPC posee dos diodos, el nodo que los une es el neutro que conlleva un modelamiento más complejo.

Tanto el 3L-NPC como el 3L-TLBC ocupan dos condensadores, mientras que el 3L-FCBC usa solo un condensador. El incluir un condensador más conlleva otro objetivo de control

importante. En un 3L-FCBC o 3L-NPC se debe controlar el voltaje V_f del condensador y un parámetro de la carga para operar, como la corriente i_o o el voltaje v_L. Mientras que en un TLBC se controla el balance entre los condensadores (V_{c1}-V_{c2}) y un parámetro de la carga.

En aplicaciones de vehículos eléctricos, la bidireccionalidad del convertidor permite hacer frenado regenerativo. De los convertidores expuestos, tanto el 3L-Fying Capacitor, TLBC y el 3L-NPC son bidireccionales. Aunque en una aplicación de carga de batería, dependiendo de la topología, puede no requerirse nunca la regeneración.

Estos convertidores se pueden modelar (sin pérdidas en semiconductores) como un sistema de segundo orden, en donde el más sencillo correspondería al 3L-FCBC por poseer un condensador menos que el resto.

1.3. Contribuciones

En este estudio en simulación y control de un convertidor de potencia en tiempo real, se presenta un modelo idealizado del FCBC en términos de pérdidas potencia o elementos disipadores, pero que sí es representativo de las rápidas dinámicas eléctricas del convertidor.

En el laboratorio PowerLab, se deja disponible un modelo en FPGA con el cual se pueden realizar simulaciones en el esquema FPGA-in-the-loop. Dado que lo que requiere una curva de aprendizaje más larga es el modelado en el módulo FPGA DS1302, esto ahorraría tiempo paran probar distintos controladores haciendo cambios únicamente en el controlador en C.

El manual del apéndice A, que explica como realizar un proyecto básico en la MLBX y simularlo en tiempo real utilizando la HMI para monitorizarlo, se realiza a partir de los manuales [11] [12] [13] [14] contribuidos por otro estudiante memorista del mismo laboratorio [15]. También se contribuye con dos vídeos [16] [17] para facilitar nuevos proyectos de investigación o educativos.

Además, supone un avance para otros estudios que utilicen un esquema F-I-L, S-I-L o H-I-L en la MLBX ,ya que se abordan los problemas de visualización y muestreo de señales que computadas por la FPGA que son más veloces que la CPU pero que interactúan entre sí a partir de buses internos. De esta manera, se aumenta el know-how del laboratorio para explotar la MLBX de manera que se pueda usar todo su potencial en prototipado y estudio de otros sistemas de potencia.

1.4. Objetivo General, Específicos, Alcances y Limitaciones

El principal objetivo de esta memoria de título es simular en tiempo real un convertidor Flying Capacitor de tres niveles operando en DC-DC. En la simulación, se deben visualizar las señales de interés que permitan concluir que el diseño del controlador permite seguir una referencias de corriente y voltaje, así como verificar que los pulsos de disparo del controlador se generen correctamente.

Se encuentran varios objetivos específicos, lo cuales se nombran a continuación:

1. Caracterizar el convertidor en un modelo discreto que permita su implementación en la FPGA de la MLBX.

2. Implementar en el entorno Simulink un modelo del FCBC con las librerías XSG y dSpace. Además, programar el control PWM del convertidor haciendo uso del módulo FPGA de la MLBX en donde se encuentran las señales más demandantes de capacidad de cómputo debido a la frecuencia de la portadora.

3. Simular el control del convertidor para verificar el modelo y su controlador PWM, especialmente las señales de control, que son las necesarias para el FCBC.

4. Realizar simulaciones en tiempo real, evaluar los controladores utilizados y el comportamiento del modelo.

5. En las simulaciones en tiempo real, visualizar las señales de voltaje y corriente físicas del FCBC en una HMI.

Como alcance, se tiene que el modelo de la planta en la FPGA debe ser lo más cercano a la planta real para tener un buen desempeño, en este trabajo no se modelan semiconductores con resistencias parásitas ni voltajes de encendidos.

Existe también un limitante en los puertos de la FPGA, que no permite muestrear a una frecuencia mayor de 100[kHz] en la HMI. Por lo que, las señales de conmutación de los semiconductores y la carrier no pueden ser observadas en tiempo real. Como solución, se observan los valores medios en tiempo real de estas señales, además de realizar pruebas no en tiempo real.

1.5. Resumen de Capítulos

El capítulo "Modelo de FCBC" se enfoca en describir analíticamente el comportamiento de las variables eléctricas de corriente y voltaje de salida, así como del voltaje del condensador para obtener un modelo representable en el entorno de Simulink. Se describe además esquema de control utilizado y el método de sintonización de los controladores PI tomando en consideración las limitaciones de hardware para obtener un control óptimo. Además, Se entregan los parámetros del FCBC en base lo utilizado en un cargador de batería de auto eléctrico.

Luego, en el capítulo "MicroLabBox" explica como utilizar la plataforma Simulink para implementar el modelo del FCBC en el módulo 1302 y el controlador en el módulo 1202. Se muestran visualmente el diagrama de bloques que representa el modelo en el entorno Simulink, así como las consideraciones de implementación, tales como las representaciones numéricas consideradas en el modelo y la latencia del sistema.

En el capítulo "Resultados" se simula el sistema implementado, verificando que las señales se tengan el comportamiento esperado. Se presentan las simulaciones del sistema ante cambios de referencia en la corriente y voltaje de la carga, comparando la similitud entre la simulación y lo esperado según las limitaciones de hardware y los valores de los PI utilizados. Además, se discute la utilidad el control del voltaje del condensador en base a simulaciones realizadas.

Finalmente, los comentarios finales se realizan en el capítulo "Conclusiones" sobre como se cumplieron los objetivos planteados y qué trabajos o pruebas se deben hacer en el futuro para validar completamente el prototipo presentado en este estudio.

1.6. Evaluaciones a Realizar

Se pretende evaluar la validez del modelo de la planta implementado en la plataforma MLBX, por medio de simulaciones que permitan observar que las variables eléctricas obedezcan las ecuaciones que las describen

Probar que en la plataforma MLBX, el modelo del módulo DS1302 funciona a una frecuencia mucho mayor que la del módulo DS1202, lo cual permite tener tener un buen control.

Se espera poder visualizar el sistema en tiempo real, tomando en cuenta que pueden haber cuellos de botella en el registro de algunos datos. Sin embargo, se deben poner a prueba múltiples estrategias para la validez del sistema simulado, y verificar que al menos el comportamiento eléctrico es visualizable en la HMI.

Se pretende evaluar un correcto desacople de los lazos de control de la corriente y voltaje de la carga con el voltaje del condensador. Esto es uno de los aspectos del esquema de control a utilizar que se evaluará en las simulaciones.

Verificar que a pesar de estar utilizando un entorno de alto nivel para la FPGA, de todas maneras los recursos están los suficientemente optimizados como para generar un modelo funcional.

Capítulo 2

MODELO DE FCBC

EN una simulación se trabaja con modelos que representan un sistema real, en este caso, se hacen uso de distintos modelos del FCBC para el análisis, control y simulación del sistema de conversión eléctrica. Se presentan las consideraciones de diseño y de modelado que indican las limitaciones del mismo y los puntos fuertes de la representación.

Los elementos que constituyen el FCBC son el voltaje de alimentación de entrada v_{dc}, el condensador C_f, dos inductores L_o, la carga representada con el resistor R_o, y cuatro semiconductores que poseen como señal actuadora S_1, S_2, S_{1n} y S_{2n}, respectivamente.

El condensador, al ser un elemento almacenador de energía del convertidor, genera una ecuación dinámica en el sistema. Este elemento se carga con el voltaje de alimentación v_{dc}, se descarga en la resistencia R_o, o bien queda "flotante" (sin flujo de corriente), según los estados de conmutación de los semiconductores.

Ambos inductores tienen una inductancia L_o, por lo que para efectos de modelado, es equivalente a utilizar una inductancia $2L_o$. Esto corresponde a otro elemento almacenador de energía, haciendo del FCBC un sistema de segundo orden.

En cada estado de conmutación, hay dos semiconductores activos y otros dos inactivos. Es por esto que cada semiconductor opera con un voltaje de $\frac{v_{dc}}{2}$. Así mismo, el objetivo de control es que el voltaje del condensador C_f sea igual a $\frac{v_{dc}}{2}$.

Los semiconductores utilizados en este modelo son ideales, aunque para efectos de representación gráfica se utiliza el símbolo de transistor GaN, dado que están siendo incorporados en aplicaciones que utilizan FCBC. Además, los transistores GaN pueden operar a la frecuencia de 100[kHz] del control diseñado, o incluso más rápido.

Hay que considerar que un semiconductor posee un tiempo de conmutación t_s en pasar de un estado a otro una vez recibida la señal de conmutación en la base. Inspeccionando la hoja de datos del transistor, se tiene una estimación del tiempo t_s. Para respetar el tiempo de conmutación, se incluye un tiempo muerto en la PS-PWM que controla los pulsos S_1, S_2, S_{1n} y S_{2n}.

Se tienen tres variables de control, el voltaje en el condensador C_f, las corriente i_o y el voltaje v_o de la carga, las cuales son manejadas con un control desacoplado PS-PWM que permanentemente controla v_f pero alterna entre i_o y v_o.

Los tres niveles del convertidor pueden ser vistos directamente en voltaje del convertidor

v_c, que conmuta entre 0[V], $\frac{v_{dc}}{2}$ y v_{dc}.

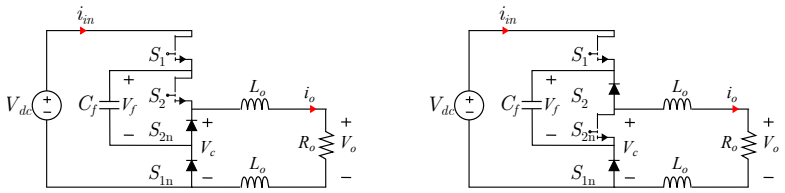
2.1. Ecuaciones del convertidor

El comportamiento del FCBC esta dado por los estados sus cuatro estados de conmutación de los semiconductores de las actuaciones S_1 y S_2. En donde para efectos de control, se considera que:

$$S_{1n} = \bar{S_1} \tag{2.1}$$

$$S_{2n} = \bar{S_2} \tag{2.2}$$

A continuación, se listan los estados de conmutación del convertidor.



(a) Estado 0: S_1 and S_1 on

(b) Estado 1: S_1 on, S_2 off

(c) Estado 2: S_1 on, S_2 off

(d) Estado 3: S_1 and S_2 off

Figura 2.1: Conmutación de los semiconductores según actuaciones

- Estado 0: C_f flotante.

  Se tiene $S_1 = S_2 = 1$.

  Se bloquea la corriente del capacitor, por lo que queda flotante manteniendo su voltaje. Por otro lado, el voltaje de alimentación alimenta al circuito RL, por tanto , $i_{in} = i_o$.

  Las ecuaciones del estado 0 del convertidor son:

$$v_{dc} - 2L_o\frac{di_o}{dt} - R_o i_o = 0 \tag{2.3}$$

$$C_f d\frac{v_f}{dt} = 0 \tag{2.4}$$

- Estado 1: C_f se carga con i_o.

  Se tiene $S_1 = 1$ y $S_2 = 0$.

Se forma un circuito RLC, en el que el condensador se carga con la corriente de la fuente, teniendo en cuenta que al estar los elementos en serie, se cumple que: $i_{in} = i_o$.

$$v_{dc} - V_f - 2L_o\frac{di_o}{dt} - R_o i_o = 0 \tag{2.5}$$

$$C_f d\frac{v_f}{dt} = i_o \tag{2.6}$$

- Estado 2, C_f se descarga con i_o

  Se tiene $S_1 = 1$ y $S_2 = 0$.

  El voltaje de alimentación queda aislado del sistema $i_{in} = 0$. Mientras que el capacitor alimenta la carga RL.

$$v_f - 2L_o\frac{di_o}{dt} - R_o i_o = 0 \tag{2.7}$$

$$C_f\frac{dv_f}{dt} = -i_o \tag{2.8}$$

- Estado 3 , Cf flotante.

  El capacitor está flotante como en el estado 0, pero se diferencia del mismo en que el voltaje de alimentación v_{dc} también queda aislado $i_{in} = 0$.

  En consecuencia la resistencia y el inductor quedan en paralelo conectadas a tierra.

$$-2L_o\frac{di_o}{dt} - R_o i_o = 0 \tag{2.9}$$

$$C_f\frac{dv_f}{dt} = 0 \tag{2.10}$$

2.2. Modelo del Convertidor

La tabla de verdad 2.1 resume las ecuaciones presentadas en los estados 0, 1, 2 y 3. En los cuales, se identifican como términos comunes X e Y, que corresponden al voltaje del circuito R_oL_o (LVK) y la corriente de C_f (ley del condensador) respectivamente:

$$X = 2L_o\frac{di_o}{dt} + R_o i_o \tag{2.11}$$

$$Y = C_f\frac{dv_f}{dt} \tag{2.12}$$

A partir de la tabla 2.1, es posible describir el comportamiento de X e Y en función de S1 y S2.

$$X = S_1 v_{dc} - (S_1 + S_2)V_f \tag{2.13}$$

$$Y = (S_1 - S_2)i_o \tag{2.14}$$

Estado	S_1	S_2	X	Y
0	1	1	v_{dc}	0
1	1	0	v_{dc}-v_f	i_o
2	0	1	v_f	$-i_o$
3	0	0	0	0

Tabla 2.1: Resumen de los estados de conmutación del FCBC

2.2.1. Modelo Conmutado

Al igualar la ecuación (2.11) con (2.13), se puede despejar $\frac{di_o}{dt}$. De la misma manera, al igualar la ecuación (2.12) con (2.14), y se despeja $\frac{dv_f}{dt}$. De esta forma se obtiene el modelo conmutado del FCBC:

$$\frac{di_o}{dt} = \frac{1}{2L_o}(S_1 v_{dc} - (S_1 + S_2)v_f - R_o i_o) \tag{2.15}$$

$$\frac{dv_f}{dt} = \frac{1}{C_f}(S_1 - S_2 i_o) \tag{2.16}$$

EL modelo conmutado tiene variables binarias S_1, S_2, así como variables y datos del conjunto real $i_o, v_f, L_o, C_f, v_{dc} \in \mathbb{R}$. Es por esto, que se considera al modelo conmutado un **modelo híbrido**.

2.2.2. Modelo Promedio

Para efectos de control, es necesario generar el modelo promedio de las ecuaciones (2.17) y (2.18), el cual se obtiene al considerar el valor promedio de las variables binarias S_1 y S_2, como sus ciclos de trabajo d_1 y d_2, respectivamente.

Las variables d_1 y d_2 representan el ciclo de trabajo del semiconductor. En consecuencia, se tiene que $d_1, d_2 \in [0, 1]$

$$\frac{di_o}{dt} = \frac{1}{2L_o}(d_1 v_{dc} - (d_1 + d_2)v_f - R_o i_o) \tag{2.17}$$

$$\frac{dv_f}{dt} = \frac{1}{C_f}(d_1 - d_2 i_o) \tag{2.18}$$

2.3. Modelo promedio discreto

Debido a las rápidas dinámicas del sistema eléctrico del FCBC, se depende de la velocidad de procesamiento de la FPGA de la MicroLabBox para implementar un modelo del convertidor. En la MLBX, las librerías de XSG permiten implementar modelos discretos mediante bloques de código, no así modelos continuos.

El modelo promedio y el modelo conmutado contienen términos derivativos que deben discretizarse dado que son continuos. Se hace uso del método Euler-Forward para la discretización. Aplicando la definición del método para la derivada de corriente i_o y v_f, se obtiene (2.19) y (2.20):

$$\frac{di_o}{dt} = \frac{i_{o_{k+1}} - i_{o_k}}{T_s} \tag{2.19}$$

$$\frac{dv_f}{dt} = \frac{v_{f_{k+1}} - v_{f_k}}{T_s} \tag{2.20}$$

En donde el término T_s corresponde a la latencia del modelo del FCBC $T_{FCBC} = 200[ns]$ implementado en el capítulo 3.

Reemplazando la definición de Euler-Forward para la derivada (2.19) y (2.20) dentro del modelo promedio (2.17) y (2.18), se hace posible despejar $i_{o_{k+1}}$ y $v_{f_{k+1}}$, obteniendo el modelo promedio discreto:

$$i_{ok+1} = \overbrace{\frac{1}{2L_o}(d_{1k}v_{dc} - (d_{1k} + d_{2k})v_{fk} - R_o i_{ok})\,T_s}^{\frac{di_o}{dt}|_k} + i_{o_k} \tag{2.21}$$

$$v_{f_{k+1}} = \underbrace{\frac{1}{C_f}(d_{1k} - d_{k2}i_{ok})\,T_s}_{\frac{dv_f}{dt}|_k} + v_{f_k} \tag{2.22}$$

2.4. Aplicación: cargador auto eléctrico

La carga de un de un auto-eléctrico consiste en una etapa que rectifica el voltaje de la red eléctrica, para luego adaptar los niveles de corriente y voltaje a las características especificas de la batería del automóvil. Esto corresponde a un esquema de conversión AC/DC-DC/DC, como se muestra en el diagrama en 2.3 y el circuito esquemático en 2.2.

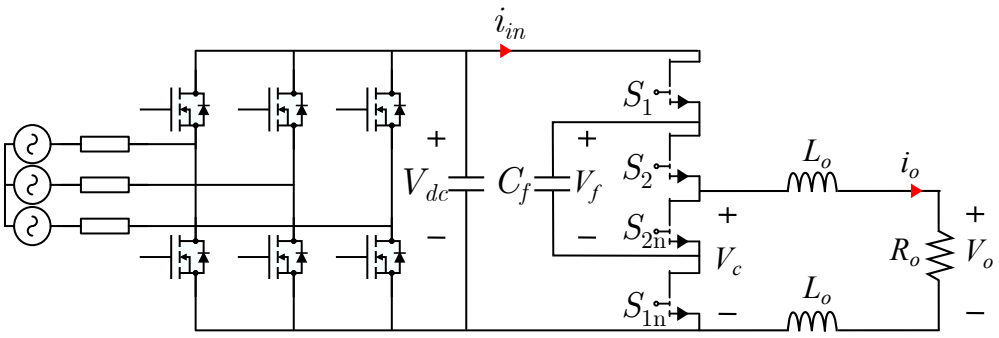


Figura 2.2: Conversión de dos etapas para alimentar una máquina eléctrica

El proceso de carga de la batería Li-On del vehículo, se efectúa con control de corriente o voltaje dependiendo del estado de carga (SoC) de la batería. Si la batería está a 85 % o menos de su capacidad, entonces la carga se hace con control de corriente. Una vez sobrepasado el 85 %, se usa control de voltaje.

