UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INGENIERÍA

Departamento de Electrotecnia

Modulación Vectorial de Inversores de Potencia

Mauricio A. Tonelli

Tesis presentada para obtener el grado de
MAGISTER EN INGENIERÍA

Director: Dra. María Inés Valla
Codirector: Ing. Pedro Eduardo Battaiotto

La Plata, febrero de 2004

AGRADECIMIENTO

Agradezco a María Inés por el gran esfuerzo y dedicación en dirigirme en esta tesis, y a Pedro por las largas discusiones y buenos consejos respecto de la implementación.

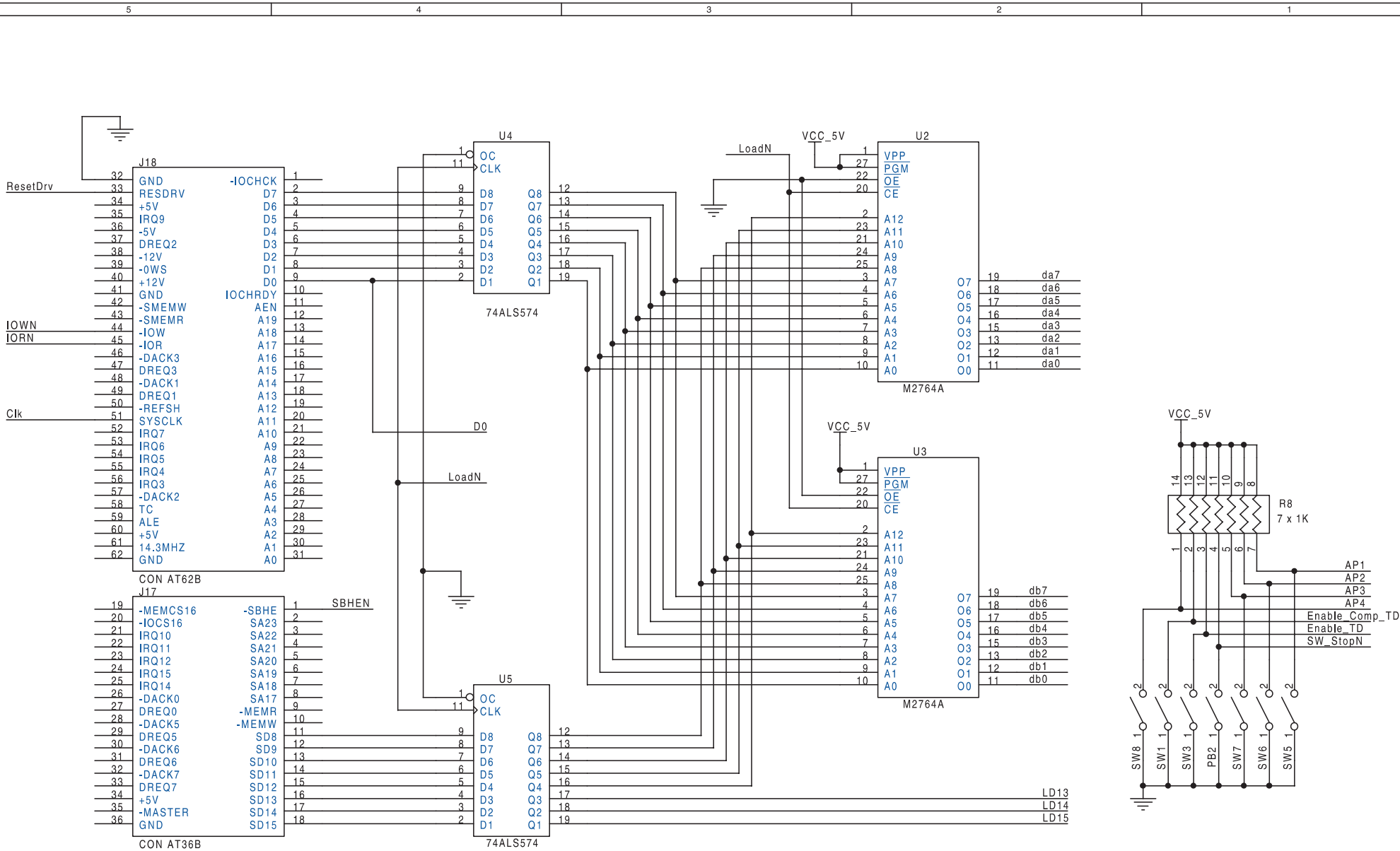
También quiero agradecer la colaboración desinteresada de mis compañeros y amigos del LEICI, en especial la de Miguel, Graciela, Sergio, Marcelo, Fernando, Enrique y Paul.

El trabajo realizado en esta tesis fue posible gracias a la beca otorgada por el Fondo para el Mejoramiento de la Calidad de la Educación Superior (FOMEC); la Universidad Nacional de La Plata (UNLP) y el Laboratorio de Electrónica Industrial, Control e Instrumentación (LEICI-UNLP).

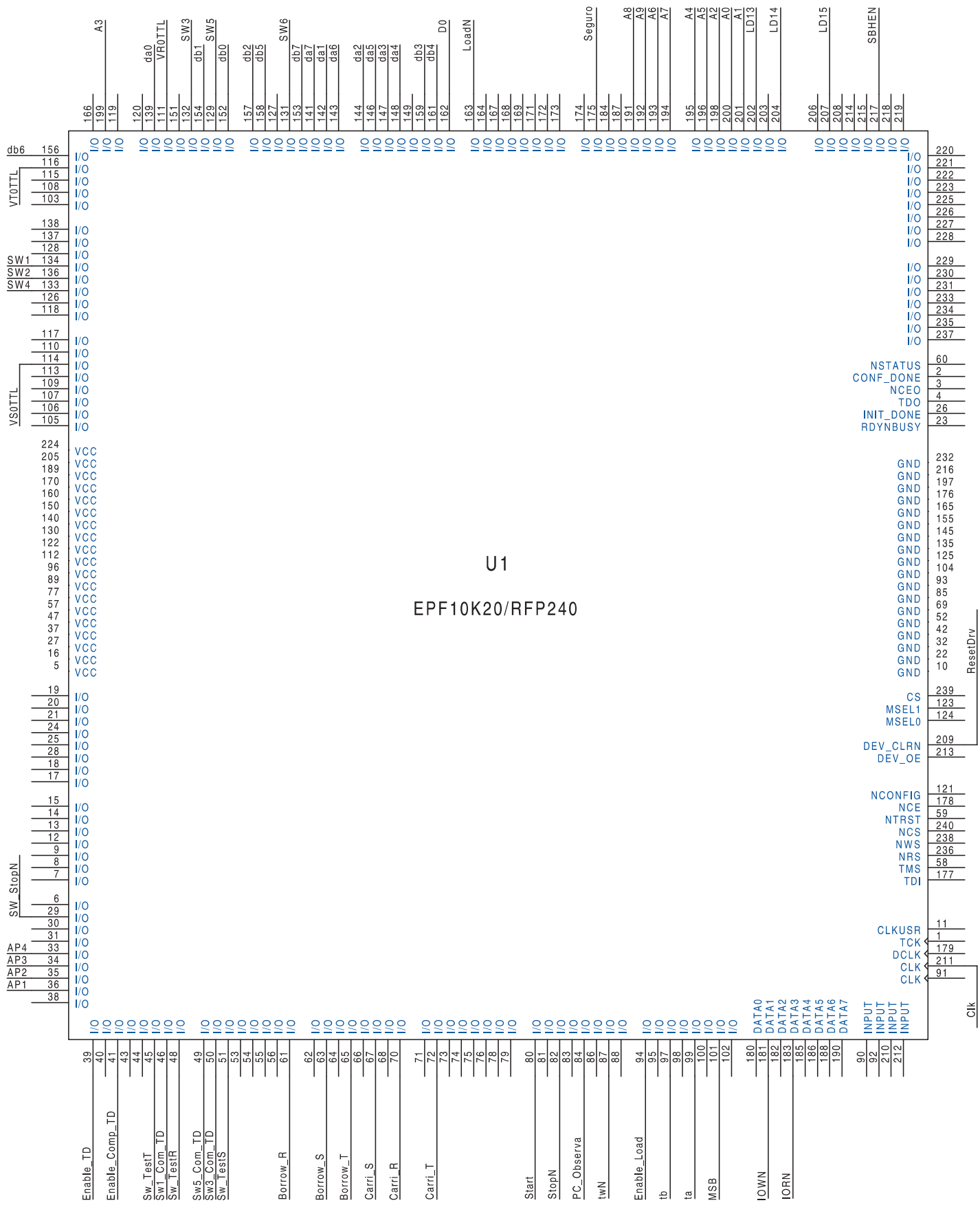
A mi esposa, María,

a mis Padres, Lía y Luis,

... y a Mateo.

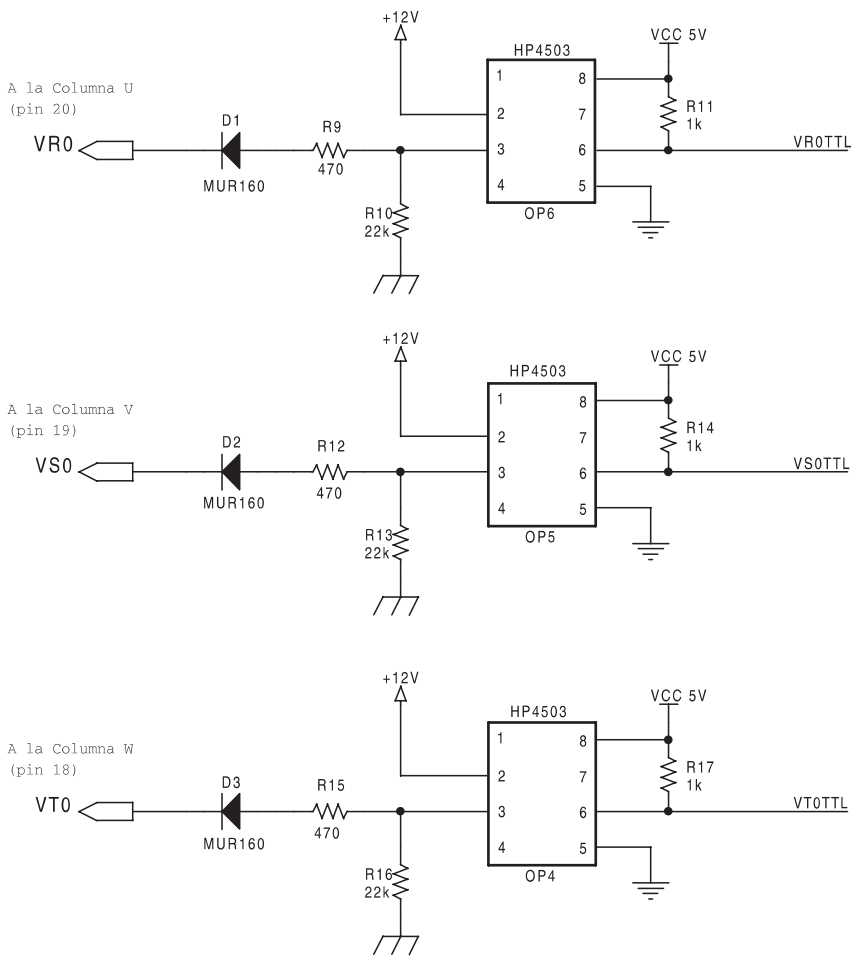


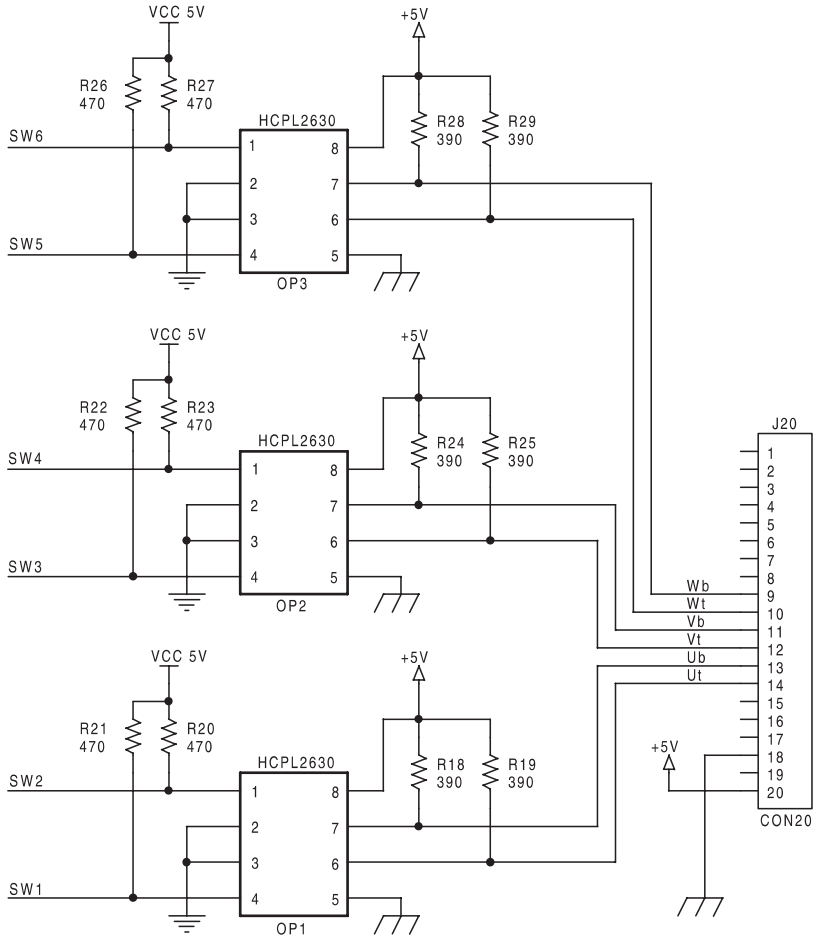
Title: Modulador Vectorial		
Size: A4	Document Number: ISA, Latch y Memorias	Rev: 1.0
Date: Sunday, September 28, 2003	Sheet 1 of 3	



Title: Modulador Vectorial		
Size	Document Number: FPLD	Rev: 1.0
Date: Sunday, September 28, 2003	Sheet 2 of 3	

Etapa de medición de Tensión de Columna





Title: Modulador Vectorial		
Size: A4	Document Number: Interfaces	Rev: 1.0
Date: Sunday, September 28, 2003	Sheet 3 of 3	

RESUMEN

Esta Tesis está dedicada al desarrollo de un modulador de ancho de pulso (PWM) para el control de inversores de potencia trifásicos. Los moduladores de PWM se utilizan ampliamente en la industria, para el control de motores y otras aplicaciones trifásicas, en donde la amplitud y frecuencia deben ser controladas. La elección e implementación de un método de PWM afecta la eficiencia del Inversor de potencia y la calidad de la forma de onda de la tensión de salida. En ese trabajo se pone especial énfasis en la modulación del vector espacial y su implementación con dispositivos de lógica programable, de modo de lograr una implementación independiente del control de la aplicación y con suficiente flexibilidad como para ser utilizado por distintas aplicaciones.

El modulador vectorial universal propuesto en esta Tesis es capaz de trabajar con transición continua, desde una amplitud nula en la zona lineal de modulación hasta la onda cuadrada con pleno aprovechamiento de la tensión continua disponible. En él se realiza una compensación del tiempo muerto de excitación de las columnas, sin utilizar sensores de corriente. El modulador recibe las señales de referencia a través del bus ISA de una PC y genera como salidas las señales de excitación de los interruptores del inversor. El análisis desarrollado se complementa con resultados de simulación en lo referente a la implementación digital, y con resultados experimentales sobre un inversor de pequeña potencia.

ABSTRACT

The present Thesis is devoted to the development of a Pulse Width Modulator (PWM) suitable for control of three phase Power Inverters. The PWM is widely used in industry in motor drives, line conditioning and other three phase applications in which the amplitude and frequency should be controlled. The selection and implementation of the PWM directly affects the inverter efficiency and the quality and harmonics contents of the output voltage. Special emphasis is given to the Space Vector Modulation (SVM) and its implementation with Field Programmable Logic Devices (FPLD), in order to obtain a PWM independent of the applications control. Moreover it presents sufficient flexibility to adapt itself to different applications and different inverters.

The universal SVM proposed in this Thesis, is capable to perform continuos transition from near zero amplitude in linear mode to six-step operation with full utilization of the available DC Bus. The modulator also compensates the effects of blanking time in the leg excitation, without current sensors. The SVM receives the reference signals from a PC through the ISA Bus, and generates at its output the driving signals for the inverter power switches. The theoretical analysis is complemented with simulation results of the digital implementation in the FPLD. Finally it is evaluated with a low power inverter.

CONTENIDOS

LISTA DE FIGURAS Y TABLAS

FIGURAS

TABLAS

LISTA DE SIMBOLOS

AHDL: *Altera Hardware Description Language*, Lenguaje de software utilizado para realizar la entrada de los diseños. Este soporta ecuaciones Booleanas, maquinas de estado, estados condicionales, etc.

ASIC: *Application Specific Integrated Circuit*, circuito integrado de aplicaciones especificas.

CA: Corriente Alterna.

CC: Corriente Continua.

CPLD: *Complex Programable Logic Device*, es un dispositivo lógico compuesto por un arreglo de celdas lógicas en un marco de interconexión.

Design entry: Entradas de diseño.

DTC: Transformadas discretas de coseno.

EAB: *Embedded Array Block*, Bloque integrado de arreglos.

Fan-in: el número de señales de entrada que alimentan todas las entradas de una celda lógica.

Fan-out: el número de señales de salida que puede manejar la salida de una celda lógica

FastTrack: canal metálico continuo: canal de conexión dedicado que alcanza todo el ancho y largo de los dispositivos de la familia FLEX. Este le permite a las señales viajar entre todos los LABs del dispositivo.

FFT: Transformadas rápidas de Fourier.

FIR: *Finite Impulse Responce*, respuesta finita al impulso.

Fitting: es el proceso de transformar un nivel de compuertas o una representación en suma de productos de un circuito, a un archivo para que sea utilizado en la programación de un PLD o CPLD. Este proceso generalmente ocurre después de la etapa de síntesis.

FLEX: *Flexible Logic Element MatriX*, es una familia de productos de Altera, considerado como CPLD.

FPGA: *Field Progammable Gate Array*, es un dispositivo lógico compuesto por un arreglo regular de celdas lógicas dentro de un marco de conexiones y canales de señales.

FPLD: *Field Progammable Logic Device*, dispositivo lógico programable en campo: es un circuito integrado usado para implementar hardware digital, permitiéndole al usuario final configurar el chip para diferentes diseños.

HDL: *Hardware Description Language*, Software para descripción de Hardware.

I/O: *Input/Output*, (Entradas/Salidas)

IOE: *I/O Element*, Elemento de entrada y salida

LAB: *Logic Array Block*, Bloque de arreglos lógicos: es el bloque básico de la familia MAX. Cada LAB contiene al menos una macrocelda, un bloque de I/O y un arreglo de expander product term.

LC: *Logic Cell*, Celdas lógica: puede ser una compuerta, un flip-flop, o alguna otra estructura.

LCA: *Logic Cell Array*, arreglo de celdas lógicas

LE: *Logic Element*, Elemento lógico: es el bloque básico de la familia FLEX. Consiste de una LUT de 4 entradas y un flip-flop programable.

Logic Array: arreglo lógico

LUT: *Look-Up Tables*, Tablas de locaciones de memoria

MAX: *Multiple Array Matrix*, es una familia de productos de Altera, considerado como CPLD.

MC: *Macrocell*, Macro celdas: en FPGAs es el bloque indivisible más pequeño, en dispositivos de la familia MAX consiste de dos partes: lógica combinatoria y un registro configurable.

MPLD: *Mask Progammed Logic Device*, dispositivo de lógica programable con máscara: es un FPLD al cual se le programo el diseño del usuario final.

Netlist: es un archivo de texto que describe un diseño.

PAL: *Programmable Array Logic*, es un pequeño PLD compuesto por un nivel de lógica programable; un plano AND programable y seguido por un plano OR fijo.

PIA: *Programmable Interconect Array*, Arreglo de Interconexión programable

PLA: *Programmable Logic Array*, es un pequeño PLD compuesto por dos niveles de lógica programable, uno es de tipo AND y el otro OR.

PLD: *Programmable Logic Device*, dispositivo lógico programable: ésta clase de dispositivos incluye a las PALs y PLAs.

PIP: *Programmable Interconnect Point*, Punto Programable de Interconexión.

Place and route: es el proceso de transformar un nivel de compuertas de un circuito, a un archivo para que sea utilizado en la programación de una FPGA. Este proceso requiere dos pasos: uno es asignar a una función lógica una locación física y especifica dentro del dispositivo FPGA, la otra es interconectar las funciones lógicas previamente asignadas en las locaciones físicas.

Postlayout: es un modelo, en un formato que puede ser VHDL, producido por el software de fitting o Place & Route. La descripción del modelo es el circuito implementado

dentro del dispositivo FPLD seleccionado. Este se utiliza para la simulación, en donde se verifican la funcionalidad y los timing del diseño.

PWM: *Pulse Width Modulation*, Modulación de ancho de Pulso.

Routability: ruteo: es una medida de la probabilidad de que una señal pueda conectarse satisfactoriamente desde una locación a otra dentro del dispositivo. *Routability* dentro de FPGA esta afectada por el número de segmentos conductores horizontales y verticales y por la arquitectura de la celda lógica. *Routability* dentro de CPLD refiere a la probabilidad de que un conjunto de señales lógicas pueda ser guiado a través de la interconexión programable y dentro de un bloque lógico.

SRAM: *Static RAM*

Switch Matrix: matrices de llaves.

UPS: Fuentes ininterrumpibles de tensión.

VHDL: es el acrónimo de VHSIC (*Very High Speed Integrated Circuit) Hardware Decription Language*, se utiliza para describir funciones, interconexiones y modelos. Una referencia del lenguaje está cubierta totalmente por IEEE 1076-1993

Verilog: es un software HDL usado para la descripción de sistemas digitales con propósitos de simulación y síntesis. Una referencia está en IEEE 1364-1995.

VSI: Puente inversor trifásico de tensión.

WE: *Write Enable*, Habilitación de escritura

Wiring: cableado.

1. INTRODUCCIÓN

1.1. Motivación

A lo largo del último siglo los sistemas de corriente alternada (CA) de amplitud y frecuencia fija se han mantenido como los más convenientes para generación y distribución eléctrica. Sin embargo es bien conocido que no es la forma de energía más conveniente para muchos procesos industriales y aplicaciones residenciales. Por otra parte, muchas aplicaciones requieren de una alimentación muy exacta y no soportan las imperfecciones de una línea de distribución. Todos estos hechos han motivado el desarrollo de circuitos convertidores de energía eléctrica para cumplir con los más variados requisitos. Como resultado de toda una evolución hoy se encuentran muchas aplicaciones industriales y también residenciales conectadas a la línea de alimentación a través de convertidores de potencia que mejoran la prestación, eficiencia y confiabilidad de los distintos sistemas. De todos los modernos convertidores de potencia el puente inversor trifásico de tensión (VSI) es posiblemente el circuito de uso más difundido en rangos de potencia que parten desde fracciones hasta cientos de kilowatt. Este circuito consta de 6 semiconductores de potencia controlados y seis diodos conectados en antiparalelo. Es capaz de generar tensiones trifásicas de amplitud y frecuencia variable, a partir de una tensión continua fija. La circulación de corriente es bidireccional, por lo tanto el circuito es capaz de convertir potencia continua en alterna y viceversa. Dado que el VSI genera tensiones discretas según cuales sean los estados de las llaves controladas, se requiere de algún método de promediación para generar tensiones de salida de amplitud y frecuencias correctas. El VSI controlado por modulación de ancho de pulso (PWM) reproduce una tensión de referencia conmutando el inversor a alta frecuencia.

Normalmente el inversor no está aislado sino que es parte integral de una aplicación como ser: controles de par, velocidad y posición de máquinas de CA, fuentes ininterrumpibles de tensión (UPS), filtros activos de potencia en el acondicionamiento de líneas eléctricas, por citar las más importantes. En todas estas aplicaciones el usuario demanda que el sistema total sea multifuncional, tenga muy buenas prestación, eficiencia, confiabilidad y que sean “amigables” con el usuario. Todo esto determina la instalación de una importante cantidad de inteligencia al servicio de la aplicación general. Por otra parte en casi todos estos sistemas el

algoritmo de control de la aplicación genera la tensión de referencia para el VSI, quien debe sintetizarla lo mejor posible utilizando algún método de modulación PWM. Aquí se ve que el modulador es el corazón de todo el sistema pues si el VSI no puede reproducir fiel y eficientemente la referencia que le envía el control, la planta no puede ser alimentada correctamente. La elección del método de modulación afecta fuertemente la calidad de la onda de tensión de salida y la eficiencia del convertidor, por lo cual es de fundamental importancia realizar un desarrollo adecuado del mismo.

Por lo anteriormente expresado en los sistemas de control de CA pueden distinguirse dos fases bien diferenciadas: 1) el control del inversor con un modulador de PWM y 2) el control de la aplicación. Es muy difícil que ambas funciones sean realizadas por un único procesador, especialmente cuando se requiere una alta prestación de la aplicación. Por otra parte, generalmente ambas acciones de control son independientes. Estas razones motivaron el uso de un ambiente de computadora personal (PC) para elaborar las estrategias de control de la aplicación y el desarrollo de un modulador de ancho de pulso (PWM) autónomo.

El objetivo de esta Tesis es desarrollar un modulador universal capaz de leer la referencia de tensión desde el Bus de la PC y generar las señales de comando para los interruptores del VSI. Para cumplir con el objetivo se selecciona el método de modulación más adecuado para una gran variedad de aplicaciones y se selecciona el medio ideal para su implementación. Respecto del primer aspecto se seleccionó la modulación vectorial por ser la más flexible para un amplio rango de aplicaciones y ofrecer características interesantes como ser:

- tener en cuenta la naturaleza trifásica de la carga,
- utilizar completamente la tensión de CC disponible
- optimizar el número de conmutaciones del inversor,
- no necesitar sincronización entre la señal de referencia y la portadora.

Respecto al medio de implementación, se optó por el uso de dispositivos de lógica programable en campo (FPLD) los cuales facilitan tanto la etapa de diseño como de implementación del modulador.

1.2. Reseña histórica de los métodos de PWM

El puente inversor trifásico de tensión comenzó a desarrollarse a partir de la aparición de los tiristores y adquirió una masiva difusión a partir del desarrollo de dispositivos de conmutación de potencia con encendido y apagado controlado como ser los MOSFETs, los IGBTs y los GTOs, por nombrar los principales. Paralelamente y acompañando el desarrollo de los dispositivos de potencia se fueron desarrollando distintas estrategias de modulación PWM desde mediados de la década del 60. La gran variedad de métodos propuestos en los últimos 40 años pueden agruparse en dos grandes líneas:

- Modulación por programación de los ángulos de conmutación,
- Modulación basada en una señal portadora.

1.2.1. Modulación por programación de los ángulos de conmutación

En los métodos por programación de ángulos, los patrones de conmutación son calculados previamente en forma de satisfacer un determinado índice de performance, y luego son almacenados en memorias. Durante la operación del inversor, las memorias son accedidas secuencialmente para generar las señales de comando para los dispositivos de potencia. Distintos criterios fueron adoptados para programar los ángulos. Inicialmente se propuso la eliminación de determinadas armónica como ser la 5^a, 7^a, 11^a , 13^a , etc, [Tur 63][Pat 73][Pat 74]. En 1977, Buja e Indri propusieron minimizar la distorsión de corriente como criterio de optimización de los ángulos de conmutación [Buj 77][Buj 80] y luego presentaron su posible implementación con microcontroladores [Buj 82]. Años más tarde Zach *et al* propusieron otros criterios de optimización como ser: maximizar eficiencia [Zac 85a], minimizar ripple de par del motor [Zac 85b]. A principios de la década del 90, Holtz *et al* retoman la idea de pre-programar los patrones de conmutación desarrollando métodos de seguimiento de trayectorias para mejorar la prestación dinámica de los accionamientos [Hol 92][Hol 94a][Hol 94b].

Todos estos métodos requieren de computadoras potentes para calcular los ángulos programados por lo cual se han visto limitados al cálculo de unos pocos ángulos por ciclo de fundamental. Estas razones han determinado que su uso se vea limitado hoy en día a

aplicaciones de alta potencia o alta velocidad, en situaciones en las que es imposible lograr una gran diferencia entre la frecuencia de conmutación y la fundamental deseada.

1.2.2. Modulación basada en una señal portadora

Los métodos de modulación basados en onda portadora pueden operar con frecuencia de conmutación alta y son más simples de diseñar e implementar. La idea básica es que en cada ciclo de portadora, el promedio de la tensión generada reproduzca la referencia deseada. En 1964 Schönung y Stemmler desarrollaron el PWM sinusoidal o método subarmónico [Sch 64]. En este método se compara la tensión de referencia de cada fase con una portadora triangular y los puntos de intersección determinan los instantes de conmutación de las columnas del inversor. Este método de modulación ha sido muy empleado y analizado durante la década del 70, demostrándose que mientras la relación entre portadora y fundamental fuese grande se mantenía una relación lineal entre la referencia y la tensión fundamental a la salida del inversor, y las componentes armónicas se concentran en bandas laterales alrededor de la portadora y sus múltiplos [Bow 75a][Bow 75b][Maz 81].

