HERRAMIENTAS EDA PARA EL DISEÑO DE CONTROLADORES DIGITALES EN VHDL

Adriano José Peña Mora

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA 2015

HERRAMIENTAS EDA PARA EL DISEÑO DE CONTROLADORES DIGITALES EN VHDL

Adriano José Peña Mora

Trabajo de grado presentado como requisito parcial para optar al título de ingeniero electrónico

Director
RICARDO ALZATE CASTAÑO, PhD

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA 2015

DEDICATORIAS

A Dios, a mis padres Manuel Peña y Olinda Mora, a mi amada esposa Johanna Páez y a mi hijo Camilo Andrés. Que son el motor de mi vida.

Adriano José Peña Mora.

AGRADECIMIENTOS

A la Universidad Industrial de Santander, por darme la oportunidad de crecer en mi vida profesional y de la cual me siento muy orgulloso.

Al profesor Ricardo Alzate, director del proyecto, por su enorme paciencia y colaboración, fue de gran ayuda para la elaboración y culminación de este trabajo.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

Título: HERRAMIENTAS EDA PARA EL DISEÑO DE CONTROLADORES DIGITALES EN VHDL[1]

Autor: ADRIANO JOSÉ PEÑA MORA[2]

Palabras Clave:

Control digital; Convertidor de potencia DC-DC; Metodología de diseño electrónico; Simulación mixta; Síntesis de hardware.

Descripción:

En el presente trabajo se presenta la integración de herramientas de simulación para circuitos de señal de naturaleza mixta (analógica + digital), como parte de una metodología para el diseño de controladores digitales en circuitos eléctricos. En particular, se analiza el diseño de un lazo de control para regular la tensión de salida de un convertidor DC-DC reductor (Buck). La herramienta PSpice A/D se combina con el software Synplify Pro y la herramienta de conversión VHDL2Pspice, para producir componentes digitales de alto nivel sintetizadas desde código VHDL. Lo anterior, permite aplicar la metodología sugerida por S. Rajagopalan para el diseño de sistemas de señal mixta. Inicialmente, se constituye un bloque de actuación PWM gobernando la trayectoria directa mediante asignación para el valor del ciclo útil. Posteriormente se ajustan los rangos de las variables del lazo de control realimentado y se estudia la atenuación de perturbaciones a partir de un control simple de tipo proporcional. Finalmente, se codifica una estrategia de control por modos deslizantes que permite validar la efectividad de la metodología propuesta en el diseño de controladores digitales. Trabajos futuros incluyen la verificación experimental de las predicciones numéricas y la extensión de la metodología propuesta hacia otro tipo de aplicaciones que involucren el procesamiento digital de señales.

[1] Trabajo de grado

[2] Facultad de Ingenierías Físico-mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Ricardo Alzate Castaño, PhD.

ABSTRACT

Title: EDA TOOLS FOR DIGITAL CONTROL DESIGN IN VHDL[3]

Authors: ADRIANO JOSÉ PEÑA MORA[4]

Key words:

Digital control; DC-DC power converter circuit; Mixed-signal design methodology; Mixed simulation; Hardware synthesis.

Description:

This report presents the application of a methodology for designing mixed signal (analogic + digital) electronic systems. In particular, the study case of an output voltage regulation in a DC-DC, Buck type, power converter circuit, is chosen to illustrate the steps involved in designing, implementation and analysis, of a digital controller governing the dynamical behavior of an analogic circuit. Integration of PSpice A/D, Synplify Pro and VHDL2Pspice software tools, is performed in order to apply the design methodology proposed by S. Rajagopalan for mixed signal circuits. As first, a PWM block is generated from VHDL to govern the switching pattern of the Buck circuit by a proper selection of the duty cycle value. Then, a closed-loop is configured by adjusting ranges of variables involved, verified by analysis of perturbation rejection under different loop gains. Finally a Sliding-mode-control algorithm is applied, showing a good performance that validates the controller design and also the correctness of the proposed methodology. Ongoing tasks include experimental verification for numerical predictions and application of the proposed methodology to solve general digital signal processing problems.

[3] Degree work.

4 Physico-mechanical Engineering Faculty. School of Electrical Engineering. Supervisor: Ricardo Alzate Castaño, PhD.

INTRODUCCIÓN

La tecnología actual nos permite encapsular en una pastilla de silicio miles de millones de transistores, constituyendo arquitecturas de configuración flexible que hacen posible la interacción de bloques funcionales específicamente diseñados, los cuales a su vez están conformados por arquitecturas que pueden ser configurables (*System on Chip* – SoC [1]). Lo anterior, demuestra la evolución del hardware dedicado, cuyas herramientas a disposición para la implementación de sistemas embebidos [1] (empotrados o de propósito particular), han madurado desde los primeros dispositivos ASIC programables, pasando por microcontroladores y DSPs, hasta llegar a CPLDs y FPGAs, e incluso dispositivos análogos como FPAAs; todos ellos integrados hoy en arquitecturas SoC, en las cuales se maximizan en conjunto las ventajas que cada una de estas opciones tecnológicas brindan de manera individual a partir de una aproximación modular en la estructura de realización del sistema implementado.

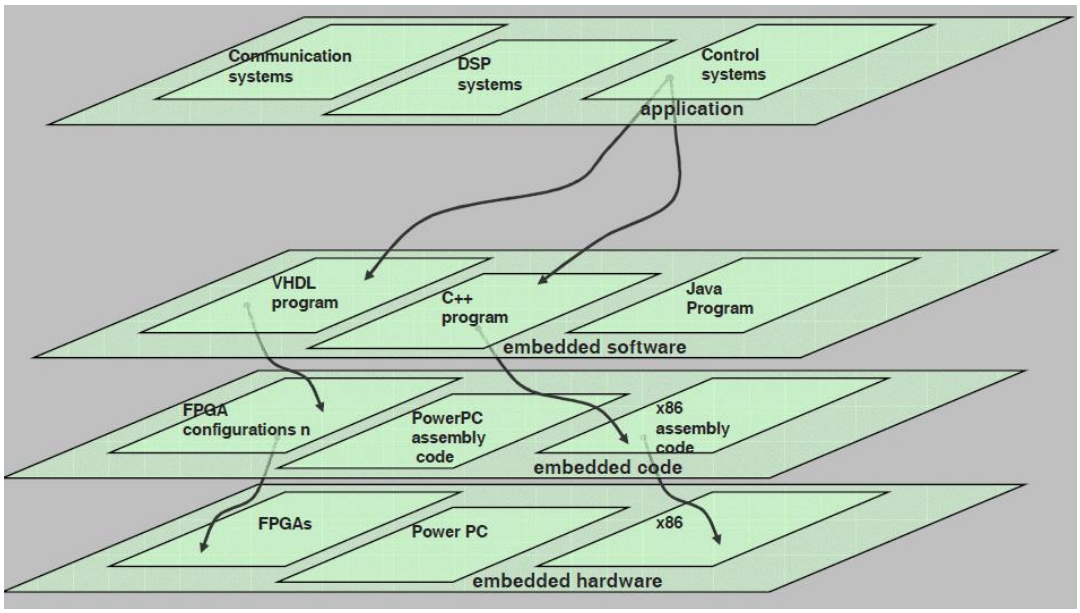
El reto para los diseñadores de sistemas electrónicos se encuentra por tanto en la capacidad que posean para producir soluciones eficientes en tiempos compatibles con las características y exigencias del contexto (muchas veces cambiante) en el cual se formulan los requerimientos. Por ejemplo, en el ámbito industrial, un retardo de diseño se manifiesta en retrasos para una línea de producción que posteriormente se evidencian en reducción de la capacidad productiva de la organización. En el caso de un dispositivo de consumo masivo, un retardo de diseño puede implicar la obsolescencia del producto, con una correspondiente desventaja en el mercado con respecto a sus competidores. El panorama podría empeorar, si adicionamos a lo anterior la posibilidad de generar un dispositivo defectuoso o carente de un diseño realizado con rigurosidad. De hecho, muy comúnmente los sistemas embebidos son empleados en situaciones de riesgo

para la integridad humana (e.g. sistemas de air-bag para automóviles), en los cuales no es tolerable la posibilidad de un fallo.

Metodologías tradicionales de tipo top-down [5] para el diseño de sistemas embebidos, implican las etapas fundamentales de descripción (que bien puede ser en términos de comportamiento y/o estructura), simulación, síntesis o implementación y verificación de prototipos [2]. La calidad de un diseño se valora entonces a partir de la efectividad que se obtenga en las rutinas de verificación. En caso de manifestarse un comportamiento inadecuado; es decir, que no satisfaga los requerimientos del problema específico, se debe proceder con un rediseño a nivel de las diferentes etapas del procedimiento. Para garantizar una eficiencia en los procesos de síntesis y verificación, metodologías modernas realizan co-diseños hardware/software adecuados a las plataformas de implementación seleccionadas para ejecutar la realización física definitiva del prototipo solución (ver Figura 1.1). Esta aproximación, ofrece la posibilidad de emplear rutinas automatizadas y optimizadas mediante aplicaciones EDA [6] como [1]: ISE para *Xilinx* (FPGAs y CPLDs), *Code Composer Studio* para *Texas Instruments* (DSPs) y *CodeWarrior* para *Freescale-Motorola* (µPs), entre otras. Evidentemente, esta no es una estrategia adecuada al momento de considerar diseños que no dependen de una plataforma de implementación específica o única, como puede ser el caso de sistemas embebidos distribuidos, ejemplificados principalmente a partir del concepto de red [1][3], en el cual se asume la interacción multidireccional de diferentes agentes (nodos) interconectados.

[5] Se refiere a la descomposición jerárquica de una descripción desde un nivel superior o abstracto (top), hasta uno inferior o específico (down), a nivel de componentes fundamentales.
[6] Electronic Design Automation

Figura 1.1. Secuencia típica para implementación de sistemas embebidos



Fuente: [3]

Como alternativa, se proponen metodologías de diseño basadas en la utilización de uno o más modelos formales (modelos de computación, o MoC de su sigla en inglés [4, 5, 6]) para describir el comportamiento del sistema embebido en un nivel de abstracción superior.

A nivel práctico sin embargo, se verifican dificultades al momento de implementar este tipo de metodologías de diseño debido principalmente al manejo individual que estas realizan con respecto a la naturaleza de las señales en el sistema. En particular, no es común encontrar simuladores de sistemas digitales que interactúen con etapas de circuito analógico, y aún más, en el caso que existan (como bien ocurre con la simulación mixta en *Orcad-Cadence*®) se requiere de configuraciones especiales en los componentes del sistema para asegurar la compatibilidad de los datos y viabilizar las tareas de análisis. Lo anterior se limita

todavía más considerando la dependencia de estas simulaciones híbridas en la disponibilidad de librerías digitales por parte de las correspondientes herramientas de simulación. Ahora bien, teniendo en cuenta que la simulación forma parte de las etapas de verificación preliminar del diseño (como paso previo a la materialización de un eventual prototipo), es todavía más relevante la necesidad de incorporar esta información de compatibilidad entre las señales de naturaleza digital y analógica, desde etapas iniciales en la formulación del diseño. Al respecto, algunos desarrollos se presentan para proponer herramientas que permitan verificar dispositivos digitales genéricos en diseños de señal mixta, a partir de simuladores tradicionales como *Orcad-Cadence*® [7, 8, 9, 10].

El presente proyecto de grado busca emplear este tipo de herramientas para validar el diseño de estrategias avanzadas de control en circuitos convertidores de potencia.

1. PLANTEAMIENTO Y DEFINICIÓN DEL PROBLEMA

En un procedimiento de diseño electrónico se busca constituir un sistema que ejecute una tarea requerida de manera eficiente a partir de consideraciones como costo, tamaño, consumo energético y calidad de señal, entre otras. Asimismo, actualmente se dispone de un número importante de opciones para realizar cálculos y simulaciones en tareas de diseño, muchas de ellas automatizadas a través de rutinas computacionales (herramientas EDA). Sin embargo, en la mayoría de casos los mundos analógico y digital se proponen, modelan y verifican de manera independiente, generando como consecuencia durante la materialización circuital en laboratorio de prototipos de señal mixta, comportamientos inadecuados al momento de ser acoplados como un conjunto (parte digital + partes analógicas), a pesar de operar de manera conforme por separado. En este contexto, surgen inquietudes de investigación como las siguientes: ¿Qué metodología de diseño electrónico es más convenientes cuando se trata de sistemas con señales de naturaleza mixta? ¿Qué herramientas EDA existen para estos casos particulares de diseño? ¿Cómo incorporar elementos de diseño digital en sistemas de naturaleza mixta como lo son los controles de circuitos convertidores de potencia?

El presente proyecto de grado busca realizar aportes y obtener resultados direccionados a la resolución de estas inquietudes, constituyendo una base para posteriores desarrollos afines que permitan abordar de manera profunda esta temática al interior del grupo de investigación CEMOS.

1.1. OBJETIVOS

1.1.1. Objetivo general

Emplear herramientas de diseño electrónico asistido por computador (EDA) para verificar algoritmos de control digital programados en VHDL aplicados a circuitos convertidores de potencia

1.1.2. Objetivos específicos

- Utilizar herramientas computacionales para el diseño de sistemas electrónicos de señal mixta

- Simular un algoritmo de control digital programado en VHDL dentro de un simulador comercial de circuitos

- Verificar en simulación un algoritmo de control digital sobre un convertidor de potencia

2. PRINCIPIOS DEL DISEÑO ELECTRÓNICO

El presente capítulo aborda el diseño electrónico, centrando su atención en las principales metodologías que permiten sintetizar circuitos, a partir de las herramientas tecnológicas disponibles en la actualidad.

2.1. INTRODUCCIÓN

Los desarrollos digitales y analógicos han tendido a evolucionar a ritmos diferentes. Sin embargo el 70% de los circuitos integrados (IC) diseñados hoy día son de señal mixta, y la cifra va en aumento. Debido a esta tendencia se busca reducir esta brecha a partir de nuevas metodologías de diseño y mejor colaboración tanto de fabricantes como desarrolladores de herramientas [24].

La mayoría de las aplicaciones actuales requieren la co-existencia de bloques analógicos y digitales, y los beneficios de la combinación de estos en un solo chip son significativos. No obstante, existen aún algunos inconvenientes que deben ser resueltos. Por ejemplo, el ruido de reloj en circuitos digitales de alta velocidad, a menudo interfiere con funciones analógicas sensibles al ruido. Además, las corrientes de conmutación de funciones analógicas de alta potencia pueden interferir con procesadores digitales de bajo voltaje.

Con el aumento de la capacidad y la potencia del procesamiento digital, muchas funciones analógicas se convierten en señales digitales. La ventaja de este enfoque es que los filtros digitales y elementos de control digitales, no son sensibles a inexactitudes por deriva (*drift*) causadas por envejecimiento o cambios de temperatura en los procesos. El resultado es un diseño mucho más robusto que un enfoque puramente analógico [25].

2.2. METODOLOGÍAS DE DISEÑO

La manera como se aborda el problema del diseño electrónico, desde la concepción inicial de la idea hasta la materialización final del prototipo circuital, constituye una aproximación metodología a través de la cual, se debe garantizar el éxito del procedimiento tras la concepción de un dispositivo funcional que satisfaga los requerimientos del diseñador. A continuación se describen algunas metodologías de diseño tradicionales.

2.2.1. Metodología Bottom-up

Es el enfoque tradicional de diseño. Realiza una descripción del circuito, empezando por describir los componentes más pequeños del sistema (resistencias, condensadores, transistores, etc), posteriormente agrupados en subconjuntos, hasta llegar a uno solo que represente el sistema deseado. Esta forma de diseñar es eficaz para pequeños diseños, pero para diseños más grandes y complejos presenta problemas para identificar anomalías de funcionamiento, por lo que acarrea pérdidas de tiempo y costos [14].

2.2.2. Metodología Top-down

Se parte de una idea abstracta, refinada progresivamente a medida que el proceso de diseño continúa. Puede comenzar con una definición de comportamiento del sistema en muy alto nivel y luego se puede acceder a detalles más precisos, como un RTL (*Register transfer level*) o descripciones a nivel de puertas. Esta metodología es popular entre diseñadores de circuitos digitales tras la llegada de lenguajes de descripción de hardware (HDL), dispositivos lógicos programables (PLD) y herramientas de síntesis lógica [13]. Como principales características se tiene [2]: el incremento en la productividad del diseño, pues se puede definir, por

ejemplo, un diseño en VHDL y generar el RTL directamente, sin necesidad de considerar su implementación a nivel de puertas; la reutilización del diseño hacia otras tecnologías de implementación, pues dicha selección se establece en etapas finales; además de una rápida detección de errores, ya que las herramientas de diseño facilitan localizar errores en el proceso de descripción y síntesis.

2.2.3. Codiseño

El diseño de los sistemas electrónicos actuales acude en gran medida al uso de arquitecturas programables y requiere en muchas ocasiones que el diseñador combine elementos de hardware y software para implementar el sistema de forma eficiente. Esta metodología de diseño híbrido se conoce con el nombre de codiseño y es una técnica que busca aprovechar los puntos más fuertes tanto de hardware como de software, en un mismo sistema [27, 15].

Diseñar un sistema electrónico utilizando únicamente software (microcontrolador o microprocesador), tiene ventajas como [15]: mayor flexibilidad, facilidad para ejecutar órdenes secuenciales y menor tiempo en el diseño. Por otro lado, implementar la misma solución en hardware (FPGA, CPLD), tiene las siguientes ventajas [15]: mayor velocidad (por lo general un procesador de propósito específico es más veloz que un procesador de propósito general), menor consumo de potencia y posibilidad de realizar tareas de manera concurrente (paralela).

2.3. HERRAMIENTAS EDA

EDA (*Electronic Design Automomation)* o automatización de diseño electrónico, es el nombre que reciben todas aquellas herramientas que ayudan en el diseño y optimización de circuitos electrónicos. Dentro de estas herramientas podemos encontrar un sub-conjunto denominadas CAD (diseño asistido por ordenador, del inglés *Computer Aided Design*), complemento esencial a la hora de diseñar [2]. En

la evolución de estas herramientas ha jugado un papel muy importante el desarrollo de equipos de cómputo con mayores prestaciones y capacidad de procesamiento concurrente o paralelo, siendo además equipos compactos gracias a la miniaturización [2].