El control de corriente y voltaje se lleva a cabo en un convertidor DC/DC, como por ejemplo un FCBC. Basado en los parámetros típicos de operación en esta aplicación, se utilizan los valores de la tabla 2.2 para simular el comportamiento del FCBC y prototipar su controlador.

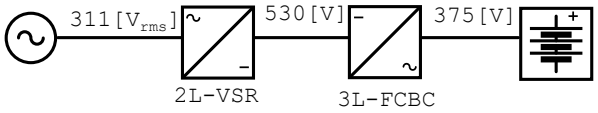


Figura 2.3: Esquema de conversión eléctrica

Parámetro	Valor	Unidad
v_{dc}	530	[V]
v_o^o	375	[V]
p_o^o	550	[kW]
C_f	4700	[μF]
R_o	2.8125	[Ω]
L_o	1	[mH]
f_s	100	[kHz]

Tabla 2.2: Valores de operación del FCBC

Nótese que la resistencia R_o corresponde al modelo de la batería, equivalente a un modelo Shepard de orden 0.

Un parámetro de interés en el FCBC corresponde al "voltaje del convertidor" v_c. Este voltaje representa la variable X en la ecuación (2.11), por medio de este voltaje se observa la transferencia de potencia del FCBC a la inductancia L_o y la carga R_o. Este voltaje nunca será negativo, dado que no existe regeneración en la aplicación presentada, en otra aplicación puede que sí sea negativo, dado que el FCBC tiene capacidad de regeneración, por lo que el valor mínimo de v_c es 0. El máximo valor que puede tomar v_c es v_{dc}, lo cual es la mayor transferencia de potencia posible conectar la fuente directamente a R_o. Por último, el tercer nivel de v_c es v_f, que con el controlador utilizado debe equivaler a $\frac{v_{dc}}{2}$.

La expresión de v_c se determina reemplazando la equivalencia $X = v_c$ en la ecuación (2.11):

$$v_c = V_{dc}s_1 - (s_1 - s_2)V_f \tag{2.23}$$

Por lo tanto, su expresión promedio es:

$$v_c = V_{dc}d_1 - (d_1 - d_2)V_f \tag{2.24}$$

2.4.1. Punto de operación

Un parámetro de interés es el punto de operación del sistema, el cual se define en $V_o^o = 375[V]$. De esta forma se tiene una potencia nominal de $P_o^o = \frac{V_o^o}{R_o} = 50[kW]$.

El valor ciclo de trabajo nominal de los semiconductores d_1^o y d_2^o se calcula evaluando el modelo continuo promedio (2.17) y (2.18) en el punto de operación V_o^o del estado estacionario.

Obteniendo que:

$$d_1^o = d_2^o \tag{2.25}$$

$$d_1^o = \frac{v_o^o}{v_{dc}} = 0{,}7075[-] \tag{2.26}$$

Concluyendo a partir de análisis teórico que en ambos ciclos de trabajo deben ser iguales. Por lo tanto, la actuación del controlador de voltaje del condensador tiene que ser nula:

$$u_i^o = 0{,}7075[-]. \tag{2.27}$$

$$u_v^o = 0[-]. \tag{2.28}$$

2.5. Control

Los objetivos de control son la corriente de salida i_o, el voltaje de salida v_o y el voltaje del condensador v_f. La topología de control utilizada se muestra en la figura 2.4.

Es importante resaltar que, por Ley de Ohm: $v_o = R_o i_o$. Dado esto, no se puede controlar i_o y v_o simultáneamente. Pero sí se puede hacer un controlador para cada uno, de forma que se pueda alternar la variable de control según los requerimientos de la aplicación. En consecuencia, se utiliza el control de voltaje de salida v_o o el control de corriente de salida i_o, mientras que el control de voltaje del condensador v_f está siempre activo.

2.5.1. Modelo Desacoplado

En los modelos presentados hasta ahora, existe acoplamiento. Dado que $\frac{dv_f}{dt}$ depende de s_1 y s_2, y por otro lado, $\frac{di_o}{dt}$ también depende de s_1 y s_2.

Para simplificar el modelo y evitar el fenómeno de acoplamiento, se utiliza el bloque D en la figura 2.4 que representa las ecuaciones de desacople (2.29) y (2.30):

$$u_i = \frac{d_1 + d_2}{2} \tag{2.29}$$

$$u_v = \frac{d_1 - d_2}{2} \tag{2.30}$$

Si se emplea la expresión de u_i (2.29) y u_v (2.30) en las ecuaciones del modelo conmutado (2.15) y (2.15), se obtiene que u_v controla directamente la variación de v_f (término desacoplado) (2.32). Mientras que u_i y u_v controlan la variación de i_o (2.31):

$$\frac{di_o}{dt} = \frac{1}{2L_o}((u_i + u_v)v_{dc} - 2u_v v_f - R_o i_o) \tag{2.31}$$

$$\frac{dv_f}{dt} = \frac{1}{C_f}(2u_v i_o) \tag{2.32}$$

Con esto se logra desacoplar una de las dos ecuaciones, lo que es suficiente para facilitar el diseño del controlador.

2.5.2. Sintonizar controladores

Aplicando la transformada de Laplace a las ecuaciones del modelo desacoplado (2.31) y (2.32), es posible encontrar la función de transferencia de la planta de corriente $G_o^{I_o}(s)$, planta de voltaje de $G_o^{V_o}(s)$ y planta de voltaje $G_o^{v_f}(s)$:

$$G_o^{I_o}(s) = \frac{I_o^o(s)}{u_i(s)} = \frac{V_{dc}}{2L_o + R_o} \tag{2.33}$$

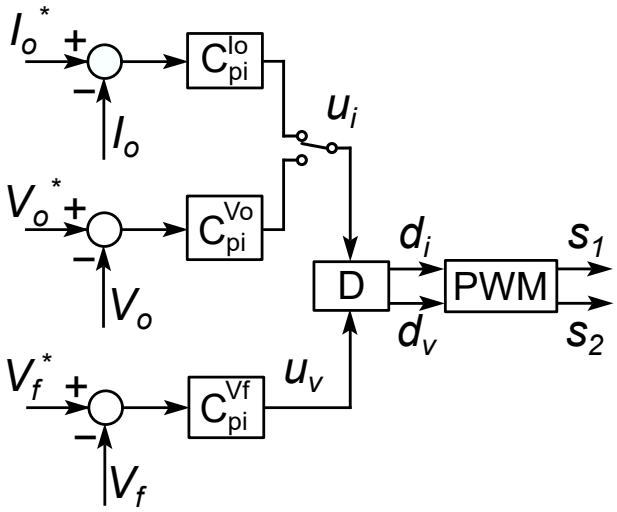


Figura 2.4: Diagrama del controlador par generar pulsos PWM del FCBC

$$G_o^{V_o}(s) = \frac{R_o I_o(s)}{u_i(s)} = \frac{V_{dc} R_o}{2L_o s + R_o} \tag{2.34}$$

$$G_o^{V_f}(s) = \frac{V_f(s)}{u_v(s)} = \frac{2I_o}{C_f s + 1} \tag{2.35}$$

En la herramienta SystemControlDesign de Matlab, se usan las expresiones de cada planta (2.33), (2.34) y (2.35) según el esquema de la figura 2.5 para encontrar los valores del controlador Proporcional Integral (PI) a utilizar. Si bien aquí se presentan las expresiones continuas de la planta, el modelo de la planta empleado en SystemControlDesign es el discreto (usando c2d()) en el cual se utiliza como tasa de muestreo del modelo la velocidad de cómputo del módulo DS1202 $T_{s1202} = 100[\mu s]$.

La expresión analítica de los controladores discretos PI para la corriente de salida, voltaje de salida y voltaje del condensador tienen la forma que se muestra en (2.36), (2.37) y (2.38):

$$C_p i^{i_o}(z) = \frac{K_p^{i_o} z + K_i^{i_o}}{z - 1} \tag{2.36}$$

$$C_p i^{v_o}(z) = \frac{K_p^{v_o} z + K_i^{v_o}}{z - 1} \tag{2.37}$$

$$C_p i^{v_f}(z) = \frac{K_p^{v_f} z + K_i^{v_f}}{z - 1} \tag{2.38}$$

Los valores de $K_p^{i_o}$, $K_p^{v_o}$, $K_p^{v_f}$, $K_i^{i_o}$, $K_i^{v_o}$ y $K_i^{v_f}$ se obtienen con SystemControlDesign utilizando los requerimientos de diseño de cada lazo. Considerando que los valores entregados por el software son de los controladores continuo, se utilizan las equivalencias (2.39) y (2.40) para obtener los valores del de los controladores discretos K_{pd} y K_{id} a partir de los valores de los controladores continuos K_{pt} y K_{it}.

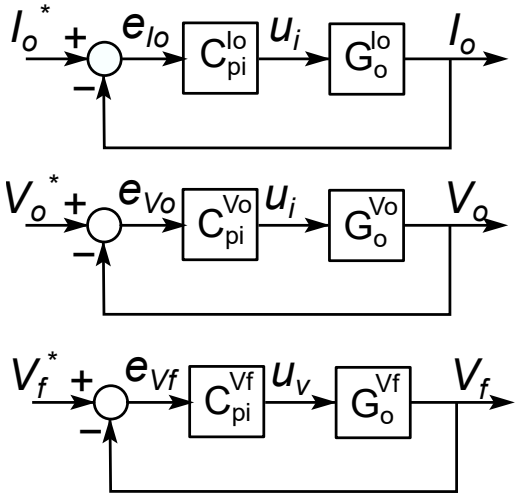


Figura 2.5: Esquema utilizado para sintonizar controladores

$$K_{pd} = K_{pt} \tag{2.39}$$

$$K_{id} = -(K_{pt} - K_{it}T_{s1202}) \tag{2.40}$$

2.5.2.1. Requerimientos de diseño para i_o y v_o.

El controlador está programado en el módulo DS1202 de la MLBX. La frecuencia de muestreo utilizada en este módulo es:

$$T_{s1202} = 100[\mu s]$$

$$f_{s1202} = 10[kHz]$$

Para tener un buen muestreo, se sintoniza $C_{pi}^{v_o}$ y $C_{pi}^{i_o}$ para que el ancho de banda del lazo $BW_{i_o v_o}$ sea la décima parte de la frecuencia de reloj del procesador. Por lo tanto:

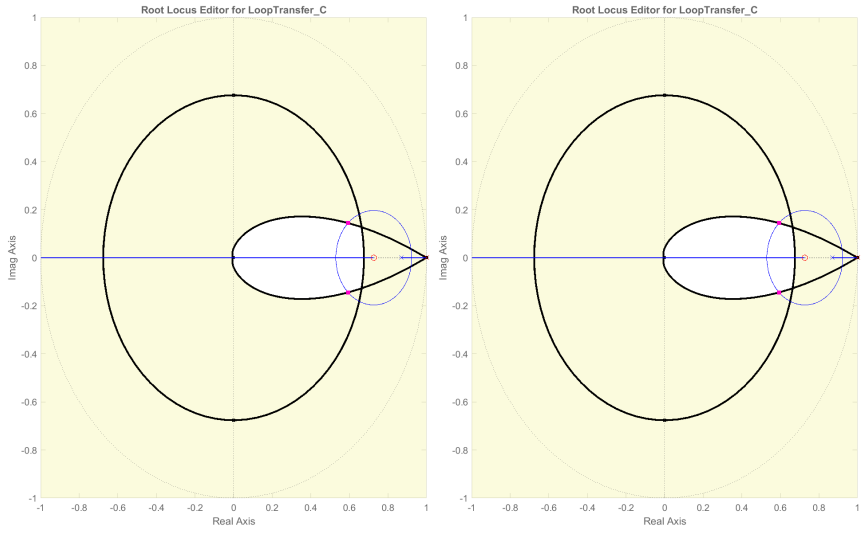
$$T_{i_o v_o} = 10T_{s1202} = 1[ms]$$
$$BW_{i_o v_o} = f_{s1202}/10 = 1[kHz]$$

Por lo tanto,el settling time utilizado es de $T_{i_o v_o} = 1[ms]$.

El valor coeficiente de amortiguamiento seleccionado es $\xi = 0{,}9[-]$, ya que se logra el menor sobrepaso en la respuesta a escalón provista por ControlSystemDesign.

Los requerimientos de $T_{i_o v_o}$ y ξ se ingresan a SystemControlDesign, en donde gráficamente (fig 2.6) se mueven los polos del lazo para cumplir con ambas restricciones.

A partir de la posición de los polos, se obtienen los valores de $K_p^{i_o}$, $K_i^{i_o}$, $K_p^{v_o}$ y $K_i^{v_o}$ presentados en la tabla 2.3



(a) LGR i_o (b) LGR v_o

Figura 2.6: Polos del lazo de corriente i_o y v_o

	Kp [-]	Ki[-]	ξ	Overshoot[%]	Settling Time [ms]
$C_{pi}^{io}(z)$	0.0275370	-0.020	0.9	13,6	0,95
$C_{pi}^{vo}(z)$	0.0097806	-0.0071	0.9	13,6	0,95
$C_{pi}^{vf}(z)$	0.017414	-0.01266	0.222	56,4	10

Tabla 2.3: Valores de controladores

2.5.2.2. Requerimientos de diseño para v_f.

El voltaje v_f del condensador depende solo de su respectiva actuación de voltaje u_v, y no se ve afectada por u_i como se ve en la ecuación (2.32).

En contraste, en (2.31) se observa que la actuación u_v relativa a v_f, afecta también a i_o y v_o. Por lo tanto, para el control de i_o y v_o, la actuación u_v que regula el voltaje del condensador v_f es una perturbación.

Por lo tanto, el ancho de banda BW_{v_f} del lazo condensador C_f debe ser 10 veces menor que $BW_{i_o v_o}$ para que la perturbación u_v no afecte a i_o y v_o. Quedando el ancho de banda y el tiempo de transiente de valores:

$$BW_{vf} = 100[Hz]$$

$$T_{vf} = 10[ms]$$

El coeficiente de amortiguación, no tiene un requisito específico pero se grafica el requerimiento de $\xi = 0{,}9[-]$ como referencia en la figura 2.7. Sin embargo, prima el requerimiento de transiente, por lo que el damping obtenido por defecto es de $\xi = 0{,}222[-]$ con un overshoot de 56[%], indicado en la tabla 2.3.

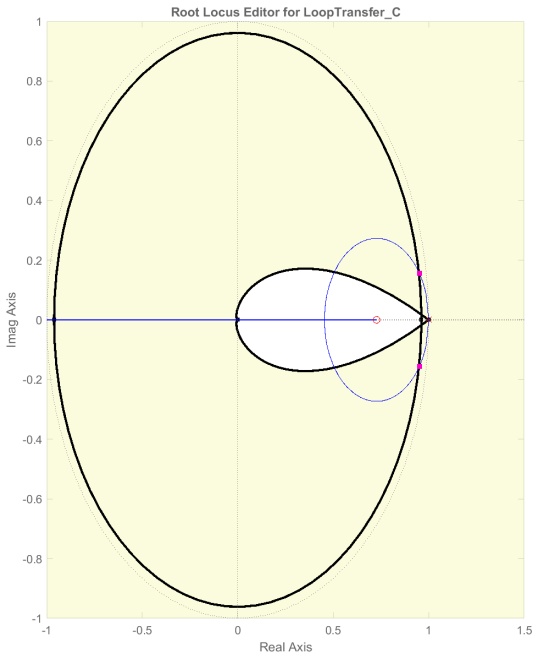


Figura 2.7: LGR v_f

2.5.3. Phase Shifted PWM

Para modular los ciclos de trabajo d_1 y d_2 en el esquema presentado, se utiliza modulación por ancho de pulso desfasada en ángulo.

Dado que existen dos variables (d_1 y d_2), el desfase utilizado en la señal portadora es de $\pi[rad]$.

La portadora elegida es una señal triangular de frecuencia mayor al ancho de banda del controlador $BW_{i_o v_o} = 1[khz]$, seleccionando en este caso una frecuencia de la portadora de $f_{carry} = 100[kHz]$. La portadora oscila entre 0[-] y 1[-].

Otra razón para usar una portadora de frecuencia tan elevada. Es contribuir en estudios y proyectos de transistores GaN del laboratorio PowerLab, los cuales son utilizados para operación a 100[kHz] o más.

2.5.4. Tiempos muertos

Los tiempos muertos para evitar cortocircuitos en el lapso en que el semiconductor conmuta de un estado a otro. De esta manera se logra una conmutación segura.

En este caso, se utiliza un tiempo muerto t_{dead} de 100 ciclos de reloj del módulo FPGA DS1302 de la MLBX ($t_{fpga} = 10[ns]$). Por lo tanto, $t_{dead} = 100 t_{FPGA} = 1\mu[s]$. Por lo tanto el t_{dead} es una décima parte del período de la carrier.

Se utiliza 1[us] de tiempo muerto tanto en el encendido como en el apagado del semiconductor. Sin embargo, dado que el modelo (2.15) (2.16) no depende de $\bar{S}_1$ ni $\bar{S}_2$, no se implementa el tiempo muerto de bajada que tiene que ver con estas señales.

Capítulo 3

MICROLABBOX

LAS plataformas de simulación enfocadas a sistemas eléctricos son parte del mercado de equipos utilizados para desarrollar hardware y software en laboratorios, universidades o empresas I+D. Esto se debe a que hay dinámicas eléctricas y de control que no pueden ser simuladas en tiempo real en computadores de uso personal debido a la alta tasa de procesamiento de datos o de adquisición de datos. En cuyo caso, se adquieren estas plataformas de simulación en tiempo real para simular distintos modelos de hardware, algoritmos de control y evaluar su rendimiento a distintas tasas de muestreo. Con lo cual es posible tomar mejores decisiones sobre los requerimientos del hardware para un controlador digital para que controle y rinda como se espera en su particular aplicación.

Existe un basto mercado de plataformas que permite elegir entre modelos que varían en velocidad de cómputo, cantidad de puertos, tipos de puertos, tasa de muestreo, lenguaje de programación, interfaz de programación, HMI, trabajo remoto e interacción con otros sistemas entre otros parámetros.

Este capítulo explica las características del equipo de dSpace modelo MicroLabBox 1202/1302, utilizado en este estudio. El cual permite diseñar y programar modelos de la planta en Simulink, así como también implementar controladores en C o en Verilog.

En la plataforma se realiza la implementación en el módulo DS1302 del modelo discreto desacoplado del FCBC descrito en el capítulo anterior. El controlador es implementado en el módulo DS1202, en donde se programan en C los controladores PI calculados. Finalmente, los pulsos de disparo de la PWM son generados en el módulo DS1302, siendo éste el único módulo capaz de generarlos a la frecuencia de 100[kHz].