La desventaja de la modulación sinusoidal es la pobre utilización de la tensión continua disponible, la cual no supera el 78,5% de la tensión fundamental de una onda cuadrada (inversor trabajando en modo six-step). En 1975 Buja e Indri proponen superar este límite inyectando una tercera armónica a la referencia de cada fase [Buj 75]. Esta propuesta se basa en el hecho que la gran mayoría de las aplicaciones no tiene conectado el neutro, y por consiguiente cualquier componente homopolar no se ve reflejada en la carga sino en la tensión relativa del neutro de la carga y el inversor. En este trabajo se demuestra que una amplitud de tercera armónica igual a 1/6 de la fundamental proveía el máximo aprovechamiento de la tensión continua. Años más tarde se propone esta misma idea con implementación digital [Hou 84], y luego se sigue trabajando buscando el valor óptimo para la inyección ya no de una tercera armónica sino de una componente homopolar genérica [Bow 97].

1.2.2.1. Modulación vectorial

Las modulaciones anteriores se basan en un sistema monofásico y luego lo extienden al trifásico simplemente agregando dos referencias más desfasadas 120° e implementando básicamente tres moduladores independientes. En ningún momento se tiene en cuenta la

interacción entre las distintas columnas del inversor. Sin embargo es posible obtener la misma tensión sobre una carga trifásica con distintos esquemas de conmutación en cada columna. Este hecho sugiere tratar al inversor trifásico como una unidad y no como tres moduladores independientes. A principios de los 80 se propuso este enfoque utilizando la representación de vectores espaciales de tensión [Pfa 82][Van 86]. Esta representación se inspira en la distribución espacial de los bobinados de una máquina de CA para producir un campo rotante y de allí toma su nombre. Básicamente transforma una terna trifásica a un vector en el plano complejo, muy similar a la representación del campo rotante. La modulación vectorial aprovecha la representación de vectores espaciales, identifica los posibles puntos de operación del inversor en el plano complejo, e intenta aproximar cualquier vector promediando los distintos puntos de operación del inversor. De este modo la conmutación de cada columna está determinada por todo el sistema trifásico no por una fase en particular. Esta modulación resulta muy atractiva para su implementación digital y se han realizado muchas propuestas en torno a ella en los últimos tiempos [Hol 87][Mur 87b][Kog 89][Oga 89][Mur 92][Hol 93a][Hol 93b][Hub 95][Tzo 97][Mon 98][Sch 98][Ton 98b][Ton 99a] [Ton 99b] [Bak 00] [Ton 01].

1.3. Contribuciones de la Tesis

La principal contribución de ésta tesis, es el desarrollo de un modulador vectorial para inversores de potencia, diseñado como un bloque independiente del control de la aplicación. Éste recibe las señales del vector de referencia, como señal de salida del control de la aplicación, y genera a la salida las señales de control de las llaves del inversor de potencia. De esta manera el algoritmo de control de la carga no tiene que perder tiempo trabajando básicamente como un “contador inteligente”.

Para la realización de este desarrollo se estudiaron con detalle: a) la teoría del vector espacial, b) las alternativas de sobremodulación para lograr una transición de un modo de operación lineal has obtener una onda cuasi cuadrada con completo aprovechamiento de la tensión continua disponible, c) la compensación del efecto de tiempo muerto [TON 00]. También se estudió la disponibilidad de dispositivos lógicos programables (FPLD) aptos para la implementación del modulador vectorial.

Con los temas analizados, primeramente se realizó un modulador simple, lineal y sin compensación de tiempo muerto, apto para una gran variedad de aplicaciones en las cuales no es critico el aprovechamiento de la continua y que no trabajan con índices de modulación muy bajos, por lo cual tampoco es crítica la distorsión introducida por el tiempo muerto en la excitación de las columnas [TON 98b] [TON 99a]. Este primer desarrollo fue presentado en una competencia estudiantil, el IEEE Myron Zucker Student Design Award en la cual obtuvo el segundo puesto [Ton 99b].

Con la experiencia ganada con el primer prototipo, se amplió de desarrollo para incorporar la sobremodulación y la compensación del tiempo muerto. De este modo se obtuvo un modulador vectorial universal que fue apto para su empleo en toda aplicación que requiera de inversores de potencia trifásicos. El modulador diseñado fue testeado con un inversor de potencia obteniendo resultados experimentales muy satisfactorios [TON 01].

1.4. Organización de la Tesis

La tesis está organizada de la siguiente manera. En el Capítulo 2 se analiza con profundidad la modulación vectorial en régimen lineal. Partiendo de las primeras definiciones y los principios fundamentales se estudia el cálculo de los tiempos para la modulación. Se realiza una comparación con la modulación sinusoidal simple y con inyección de componente homopolar. Finalmente se dan algunas nociones sobre la posible implementación de un modulador vectorial.

En el Capítulo 3 se estudian las familias de dispositivos lógicos programables en campo (FPLD) con el objetivo de seleccionar el más adecuado para implementar el modulador vectorial universal.

En el Capítulo 4 se presenta el diseño e implementación de un modulador vectorial lineal y se muestran los resultados de simulación digital.

En el Capítulo 5 se analiza el funcionamiento del modulador vectorial fuera del rango lineal extendiendo su funcionamiento hasta el modo six-step o de onda cuadrada. Se distinguen dos modos distintos de sobremodulación que permiten aumentar el rango de utilización de la tensión continua disponible pagando el precio de introducir armónicas de bajas frecuencias. Luego del análisis se dan indicaciones sobre las modificaciones que se deben introducir en el modulador básico del capítulo 4 para incluir la sobremodulación.

En el Capítulo 6 se analiza el efecto de tiempo muerto. Este efecto es una consecuencia directa de intentar evitar el cortocircuito del Bus de continua cerrando simultáneamente las dos llaves de una columna del VSI. Para ello se introduce un retardo en el encendido de una llave respecto al apagado de la otra llave de la misma columna. Durante ese tiempo la tensión de la columna queda fijada por la carga alejándose de la referencia deseada. Se analiza el problema y se estudian posibles métodos de corrección del mismo. Se elige el más apropiado para ser incluido en el modulador y se dan pautas sobre como implementarlo.

En el Capítulo 7 se describe detalladamente la implementación del modulador vectorial universal. Éste se basa en el modulador presentado en el capítulo 4 al cual se le incorporan: a) sobremodulación, para lograr un completo aprovechamiento de la tensión continua disponible, y b) compensación de tiempo muerto con la finalidad de evitar la distorsión de la corriente de carga y también para una mejor utilización de la tensión continua. Se presentan primero los resultados de simulación digital para evaluar el diseño de la FPLD y finalmente los resultados experimentales obtenidos sobre un inversor prototipo.

Finalmente en el Capítulo 8 se presentan las conclusiones de la Tesis y se realizan algunas sugerencias para futuros trabajos en el tema.

2. MODULACIÓN VECTORIAL

2.1. Introducción

El control de la potencia eléctrica se realiza utilizando convertidores electrónicos de potencia. Los convertidores controlan el caudal de flujo de energía eléctrica, usando dispositivos electrónicos que actúan como llaves que se cierran y abren rápidamente. Los algoritmos que controlan las llaves, realizando la modulación de ancho de pulso (PWM) son varios [Bos 97]. La frecuencia de conmutación de los primeros convertidores de potencia era típicamente baja. Con el continuo crecimiento en la capacidad de conmutación de los dispositivos electrónicos, los convertidores electrónicos de potencia pueden transferir energía a la carga en cantidades pequeñas y controladas de tal manera que las armónicas y sus efectos perjudiciales sean substancialmente reducidos.

La técnica de PWM desempeña un rol importante en la minimización de componentes armónicas y en las pérdidas de conmutación en el convertidor, por lo tanto es fundamental implementar una buena estrategia de modulación. La posibilidad de disponer dispositivos electrónicos de conmutación cada vez más rápidos, permite implementar diversas técnicas de modulación para controlar los inversores de potencia. En principio el más utilizado era el método subarmónico o modulación sinusoidal, que era el más apto para la implementación analógica. Hoy la teoría del vector espacial provee un camino natural para implementar la modulación de ancho de pulso en inversores de potencia trifásicos [Hol 94c] [Kaz 91]. Este método permite una utilización completa del bus de tensión continua sin necesidad de inyectar señales de secuencias cero (o armónicas triples), como es el caso del método de muestreo regular [Bow 97].

En sus comienzos la implementación del PWM involucró circuitos analógicos. Debido al incremento en la complejidad de los diseños de PWM y la demanda de reducir el tamaño y el costo del control, tuvo primacía el desarrollo de controladores completamente digitales; haciendo uso de distintos dispositivos tales como, microcontroladores de alta perfomance (μC), procesadores digitales de señal (DSP), circuitos integrados de aplicaciones especificas (ASIC) y dispositivos de lógica programable (PLD).

La generación de una forma de onda monofásica de AC usando amplificadores de conmutación con PWM, es una tarea bastante sencilla. El problema se vuelve más complejo cuando se considera un sistema trifásico con PWM. El circuito más usado como convertidor

de potencia, el puente inversor, está compuesto por tres columnas como muestra la Fig. 2.1. Cada columna tiene dos llaves, donde una y sólo una está cerrada cada vez. En este capítulo se analizan las principales características de la modulación del vector espacial. Se realiza una comparación con la modulación empleando el método subarmónico con inyección de componentes homopolares. Finalmente se dan algunas nociones básicas sobre la implementación del modulador vectorial con el objetivo de poder especificar el dispositivo digital a emplear.

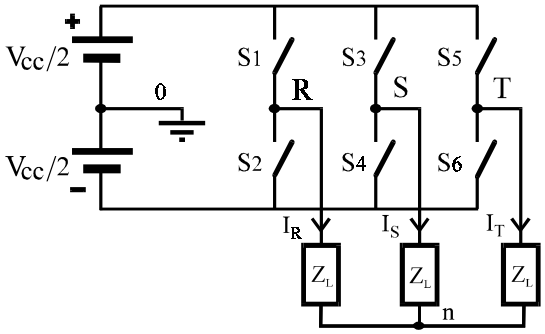


Figura 2.1: Inversor trifásico.

2.2. Teoría del vector espacial

Para un sistema trifásico simétrico y equilibrado, se define el vector espacial de tensión en términos de las tensiones de fase $u_{Rn}(t)$, $u_{Sn}(t)$, $u_{Tn}(t)$ como:

$$\bar{\mathbf{V}} = \frac{2}{3}\left(u_{Rn}(t) + u_{Sn}(t)\, e^{j\frac{2\pi}{3}} + u_{Tn}(t)\, e^{j\frac{4\pi}{3}} \right) \tag{2.1}$$

Ésta, es una representación compacta y poderosa, que transforma las tres variables de fase en el plano complejo. De esta manera se trata al sistema trifásico como un conjunto y no cada fase en forma individual. El coeficiente 2/3 en (2.1) garantiza que la amplitud sea invariante con la transformación. Es decir que el vector espacial de una terna trifásica tiene igual amplitud que las componentes de la terna. La definición puede extenderse directamente a otras variables como ser: corriente y flujo.

El puente inversor trifásico tiene tres columnas, donde cada una tiene dos llaves como muestra la Fig. 2.1. Para evitar cortocircuitos del bus de continua y fijar la tensión sobre la carga, una y sólo una, de las llaves de cada columna, debe estar cerrada en cada instante. Así resultan sólo ocho combinaciones diferentes de las llaves. Cada combinación

define un vector espacial de tensión, determinado por el estado de las seis llaves semiconductoras (S_1...S_6).

Los ocho vectores espaciales quedan representados por:

$$\vec{\mathbf{V}}_{\mathbf{k}} = \begin{cases} \frac{2}{3} V_{cc}\, e^{j\frac{\pi}{3}(k-1)} & k = 1,...,6 \\ 0 & k = 0, 7 \end{cases} \tag{2.2}$$

El vector de tensión de salida del inversor ($\vec{V}_k$) puede asumir sólo siete ubicaciones en el plano complejo. En Fig. 2.2 se muestran los vectores espaciales de tensión, donde la notación V_1 = (S_1 S_4 S_6) indica qué llave está cerrada para cada ubicación. Los vectores espaciales de tensión dividen al plano complejo en seis sectores, los cuales están limitados por dos de ellos. Hay una redundancia en (2.2) para k = 0 y k = 7, que genera el mismo vector nulo cuando las tres llaves de arriba, o las tres de abajo, están cerradas. Este vector se denomina vector espacial nulo, y los otros seis, vectores espaciales activos. Se aprovechará la redundancia del vector espacial nulo para minimizar las conmutaciones del inversor.

En la Fig. 2.3, se muestran las formas de onda resultantes de la aplicación sucesiva de los 6 vectores espaciales activos. El período fundamental se subdivide en seis intervalos de tiempo iguales. Este funcionamiento es referido como el modo six-step, en el cual la frecuencia de conmutación de las llaves es igual a la frecuencia fundamental de la tensión de salida del inversor. En la Fig. 2.3.a se muestra la tensión de columna de los terminales del inversor respecto del terminal de referencia (0 en la Fig. 2.1). La Fig. 2.3.b da la tensión del neutro respecto de la referencia, asumiendo la carga simétrica y balanceada. Ésta es positiva

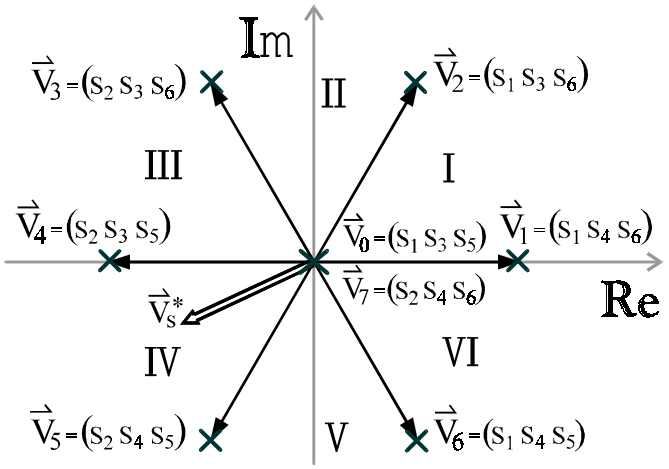


Figura 2.2: Ubicación de los vectores espaciales de tensión $\mathbf{V_k}$ *y del vector de referencia* $\vec{\mathbf{V}}_{\mathbf{s}}^{*}$.

cuando más de una de las llaves de la parte alta de la columna está cerrada, y negativa en el caso contrario. Su amplitud es de un sexto de la tensión en los terminales de cc del inversor. Esta forma de onda tiene todos los armónicos triples que no están en la tensión de fase. Los armónicos triples forman un sistema de secuencia cero, éstos no producen corrientes en los bobinados de la máquina cuando no existe conexión del centro de la estrella de la carga. Las tensiones de fase de la carga se muestran en la Fig. 2.3.c. Puede observarse en la Fig. 2.3 que un cambio de tensión en alguna de las columnas, invariablemente influye en la tensión de las otras dos fases. Por lo tanto, es oportuno para el diseño de la estrategia de PWM, considerar al sistema trifásico como un conjunto, en vez de considerar las tensiones de fase de forma individual. La técnica de modulación del vector espacial cumple exactamente con este requisito.

El valor instantáneo del vector de tensión del inversor tiene sólo siete valores discretos, pero es posible obtener diferentes tensiones como el valor medio de sucesivas conmutaciones. Tomando el valor medio de la aplicación sucesiva de distintos vectores espaciales, se puede generar cualquier vector de tensión que esté dentro del hexágono cuyos vértices son los vectores espaciales activos (V_1...V_6). En particular, es posible sintetizar el

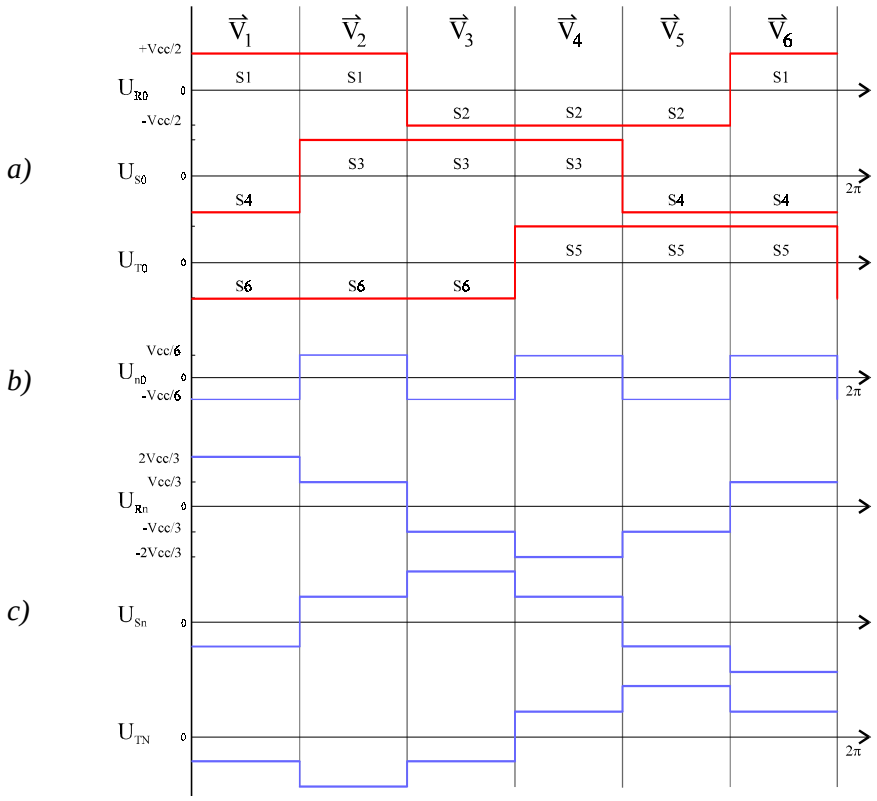


Figura 2.3: a) Tensiones de columna u_{R0}, u_{S0} y u_{T0}; b) Tensión de neutro u_{n0}; c) Tensiones de fase u_{Rn}, u_{Sn} y u_{Tn}.

vector de tensión que corresponde a un sistema trifásico sinusoidal balanceado $\vec{\mathbf{V}}_\mathbf{s}^*$. Este vector describe trayectorias circulares, donde la amplitud de la sinusoide fija el radio de giro, y la velocidad de rotación corresponde a su frecuencia.

$$\vec{\mathbf{V}}_\mathbf{s}^* = V^* e^{j\omega t} \tag{2.3}$$

La estrategia de modulación del vector espacial consiste en aproximar una muestra del vector de referencia $\vec{\mathbf{V}}_\mathbf{s}^*$ por la aplicación sucesiva de tres vectores espaciales V_a, V_b y V_N durante un intervalo de tiempo $\Delta t = 1/(2\ f_s)$. V_a y V_b son los dos vectores espaciales activos (V_1,...,V_6) adyacentes al vector de referencia, V_N es uno de los vectores nulos V_0 ó V_7 y f_s es la frecuencia de conmutación. Al intervalo de tiempo Δt se lo denomina tiempo de subciclo. En la Fig. 2.2 se representa una situación en la cual $\vec{\mathbf{V}}_\mathbf{s}^*$ se encuentra en el sector IV que está limitado por V_4 y V_5; entonces debe tomarse $V_a = V_4$, $V_b = V_5$ y $V_N = V_0$ ó $V_N = V_7$.

El valor medio de tensión que resulta de la aplicación sucesiva de los vectores espaciales en un intervalo de tiempo Δt, debe ser igual al vector de referencia. Entonces,

$$\vec{\mathbf{V}}_\mathbf{a}\ t_a + \vec{\mathbf{V}}_\mathbf{b}\ t_b + \vec{\mathbf{V}}_\mathbf{N}\ t_0 = \vec{\mathbf{V}}_\mathbf{a}\ t_a + \vec{\mathbf{V}}_\mathbf{b}\ t_b = \vec{\mathbf{V}}_\mathbf{s}^*\ \Delta t \tag{2.4}$$

con

$$t_a + t_b + t_0 = \Delta t \tag{2.5}$$

donde t_a, t_b y t_0 son los intervalos de tiempo durante los cuales se aplican los vectores espaciales V_a, V_b y V_N respectivamente.

2.2.1. Generación de un vector de referencia sin utilizar los vectores nulos

En esta sección se buscará una expresión que permita calcular los tiempos de los vectores activos t_a y t_b, cuando no se utilizan los vectores nulos en la secuencia ($t_0 = 0$). Se utilizará para el análisis un sector genérico, limitado por los vectores espaciales activos V_a y V_b, que representa a cualquiera de los 6 sectores en el plano complejo. Cuando los vectores nulos no se aplican ($t_0 = 0$), se tienen las siguientes ecuaciones genéricas:

$$\begin{aligned} &\Delta t = t_a + t_b \\ &\vec{V} = \frac{1}{\Delta t}\left[\int_0^{t_a} \vec{V}_a\, dt + \int_{t_a}^{\Delta t} \vec{V}_b\, dt\right] = \frac{1}{\Delta t}\left[\vec{V}_a.t_a + \vec{V}_b.\left(\Delta t - t_a\right)\right] \end{aligned} \tag{2.7}$$

Para sintetizar la muestra del vector de referencia, se debe identificar el sector donde se encuentra y reemplazar los vectores genéricos por aquellos que limitan el sector identificado.

$$\begin{aligned} \vec{V}_a &= \tfrac{2}{3} V_{cc} e^{j\frac{\pi}{3}(k-1)} \\ \vec{V}_b &= \tfrac{2}{3} V_{cc} e^{j\frac{\pi}{3}k} \qquad k = 1,\ldots,6 \end{aligned} \tag{2.8}$$

Luego, reemplazando (2.8) en (2.7):

$$\vec{V} = \tfrac{2}{3}\tfrac{1}{\Delta t} V_{cc} e^{j\frac{\pi}{3}k} \left[\Delta t - \tfrac{1}{2} t_a - j \tfrac{\sqrt{3}}{2} t_a) \right] \tag{2.9}$$

En cada sector (k = 1...6), la expresión (2.9) representa una recta en el plano complejo. Estas seis rectas forman un hexágono, cuyos vértices son los vectores espaciales activos, como se muestra en la Fig. 2.4. El hexágono es el lugar geométrico del máximo vector de referencia que se puede generar. Si se utilizan los vectores nulos durante un tiempo constante (t_0 = cte), el lugar geométrico del vector de referencia también será un hexágono, pero más chico que el correspondiente a $t_0 = 0$ [Ton 98]. Está claro entonces que se puede generar cualquier otro vector de referencia siempre que éste se encuentre dentro del hexágono.

Para obtener la expresión del tiempo t_a, cuando el vector de referencia se mueve sobre el hexágono, se igualan las componentes real e imaginaria de (2.9) con las componentes del vector de referencia $\vec{\mathbf{V}}_\mathbf{s}^*$ (ver Fig. 2.4):

$$\begin{aligned} \text{Re}: \left|\vec{V}_s^*\right| \cos(\alpha) &= \tfrac{2}{3}\tfrac{V_{cc}}{\Delta t}\left[(\Delta t - \tfrac{1}{2} t_a) \cos(\tfrac{\pi}{3} k) + \tfrac{\sqrt{3}}{2} t_a \sin(\tfrac{\pi}{3} k) \right] \\ \text{Im}: \left|\vec{V}_s^*\right| \sin(\alpha) &= \tfrac{2}{3}\tfrac{V_{cc}}{\Delta t}\left[(\Delta t - \tfrac{1}{2} t_a) \sin(\tfrac{\pi}{3} k) - \tfrac{\sqrt{3}}{2} t_a \cos(\tfrac{\pi}{3} k) \right] \end{aligned}$$

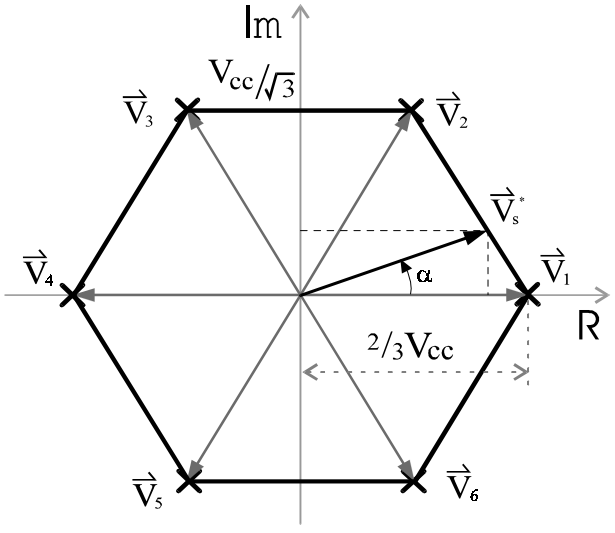


Figura 2.4: Límite para el vector espacial.

eliminando el módulo: $$\frac{\cos(\alpha)}{\sin(\alpha)} = \frac{\Delta t \cos(\frac{\pi}{3}k) - t_a \cos(\frac{\pi}{3}k + \frac{\pi}{3})}{\Delta t \sin(\frac{\pi}{3}k) - t_a \sin(\frac{\pi}{3}k + \frac{\pi}{3})}$$

$$t_a = \Delta t \frac{\sin(\frac{\pi}{3}k)\cos(\alpha) - \cos(\frac{\pi}{3}k)\sin(\alpha)}{\sin(\frac{\pi}{3}k + \frac{\pi}{3})\cos(\alpha) - \cos(\frac{\pi}{3}k + \frac{\pi}{3})\sin(\alpha)} = \Delta t \frac{\sin\left[\frac{\pi}{3}k - \alpha\right]}{\sin\left[\frac{\pi}{3}(k+1) - \alpha\right]} \qquad (2.10)$$
$$t_b = \Delta t - t_a$$

El único dato necesario para calcular los tiempos de aplicación de los vectores dados por (2.10) es el ángulo α. Es decir, la ubicación del vector de referencia en el plano complejo.

2.2.2. Generación de un vector de referencia con trayectoria circular

Como se vio en el punto 2.2 una terna trifásica sinusoidal determina un vector espacial que describe trayectorias circulares. Por ello se tiene especial interés en generar estas trayectorias. Para que el vector espacial de tensión media siga una trayectoria circular y no hexagonal, se deben aplicar los vectores nulos durante tiempos variables para que el valor del módulo del vector de referencia se mantenga constante.

Los intervalos de tiempo t_a y t_b se obtienen igualando las componentes de (2.4), mientras que t_0 se calcula directamente con (2.5). En la Fig. 2.5 se muestran el vector de referencia con sus componentes en un sector genérico. Las componentes del vector $\vec{\mathbf{V}}_s^*$ son $\vec{V}_A$ y $\vec{V}_B$ que podemos escribirlas como:

$$\begin{cases} \vec{V}_A = \frac{1}{\Delta t}\vec{V}_a t_a \\ \vec{V}_B = \frac{1}{\Delta t}\vec{V}_b t_b \end{cases} \qquad (2.11)$$

donde: $\Delta t = t_a + t_b + t_0$

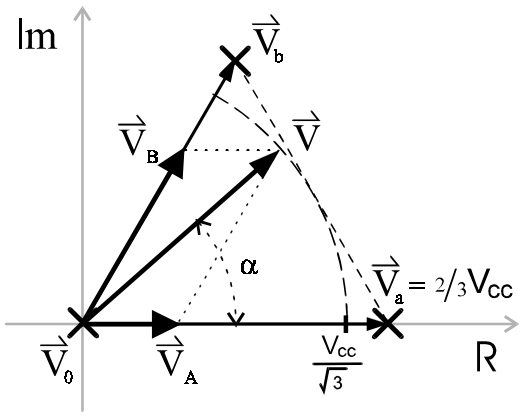


Figura 2.5: Consideraciones geométricas para los tiempos t_a y t_b.

Expresando las proyecciones de $\vec{\mathbf{V}}_s^*$ sobre los ejes Real e Imaginario en función de sus componentes:

$$\text{Im: } |\vec{V}| \sin\alpha = |\vec{V}_B| \sin 60^o$$
$$\text{Re: } |\vec{V}| \cos\alpha = |\vec{V}_A| + |\vec{V}_B| \cos 60^o$$

$$|\vec{V}_B| = \frac{|\vec{V}|}{\sin 60^o}\sin\alpha$$
$$|\vec{V}_A| = \frac{|\vec{V}|}{\sin 60^o}\sin(60^o - \alpha) \qquad (2.12)$$

Reemplazando (2.11) en (2.12) obtenemos las ecuaciones de los tiempos t_a, t_b y t_0. Se obtiene una solución genérica para cada sector en el plano complejo; donde α siempre varía en el intervalo [0°, 60°).

$$t_a = \sqrt{3}\,\frac{|\vec{V}|}{V_{cc}}\,\Delta t\,\sin(\pi/3 - \alpha)$$
$$t_b = \sqrt{3}\,\frac{|\vec{V}|}{V_{cc}}\,\Delta t\,\sin(\alpha) \qquad (2.13)$$
$$t_0 = \Delta t - t_a - t_b$$

El valor máximo que puede alcanzar el módulo del vector de referencia, corresponde a la trayectoria tangente a cada lado del hexágono. Siendo su valor $\frac{V_{cc}}{\sqrt{3}}$, como puede deducirse de la Fig. 2.5.

La Fig. 2.6 muestra los intervalos de tiempo (t_a, t_b y t_0) para dos valores del vector de referencia, que describen trayectorias circulares. Las líneas sólidas representan los intervalos de tiempo para la máxima amplitud, y las líneas de trazos para la mitad de la amplitud máxima. Para el caso de máxima amplitud, cuando α = 30°, $t_a = t_b = \Delta t/2$ y $t_0 = 0$. Esto significa que el vector de referencia es tangente al hexágono y su valor es máximo.