2.3.1. Lenguajes de descripción de hardware

Como consecuencia de la creciente necesidad de integrar un mayor número de dispositivos en un solo circuito integrado, se desarrollaron herramientas EDA complejas, que integran en el mismo marco de trabajo herramientas de descripción, de síntesis e implementación, que auxilian a los diseñadores electrónicos en sus diseños más complejos. También surgió la necesidad de disponer de una descripción del circuito que permitiera el intercambio de información entre las diferentes herramientas [2]. Esto permitió que en la década de los cincuenta aparecieran los primeros lenguajes de descripción de hardware (HDL) como una opción esencial en el ciclo de diseño [14].

Los primeros lenguajes de descripción de hardware, permitían mediante un conjunto de instrucciones describir un circuito a partir de sus componentes y sus conexiones, razón por la cual son llamados lista de conexiones (*Netlist*), pues en esencia el diseñador describe los componentes del circuito y sus conexiones en un diseño altamente manual [15]. Estos lenguajes alcanzaron mayor desarrollo durante los años setenta, periodo en el que se desarrollaron varios de ellos como IDL de IBM, TI-HDL de *Texas Instruments*, ZEUS de *General Electric*, etc., todos orientados al área industrial, así como los lenguajes en el ámbito universitario (AHPL, DDL, CDL, ISPS, etc.). Los primeros no eran de dominio público, mientras que los segundos carecían de soporte técnico para su utilización en la industria [14].

El desarrollo continuó y en la década de los ochenta surgieron lenguajes como VHDL, Verilog, ABEL, etc., considerados lenguajes de descripción de hardware

porque permitieron describir los circuitos con un alto grado de abstracción, no desde el punto de vista estructural (componentes y conexiones), sino desde un punto de vista funcional. Con este enfoque de los HDLs, el diseñador puede prestar más atención al funcionamiento del circuito que a su estructura [15]. En la actualidad, el lenguaje de descripción de hardware más utilizado a nivel industrial es VHDL, seguido de Verilog [13].

- **Lenguaje VHDL.** VHDL es el acrónimo que representa la combinación de VHSIC y HDL, donde VHSIC (*Very High Speed Integrate Circuit*) y HDL (*Hardware Description Language*), es decir, lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad. Este lenguaje de descripción de hardware fue desarrollado por el Departamento de Defensa de los Estados Unidos a finales de los años setenta [14], con la finalidad de diseñar y simular circuitos complejos, de forma en que los humanos y las maquinas puedan leer y entender la funcionalidad y la organización de sistemas hardware [2]. Después de varias versiones revisadas por el gobierno de los Estados Unidos, industrias y universidades, el IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) publicó en diciembre de 1987 el estándar IEEE 1076-1987, en 1993 el estándar se actualizó como el estándar IEEE 1164-1993 y después en 1996 el estándar IEEE 1076.3-1996, siendo este último y el 1164 los estándares más utilizados en la síntesis de circuitos digitales [14].

  VHDL es un lenguaje con una sintaxis amplia y flexible que permite la descripción de un circuito utilizando tres estilos de descripción o programación [17]: descripción de comportamiento[7], descripción estructural[8] y descripción de flujo de datos[9].

[7] Describe el comportamiento del circuito. Este tipo de descripción es la que más se parece a los lenguajes convencionales ya que las sentencias son secuenciales, estas sentencias secuenciales

- **Lenguaje Verilog.** Fue diseñado por *Phil Moorby* en 1985 cuando trabajaba en la compañía *Automated Integrated Design Systems*, que más tarde fue renombrada *Gateway Design Automation*. El objetivo de *Moorby* fue diseñar un lenguaje de descripción de Hardware con una sintaxis similar a la del lenguaje de programación C, de tal manera que le resultara familiar a los ingenieros y así fuese rápidamente aceptado. Con el incremento en el éxito del lenguaje VHDL, en 1990 se decidió abrir el lenguaje Verilog al domino público y dejar disponible su estandarización a través de Open Verilog International, actualmente conocida como *Acellera*. Verilog fue enviado a un grupo de trabajo de la IEEE que lo reviso y estableció el estándar IEEE 1364-1995, frecuentemente referido como Verilog 95. Ante algunas deficiencias en el lenguaje las cuales fueron revisadas y mejoradas, el grupo de trabajo IEEE público el nuevo estándar IEEE 1364-2001 conocido como Verilog 2001 [15, 16]. Un diseño en Verilog consiste de una jerarquía de módulos, los cuales son definidos con conjuntos de puertos de entrada, salida y bidireccionales. Internamente un módulo contiene una lista de cables y registros. Las sentencias concurrentes y secuenciales definen el comportamiento del módulo, describiendo las relaciones entre los puertos, cables y registros. Las sentencias secuenciales son colocadas dentro de un bloque begin/end y ejecutadas en orden secuencial, pero todas las sentencias concurrentes y todos los bloques begin/end son ejecutadas en paralelo en el diseño. Un módulo puede contener una o más instancias de otro módulo para definir un sub-comportamiento [15].

---

se encuentran dentro de bloques de proceso, estos procesos son ejecutados en paralelo con asignaciones concurrentes de señales y con las instancias a otros componentes.

[8] Instancia diferentes componentes jerarquizados y los interconecta entre sí, con el propósito de construir un diseño de jerarquía superior. Este estilo de programación puede asemejarse mucho a un Netlist, y generalmente se usa para interconectar bloques.

[9] Este tipo de descripción se encuentra a mitad de camino entre una de comportamiento y una estructural, pues aunque se trata de una especie de Netlist, las interconexiones no se hacen entre componentes sino sobre elementos abstractos del lenguaje.

2.3.2. Simulación numérica

El conocimiento del comportamiento de los circuitos eléctricos requiere la solución simultánea de un número de ecuaciones. El problema más fácil es el de encontrar el punto de operación en CC de un circuito lineal, que requiere resolver un conjunto de ecuaciones derivadas de la ley de voltajes y corrientes de Kirchhoff (KVL y KCL). El análisis de los circuitos que contienen elementos descritos por una relación no lineal entre la corriente y el voltaje añade otro nivel de complejidad, que requiere la solución de ecuaciones no lineales simultáneas basadas en las leyes de *Kirchhoff*. La complejidad sigue aumentando cuando se tiene que predecir el comportamiento de un circuito eléctrico en el tiempo o en la frecuencia. Las ecuaciones no lineales se convierten en ecuaciones integro-diferenciales, que pueden ser resueltas sólo con aproximaciones como la de pequeña señal u otras restricciones limitantes [19].

Los primeros simuladores de circuitos electrónicos, no eran tal como los conocemos hoy día. Dos grupos han contribuido significativamente al desarrollo de simuladores de circuitos electrónicos: el grupo ASTAP de IBM que desarrolló muchos de los métodos numéricos utilizados y el grupo de SPICE en la Universidad de California – *Berkeley*, que desarrolló y propagó el simulador estándar por defecto [18].

2.3.2.1. Simulación analógica. Este tipo de herramientas llevan varios años desarrollándose y mejorando sus características, uno de los primeros simuladores analógicos creados fue SPICE y a partir de éste se han desarrollado otros descendientes.

- **SPICE**: Acrónimo inglés de (*simulation program with integrated circuit emphasis*) o programa de simulación con énfasis en circuito integrado. El funcionamiento de este programa se basa en línea de órdenes, las cuales no son tan fáciles de utilizar al momento de diseñar todo un circuito complejo. Sin embargo,

SPICE es la base fundamental para otros programas con mayor interactividad con el usuario. (e.g. OrCAD). SPICE es un simulador de circuitos electrónicos analógicos de código abierto, utilizado para verificar diseños de circuitos integrados analógicos a nivel de tarjeta y para predecir el comportamiento del circuito antes de su fabricación [17, 26].

2.3.2.2. Simulación digital. Los simuladores digitales son paquetes de software (o Suites) que emulan cualquier lenguaje de descripción de hardware (HDL). Estas herramientas software de simulación HDL, han recorrido un largo camino desde su origen como producto patentado ofrecido por empresas. Hoy día, los simuladores están disponibles desde muchos proveedores y a varios precios, siendo representativos los productos de *Xilinx* y *Altera*. Las Suites empaquetan el motor del simulador como un entorno de desarrollo completo, es decir: un editor de texto, un visor de forma de onda, y el explorador de nivel RTL [14].

- **ISE**: Esta herramienta desarrollada por la compañía *Xilinx*, permite hacer descripciones en hardware utilizando diferentes HDLs (VHDL, Verilog y Abel). Adicionalmente realiza desde la síntesis hasta la implementación del diseño en el FPGA, empleando componentes como: *ISE Simulator* [10] , *ChipScope ILA*[11], *PlanAhead*[12], *System Generator for DSP*[13] y el *Plataform Studio and EDK*[14] [12]

[10] Es un simulador integrado de diseños en HDL, que permite hacer simulaciones funcionales y en tiempo para diseños Xilinx FPGA y CPLD.
[11] Herramienta para capturar y analizar las señales internas del FPGA en tiempo real para verificación.
[12] Permite optimizar los diseños de hardware a partir de la mejor ubicación de cada uno de los módulos dentro del FPGA.

- **Quartus II**: Es la herramienta de descripción de hardware para productos *Altera*. Posee como componentes básicos a paquetes como el *Modelsim-Altera* y el *DSP Builder*. Soporta los lenguajes Verilog, VHDL y AHDL.

- **Synplify Pro**: Es un software de síntesis lógica de FPGAs y CPLDs, desarrollado por *Synopsys, INC*. Por entrada el software recibe diseños de alto nivel (*Top-Down*) escritos en Verilog o VHDL. Usando su única y patentada herramienta BEST™ (*Behavior Extracting Synthesis Technology*®) realiza la optimización en un alto nivel, convirtiendo primero el HDL en un netlist antes de sintetizar el código RTL en la lógica de la FPGA, según el proveedor especificado por el diseñador. Esto elimina la lógica innecesaria para mejorar y optimizar el rendimiento del diseño, que a menudo hace que sea posible para los diseñadores seleccionar un dispositivo FPGA más pequeño y menos costoso de acuerdo a los requerimientos mínimos del diseño.

2.3.2.3. Simulación mixta o híbrida. Este caso particular combina los anteriores; es decir, relaciona la simulación de circuitos que poseen señales de naturaleza analógica y digital, y por tanto propende por una integración de las metodologías de diseño para cada caso. Son pocas sin embargo las herramientas tecnológicas que permiten abordar este tipo de situaciones, entre ellas se tienen:

[13] Con esta herramienta es posible diseñar sistemas DSP a partir del entorno Simulink de Matlab con el fin de implementarlos en un FPGA.
[14] Ambiente de trabajo que permite realizar diseños integrados de hardware y software. El hardware se implementa en los recursos lógicos del FPGA y para la parte de software se pueden utilizar los procesadores MicroBlaze y PowerPC.

- **PSpice A/D**: Es una herramienta de simulación que hace parte de la suite *OrCAD-CADENCE*, que se utiliza para analizar y predecir el rendimiento de los circuitos de señal mixta (analógicas y digitales). Es muy popular por los ingenieros que comprueban diseños a nivel de placa o de tarjetas de circuitos impresos (PCB). Sin embargo, *PSpice A/D* en la actualidad carece de la capacidad para simular componentes analógicos conectados a los circuitos digitales que se modelan usando lenguaje de descripción de hardware (HDL), como VHDL o Verilog. Más del 60% de las PCB diseñadas hoy día contienen al menos un FPGA o CPLD. De ahí la necesidad de buscar la posibilidad de simular modelos VHDL en PSpice A/D. [13].

- **PROTEUS**: *Proteus Virtual System Modelling* (VSM), es una compilación de herramientas de diseño y simulación electrónica desarrollada por Labcenter Electronics, que combina la simulación de circuitos SPICE de modo mixto con componentes animados (LEDs, pantallas LCD, switches, botones) y modelos de microprocesadores, para facilitar la simulación de diseños basados en microcontroladores completos. Esta es una solución pionera, que generó beneficios en términos de reducción de tiempo y costos de desarrollo [22].

- **TINA Design Suite**: Es un potente paquete de programas para analizar, diseñar y probar en tiempo real circuitos analógicos, digitales, mixtos, HDL, MCU y circuitos impresos (PCB). TINA fue desarrollado por DesignSoft. Sus circuitos pueden contener bloques HDL editables de las bibliotecas de TINA, *Xilinx* u otros componentes HDL creados por el usuario o descargados de Internet. TINA compila HDL en un código de máquina altamente

eficiente. Puede combinar libremente macros SPICE y HDL en los componentes esquemáticos de TINA. También puede editar la fuente HDL de cualquier componente HDL y después simular y ver el resultado al instante. Con el depurador incorporado puede ejecutar el código HDL paso a paso, agregar puntos de interrupción, de observación, visualizar información variable y más [23].

- **PSIM**: Es un software de simulación y diseño de electrónica de potencia, caracterizado por ser uno de los más rápidos en esta área, manteniendo una excelente precisión en la simulación. PSIM tiene incorporado un compilador C que le permite introducir su propio código C directamente sin compilar. Esto hace que sea muy fácil y flexible para aplicar métodos de control con micro-controladores [20]. La suite de PSIM incluye el paquete ModCoupler Modules, que es un recurso para co-simulación entre PSIM y ModelSim. Existen dos versiones del enlace: ModCoupler-VHDL que soporta código VHDL y ModCoupler-Verilog que soporta código Verilog [21].

2.3.2.4. Co-simulación. Es un proceso de validación de diseños donde sus componentes son de diferentes tipos, como partes mecánicas, partes digitales, análogas o de hardware y software, con lo cual los componentes pueden ser descritos en diferentes niveles de abstracción. Por lo general, el modelado del diseño o sistema, se realiza a nivel de subsistemas sin tener en cuenta el problema de acoplamiento. La co-simulación acoplada se lleva a cabo mediante la ejecución de los subsistemas en forma de caja negra. Durante la co-

simulación los subsistemas intercambiarán datos entre los diferentes simuladores asociados a cada parte.

2.3.2.5. Simulación HIL. La simulación hardware en el lazo (HIL, *Hardware In the Loop*) es una técnica usada para el desarrollo y comprobación de sistemas de control complejos (sistemas embebidos sobre FPGA) en tiempo real. Con la simulación HIL la parte física de una máquina o sistema se sustituye por una simulación. Esto se realiza mediante modelos matemáticos de todos los sistemas dinámicos relacionados con la planta bajo control, formando lo que se denomina "simulación de la planta". El sistema de control que se está comprobando interactúa con esta simulación de la planta, haciéndolo creer que está interactuando con la planta real, metodología que permite la validación del controlador verificando la respuesta deseada del sistema. El objetivo de esta metodología es disminuir riesgos de avería en el proceso de prueba del controlador sobre la planta real, además permite el desarrollo y evaluación de nuevos algoritmos de control de una manera rápida. Gracias a que el controlador se desarrolla sobre plataformas reconfigurables (FPGA) podemos obtener un sistema de prueba económico y tiempos de desarrollo cortos [28].

3. CONTROL DIGITAL DE UN CONVERTIDOR DE POTENCIA

Una vez realizada la revisión general de las metodologías y herramientas de diseño (clásicas y contemporáneas) para sistemas electrónicos, se propondrá en el presente capítulo la manera de abordar una simulación mixta para verificar el comportamiento de una estrategia de control digital, sobre un circuito analógico correspondiente con un convertidor electrónico de potencia DC-DC.

3.1. DESCRIPCIÓN DEL SISTEMA A CONTROLAR

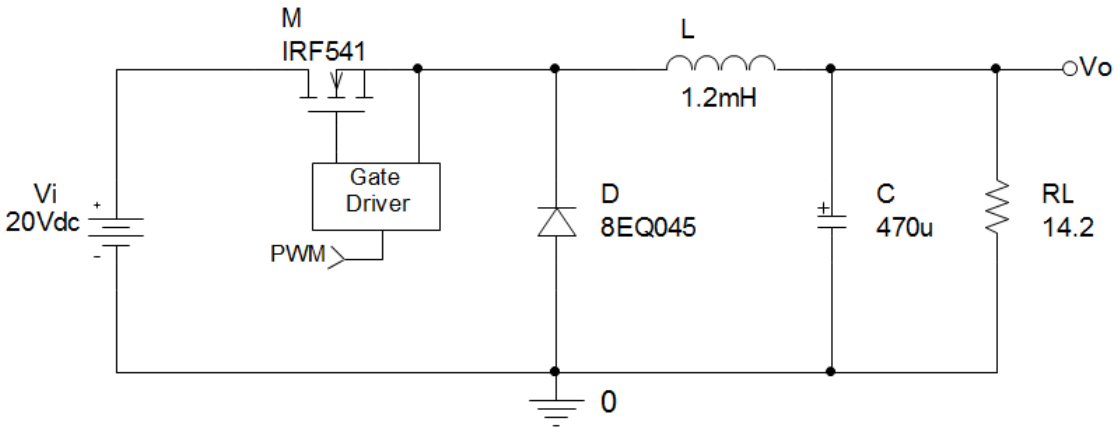
El sistema o planta a controlar, corresponde con un circuito convertidor de potencia DC-DC reductor o tipo *Buck*. Un convertidor de potencia DC-DC, es un circuito electrónico que convierte una tensión continua aplicada en su entrada, hacia niveles de tensión continua en su salida, mediante la conmutación de un dispositivo (transistor o tiristor). En el caso del convertidor *Buck*, la tensión entregada en su salida es menor que la aplicada en su entrada. La conmutación en este dispositivo permite reducir las pérdidas por disipación y por ende entregar potencia a la carga en un modo más eficiente (es decir, con menores pérdidas).

La Figura 3.1 ilustra el diagrama esquemático para la topología de un circuito convertidor de potencia DC-DC tipo *Buck*. Los valores de parámetro para el circuito fueron dimensionados considerando una tensión de entrada V_i = 20 VDC, una tensión de salida V_0 = 10 VDC, una frecuencia de conmutación f = 10 kHz y una resistencia de carga R_L = 14.2 Ω.