Los diagramas conseguidos con el análisis y las prácticas recomendadas en este capítulo se adjuntan en el apéndice 2. Por otro lado, los códigos en C utilizados se encuentran en el apéndice 3.

3.1. Hardware MLBX

La plataforma MicroLabBox, cuyas características principales se enlistan en la figura 1.2.

En el diagrama de la figura 3.1, se muestra la arquitectura interna de la MLBX, en donde se distingue como el hardware se divide entre módulo DS1202 y módulo DS1302, existiendo

un único canal de comunicación entre módulos, el cual corresponde al Bus local de 32-bit.

El PC se conecta mediante ethernet, el cual tiene tres funciones, la primera es cargar el modelo del módulo DS1202 al procesador principal.

La segunda es, usar la conexión con el procesador principal, para cargar el modelo de la planta a simular en el módulo DS1302 que contiene la FPGA.

La última función del PC, es monitorizar el sistema mientras la simulación en tiempo real está en marcha utilizando la HMI ControlDesk. Para realizar esta función, se utiliza el Co-procesador de comunicación.

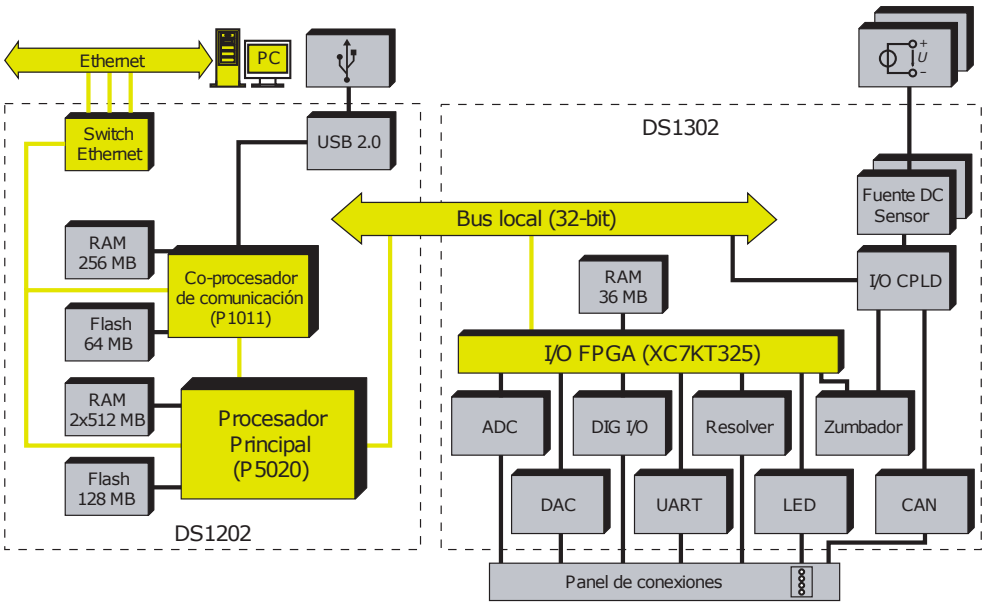


Figura 3.1: Arquitectura de la MLBX

3.1.1. Software para MLBX

Para programar el controlador del FCBC en el módulo DS1202, se utiliza Simulink como interfaz para implementar un controlador previamente programado en .C.

Existe la opción de usar C en una IDE genérica para describir el modelo a implementar en la FPGA utilizando la librería [6] que transforma código C a VHDL. En este caso, se deben crear los archivos de .sdf y .trc manualmente para enlazar con la HMI y así tener una interfaz interactiva. Sin embargo, el controlador implementado en este trabajo de memoria de título consiste en usar Simulink, con el bloque S-funcion builder, el cual se programa con C.

De esta misma forma, para describir hardware en el módulo DS1302 que contiene la FPGA, se usa Simulink, en donde se usan las librerías XSG de Xilinx [5] para crear modelos de sistemas en FPGA, así como la librería Real-Time Interface [18] de dSpace para los buses y puertos de la MLBX.

Sin embargo, hay otras opciones disponibles para describir hardware en el módulo 1302,

como por ejemplo utilizar una IDE de C y aprovechar librerías especiales que convierten el código en C a un lenguaje de FPGA como Verilog o HDL.

Finalmente, el método que utiliza los recursos de la FPGA de manera más óptima a bajo nivel, es directamente utilizar Verilog para describir, en este caso, el modelo del convertidor. Dado que los métodos expuestos anteriormente, si bien son funcionales y requieren una curva de aprendizaje más corta, no están tan depurados en ámbitos de conversión de un lenguaje a otro.

3.2. Implementación

Para implementar el controlador en el módulo DS1202, el modelo del FCBC en el DS1302 y los pulsos de disparo en el módulo DS1302, se ha tenido en cuenta el esquema de la figura 3.2, en donde mediante la HMI, se monitorizan las señales presentes en DS1202. Para visualizar las señales del módulo DS1302, se realiza un muestreo de estas a frecuencia del módulo DS1202, en caso de que las señales conmuten muy rápido, como las asociadas a los pulsos de disparo y el voltaje v_c multinivel del convertidor, se puede solo calcular el valor medio y desplegarlo en la HMI.

Si bien, se habla de que el controlador está implementado en el procesador del módulo 1202, no es así por completo. En los sistemas de control de hardware real, también existe el problema de que el procesador es demasiado lento para generar los pulsos de disparo a alta frecuencia necesarios para los semiconductores. Es por esto que se deja la generación de los pulsos en la FPGA, ya que es el mejor uso de los recursos para la simulación y para el control real del circuito.

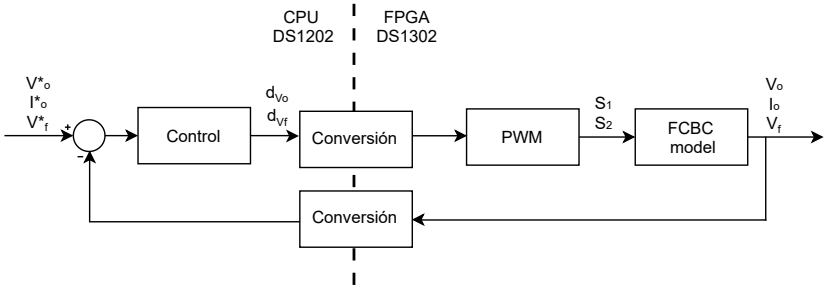


Figura 3.2: Distribución de módulos DS1202 y DS1302 en simulación FPGA-in-the-Loop

3.2.1. Módulo DS1202

En el módulo DS1202, se implementa en Simulink el esquema de control de la figura 2.4 diseñado en el capítulo anterior, utilizando el bloque S-function builder para programar en C, y realizando el desacople con bloques.

En donde las referencias de los controladores son trenes de pulsos y escalones que son editadas desde la HMI para realizar las pruebas de simulación. En el bloque de programación en C "S.Function Builder", se usan los valores de los controladores discretos sintonizados en 2.36, 2.37 y 2.38. En donde, se puede cambiar entre control de i_o o v_o. Todo esto bajo el esquema 2.4 del capítulo en que se explica el control del FCBC, en este caso el controlador.

Adicionalmente, se puede activar y desactivar el control de v_f con fines de análisis del comportamiento del sistema

El reloj del módulo DS1202 está configurado en $T_{s1202} = 100[\mu s]$ ($F_{s1202} = 10[kHz]$), ya que a frecuencias más altas, existe el riesgo de que la frecuencia no se mantenga constante debido a que se le exige demasiado al procesador. Tal como se explica en el capítulo "Modelo de FCBC", esta es la consideración de diseño del BW del control de v_o y i_o. Según el Turn-Around-Time medido en los resultados de simulación, se determinará el tiempo de operación del módulo DS1202, para ver si se puede aumentar la frecuencia de muestreo del módulo y aumentar el ancho de banda del controlador.

En la figura B.6, están presente el bloque "Processor Setup" en donde se vincula el archivo .bit del bitstream a emplear en el módulo DS1302 para la simulación, que en este caso está contenido en el bloque de nombre "M" de la figura B.6.

Así como también, se tienen bloques de envío de datos hacia la FPGA "PROC_XDATA_WRITE_BLX" por los cuales se envían los ciclos de trabajo d_1 y d_2 que deben utilizarse. Mientras que por los bloques "PROC_XDATA_READ_BLX" se reciben las realimentaciones de control v_o, i_o y v_f, así como otras señales para verificar el correcto funcionamiento de la simulación. Estos bloques pertenecen a la libería Real-Time interface de dSpace, con la que se puede emplear el bus de 32-bit para comunicar ambos módulos de la MLBX

Esto es lo utilizado en la simulación en tiempo real, pero para análisis del comportamiento del modelo del FCBC presente en el módulo DS1302, se realizan simulaciones en "tiempo de CPU", es decir, más lento que tiempo real, para lograr muestrear los valores instantáneos de S_1, S_2, la señal carrier, y otros voltajes de alta frecuencia presentes en el modelo. Estas señales están presentes en el módulo DS1302 y son analizadas directamente en los scopes de Simulink, sin hacer uso de la HMI.

3.2.2. Módulo 1302

En el módulo DS1302, posee una capacidad de cómputo mayor al módulo DS1202. El reloj del chip Xilinx de la FPGA es de $T_{s1302} = 10[ns]$, lo que es equivalente a $F_{s1302} = 100[Mhz]$, es decir, 10.000[-] veces la frecuencia configurada en el módulo DS1202.

La potencia de cómputo del módulo DS1302, se utiliza en dos cosas, la generación de pulsos de alta frecuencia de control, y la representación en tiempo real del modelo del FCBC para que sea fidedigna con los tiempos de respuesta del un hardware real.

Nótese que la implementación en Simulink de la FPGA se muestra en la figura B.5, en donde está el subsistema PWM y el subsistema FCBC, análogo a lo presentado en la figura 3.2.

Existen ciertas consideraciones generales de trabajo, como la configuración de un reloj interno dado por la latencia más grande de los distintos bloque combinacionales. En este caso, se tienen los bloques combinacionales, PWM y modelo FCBC, así como también se podría considerar como bloque combinacional el módulo DS1202, ya que también hay intercambio de datos.

Se tienen registros con enable que son controlados por el reloj modelado en B.4 (observado en la figura B.5 como subsistema). También se tienen registros dispuestos justo antes de los bloques "FPGA_XDATA_READ", dado que eso indica el manual de buenas prácticas del Blockset de Xilinx para FPGA [19], usando dos bloques antes de una escritura, aunque pueda parecer redundante. Cosa importante también es que los tipos de datos sean concordantes en la representación numérica, es por eso que se usa un bloque "convert" según sea el caso de las señal.

El bloque "Assert" se utiliza para verificar que el formato de representación numérica se

cumpla, en caso contrario se activa un Flag de error. Este bloque es usado recurrentemente para verificar la correcta operación de los datos, tal como se indica en el manual de Xilinx [19].

El módulo DS1302 recibe las entradas d_i y d_v en formato Fixed unsigned de 64 bit, con 32 bit para la parte decimal. Principalmente esta es la representación numérica utilizada, dado que se indica en la documentación de la librería XSG que este formato funciona mejor en relación a flotante.

En el subsistema PWM se utiliza un comparador, resultando un dato booleano, el cual es conveniente también por el uso de tiempos muertos como se ve en la figura B.2. Después de utilizar las compuertas AND de los tiempos puertos, se obtienen los pulsos S_1 y S_2 que se utilizan en el subsistema FCBC, operando nuevamente en punto fijo. Los datos i_o, v_o y v_f se envían como flotante de doble precisión al módulo DS1202.

3.2.2.1. PWM

Para generar la carrier, se utiliza el esquema de la figura B.1. En este, se tiene un contador up/down de limites $\frac{0,5}{f_{carrier} f_{s1302}}$ y 0 detectados por un FFsr. De esta forma, se tiene una carrier de alta resolución con 1.000[-] muestras por período.

Luego, se escalan los valores del contador a una amplitud unitaria, que tenga como límite superior 1[-] y límite inferior 0[-]. Con lo que se tiene lista la carrier que se compara con d_1 para generar s_1, A partir de esta , se genera una carrier desfasada en 180° para compararse con d_2 y generar s_2.

En la figura B.2, se muestran la implementación del control PWM. En donde se compara la señal de carrier generada con los ciclos de trabajo dados por el controlador en el módulo DS1202.

Realizada la comparación con los bloques AND, se implementan los tiempos muertos con valores de $t_{dead} = T_{carrier}/10 = 1\mu[s]$

3.2.2.2. Modelo FCBC

Se implementa el modelo discreto de las ecuaciones (2.21) y (2.22), como se muestra en la figura B.3.

Mediante la herramienta Timing Analysis, se verifica que se respeten los tiempos de procesamiento en los datapath, resultando en que se tiene una latencia total del modelo FCBC de 19[ciclos]. Por lo tanto, se configura el reloj que controla los registros B.4, que generan las señales i_{ok+1},v_{ok+1} y v_{fk+1} a partir de i_{ok},v_{ok} y v_{fk} en una latencia de 20[ciclos] de $T_{s1302} = 200[ns]$.

Es importante notar que en la implementación del modelo del FCBC según el método de Euler-Forward, incluye un término multiplicativo T_s, correspondiente en este caso a la latencia del sistema $T_{sFCBC} = 200[n]$.

Se puede calcular la latencia del módulo DS1302 completo, dado que se tiene 20[ciclos] para el FCBC, 3[ciclos] en PWM, y 4[ciclos] más por los registros de "buenas prácticas". Dando un resultado de 27[ciclos] de reloj $T_{s1302} = 10[ns]$ que corresponden a $T_{FCBC-PWM} = 270[ns]$.

La latencia $T_{FCBC-PWM}$ corresponde a la latencia del sistema completo simulado en el módulo DS1302, siendo esta 370 veces menor que el reloj del módulo DS1202 ($T_{s1202} = 100[us]$), es decir, el modelo del módulo DS1302 itera 370 veces antes de que el controlador del módulo 1202 cambie el valor de la actuación u_i o u_v.

Tiempo	Variable	Valor numérico
Latencia del módulo DS1302	T_{s1302}	10 [ns]
Latencia del modelo FCBC (k)	T_{FCBC}	200[ns]
Latencia del subsistema PWM y FCBC	$T_{FCBC-PWM}$	270[ns]
Periodo de los pulsos de disparo S_1 y S_2	$T_{carrier}$	10[us]
Turn-Around-Time	T_{TAT}	10[us]
Tiempo de muestreo del módulo DS1202	T_{s1202}	100[us]
Ancho de banda del control de i_o y v_o	BW_{i_o,v_o}^{-1}	1[ms]
Ancho de banda del control de v_f	BW_{i_o,v_o}^{-1}	10[ms]

Tabla 3.1: Resumen de los valores temporales considerados en el diseño del modelo y del controlador

Capítulo 4

RESULTADOS

En este capítulo se exponen los resultados de simulación de los procedimientos y decisiones de modelado de los capítulos anteriores.

Para realizar las pruebas experimentales en tiempo real, se utiliza la herramienta de software ControlDesk Next Generation de Dspace diseñada como HMI de la MicroLabBox. Esta HMI es personalizable por el usuario, en este caso, se creó el entorno de la figura 4.1 que permite alternar entre control de voltaje o control de corriente, así como, regular su señal de referencia e introducir una perturbación en forma de tren de pulsos. Adicionalmente, esta agregada la opción de desactivar y activar el control de voltaje v_f del condensador para realizar pruebas de activado y desactivado. Estos cambios son visibles en gráficos dispuestos en la misma HMI.

Usando también la HMI, se simula el convertidor en el punto de operación $V_o^o = 375[V]$ diseñado para la aplicación de carga de batería con los valores de la tabla 2.2. Observando si el comportamiento de la PWM corresponde al calculado analíticamente en el punto de operación.

La HMI (limitada por el hardware del módulo DS1202 de la MLBX) tiene un período de muestreo T_{sHMI} igual a $100[\mu s]$, igual a las señales del módulo DS1202 que se actualizan a un período de $T_{s1202} = 100[\mu s]$.

Lamentablemente, en tiempo real no es posible muestrear señales del módulo DS1302 lo suficientemente rápido como para ver las variaciones de ancho de pulso en s_1, s_2 y v_c generadas por el control PWM a $T_{carrier} = 10[\mu s]$, debido a que los pulsos de disparo tienen un período menor al periodo de muestreo de la HMI.

En consecuencia a esta problemática, se han realizado simulaciones en MATLAB del modelo en tiempo de CPU (no en tiempo real) para visualizar el correcto funcionamiento de lo modelado en la FPGA del bloque DS1302, en especial los pulsos s_1, s_2 y el voltaje multinivel v_c, ásí como el correcto funcionamiento de la PS-PWM.

Para tener una idea del comportamiento del voltaje v_c en tiempo real, se grafica su valor promedio utilizando bloques en Simulink en el módulo DS1202. Este valor promedio se obtiene a partir de la expresión (2.24), que depende de los ciclos de trabajo d_1 y d_2 generados por los controladores PI.

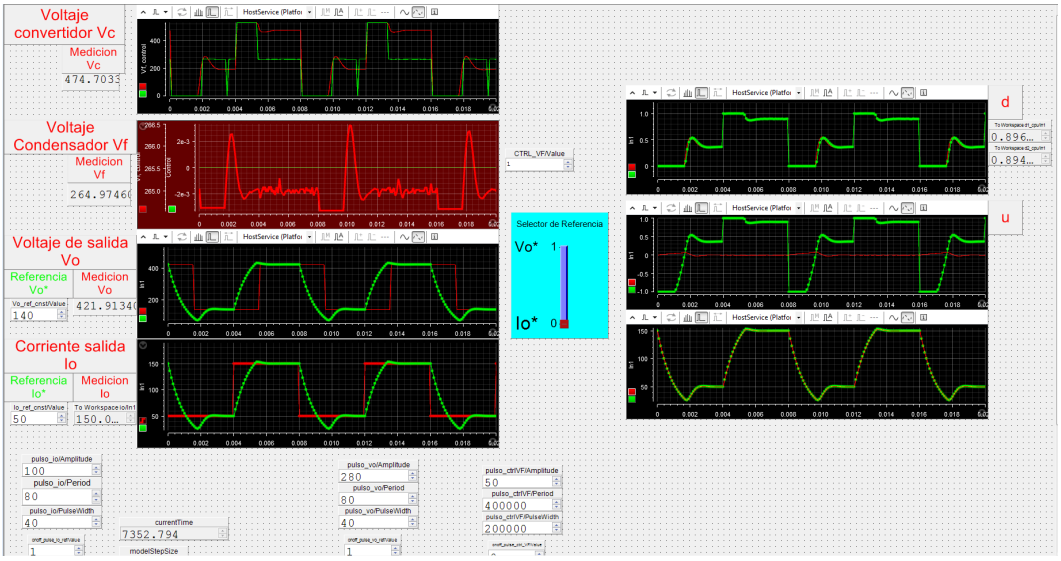


Figura 4.1: Panel de ControlDesk para pruebas en tiempo real. Los gráficos corresponden a una prueba de seguimiento de referencia i_o^* ante un tren de pulsos.