El lugar que queda entre la circunferencia inscripta y el hexágono (Fig. 2.5), se denomina región de sobremodulación. En ésta región se puede aprovechar más la tensión continua disponible, pagando el precio de un aumento en el contenido armónico de baja frecuencia.

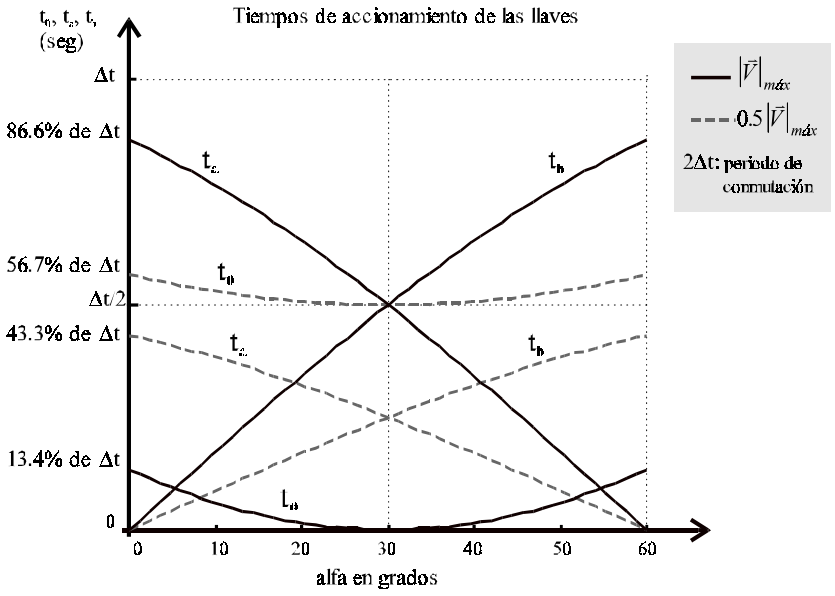


Figura 2.6: Tiempos t_a, t_b y t_0 para tensiones sinusoidales.

2.2.3. Índice de modulación (m)

Se define el índice de modulación (m), como la relación entre la componente fundamental de la tensión de columna (u_{R01}) y la componente fundamental de tensión en operación six-step ($u_{1six\text{-}step}$).

$$m = \frac{u_{R01}}{u_{1\ six-step}} = \frac{\pi}{2V_{cc}} u_{R01} \tag{2.14}$$

El índice de modulación varía entre 0 y 1. El valor m = 1 sólo puede alcanzarse en el modo six-step. Esta definición de m permitirá comparar distintas técnicas de modulación de ancho de pulso. Para el caso del vector de referencia con trayectoria circular, el índice de modulación máximo (m_{max}) ocurre cuando la trayectoria resulta tangente al hexágono en $\alpha = 30°$. Esto ocurre para $u_{R01} = V_{CC}/\sqrt{3}$ y el índice de modulación es:

$$m_{\max} = \frac{\pi}{2\sqrt{3}} = 0.9069 \tag{2.15}$$

La (2.15) muestra que la modulación vectorial cubre el 90.7% del aprovechamiento del bus de continua, manteniendo una relación lineal entre el vector de referencia y la tensión de salida (componente fundamental de la tensión de columna).

2.3. Comparación entre la modulación vectorial (SVM) y el método subarmónico (SPWM)

Para comparar los resultados de la modulación del vector espacial con otros métodos, se evalúa la tensión promedio de columna, sobre un periodo de conmutación [Van 86]. En la Fig. 2.7 se muestra la forma de onda genérica de la secuencia de conmutación de SVM para cada sector; correspondiente a la tensión de columna u_{R0} (Fig. 2.1). El valor medio de u_{R0} se

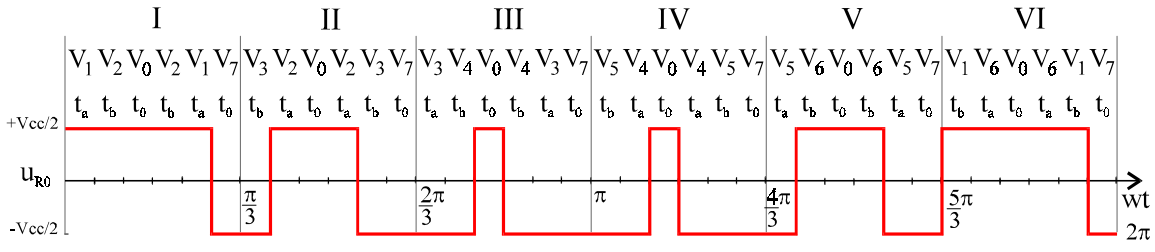


Figura 2.7: Conmutaciones en cada sector.

obtiene como muestran las expresiones (2.16), en éstas los tiempos t_a y t_b se reemplazan por (2.13) quedando en función del ángulo α que varía entre 0 y $\pi/3$. El mismo desarrollo puede aplicarse a las otras dos tensiones de columna u_{S0} y u_{T0}.

Sector I:

$$\begin{aligned}\overline{U}_{R0} &= \frac{V_{cc}/2}{2\Delta t}\left[t_a + t_b + t_0 + t_b + t_a - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[t_a + t_b\right] = \\ &= \frac{V_{cc}/2}{\Delta t}\sqrt{3}\frac{|\vec{V}|}{V_{cc}}\Delta t\left[sin(\pi/3 - \alpha) + sin(\alpha)\right] = \frac{\sqrt{3}}{2}|\vec{V}|sin(\pi/3 + \alpha)\end{aligned} \qquad (2.16.a)$$

Sector II:

$$\overline{U}_{R0} = \frac{V_{cc}/2}{2\Delta t}\left[-t_b + t_a + t_0 + t_a - t_b - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[t_a - t_b\right] = \frac{3}{2}|\vec{V}|sin(\pi/6 - \alpha) \qquad (2.16.b)$$

Sector III:

$$\overline{U}_{R0} = \frac{V_{cc}/2}{2\Delta t}\left[-t_a - t_b + t_0 - t_b - t_a - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[-t_a - t_b\right] = -\frac{\sqrt{3}}{2}|\vec{V}|sin(\pi/3 + \alpha) \qquad (2.16.c)$$

Sector IV:

$$\overline{U}_{R0} = \frac{V_{cc}/2}{2\Delta t}\left[-t_a - t_b + t_0 - t_b - t_a - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[-t_a - t_b\right] = -\frac{\sqrt{3}}{2}|\vec{V}|sin(\pi/3 + \alpha) \qquad (2.16.d)$$

Sector V:

$$\overline{U}_{R0} = \frac{V_{cc}/2}{2\Delta t}\left[-t_a + t_b + t_0 + t_b - t_a - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[-t_a + t_b\right] = -\frac{3}{2}|\vec{V}|sin(\pi/6 - \alpha) \qquad (2.16.e)$$

Sector VI:

$$\overline{U}_{R0} = \frac{V_{cc}/2}{2\Delta t}\left[t_a + t_b + t_0 + t_b + t_a - t_0\right] = \frac{V_{cc}/2}{\Delta t}\left[t_a + t_b\right] = \frac{\sqrt{3}}{2}|\vec{V}|sin(\pi/3 + \alpha) \qquad (2.16.f)$$

El ángulo α de las ecuaciones (2.16) puede expresarse en función del tiempo y de los sectores como:

$$\alpha = \omega t - (k-1)\frac{\pi}{3}$$

donde k = 1 corresponde al Sector I, k = 2 corresponde al Sector II, etc.

Por lo tanto, en la expresión (2.17) se da la tensión promedio de columna u_{R0} en función del tiempo.

$$\overline{U}_{R0} = \begin{cases} \frac{\sqrt{3}}{2}|\vec{V}|\cos(\omega t - \frac{\pi}{6}) & 0 \le \omega t < \frac{\pi}{3};\ \ \pi \le \omega t < \frac{4\pi}{3} \\ \frac{3}{2}|\vec{V}|\cos(\omega t) & \frac{\pi}{3} \le \omega t < \frac{2\pi}{3};\ \ \frac{4\pi}{3} \le \omega t < \frac{5\pi}{3} \\ \frac{\sqrt{3}}{2}|\vec{V}|\cos(\omega t + \frac{\pi}{6}) & \frac{2\pi}{3} \le \omega t < \pi;\ \ \frac{5\pi}{3} \le \omega t < 2\pi \end{cases} \qquad (2.17)$$

Expresando (2.17) en serie de Fourier se obtiene:

$$\bar{U}_{R0} = \left|\vec{V}\right| \sum_{n=1}^{\infty} A_n \cos(n\omega t) \qquad \text{donde n pertenece a los enteros positivos} \qquad (2.17.a)$$

El coeficiente A_n es:

$$A_n = \frac{\left[3n\cos(\frac{n\pi}{6}) + \sqrt{3}\sin(\frac{n\pi}{6})\right]\left[-\sin(\frac{n\pi}{6}) + \sin(\frac{n\pi}{2}) - \sin(\frac{5n\pi}{6}) + \sin(\frac{7n\pi}{6}) - \sin(\frac{3n\pi}{2}) + \sin(\frac{11n\pi}{6})\right]}{2\pi(n^2 - 1)} \qquad (2.17.b)$$

De (2.17.b) puede observarse para n = 1, aplicando L'Hopital, que la amplitud de la componente fundamental (2.18) es igual al módulo del vector espacial de referencia. También puede observarse que la tercera armónica es de aproximadamente el 20% de la fundamental.

$$\bar{U}_{R01} = \left|\vec{V}\right| \cos(\omega t) \qquad (2.18)$$

La tensión de línea $u_{RS}(\omega t)$ está dada por:

$$\bar{U}_{RS} = \bar{U}_{R0} - \bar{U}_{S0} = -\sqrt{3}\left|\vec{V}\right| \sin(\omega t - \tfrac{\pi}{3}) \qquad (2.19)$$

La Fig. 2.8 muestra las gráficas de (2.17), (2.18), (2.19) y la tensión de columna $u_{S0}(\omega t) = u_{R0}(\omega t - 120°)$, correspondiente al módulo del vector de referencia igual al máximo sin sobremodulación ($V_{cc}/\sqrt{3}$).

Con la modulación vectorial, la tensión de línea vista por la carga es sinusoidal, como lo indica (2.19). Sin embargo, la tensión de columna dada por (2.17) no es sinusoidal. Armónicas de orden triple adicionadas a las tensiones de columna no afectan a las tensiones de línea. La máxima amplitud de (2.17) se da cuando el módulo del vector de referencia vale $V_{cc}/\sqrt{3}$, siendo este valor la máxima tensión de salida del inversor para la modulación lineal. Debido a que la máxima tensión de salida del inversor se obtiene en el modo six-step, resulta interesante conocer cuál es el porcentaje que cubre la modulación vectorial sin sobremodular. Haciendo la relación entre $V_{cc}/\sqrt{3}$ y $2/\pi V_{cc}$, se observa que la modulación vectorial cubre el 90,7 % de lo que se puede obtener con el modo six-step.

En un modulador sinusoidal, la forma de onda dada por (2.17) sería la modulante. Para la modulación sinusoidal, la máxima tensión de salida está dada por el bus de continua, siendo ésta de $V_{cc}/2$ en el caso de la Fig. 2.1. Entonces la modulación sinusoidal cubre sólo el

78,5% del valor que se puede obtener con el modo six-step; éste puede incrementarse inyectando armónicas de orden triple como propuso Buja en 1975.

Por lo tanto, con la modulación vectorial, se obtiene aproximadamente el 15% más del aprovechamiento del bus de continua, que con la modulación sinusoidal. Esto se logra de manera natural, o sea que no se inyectan componentes de secuencia cero.

En ambos casos para cubrir el 100% de la tensión de salida, o sea aprovechando al máximo la tensión del bus de continua, se debe sobremodular.

2.4. Nociones de implementación

Un modulador vectorial debe tomar muestras del vector de referencia, a la frecuencia de conmutación $f_s = 1/(2\ \Delta t)$, para poder calcular los tiempos dados por (2.13). Siendo f_s independiente de la frecuencia fundamental f_0 del vector de referencia, la modulación es asincrónica. Una muestra en módulo y fase del vector de referencia, permite identificar el sector donde está localizado y los vectores espaciales adyacentes que deben utilizarse. Para obtener los tiempos de conmutación que deben ser efectivamente implementados para

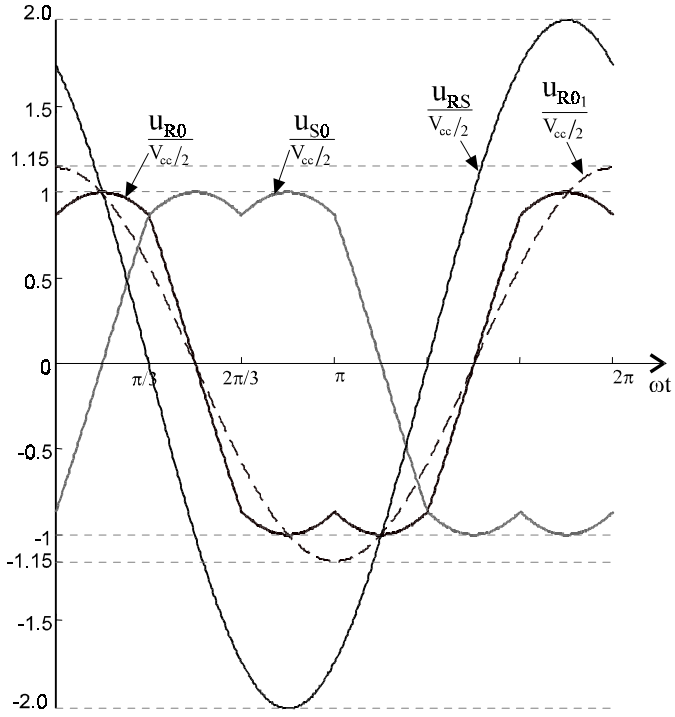


Figura 2.8: Formas de onda de: tensión de columna $u_{R0}(\omega t)$, $u_{S0}(\omega t)$, tensión de línea $u_{RS}(\omega t)$, y de la componente fundamental $u_{R01}(\omega t)$.

generar las señales de excitación de las llaves del inversor, se resuelve (2.13). La Fig. 2.9 muestra un diagrama en bloques que describe el método sintéticamente.

Existen distintas secuencias de vectores espaciales que permiten sintetizar el vector de referencia. Debe elegirse una conveniente para minimizar las conmutaciones de las llaves. Hay muchos trabajos que describen el uso de diferentes secuencias de acuerdo a sus objetivos [Hol 91][Hol 94c][Kaz 91]. En esta Tesis, la elección de la secuencia dentro de cada sector, será: $|<V_a><V_b><V_N>|<V_b><V_a><V_N>|$ [Del 91]. La Tabla 2.1 presenta las secuencias propuestas para cada sector. Utilizando estas secuencias, la transición de un vector espacial al siguiente, se realiza con la conmutación de una sola columna del inversor de potencia.

Tabla 2.1: Secuencias de conmutación propuestas

Sector	*Secuencia*					
	Δt (impar)			*Δt (par)*		
I	V_1	V_2	V_0	V_2	V_1	V_7
II	V_3	V_2	V_0	V_2	V_3	V_7
III	V_3	V_4	V_0	V_4	V_3	V_7
IV	V_5	V_4	V_0	V_4	V_5	V_7
V	V_5	V_6	V_0	V_6	V_5	V_7
VI	V_1	V_6	V_0	V_6	V_1	V_7

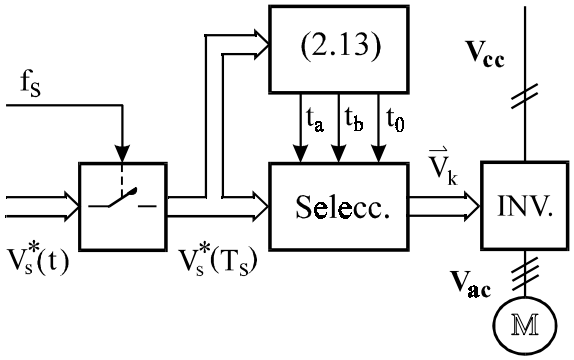


Figura 2.9: Diagrama en bloques del modulador vectorial.

El método de modulación vectorial puede ser implementado según el esquema de la Fig. 2.9 de dos maneras completamente diferentes, por software o por hardware. Ambas han sido ampliamente tratadas en la bibliografía [Bak 00][Hol 87][Mon 98][Ton 98]. Las implementaciones por software, normalmente realizan los cálculos de los tiempos t_a, t_b y t_0 de manera on-line. Si embargo, con el avance de la frecuencia de conmutación de los dispositivos de potencia, no pueden realizarse algoritmos de modulación demasiado complejos, aún empleando procesadores digitales de señales avanzados. Las realizaciones que utilizan hardware se basan en el cálculo off-line de los tiempos. En los últimos años, implementaciones de hardware utilizando dispositivos de aplicaciones específicas o dispositivos lógicos programables (ASIC/FPLD) están recibiendo una importante atención [Mon 98][Sch 98]. El empleo de FPGA para realizar técnicas de PWM provee ventajas tales

como, el diseño del hardware del modulador por medio de software, un hardware simple y robusto y una alta frecuencia de conmutación.

2.4.1. Diseño de la máquina secuencial del modulador lineal

El modulador debe ser capaz de controlar los intervalos de tiempo t_a, t_b y t_0 en que se aplican diferentes vectores espaciales. También debe seleccionar adecuadamente el vector espacial que corresponde al sector en cuestión y al intervalo de tiempo que se esté implementando. Para llevar a cabo estos requerimientos, el modulador debe constar de contadores, una máquina secuencial, una tabla y memorias. Los tiempos t_a y t_b, dados por (2.13), se calculan off-line y se almacenan en las memorias. En los contadores (Ct_a, Ct_b) se cargan los tiempos t_a y t_b respectivamente, para su implementación. Otro contador (C) genera el tiempo de subciclo Δt, contando un número fijo de periodos de clock. El tiempo t_0, que no se calculó off-line, se implementa en el intervalo de tiempo en que los contadores que implementan t_a o t_b están detenidos. En la tabla se guardan los estados de las seis llaves del inversor, correspondientes a cada vector espacial. Las entradas de la máquina secuencial son: el clock, las señales de encendido y apagado, las señales de control de los contadores y el subciclo. Ésta controla el encendido y apagado de los tres contadores, saltando de un estado a otro de acuerdo al valor lógico de las señales. También direcciona la tabla, seleccionando el vector espacial adecuado.

La máquina secuencial se diseña teniendo en cuenta que cada muestra del vector de referencia se implementa en dos subciclos consecutivos, por una secuencia de tres vectores espaciales por cada subciclo (Fig. 2.10). La secuencia elegida es $|<V_a><V_b><V_N>|$ en los

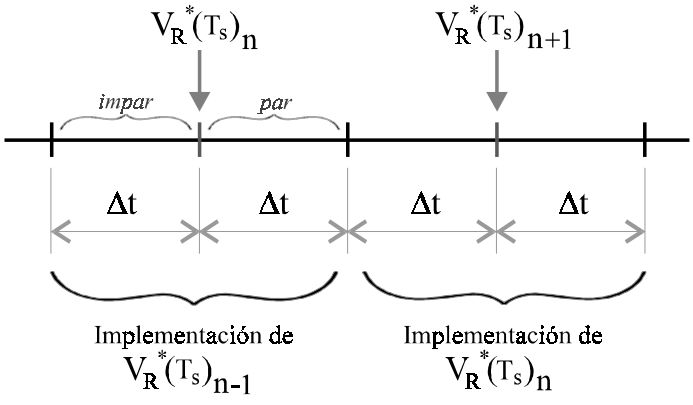


Figura 2.10: Diagrama de implementación de las muestras del vector de referencia.

subciclos impares y $|<V_b><V_a><V_N>|$ en los pares. Debido a esto, la máquina secuencial debe tener al menos tres estados para implementar los intervalos de tiempo t_a, t_b y t_0. En la Fig. 2.11 se muestra un diagrama de estados de la máquina secuencial.

El funcionamiento de la máquina secuencial tiene como base la señal de subciclo **Δt** del contador C, por lo que se debe poder controlar el encendido y apagado de dicho contador. Esto se realiza manejando la señal **Inicio** ya sea por software o por hardware. El estado inicial de la máquina secuencial es **S1**, en donde se implementa el intervalo de tiempo t_a. En los estados **S2** y **S3** se implementan los intervalos de tiempo t_b y t_0 respectivamente.

Durante los subciclos impares la señal **Δt**, está en estado bajo; mientras que permanece en estado alto para los subciclos pares. La implementación de una muestra del vector de referencia comienza con un subciclo impar (**Δt**=0), en el estado **S1**. Entonces el contador Ct_a debe iniciar la cuenta del tiempo t_a. Cuando Ct_a termina de implementar a t_a, entrega una señal de fin de cuenta $\mathbf{FC_{Ca}}$. Por lo que la maquina secuencial salta al estado **S2**, en donde el contador Ct_b se enciende implementado a t_b. Al finalizar la implementación de t_b, la señal de fin de cuenta $\mathbf{FC_{Cb}}$ se activa, haciendo que la maquina secuencial salte al estado **S3**. Aquí los contadores Ct_a y Ct_b permanecen detenidos, mientras se implementa el intervalo de tiempo t_0 ya que $t_0 = \Delta t - (t_a + t_b)$. Cuando la señal de subciclo **Δt** cambia al estado alto, comienza el subciclo par y la máquina secuencial salta al estado **S2**. Ahora la secuencia es

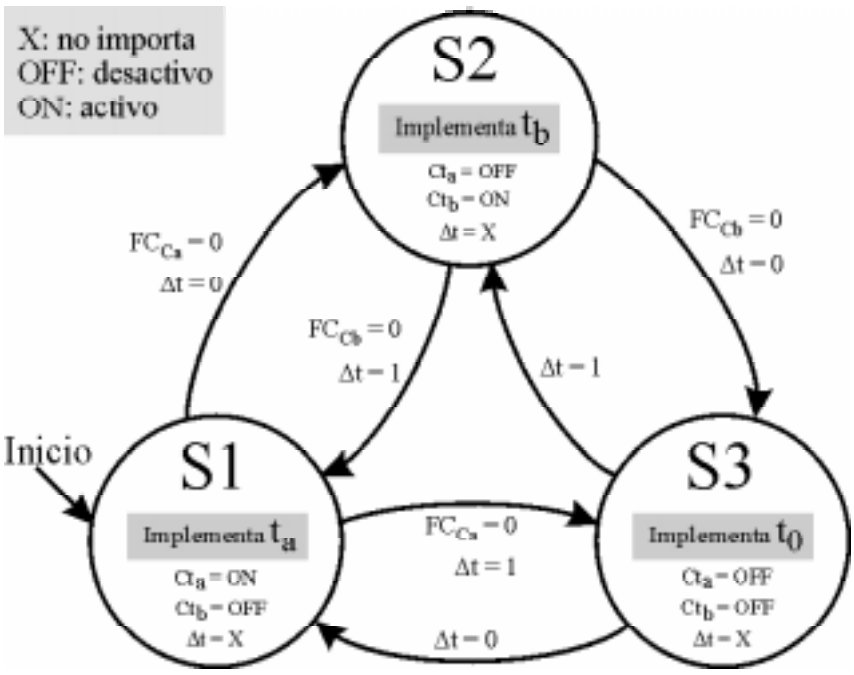


Figura 2.11: Diagramas de estados de la máquina secuencial.

inversa, como muestra la Fig 2. 11. La muestra termina de ser implementada en el estado **S3**, y cuando comienza la implementación de una muestra nueva la maquina secuencial salta al estado **S1** ya que $\Delta t = 0$.

2.5. Resumen

En este capítulo se presentó la teoría del vector espacial, mostrándose como se genera un vector de referencia con trayectoria circular. Así se llegó a obtener las ecuaciones de los tiempos de aplicación de cada vector espacial.

La modulación vectorial aprovecha el bus de continua un 15% más que la modulación sinusoidal. Esto se logra de manera natural, sin incorporar componentes de secuencia cero. Para aprovechar el 100% del bus de DC, se debe sobremodular llegando al six-step.

Se presentó un esquema en bloques para la implementación de la modulación vectorial. Se desarrolló la implementación de una máquina secuencial para un modulador lineal. De este modo quedan fijadas las exigencias de la modulación por vector espacial, sobre el sistema digital que deba implementarla.

3. LÓGICA PROGRAMABLE

3.1. Introducción

Existe una gran variedad de dispositivos para implementar diseños digitales [Sal 00][Ham 00][Ska 96], como se muestra en Fig. 3.1. Los componentes tradicionales de lógica estándar LSI-MSI, tienen un funcionamiento definido por el fabricante del integrado y el usuario debe conectar varios integrados para construir un circuito. Un paso adelante en la integración del sistema se logra con circuitos integrados de aplicaciones especificas (ASIC - Application Specific Integrated Circuit) y con dispositivos lógicos programable en campo (FPLD – Field Programmable Logic Device). Ambos son circuitos integrados en donde la funcionalidad interna la define el usuario, pero con metodologías bien diferentes. Por un lado los ASICs requieren de un paso adicional y personalizado de manufactura, para obtener la funcionalidad definida por el usuario. En cambio, los FPLDs sólo requieren de una programación adecuada por parte del usuario. El desarrollo de un diseño a nivel transistor (Full Custom), puede requerir muchos años de ingeniería de diseño y prueba. La única manera de garantizar el gran esfuerzo es con un alto volumen de producción. Ejemplos de dispositivos Full Custom, son los microprocesadores y las memorias usadas en las computadoras personales PC. Generalmente hoy en día se comienza con un diseño en dispositivos FPLD y cuando el diseño está bien probado y el volumen de producción lo justifican, se pasa a la etapa de manufactura para obtener un dispositivo “full custom”.

Los FPLDs representan un desarrollo relativamente nuevo en el campo de los circuitos de VLSI (Very Large Scale Integration). Ellos implementan cientos de compuertas lógicas en estructuras multi-nivel. La arquitectura de un FPLD consiste de una formación

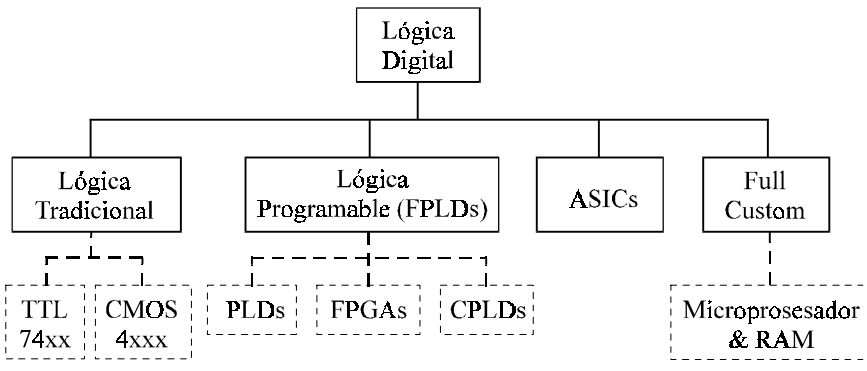


Figura 3.1: Tecnología en lógica digital

(array) de celdas lógicas, que pueden conectarse entre sí por llaves de programables; de la misma manera que en los tradicionales dispositivos lógicos programable (PLD) [Sal 00]. La programación eléctrica de las llaves, o sea la interconexión de las celdas lógicas, es la diferencia que existe entre un FPLD y un dispositivo lógico de máscara (MPLD), el cuál es programado utilizando técnicas de fabricación de integrados haciendo las interconexiones de metal. Los FPLDs pueden lograr niveles de integración más altos que los tradicionales PLDs, debido a su arquitectura más compleja de "cableado" (routing) e implementación lógica. El primer PLD que se desarrolló para la implementación de circuitos digitales fue el PLA (field-Programmable Logic Array). Estos dispositivos son desarrollados utilizando lógica AND-OR, o sea con entrada programable de compuertas AND seguida por un plano programable de compuertas OR. Por lo tanto las PLAs implementan lógica en forma de dos niveles de suma de productos. El siguiente paso en el desarrollo de PLDs fue la introducción de las PALs (Programmable Array Logic). Las PALs son dispositivos con un solo nivel de programación, compuertas AND programables seguidas de compuertas OR fijas. En muchos PLDs la salida de la compuerta OR esta conectada a un Flip-Flop, con la intención de poder implementar circuitos secuenciales. Una variante de la arquitectura básica de los PLDs aparece en muchos FPLDs. Los FPLD combinan varios PLDs simples en un solo chip, utilizando estructuras de interconexión programable. A esta combinación se la conoce con el nombre de CPLD (Complex PLD). La arquitectura de una FPGA (Field Programmable Gate Array) esta compuesta de un arreglo1 de celdas lógicas que se comunican entre sí y con los pines de I/O (Entradas/Salidas), por medio de conductores dentro de los "routing channels" [Ska 96].

La Fig. 3.2 pretende resumir algunas de las características principales de las alternativas actualmente utilizadas para el diseño de sistemas digitales, y mostrar por qué se

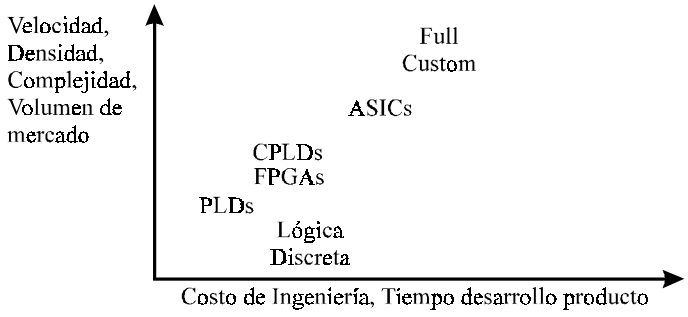


Figura 3.2: Características principales de las tecnologías utilizadas en el diseño de sistemas digitales.

considera que los FPLDs son la tecnología más prometedora para la implementación de una gran variedad de sistemas digitales.