De esta manera, inicialmente se determina el ciclo útil nominal D mediante:

$$D = \frac{V_0}{V_i} = \frac{10}{20} = 0.5 \tag{3.1}$$

Figura 3.1. Convertidor de potencia DC-DC tipo *Buck*



Posteriormente, se determina la inductancia mínima L_{min}, como:

$$L_{min} = \frac{(1-D)R_L}{2f} = 355\ \mu H. \tag{3.2}$$

Este valor mínimo se satisface eligiendo un valor comercial de 1.2 mH para el inductor L.

Finalmente, el valor para la capacitancia mínima C_{min} del circuito, considerando una tensión de rizado ΔV_O del 0.6% en la salida, se calcula como:

$$C_{min} = \frac{1-D}{8L\left(\frac{\Delta V_o}{V_o}\right)f^2} \cong\ 87\ \mu F. \tag{3.3}$$

Este valor mínimo se satisface eligiendo un valor comercial de 470 µF para el capacitor C.

La Tabla 3.1, resume los valores de parámetro circuitales obtenidos. Para una mayor ilustración sobre los cálculos presentados y la elección de componentes, se recomienda al lector interesado consultar la referencia [29].

Tabla 3.1. Valores de parámetro para el circuito convertidor de potencia

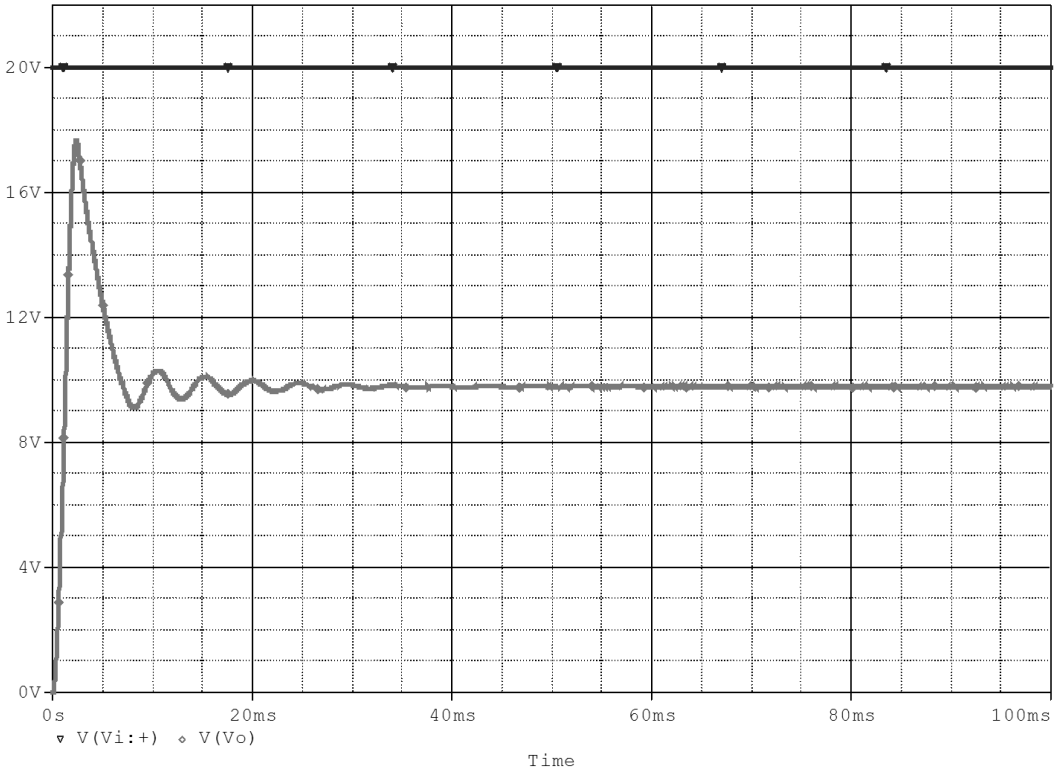
Parámetro	**Valor**
Tensión de entrada Vi	20 [V]
Tensión de carga Vo	10 [V]
Resistencia de carga R_L	14.2 [Ω]
Tensión de rizado ∆Vo	0.6% Vo
Frecuencia de conmutación	10 [kHz]
Ciclo útil del PWM	50 [%]
Inductancia L	1.2 [mH]
Capacitor C	470 [µF]

3.2. RESPUESTA DEL SISTEMA EN LAZO ABIERTO

El circuito convertidor de potencia de la Figura 3.1 fue implementado esquemáticamente (*Capture*) y simulado (*PSpice A/D*), en la suite de OrCAD-Cadence®. En el circuito, la acción de conmutación sobre la compuerta del interruptor es realizada a través de una señal de PWM nominal (con *f* = 10 kHz y *D* = 0.5, según calculado).

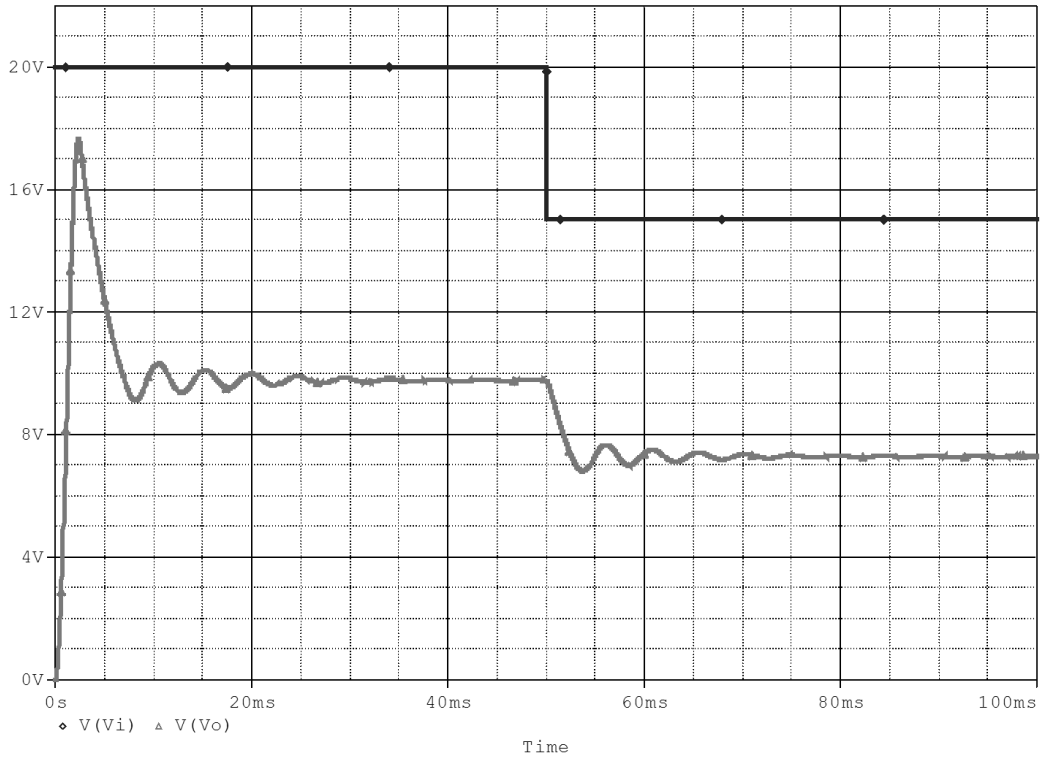
Como resultado, la Figura 3.2 permite observar la respuesta del sistema (tensión de salida V_0) cuando la tensión de suministro V_i es un valor fijo de 20 VDC de tensión. A partir de la gráfica es evidente como el sistema alcanza el valor deseado de 10 VDC posterior a un transitorio de alrededor 30 ms, verificando la pertinencia del diseño y por ende, el buen funcionamiento del convertidor Buck.

Figura 3.2. Respuesta del circuito convertidor de potencia *Buck* en lazo abierto



Posteriormente, la Figura 3.3 permite observar la respuesta del sistema cuando se presenta una perturbación en la tensión de suministro V_i correspondiente a una reducción del 25% en su valor a partir de 50 ms. Como se observa de la gráfica, el sistema no consigue la tensión deseada en su salida, la cual cae hasta un valor no deseado de alrededor 7 VDC. Este hecho justifica la necesidad de incluir una acción de control, que permita regular los niveles de tensión en la salida del circuito a pesar de la influencia de perturbaciones.

Figura 3.3. Respuesta del circuito convertidor de potencia *Buck* en lazo abierto ante perturbación en la tensión de suministro

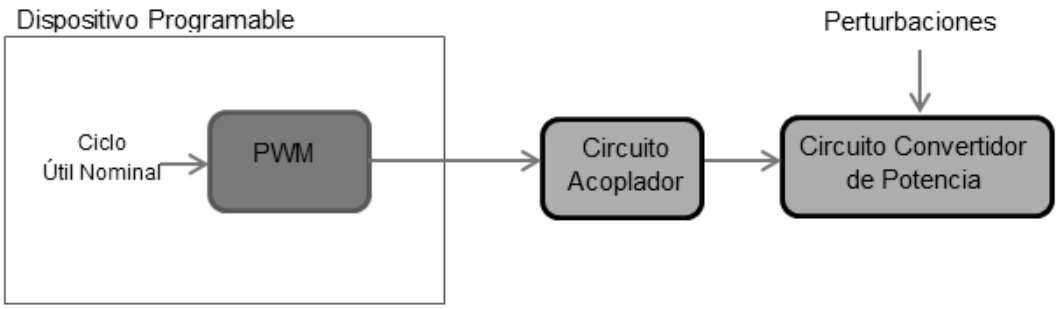


3.3. RESPUESTA DEL SISTEMA EN LAZO CERRADO

En el camino hacia obtener una simulación del sistema controlado en lazo cerrado, el primer paso corresponde con la verificación del comportamiento de la planta en lazo abierto, gobernado desde el dispositivo digital de proceso. En este paso inicial, se buscará replicar los resultados obtenidos en la sección 3.2 a partir de la construcción del subsistema de trayectoria directa ilustrado en la Figura 3.4.

Para este fin se requiere programar el bloque de actuación correspondiente con un modulador de ancho de pulso (generador PWM), para lo cual se realizará la síntesis de un bloque en PSpice A/D que facilite la simulación para esta componente interactuando con el circuito convertidor de potencia (Figura 3.1).

Figura 3.4. Diagrama de bloques de trayectoria directa con PWM generado en VHDL



En el Capítulo 2, se presentaron las bases conceptuales necesarias para entender la diferencia entre una simulación analógica y una digital; además se abordó la problemática a la hora de comprobar diseños de señal mixta, debido a la escasez en el mercado para esta clase de herramientas de simulación. Este último caso, será el requerido para diseñar un control digital que actúe sobre el convertidor *Buck*.

En particular, se empleará como base la metodología propuesta en [7] para desarrollar sistemas electrónicos con señales de naturaleza mixta, empleando PSpice A/D y componentes digitales desarrollados en código VHDL. Dicha metodología, se divide en dos ramas, analógica y digital, según se muestra en la Figura 3.5.

Mientras los circuitos analógicos están diseñados siguiendo la metodología tradicional de diseño Bottom-Up (en el editor de esquemas de PSpice A/D), la parte digital en VHDL sigue la metodología Top-Down.

Es bien sabido que PSpice A/D es configurable para realizar simulaciones de señal mixta (véase por ejemplo [30]), sin embargo, este carece de habilidad para simular componentes analógicos interconectados a circuitos digitales modelados

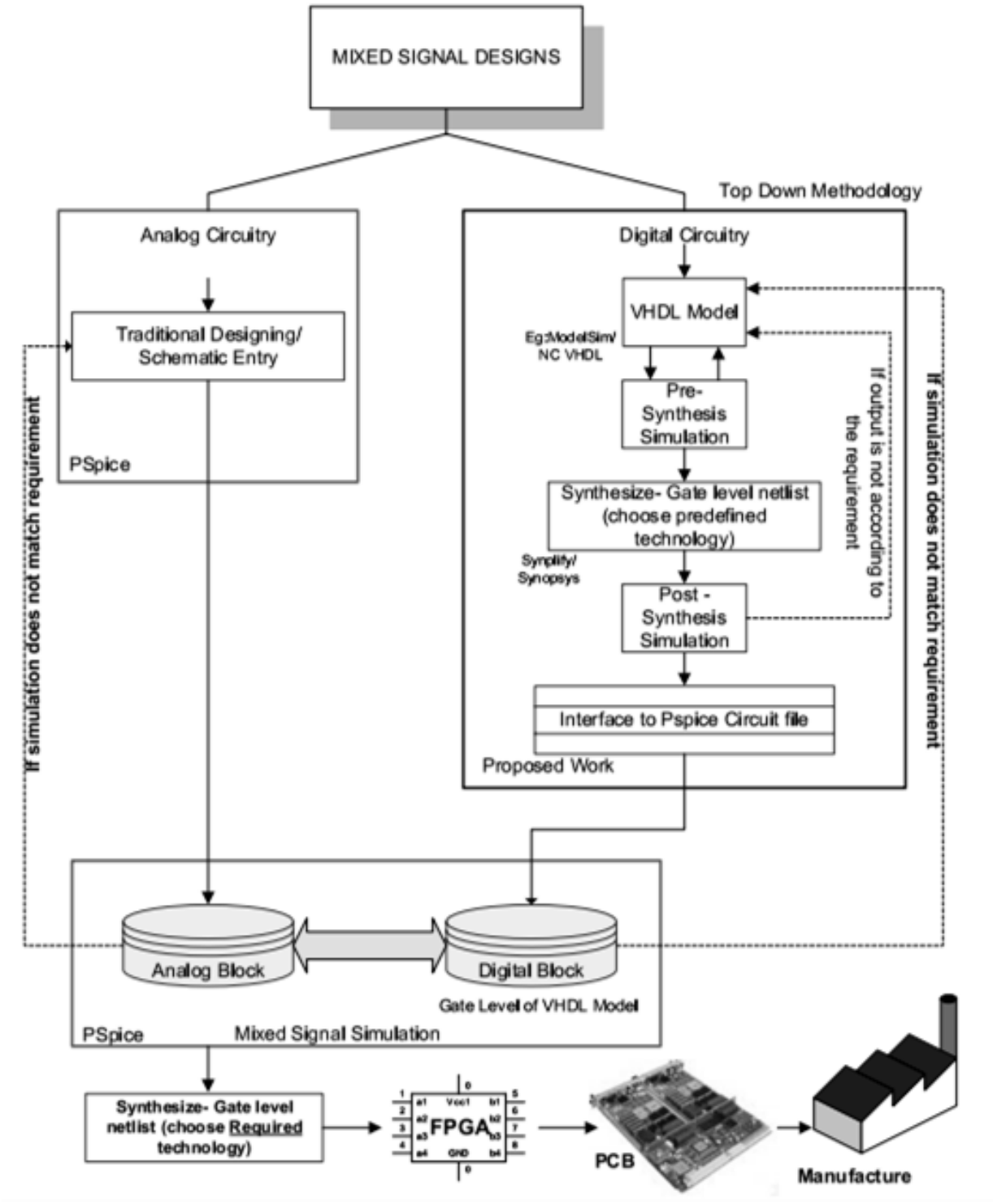
utilizando lenguajes de descripción de hardware, como VHDL o Verilog, limitando por tanto los niveles de abstracción accesibles en el diseño. Adicionalmente, la mayoría de construcciones digitales en PSpice A/D están disponibles sólo a nivel de compuertas.

Por lo tanto para llevar a cabo una simulación digital en PSpice A/D de componentes VHDL, estas deben ser traducidas a nivel de compuertas. Lo anterior se obtiene empleando la herramienta *Synplify Pro* (mencionada en la sección 2.3.2.2), la cual será responsable de convertir el código HDL en un archivo netlist apropiado para la arquitectura del fabricante seleccionado.

Sin embargo, es necesario realizar una conversión entre este archivo a nivel de compuertas (netlist entregado por *Synplify Pro*) y el tipo de formato reconocido por PSpice A/D. Para ello se hará uso de la herramienta *VHDL2PSpice*, desarrollada por Sreeram Rajagopalan en su proyecto de tesis de maestría [31] y la cual fue suministrada al director del presente proyecto (profesor Ricardo Alzate), vía correo electrónico en julio de 2012, por parte del *Sr. Manny Marcano*, presidente de la compañía *EMA Design Automation Inc*. en los Estados Unidos, patrocinador del citado trabajo de desarrollo de tecnología.

Cabe anotar que dicha herramienta, y por ende, los desarrollos presentados, han sido realizados tomando en cuenta una tecnología de implementación *LATTICE MACH* de la familia 111.

Figura 3.5. Metodología de diseño para sistemas de señal mixta



Fuente: [7]

3.3.1. Simulación de modelos VHDL en PSpice A/D

Por tanto, para generar el bloque PWM, se asume un tamaño de palabra de 8 bits y una estructura simple que consta de un contador y un comparador. El contador permite generar una señal diente de sierra (entre 0 y 2^N, donde N=8 es la resolución en bits) que es comparada todo el tiempo con la entrada (ciclo de trabajo, entre 0 y 2^N). De esta manera el comparador va a producir un '1' mientras el valor de la señal diente de sierra sea menor a la entrada, de lo contrario la salida es un '0'. Es así como es generada la señal de PWM con ciclo útil variable.

Este procedimiento es codificado en VHDL empleando la herramienta *Xilinx ISE*. Posteriormente, se realiza sobre el código la transformación a nivel de compuertas (*netlist, Synplify Pro*) y posterior conversión a PSpice A/D (VHDL2PSpice), siguiendo el procedimiento descrito en el Anexo A.

El diagrama esquemático resultante para el bloque de PWM generado, interconectado con el circuito convertidor de potencia se incluye en la Figura 3.6, en la cual, el ciclo útil o valor de entrada al actuador (PWM) es una señal binaria de 8 bits, con valor nominal D = 0.5 ajustado a la representación 0x10000000.