4.1. Resultados en tiempo de CPU

Utilizando la interfaz de Simulink para simular de forma offline en tiempo de cpu las variables de interés, se obtiene la figura 4.2 que muestra el transiente de encendido y la figura 4.3.

La simulación en tiempo de CPU, mostrada en la figura 4.2 comienza desde el instante 0[s] en que el condensador está completamente descargado, aunque el voltaje V_{dc} alimenta con 530[v] constantes.

En menos de 2[ms], la corriente i_o se estabiliza en la referencia de 50[A].

El voltaje del condensador se carga en 17.5[ms], llegando a la referencia $v_f^* = \frac{v_{dc}}{2} = 265[V]$.

Los resultados en estado estacionario de las señales se muestran en detalle en la figura 4.3 para medir el ancho de los pulsos del control PWM.

El voltaje del convertidor, para este nivel de operación conmuta entre $\frac{v_{dc}}{2}$

La actuación de u_v del voltaje del condensador se hace 0[-], mientras que la actuación de corriente u_i se estaciona en 0.37[-], al igual que los ciclos de trabajo de los semiconductores.

Es posible observar que para el ciclo de trabajo $d_1 = d_2 = 0{,}37[-]$. los pulsos de disparo s_1 y s_2, tienen un tiempo de encendido de 2,7[μs], ya que también está presente el tiempo muerto de 1[μs].

El modelo descrito a partir de las ecuaciones (2.15) y (2.16) no utiliza las señales $\bar{s_1}$ ni $\bar{s_2}$ por lo que no se considera necesario generarlas ni graficarlas.

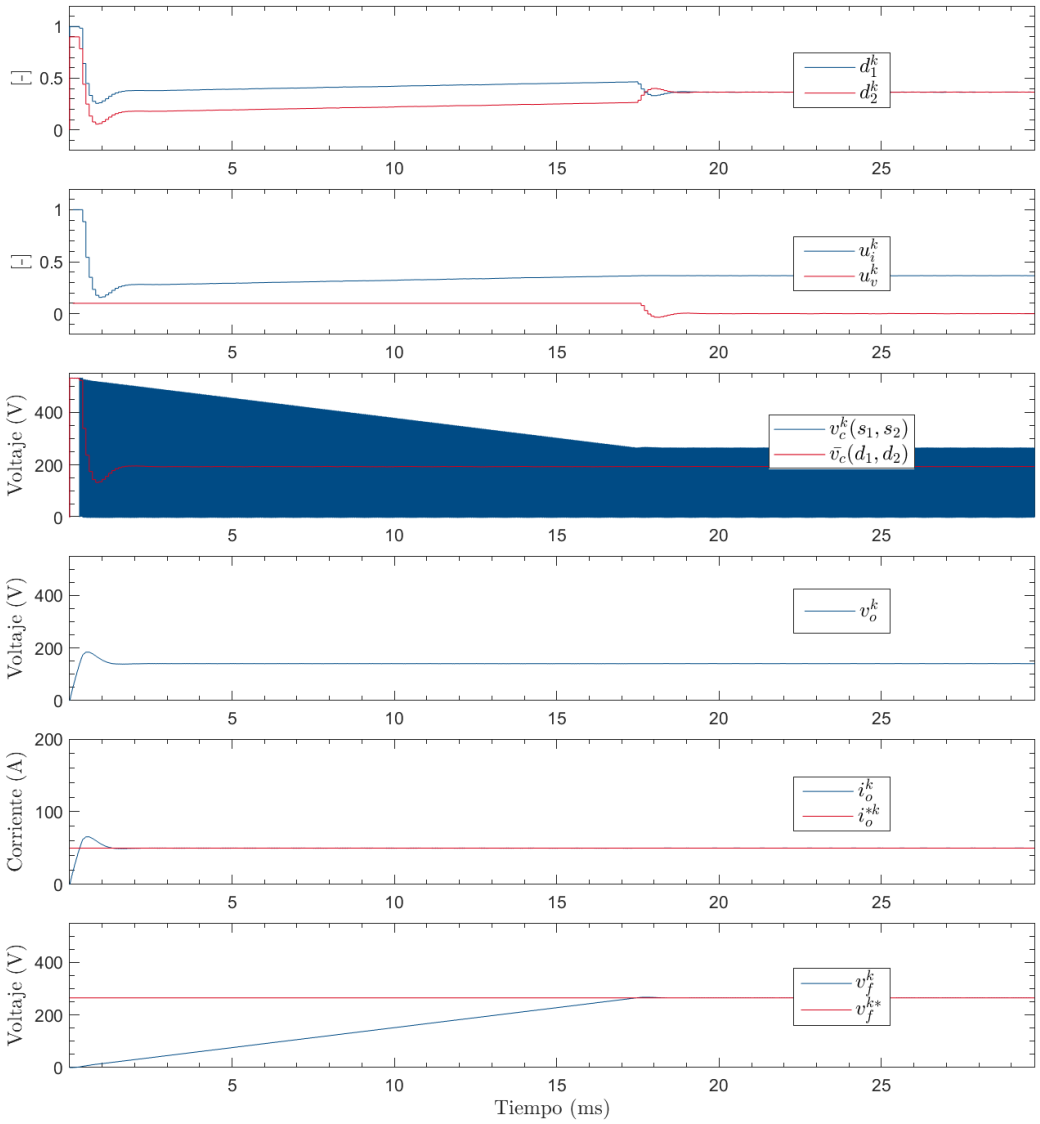


Figura 4.2: Simulación offline. Transiente de encendido del modelo.

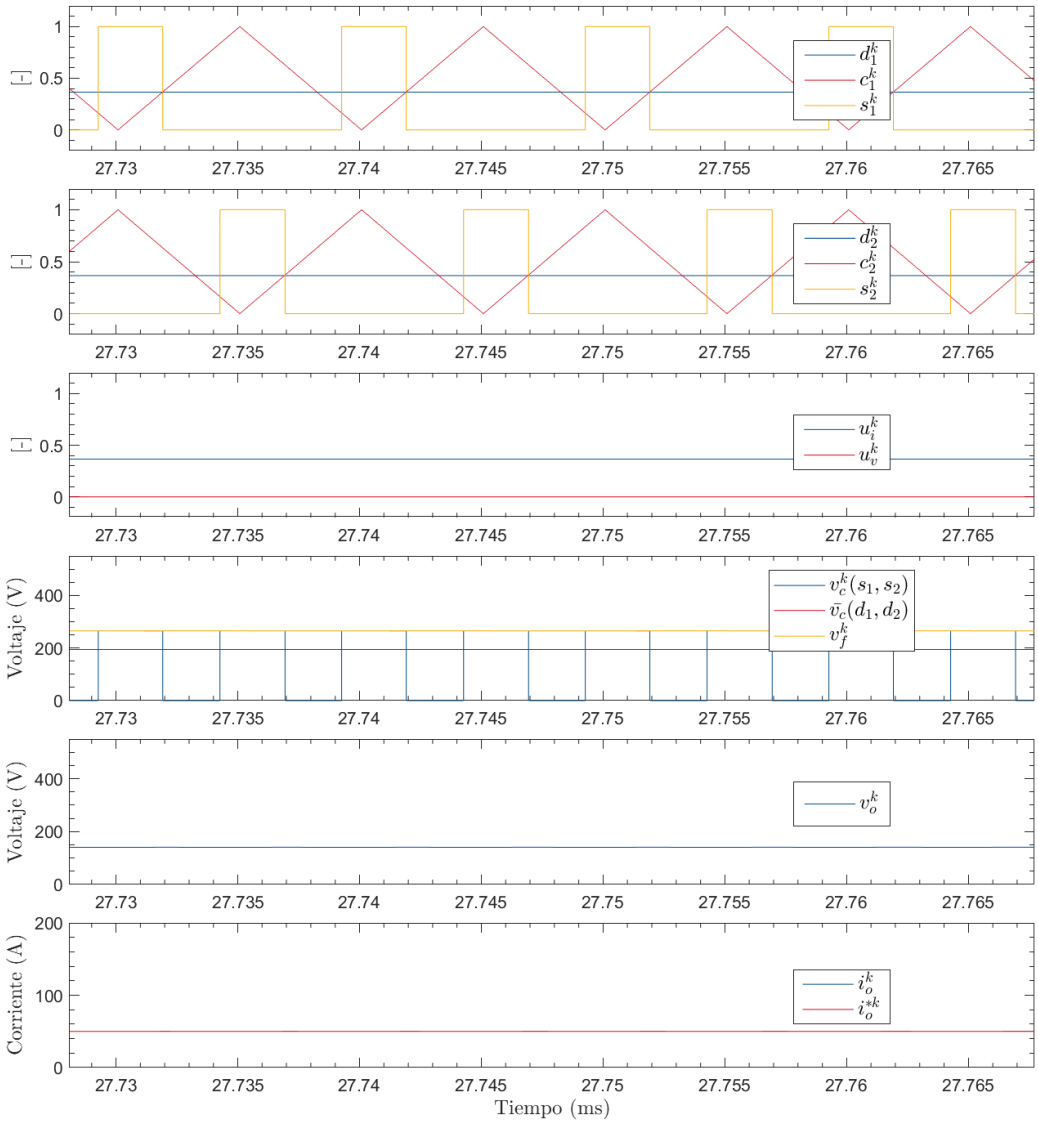


Figura 4.3: Simulación offline. Estado estacionario.

4.2. Resultados en tiempo real

Utilizando la HMI ControlDesk para simular en tiempo real el modelo de segundo orden del convertidor, se obtiene como resultado la figura 4.4 en estado estacionario operando para valores del punto de operación descrito en la tabla 2.2.

Existen diferencias entre el valor analítico y el valor de la simulación, producto de la incorporación del tiempo muerto de 1[μs]. El valor analítico de los ciclos de trabajo d_1 y d_2 es 0.707[-], mientras que el valor simulado es 0.808[-] logrando compensar el tiempo muerto y alcanzando la referencia de voltaje en 375[V].

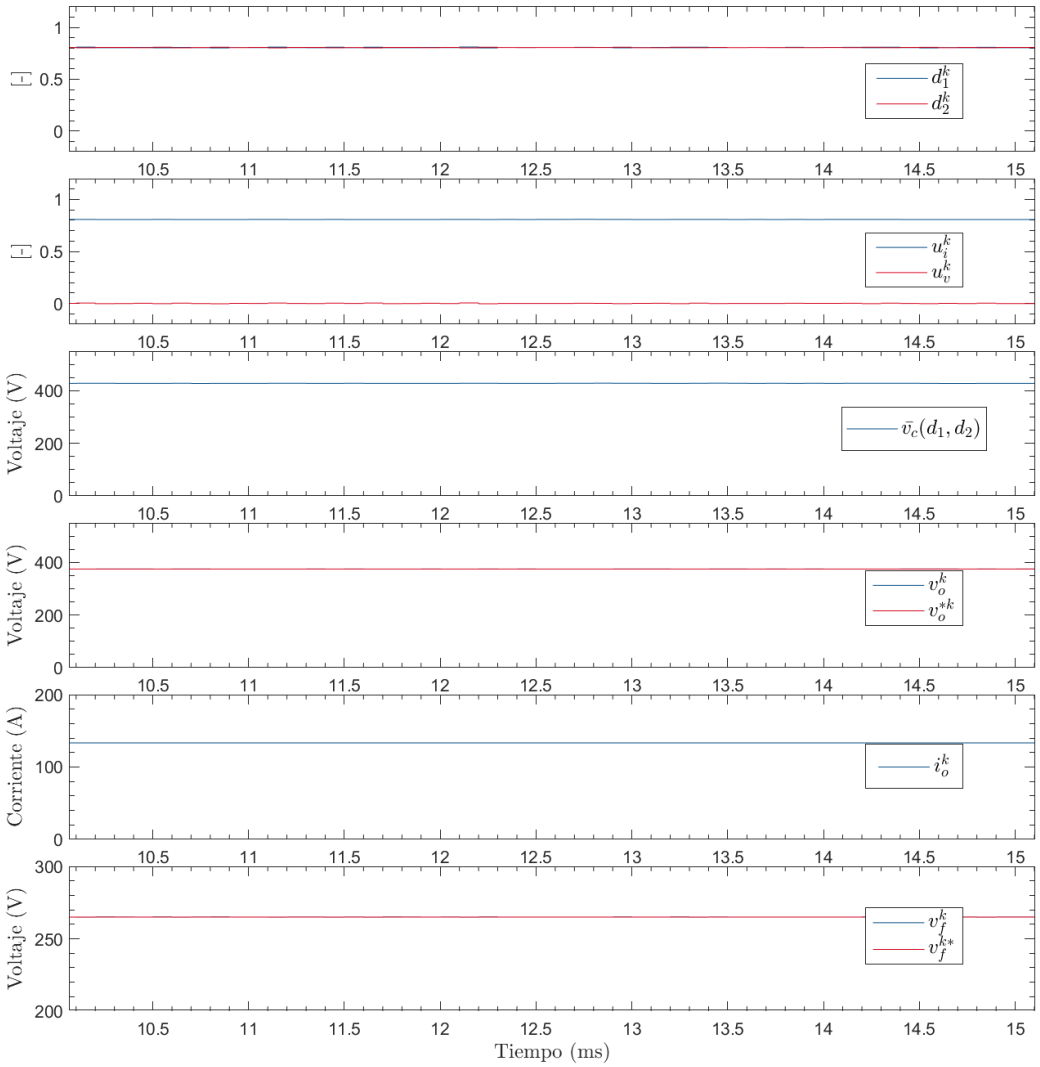


Figura 4.4: Simulación en el punto de operación $v_o^o = 375[V]$

Se realizan simulaciones de seguimiento ante cambios de referencia en la figura 4.5 para el lazo de corriente de salida i_o y en la figura 4.6 de voltaje de salida v_o.

Los tiempo de respuesta del controlador para llegar a estado estacionario son concordantes con la simulación offline en tiempo de cpu.

Se observa el desacople de las actuaciones u_i y u_v, que a su vez tienen la misma forma de onda de i_o, vo y v_f, respectivamente.

El voltaje v_f del condensador perturba en más de 0.5[v] por el control de i_o, v_o.

Existe una proporcionalidad entre i_o y v_o, correspondiente al valor de la resistencia $R_o = 2{,}8125[\Omega]$. Es por esto que ambas simulaciones 4.5 y 4.6 son similares.