En este capítulo se presentarán, para algunas familias, las distintas arquitecturas de los dispositivos lógicos programables. También se dará una introducción a las herramientas de programación de estos dispositivos como lo es el HDL (Hardware Description Language).

3.2. Dispositivos de Lógica Programable

Debido a la velocidad con que progresa la tecnología de los FPLDs, en este capítulo no se pretende hacer un estudio intensivo de todos ellos. Sólo nos concentraremos en los detalles más importantes de algunas familias de FPLD de Altera y Xilinx que resultan particularmente atractivos para la aplicación que nos ocupa. Estas empresas proveen la mayor variedad de FPLDs en el mercado, en términos de capacidad y propiedades.

Altera tiene dos tipos de FPLDs: los dispositivos de propósitos generales basados en tecnología de programación de compuerta flotante (EEPROM) utilizados en las series MAX (Multiple Array Matrix) 5000, 7000 y 9000. Y los dispositivos de tecnología SRAM (Static RAM) de la serie FLEX (Flexible Logic Element MatriX) 6000, 8000, 10K y 20K. Todos los dispositivos de Altera son de tecnología CMOS.

En este punto se describirán las familias MAX 7000S y FLEX10K para ALTERA, la familia XC4000 de XILINX, y brevemente la familia AT40K de ATMEL.

3.2.1. Altera MAX 7000 - Conceptos generales

La familia de CPLDs MAX, conjuntamente desarrollada por Altera y Cypress Semiconductor, fue la primer familia de CPLDs en el mercado (Altera los llamo MAX5000 y Cypress MAX340) [Ska 96]. La familia de dispositivos MAX7000 son CPLD con un equivalente de compuertas que va desde 600 a 20000. Pueden configurarse por medio de la programación de una EEPROM interna, lo cual permite que se mantenga la configuración cuando el dispositivo se desconecta de la fuente de alimentación. Estos dispositivos permiten ser reprogramados en el circuito (in-circuit reprogrammability). Contienen desde 32 a 256 celdas lógica (Logic Cells LC) llamadas macroceldas (macrocell). Estas se combinan en grupos de 16 dando lugar al denominado bloque de arreglos lógicos (Logic Array Block - LAB). Los LABs se pueden conectar con los adyacentes de manera directa, y con los demás

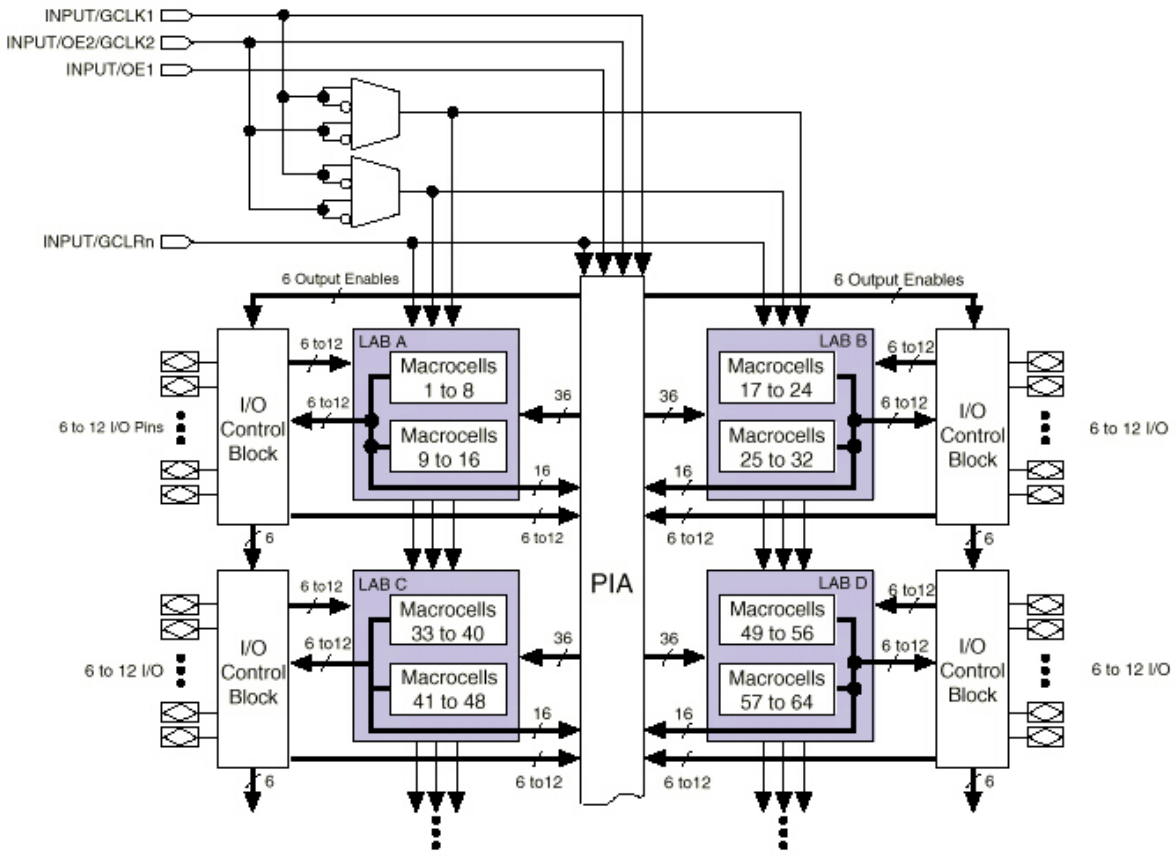


Figura 3.3: Diagrama en bloques del MAX 7000.

LABs utilizando el arreglo de interconexión programable (PIA - Programmable Interconect Array), la cual también puede utilizarse para conducir datos desde ó hacia los pines I/O. Los pines I/O pueden programarse como entradas, salidas, salidas con driver tri-state ó tri-state bi-direccionales. En la Fig.3.3 se muestra un diagrama en bloques del FPLD MAX 7000 en donde pueden observarse las conexiones de los bloques.

En las siguientes secciones se describirá la funcionalidad de los bloques del CPLD de la familia MAX7000.

3.2.1.1. Macro celda (Macrocell - MC)

La unidad fundamental en un dispositivo MAX es la macro celda (MC). En la familia MAX7000 la MC puede ser configurada para un uso combinatorio ó secuencial. En la Fig.3.4 se muestra una MC de la familia MAX 7000S, ésta consiste de tres partes:

- ❖ Un arreglo lógico, que implementa las funciones de lógica combinatoria.
- ❖ Una matriz de selección de términos producto, ésta selecciona los términos producto que toman parte en la implementación de la función lógica.

❖ Un Flip-Flop (F-F) programable de tipo D, JK, T ó SR, que puede ser omitido.

Un arreglo lógico (Logic Array) consiste en una serie de compuertas AND programables seguidas de compuertas OR fijas, este arreglo es conocido como PAL (Programmable Array Logic) y es el utilizado por ejemplo en los PLD PAL22V10. Las entradas a las AND, provienen de los pines de entrada dedicados de la macro celda y de los caminos de realimentación de los I/O, pudiendo ingresar de manera negada o no. El arreglo lógico tiene 5 términos producto. La matriz de selección de términos producto, asigna los términos producto primeramente a las entradas de lógica, las compuertas OR y XOR para ser usados en la implementación de funciones lógicas; o secundariamente a las entradas del registro de la MC, las señales de Preset, Clear, Clock y Clock Enable (Fig.3.4). Solo un término producto por MC puede negarse y realimentarse dentro del arreglo lógico. Este término producto compartido puede conectarse a cualquier término producto dentro del LAB. Por lo tanto cada LAB posee 16 términos productos provenientes de la lógica compartida, como muestra la Fig.3.5.

El Flip-Flop (F-F) de la MC puede programarse para que emule uno de tipo D, JK, T o SR; de ser necesario, puede omitirse permitiendo la operación de funciones de combinatoria sin registro. El clock del F-F puede comandarse de tres modos diferentes:

* El primero por la señal de clock global. Éste modo es el que proporciona menor retardo permitiendo, por ejemplo, la implementación de contadores.

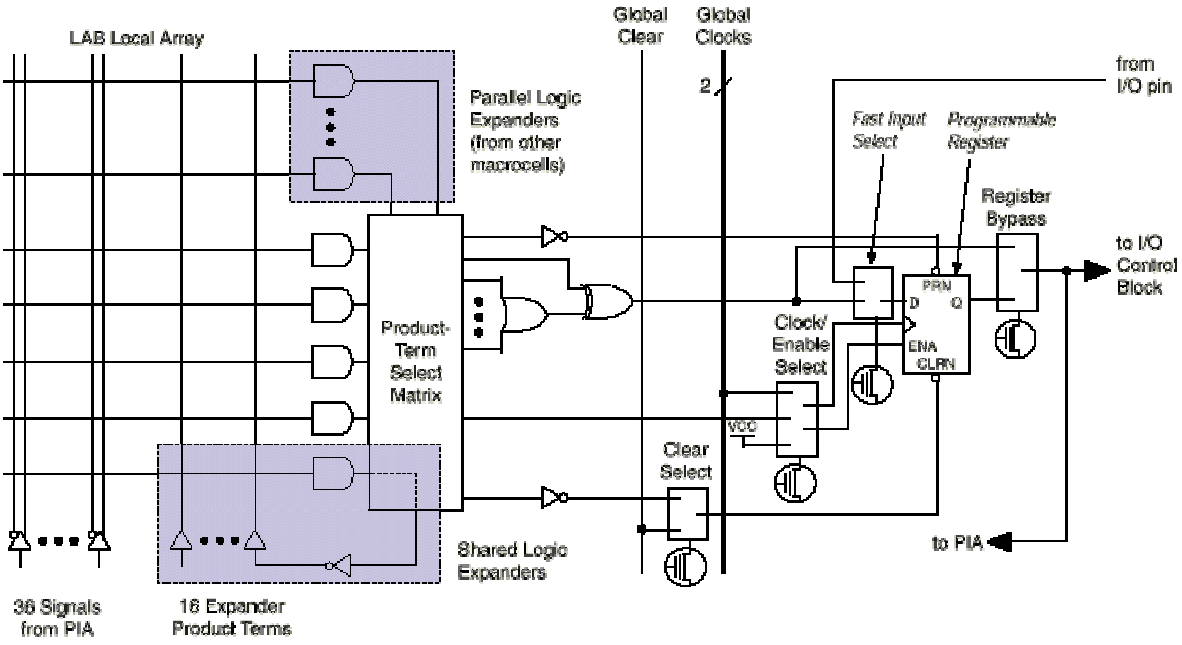


Figura 3.4: Macro celda (MC) de la familia MAX 7000

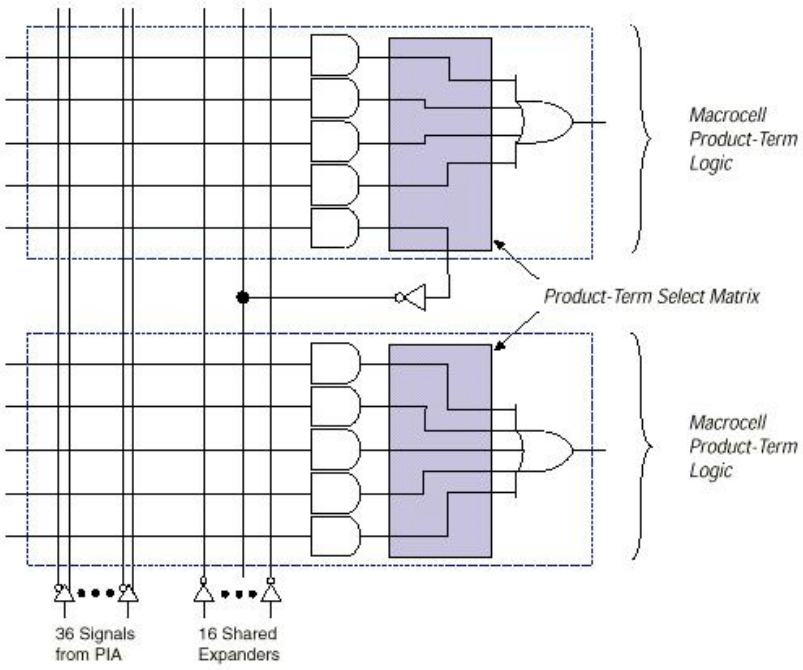


Figura 3.5: Término producto compartido

* El segundo utilizando la señal de clock global habilitándola con la señal de habilitación de clock. Este modo permite la habilitación del clock de cada F-F, manteniendo la perfomance del clock global.

* Finalmente el tercero por medio de un término producto. En este modo el F-F puede ser disparado desde un término producto o desde un I/O pin.

Cada F-F soporta las funciones de Preset y Clear asincrónico a través de términos producto seleccionados por la matriz. También permiten la función de Clear por medio de la señal de Clear Global. La entrada de datos a los F-F puede hacerse directamente desde los I/O pines evitando pasar por la PIA y la lógica combinatoria, lográndose un rápido tiempo de setup (3 ns) y permitiendo el uso de los F-F como registros.

Las funciones lógicas muy complejas, que requieran más de 5 términos producto, pueden implementarse utilizando términos producto compartidos y paralelos (shared and parallel expander), como muestra la Fig.3.4. Los términos producto paralelo son los que no se utilizan en una MC y pueden ser utilizados por MC vecinas, como muestra la Fig.3.6.

3.2.1.2. Bloque de Entrada/Salida (I/O Block)

El bloque de I/O de la familia MAX7000S tiene un buffer tri-state controlado por una de las seis líneas globales de habilitación de salida. Los I/O pin pueden configurarse como entradas, salidas, o bidireccionales. Cuando un I/O pin es configurado como entrada, la MC

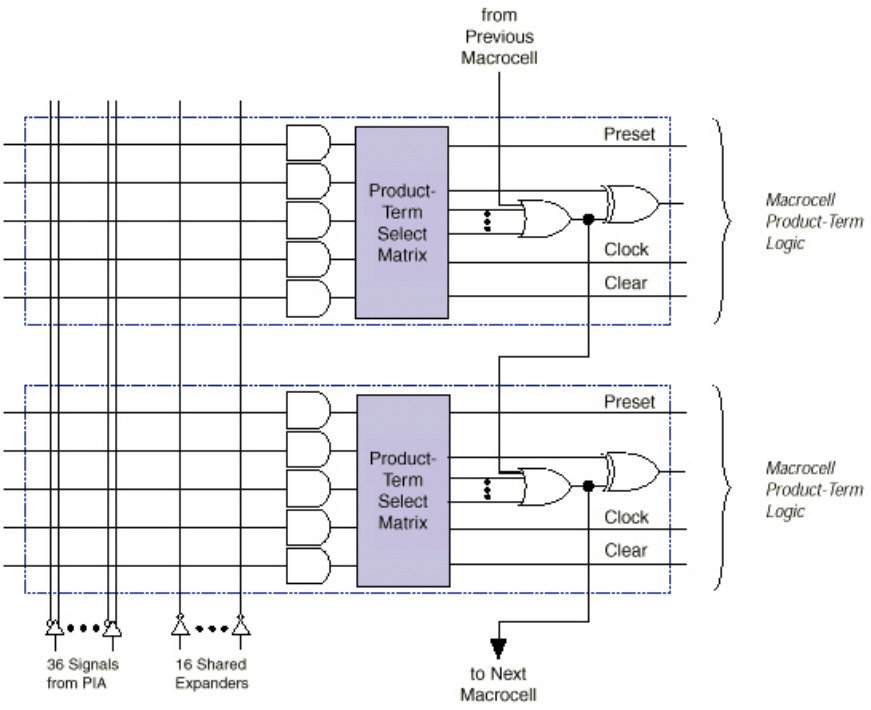


Figura 3.6: Términos producto paralelo.

asociada puede utilizarse como lógica escondida. En la Fig.3.7 se muestra el bloque de control de I/O.

La familia MAX7000S tiene la posibilidad de que cada MC pueda ser programada para la operación en alta velocidad o en bajo consumo. El buffer de salida de cada pin tiene un control del slew rate, el cual puede configurarse para la utilización con bajo ruido o con alta velocidad. La configuración del slew rate para alta velocidad, debe realizarse en circuitos donde la velocidad es crítica y se debe tener la precaución de que el sistema esté debidamente protegido contra ruido. La salida también puede ponerse como colector abierto.

3.2.1.3. LAB (Logic Array Block)

La lógica programable en los FPLD MAX se desarrolla dentro de los bloques de arreglos lógicos LAB. Cada LAB contiene un arreglo de MC, un arreglo de extensiones de términos producto, y un bloque de control de I/O. El número de MC y extensiones varía para cada dispositivo [Alt 98a]. Cada LAB es accesible a través de las líneas del arreglo de interconexión programable (PIA) y de las líneas de entrada, en la Fig.3.3 pueden observarse las conexiones de los LABs.

Las MC son el principal recurso para la implementación de la lógica, y las extensiones de términos producto suplementan la capacidad de cualquier MC. La salida de una MC

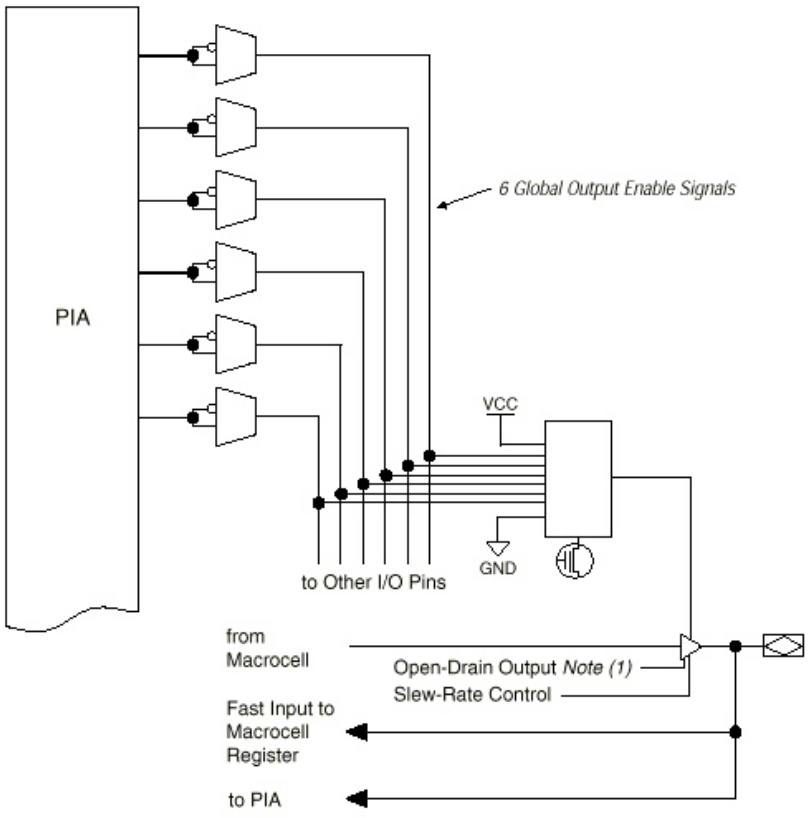


Figura 3.7: Bloque de control de entrada salida para la familia MAX 7000S

alimenta al bloque de I/O, el cual consiste en un grupo de buffer tri-state programables. Cada LAB tiene dos modos de operación del clock, asincrónico y sincrónico. En el modo asincrónico cada clock es manejado por un término producto, por lo tanto puede estar controlado por cualquier pin de entrada o por alguna función lógica. Además cada F-F puede configurarse para ser disparado por un flanco de subida o de bajada. Por otro lado, en el modo sincrónico, el clock puede utilizarse solamente con el pin dedicado de entrada y sólo permite el disparo de los F-F con el flanco de subida.

Los FPLD de la familia MAX tienen una arquitectura expansible y modular permitiendo la integración de más de 10000 compuertas en un solo encapsulado. Estos están basados en una arquitectura matricial de lógica, compuesta de una matriz de LABs, conectados con la PIA. La PIA provee un camino de conexión, con un pequeño y fijo retardo, entre todas las señales internas.

3.2.1.4. Arreglo de interconexión programable (PIA - Programmable Interconnect Array)

La lógica es conducida entre los LABs a través del arreglo de interconexión programable (PIA). La PIA es un bus global programable, que permite la conexión de cualquier fuente de señal a cualquier destino sobre el dispositivo. Todos los pines dedicados, los pines de I/O y las salidas de las MC alimentan la PIA, dejándolos disponibles en todo el dispositivo.

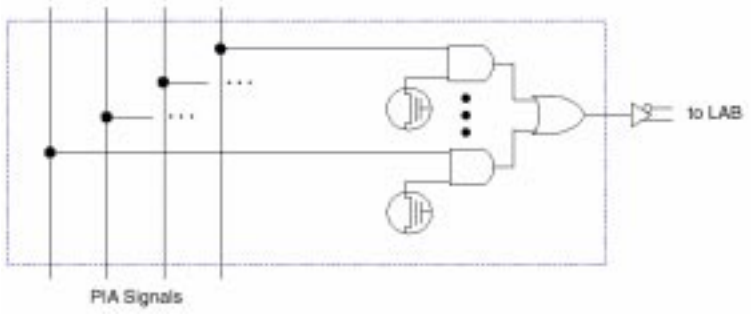


Figura 3.8: Arreglo de interconexión programable (PIA - Programmable Interconnect Array)

Una celda de EEPROM controla una de las dos entradas de una compuerta AND, la cual selecciona una señal de la PIA y la conduce hacia un LAB, como muestra en la Fig.3.8. Sólo se conectan las señales que requiere cada LAB.

Los dispositivos MPGA y FPGA tienen retardos de ruteo variables y dependientes del camino. En cambio la PIA del FPLD MAX tiene un retardo fijo. Esto elimina el skew entre señales y permite al usuario determinar el peor caso de retardo para cualquier diseño.

3.2.2. Altera FLEX 10K - Conceptos generales

La familia FLEX (Flexible Logic Element Matrix) son dispositivos CPLD basados en tablas (Look-Up Tables - LUT), con capacidades que van desde 10.000 a 250.000 compuertas. Los dispositivos FLEX se configuran cargando una memoria estática de acceso aleatorio (SRAM). Por lo tanto la configuración se pierde con el corte de la alimentación. Sin embargo estos dispositivos ofrecen la posibilidad de la conexión de una pequeña memoria serial (PROM) de bajo costo o la conexión a un controlador, para que automáticamente al conectar la fuente de alimentación, se cargue la información de la programación del dispositivo. El tiempo que toma la configuración es menor a 200mseg [Alt 98b].

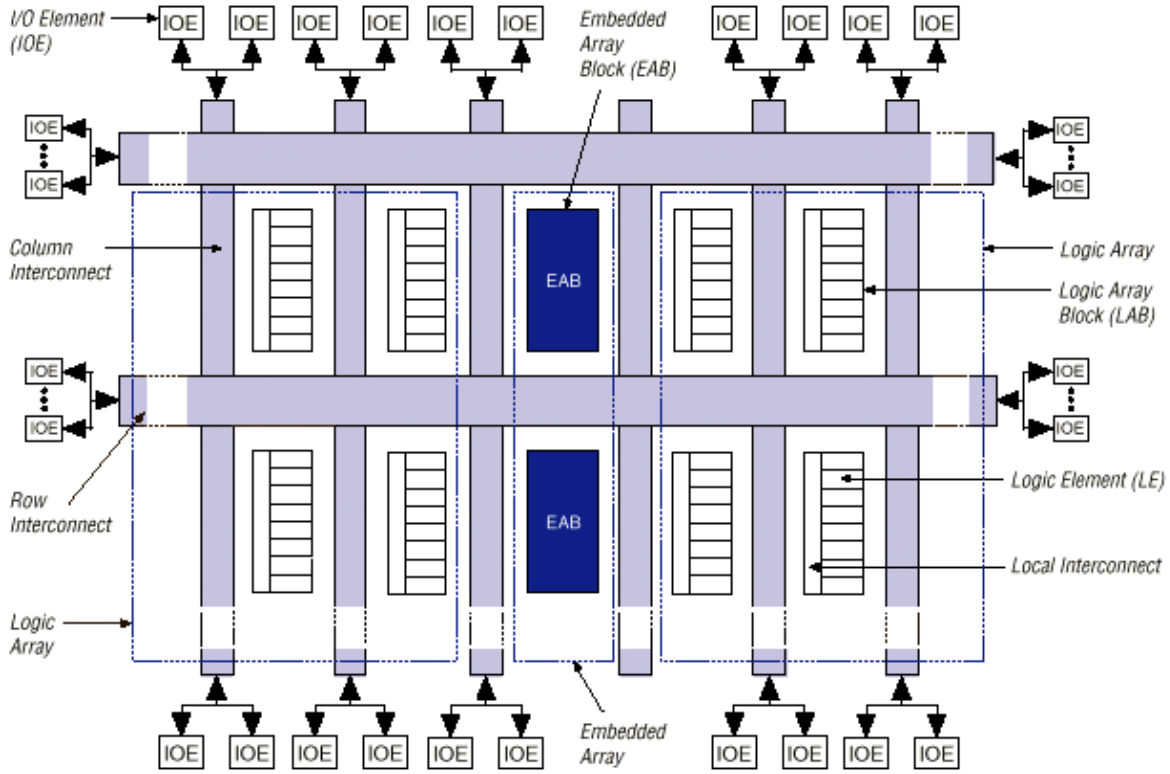


Figura 3.9: Arquitectura de un FLEX 10K.

La arquitectura FLEX incorpora matrices de compactas celdas lógicas llamadas elementos lógicos (LE). Los LE son agrupados de a ocho, creando un bloque de arreglo lógico (LAB). Cada LAB es una estructura independiente con entradas, interconexiones y señales de control comunes. Los LABs están dispuestos entre filas y columnas, como si fueran los elementos de una matriz, ver Fig. 3.9. Los pines de entrada y salida (I/O), soportados por los elementos de entrada y salida (IOE), están localizados al final de las columnas y de las filas de interconexión. Cada IOE esta compuesto de un buffer bidireccional y un Flip-Flop, que puede utilizarse como registro de entrada o salida. La interconexión de señales dentro de los FLEX y con los pines I/O, se realiza por unos canales metálicos continuos, denominados FastTrack. Estos componen las columnas y filas de interconexión, que recorren a lo largo y ancho todo el dispositivo. En la Fig. 3.9 se muestra la arquitectura de la familia FLEX 10K.

Cada dispositivo FLEX 10K tiene incorporado en el silicio áreas dedicadas de un arreglo particular, denominado EAB (Fig. 3.9), los cuales se utilizan para implementar funciones lógicas especiales. Los EAB consisten en un arreglo de memoria rodeado por lógica programable, la cual puede ser fácilmente programada para implementar la función

lógica deseada. Los FLEX 10K también poseen un arreglo de lógica para la implementación de lógica en general.

En las siguientes secciones describiremos las unidades funcionales de los CPLD de la familia FLEX.

3.2.2.1. Elemento lógico (LE)

El elemento lógico (LE) es la unidad lógica más pequeña de la arquitectura FLEX. Cada LE posee una tabla (Look-Up Table - LUT) de 4 entradas, un Flip-Flop programable, un carry chain y un cascade chain como se muestra en la Fig. 3.10.

La LUT (Look-Up Table) es una SRAM de alta velocidad organizada en 16 x 1,

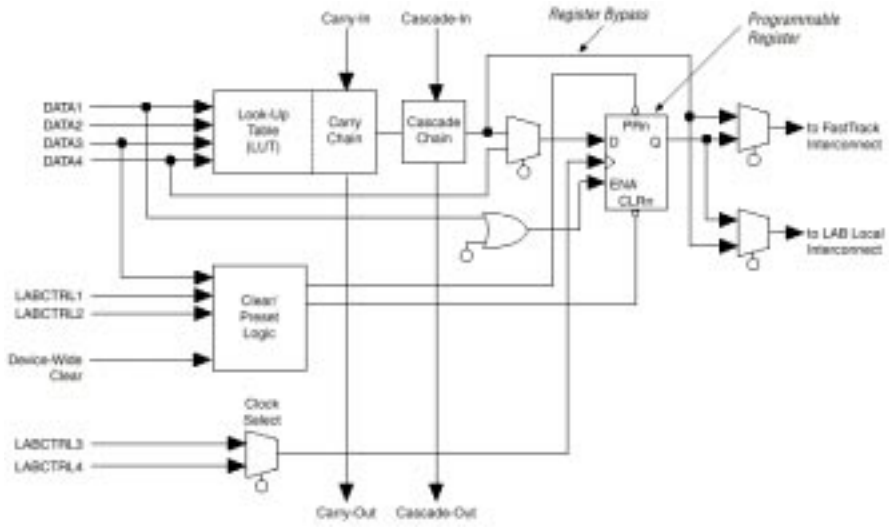
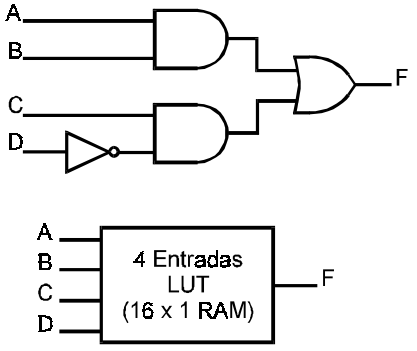


Figura 3.10: Elemento Lógico (LE) de la familia FLEX.