Los resultados obtenidos tras simular el circuito en PSpice A/D se muestran en la Figura 3.7, presentando un comportamiento idéntico al obtenido en la Figura 3.3, bajo las mismas condiciones de simulación, lo cual comprueba la efectividad de la metodología de diseño del sistema de simulación mixta y de la correspondiente simulación del circuito resultante.

Figura 3.6. Esquemático del circuito convertidor de potencia en lazo abierto con PWM generado en VHDL

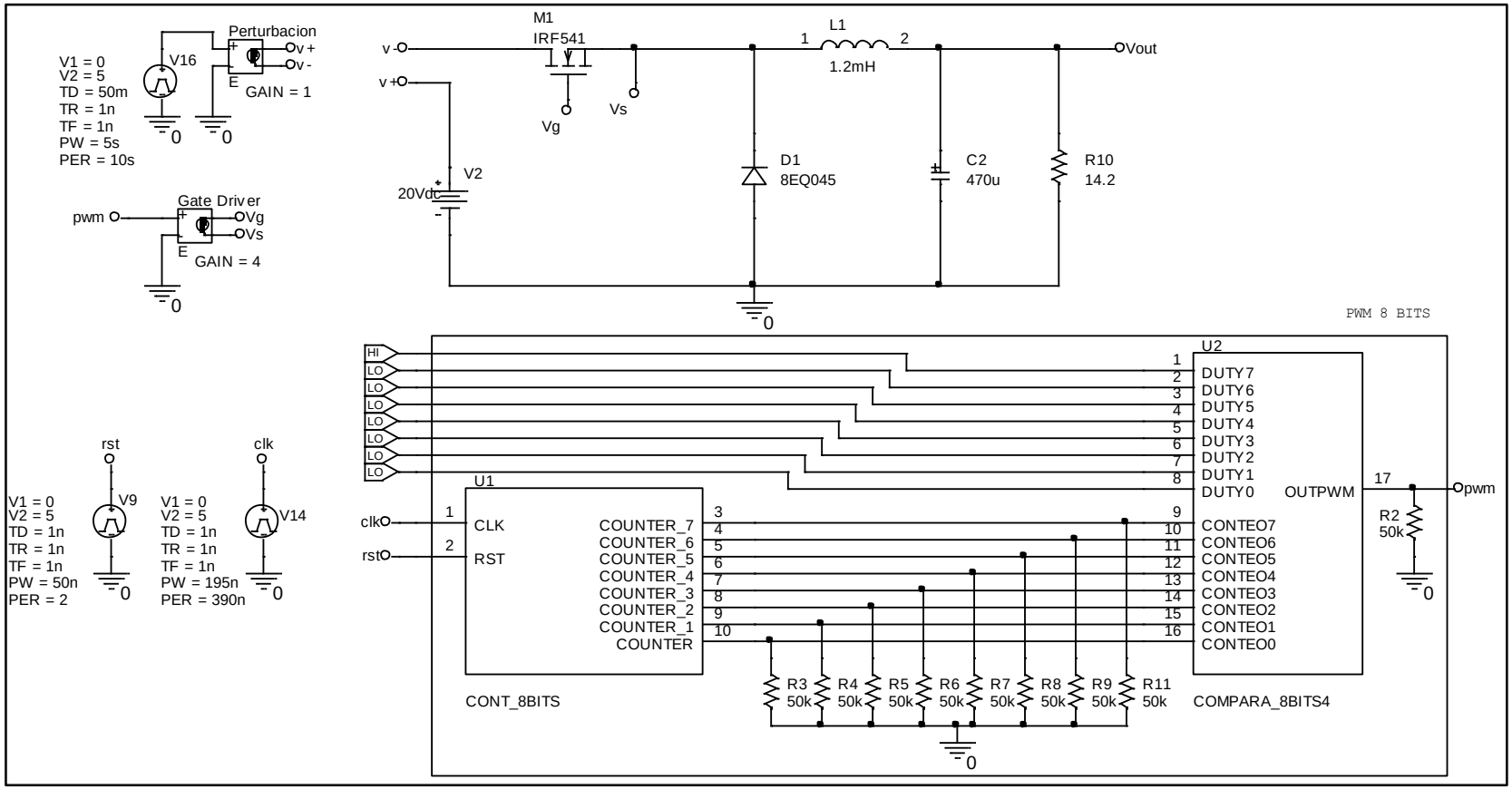
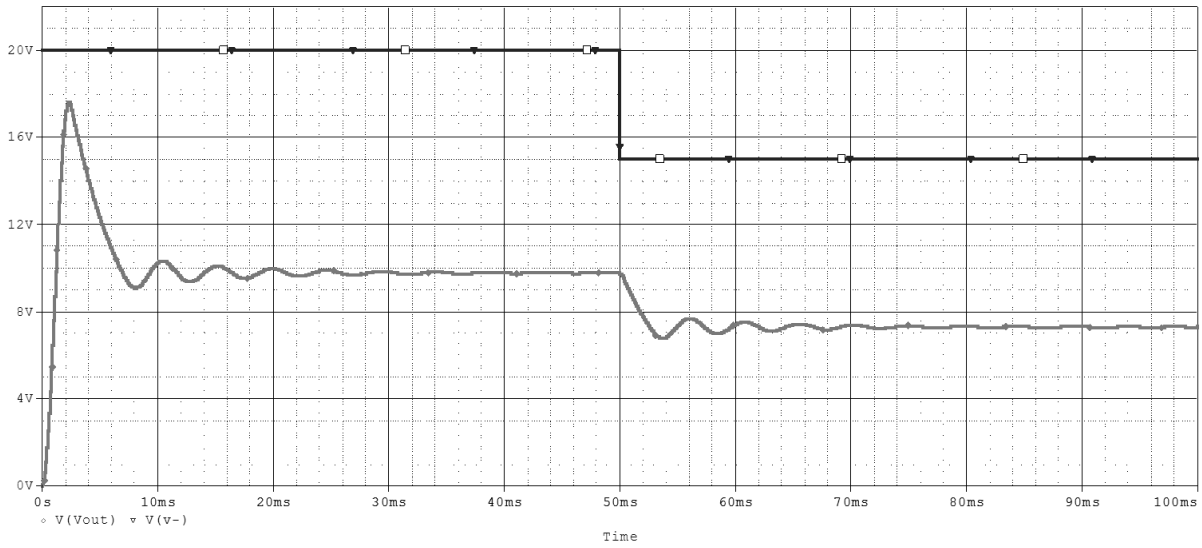


Figura 3.7. Respuesta del circuito convertidor de potencia *Buck* en lazo abierto ante perturbación en la tensión de suministro, para PWM generado en VHDL

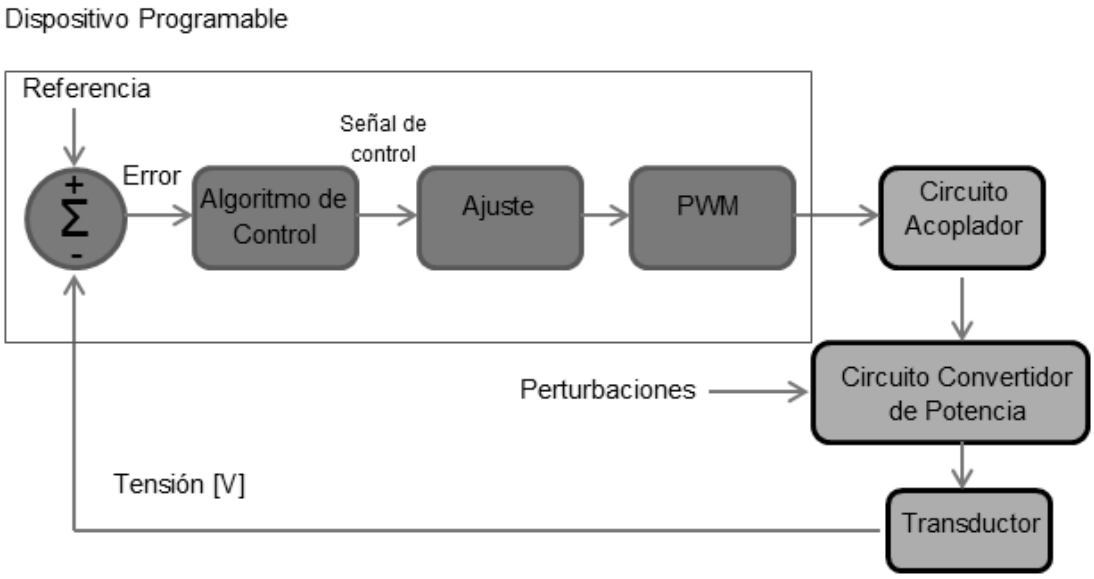


3.3.2. Ajuste de rangos para el lazo de control

Para incorporar una acción de control en el circuito, se requiere implementar un lazo realimentado como el ilustrado en la Figura 3.8. Desde el punto de vista práctico, este cierre de lazo se efectúa en un dispositivo de proceso digital, al cual se le asigna un valor de referencia preprogramado que se compara (resta) con la medida proveniente del sensor, para crear una señal de error, posteriormente transformada por la acción de control, y que finalmente es aplicada como entrada de la planta, previo a ser ajustada por un bloque de acondicionamiento de señal.

Es importante entonces, asegurar que los niveles de señal al interior del lazo de control corresponden con valores coherentes, que eviten operar en valores saturados que reducirían el desempeño de la acción de control.

Figura 3.8. Diagrama de bloques para sistema en lazo cerrado

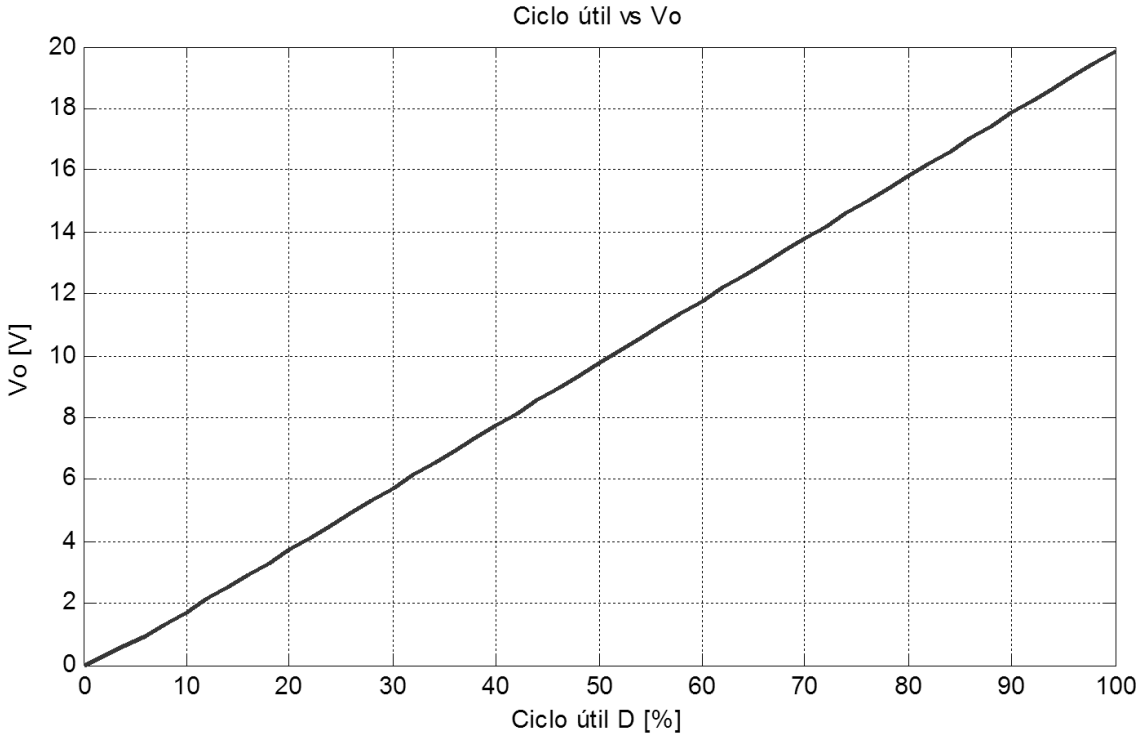


Este aspecto puede considerarse aún más crítico, tomando en cuenta que se dispone de una resolución de 8 bits para representar las cantidades en el dispositivo de proceso digital seleccionado.

Por tal razón, el segundo paso hacia una configuración apropiada del lazo de control (posterior al primero, ya ejecutado, consistente en el diseño apropiado de un actuador), corresponde a la caracterización del comportamiento de la planta (sistema en lazo abierto, PWM + circuito *Buck*) para determinar los rangos efectivos de operación del sistema.

La Figura 3.9, muestra la relación entrada (ciclo útil D) / salida (voltaje V_o en la carga) para valores en estado estacionario. De la misma se puede concluir que dicha relación es estrictamente lineal, sin zonas muertas o regiones con saturación, lo cual sugiere un comportamiento fácilmente ajustable para valores alrededor del valor nominal, seleccionado como el punto medio del rango efectivo; es decir: (V_o, D) = (10, 50).

Figura 3.9. Caracterización del sistema en lazo abierto



Por tanto, se seleccionan como valores de rango prácticos para el ciclo útil $D \in [0, 100]$ y con base en ellos, se determinan como valores límite de error $e(t) \in [-10,\ 10]$.

Ahora bien, al cerrarse el lazo de control se constituye de manera natural una acción de control proporcional de ganancia unitaria, es decir:

$$u(t) = e(t),$$

a partir de ello, deberá construirse una función de mapeo que realice la conversión de escala entre las unidades de la señal de control $u(t)$ (en este caso equivalente al error $e(t)$) y la entrada a la planta (ciclo útil D para el PWM), de manera tal que a error cero se obtenga el valor nominal y para otros valores se asegure una

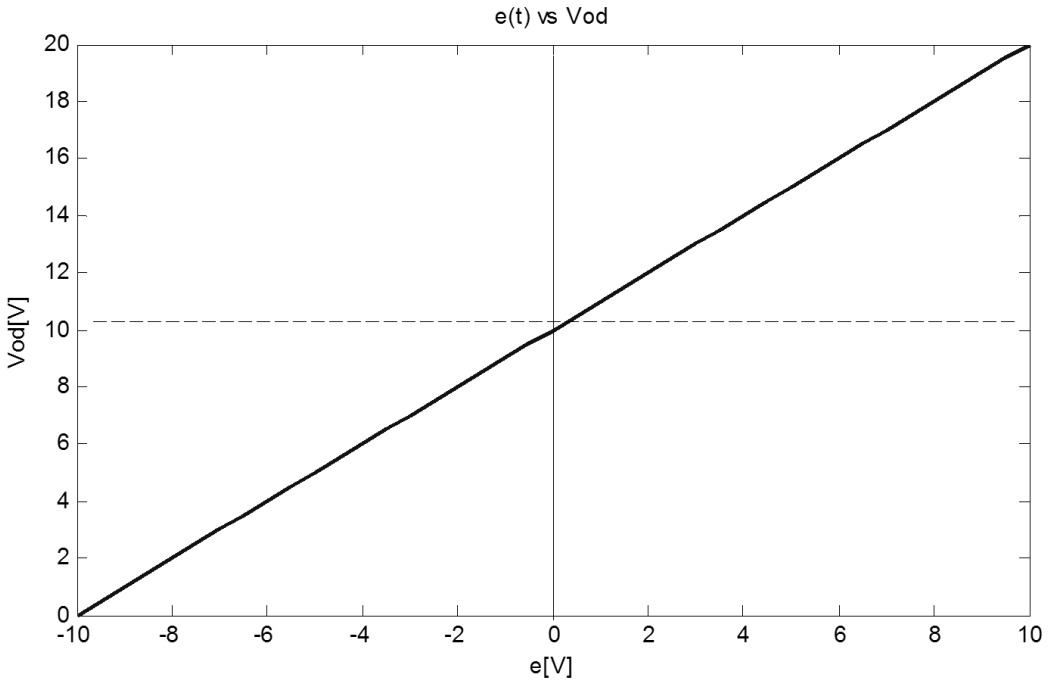
excursión al interior de los límites máximos pre-establecidos. Esta función de mapeo se presenta como sigue:

1) A partir de la Figura 3.9, se determina una relación matemática entre D y V_0, dada por la siguiente expresión:

$$V_0 = 20\,D,$$

2) Tomando en consideración los valores límites del error $e(t)$, la equivalencia entre este error y la señal de control $u(t)$, además de los valores deseados de salida, se construye el siguiente gráfico:

Figura 3.10. Relación matemática entre el error y el voltaje de salida deseado



A partir de lo anterior, se determina la relación matemática entre el error $e(t)$ (equivalente a la señal de control $u(t)$) y el valor deseado de la salida V_{0d} dada por:

$$V_{0d} = e(t) + 10,$$

3) La combinación de las dos expresiones anteriores, permite obtener una relación entre la señal de control $u(t)$ y el ciclo útil D definida mediante la expresión:

$$D = \frac{V_{0d}}{20} = \frac{1}{20}(e(t) + 10) = \frac{1}{20}e(t) + \frac{1}{2},$$

que constituye el mapeo requerido, siendo equivalente al acondicionamiento de señal denominado *"Ajuste"* en el diagrama de bloques de la Figura 3.8.

Por tanto, la síntesis VHDL para esta función de ajuste en cascada con el PWM (tomando en cuenta que el error se calcula como la diferencia entre una referencia preprogramada en el valor nominal deseado de salida V_{ref} = 0x10000000 y la lectura del sensor correspondiente con un convertidor analógico a digital de 8 bits) se realizó conforme a la manera descrita en el Anexo A, permitiendo obtener el esquema de la Figura 3.11 para PSpice A/D, con simulación asociada ilustrada en la Figura 3.12, donde se observa que sin perturbación el sistema en lazo cerrado obtiene el valor nominal deseado, mientras que una vez perturbado en alrededor de 50 ms, permite obtener una degradación levemente menor al caso verificado en lazo abierto, lo cual coincide con las predicciones teóricas de un sistema realimentado, reafirmando la validez del diseño y ajuste del lazo de control digital.