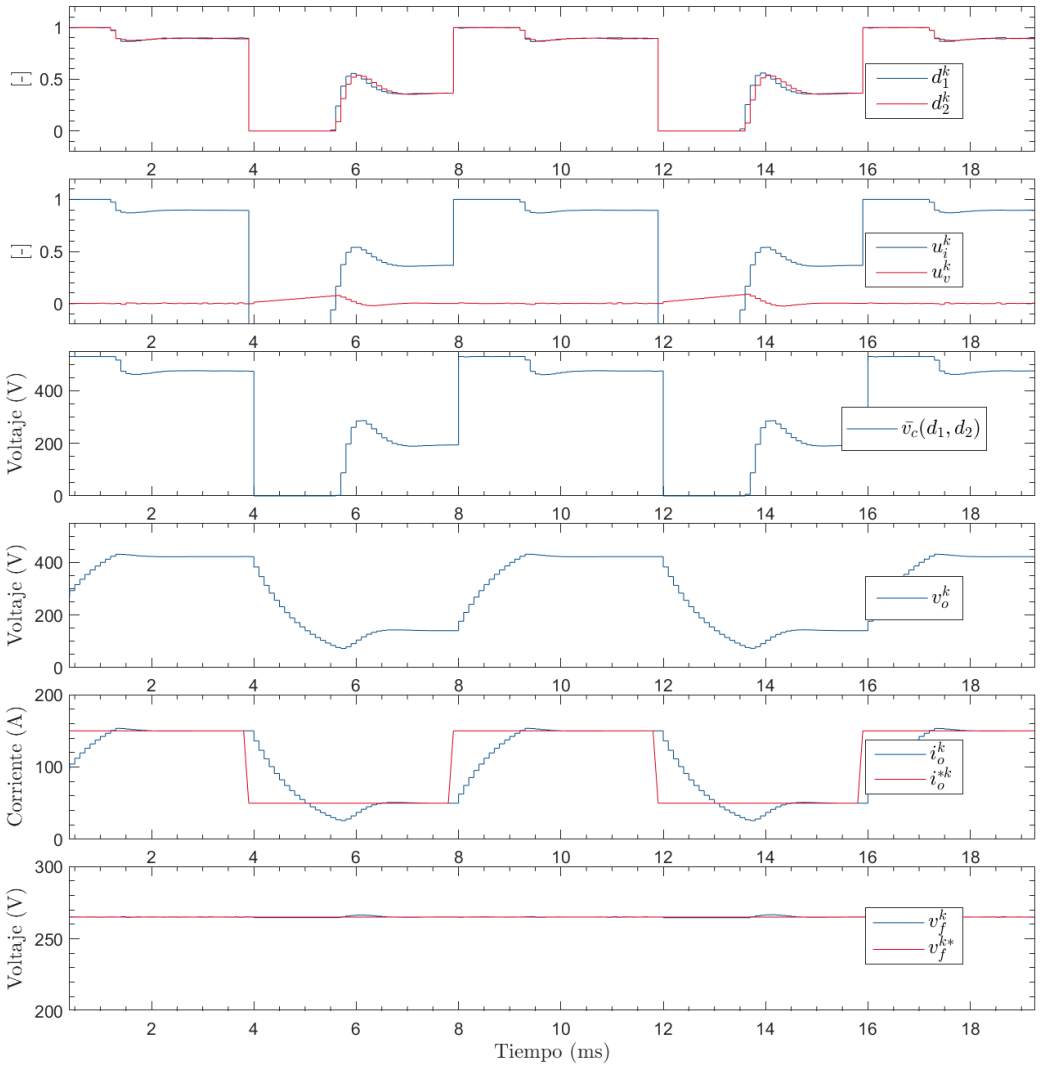


Figura 4.5: Comportamiento de la corriente i_o ante un cambio de referencia

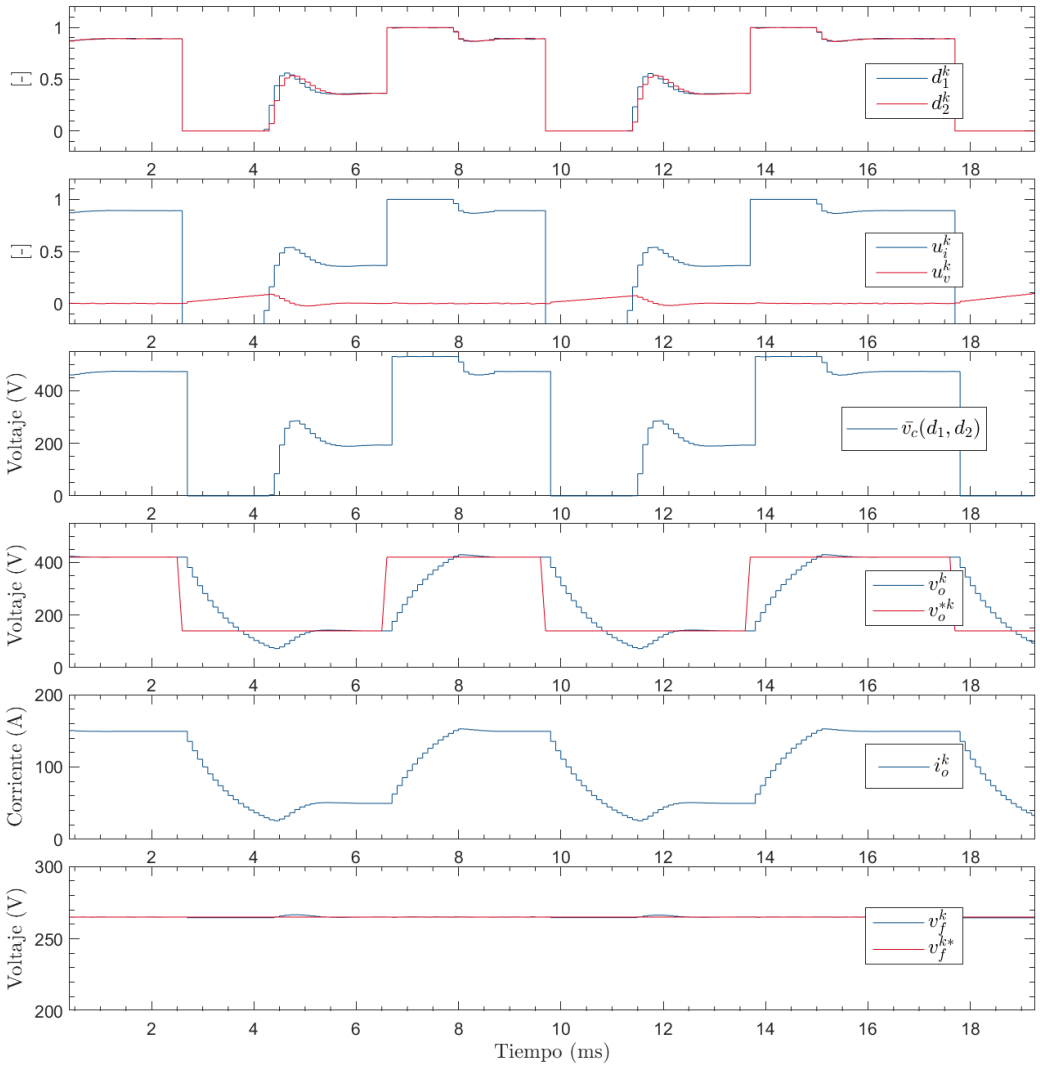


Figura 4.6: Comportamiento del voltaje i_o ante un cambio de referencia

Otro experimento se realiza en la figura 4.7, en el cual el convertidor se encuentra siguiendo la referencia de 50[A] de corriente de salida i_o, mientras mantiene el voltaje v_f del condensador C_f estable en 265[V].

Para verificar la necesidad del PI diseñado para el voltaje v_f, se desactiva su control forzando por medio de la HMI la actuación u_v a 0[-]. Esto genera que el condensador se descargue mientras esté el control desactivado. En este lapso es evidente que el condensador se aleja de su valor estable cargándose hasta llegar a un voltaje de 240[V]. Al reactivarse el control de voltaje V_f, se vuelve a la referencia de control en 2[ms]. Durante todo el experimento, la corriente i_o no presenta perturbación alguna.

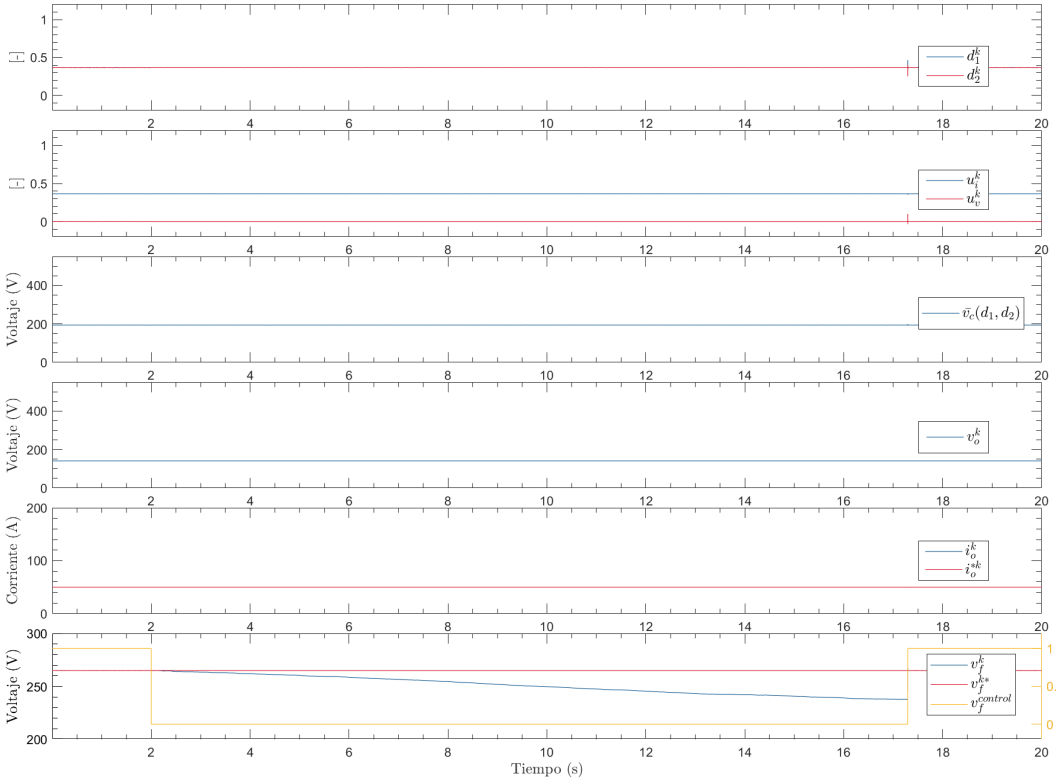


Figura 4.7: desactivación de control de voltaje v_f del condensador

Por medio de la HMI, es posible conocer el parámetro Turn-Around-Time, que indica la cantidad de tiempo que tarda el controlador, es decir, el módulo DS1202, en procesar las mediciones y generar una actuación a la planta. El valor medido es $T_{turn-around-time} = 4[\mu s]$, lo que indica que el módulo DS1202, configurado en este estudio para intercambiar datos con el módulo DS1302 cada $T_{s1202} = 100[\mu s]$ fue utilizado por debajo de su frecuencia de muestreo mínimo, el cual podría ser $4[\mu s]$ para este modelo.

Capítulo 5

CONCLUSIONES

EL trabajo realizado contempló la programación de alto nivel de un modelo de FCBC implementable en en la plataforma de simulación en tiempo real MLBX, utilizando la librería XSG para el entorno de Simulink que es compatible con la FPGA de Xilinx.

Los módulos CPU y FPGA de la MLBX fueron utilizados satisfactoriamente en un esquema de simulación FPGA-in-the-loop.

Por medio de simulaciones offline es posible corroborar el correcto funcionamiento de las señales de disparo generadas por el control PWM necesario para la conmutación de los semiconductores del FCBC, las cuales contemplan tiempos muertos y una frecuencia de conmutación operable en transistores GaN. Además, es posible ver los diferentes niveles del convertidor.

Al realizar simulaciones en tiempo real, el modelo funciona correctamente en concordancia con las simulaciones offline y el análisis del lazo realizado. Comprobando entonces que el módulo DS1302 sirve para simular tanto modelos de convertidores de potencia y como señales de disparo PWM.

La HMI ControlDesk resulta un medio útil para realizar experimentos en tiempo real, sin necesidad de interrumpir una simulación se puede activar y desactivar controladores PI. También resultó posible cambiar la referencia de las señales controladas de forma manual o por trenes de pulso. Si bien la HMI tiene limitaciones de muestreo, se puede conocer el comportamiento de los pulsos de disparo a partir de el valor numérico del ciclo de trabajo, y del voltaje multinivel v_c del convertidor a partir de su valor medio.

Por medio de la HMI también resultó posible medir el tiempo que tarda el módulo DS1202 en generar una actuación a partir de las mediciones de la planta del módulo DS1302. A partir de este dato, se determina si se puede subir o bajar el ancho de banda del controlador para tener un control lo más rápido posible para el hardware en cuestión.

A partir de los resultados de simulación del modelo del FCBC de segundo orden, se evidencia el desacople de las actuaciones del controlador lineal, correspondiente a un modelo SISO.

5.1. Trabajo Futuro

Es posible, por medio del DAC de la MLBX y la tarjeta de disparo diseñada previamente [15] en el laboratorio, utilizar las señales de disparo de la PWM para controlar una Flying Capacitor. El DAC tiene una frecuencia de muestreo de 10[ns], lo cual es suficiente para representar pulsos de 10[μs].

Se deja a disposición para otras simulaciones en la MLBX, el manual del apéndice A. Facilitando la continuidad del proyecto y el trabajo de generación de prototipos mediante simulaciones Model-Based.

Para la aplicación de carga de batería expuesta, se puede realizar la una prueba en que se haga la carga de batería desde 0 % hasta 100 % de estado de carga (SoC), alternando entre carga por corriente y carga por voltaje. Para esto se puede programar el comportamiento en el módulo DS1202, aunque también de forma más sencilla, se puede realizar la prueba mediante la HMI sin editar el modelo existente.

Se puede simular una topología que permita más niveles de operación en el Flyng Capacitor y/o comparar su comportamiento con un Three-level-buck o el Neutral-point-clamped. El interés de estás topologías radica en que en aplicaciones de EV, el voltaje de las baterías eléctricas está elevándose, llegando a los 600[V].

Apéndice A

MANUAL DE USUARIO

Advanced Center
for Electrical and Electronic Engineering

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Universidad Técnica Federico Santa María

Laboratorio de Conversión de Energías Renovables

Tutorial MicroLabBox

Autor:
Felipe Osorio

Supervisor:
Christian Rojas

15 de febrero 2020

Índice

1. Material de apoyo

Se recomienda revisar el material generado en paralelo a este documento en https://youtu.be/CodR_I8dovU y https://youtu.be/h0jui8LohA4 .Al pie de los videos se encuentra documentación que puede resultar útil para esta plataforma. Ante cualquier inconveniente contactar a felipe.osoriov@sansano.usm.cl

2. Introducción: El equipo y su Software

Este documento es una simplificación de los manuales [1],[2],[3] y[4], en los que se explica en detalle como emplear Verilog, Simulink, ControlDesk y C para configurar el MicrolabBox de dSpace.

El hardware MicroLabBox incluye una FPGA y una DSP, se puede encontrar información más detallada en [5]. La arquitectura es Hardware-in-the-Loop (HIL), que consiste en generar un modelo de la dinámica de un sistema en FPGA y controlarlo usando la CPU en tiempo real.

MicroLabBox, top panel variant with BNC connectors

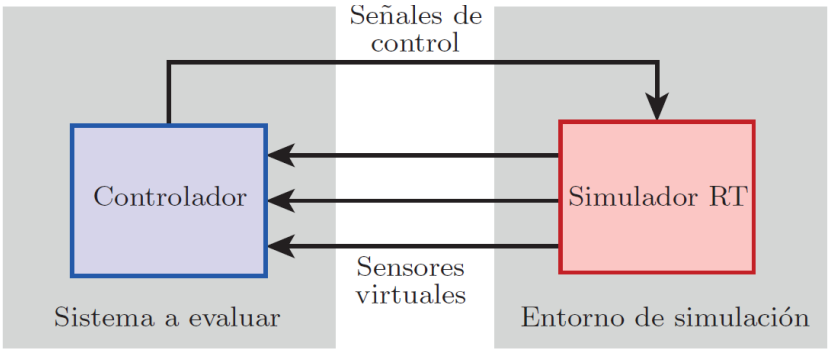


Figura 1: Diagrama Hardware In the Loop

El modelo del sistema se construye en Simulink con los bloques de la librería de Xilinx que son compatibles con la FPGA del MicroLabBox.

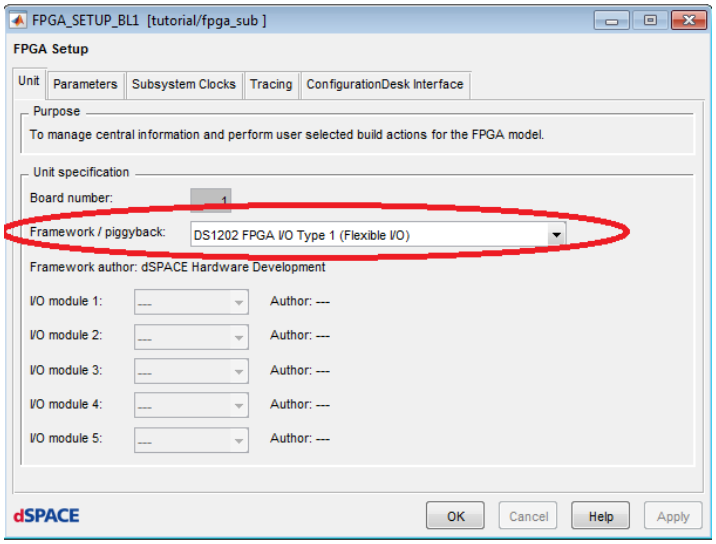
El controlador se implementa en Simulink con los bloques estándar, aunque al final de este documento se explica como se pueden generar los controladores en C e importarlos a Simulink.

Adicionalmente, se presenta la interfaz gráfica ControlDesk next Generation. Una cómoda HMI para ver el comportamiento de las señales en tiempo real y así prescindir de los scopes de Simulink.

3. Framework del proyecto

En Simulink, se debe configurar la FPGA con un Framework, en el manual se muestra cómo crear desde cero el Framework DS1302_XC7K325T. Sin embargo, el Framework realmente útil es *DS1302_XC7K325T_FLEXIBLEIO* ya que permite usar los periféricos del MicrolabBox tales como DAC, ADC y GPIO.

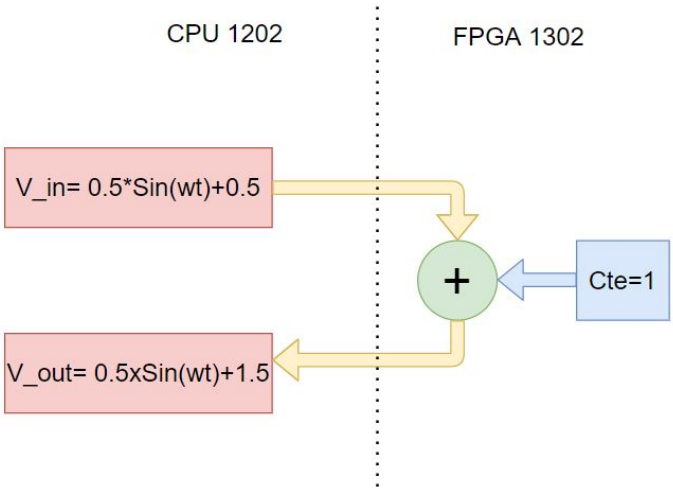
En caso de ya estar creado el Framework, basta con seleccionarlo en la ventana de *FPGA_setup_BL>framework*. Si no aparece el Framework deseado, entonces se deben seguir las instrucciones del manual[1]



.

4. Programación en Simulink

En esta sección se indican las instrucciones para llevar a cabo un proyecto simple en el cual se genera una señal senoidal en la CPU, se envía a la FPGA donde se le suma un valor constante y se retorna el resultado a la CPU.

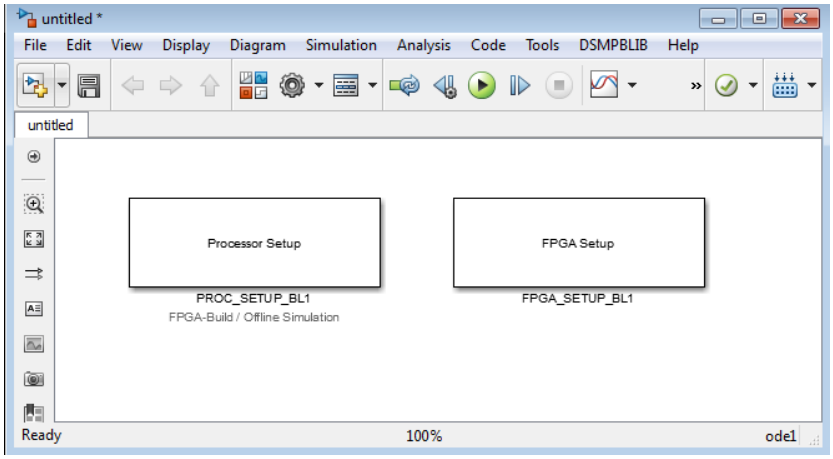


4.1. Programas y compatibilidad

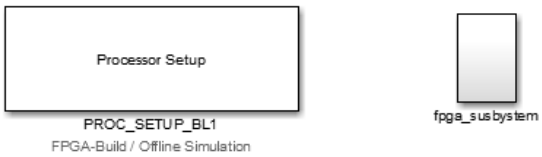
Si bien todo el proyecto se arma en la herramienta Simulink de Matlab, también es necesario tener instalado Vivado. En este ejemplo se usa Matlab 2014, Vivado 2016.1 en Windows 7 professional 64-bit. Se pueden usar otras versiones pero se debe revisar la compatibilidad entre Matlab, Vivado y el sistema operativo.

4.2. Diagrama de Bloques

Abrir matlab y simulink .
Abrir Simulink library Browser, en la categoría *dSpace RTI FPGA Programming Blockset*, hay dos subcategorías *FPGA INTERFACE* y *PROCESSOR INTERFACE*, los cuales contienen *FPGA_SETUP_BL1* y *PROCESSOR_SETUP_BL1*, respectivamente. Arrastrárlos a Simulink.