Contenido de la SRAM				
Dirección (Entradas)				Dato (Salida)
A	**B**	**C**	**D**	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Figura 3.11: Ejemplo de utilización de una LUT para implementar funciones lógicas.

pudiendo rápidamente implementar cualquier función lógica Booleana de 4 entradas. La tabla de verdad, que implementa la función lógica deseada, se carga durante la programación del dispositivo. Un ejemplo de cómo trabaja la LUT se muestra en la Fig.3.11, donde las 4 entradas A,B,C,D direccionan la SRAM de 16x1, siendo la salida la línea F. Los datos que se almacenan en la SRAM son los correspondientes a la Tabla de verdad, que representa la función lógica deseada.

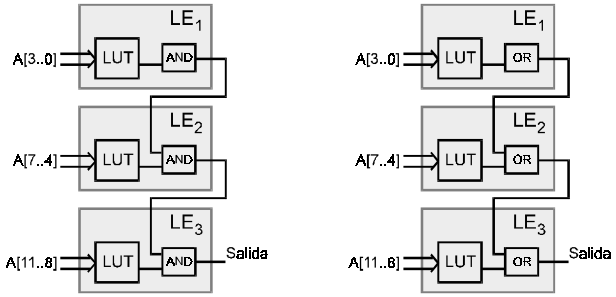
El Flip-Flop programable puede configurarse como D, JK, T o SR. Las señales de control (Clock, Clear y Preset) pueden manejarse desde los pines de entrada dedicados, los pines de I/O de propósitos generales o por señales lógicas internas. Las funciones lógicas combinatorias que no requieren del F-F pueden saltearlo, por lo que la salida de la LUT se conectará directamente a la salida del LE.

El LE tiene dos salidas, una se conecta a la red de interconexión local y la otra a una fila o columna del FastTrack. Las dos salidas pueden controlarse independientemente, o sea que el F-F puede ser utilizado por una función lógica que no pertenezca al mismo LE. A este uso se le conoce con el nombre de Register Packing [Alt 98b].

Para el camino de datos, la arquitectura de los FLEX ofrece dos alternativas que son muy rápidas (retardo muy pequeño) llamadas Carry Chain y Cascade Chain. Estos caminos no utilizan el interconexionado local y se conectan a los LE adyacentes. Se explicará la funcionalidad del Carry Chain y del Cascade Chain con ejemplos.

Carry Chain provee un muy rápido transporte de la función entre LEs (menos que 0.5 nseg en FLEX10K). Esto permite la implementación de funciones lógicas como contadores, sumadores y comparadores de ancho arbitrario. Un sumador completo (full adder) de 4 bits puede implementarse con 4+1=5 LEs, usando Carry Chain como muestra la Fig.3.12. Los LE están divididos en dos partes: una implementa la suma de dos bits usando las señales de entrada y el Carry-in. La otra parte implementa el carry de salida, que se conecta directamente a la señal de carry-in del LE adyacente. Y así hasta que termina la implementación de los 4 bits.

Utilizando Cascade Chain se pueden implementar funciones lógicas de muchas variables (ancho fan-in). En la síntesis de una función lógica de muchas variables, un conjunto de LUT adyacentes implementan en paralelo una parte cada una, los resultados intermedios se conectan a través del cascade chain. El cascade chain puede utilizar lógica AND u OR (usando la inversión de De Morgan) para conectar las salidas de los LE adyacentes. Por lo



a) *b)*

Figura 3.13: Utilización de Cascade Chain para implementar una función de 11 variables. a) utilizando lógica AND, b) utilizando lógica OR.

tanto, cada LE provee 4 entradas y adiciona un retardo menor a 1 nseg. En la Fig.3.13 se muestra la implementación de una función de 11 variables.

Los LEs poseen cuatro modos de operación siendo: Modo Normal (Normal Mode), Modo aritmético (Arithmetic Mode), Modo contador Up/Down (Up/Down Counter Mode) y Modo Contador Borrable (Clearable Counter mode). En cada modo de operación siete de las diez señales de entrada al LE se utilizan para implementar la función lógica deseada, las tres señales restantes se utilizan para el control del registro.

El Modo Normal de operación es adecuado para aplicaciones generales de lógica y funciones de decodificación con muchas entradas, que pueden tomar ventajas haciendo uso de la cadena en cascada (Cascade chain).

El Modo aritmético dispone dos LUT de 3 entradas, por lo tanto es ideal para implementar sumadores, acumuladores, y comparadores. Este modo también soporta la cadena en cascada (Cascade Chain).

El Modo Contador Ascendente/Descendente ofrece las opciones de la señal de habilitación, control sincrónico para la cuenta ascendente/descendente y la carga de datos. Igual que en el modo aritmético utiliza 2 LUT de 3 entradas, una genera la cuenta de los

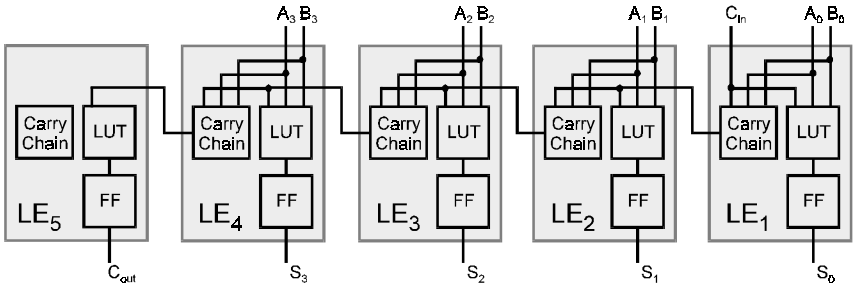


Figura 3.12: Utilización de Carry Chain para implementar un sumador completo de 4 bits.

datos del contador y la otra el bit de carry. La carga de datos sincrónica se realiza a través de un multiplexer de 2 a 1. La carga asincrónica puede realizarse utilizando las señales de control clear o preset del registro.

El Modo Contador Borrable es similar al Modo Contador Ascendente/Descendente, pero soporta el clear sincrónico y no tiene el control Up/Down.

Las señales de control Clear y Preset del registro, son manejadas por una lógica que tiene como entradas las señales de entrada del LE: DATA3, LABCTRL1 y LABCTRL2 (Fig.3.10). La lógica de Clear y Preset tiene seis modos de operación [Alt 98b], el cual es seleccionado por el software en el momento de la compilación.

3.2.2.2. Bloque de arreglos lógicos (Logic Array Block - LAB)

Un LAB está compuesto de 8 LEs, las señales asociadas de Carry y Cascade chain, las señales de control del LAB, y la interconexión local del LAB, como muestra la Fig. 3.14. Las 4 señales de control que provee cada LAB pueden utilizarse en los 8 LE. Dos pueden utilizarse para manejar el clock y las otras dos las señales de Clear y Preset de los LEs.

3.2.2.3. Bloque integrado de arreglos (Embedded Array Block - EAB)

Este bloque es la diferencia que existe entre la arquitectura de la familia FLEX 8000 y FLEX 10K. Un dispositivo FLEX 10K puede tener hasta 12 EABs. Los EAB pueden utilizarse de manera independiente o combinada, para implementar funciones complejas. Si el

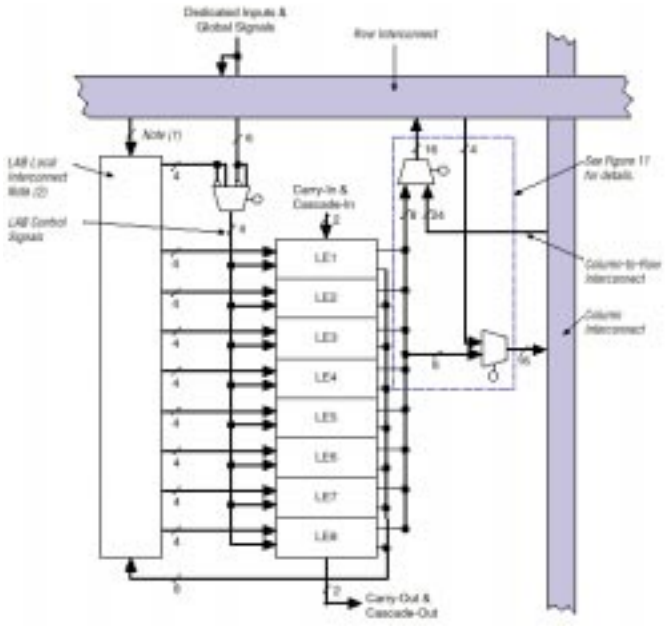


Figura 3.14: Estructura de un LAB para la familia Flex.

EAB se utiliza para crear funciones de memoria, puede proveer hasta 2048 bits; los cuales pueden utilizarse para crear RAM, ROM o FIFO de una o dos compuertas. Cuando el EAB se utiliza para implementar funciones lógicas, cada uno puede proveer el equivalente de 100 a 600 compuertas lógicas.

El EAB es un bloque flexible de RAM con registros en los puertos de entrada y salida. Su flexibilidad permite implementar memorias de los siguientes tamaños: 2048 x 1, 1024 x 2, 512 x 4 ó 256 x 8. Esta flexibilidad los hace apropiados también para implementar funciones lógicas muy complejas, como ser multiplicadores, circuitos de corrección de error, etc, siendo utilizados como LUT. Por ejemplo, un EAB puede implementar un multiplicador de 4 x 4 con 8 entradas y 8 salidas proveyendo alta perfomance con un rápido y predecible tiempo de acceso. EABs dedicados son simples de utilizar, eliminando las preocupaciones de Timing y Ruteo, y proporcionando un retardo predecible.

Los EABs pueden utilizarse para implementar tanto RAM sincrónicas como asincrónicas. En el caso de las RAM sincrónicas, el EAB genera su propia señal de habilitación de escritura (WE), la cual está en timing con la señal de clock global; por lo tanto un circuito que utilice esta RAM sólo deberá cumplir con las especificaciones de los tiempos de setup y hold del clock global. Cuando los EAB se utilizan en RAM asincrónicas, un circuito adicional deberá manejar la señal WE, asegurando que las señales de dirección y datos cumplen las especificaciones de tiempo de setup y hold relativas a la señal WE. Es por

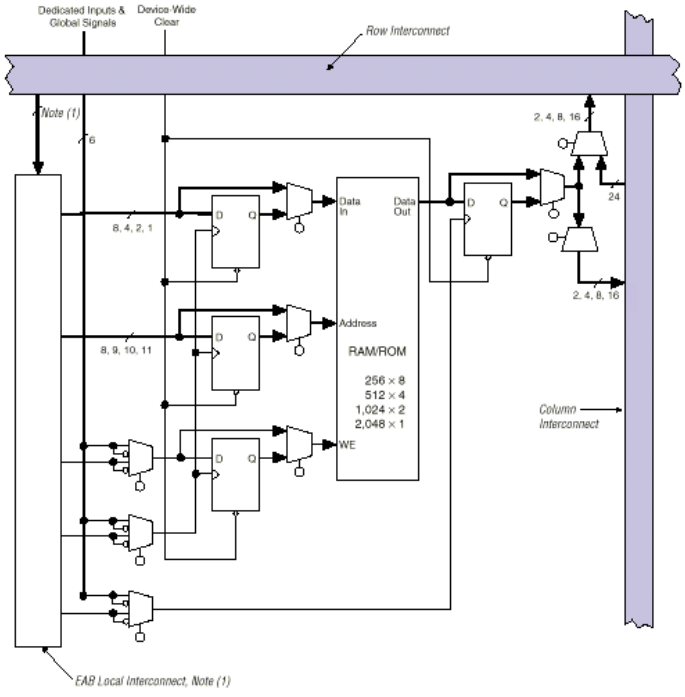


Figura 3.15: Arquitectura de un EAB de la familia FLEX 10K.

eso que se recomienda, en la medida de lo posible, utilizar RAM sincrónicas. Los EAB pueden combinarse en serie o paralelo para crear memorias más grandes.

Las señales globales de la familia Flex 10K, los pines de clock dedicados, y las interconexiones locales del EAB pueden manejar las señales de clock del EAB. Debido a que los LE se pueden conectar al bloque de interconexiones locales del EAB, éstos pueden manejar las señales de habilitación de escritura (WE) y las señales de clock del EAB. En la Fig. 3.15 se muestra la arquitectura de un EAB.

En contraste con los LE, que implementan funciones lógicas muy simples en un solo elemento y funciones más complejas en estructuras de mas niveles, el EAB puede implementar funciones complejas, incluyendo aquellas de un gran fan-in, en un solo nivel de lógica resultando en una utilización más eficiente del dispositivo y una alta perfomance. Una misma función, implementada en un EAB ocupa menos área en dispositivo, tiene un retardo muy pequeño, y opera más rápido que si estuviera implementada usando LE. Las funciones lógicas son implementadas en el EAB durante el proceso de configuración del dispositivo, utilizando un patrón de sólo lectura, creándose una gran LUT. El patrón puede cambiarse y reconfigurarse, para cambiar la función lógica, durante la operación del dispositivo [Sal 00]. Cuando una función lógica se implementa utilizando EAB, los datos de entrada manejan las señales de dirección del EAB buscando el resultado en la tabla y conduciéndolo a la salida. El uso de LUT para encontrar el resultado de una función, es más rápido que usar algoritmos implementados en lógica general y en LEs.

El EAB tiene ventajas sobre las FPGAs que tienen bloques pequeños de RAM distribuidos en el dispositivo. Estas FPGA tienen retardos que son menos predecibles cuando aumenta el tamaño de la RAM a ser implementada. Además, los pequeños bloques de RAM en las FPGAs son propensos a tener problemas de ruteo, debido a que grandes RAM necesitan la conexión de varios pequeños bloques de RAM.

3.2.2.4. Filas y columnas de Interconexión (FastTrack)

Las conexiones entre los LEs y los dispositivos de entrada/salidas (I/O), son provistas por un mecanismo de interconexión denominado FastTrack. Éste está representado por canales Horizontales y Verticales continuos que atraviesan todo el dispositivo. Estos canales están compuestos por una serie de conductores. Esta estructura de ruteo global proporciona una perfomance predecible, aún en diseños complejos. En cambio, los ruteo por segmentos

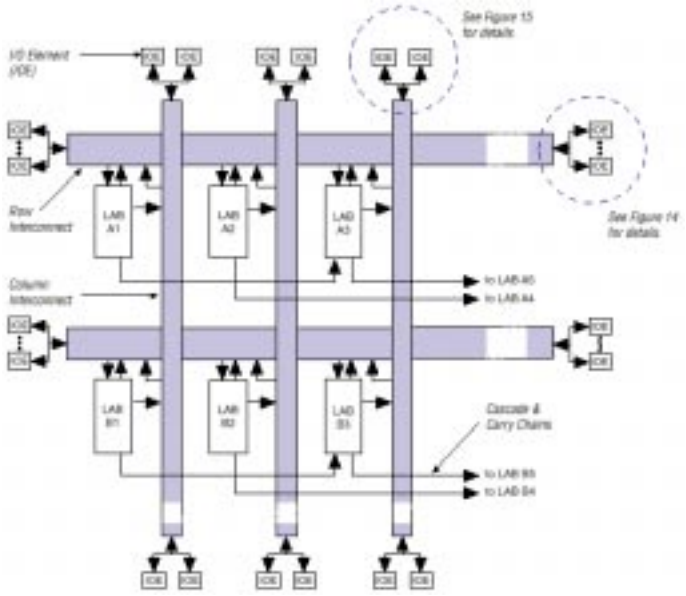


Figura 3.16: Recursos de interconexión en dispositivos FLEX 10K.

en las FPGAs requieren de matrices de conmutación para conectar un número variable de caminos de ruteo, incrementando los retardos y reduciendo la perfomance.

Los LABs están acomodados dentro del dispositivo como si fueran los elementos de una matriz, estando separados entre sí por los canales de horizontales y verticales. Cada canal horizontal (fila) tiene asociado una conexión dedicada, la cual conduce señales desde y hacia el LAB. Los canales horizontales también pueden conducir pines de I/O ó alimentar otros LABs en el dispositivo. Los canales verticales (columnas) pueden manejar pines de I/O e interconectar canales horizontales. La Fig. 3.16 muestra las interconexiones de los LABs, con las filas, las columnas, las interconexiones locales, y las conexiones de carry y cascade chain.

Para mejorar el ruteo los canales de interconexión horizontales constan de una combinación de canales de longitud completa y canales con la mitad de la longitud. Entonces dos LABs vecinos pueden conectarse utilizando un canal horizontal de media longitud, dejando disponible la otra mitad. Esta flexibilidad en el ruteo permite que el mismo pueda utilizarse más eficientemente incrementando los recursos de interconexión.

3.2.2.5. Elementos de Entrada/Salida (IOE)

La arquitectura de los elementos de Entrada/Salida (IOE) se presenta en la Fig. 3.17. Los IOEs se localizan al final de los canales de interconexión horizontales y verticales. Los pines de Entrada/Salida (I/O) pueden usarse como entradas, salidas ó bidireccionales. Cada pin I/O tiene un registro que puede utilizarse para registrar la entrada o la salida, en

operaciones que requieran una alta perfomance (rápido tiempo de setup ó corto retardo entre clock y salida). El IOE tiene un buffer de salida con un slew rate ajustable, que puede configurarse para aplicaciones de bajo ruido o alta velocidad.

Un slew rate rápido (pendiente de bajada o subida muy pronunciada) debe utilizarse en aplicaciones donde la velocidad sea crítica y el sistema esté adecuadamente protegido contra el ruido. Un slew rate lento disminuye el ruido en el sistema y adiciona un retardo máximo de 4.5 nseg [Alt 98b]. Este ajuste puede realizarse para cada pin de manera independientemente. También puede especificarse a cada pin de salida como Open drain.

Las señales de control: Clock, habilitación de Clock, Clear, y habilitación de la salida (OE), son provistas al IOE por una red de señales de control de I/O denominado Bus de Control Periférico. Estas señales son manejadas por pines dedicados o por lógica interna. El Bus de Control Periférico utiliza drivers de alta velocidad para conducir las señales por la periferia del dispositivo.

3.2.3. Xilinx XC4000 - Conceptos generales

La familia de FPGAs XC4000 de la compañía Xilinx, dispone de una regular, flexible y programable arquitectura de Bloques Lógicos Configurables (CLB). Los CLB están interconectados jerárquicamente por una versátil fuente de ruteo y rodeados periféricamente

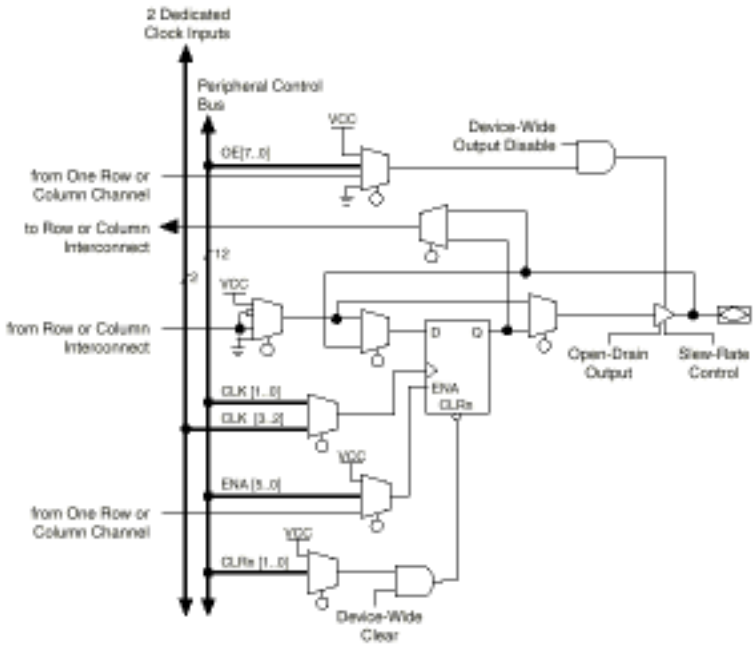


Figura 3.17: Arquitectura de un elemento de Entrada/Salida (IOE) de la familia FLEX 10K

por Bloques de Entrada/Salida (IOB). Pueden prepararse dispositivos con mascaras, adaptándose a las necesidades del cliente; o sea se programan, cargando los datos de configuración en celdas de memoria estáticas (SRAM).

Los bloques constructivos básicos de la familia XC4000 incluyen:

- Look-up Tables (LUT) para la implementación de funciones lógicas.
- Punto Programable de Interconexión (PIP), que es un transistor de paso controlado por una celda de memoria. El PIP es la unidad básica del mecanismo configurable de interconexión. El transistor de paso introduce resistencia en el camino de la interconexión, por lo tanto adiciona retardo.

Un multiplexer es un caso especial de la estructura de ruteo direccional, controlado por una celda de memoria. Los multiplexer pueden ser de cualquier ancho, en donde los multiplexer más anchos requieren más bit de configuración (celdas de memoria).

La FPGA puede programarse por si sola (Master Mode), leyendo los datos desde una memoria serie o paralela. También puede programarse desde otro dispositivo (Slave Mode), esto permite la reconfiguración de la FPGA en el sistema, ofreciéndole al diseñador un grado de libertad adicional.

Los CLB aportan elementos funcionales para que el usuario pueda construir la lógica. Los IOB proveen la interfaz entre los pines del encapsulado y las líneas de señal internas. Los recursos de interconexión programable, proveen los caminos de ruteo para conectar las salidas y entradas de los CLB e IOB a las correspondientes redes. La configuración final del cliente en el dispositivo, se realiza programando las celdas de memoria estática interna que determinan la función lógica y las interconexiones en el arreglo de celdas lógicas (Logic Cell Array - LCA). La familia XC4000 de Xilinx puede utilizarse en diseños que requieren cambios dinámicos en el hardware o donde el hardware debe adaptarse a diferentes aplicaciones del usuario.

En las siguientes secciones describiremos las principales unidades funcionales de la FPGA XC4000 de Xilinx.

3.2.3.1. Bloque lógico configurable (CLB)

La arquitectura de un CLB, que se muestra en la Fig. 3.18, está compuesta por dos Flip-Flop y dos generadores de funciones independientes de cuatro entradas. Los dos generadores (F y G), pueden implementar cualquier función Booleana de 4 variables.

Los generadores de funciones se implementan con LUT. Un tercer generador de funciones (H), puede implementar cualquier función Booleana de 3 entradas, dos de las cuales provienen de los generadores F y G y la tercera entrada viene de afuera del CLB. Por lo tanto, haciendo uso de la LUT H pueden implementarse funciones lógicas de 9 variables en un solo CLB. El CLB posee dos FF tipo D con clock disparado por flanco, donde las señales de entrada de clock (K) y clock enable (EC) son comunes a ambos. Otra entrada común a ambos es la de Set/Reset (S/R), que puede programarse independientemente como Set o Reset asincrónico. Esta entrada también puede anularse de forma independiente para cada Flip-Flop. La entrada de datos al F-F puede ser F, G, H o la entrada (DIN) externa al CLB.

Cada CLB incluye un circuito de carry de alta velocidad, que puede activarse por configuración. Los generadores de funciones de 4 entradas pueden configurarse como un sumador de 2 bits con el circuito oculto de carry, siendo de esta manera muy rápido y eficiente. La posibilidad de un rápido carry abre las puertas a aplicaciones con operaciones aritméticas, como por ejemplo sumas de alta velocidad en procesadores digitales de señal.

Un modo opcional para cada CLB es que los generadores de funciones (F y G) puedan utilizarse como memorias de Lectura/Escritura organizadas como 16 x 2 ó 32 x 1. Las

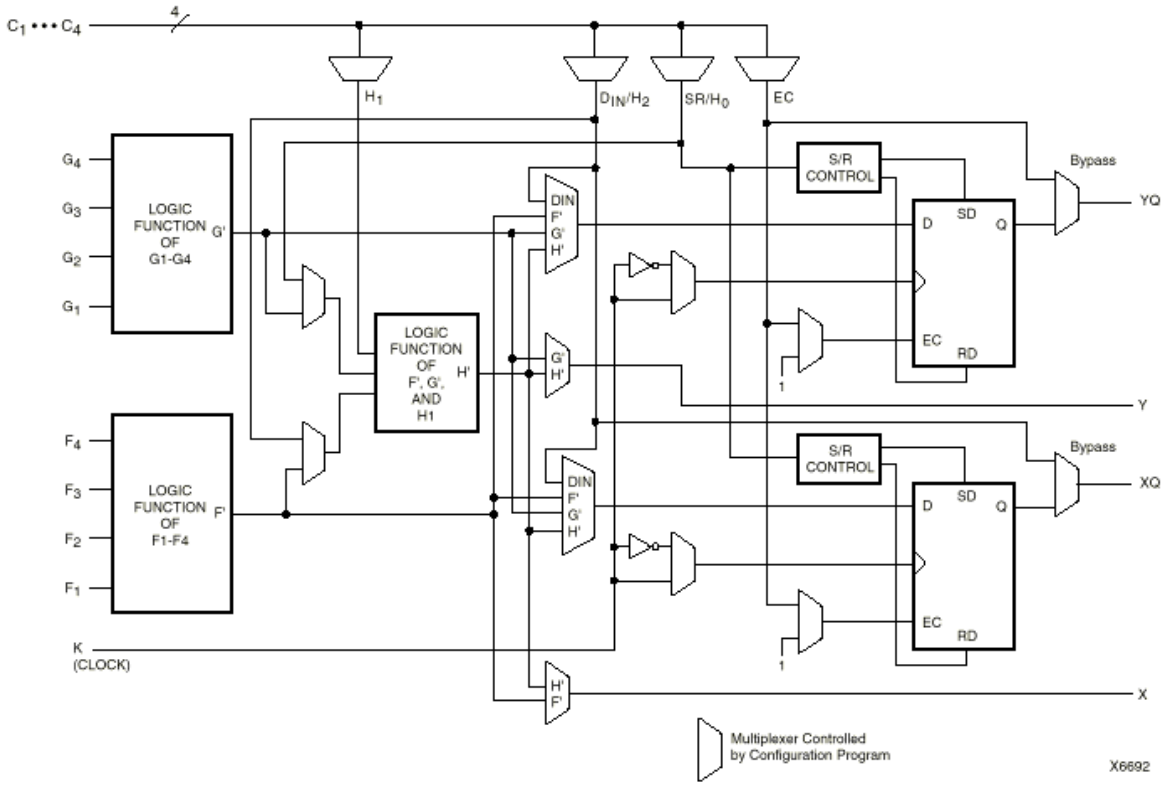


Figura 3.18: Arquitectura de un bloque lógico configurable (CLB) de la familia XC4000

entradas de los generadores de funciones son utilizadas como líneas de dirección, y entradas adicionales al CLB actúan como Write, Enable, y Dato de entrada. La lectura de la memoria se realiza de la misma manera que en la implementación de una función.

3.2.3.2. Bloque de Entrada/Salida (IOB)

El IOB programable por el usuario hace de interfaz entre los pines del encapsulado del dispositivo y la lógica interna del mismo, como se muestra en la Fig. 3.19. Cada IOB controla un solo pin del dispositivo. Dos líneas, denominadas I1 y I2, se encargan de llevar al interior del dispositivo las señales externas. Éstas pueden ser mantenidas en un registro de entrada, que puede activarse por flanco o nivel. La señal de salida puede ser negada o no y puede pasar directamente al pin de salida o quedar almacenada en un flip-flop activado por flanco. Opcionalmente, puede utilizarse una señal de habilitación de salida para poner al buffer de salida en estado de alta impedancia, implementando salida tri-state o bidireccional.

Hay otras opciones en el IOB, tales como resistores pull-up y pull-down programables, señales de clock de entrada y salida separados, y señales globales de Set/Reset.

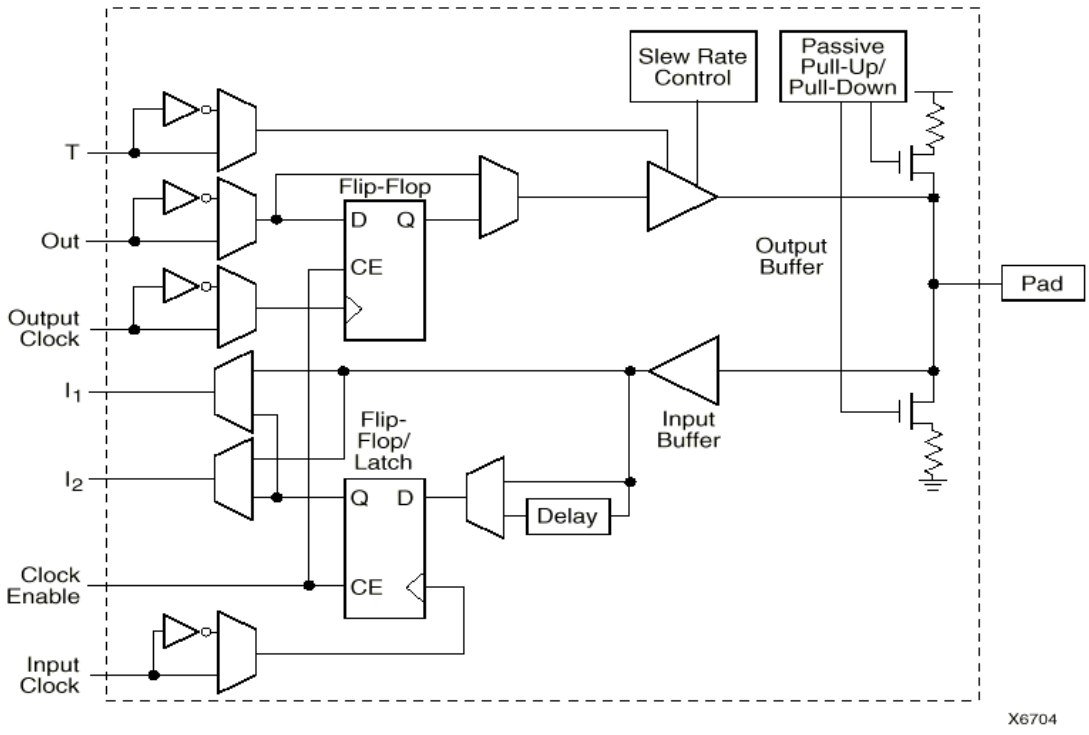


Figura 3.19: Arquitectura del bloque de entrada/salida (IOB) de la familia XC4000

3.2.3.3. Mecanismo de interconexión programable

Todas las conexiones internas están compuestas de segmentos de metal, con puntos de llaves programables para implementar el ruteo deseado. El número de canales de ruteo es proporcional al tamaño del arreglo, y se incrementa con el aumento del arreglo. Las entradas y salidas del CLB, están en los cuatro lados del bloque como muestra la Fig. 3.20. Hay tres tipos de interconexiones distinguibles por la longitud relativa de sus segmentos: líneas de una longitud, líneas de longitud doble y líneas largas.