Figura 3.11. Esquemático del circuito convertidor de potencia en lazo cerrado generado en VHDL con ajuste de rango

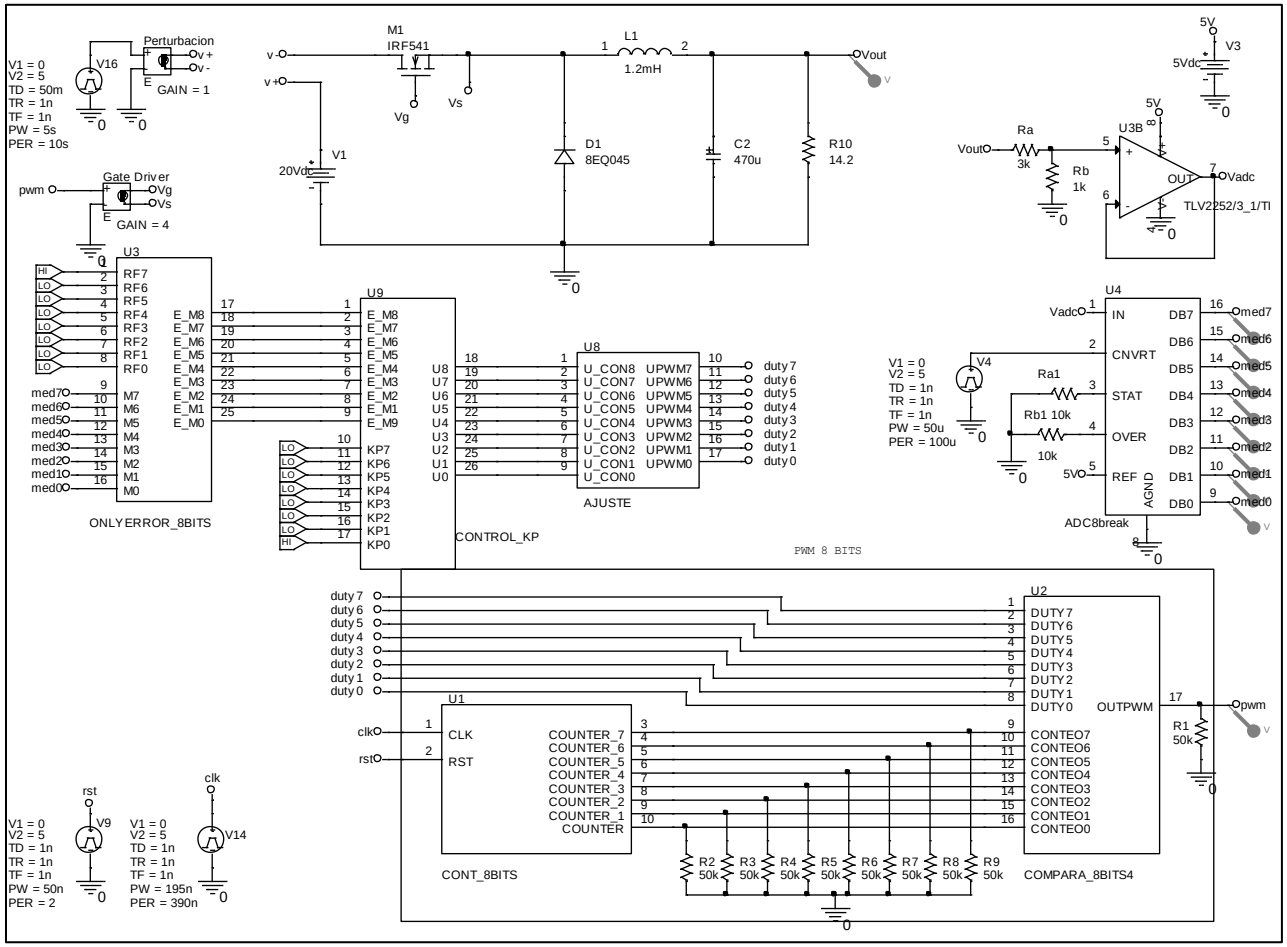
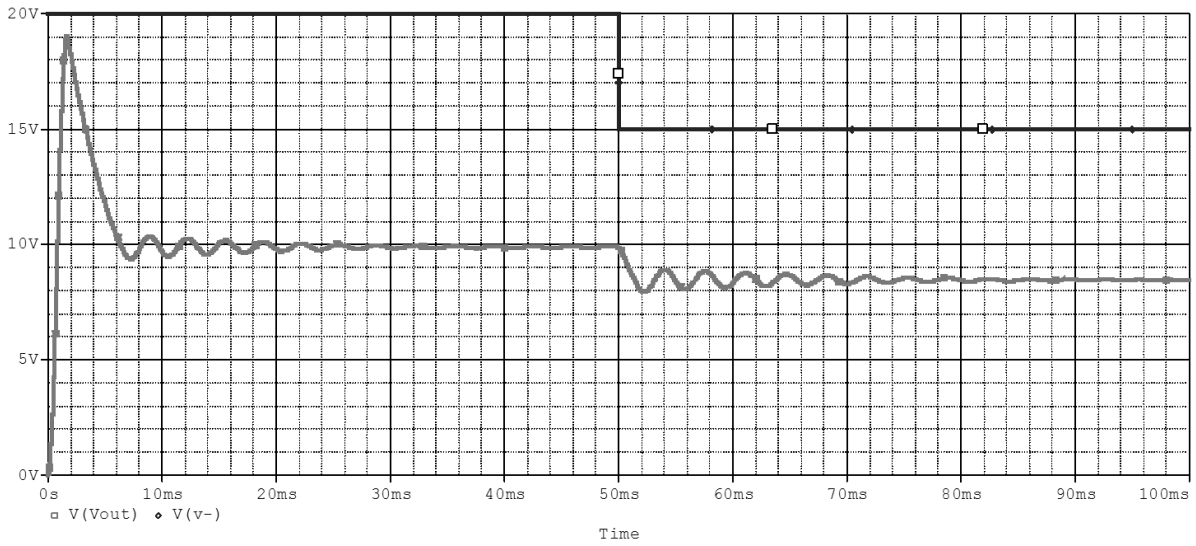


Figura 3.12. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado ante perturbación en la tensión de suministro, para PWM generado en VHDL



4. RESULTADOS PARA ALGORITMOS DE CONTROL IMPLEMENTADOS

Posterior al diseño y calibración de las etapas requeridas para cerrar el lazo de control, se presenta en este apartado del texto la verificación del comportamiento para el sistema controlado empleando leyes de tipo proporcional y por modos deslizantes, buscando corroborar la validez de la metodología de diseño propuesta, para controladores digitales de circuitos convertidores de potencia (analógicos), a partir de herramientas de simulación para sistemas de señal mixta.

4.1. ANÁLISIS DE GANANCIA PROPORCIONAL

Como ya mencionado en la sección 3.3.2, al cerrar el lazo de control se constituye de manera natural una acción de control proporcional de ganancia unitaria. Este caso es por tanto una versión particular del controlador proporcional, definido mediante:

$$u(t) = Ke(t),$$

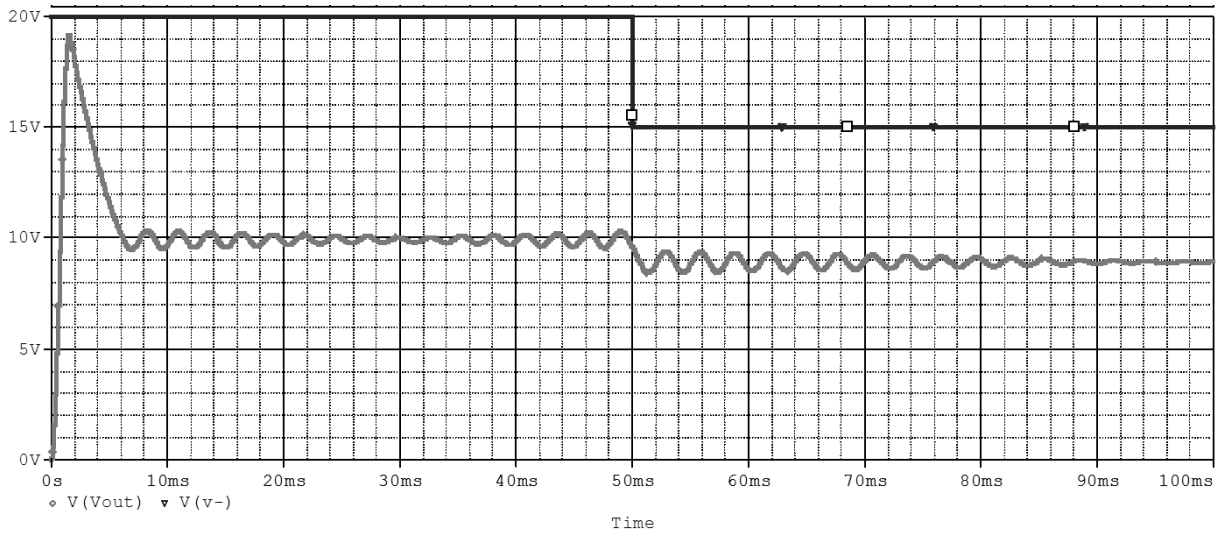
siendo K la ganancia de lazo directo o ganancia proporcional. Por tanto, y atendiendo a las predicciones sugeridas en la literatura [32], se analizará inicialmente el efecto de atenuación sobre las perturbaciones del sistema para diferentes valores de dicha ganancia de lazo.

La perturbación considerada será un decremento en la tensión de suministro V_i en el circuito convertidor de potencia, pasando de 20 VDC a 15 VDC en 50 ms (escenario idéntico al considerado en las Figuras 3.3, 3.7 y 3.11).

De esta manera, la Figura 4.1 muestra el comportamiento del sistema perturbado en lazo cerrado para el caso en que la ganancia de lazo K corresponde con 2 unidades. Como se observa, es visible el efecto de la perturbación aplicada en 50

ms, aunque con menor incidencia si se compara con el caso de ganancia unitaria presentado en la Figura 3.11, pues en este caso el valor final alcanza los 9 VDC.

Figura 4.1. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado perturbado ante acción proporcional programada en VHDL, caso $\boldsymbol{K}$ = 2



Este comportamiento se verifica para otros valores de ganancia de lazo (K = 4 en la Figura 4.2, K = 8 en la Figura 4.3 y K = 16 en la Figura 4.4), permitiendo comprobar cómo efectivamente a medida que se incrementa el valor de la ganancia, se reduce en la misma medida, el efecto de la perturbación aplicada en el sistema.

Este resultado no sólo permite corroborar las predicciones teóricas, sino que, a partir del él también se valida el ajuste apropiado para los rangos de señal en el lazo de control y la correcta síntesis VHDL de los modelos digitales incorporados en la simulación. Particularmente, es conveniente aclarar que la acción de control proporcional se codifica de manera sencilla en VHDL e incorpora en el esquema de lazo cerrado ilustrado en la Figura 3.8, en el bloque denominado “Algoritmo de Control”.

Figura 4.2. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado perturbado ante acción proporcional programada en VHDL, caso $\boldsymbol{K}$ = 4

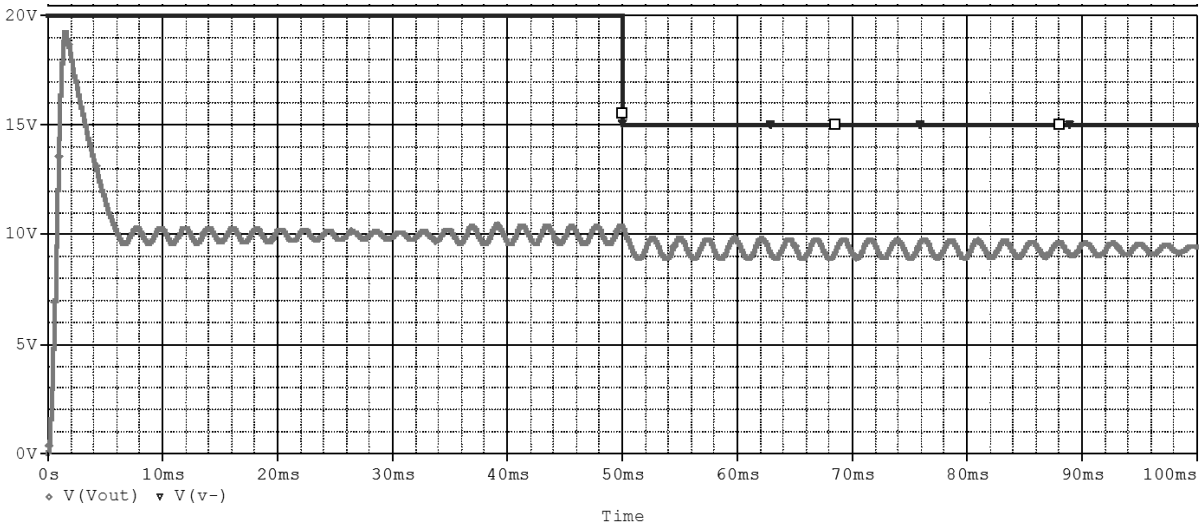


Figura 4.3. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado perturbado ante acción proporcional programada en VHDL, caso $\boldsymbol{K}$ = 8

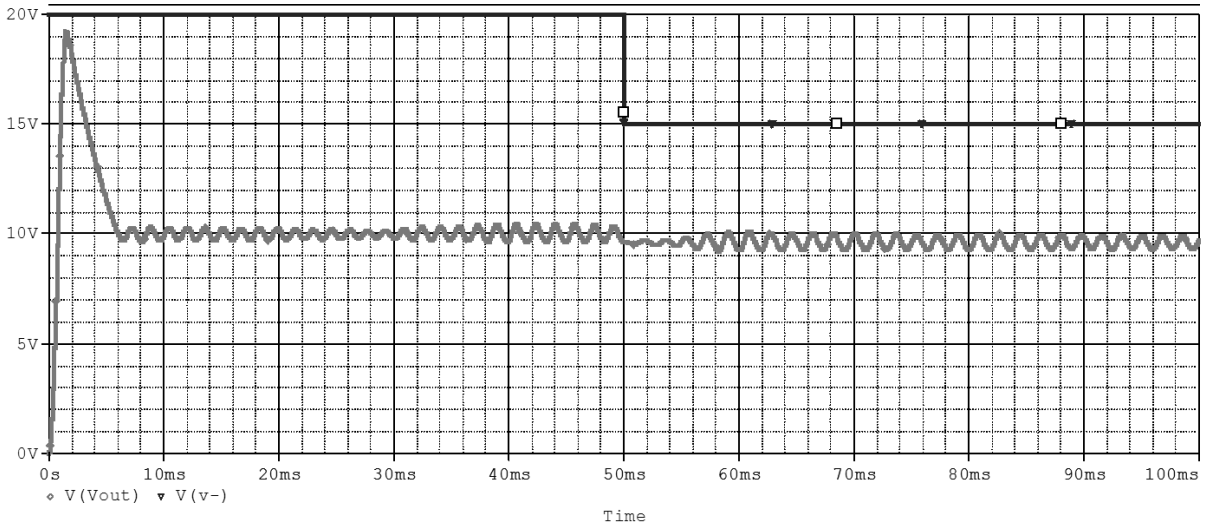
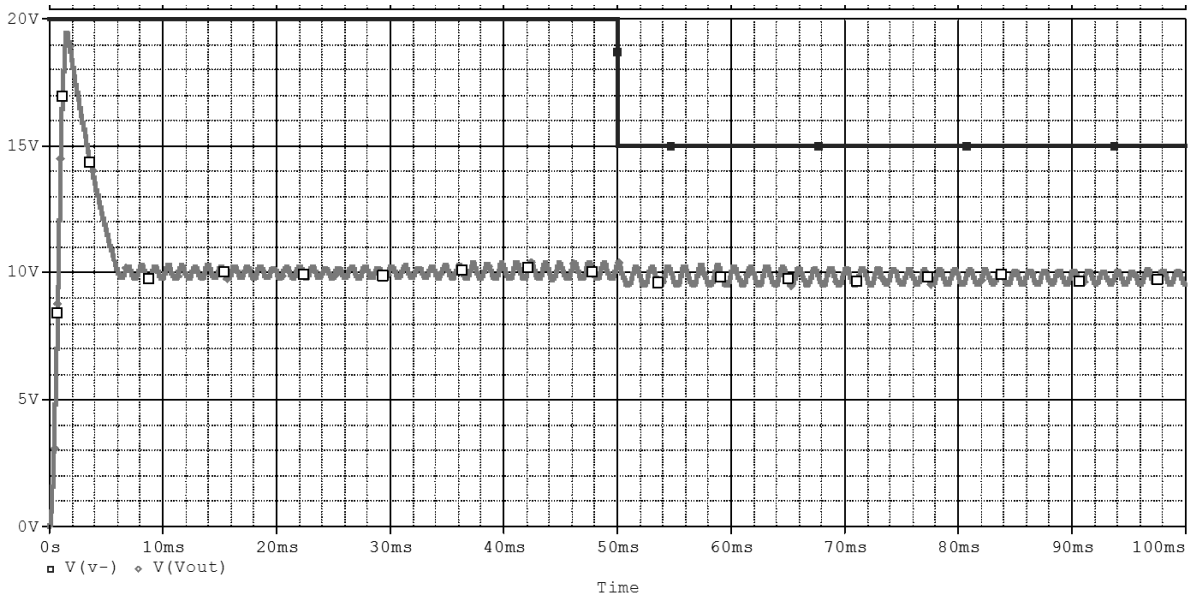


Figura 4.4. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado perturbado ante acción proporcional programada en VHDL, caso $\boldsymbol{K}$ = 16



4.2. CONTROL POR MODOS DESLIZANTES

La estructura configurada hasta este punto, permite configurar de manera simple, estrategias de control convencional como un PID o cualquier otro tipo de compensación básica diseñada a partir de métodos clásicos.

Sin embargo, existen otro tipo de técnicas (como son las estrategias de control en el espacio de estados) que revisten un mayor interés para la comunidad científica debido a su mejorado desempeño en aplicaciones. En particular, para el caso de control en circuitos convertidores de potencia, es común el empleo de la técnica de control por modos deslizantes [29].