Enmascarar *FPGA_SETUP_BL1*



4.2.1. Modelo FPGA

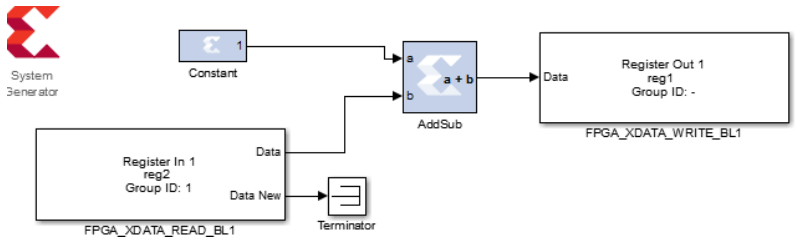


Figura 2: Modelo FPGA enmascarado

En Library Browser, seleccionar los bloques de las librerías Xilinx *Constant* y *Addsub*. Configurar el bloque *Addsub* con latencia 0 (combinacional). [1]

Para comunicar FPGA con la CPU se deben usar los bloques *FPGA_XDATA_READ_BL1* y *FPGA_XDATA_WRITE_BL1*, que representan los registros de lectura y escritura de la FPGA,respectivamente.Agregar un *Terminator* para evitar puertos al aire. [2]

4.2.2. Modelo CPU:

En el bloque *PROC_SETUP_BL1*, asignar el modelo FPGA creado en el punto anterior. Luego, seleccionar *Generate*(ver figura 3) para generar los bloques de lectura y escritura de la CPU. Arrastrar los bloques a la raíz del modelo Simulink.

[1]En el manual [2] se usa la herramienta Time Analysis para la cual la licencia puede no estar vigente. No obstante, se puede prescindir de esta herramienta configurando con cuidado las latencias de los bloques en la FPGA

[2]El bloque *System Generator* se genera automáticamente luego de presionar *Build Model* FPGA (más adelante en este tutorial)

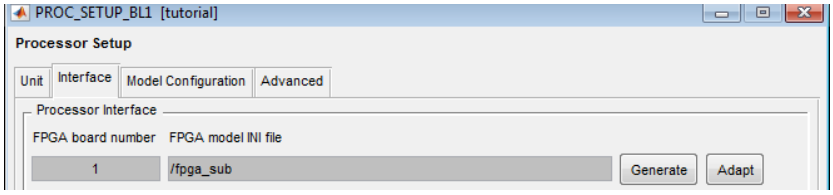


Figura 3: Comunicar modelo FPGA con la CPU

Seleccionar como entrada al modelo FPGA un bloque *Sin Wave* y configurarlo con los parámetros

- Amplitud 0.5
- Bias 0.5
- frequency $1000 * 2\pi[rad/s]$
- Sample time 1e-5

De forma que para un período del seno existan 100 muestras en la CPU y todas mayores a 0.

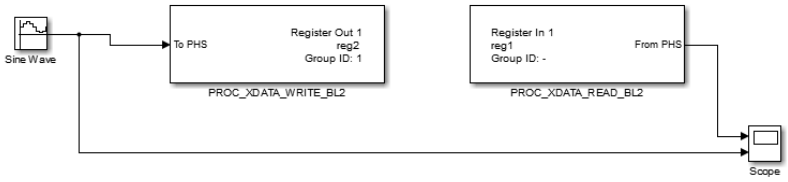


Figura 4: Raíz del modelo en Simulink

4.3. Síntesis del modelo FPGA

Ya que se tiene la CPU y FPGA configurados correctamente en Simulink. En la pestaña *parameters* de*FPGA_SETUP_BL1*, seleccionar las opciones del reloj y, finalmente, *Execute* (*Parallel Build*) para iniciar la síntesis.

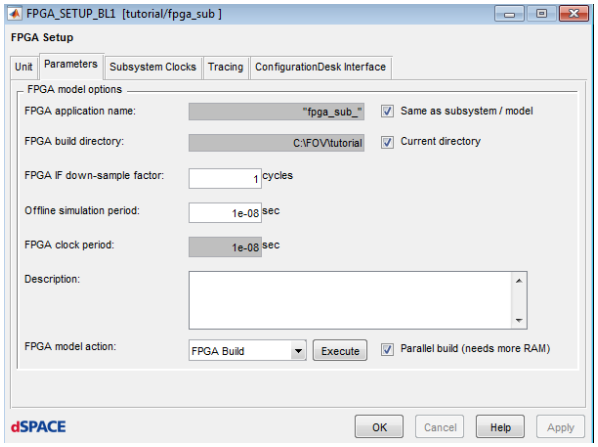


Figura 5: Build FPGA model

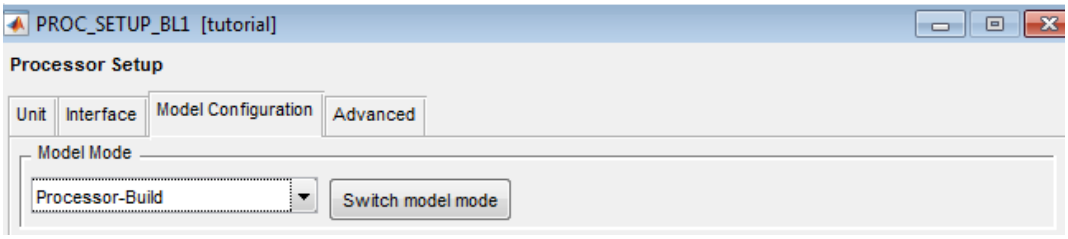
Se tarda entre 30 a 50 minutos en ejecutar está acción, en que se crea la carpeta *tutorial_rtiFPGA* que contiene 2 carpetas importantes:

- fpga_sub__4D4B1474562A4A : Lleva el nombre de la máscara en Simulink seguido por una ID única. En su contenido están los archivos de Vivado.
- ini : Contiene un archivo con la configuración inicial de la FPGA.
  Cada vez que se haga una nueva síntesis, se generará un nuevo archivo .ini y una nueva carpeta *nombre_de_máscara_NuevaID*.

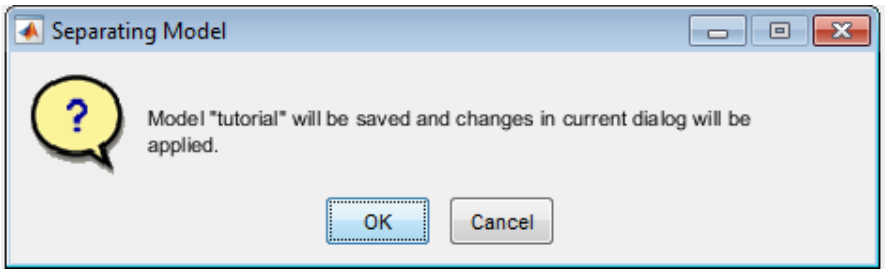
```
Starting model preparation for code generation...

The path selected for the build process is currently 64 characters long (max. recommended:
110 characters). Long build paths might lead to problems with filesystem operations in the
build process. In order to avoid these problems, please ensure, that the sum of the length
of Working Directory, model name and the name of the FPGA subsystem to be built is minimized.
Updating Model...
Starting System Generator code generation...

Starting synthesis of System Generator output files...
Packing netlists into one netlist file (EDF)...

Starting synthesis...
Starting implementation...
Starting generation of bitstream file...

WORK   DIRECTORY: C:\FOV\tutorial
BUILD  DIRECTORY: C:\FOV\tutorial\tutorial_rtiFPGA\fpga_sub__4D4B1474562A4A
RESULT FILE:      C:\FOV\tutorial\tutorial_rtiFPGA\ini\fpga_sub__4D4B1474562A4A.ini

                                                Type     Used   Available   Utilization [%]
   Configurable Logic Block Slices (LUTs, Flip-Flops)    44422       50950             87.19
                  Configurable Logic Block Slice LUTs   135928      203800             66.70
            Configurable Logic Block Slice Flip-Flops   109146      407600             26.78
                                Block RAM Blocks 36 Kb        7         445              1.57
                                Block RAM Blocks 18 Kb       36         890              4.04
                                            DSP Slices      150         840             17.86

FPGA Build Done
Elapsed time is 01:00:18.
```

Figura 6: Mensaje en Matlab de una síntesis exitosa

Terminada la compilación del subsistema de la FPGA, se debe cambiar la opción *FPGA-Build/Offline Simulation a Processor-Build*, en la pestaña de *Model Configuration* en el bloque *PROC_SETUP* y hacer clik en la opción *Switch model mode*.



Confirmar el mensaje que solicita guardar el modelo en Simulink de FPGA. Lo que genera el archivo *tutorial_rtiFPGASeparationFile.slx* en el directorio. Simultáneamente el subsistema de la FPGA deja de ser visible en Simulink.



4.4. Solver Simulink

En la barra de herramientas (figura 7) de Simulink, seleccionar el ícono de engranaje y configurar el Solver.

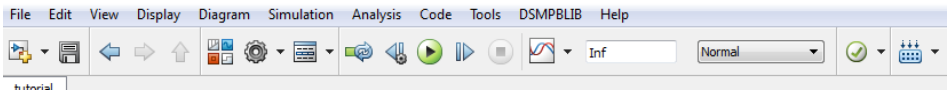


Figura 7: Barra de herramientas de Simulink

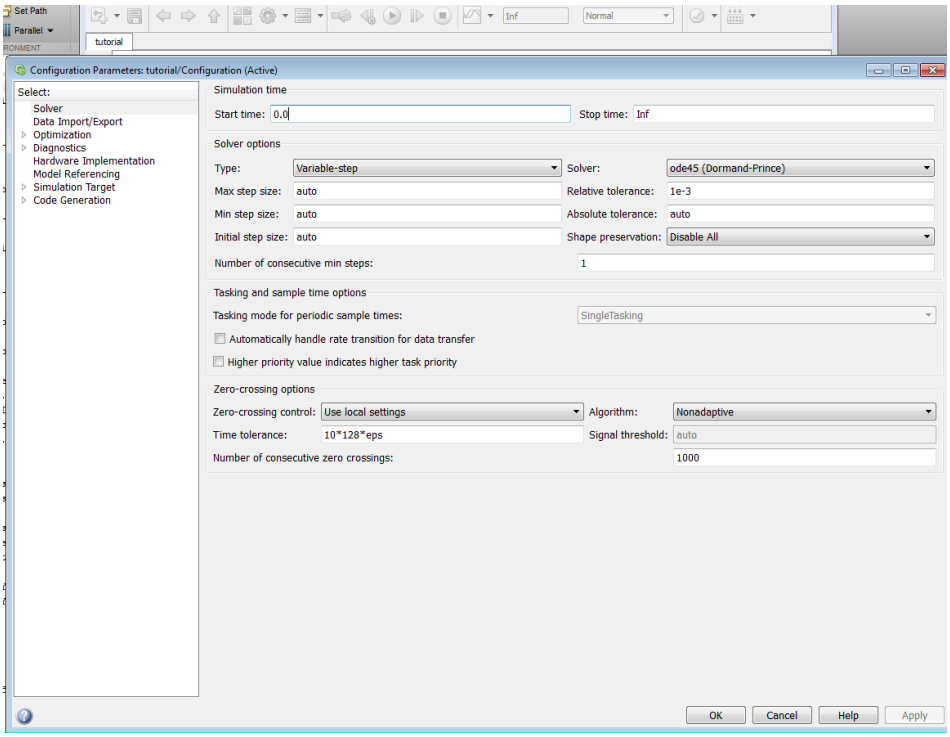


Figura 8: Configuración del Solver de Simulink

4.5. Generar Modelo CPU

Iniciar la compilación del modelo de la CPU en Simulink al hacer clic en *Build Model* en la barra de herramientas de Simulink o apretando las teclas Ctrl+b.

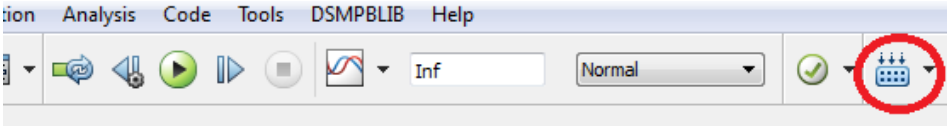


Figura 9: Build model CPU en modo processor build

Los archivos referentes al modelo se guardan automáticamente en una carpeta en el directorio del proyecto de nombre *tutorial_rti1202* (1202 es el nombre del módulo DSP y 1302 corresponde a la FPGA), dentro de la carpeta se destacan los archivos:

- tutorial.c Corresponde al main del modelo en c que genera simulink al ejecutar *Build Model*.
- fpga_sub__4D4B1474562A4A.c ,donde *fpga_sub* es el nombre de la máscara en simulink y 4D4B1474562A4A es la ID única de cada modelo FPGA.

En el directorio se generan los archivos:

- tutorial.rta
- tutorial.sdf
- tutorial.trc

Los cuales son necesarios para hacer una conexión con ControlDesk si es que se quiere usar ese programa para ver las señales del proyecto.

4.6. Detalle del directorio

4.7. Scope

En el modelo en Simulink se observan las señales con un Scope (figura 4). Basta con darle a *run* en la barra de herramientas (ver figura 10).

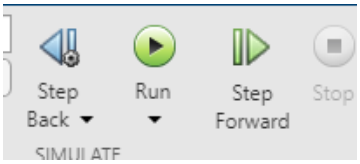


Figura 10: Ícono para iniciar la simulación

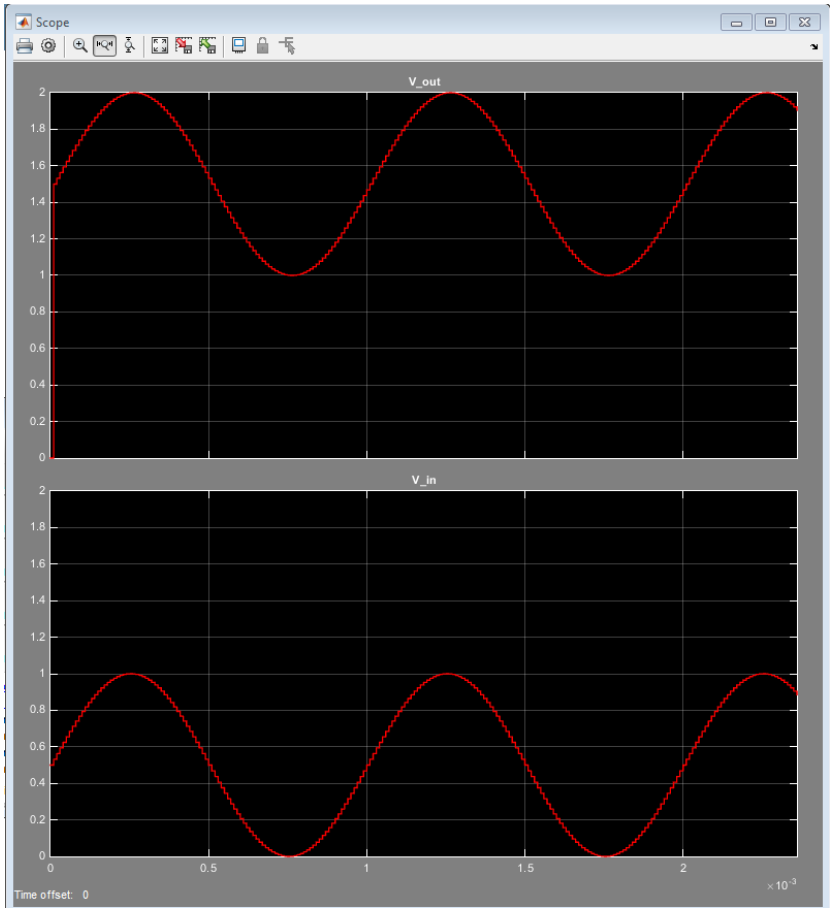


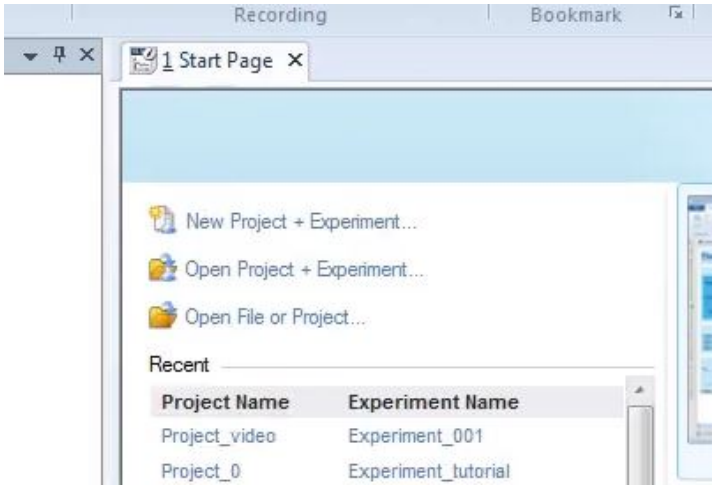
Figura 11: Scope de Simulink.De arriba a abajo (1) V_out= 0.5Sin(wt)+2 (2)V_in=0.5Sin(wt)+1

5. ControlDesk Next generation

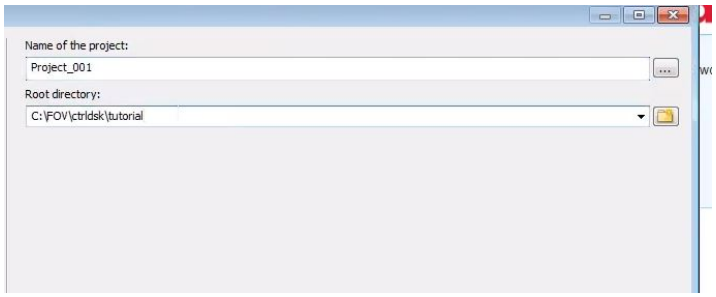
CONTROLDESK Next Generation es una HMI que permite desplegar distintas herramientas gráficas en pantalla para monitorizar datos experimentales en tiempo real en formato *CSV* o *MAT*.
En esta sección se explica como utilizarla para mostrar los resultados del modelo sumador creado en Simulink.

5.1. Preparar el proyecto

Al abrir ControlDesk, se despliega una página de bienvenida, desde la cual se puede crear un nuevo proyecto y experimento *(New Project + Experiment...)*, o abrir uno existente *(Open Proyect+ Experiment...)*.