Las líneas de una longitud son una grilla de líneas horizontales y verticales que unen las matrices de llaves entre cada bloque. Cada matriz de llaves consiste en transistores programables, utilizados para establecer las conexiones (Fig.3.21). Por ejemplo, una señal que entra por el lado derecho de la matriz de llaves puede rutearse a otra línea de una longitud en el lado de arriba, abajo o izquierdo o en cualquier combinación si se requieren múltiples ramas. Las líneas de una longitud normalmente se utilizan para conducir señales dentro de áreas localizadas y proveer ramas para redes con fanout mayores a uno.

Las líneas de longitud doble consisten de una grilla de segmentos metal del doble de largo que las líneas de una longitud. Estas están agrupadas en pares con cada matriz de llaves (Fig. 3.20) y no pueden conectarse a otras líneas.

Las líneas largas forman una grilla de segmentos de metal que recorren todo el ancho y alto del arreglo, como se ve en la Fig. 3.20. Líneas largas adicionales pueden ser manejadas por buffer especiales globales, diseñados para distribuir clock y otras señales de control con un gran fanout por todas partes del arreglo con mínimo skew. Seis de las líneas largas en cada canal son de propósitos generales con gran fanout y alta velocidad. Las entradas al CLB

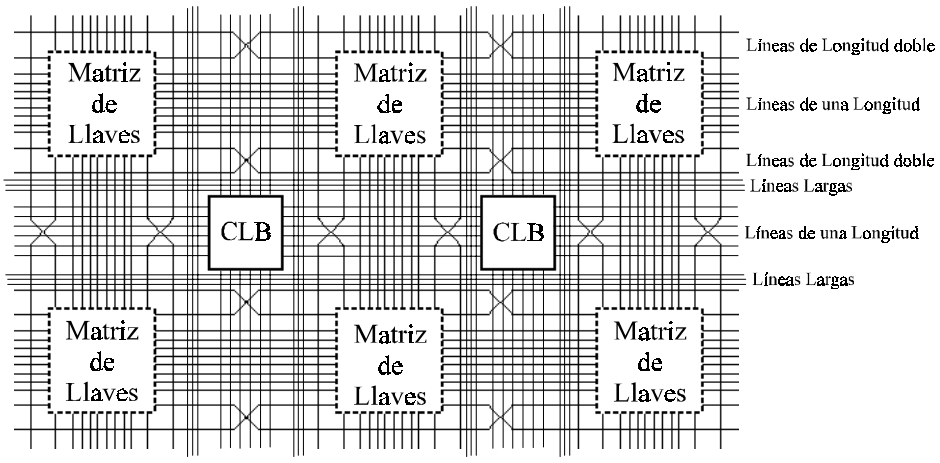


Figura 3.20: Interconexiones con líneas de una longitud, dobles y largas

pueden manejarse desde un subconjunto de líneas largas adyacentes. Las salidas del CLB son dirigidas a las líneas largas por un camino de buffer de alta impedancia o a las líneas de una longitud. La comunicación entre líneas largas y líneas de una longitud está controlada por los puntos de interconexión programables en la intersección de las líneas.

Un par de buffer de alta impedancia, asociados con cada CLB en el arreglo, pueden utilizarse para conducir señales hacia la línea larga horizontal más cercana por encima o debajo del bloque. La entrada de los buffer de alta impedancia pueden manejarse desde cualquier señal de salida del CLB vecino (X, Y, XQ, YQ) o desde una próxima línea de una longitud. Otro buffer de alta impedancia está localizado cerca de cada IOB, a lo largo de los bordes izquierdo y derecho del arreglo. Estos buffer pueden utilizarse para implementar multiplexer o buces bidireccionales sobre las líneas largas horizontales.

La familia XC4000 tiene miembros con diferentes cantidades de cableados en los distintos tamaños. Para el arreglo de CLB desde 14 x 14 a 20 x 20, cada canal de cableado incluye ocho líneas de una longitud, cuatro de longitud doble, seis de longitud larga y cuatro líneas globales. La distribución de estas líneas se originó analizando las necesidades de

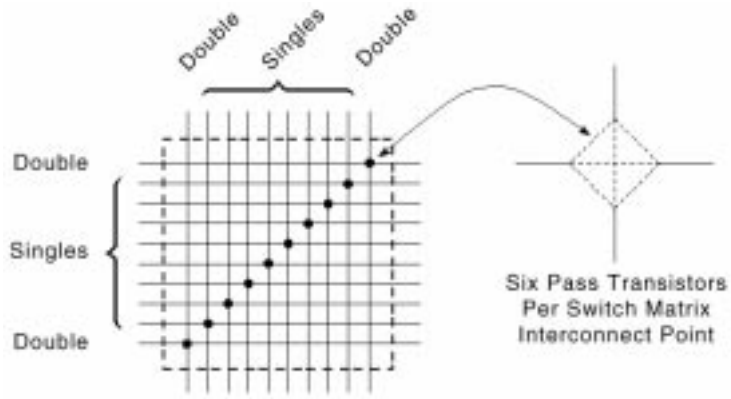


Figura 3.21: matriz de interconexión programable (PSM)

cableado de muchos diseños existentes.

3.2.3.4. Configuración del dispositivo

La configuración es un proceso de carga del diseño especifico, programando los datos dentro del LCA para definir la funcionalidad y las interconexiones de los bloques internos. O sea que es como cargar los registros de control de un chip programable. La XC4000 utiliza

alrededor de 350 bits de datos de configuración por CLB y sus interconexiones asociadas. Cada bit define el estado de una celda SRAM que controla el bit de una LUT, la entrada de un multiplexer, o el transistor que realiza la interconexión.

La XC4000 tiene seis modos de configuración, que pueden seleccionarse por un código de tres bit. Los modos son similares a los de Altera: tres son modos Master de auto carga, dos son modos periféricos y uno es modo esclavo serial.

3.2.4. *Atmel AT40K - Conceptos generales*

La familia AT40K de la compañía ATMEL es una FPGA basada en SRAM. El rango en tamaño equivalente a compuertas lógicas va desde 5000 a 50000 y soporta diseños en 3V y 5V. Está diseñada para implementaciones rápidas de alta perfomance y gran cantidad de compuertas. Algunas de sus extraordinarias características son sistemas de velocidades hasta 100MHz, arreglo de multiplicadores de 50MHz, SRAM de alta velocidad, y capacidad interna de alta impedancia para cada celda lógica.

Los dispositivos AT40K pueden utilizarse como co-procesadores para diseños de alta velocidad, implementando una gran variedad de funciones aritméticas. Éstas incluyen filtros adaptivos de respuesta finita al impulso (FIR), transformadas rápidas de Fourier (FFT), transformadas discretas de coseno (DCT) que se requieren para la compresión y descompresión de video, convolución y otras aplicaciones de multimedia. La familia AT40K tiene una capacidad de compuertas lógicas menor a las correspondientes familias de Altera y Xilinx, pero posee características que no se encuentran en las familias de Altera y Xilinx.

La familia AT40K está organizada en un arreglo simétrico (matriz) de celdas idénticas, como muestra la Fig. 3.22. El arreglo es continuo desde un borde al otro, excepto por un bus que se repite espaciosamente cada cuatro celdas, dividiendo al dispositivo en áreas de celdas de 4 x 4 llamadas sectores. En la esquina inferior derecha de cada sector hay una SRAM de 32 x 4 que es accesible por los buses adyacentes. La SRAM puede configurarse como una RAM de doble o simple compuerta, con operación sincrónica o asincrónica.

3.2.4.1. Celda lógica

La celda lógica tiene dos LUT de tres entradas, pudiendo implementar cualquier función lógica de tres variables. La salida de cada LUT puede conectarse directamente a las

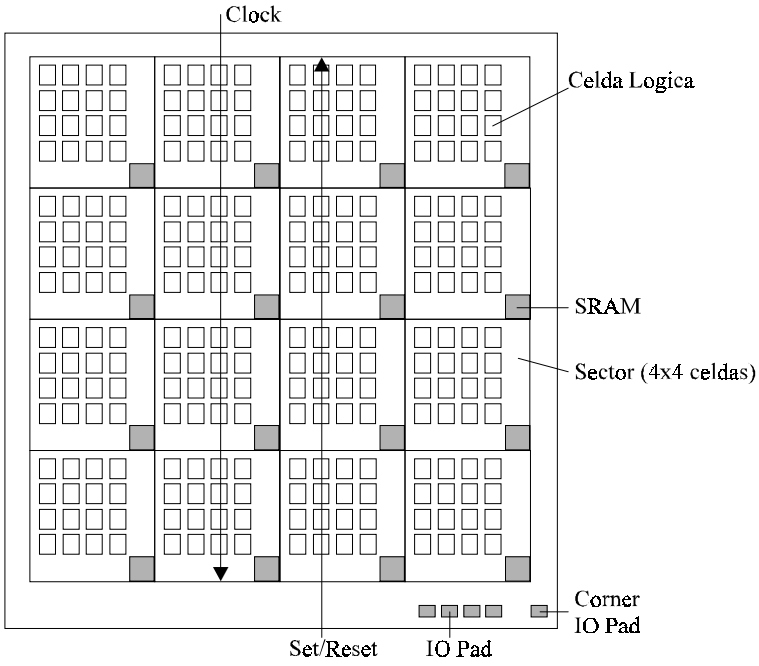


Figura 3.22: Organización general de la familia AT40K de Atmel

celdas vecinas o registrarse usando un flip-flop tipo D. Cada celda lógica contiene solo un flip-flop de tipo D. La organización de una celda lógica se muestra en la Fig. 3.23

Las celdas lógicas pueden configurarse en varios modos de operación especializados, encontrados en la mayoría de las áreas de aplicación de sistemas digitales:

- Modo de Síntesis, combina ambas LUT en una sola de 16 x1 permitiendo implementar funciones lógicas de cuatro variables. La salida puede ingresar al FF.
- Modo Tri-state/Mux, permite la implementación de multiplexers combinando una LUT con un buffer de alta impedancia (Tri-state).
- Modo Aritmético, las dos LUT se utilizan para implementar funciones lógicas de tres entradas, por Ej: suma y carry. Una LUT puede registrarse.
- Modo DSP/Multiplicador, con el agregado de una compuerta AND, dos LUT pueden calcular eficientemente el producto y carry de una multiplicación. Esto puede ser eficientemente utilizado para implementar elementos de un filtro FIR o unidades de multiplicación acumulativas.
- Modo contador, una celda lógica puede implementar completamente un contador de un estado con carry, usando caminos de realimentación interna y un F-F.

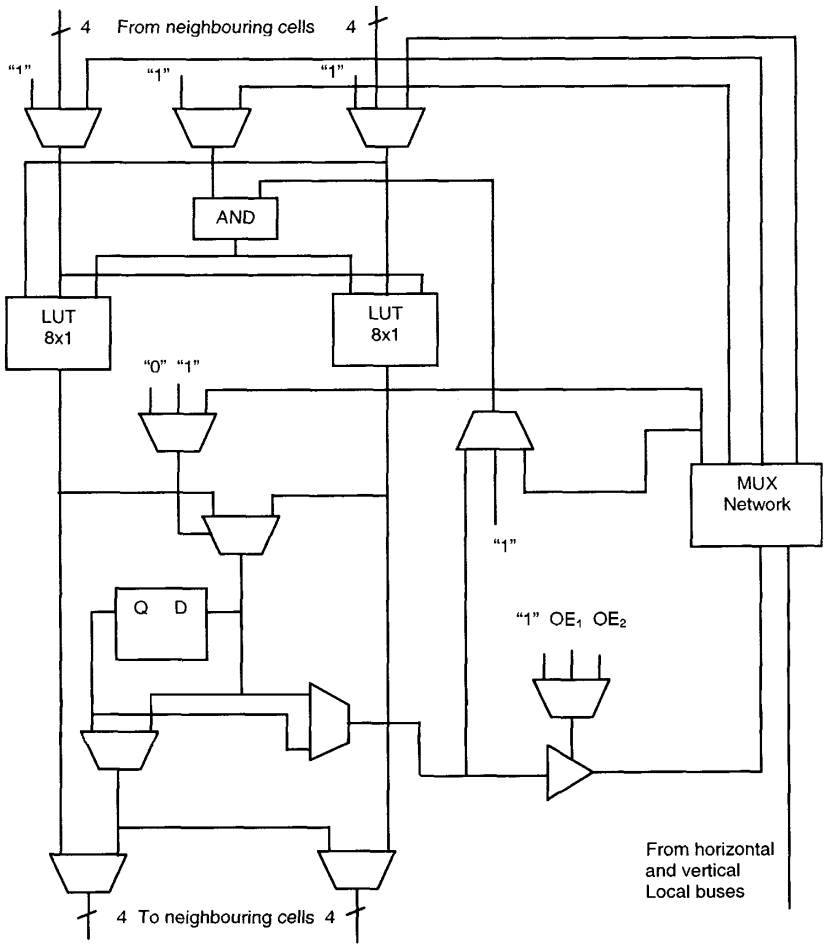


Figura 3.23: Arquitectura de la Celda Lógica de la familia AT40K de Atmel

3.2.4.2. Bloque de memoria

La familia AT40K incluye bloques de SRAM, que pueden utilizarse sin perder los recursos que implementan lógica. Estos bloques permiten la creación de múltiples e independientes funciones RAM, pudiendo ser sincrónicas o asincrónicas, de compuerta simple o doble. Ellas están construidas utilizando las celdas de AT40K SRAM, llamadas celdas de RAM libre (freeRAM) que se muestran en la Fig. 3.24.

3.2.4.3. Reconfiguración dinámica

La familia AT40K es capaz de implementar una reconfiguración lógica dinámica total o parcial sin pérdida de datos, en sistemas de lógica adaptiva. Sólo aquellas partes del sistema que están activas en un determinado tiempo son implementadas en la FPGA, mientras que las partes inactivas se almacenan externamente en la memoria de configuración. Así cuando se requiera una función lógica nueva, ésta puede descargarse en el cache de lógica sin pérdida

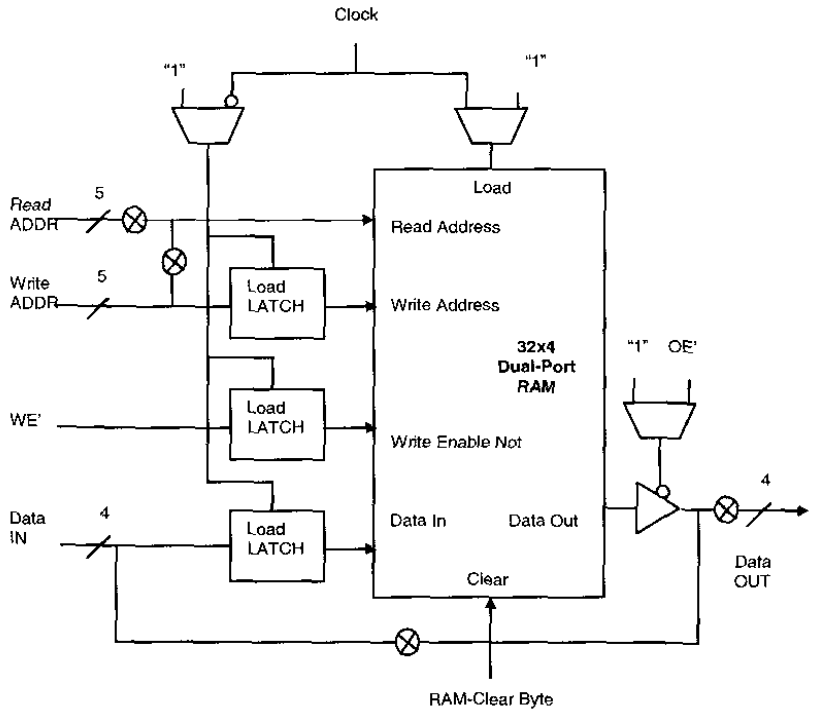


Figura 3.24: Celda de RAM libre de la familia AT40K de Atmel

de los datos que están allí y sin interrumpir la operación del resto del chip, reemplazando o complementando la lógica activa. De esta manera la AT40K puede trabajar como un co-procesador reconfigurable.

3.3. Proceso de diseño para FPLD

La utilidad de la arquitectura de los FPLD, se hace más dependiente de herramientas de software automáticas para la síntesis de la lógica y del conexionado.

El proceso de diseño con FPLD, es similar al diseño de cualquier otro dispositivo programable, y puede separarse en 6 pasos [Ska 96]:

1. Definir los requerimientos de diseño
2. Describir el diseño en HDL
3. Simular el código fuente
4. Sintetizar, optimizar y ubicar (fit o place and route) el diseño
5. Simular el modelo de post-layout del diseño
6. Programar el dispositivo

En la Fig. 3.25 se muestra un diagrama de flujo detallado del proceso de diseño usando dispositivos FPLD.

3.3.1. Definir los requerimientos de diseño

Antes de empezar a escribir el código del diseño, debe tenerse una clara idea de los objetivos y requerimientos del diseño. Conocer con certeza los requerimientos del diseño, puede ayudar al diseñador a elegir la metodología y la arquitectura del dispositivo, en el cual se realizará el inicio del diseño.

3.3.2. Describir el diseño en HDL

Una vez definidos los requerimientos del diseño, es conveniente elegir una metodología de diseño antes de empezar a escribir el código. Por lo tanto, este paso se separa en dos etapas, en donde la primera es la elección de una metodología de diseño y la segunda

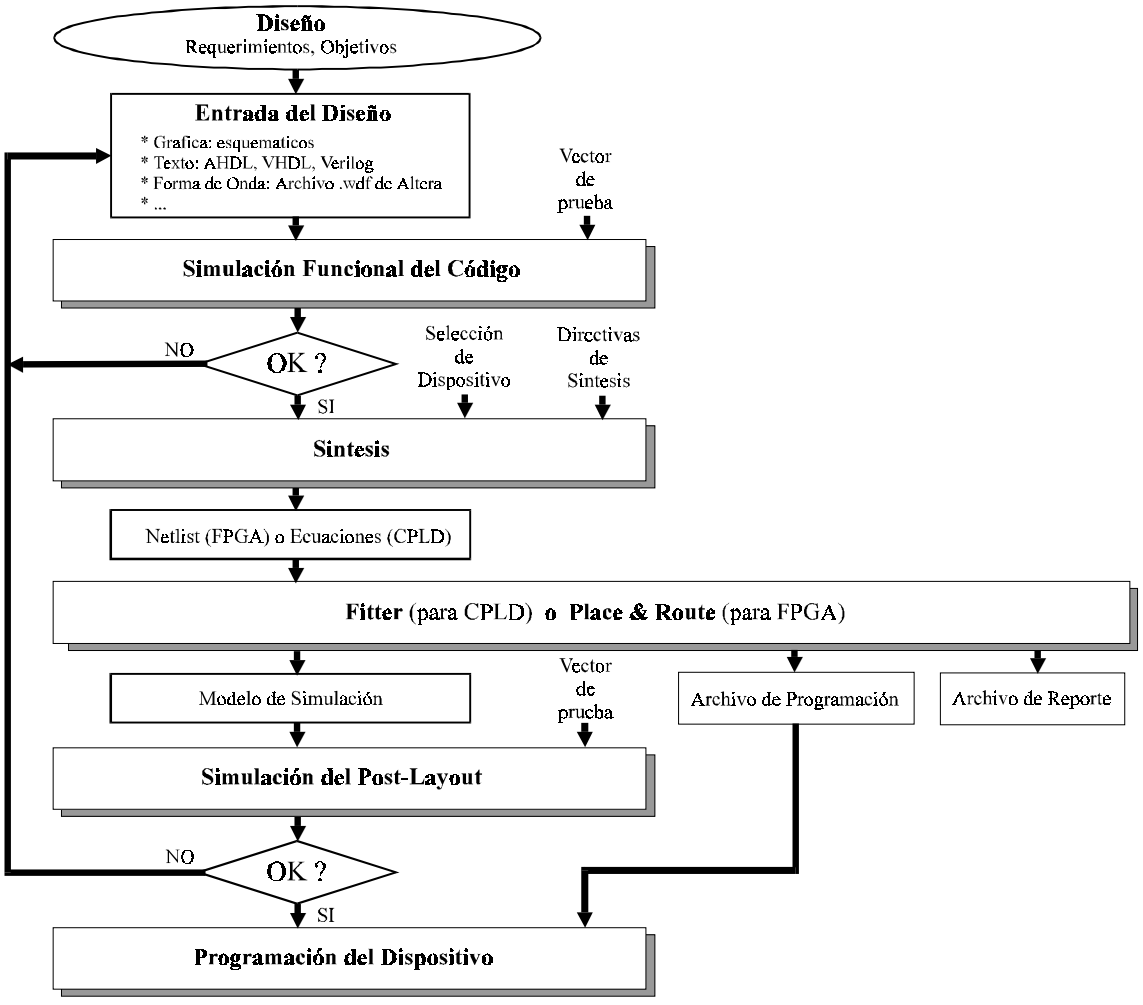


Figura 3.25: Diagrama de flujo del proceso de diseño con FPLD

es la escritura del código.

Existen tres metodologías de diseño: top-down, botton-up y flat. Las dos primeras involucran la creación de diseños jerárquicos, y la última diseños en un plano.

Las metodologías de diseño planas (flat), son útiles en diseños pequeños. En donde un detallado nivel de definición de un bloque funcional, no distrae el entendimiento de la funcionalidad del diseño. Proyectos jerárquicos son útiles en grandes diseños, consisten de múltiples componentes funcionales. Los distintos niveles de jerarquía ayudan a clarificar la interconexión de los componentes.

Luego de decidir la metodología de diseño, debe de comenzarse con la escritura del código, siendo cuidadoso con la sintaxis y semántica acorde a la herramienta de software a utilizar. El punto clave para escribir un buen código de HDL es pensar en términos de hardware [Ska 96]. Mas específicamente, pensar como el software de síntesis "piensa", para entender como se realizará el diseño.

3.3.3. Simular el código fuente

Para grandes diseños, simular el código fuente con un simulador de HDL evidencia un eficiente uso del tiempo disponible para el diseño. La simulación del código fuente puede detectar banderas de error tempranamente en el ciclo de diseño, permitiendo hacer las correcciones necesarias con el menor impacto posible sobre la agenda del mismo. Por otro lado, los grandes diseños son en general jerárquicos, consistiendo de varios subdiseños o módulos. Esta modularidad permite la simulación y debug de cada subdiseño antes del ensamble jerárquico.

En pequeños diseños, la simulación del código fuente puede resultar innecesaria y en algunos casos consumir tiempo de desarrollo innecesariamente.

3.3.4. Sintetizar, optimizar y ubicar (fit o place and route) el diseño

Síntesis: es la reducción de la descripción de un diseño a una representación de más bajo nivel, como ser un netlist o ecuaciones lógicas. En otras palabras, síntesis es el proceso por el cual las ecuaciones lógicas o los netlist, son creados a partir de la descripción del diseño, el cual puede ser abstracto. Por lo tanto, el diseñador presentará una descripción abstracta de su diseño, especificando el comportamiento del circuito digital, y la herramienta de software de síntesis dará como salida un conjunto de ecuaciones lógicas o netlist, que

serán ubicadas en un PLD/CPLD o placed and routed en una FPGA respectivamente. La síntesis es dependiente de la tecnología.

Optimización: el proceso de optimización depende de: la forma de las expresiones Booleanas, el tipo de recurso disponible, y de las directivas de síntesis automáticas o de usuario (también llamadas restricciones). La optimización para CPLD, usualmente involucra la reducción de la lógica a la mínima suma de productos. Esto reduce la utilización de los términos producto y el número de entrada de los bloques lógicos, requeridos para una dada expresión. Por el contrario, la optimización para FPGA típicamente requiere que la expresión lógica se exprese en otra forma que en suma de productos.

Ubicar: ubicar hace referencia a colocar el diseño dentro del dispositivo. En este punto se utilizarán dos términos distintos: "fitting" el cual es utilizado para describir el proceso de ubicación del diseño en un CPLD, y "place and route" que se utiliza para las FPGA [Ska 96]. "Fitting" es el proceso de tomar la lógica producida por la síntesis y optimización, y ubicarla dentro del dispositivo lógico, transformando la lógica (si es necesario) para obtener la mejor ubicación. "Place and route" es el proceso de tomar la lógica producida por la síntesis y optimización, transformarla si es necesario, encajonarla en la estructura (celdas) lógica de la FPGA, ubicando las celdas lógicas en las posiciones optimas, y asignar las rutas de señales desde celdas lógicas a celdas lógicas o I/O.

3.3.5. Simular el modelo de post-layout del diseño

La simulación del post-layout permite la verificación, no solo de la funcionalidad del diseño, sino también los tiempos, tales como los de setup, clock-to-output, register-to-register, etc. Por lo tanto, si el diseño no cumple con los requerimientos, este puede volver a sintetizarse y ubicarse nuevamente en el dispositivo, utilizando distintas directivas de compilación. También puede elegirse un dispositivo con diferente velocidad o hasta cambiar de familia de FPLD.

3.3.6. Programar el dispositivo

Este es el último paso para la implementación del diseño. Por lo cual requiere que los pasos de descripción del diseño, síntesis, optimización, ubicación y simulación, pasen con éxito, para así programar el dispositivo. El software de síntesis, optimización y ubicación produce un archivo usado para programar el dispositivo.

3.4. Herramienta de diseño (Max+Plus II)

La complejidad de los FPLDs requiere de sofisticadas herramientas de diseño, que puedan manejar a grandes diseños de forma eficiente. Estas herramientas de software, usualmente integran los pasos de proceso vistos en 3.3 en un único y uniforme ambiente, permitiendo al diseñador trabajar con diferentes herramientas en un mismo marco de diseño. Por lo tanto, el diseñador puede trabajar en un ambiente abstracto de alto nivel, y al mismo tiempo ver dentro del dispositivo el nivel físico, o sea el nivel mas bajo.

El software de diseño debe cumplir con las siguientes funciones primarias:

Entradas de diseño (Design entry): el software debe proveer un independiente ambiente de diseño que se adapte fácilmente a las necesidades especificas del diseñador. La entrada de diseño más común es la gráfica, compuesta de esquemáticos; pero también pueden ser de HDL, de formas de onda o alguna otra herramienta de software que permita transferir las necesidades del diseñador al traductor.

Traductor (Translator): el diseño de entrada ya sea en forma gráfica, HDL o sus combinaciones, debe traducirse a la forma interna estándar; para que luego pueda procesarse para diferentes arquitecturas de FPLDs. El software traductor realiza las tareas de síntesis lógica, compilación y ubicación de la lógica dentro del dispositivo (partitioning and fitting) elegido. El mecanismo traductor también provee información a las herramientas utilizadas en las siguientes fases del diseño.

Verificación (Verification): el diseño debe verificarse con herramientas de simulación funcional y de timing. Así los errores encontrados pueden corregirse usando la herramienta de entrada de diseño, antes de programar el dispositivo.

Programador de dispositivo (Device programming): esta función permite descargar el diseño dentro del dispositivo FPLD elegido.

Reutilización (reusability): el fabricante de la herramienta de software debe proveer librerías que pueden utilizarse en diferentes diseños, y el diseñador debe poder crear sus propias librerías.

Un ambiente integrado de diseño es aquel que incorpora los marcos de trabajo de todas las etapas del proceso de diseño, empezando con la entrada y terminando con la programación del dispositivo. Altera proporciona un paquete de software llamado

Max+PlusII, el cual es un ambiente integrado de diseño para sus dispositivos [Alt 97a]. El ambiente de diseño para Max+Plus II se muestra en la Fig. 3.26

El corazón del ambiente es un compilador capaz de aceptar diferentes formatos de entradas, y producir archivos con dos propósitos principales: verificación del diseño y programación del dispositivo. La verificación del diseño se realiza con simulaciones funcionales o de timing, esta última involucra los timing del dispositivo seleccionado. La programación puede realizarse con programadores de Altera o de otros fabricantes. Los archivos de salida producidos por el compilador del Max+Plus II, pueden ser leídos y utilizados por otras herramientas de software.

3.4.1. Entradas de diseño

Max+Plus II provee tres editores para la entrada del diseño: esquemático, texto y forma de onda. Los métodos más comunes de entrada de diseño soportadas por este software son:

Esquemático: permite la carga de archivos de diseño gráfico de Altera (.gdf), y archivos esquemáticos de OrCAD (.sch).