La técnica de control por modos deslizantes (SMC de su acrónimo en inglés) constituye una acción de tipo robusto en el espacio de estados que, fundamentalmente reduce de un espacio n-dimensional a un caso escalar, el

problema de convergencia asintótica hacia cero del error en el lazo de realimentación del sistema.

En otras palabras, si se define como superficie de deslizamiento $s(t)$ a la diferencia entre el valor deseado $x_{1d}(t)$ y el medido $x_1(t)$ para la tensión de salida del convertidor de potencia en la resistencia de carga R_L, es decir:

$$s(t) = x_1(t) - x_{1d}(t) = -e(t),$$

es posible forzar la convergencia de la trayectoria en el espacio de estados a caer en un tiempo finito sobre dicha superficie, mediante la acción de control por modos deslizantes definida por:

$$u(t) = \frac{1}{2}\big(1 - sgn(s)\big),$$

donde:

$$sgn(s) = \begin{Bmatrix} 1, \ si \ \ s \geq 0 \\ -1, \ si \ \ s < 0 \end{Bmatrix}.$$

Por tanto, la señal de control presentará un comportamiento definido por los siguientes dos valores:

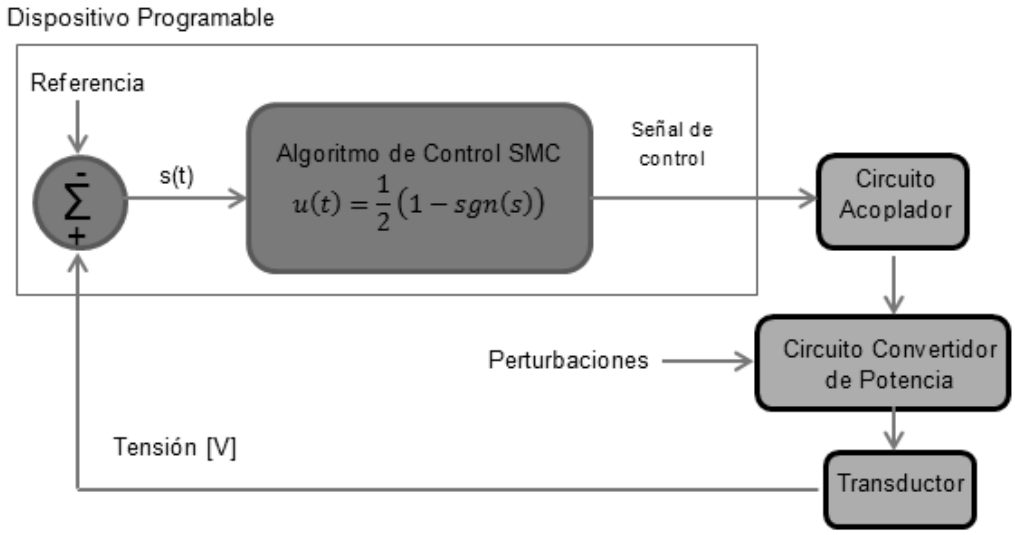
$$u(t) = \begin{Bmatrix} 0\,, \ si \ \ e < 0 \\ 1, \ si \ \ e \geq 0 \end{Bmatrix},$$

sugiriendo un comportamiento equivalente al de un controlador encendido-apagado. Note como el comportamiento de $u(t)$ relaciona de manera opuesta el error de medida $e(t)$ y la superficie de deslizamiento $s(t)$, lo cual equivale a un acondicionamiento de señal que evidencia la filosofía del control deslizante; es decir, la acción de control se realiza en una dirección opuesta al cambio de la señal de control para buscar compensarlo.

También debe observarse como la salida del control sugiere un ciclo útil variante, dependiendo del valor asignado a la señal $u(t)$. Así, la interpretación práctica será $u(t) = D$ y por tanto será más conveniente hacer que dicha señal de control sea directamente aplicada como gobierno de conmutación en el circuito, eliminando la necesidad del bloque PWM. Esto aumenta la velocidad de respuesta del sistema controlado puesto que el modo de deslizamiento equivale con un PWM de frecuencia variable.

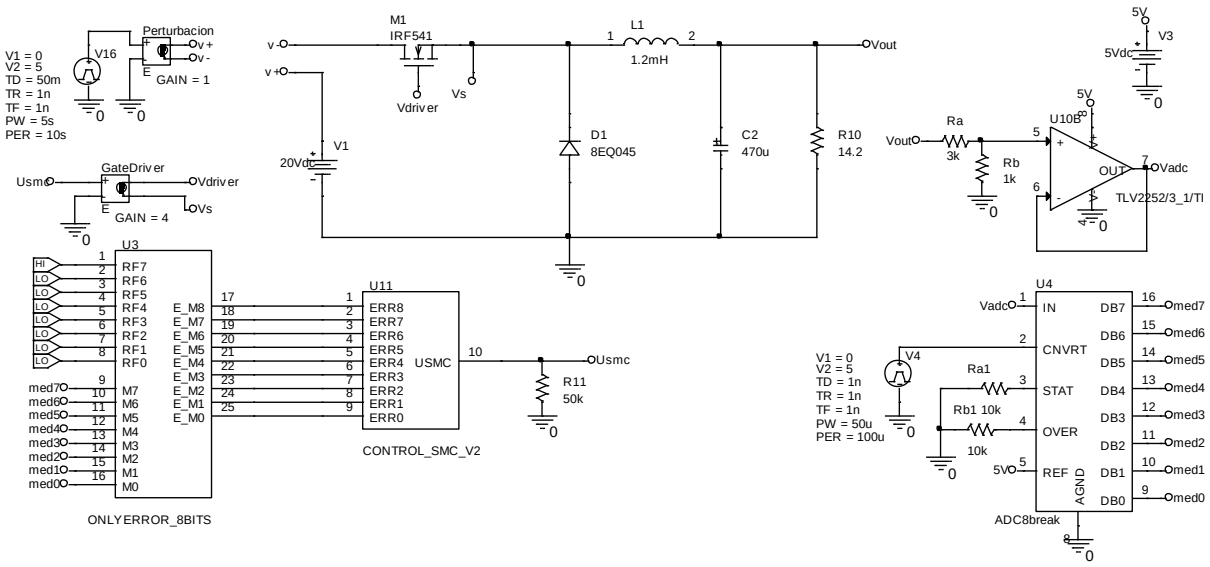
De esta manera, el esquema de lazo cerrado de la Figura 3.8 se modifica a la configuración ilustrada en la Figura 4.5, donde se observa que la ley de control no requiere el bloque de “Ajuste” ni el “PWM”.

Figura 4.5. Sistema en lazo cerrado controlado por modos deslizantes



En consecuencia, el esquema simulado en PSpice A/D se reduce de manera ostensible comparado con la implementación inicial propuesta en la Figura 3.11, según se muestra en la Figura 4.6.

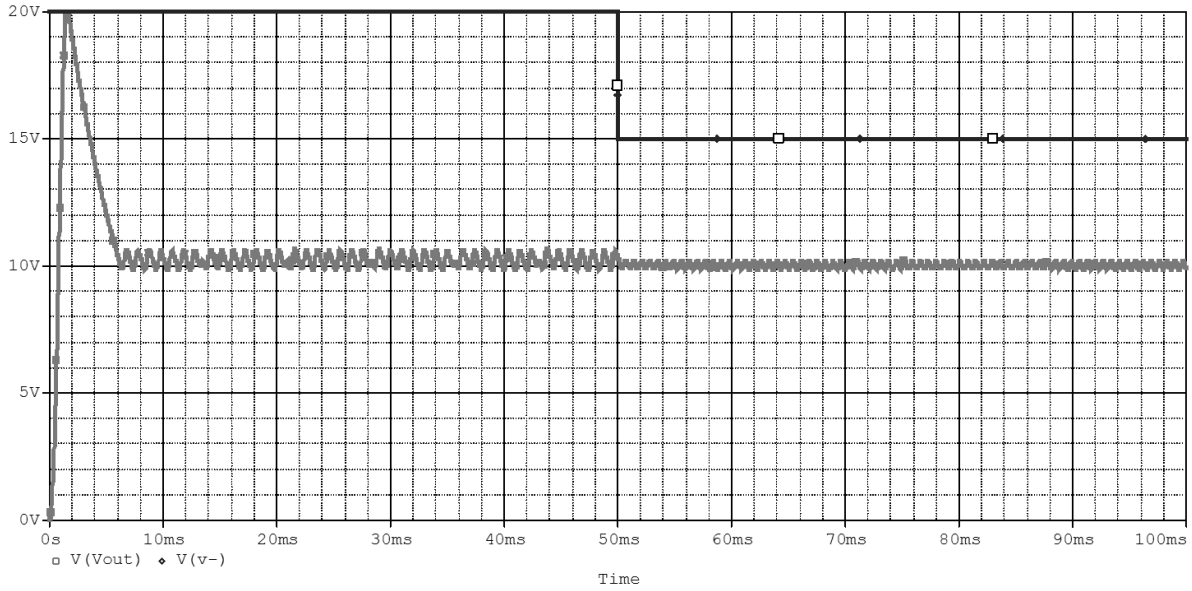
Figura 4.6. Esquemático del circuito convertidor de potencia en lazo cerrado generado en VHDL para control por modos deslizantes



Finalmente, la simulación del sistema controlado en la Figura 4.7 permite visualizar el apropiado comportamiento del control por modos deslizantes propuesto para atenuar el efecto de perturbaciones en el voltaje de salida del circuito. Nótese como se obtienen oscilaciones de alta frecuencia (e incluso de menor rizado) a partir del cambio de tensión de entrada acontecido a 50 ms. También, es visible como se retoma el valor final deseado, siendo un desempeño mejor comparado con los resultados de la Figura 4.4.

De esta manera se verifica aún más la pertinencia de los diseños presentados y la validez de los procedimientos de diseño digital y codificación VHDL empleados, para simular el sistema de señal mixta correspondiente con el conjunto controlador digital + convertidor de potencia DC-DC.

Figura 4.7. Respuesta del circuito convertidor de potencia *Buck* en lazo cerrado perturbado ante acción por modos deslizantes programada en VHDL



5. CONCLUSIONES

Con base en los desarrollos y resultados presentados en el presente proyecto de grado, se puede concluir que:

- Se identificaron herramientas computacionales para el diseño de sistemas electrónicos de señal mixta, a partir de un estudio de metodologías de diseño electrónico y revisión de herramientas EDA disponibles en el mercado, para analizar circuitos con señales de naturaleza analógica y digital combinadas (Capítulo 2). En particular, se empleó el paquete comercial de simulación de circuitos PSpice A/D como esquemático principal, combinado con *Synplify Pro* y *VHDL2Pspice* para generar componentes circuitales digitales, compiladas originalmente en VHDL desde *Xilinx ISE*.

- Se simuló un algoritmo de control digital en un simulador convencional de circuitos. Para ello se siguió la metodología de diseño sugerida por S. Rajagopalan [31]. Inicialmente, se analizó un circuito convertidor de potencia DC-DC reductor (*Buck*) en PSpice A/D. Posteriormente se sintetizó desde VHDL un bloque actuador representado por un generador PWM, que gobernó el disparo del dispositivo de conmutación en el circuito. A través de simulación, se pudo constatar que la respuesta del circuito en el tiempo coincide para los casos en que el PWM es generado a través de una fuente de señal (componente PSpice) y mediante el PWM sintetizado en VHDL (a partir de la nueva metodología). Con base en ello, el parámetro de control se constituye en el ciclo útil variable de la señal modulada y permite manipular el comportamiento del circuito (Capítulo 3).

- Se verificó en simulación un algoritmo de control digital sobre un convertidor de potencia (Capítulo 4). A partir del componente de actuación desarrollado (PWM + ciclo útil), se constituye la función de trayectoria directa al acoplarse en cascada con la planta (circuito convertidor). Para aplicar una estrategia de control realimentado se debió ajustar el rango de las variables de control. Para verificar la pertinencia del cierre de lazo digital se comparó el comportamiento de lazo abierto con el comportamiento realimentado, para varios valores de ganancia. Una vez verificado el apropiado desempeño del sistema realimentado ante la acción de perturbaciones, se implementó una ley de control por modos deslizantes que corrobora el apropiado desempeño del sistema controlado y, por tanto, de la metodología de diseño de sistemas de señal mixta propuesta.

- Todo lo anterior demuestra que es posible, viable y oportuno emplear herramientas computacionales para simular sistemas continuos y discretos, como parte del proceso de diseño de controladores digitales aplicados a circuitos eléctricos.

6. RECOMENDACIONES

La herramienta de conversión *VHDL2PSpice*, empleada para traducir el código RTL a nivel de compuertas sintetizado por *Synplify Pro*, hacia componentes PSpice A/D, presenta ciertos errores al producir librerías.

En particular, al ser la librería un archivo de texto, los nombres de entradas y salidas generados, algunas veces no presentan apropiada sintaxis (separación por espacio en blanco) para su individualización como terminal de entrada o salida. Otro error común es cuando un terminal no es declarado como puerto, pero si como señal interna, lo cual impide su apropiada simulación.

Es necesario por tanto, realizar un chequeo o inspección manual a los puertos de entrada y salida, y de ser necesario editar apropiadamente el archivo, antes de proceder a construir la componente PSpice A/D a simular.

TRABAJO FUTURO

Como trabajo futuro, se propone:

- El análisis y la implementación de técnicas avanzadas de control, programadas en VHDL para manipular el comportamiento dinámico de circuitos convertidores de potencia.

- Explorar el uso de otras opciones tecnológicas como herramientas de simulación mixta, como PSIM®.

- Validar experimentalmente las predicciones numéricas propuestas por la metodología de diseño presentada.

- Acondicionar los diseños VHDL para ejecución en plataformas de ejecución hardware con mayores recursos físicos y mejores prestaciones.

- Ampliar la metodología propuesta a otro tipo de aplicaciones, como bien pueden ser todas aquellas que impliquen el procesamiento digital de señales o los sistemas de telecomunicación.

REFERENCIAS BIBLIOGRÁFICAS

[1]. R. Zurawski. "*Embedded Systems Handbook: Industrial Information Technology* – Vol 1". CRC press. 2nd Edition. 2009.

[2]. Fernando Pardo, José A Boluda - *VHDL Lenguaje para Síntesis y Modelado de Circuitos*, Alfaomega, 2000, pag 239, Mexico D.F

[3]. T. J. Koo. "*Embedded Hydrid Systems*". Workshop on Hybrid and Embedded Systems: Technologies and Applications. The Chinese University of Hong-Kong. Febrero. 2006.

[4]. S. Edwards, L. Lavagno, E. A. Lee and A. Sangiovanni. "*Design of Embedded Systems: Formal Models, Validation, and Synthesis*". Proceedings of the IEEE. Vol 85. Nº 3. pp 366-390. Marzo. 1997.

[5]. A. Jantsch and I. Sander. "*Models of Computation and Languages for Embedded System Design*". IEE Proceedings – Computers and Digital Techniques. Vol 152. Nº 2. pp 114-129. Marzo. 2005.

[6]. G. Karsai, S. Neema and D. Sharp. "*Model-driven Architecture for Embedded Software: a Synopsis and an Example*". Science of Computer Programming. Vol 73. Nº 1. pp 26-38. Septiembre. 2008.

[7]. Sreeram Rajagopalan – *Mixed Level and Mixed Signal Simulation using PSpice A/D* and VHDL, 2005

[8]. Ken Kundert, Henry Chang - *Top-Down Design and Verification of Mixed-Signal Circuits*, 2005

[9]. Saeid Moslehpour, Chandrasekhar Puliroju, Christopher L Spivey - *Simulating VHDL in PSpice Software*, 2008

[10]. Saeid Moslehpour, Chandrasekhar Puliroju , Akram Abu-aisheh - *Design of RISC Processor Using VHDL and Cadence*, 2010

[11]. Nagel, L. W, and Pederson, D. O. – *SPICE (Simulation Program with Integrated Circuit Emphasis)*, Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973

[12]. Xilinx Inc. *ISim User Guide* (en línea) <http://www.xilinx.com/> [consultado en enero de 2014]

[13]. Basil M. Al-Hadithi, Juan Suardíaz Muro – *Nuevas Tendencias en el Diseño Electrónico Digital: Codiseño Hardware/Software,* Tecnologí@ y Desarrollo *(Revista de Ciencia, Tecnología y Medio Ambiente)* Volumen II, año 2004.

[14]. David G. Maxinez, Jessica Alcalá – *VHDL El Arte de Programar Sistemas Digitales,* Compañía Editorial Continental, Primera Edición México, 2002.

[15]. Carlos Augusto Fajardo Ariza – *Apropiación Tecnológica del Diseño de Embededd Systems Implementados sobre FPGAs y CPLDs.* Bucaramanga, 2010, 78 p, Trabajo de grado (Magister en Ingeniería Electrónica). Universidad Industrial de Santander.