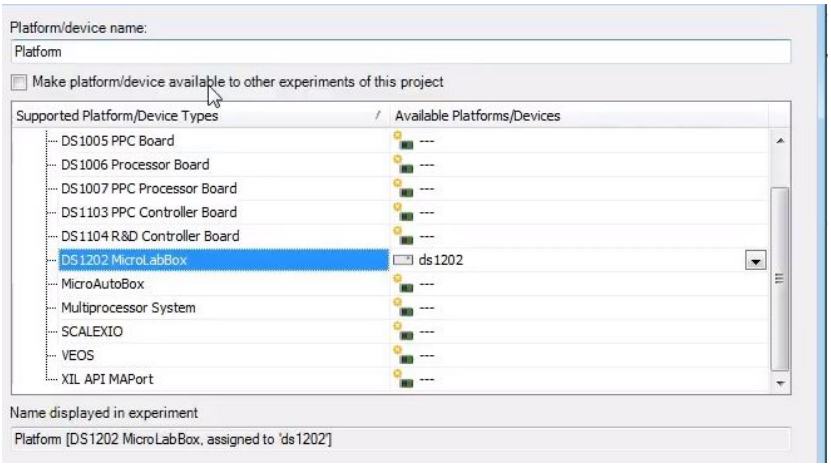


Al crear un nuevo proyecto, se abre una nueva ventana desde la cual se configuran las propiedades de este. Inicialmente se debe indicar el nombre y la ruta del proyecto:

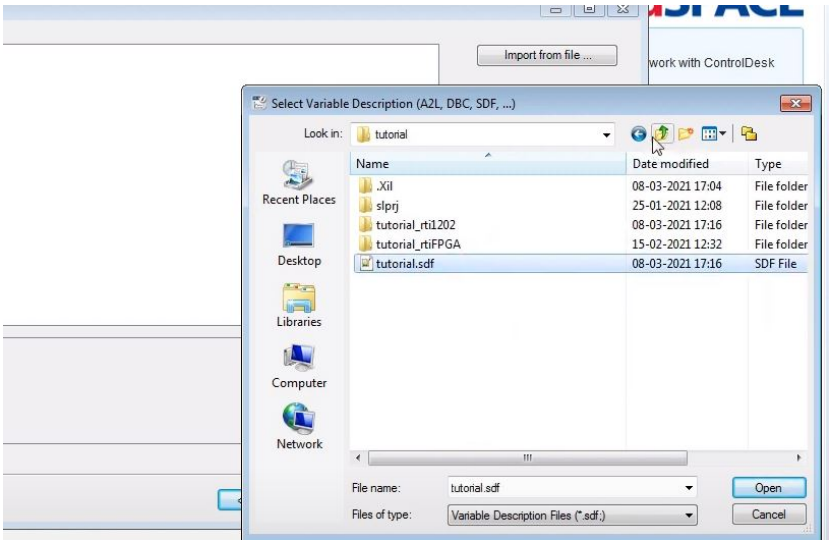


Al seleccionar *Next*, se debe ingresar el nombre del nuevo experimento, seleccionando nuevamente *Next*:

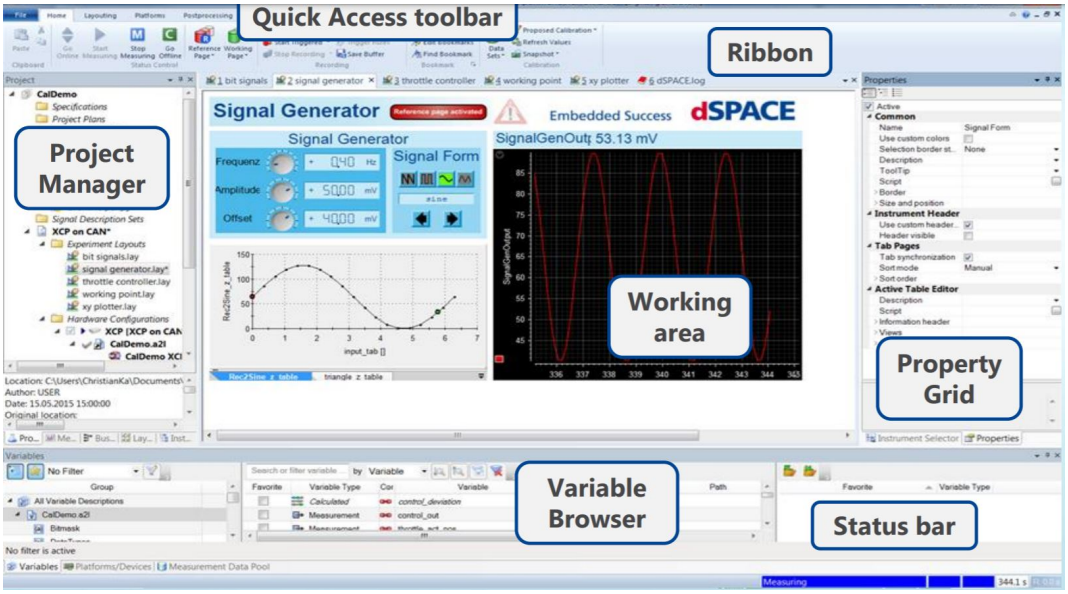
Luego, se debe seleccionar las plataformas dSPACE a usar en el experimento. si se encuentra correctamente registrado el equipo MLBX en el PC-Host, esta aparece dentro de las plataformas soportadas:



En la siguiente ventana se debe importar un archivo *.sdf* mediante *Import from file...*. Al realizar la programación mediante Simulink, este archivo es creado de forma automática al presionar *Build model* en la CPU (figura 9). En el caso de utilizar *C*, este debe ser creado de forma manual.



Una vez importado, se selecciona *Finish*, desplegando así un entorno de desarrollo, desde el cual se administran las plataformas del experimento, mediciones e implementaciones de HMI's (layout).



5.2. Crear experimento en el proyecto

Un proyecto en ControlDesk puede contener múltiples Experimentos, donde cada uno se encuentra asociado a la carga de un código o programa en particular, teniendo además sus propios HMI y plataformas. Ingresando a *File, New, New Experiment*, se puede agregar un nuevo experimento al proyecto activo.

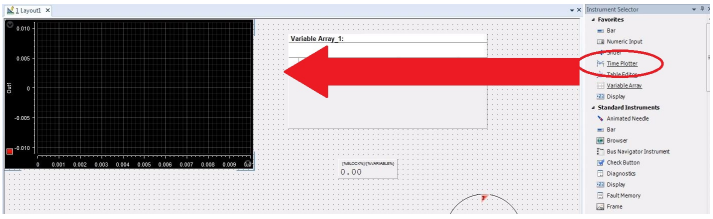
Al agregar un nuevo experimento, este se agrupa en la ventana de navegación o *Project Manager* (figura 5.1).

Para activar un experimento, se debe realizar clic derecho sobre él y seleccionar *Activate*. Sólo se puede tener un experimento activo a la vez. Cada vez que se cambie de experimento, tanto *Go Online* como *Start Measuring* volverán a realizar la carga de la aplicación en la MLBX, indicada por el archivo *.sdf*.

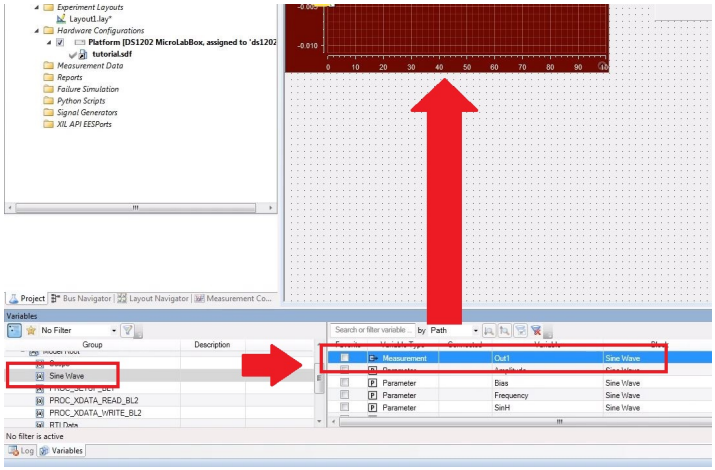
5.3. Control y Monitoreo de un modelo Simulink

Con el ejemplo del sumador en Simulink de la sección anterior, el objetivo ahora es ver las señales antes desplegadas en el scope de Simulink (V_{in}yV_{out}).

Arrastrar dos *Time Plotter* desde el *Instrument Selector* (fig 5.1) hacia el área de trabajo.

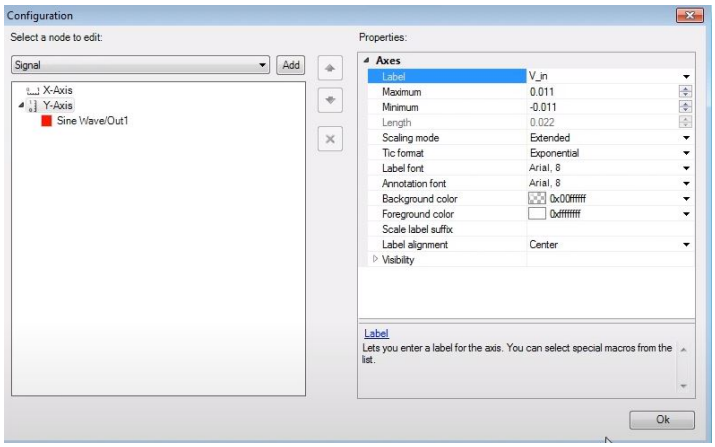


En el *Variable Browser* (fig5.1) seleccionar la variable deseada a graficar. Al hacerlo, se despliega una lista a la derecha, seleccionar *Measurementes* y arrastrarla hasta la ventana de *Time Plotter*

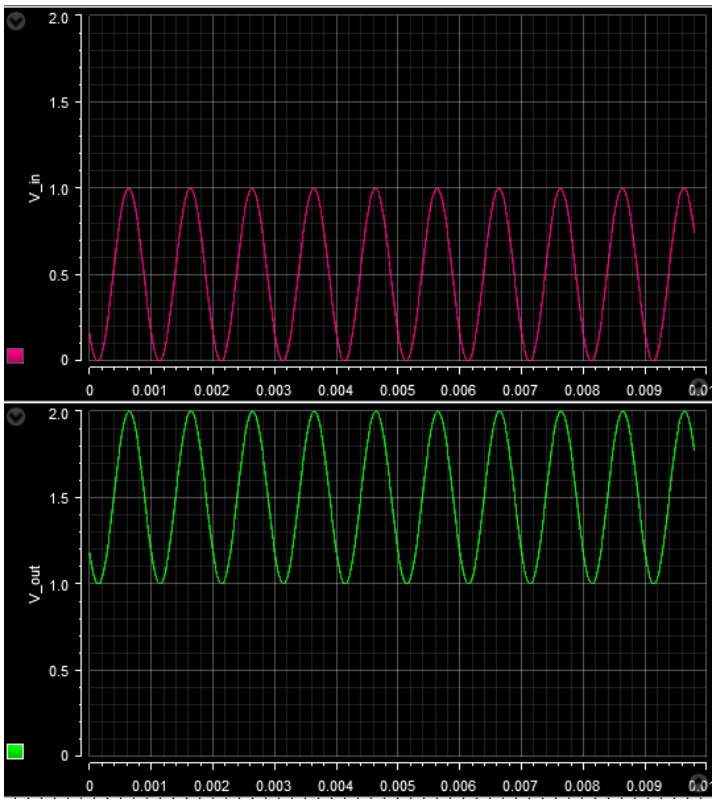


5.4. Drawing mode y ajuste del rango vertical de un Time Plotter

Por defecto estos instrumentos están configurados para refrescar cada gráfico cuando se reciben nuevos valores; la razón por la cual algunas capturas desplegadas no cubren la ventana temporal del instrumento y son mostrados de forma parcial.
Para configurar la actualización de la información desplegada por el instrumento cada vez que se complete su ventana de tiempo y no ante un nuevo valor, se debe hacer clic derecho sobre él y seleccionar *Instrument Properties*. En *Properties, Plotter, DAQ*, se debe modificar el parámetro *Drawing mode* de *On new value* a *On new capture*.
Por defecto el rango vertical de estos instrumentos es dinámico respecto a los valores de las variables. Para especificar un rango vertical estático, se debe hacer clic derecho sobre el instrumento y seleccionar *Axes/Signal Properties*, con lo cual se despliega la siguiente ventana.



Cada variable posee su propia escala vertical *Y-Axis*. Seleccionando cada una de ellas, se debe modificar el parámetro *Scaling mode* de *Extended* a *Fixed*, habilitando la edición del rango mediante los parámetros *Maximum* y *Minimum*.
Una vez listas las configuraciones, basta con hacer clic en *Start Measurment* para observar las señales.