Texto: diseños escritos en AHDL (.tdf), VHDL (.vhd), Verilog (.v), utilizando cualquier editor estándar de texto.

Forma de onda: esta es una entrada particular de Altera, permitiendo la carga de archivos (.wdf) en donde el diseño se especifica con formas de onda.

Netlist: diseños en forma de archivos netlist, o generados por otros fabricantes basados en el

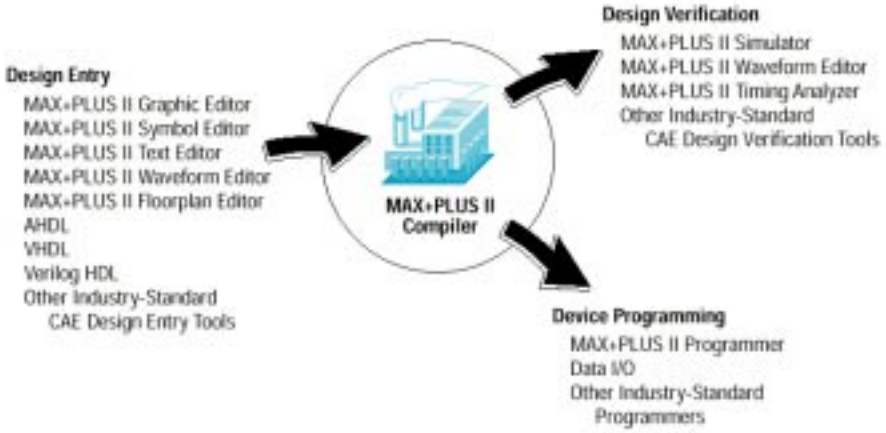


Figura 3.26: Ambiente de diseño para Max+Plus II

estándar de la industria como ser Xilinx (.xnf), pueden ser importados por Max+Plus II.

El diseño de entrada se asocia a un proyecto, el cual contiene en diferentes archivos, la información por ejemplo del dispositivo utilizado, opciones de compilación, etc.

3.4.2. Procesamiento del diseño

Una vez que se cargó el diseño, éste se procesa por el compilador de Max+Plus II generando varios archivos de salida, usados para la verificación, programación y reportes. El compilador consiste de una serie de varios módulos que revisan si existen errores de diseño, sintetizan la lógica, ponen el diseño en uno o varios dispositivos de Altera, generan archivos para la simulación, archivos para el análisis de timing, y archivos para la programación. Las entradas al compilador son los archivos de entrada de diseño, los archivos de asignaciones y de configuración del proyecto, archivos de símbolos, etc [Ska 96].

3.4.3. Verificación del proyecto

El proceso de verificación del proyecto se realiza con dos herramientas: el simulador y el analizador de timing.

El simulador prueba el funcionamiento lógico y el timing interno del proyecto. Para simular el proyecto, el compilador genera un archivo (.snf) netlist de simulación, el cual se carga en la herramienta de simulación. También deben cargarse en éste, los vectores de entrada, los cuales pueden ser de forma gráfica (.scf) o de texto ASCII (.vec). El editor de formas de onda crea por defecto un archivo de tipo gráfico (.scf), que puede ser utilizado por el diseñador. La simulación permite observar, en las salidas elegidas, glitches, oscilaciones y violaciones de tiempos de setup y hold.

El analizador de timing permite al diseñador analizar la perfomance de timing del proyecto, después que fue optimizado por el compilador. Todos los caminos de las señales del proyecto pueden rastreadas, determinando caminos críticos de velocidad y caminos que limitan la perfomance del proyecto. Una vez que el analizador de timing completa el análisis, es posible seleccionar un nodo fuente o destino y listar los retardos asociados con su camino.

3.4.4. Programación del dispositivo

La última porción del ambiente integrado de diseño, es el software y hardware necesarios para programar dispositivos de Altera. La parte del software en Max+Plus II se llama “Programmer”. El Hardware puede ser un programador de Altera o cualquier otro programador estándar de la industria. Altera posee dispositivos que pueden ser configurados utilizando JTAG (Join Test Action Group), por lo cual existen interfaces que se conectan en el puerto serial (BitBlaster) o paralelo (ByteBlaster) de la PC.

3.4.5. Sistema de desarrollo – Altera UP1 placa de evaluación

Altera a desarrollado el paquete UP1 (University Program 1), para la implementación de prototipos digitales, con propósitos académicos y en el ámbito universitario. El paquete esta compuesto por una placa de evaluación, interface ByteBlaster, y un ambiente de desarrollo Max+Plus II en versión estudiantil [Alt 97b]. En la Fig. 3.27 se presenta el ambiente de desarrollo junto con una foto de la placa UP1.

El ambiente de diseño integrado Max+Plus II se introdujo en los puntos anteriores. La interfaz ByteBlaster se conecta al puerto paralelo de la PC. A través de esta y utilizando JTAG, se configuran los dos tipos de FPLD presentes en la UP1. Uno de los dispositivos esta basado en términos productos perteneciendo a la familia de CPLD MAX (EPM7128S). El otro esta basado en Look-up table perteneciendo a la familia de CPLD FLEX (EPF10K20), y utilizando SRAM para su configuración.

La parte correspondiente al MAX, esta compuesta por un EPM7128S-7 de 128 Macro celdas, 7 nano segundos de retardo y un equivalente a 2500 compuertas lógicas. Posee 84 pines, en donde todos están accesibles a través de conectores en la placa. La parte que corresponde al FLEX, tiene un EPF10K20-240 con 1152 elementos lógicos, 6 EAB de 2048 bits de SRAM y un total de 240 pines. Con un equivalente de 20000 compuertas, es conveniente para grandes diseños incluyendo sistemas computacionales complejos, comunicaciones, sistemas de DSP, etc.

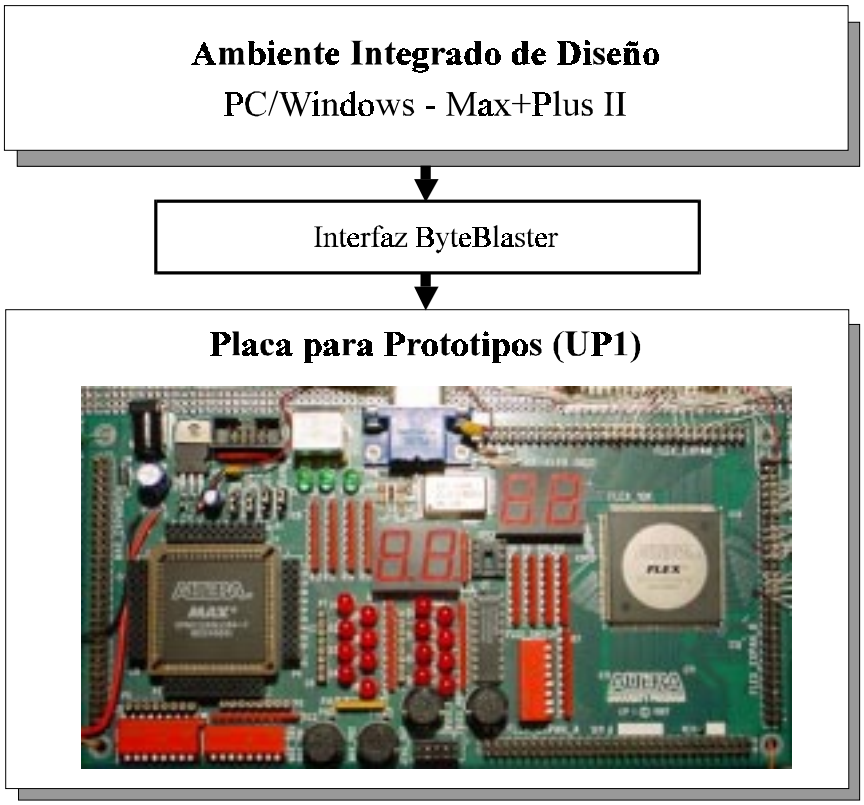


Figura 3.27: Ambiente global de diseño utilizando la placa de evaluación (UP1) de Altera

3.5. Resumen

En este capítulo se mostró que tanto los CPLD como las FPGA tienen sus ventajas y desventajas. Los CPLD usualmente proveen una alta perfomance, pero contienen menos registros que las FPGA. Debido a que los CPLD pueden implementar grandes funciones en un solo arreglo lógico, el timing es predecible y por lo tanto de fácil análisis. Los recursos de los CPLD están separados en arreglos lógicos, lo cual impone restricciones a como pueden ser utilizados. Las FPGA tienen arquitecturas de grano fino. Estas realizan bien diseños pipelined, pero también pueden implementar grandes caminos de datos poniendo en cascada múltiples celdas lógicas. Las FPGA contienen arreglos de celdas lógicas, estando menos separadas que en los CPLD. Esto hace que el dispositivo sea más accesible, pero a expensas de una más difícil optimización y análisis de timing [Ska 96][Alt 95].

El modulador se implementó con la placa de evaluación de Altera denominada UP1, la cual consta de dos FPLDs uno de la familia MAX7000S y otro de la familia FLEX10K. Esta también incluye el software de desarrollo de ambiente integrado basado en Windows

Max+Plus II en versión estudiantil. Permitiendo compilar y realizar simulaciones con los timing del dispositivo seleccionado.

Debido a la rápida y creciente evolución en la tecnología, es posible que los datos específicos queden desactualizados. Esto tiene aparejado una difícil y poco redituable tarea para resumir el estado del arte en los FPLDs. Por lo tanto se expondrán solo algunos puntos en la futura dirección de la lógica programable:

- Los requerimientos de perfomance y densidad se incrementarán
- Los mercados de 3.3V estan establecidos y continuarán creciendo hacia 1.2V y 0.8V
- El uso de RAM on-board y otras funciones e interfaces dedicadas pueden proliferar
- La distinción entre CPLD y FPGA puede hacerse menos clara, a medida en que los fabricantes de lógica programable experimenten con FPGA separadas en bloques de lógica con recursos de ruteo dedicados
- La programabilidad en el sistema continuara siendo utilizada

4. DESARROLLO DE UN MODULADOR LINEAL

4.1. Introducción

El modulador lineal encuentra aplicación, tanto en el control de motores como en el de filtros activos. Está especialmente indicado cuando no son críticos: ni el uso completo de la tensión continua (por ejemplo aplicaciones de baja tensión), ni la distorsión debida al tiempo muerto en la excitación de las llaves de potencia (cuando no se usan índices de modulación bajos, por ejemplo motores que no deban girar a muy baja velocidad).

En este capítulo se presenta el desarrollo de un modulador vectorial que trabaja en la zona lineal. Primero se determinan los principales parámetros a tener en cuenta para su diseño como ser cuantización de las muestras, discretización del tiempo, frecuencia de muestreo, frecuencia de reloj, tamaño de la memoria necesaria. Luego se desarrolla el modulador a ser construido con un FPLD. La implementación de un prototipo del modulador vectorial lineal, se lleva a cabo utilizando un circuito de ALTERA y dos memorias EPROM. En él se implementa toda la lógica necesaria para el algoritmo del modulador lineal, la comunicación con la PC y el inversor de potencia; mientras que en las memorias se almacenan los tiempos t_a y t_b. Esta etapa de la tesis dio como resultado un trabajo presentado en Congreso Brasileño de Electrónica de Potencia (COBEP'99) [Ton 99] y la obtención del segundo puesto en la competencia estudiantil *IEEE Myron Zucker Student Design Award* [Ton 99b].

4.2. Cuantización de las muestras y discretización del tiempo en la implementación del modulador

En el capítulo 2 se dieron las nociones básicas para la implementación de un modulador vectorial, y en este capítulo se desarrollará la implementación de un modulador vectorial lineal. A continuación se presentarán los principales parámetros a tener en cuenta para su diseño.

- *la frecuencia de conmutación (f_s) que define el tiempo de sub-ciclo (Δt).*
- *el número de bits (n), necesarios para implementar los tiempos t_a, t_b y t_0, que determinan la cantidad de valores discretos que puede tener el vector de referencia.*
- *la frecuencia del clock,* $f_{clk} = 2^{(n-1)} / f_s$

- *el número de bits utilizados para cuantificar el módulo y la fase del vector de referencia.*
- *el tamaño de las EPROM, que está determinado por la cuantización del vector de referencia.*

4.2.1. Cuantización de los vectores espaciales debido a la discretización del tiempo

Para mostrar la cuantización de los vectores espaciales debido a la discretización del tiempo, se empleará un ejemplo suponiendo n = 5 bits. El subciclo Δt estará implementado con 32 (2^5) períodos de clock. Entonces, la fracción más pequeña del vector espacial que puede ser distinguida es $\frac{2}{3}Vcc/32$, siendo esta la cuantización de los vectores espaciales determinada por la discretización del tiempo.

El vector de referencia es implementado con valores cuantizados de los vectores espaciales, por lo que puede tomar solo valores discretos [Hol 87], como se muestra en la Fig. 4.1. Para n = 5 bits, se tienen 2^5 x 2^5 / 2 = 512 valores discretos por sector. En consecuencia, la muestra del vector de referencia se aproxima a uno de los $2^{(2n - 1)}$ valores discretos que pueden encontrarse en cada sector. En la Fig. 4.1 se muestran los puntos posibles para implementar un vector de referencia en el sector I, una situación similar puede encontrarse para los demás sectores. Puede observarse, por ejemplo, que para implementar a $\mathbf{V_R}^*$ es necesario que $\mathbf{V_1}$ se aplique durante 3 periodos de reloj (T_{CLK}), $\mathbf{V_2}$ durante $17T_{CLK}$, y durante $11T_{CLK}$ a $\mathbf{V_N}$

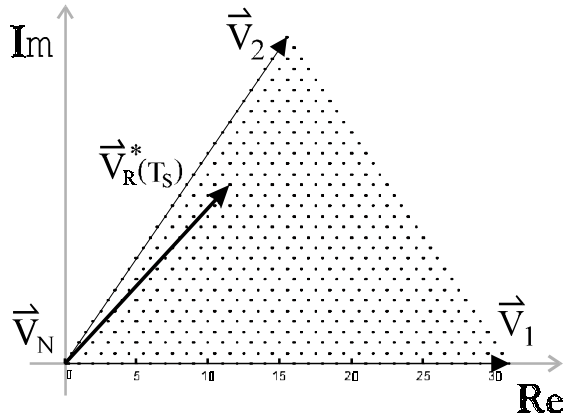


Figura 4.1: Cuantización de los vectores espaciales, debido a la discretización del tiempo.

Al aumentar n, se tendrán más valores discretos para aproximar a $\mathbf{V_R}^*$. Cada bit que se adiciona a la discretización del tiempo, incrementa cuatro veces la resolución en cada sector. Al mismo tiempo aumentará la frecuencia del clock, y por ende los tiempos de acceso de las memorias no serán despreciables. Por lo tanto, para la elección del número de bits n existe una relación de compromiso entre la resolución de la cuantización de los vectores espaciales y el timing de los elementos del circuito.

Un aspecto importante relacionado a la frecuencia de conmutación es el ruido acústico que genera. La radiación acústica está fuertemente relacionada con la distribución espectral de las componentes armónicas de las corrientes, las cuales están fijadas por la frecuencia de conmutación del inversor de potencia. La sensibilidad del oído humano determina que frecuencias de conmutación debajo de 500 Hz y encima de 10 kHz sean menos críticas, mientras la máxima sensibilidad del oído está alrededor de 1-2 kHz.

Teniendo en cuenta lo dicho anteriormente se eligió $\mathbf{f_s = 20\ Khz}$, la cual resulta razonable para los dispositivos IGBTs utilizados en esta aplicación.

Para la discretización del tiempo se eligió **n = 8 bits** [Han 92]; dando una resolución de $\Delta t/256 = 1/(512\ f_s) = 1/10.24MHz = \mathbf{T_{CLK} = 98\ nseg}$. Así resulta una frecuencia de reloj ($\mathbf{f_{CLK} = 10.24MHz}$) que se puede manejar sin inconvenientes con la lógica TTL, que se utilizará en su implementación. La cantidad de valores discretos (trama) que puede tomar el vector de referencia es de 32768 por sector, como se vio anteriormente.

Como la comunicación del modulador se hace a través del Bus ISA extendido de la PC, se aprovecha el uso del clock que éste provee. Por lo tanto se tiene definida la frecuencia del clock, siendo $\mathbf{f_{CLK} = 8.375\ MHz}$, y el número de bits a utilizar (**n = 8 bits**) con lo que se modifica la frecuencia de conmutación a: $\mathbf{f_s} = f_{CLK}/512 = \mathbf{16.4\ KHz}$, siendo un valor razonable.

4.2.2. Problemas debido al tiempo de conmutación de los dispositivos de potencia

Los dispositivos semiconductores utilizados como interruptores electrónicos no son ideales, por lo que éstos reaccionarán con cierto retardo a sus señales de control de encendido y apagado. El tiempo de retardo depende del tipo del semiconductor, sus rangos de corrientes y tensiones, la temperatura del dispositivo, y de la corriente real a ser conmutada (ver el punto 6.2). Para evitar cortocircuitos en las columnas del inversor, el circuito de control de las señales de excitación del inversor debe introducir un tiempo de retardo T_D, en la señal de activación del dispositivo.

El inversor utilizado para obtener los resultados experimentales, recomienda un tiempo T_D = 2 μseg, el cuál es un poco inferior a 21 periodos de Clock (2,058 μseg). Cuando alguno de los tiempos t_a, t_b o t_0, resulte inferior a $21T_{CLK}$, no podrá ser implementado correctamente por las llaves del inversor. Para salvar esta situación, el tiempo correspondiente se aproxima al valor más próximo, que se encuentra entre 0 y $21T_{CLK}$. Esta aproximación tiene como resultado un error en la tensión de salida del inversor [Han 92]. En la Fig. 4.2 se detallan los

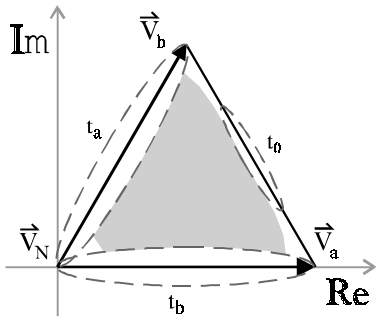


Figura 4.2: Limitaciones en la cuantización debido a los tiempos mínimos.

lugares donde los tiempos son inferiores al tiempo de transición, siendo el lugar donde se realiza la aproximación.

Cabe aclarar que el no poder implementar tiempos menores a T_D (2 μseg) se debe a un problema físico, no teniendo nada que ver con la resolución empleada. Recordar, que la resolución del tiempo es el período del clock utilizado (T_{CLK}).

4.2.3. Resolución del módulo (V^*) y del ángulo (α) del vector de referencia

Las muestras del vector de referencia se cuantifican en módulo y fase, dividiendo al plano complejo con líneas radiales y círculos concéntricos, cuyas intersecciones son los valores cuantificados. Por otro lado, la cuantización del tiempo tiene como consecuencia la cuantización de los vectores espaciales activos y nulos. Tomando la suma de las cuantizaciones de los vectores espaciales en un período de conmutación, se obtienen los valores promedios que puede sintetizar el inversor de potencia como muestra la Fig. 4.1.

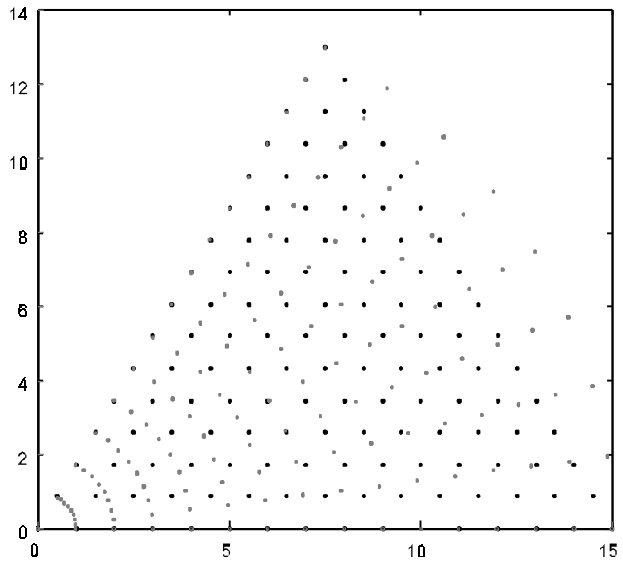


Figura 4.3: Diferencias entre las muestras que puede sintetizar el inversor de potencia (negro) y las muestras cuantizadas del vector de referencia (gris).

Ambas cuantizaciones se detallan en la Fig. 4.3, donde se eligió para la discretización del tiempo n = 4 bits, para el módulo 4 bits y para la fase 3 bits, con el objetivo de una buena visualización de las diferencias. Con color negro se muestran los valores que puede implementar el inversor y con color

gris los valores que resultan de la cuantización del módulo y la fase del vector de referencia. La muestra del vector de referencia (punto gris) debe aproximarse al valor más cercano posible (punto negro), para que pueda ser implementada por el inversor de potencia.

La comunicación se realiza a través del Bus de Datos de la PC en 16 bits, de los cuales 3 se utilizan para decodificar en qué sector se encuentra el vector de referencia ($\mathbf{V_S}^*$). Los 13 bits restantes son repartidos para las resoluciones del módulo (V^*) y del ángulo (α) de $\mathbf{V_S}^*$. Una buena elección es de 7 bits para el V^* y 6 bits para α, donde α se encuentra entre [0 60º).

4.2.4. Descripción de la implementación de una muestra.

Cada muestra del vector de referencia $\mathbf{V_S}^*$ es implementada durante dos subciclos ($2\Delta t$) consecutivos, de modo de obtener una modulación simétrica que permite disminuir el ripple de corriente, como muestra la Fig. 4.4. Es conveniente tomar la muestra del vector de referencia al final del subciclo impar, para comenzar su implementación al principio del siguiente subciclo par, como indica la Fig. 4.4. Con éste criterio se tendrá un retardo constante de Δt, y no existirá mucha exigencia en los tiempos de acceso de las EPROMs.

4.3. Implementación de un modulador lineal en un FPLD de Altera

En la Fig. 4.5 se presenta un diagrama en bloques del modulador vectorial lineal propuesto, el cuál consta básicamente de tres etapas:

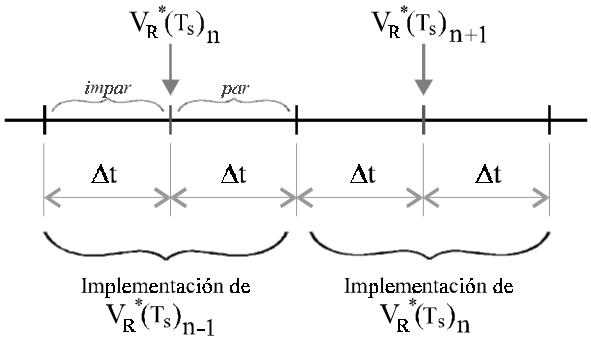


Figura 4.4: Diagrama de implementación de las muestras del vector de referencia

- *Entrada*: usada para la comunicación entre el modulador y la PC, a través del bus ISA. Esta etapa puede ser fácilmente adaptada para comunicar al modulador por medio de otro bus como el PCI, o con otro sistema como ser con un DSP. Consta de dos bloques, uno es un registro (Latch IN) donde se retiene al nuevo vector espacial a implementar y el otro es el bloque de decodificación que genera la señal de habilitación del modulador vectorial.
- *Algoritmo de Modulación Vectorial:* Esta es la etapa principal del Modulador Vectorial Lineal. En él se genera el algoritmo de modulación vectorial, trabajando en la zona lineal y manteniendo el número de conmutaciones en un mínimo. Está compuesto por cuatro bloques. El registro de entrada (Latch II), donde se almacenan los tiempos de los vectores espaciales activos (t_a, t_b) que deben ser sintetizados por los contadores. Estos tiempos se calculan "off-line" y se guardan en memorias (EPROM). La elección de almacenar los tiempos en memorias externas se hizo para disminuir el tamaño del CPLD empleado, pero se podrían incluir tranquilamente en él. El bloque de los contadores, se encarga de la implementación del período de conmutación (T_S) y de los tiempos en que deben aplicarse los vectores espaciales activos (t_a y t_b). La máquina secuencial se ocupa principalmente de decidir qué contador va a actuar y cual no, controlando así los intervalos de tiempo e implementando la secuencia de vectores espaciales deseada. Esta secuencia se selecciona para que el Inversor de Potencia tenga mínimas conmutaciones, como se vio en el punto 2.4. El bloque de salida es la tabla de estados, en donde se almacenan los estados de los ocho vectores espaciales; siendo manejada por la máquina secuencial y el Latch II.
- *Salida:* acondiciona las señales de excitación de las llaves del inversor, que provienen de la etapa del algoritmo de modulación vectorial. Es una etapa importante debido a que genera los retardos necesarios para que no se produzcan cortocircuitos en las columnas del inversor.

En los siguientes puntos se darán los detalles para la implementación de cada una de las etapas del modulador vectorial lineal.

4.3.1. Etapa de entrada

La función de esta etapa es hacer de interfaz entre el modulador vectorial lineal y la PC, a través del Bus ISA utilizando los 16 bits del Bus de datos. El protocolo de comunicación que se utiliza es de tipo "*Hand-shake*". El modulador recibe desde la PC el vector espacial de referencia que debe implementar. Cuando el algoritmo de control en la PC calculó un nuevo

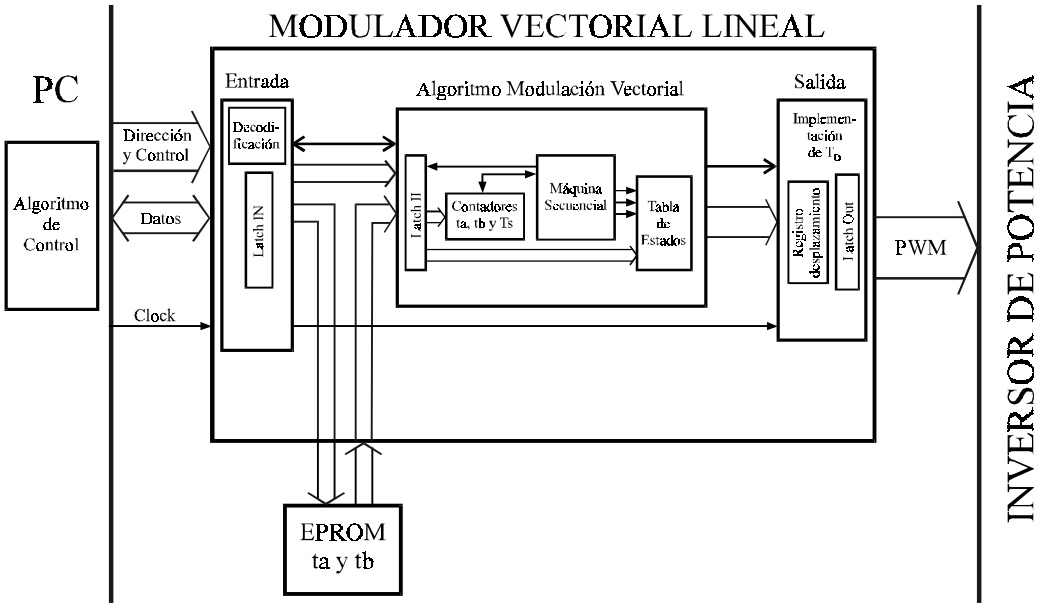


Figura 4.5: Diagrama en bloques del modulador vectorial lineal propuesto.

vector espacial, la PC inicia una comunicación con el modulador preguntándole si está preparado para recibirlo. Si el modulador está preparado, entonces la PC se lo pasa y el modulador lo carga, sino espera a que lo esté. El modulador le indica a la PC que está listo para recibir un nuevo vector espacial, con un determinado dato el cuál es leído por la PC. En el caso particular que el algoritmo de control no entregue al modulador un nuevo vector espacial, éste repetirá la implementación del vector espacial que tenga cargado. El modulador vectorial carga el nuevo vector espacial durante la implementación del viejo.

La etapa de entrada consta principalmente de dos bloques, uno es de decodificación y el otro es un registro de entrada llamado Latch IN. La PC escribe sobre la dirección del modulador en un solo ciclo de escritura del bus ISA. Así los 16 bits del bus de dato (D_{15}...D_0), que contienen la información del sector, módulo y fase de la muestra del vector de referencia, se mantienen en el registro Latch IN garantizando el direccionamiento de las EPROMs. Los 3 bits más significativos D_{15}, D_{14} y D_{13} se usan para decodificar los 6 sectores y controlar el apagado del modulador, como se detalla en la Tabla 4.1. De los 13 bits restantes del bus de datos, 7 bits (D_{12},...,D_6) se utilizan para cuantizar el módulo y 6 (D_5,...,D_0) la fase de la muestra del vector de referencia. Estos 13 bits forman la dirección de las EPROM, en donde se encuentran almacenados los tiempos de activación de las llaves del inversor de potencia.

La dirección del modulador se forma con las líneas A_9,...,A_0 del bus de direcciones, ocupando una sola posición en el mapa de direcciones de la PC. Para lograr una cierta flexibilidad en la ubicación del modulador dentro del mapa de direcciones, las líneas

A_4,...,A_1 se comparan con el estado de cuatro llaves. Estas llaves sirven para definir la dirección del modulador en el mapa de direcciones de la PC. De este modo, el modulador puede localizarse en cualquier dirección par que esté comprendida entre 0300h y 031Eh. Junto con las líneas de dirección, se utilizan las líneas de control para garantizar una transferencia de datos en 16 bits, y asegurar que los datos son estables durante la transferencia.