[16]. Accellera, *Verilog-AMS Language Reference Manual* (en línea) <http://www.designers-guide.org/> [consultado en marzo de 2014]

[17]. Stephen Brown, Zvonko Vranesic - *Fundamentos De Lógica Digital Con Diseño Vhdl*, McGraw-Hill/Interamericana Editores, Segunda Edición, 2000

[18]. Kenneth S. Kundert – *The Designer's Guide to SPICE and Spectre*, Kluwer Academic Publishers, 1995

[19]. Andrei Vladimirescu – *The Spice Book, John Wiley & Sons*, Inc. 1994, Canada

[20]. Powersim Inc. *PSIM® User's Guide* (en línea) <http://powersimtech.com/> [consultado en junio de 2014]

[21]. Powersim Inc. *ModCoupler-VHDL User's Guide* (en línea) <http://powersimtech.com/> [consultado en junio de 2014]

[22]. Labcenter Electronics Ltd. *PROTEUS V8.2* (en línea) <http://labcenter.com/> [consultado en junio de 2014]

[23]. DesignSoft Inc. *TINA The Complete Electronics Lab for Windows, User's Manual,* (en línea) <http://www.designsoftware.com> [consultado en junio de 2014]

[24]. Richard Goering, *Panelists: Bridging the Gap Between Analog and Digital Design* (en línea) <http://www.cadence.com/> [consultado en mayo de 2014]

[25]. Murari, B., "*Bridging the gap between the digital and real worlds: the expanding role of analog interface technologies*", Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International , vol., no., pp.30,35 vol.1, 13-13 Feb. 2003

[26]. Joshua Cantrell, *Writing Simple Spice Netlists* (en línea) <http://jjc.hydrus.net/> [consultado en julio de 2014]

[27]. E. del Toro, Santiago Sánchez-Solano, Alejandro Cabrera, Angel Barriga, *Metodología para el diseño multilenguaje de sistemas de control empotrados sobre FPGAs*, 2010

[28]. Martinez Q. Juan Carlos – Andrade R. Jaime Eduardo, *Implementación De Controladores En Sistemas Realimentados Usando Electrónica Embebida y Simulación Hardware In the Loop*. Universidad Tecnológica de Pereira, 2013.

[29]. Jácome Minorta, D. F., & Navarro Minorta, L. G. (2013). *Control conmutado basado en FPGA para un convertidor DC-DC.* Trabajo de grado (Ingeniero Electrónico). Universidad Industrial de Santander.

[30]. Cadence Design Systems, *PSpice ® User's Guide* (en línea) <http://www.cadence.com/> [consultado en febrero de 2014]

[31]. Rajagopalan, S. (2005). *A New Design Methodology for Mixed Level and Mixed Signal Simulation using* PSpice A/D and VHDL. Master Thesis (Computer Engineering). Rochester Institute of Technology.

[32]. Katsuhiko Ogata – *Ingeniería de control moderna*, tercera edición, PRENTICE-HALL HISPANOAMERICANA, 1998, pag 215, Mexico D.F

BIBLIOGRÁFIA

[5]. A. Jantsch and I. Sander. "*Models of Computation and Languages for Embedded System Design*". IEE Proceedings – Computers and Digital Techniques. Vol 152. Nº 2. pp 114-129. Marzo. 2005.

[16]. Accellera, *Verilog-AMS Language Reference Manual* (en línea) <http://www.designers-guide.org/> [consultado en marzo de 2014]

[19]. Andrei Vladimirescu – *The Spice Book, John Wiley & Sons*, Inc. 1994, Canada

[13]. Basil M. Al-Hadithi, Juan Suardíaz Muro – *Nuevas Tendencias en el Diseño Electrónico Digital: Codiseño Hardware/Software,* Tecnologí@ y Desarrollo *(Revista de Ciencia, Tecnología y Medio Ambiente)* Volumen II, año 2004.

[30]. Cadence Design Systems, *PSpice ® User's Guide* (en línea) <http://www.cadence.com/> [consultado en febrero de 2014]

[15]. Carlos Augusto Fajardo Ariza – *Apropiación Tecnológica del Diseño de Embededd Systems Implementados sobre FPGAs y CPLDs.* Bucaramanga, 2010, 78 p, Trabajo de grado (Magister en Ingeniería Electrónica). Universidad Industrial de Santander.

[14]. David G. Maxinez, Jessica Alcalá – *VHDL El Arte de Programar Sistemas Digitales,* Compañía Editorial Continental, Primera Edición México, 2002.

[23]. DesignSoft Inc. *TINA The Complete Electronics Lab for Windows, User's Manual,* (en línea) <http://www.designsoftware.com> [consultado en junio de 2014]

[27]. E. del Toro, Santiago Sánchez-Solano, Alejandro Cabrera, Angel Barriga, *Metodología para el diseño multilenguaje de sistemas de control empotrados sobre FPGAs*, 2010

[2]. Fernando Pardo, José A Boluda - *VHDL Lenguaje para Síntesis y Modelado de Circuitos*, Alfaomega, 2000, pag 239, Mexico D.F

[6]. G. Karsai, S. Neema and D. Sharp. "*Model-driven Architecture for Embedded Software: a Synopsis and an Example*". Science of Computer Programming. Vol 73. Nº 1. pp 26-38. Septiembre. 2008.

[29]. Jácome Minorta, D. F., & Navarro Minorta, L. G. (2013). *Control conmutado basado en FPGA para un convertidor DC-DC.* Trabajo de grado (Ingeniero Electrónico). Universidad Industrial de Santander.

[26]. Joshua Cantrell, *Writing Simple Spice Netlists* (en línea) <http://jjc.hydrus.net/> [consultado en julio de 2014]

[32]. Katsuhiko Ogata – *Ingeniería de control moderna*, tercera edición, PRENTICE-HALL HISPANOAMERICANA, 1998, pag 215, Mexico D.F

[8]. Ken Kundert, Henry Chang - *Top-Down Design and Verification of Mixed-Signal Circuits*, 2005

[18]. Kenneth S. Kundert – *The Designer's Guide to SPICE and Spectre*, Kluwer Academic Publishers, 1995

[22]. Labcenter Electronics Ltd. *PROTEUS V8.2* (en línea) <http://labcenter.com/> [consultado en junio de 2014]

[28]. Martinez Q. Juan Carlos – Andrade R. Jaime Eduardo, *Implementación De Controladores En Sistemas Realimentados Usando Electrónica Embebida y Simulación Hardware In the Loop*. Universidad Tecnológica de Pereira, 2013.

[25]. Murari, B., "*Bridging the gap between the digital and real worlds: the expanding role of analog interface technologies*", Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International , vol., no., pp.30,35 vol.1, 13-13 Feb. 2003

[11]. Nagel, L. W, and Pederson, D. O. – *SPICE (Simulation Program with Integrated Circuit Emphasis)*, Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973

[20]. Powersim Inc. *PSIM® User's Guide* (en línea) <http://powersimtech.com/> [consultado en junio de 2014]

[21]. Powersim Inc. *ModCoupler-VHDL User's Guide* (en línea) <http://powersimtech.com/> [consultado en junio de 2014]

[1]. R. Zurawski. "*Embedded Systems Handbook: Industrial Information Technology* – Vol 1". CRC press. 2nd Edition. 2009.

[31]. Rajagopalan, S. (2005). *A New Design Methodology for Mixed Level and Mixed Signal Simulation using* PSpice A/D and VHDL. Master Thesis (Computer Engineering). Rochester Institute of Technology.

[24]. Richard Goering, *Panelists: Bridging the Gap Between Analog and Digital Design* (en línea) <http://www.cadence.com/> [consultado en mayo de 2014]

[4]. S. Edwards, L. Lavagno, E. A. Lee and A. Sangiovanni. "*Design of Embedded Systems: Formal Models, Validation, and Synthesis*". Proceedings of the IEEE. Vol 85. Nº 3. pp 366-390. Marzo. 1997.

[9]. Saeid Moslehpour, Chandrasekhar Puliroju, Christopher L Spivey - *Simulating VHDL in PSpice Software*, 2008

[10]. Saeid Moslehpour, Chandrasekhar Puliroju , Akram Abu-aisheh - *Design of RISC Processor Using VHDL and Cadence*, 2010

[7]. Sreeram Rajagopalan – *Mixed Level and Mixed Signal Simulation using PSpice A/D* and VHDL, 2005

[17]. Stephen Brown, Zvonko Vranesic - *Fundamentos De Lógica Digital Con Diseño Vhdl*, McGraw-Hill/Interamericana Editores, Segunda Edición, 2000

[3]. T. J. Koo. "*Embedded Hydrid Systems*". Workshop on Hybrid and Embedded Systems: Technologies and Applications. The Chinese University of Hong-Kong. Febrero. 2006.

[12]. Xilinx Inc. *ISim User Guide* (en línea) <http://www.xilinx.com/> [consultado en enero de 2014]

ANEXO A

A.1. METODOLOGÍA DE DISEÑO PROPUESTA PARA SIMULAR VHDL EN PSPICE A/D

En los capítulos 2 y 3 se presentaron los antecedentes necesarios para entender la diferencia entre PSpice A/D, el lenguaje descriptivo hardware VHDL y los diferentes tipos de simulación; además de la escasez de estas herramientas de simulación a la hora de comprobar diseños de señal mixta. Como se mencionó específicamente en la sección 3.3 se empleó la metodología propuesta en [7].

El flujo de diseño típico para esta metodología es el siguiente:

1. La lógica digital en VHDL (metodología de diseño top down) es verificada usando simuladores digitales como *Xilinx ISE*.
2. El RTL del código VHDL es sintetizado en *Synplify Pro* usando como dispositivo final *Lattice MACH* 111, como resultado obtenemos un netlist en VHDL a nivel de compuertas que contiene específicamente el dispositivo.
3. El netlist a nivel de compuertas se convierte ahora en un archivo de circuito PSpice A/D utilizando el software de interfaz (*VHDL2PSpice*) mencionado en la sección 3.3.
4. El archivo de circuito se convierte en un símbolo *OrCAD* que se puede colocar en un editor de esquemas junto con componentes analógicos y de este modo se pueden realizar simulaciones de señal mixta.

A.2. CONVERSIÓN DE VHDL EN UN ARCHIVO CIRCUITO PSPICE

Esta sección cubrirá cómo convertir con éxito un modelo VHDL a un componente de trabajo en PSpice A/D. Partiremos como ejemplo de un código VHDL un contador de 8 bits como muestra la Figura A.1, además supondremos que el código VHDL ya fue verificado y simulado en un programa como *Xilinx ISE*.