Referencias

[1] Programación en Verilog

[2] Programación en Simulink

[3] Programación en C

[4] Programación en Control Desk

[5] dSPACE MicroLabBox Product Brochure

Apéndice B

MÓDULO DS1302: BLOQUES EN SIMULINK

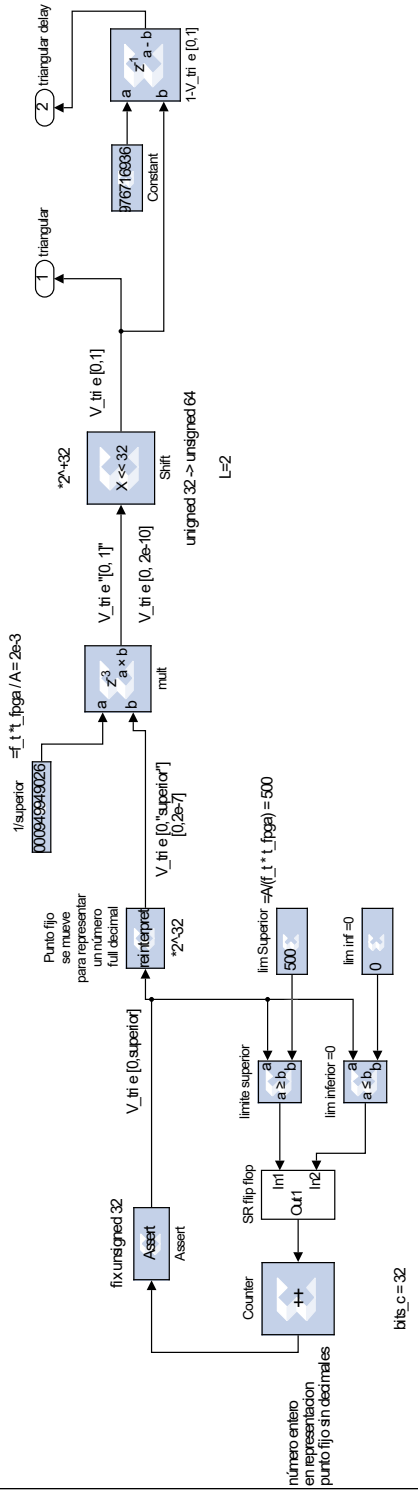


Figura B.1: Bloques de FPGA que generan la señal de carrier

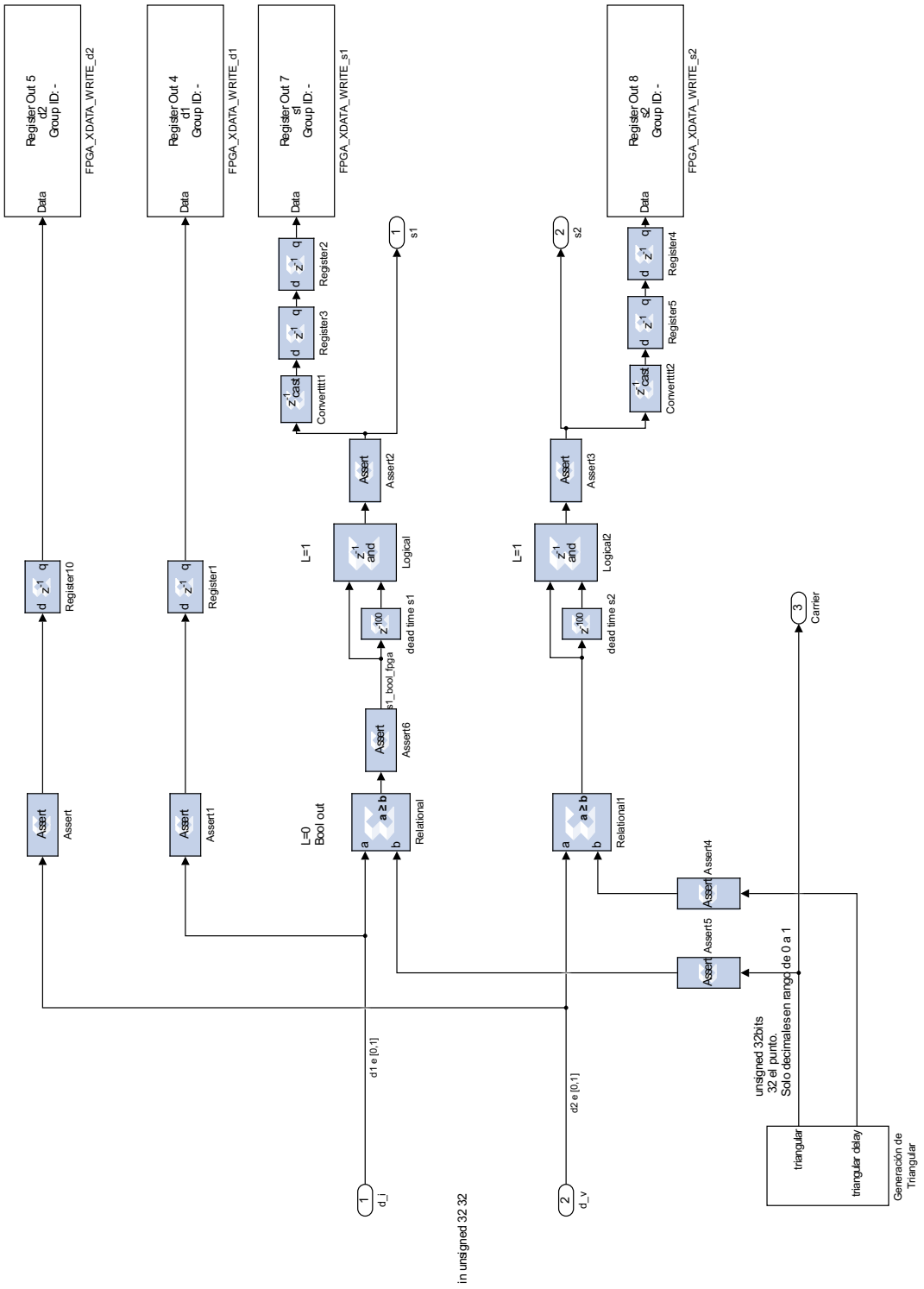


Figura B.2: Modelo del control PWM

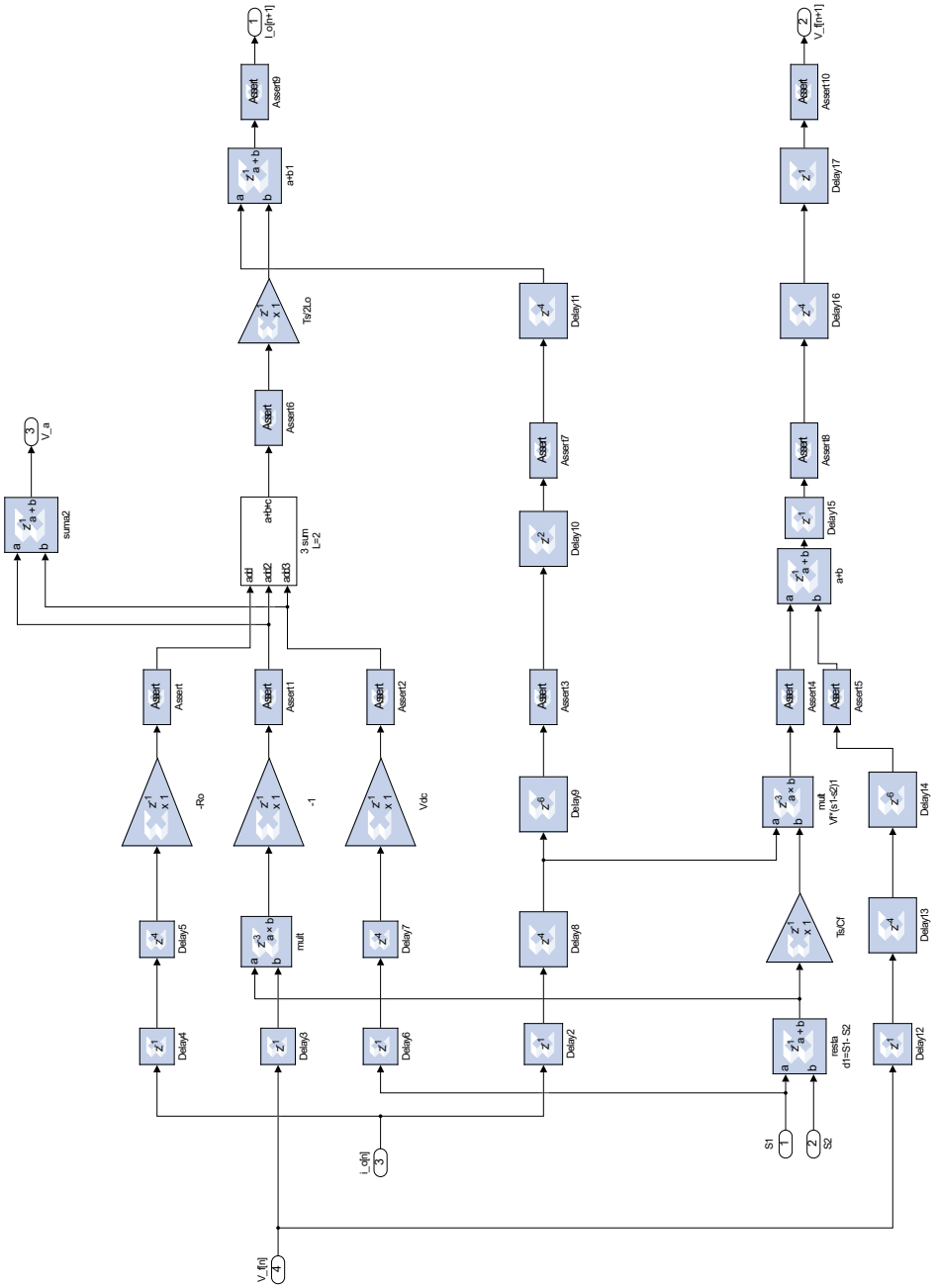


Figura B.3: Modelo en Simulink del FCBC

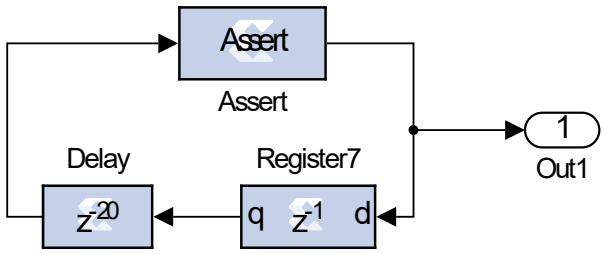


Figura B.4: Subsistema de Simulink del reloj utilizado para los registros con enable

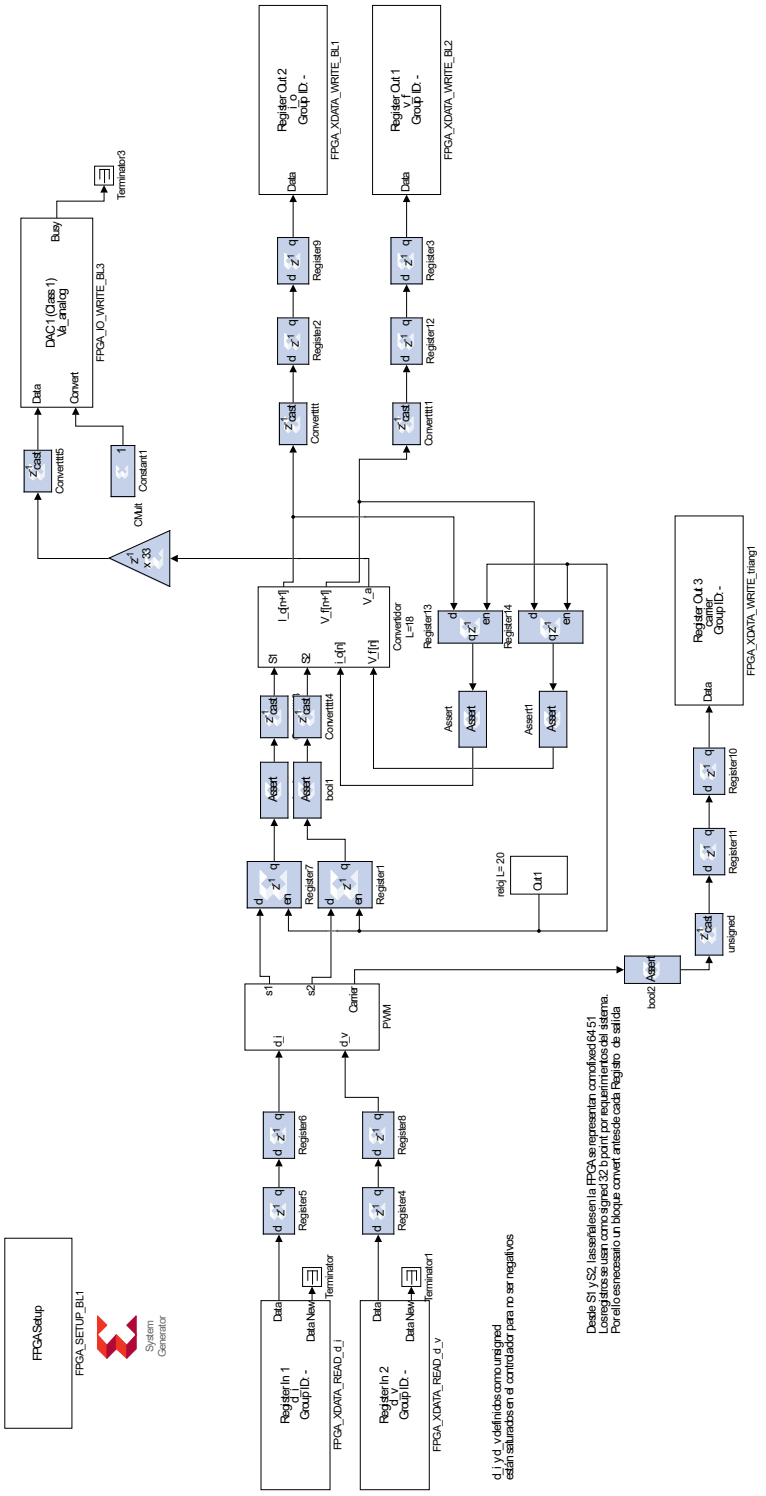


Figura B.5: Diagrama de bloques de control PWM y Modelo FCBC

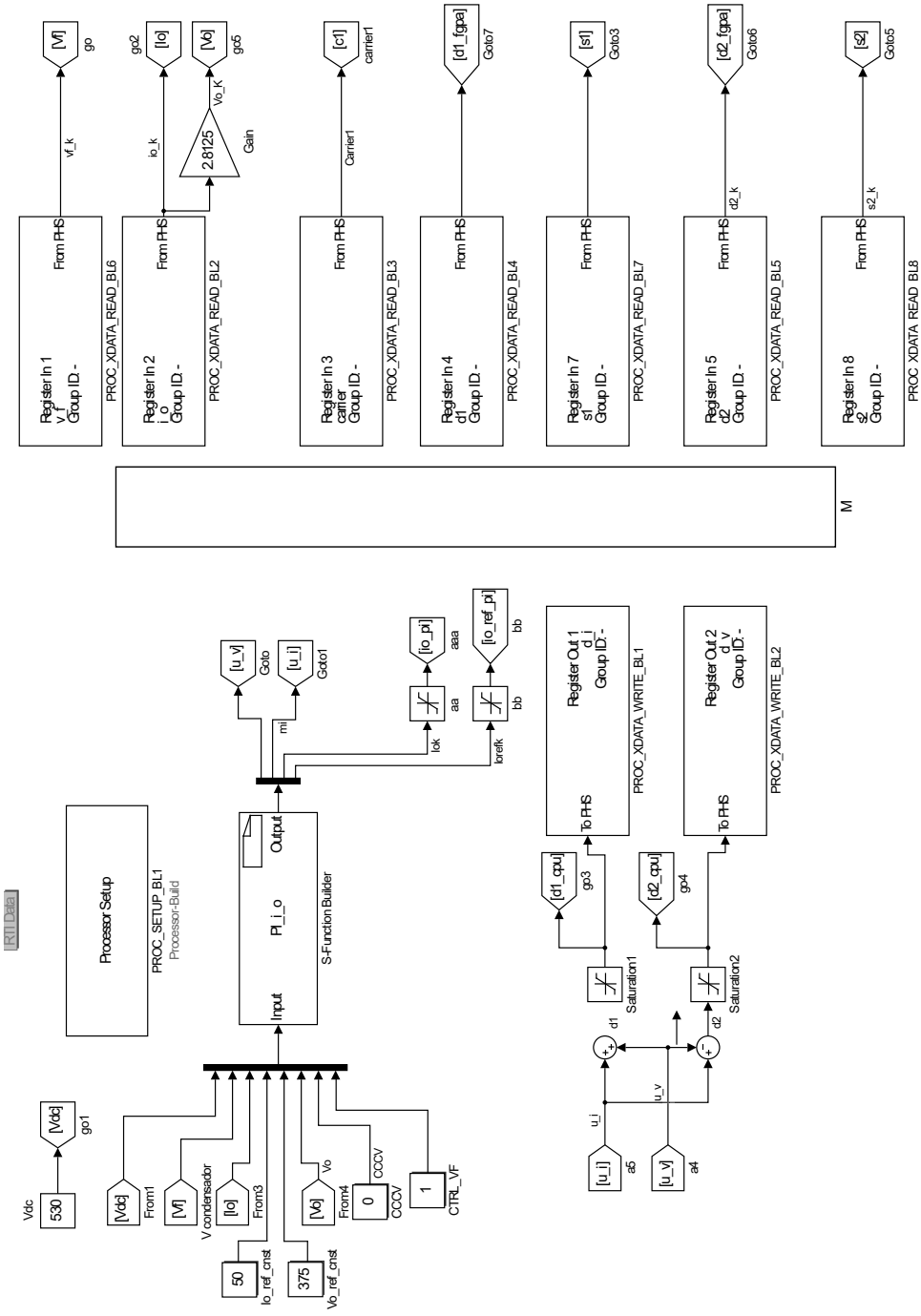


Figura B.6: Modelo en Simulink del Controlador

Apéndice C

MÓDULO DS1202: CÓDIGO C

```
1
2
3       */
4       /* %%%FUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
5
6       /* %%%FUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */
7
8       /*
9        * Output functions
10       *
11       */
12      void PI_i_o_Outputs_wrapper(const real_T *Input,
13            real_T *Output)
14      {
15      /* %%%FUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
16      //------------------------------from mail text
        ------------------------------------------------
17
18      Vin = Input[0];
19      Vfc = Input[1];
20      ILk  = Input[2];
21      Voutref = Input[3];
22      ILref = Input[4];
23      Vout = Input[5];
24      CCCV = Input[6];
25      Vf_ctrl = Input[7];
26
27      if(CCCV == 0) {
28        m = PI1(ILref,ILk,-1.0,1.0,Kpi,Kii);
29      } else {
30        m = PI2(Voutref,Vout,-1.0,1.0,Kpv,Kiv);
31      }
32      if(Vf_ctrl==1){
33          dV = PI3(Vin,2.0*Vfc,-0.1,0.1,Kpc,Kic);
34      }
35      else{
36          dV= 0.0;
```

```
37      }
38      Output[0] = dV;
39      Output[1] = m;
40      Output[2] = ILk;
41      Output[3] = ILref;
42      /* %%%FUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
```

Listing C.1: Algoritmo de control implementado en S-function Builder

```
1  // System Parameters
2  #define Lo    2.0e-1      // Filter Inductance, [H]
3  #define Ro    2.8125    // Load Resistance, [Ohm]
4  #define Cf    0.47e-2   // Capacitance. [F]
5
6  // Controller parameters
7  double Ts = 100.0e-6;
8
9  // Function to structure field
10 typedef struct {
11   double fc1;
12   double fc2;
13   double fc3;
14   double fc4;
15 } fld;
16
17 // Measurements
18 double Vin = 0.0, Vfc = 0.0, Vout = 0.0, Voutref = 0.0;           // dc-link
       Capacitor Voltages
19 double ILk = 0.0, ILref = 0.0;         // Inductance Current and its reference
20 int CCCV = 0.0;
21 int  Vf_ctrl = 0.0;
22
23 // Actuations
24 double dV = 0.0;
25 double m = 0.0;
26
27 // PI Voltage Controller vector
28 //#define sizepi 3       // Size of PVCTRL array
29 //#include <myPVCTRL.h>
30
31 #include "PIcontrol1.h"
32 #include "PIcontrol2.h"
33 #include <PIcontrol3.h>
34
35 // Controller Parameters
36 double Kpi =  0.015801;//0.016562995406237564221/100;
37 double Kii = -0.011485;//-0.013132581139437310501/100;
38 double Kpv =  0.005889045853391427053 5;
39 double Kiv = -0.004669345043202409115 6;
40 double Kpc =  0.000840608395879474540 69;
```

Listing C.2: data.h valores de las constantes utilizadas

```
//*****************************************************************
//   PIDcontroller: USM2020 - ruben.gonzalez.13@sansano.usm.cl
//       Original Code: UNSW2017 - c.a.rojas@ieee.org
//*****************************************************************

// Y(z)     Kpd*z - (Kpd - Kid*Ts)
// ---  = ------------------------     by using ZOH
// X(z)            z - 1

extern double PI1(double Rk, double Mk, double lowerLimit, double upperLimit,
     double Kpd, double Kid)
{
    // PI parameters
    static double Kp;
    static double Ki;
    static double KU;
    static double KX;

    // PI variables
    static double xRk;
    static double xMk;
    static double xlowerLimit;
    static double xupperLimit;
    static double xXk[2];
    static double xUk[2];

    // PI variables initialization
  xRk = Rk;
  xMk = Mk;
  xlowerLimit = lowerLimit;
  xupperLimit = upperLimit;
  Kp = Kpd;
  Ki = Kid;
  KU = -(Kp + Ki)/(Kp*Kp);
  KX = -Ki/Kp;

  // PI implementation
  xXk[1] = xXk[0];
  xUk[1] = xUk[0];
  xXk[0] = KU*xUk[1] + KX*xXk[1];
        xUk[0] = Kp*(xRk - xMk - xXk[0]);

  // Saturation
        xUk[0] = ( xUk[0] > (xupperLimit) ) ? (xupperLimit) : ( xUk[0] < (
    xlowerLimit) ) ? (xlowerLimit) : xUk[0];

    return xUk[0];

}
```

Listing C.3: Funcion PI utilizada en los archivos P1.h PI2.h y PI3.h

BIBLIOGRAFÍA

[1] J. Aravena, M. D. Dante Carrasco, R. C. Matias Uriarte, Felix Rojas, and J. C. Travieso, "Design and implementation of a low-cost real-time control platform for power electronics applications," *energies*, vol. 55, no. 2, pp. 1527–1542, 2020.

[2] "Plexim website," https://www.plexim.com/products/rt_box/comparison.

[3] "dspace microlabbox product brochure," 2020, https://www.dspace.com/en/inc/home/products/hw/microlabbox.cfm.

[4] "Op4510 simulator: Rt-lab / rcp / hil system."

[5] *System Generator for DSP: Reference Guide*, 2009.

[6] *MicroLabBox: RTLib Reference*, 2016.

[7] J. Mina, Z. Flores, E. Lopez, A. Perez-Flores, and H. Calleja, "Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in fpga," 06 2016, pp. 172–177.

[8] "Mathworks:hardware-in-the-loop (hil) simulation," https://la.mathworks.com/discovery/hardware-in-the-loop-hil.html.

[9] W.-H. L. Hung-Chi Chen, Che-Yu Lu and T.-H. Chen, "Active capacitor voltage balancing control for three-level flying capacitor boost converter based on average-behavior circuit model," *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS*, vol. 55, no. 2, pp. 1628–1638, 2019.

[10] X. M. Chunjiang Zhang, Meina Xu, "A high-gain floating-interleaved three-level boost converter."

[11] *Módulo DS1302: Programación en Verilog*, 2017.

[12] *DS1302: Programación en Simulink*, 2017.

[13] *DS1202: Programación en C.*, 2017.

[14] *MicroLabBox: ControlDesk Next Generation*, 2017.

[15] M. L. C., "Diseño de tarjetas de comunicación para equipo dspace microlabbox 1202/1302," *IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS*, 2017.

[16] "Programming dspace microlabbox in simulink, using xilinx add-on: Your first model (an adder) -part1," https:www.youtube.comwatch?v=CodR_I8dovU&t=2s&ab_channel=FelipeOsorioVillalobos.

[17] "How to watch signals of a dspace microlabbox on controldesk next generation." https:www.youtube.comwatch?v=h0jui8LohA4&ab_channel=FelipeOsorioVillalobos.

[18] *Implementation Software: Real-Time Interface*, 2020.

[19] *Vivado Design Suite Tutorial Model-Based DSP Design Using System Generator*, May 23, 2016.

COPYRIGHT