Figura A.1. Código VHDL de un contador de 8 bits

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity cont_8bits is
7       Port ( clk  : in std_logic;
8              rst  : in std_logic;
9              data : out std_logic_vector(7 downto 0));
10  end cont_8bits;
11
12  architecture Behavioral of cont_8bits is
13    signal counter : std_logic_vector(7 downto 0);
14
15  begin
16    count_it : process(clk,rst)
17    begin
18      if (rst = '1') then
19          -- reset actions
20          counter <= "00000000";
21      else
22          if (rising_edge(clk)) then
23          -- main process
24          counter <= counter+1;
25          else
26          counter <= counter;
27        end if;
28      end if;
29    end process;
30    data <= counter;
31
32  end Behavioral;
```

A.2.1. Synplify Pro

El segundo paso es sintetizar el RTL del código VHDL, usando como herramienta el programa *Synplify Pro*, una vez en el programa tenemos el entorno de trabajo que muestra la Figura A.2, damos clic en *File/New* y empezamos un nuevo proyecto. En la pantalla se muestra una ventana emergente como se muestra en la Figura A.3, en esta damos clic en *Project File* y asignamos un nombre al proyecto y la ubicación deseada por el usuario.

Figura A.2. Perfil en Synplify Pro

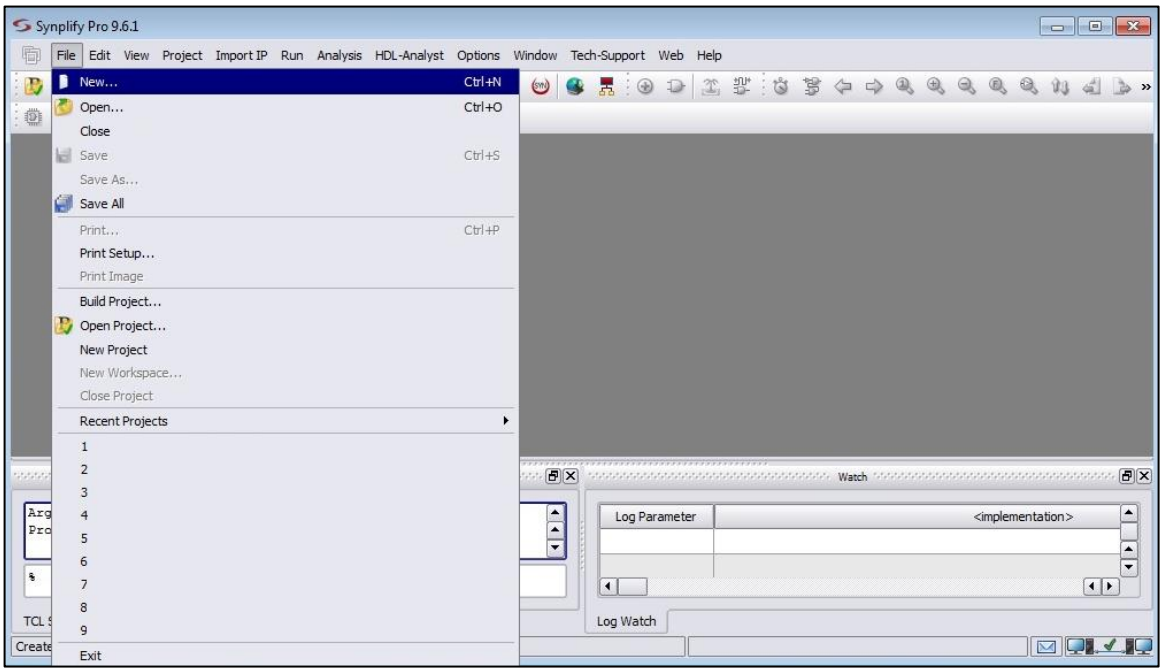
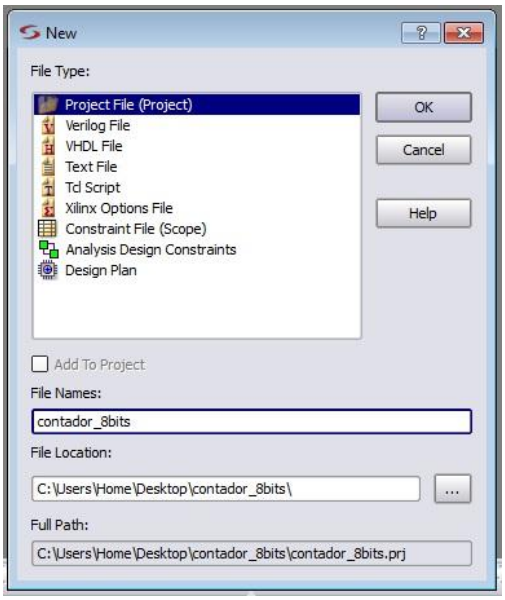
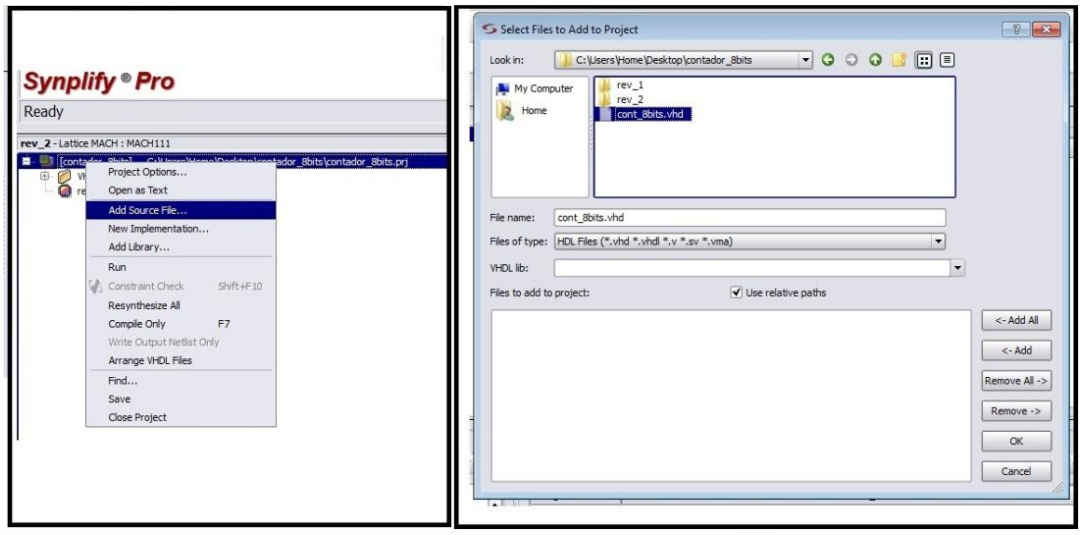


Figura A.3. Nuevo proyecto en Synplify Pro



A continuación cargamos el archivo fuente. En la pantalla de interfaz del proyecto, damos clic derecho sobre el proyecto como muestra la parte (a) de la Figura A.4, nuevamente aparece una ventana emergente en la que seleccionamos el archivo fuente (*.vhd) a añadir tal como muestra la parte (b) Figura A.4.

Figura A.4. Cargando archivo fuente en Synplify Pro

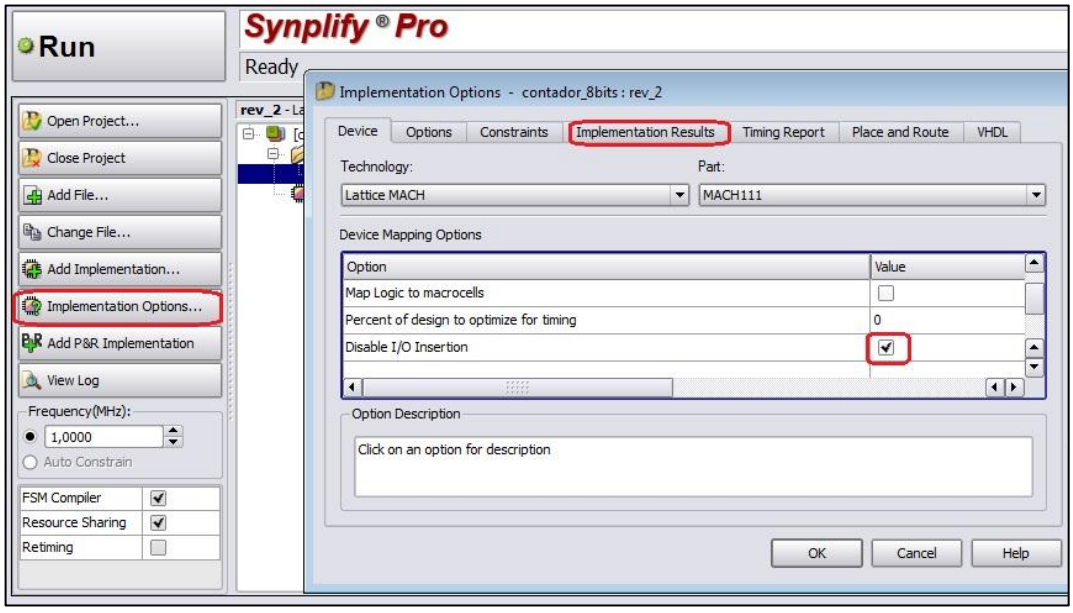


(a) (b)

Una vez el archivo VHDL es añadido, el usuario puede visualizarlo haciendo clic en la carpeta *VHDL* en la ventana de interfaz del proyecto y luego damos doble clic sobre el archivo fuente. El archivo es abierto en una ventana con líneas numeradas como se muestra en la Figura A.1, además el programa me permite realizar modificaciones sobre el mismo y guardar estos cambios sobre el archivo original.

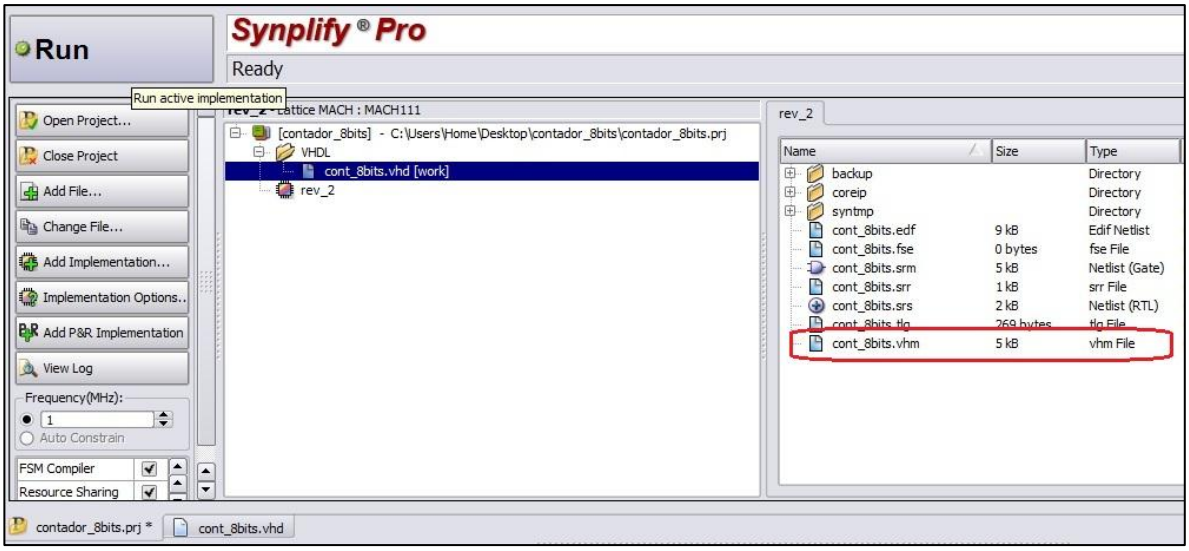
El proyecto está listo para sintetizarse, sin embargo antes de realizar este paso, el usuario debe establecer ciertas condiciones para que la síntesis funcione correctamente. Hacemos clic en el botón *Implementation Options* ubicado en la columna izquierda del entorno de trabajo como se puede observar en la Figura A.5, a continuación aparecerá una ventana emergente con el nombre de *Implementation Options* y varias pestañas de opciones.

Figura A.5. Ajustes Synplify Pro



En la pestaña *Device* nos aseguramos de que la opción *Technology* se establezca en Lattice MACH, y la opción *Part* en MACH111. En la subventana *Device Mapping Options* damos clic en el cuadro de la opción "*Disable I/O Insertion*" para marcarlo. A continuación vamos a la pestaña de *Implementation Results* y hacemos clic en la casilla *Write Mapped VHDL Netlist.* Después de realizado esto, el programa está listo para ser sintetizado. Damos clic en el botón *RUN* ubicado en la columna izquierda del menú del proyecto, en la ventana de la derecha, se generaran varios archivos como muestra la Figura A.6.

Figura A.6. Procedimientos Synplify Pro



Estamos interesados en el archivo VHDL a nivel de compuertas (RTL). Este es el archivo con la extensión VHM ubicado en la lista de la ventana derecha. Para ver su contenido damos doble clic en el archivo VHM. Una vez allí, el archivo debe ser guardado en C:\ProgramFiles\Vhdl2PSpice\Source. Guardado el archivo, el

usuario ya puede salir del programa Synplify Pro porque ya no es necesario para la conversión.

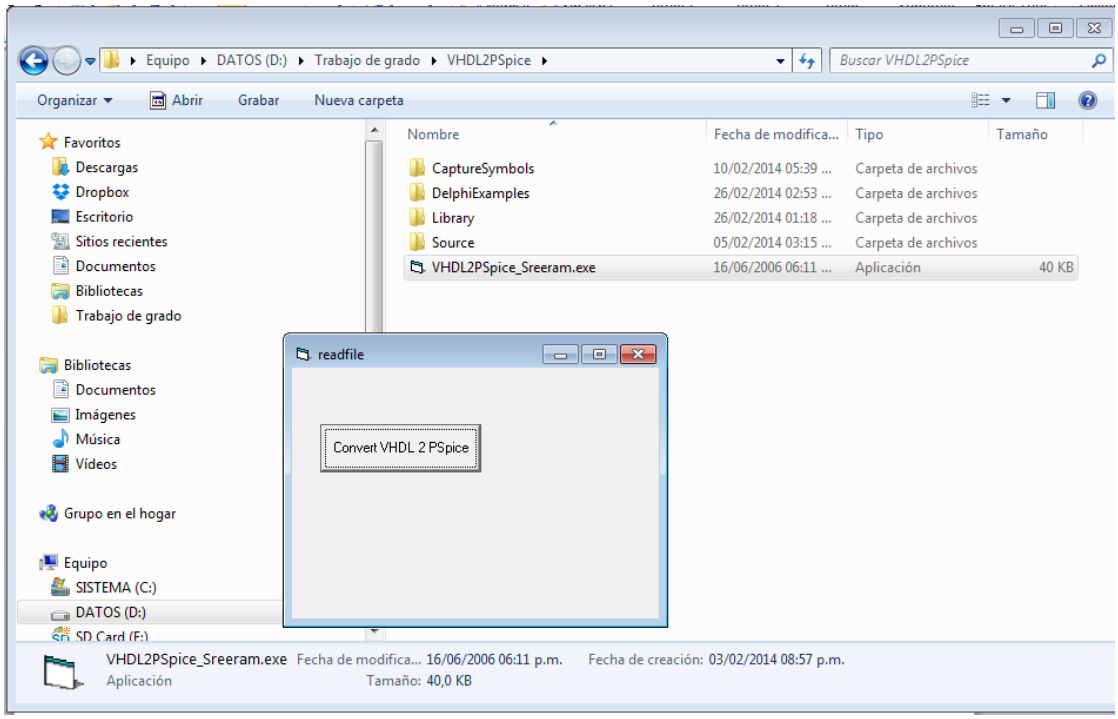
A.2.2. VHDL2PSpice

Como tercer paso utilizaremos la interfaz VHDL2PSpice desarrollada por [7] para convertir un modelo de síntesis VHDL (RTL) a un componente de trabajo en PSPICE, para tener éxito en el proceso. En primer lugar el usuario tendrá que extraer el programa en el directorio *C:\Program Files*. Esta parte es importante debido a que el programa no funcionará a menos que se ubique dentro de la carpeta llamada *VHDL2PSPICE* en el directorio especificado.

Todos los archivos necesarios se mostrarán en el contenido de las carpetas. Las carpetas creadas son *CaptureSymbols*, *DelphiExamples*, *Library*, y *Source*. Estas carpetas son necesarias para que el programa guarde apropiadamente los símbolos de Capture y las librerías.

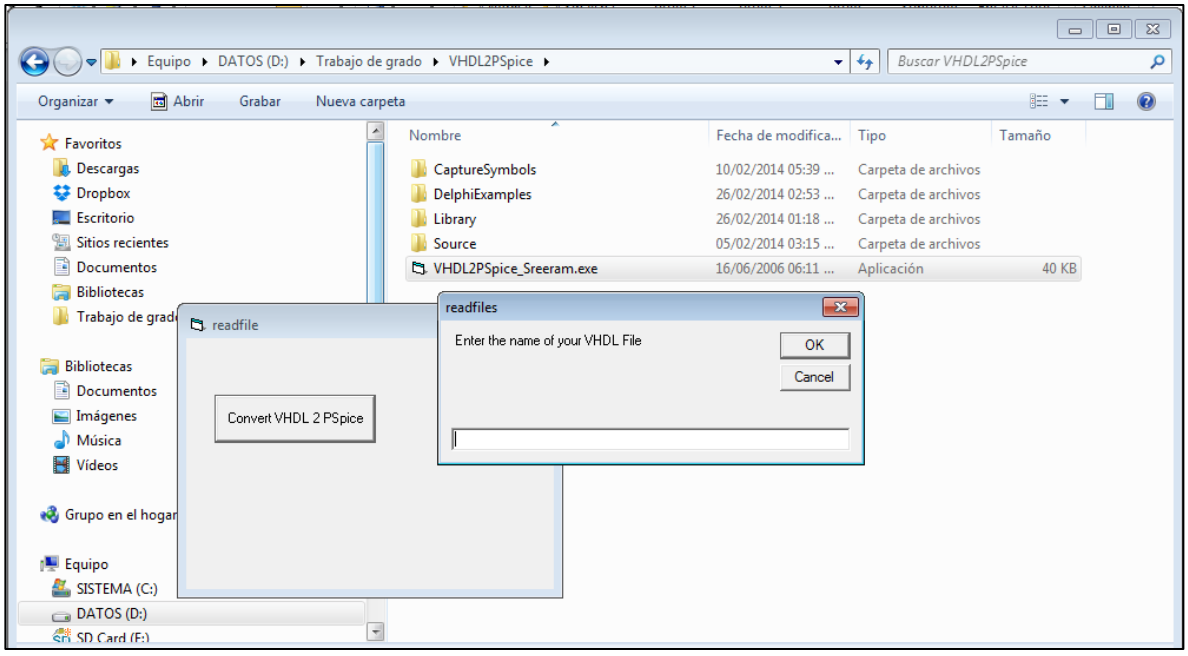
Ahora en C:\ProgramFiles\Vhdl2PSpice, ejecutamos la aplicación Vhdl2PSpice.exe. Una ventana emergente aparece en la pantalla con título *readfile* y con un botón llamado "Convertir VHDL 2 PSpice", como muestra la Figura A.7.

Figura A.7. Directorio raíz de la interfaz VHDL2PSpice



Seleccionamos la opción del botón "Convertir VHDL 2 PSpice" y una nueva ventana emergente aparecerá pidiendo introducir el nombre del archivo VHDL que se guardó en el directorio C:\ProgramFiles\Vhdl2PSpice\Source como muestra la Figura A.8. No hay necesidad de escribir la extensión (*.vhm), sólo el nombre del archivo. Después damos clic en Aceptar, el usuario debe ahora introducir el nombre de la entidad de nivel superior (en este ejemplo, el dispositivo se llama cont_8bits). Hacemos clic en *Aceptar* y de nuevo aparecerá una ventana emergente de éxito.

Figura A.8. Cargando archivo de la interfaz VHDL2PSpice



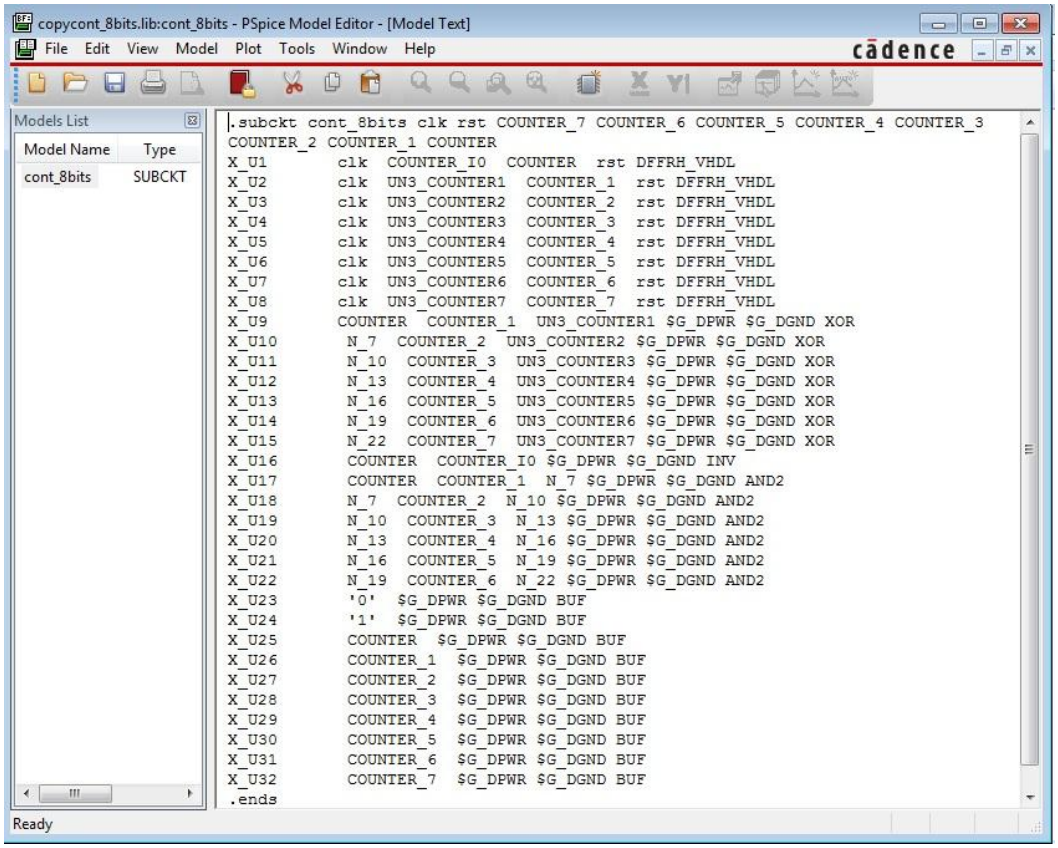
El programa ha creado ahora archivos de salida que pueden ser utilizados en PSpice A/D. En la carpeta *Library* podemos ver los archivos convertidos y almacenados por la aplicación. El archivo que el usuario debe buscar será con el mismo nombre, sin embargo con la palabra "copy" al inicio del nombre del archivo. La extensión del archivo es .LIB. También hay un archivo llamado LIBPOINTER.LIB, que es un archivo de biblioteca que dice PSpice A/D dónde se deben buscar las piezas creadas.

A.2.3. PSpice Model Editor

Como último paso antes de poder realizar una simulación donde se involucren tanto componentes análogos como digitales, el usuario debe abrir el programa *PSpice Model Editor* y abrir el archivo *copycont_8bits.lib* creado en el paso

anterior. Este debe encontrarse en el directorio C:\ProgramFiles\Vhdl2PSpice\Library. En esta librería encontraremos la representación estructural del código VHDL que fue convertido (RTL) por PSpice A/D, que consiste en un subcircuito de componentes digitales primarios (compuertas, flip-flop, inversores, buffer), tal como muestra la Figura A.9.

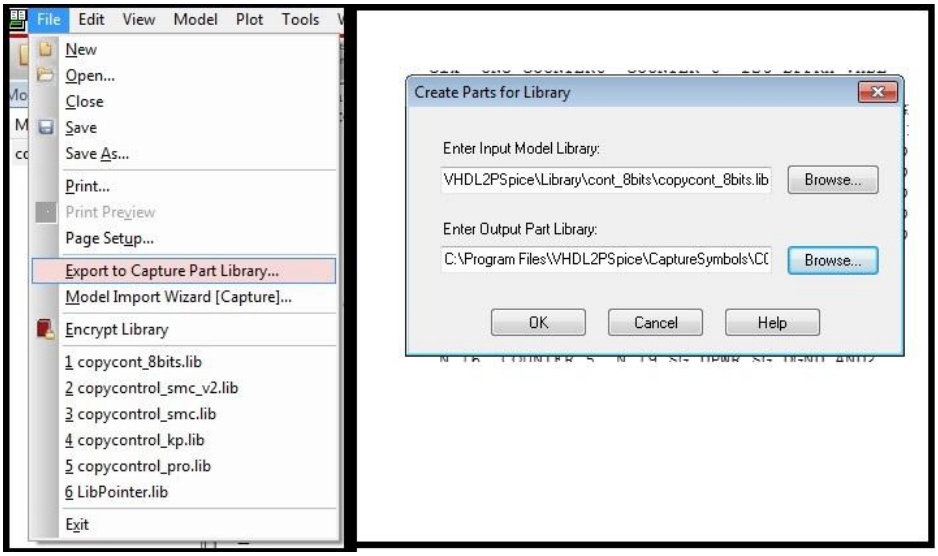
Figura A.9. Código VHDL convertido a nivel de compuertas en PSpice A/D



Para finalizar este paso, dentro del programa *PSpice Model Editor* damos clic en *File*/*Export to Capture Part Library*, como muestra la Figura A.10. En este punto

del procedimiento el archivo de la librería está creado, sin embargo, el programa no sabrá cómo mostrar esta parte salvo que se exporte correctamente.

Figura A.10. Exportación de parte a biblioteca



Debemos especificar C:\ProgramFiles\Vhdl2PSpice\CaptureSymbols como directorio a donde se exportará la parte, porque el archivo Libpointer.lib que está en la carpeta *Library* dirá a PSpice donde se guardan estos archivos. Cuando se abra el programa *Capture*, el usuario puede ser capaz de colocar el símbolo en la página de cualquier esquemático como se muestra en la Figura A.11 y trabajar con el simulador PSpice A/D bajo entornos de señal mixta para así poder verificar algoritmos de control programados en VHDL que actúan sobre circuitos convertidores de potencia.

Figura A.11. Parte en página de esquemático Capture