Tabla 4.1: Decodificación de los 3 bits más significativos del bus de datos

D_{15}	D_{14}	D_{13}	Decodificación
0	0	0	Sector I
0	0	1	Sector II
0	1	0	Sector III
0	1	1	Sector IV
1	0	0	Sector V
1	0	1	Sector VI
1	1	0	No importa
1	1	1	Apagado

En la Fig. 4.6 se muestran todos los bloques y señales que componen la etapa de entrada. Como puede observarse esta etapa está compuesta por cuatro bloques. Los bloques principales, como se dijo anteriormente, son la decodificación y el registro Latch IN.

El bloque de Habilitación de Carga se encarga de comunicar a la PC, a través del bit $\boldsymbol{D_0}$ del Bus de Datos, si se puede cargar o no una nueva muestra del vector de referencia. Este bloque tiene como entradas las señales de salida del bloque de decodificación (***PC_Observa*** y ***LoadN***), y las señales ***MSB*** y ***twN*** que provienen de la etapa Algoritmo Modulación Vectorial.

El bloque ON/OFF comanda el encendido y apagado del algoritmo de modulación vectorial, teniendo como entradas a las señales ***twN***, ***LoadN*** y tres señales de salida del registro Latch IN (***LD*$_{15}$**, ***LD*$_{14}$** y ***LD*$_{13}$**). Este bloque también actúa como protección, ya que si por cualquier motivo la PC deja de entregarle nuevas muestras de referencia al modulador, éste repetirá tres veces la implementación de la misma muestra. De esta manera, se evita la generación de una señal continua en el inversor de potencia.

El modulador empieza a trabajar cuando la señal ***Start*** se activa poniéndose en uno, esto ocurre en el primer acceso al modulador. Con la señal ***Start*** activa, se habilita la cuenta de un contador binario (C) *"free running"* en la etapa Algoritmo Modulación Vectorial. Este

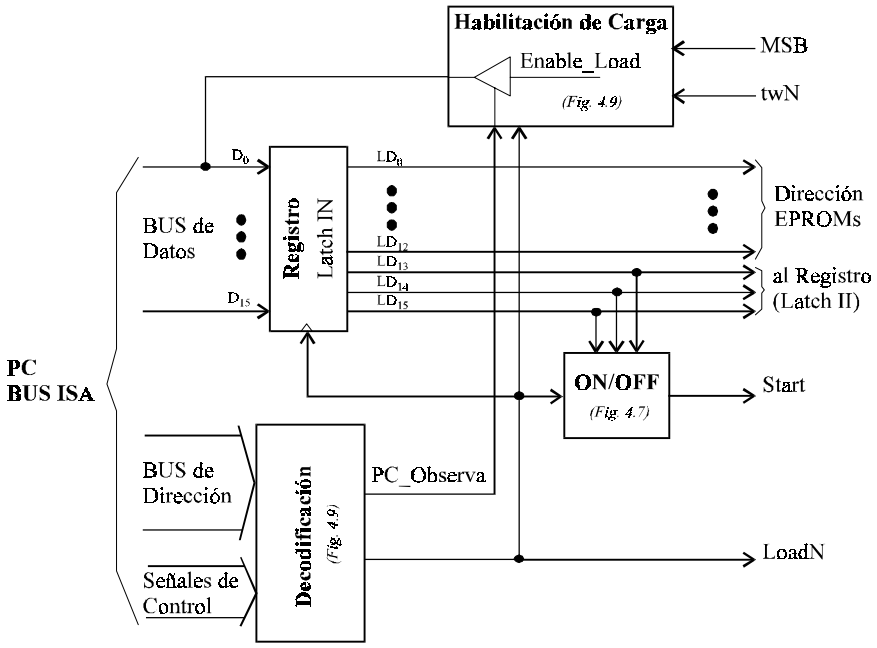


Figura 4.6: Diagrama en bloques y señales que componen la etapa de entrada

genera los subciclos Δt identificados por la señal ***MSB***. Cuando la señal de ***Start*** se desactiva, el contador C se detiene y la señal ***MSB*** que identifica al subciclo par o impar permanece invariable. La señal ***Start*** puede desactivarse de varias maneras: una es por medio de la línea ***ResetDrv*** que pertenece a las señales de control del Bus ISA, otra por la línea ***Seguro*** y la última es por medio de los bits del Bus de Datos D_{15}, D_{14} y D_{13}, como indica la Tabla 4.1. La primera situación corresponde a algún problema en la PC, mientras que la segunda ocurre cuando no se cargan muestras nuevas de referencia, y la última es cuando se decide detener el funcionamiento del modulador por medio del software. En la Fig. 4.7 se muestra el diagrama esquemático del circuito que implementa el bloque ON/OFF de la etapa de entrada, puede observarse como se genera la señal ***Start***.

Cada vez que la PC escribe en la dirección del modulador, la señal ***LoadN*** se activa pasando al nivel bajo ("0" lógico), produciendo luego con el flanco de subida la carga de una nueva muestra del vector de referencia en el registro Latch IN. Cuando la PC lee el bit D_0 en la dirección del modulador, observando si el modulador puede o no cargar una nueva muestra, la señal ***PC_Observa*** se activa pasando al estado alto y habilitando al buffer tri-state del bloque de Habilitación de Carga (Fig. 4.6).

En la Fig. 4.8 se muestra un diagrama con las señales a utilizar para explicar gráficamente el "*Hand-Shake*". La señal ***MSB*** identifica al subciclo Δt, en ***MSB*** = 0 el subciclo es impar y en ***MSB*** = 1 el subciclo es par. Con el flanco de subida de la señal ***LoadN*** se carga en Latch IN una nueva dirección de las EPROM, formada como se dijo por el módulo y la fase del

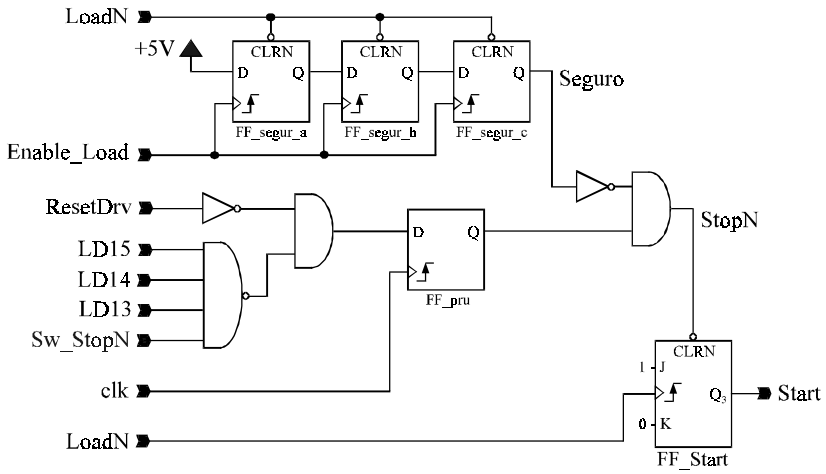


Figura 4.7: Diagrama esquemático del Bloque ON/OFF de la etapa de Entrada

vector de referencia. Luego del tiempo de acceso (t_{acc}) de las EPROMs, estarán disponibles los tiempos t_a y t_b que ingresan a la etapa Algoritmo Modulación Vectorial. La activación de la señal ***LoadN*** significa la carga de una muestra, por lo que la señal ***Enable_Load*** se desactiva pasando al estado bajo impidiendo la carga de otra muestra. Esta señal se activa por medio de ***MSB***, poniéndose en "1" al comienzo de todo subciclo par.

Un intervalo de tiempo t_w antes de que finalice el subciclo impar, la señal ***twN*** hace que ***Enable_Load*** se desactive, impidiéndose la carga de una muestra al modulador. Esto se realiza para salvar el tiempo de acceso de las EPROMs, por lo tanto el tiempo t_w tiene que ser mayor que t_{acc} (320 nseg); en este caso se toma $t_w = 4*T_{CLK} \cong 480$ nseg.

Con el inicio de un nuevo subciclo impar, comienza la implementación de la muestra que se encuentra en la entrada de la etapa Algoritmo Modulación Vectorial, siendo la salida de las EPROMs. Si durante un período de conmutación ($T_s = 2\Delta t$) no se cargó ninguna muestra,

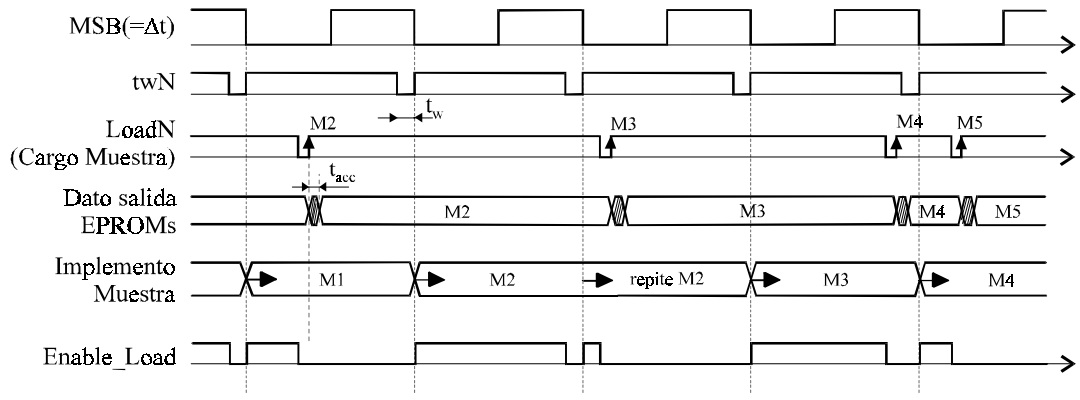


Figura 4.8: Diagrama de las señales para comunicar el modulador con la PC

entonces el modulador repite la implementación de la muestra anterior; esto puede repetirse como máximo hasta tres veces, luego el modulador se apagará.

En la Fig. 4.9 se muestran los diagramas esquemáticos de los circuitos que sintetizan el funcionamiento de los bloques de Decodificación y Habilitación de Carga, en la etapa de Entrada; puede observarse como se generan las señales ***LoadN***, ***PC_Observa*** y ***Enable_Load***. La señal ***twN*** se sintetiza utilizando las salidas del contador binario C, en la etapa Algoritmo Modulación Vectorial, de manera tal de ajustarla al tiempo $t_w = 4*T_{CLK}$. La señal ***MSB*** es el bit más significativo del contador binario C. El registro Latch IN se sintetiza con 16 Flip-Flop tipo D, que actúan con el flanco de subida.

4.3.2. Etapa del Algoritmo de Modulación Vectorial Lineal

El algoritmo de modulación vectorial lineal debe ser capaz de controlar los intervalos de tiempo t_a, t_b y t_0 en que se aplican diferentes vectores espaciales. También debe seleccionar adecuadamente el vector espacial que corresponde al sector en cuestión y al intervalo de

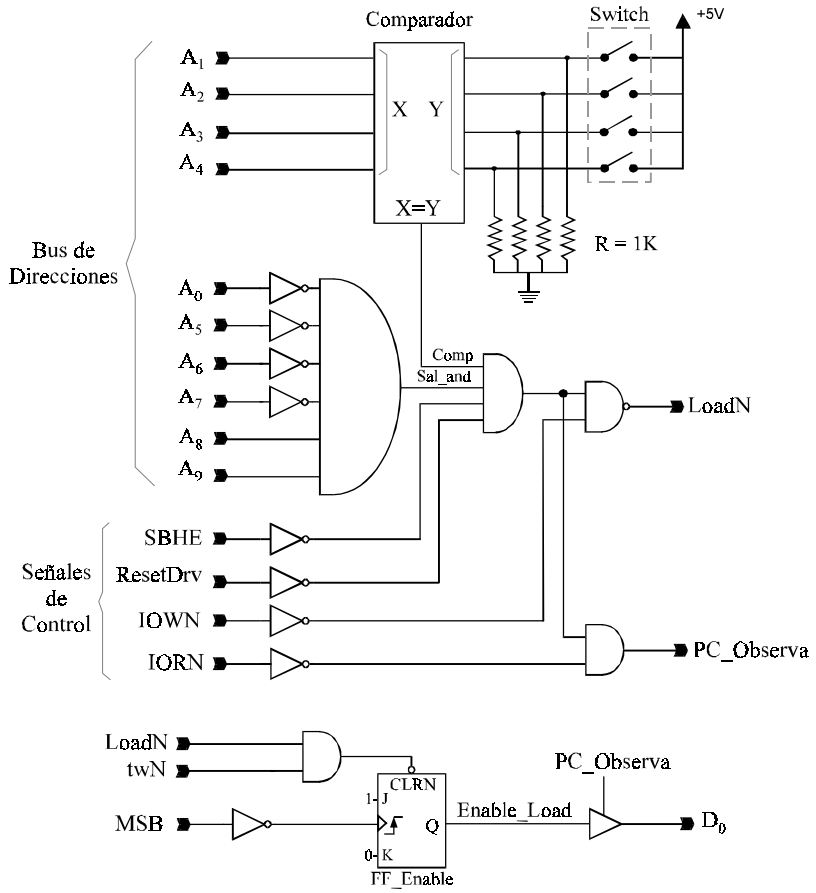


Figura 4.9: Diagrama esquemático del bloque de Decodificación y del bloque de Habilitación de Carga

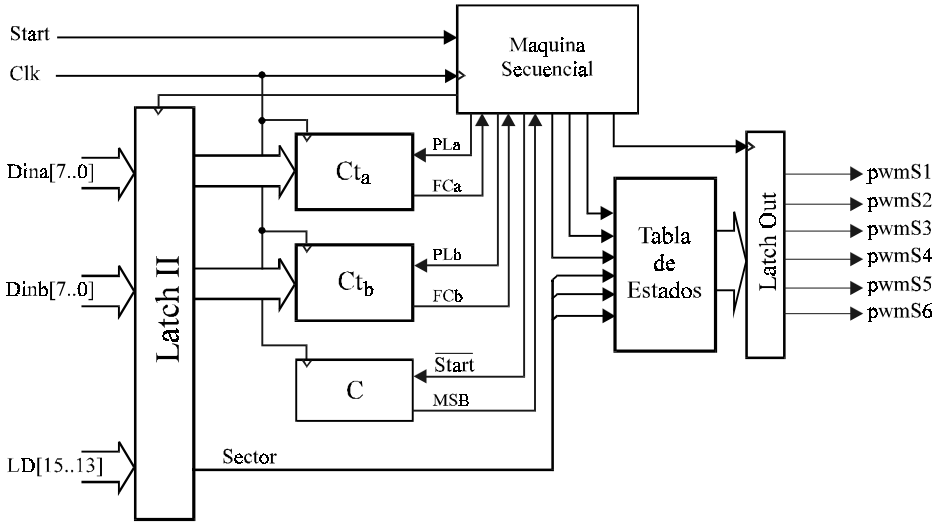


Figura 4.10: Diagrama en bloques de la etapa principal del modulador vectorial lineal

tiempo que se esté implementando. Para llevar a cabo estos requerimientos, el modulador vectorial lineal emplea tres contadores, una máquina secuencial, un registro y una tabla. En la Fig. 4.10 se muestra en diagrama de bloques los componentes del modulador vectorial lineal.

Los tiempos t_a y t_b, dados por (2.13), se calculan off-line y se almacenan en las memorias EPROM t_a y EPROM t_b respectivamente. En los contadores Ct_a y Ct_b se cargan los tiempos t_a y t_b respectivamente, para su implementación. El contador binario C genera el tiempo de subciclo Δt, contando un número fijo de períodos de clock en el modo *"free running"*. El tiempo t_0, que no se calculó off-line, se implementa en el intervalo de tiempo que existe entre que se detiene el contador Ct_a (Ct_b) y el fin del subciclo par (impar). La tabla de estados almacena los estados de las seis llaves del inversor, correspondientes a cada vector espacial. Las direcciones de la tabla están compuestas por una parte con la información del sector en que se encuentra el vector de referencia, y por otra por la máquina secuencial. Las entradas de la máquina secuencial son el clock, las señales de encendido y apagado, las señales de fin de cuenta de Ct_a y Ct_b y el subciclo. Ésta genera las señales de habilitación del Latch II y Latch Out, controla el encendido y apagado de los tres contadores y también genera tres líneas para direccionar la tabla de estados.

La máquina secuencial se diseña teniendo en cuenta que cada muestra del vector de referencia se implementa en dos subciclos consecutivos, por una secuencia de tres vectores espaciales por cada subciclo (Fig. 4.4). La secuencia elegida es $|<V_a><V_b><V_N>|$ en los subciclos impares y $|<V_b><V_a><V_N>|$ en los pares. Durante los subciclos impares la señal ***MSB***, dada por el bit más significativo del contador C, está en estado bajo; mientras que

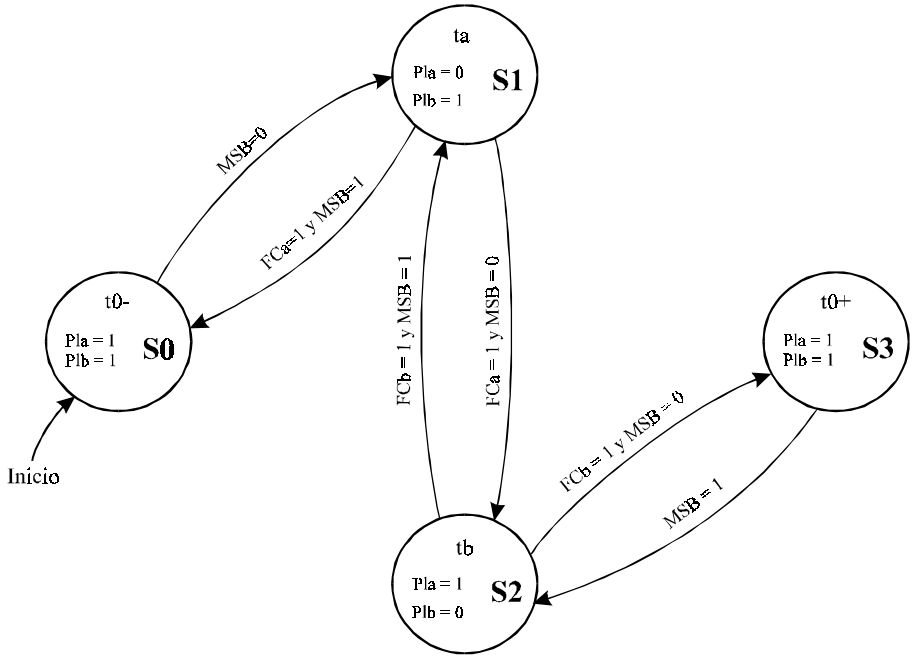


Figura 4.11: Diagramas de estados de la máquina secuencial.

permanece en estado alto para los subciclos pares. Una transición de alto a bajo en ***MSB*** determina el comienzo de un subciclo impar, entonces el contador Ct_a debe iniciar la cuenta del tiempo t_a. Cuando Ct_a termina de implementar a t_a, debe entregar una señal (***FCa***) para iniciar la cuenta de Ct_b y auto detenerse. Después del fin de la implementación de t_b, se implementa el tiempo t_0. Éste se lleva a cabo desde que termina Ct_b hasta que aparece una transición de bajo a alto en ***MSB***. Entonces empieza el subciclo par, por lo que la secuencia debe ser inversa como se muestra en la Fig. 4.11, donde se realizó un diagrama de estados de la máquina secuencial. Las señales ***FCa*** y ***FCb*** son generadas por los contadores Ct_a y Ct_b respectivamente, cuando terminan de contar. Estas detienen al propio contador e inician la cuenta del siguiente.

El funcionamiento de la máquina secuencial está basado en el contador C, por lo que se debe poder controlar el encendido y apagado de dicho contador. Esto se realiza manejando la señal ***Start*** ya sea por software, a través de la etapa de decodificación o por hardware cuando ocurre algún problema en la PC. La Tabla 4.2 resume los niveles lógicos y las transiciones de las señales para cada estado de la máquina secuencial.

En la Fig. 4.12 se muestran los estados de las principales señales del modulador, que se explicaran a continuación para entender mejor el funcionamiento de la máquina secuencial.

Las líneas $\boldsymbol{PL_a}$ y $\boldsymbol{PL_b}$ controlan la carga de datos e inicio de cuenta de los contadores Ct_a y Ct_b respectivamente. Cuando están en alto el contador carga los datos (t_a o t_b), y cuando están en bajo comienza la cuenta descendente hasta llegar a cero. Cuando el contador C está

trabajando, la máquina secuencial está en marcha, y el nivel lógico de la señal ***MSB*** define en qué subciclo Δt se está operando. Supóngase que los tiempos t_a y t_b están cargados en los contadores y que se empieza a operar en el subciclo impar, entonces el primer intervalo de

Tabla 4.2: Estados de las señales para los diferentes modos de operación de la máquina secuencial.

Entradas				Salidas		Modos de Operación
FCa	***FCb***	***MSB***	***Start***	PL_a	PL_b	
1	1	↓	1	0	1	implementa $\mathbf{t_a}$
↑	1	0	1	1	0	implementa $\mathbf{t_b}$
1	↑	0	1	1	1	implementa $\mathbf{t_0}$
1	1	↓	1	1	0	implementa $\mathbf{t_b}$
1	↑	1	1	0	1	implementa $\mathbf{t_a}$
↑	1	1	1	1	1	implementa $\mathbf{t_0}$
1	1	1	1	1	1	implementa $\mathbf{t_0}$
x	x	x	0	1	1	Apagada

x: no importa

tiempo que debe implementarse es t_a, seguido de t_b finalizando el subciclo con t_0. El flanco de bajada de ***MSB*** (fl-1) inhabilita la línea de control PL_a, y el contador Ct_a empieza la cuenta descendente implementando el intervalo de tiempo t_a. Cuando Ct_a termina, la señal de fin de cuenta ***FCa*** genera un flanco de subida (fl-2). Aquí la máquina secuencial cambia de estado, cambiando los estados de la línea de control PL_a que pasa a alto y de PL_b que pasa a bajo. En este estado de la máquina secuencial, se implementa el intervalo de tiempo t_b a través del contador Ct_b. La señal de fin de cuenta ***FCb*** de Ct_b genera un flanco de subida (fl-3) cuando Ct_b llega a cero. Luego la máquina secuencial cambia de estado, en el cual los contadores Ct_a y Ct_b permanecen detenidos reteniendo los tiempos t_a y t_b para su posterior implementación en el subciclo par. En este nuevo estado se implementa el tiempo t_0, completándose con el flanco de subida de ***MSB***.

El flanco (fl-4) define el comienzo del subciclo par, haciendo cambiar a la máquina secuencial del estado actual (t_0) al estado de implementación de t_b. Cuando se termina de implementar t_b, la señal ***FCb*** se activa haciendo que la máquina secuencial cambie al estado de implementación del tiempo t_a. En el estado t_a, la línea PL_b está en estado alto (Ct_b detenido) y PL_a esta en estado bajo (Ct_a implementa t_a). Cuando Ct_a llega a cero se activa la señal FC_a, produciéndose un cambio de estado en la máquina secuencial al estado t_0. La máquina secuencial permanece en este estado implementando el tiempo t_0, hasta que aparece

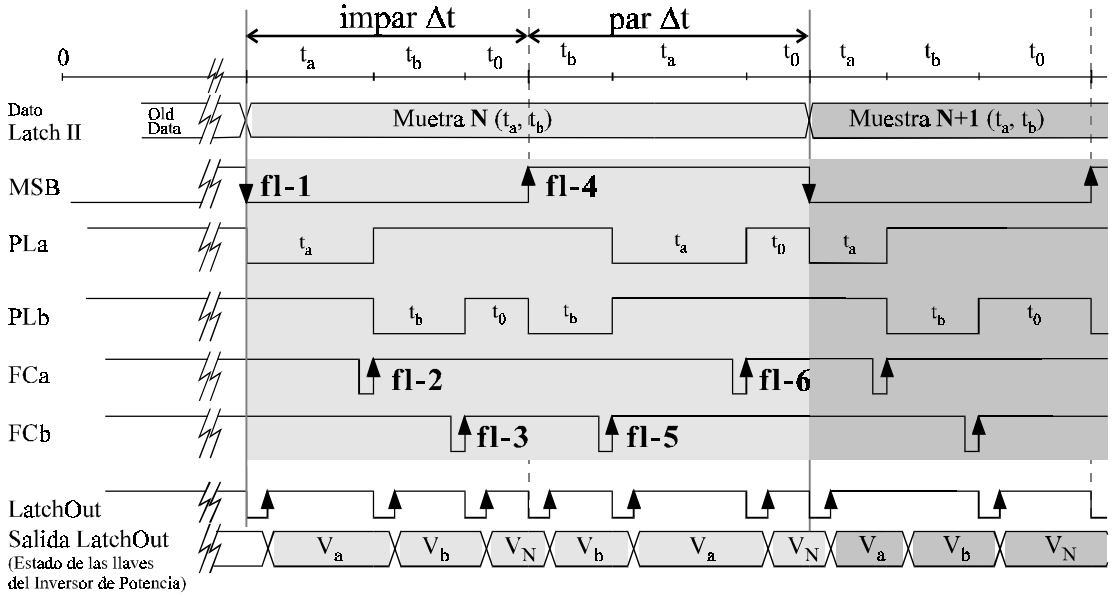


Figura 4.12: Diagrama de tiempo de las señales del modulador

el flanco de bajada de ***MSB***. Así comienza un nuevo período de conmutación, por lo tanto inicia un nuevo ciclo en la secuencia de implementación de la nueva muestra del vector de referencia.

4.3.3. Etapa de Salida

Esta etapa acondiciona las señales de excitación de las llaves del inversor, que provienen de la etapa Algoritmo Modulación Vectorial. Es una etapa importante debido a que genera los retardos necesarios para que no se produzcan cortocircuitos en las columnas del inversor. Las entradas son las señales de salida del Latch Out (***pwmS1***,..., ***pwmS6***), el cual se encuentra en la etapa Algoritmo Modulación Vectorial, y el reloj ***clk***. Las salidas son las señales ***Sw1***,..., ***Sw6***, que ingresan directamente al inversor de potencia.

La implementación del tiempo de seguridad (T_D) se realiza con registros de desplazamiento, como muestra la Fig. 4.13. Los registros de desplazamiento poseen 18 Flip-Flop para poder implementar el tiempo de seguridad deseado $T_D = 2\mu seg$.

4.3.4. Resultados de simulaciones digitales

El modulador vectorial lineal descrito en los puntos anteriores, se implementa en un CPLD EPM7128S de ALTERA. Para ello se utilizan las herramientas de software MAX+plus II, provisto por la misma compañía. Este software permite ingresar los diseños de varias

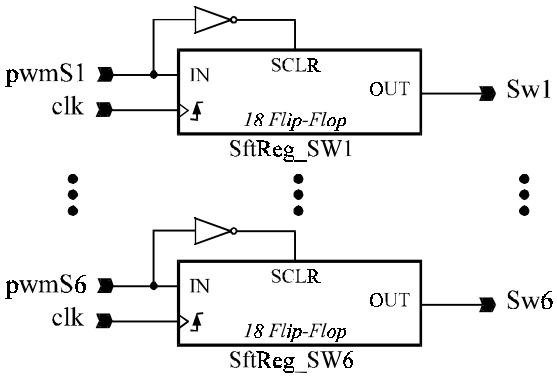


Figura 4.13: Implementación del tiempo de seguridad T_D.

maneras: por circuitos esquemáticos, por un generador de señales, o programas en AHDL (Altera Hardware Description Language) este último es el utilizado.

En la Fig. 4.14 se muestran los resultados obtenidos de una simulación de tiempo del modulador, donde se muestra la comunicación con la PC (Hand-shake). Las señales que intervienen son las correspondientes al Bus ISA de la PC (***SBHEN***, ***IORN***, ***IOWN***) y las que genera el modulador para poder comunicarse (***PC_Observa***, ***LoadN***, ***twN***, ***Enable_Load***, ***D0***, ***Seguro***). Cada semiperíodo de la señal ***MSB*** corresponde a un subciclo (Δt), por lo que un período de ***MSB*** corresponde a un período de conmutación del modulador.

Puede observarse en la Fig. 4.14 como en un ciclo de lectura se activan las señales ***SBHEN*** e ***IORN,*** dando lugar a la activación de ***PC_Observa*** la cual desactiva el modo tri-state de ***D0*** para que la PC pueda leer su estado, el cual es igual al de la señal ***Enable_Load***. También puede observarse como en un ciclo de escritura (***SBHEN*** y ***IOWN***), se activa la señal ***LoadN*** la cual efectúa el cambio al estado cero de ***Enable_Load***. Pueden observarse como la PC puede leer el estado del modulador varias veces (cuatro en la simulación de la Fig.4.14), hasta encontrar habilitada la carga de nuevos datos (cuando ***D0***=1) y entonces realizar la escritura. Por último se observa el comportamiento de la señal ***Seguro***, la cual se activa pasando a uno deteniendo el funcionamiento del modulador, después de que no se realicen cargas de muestras durante dos periodos de conmutación. En la Fig. 4.14 se ve como se carga la muestra ta=100 y tb=25, la cual se repite en la implementación debido a que no se carga una nueva muestra. Aquí puede observarse el efecto de señal ***twN*** sobre la señal ***Enable_Load***, al llegar al fin del período de conmutación (fin del segundo subciclo) en el cual no se permite la carga de muestras debido al retardo de las memorias.

Una vez detenido el funcionamiento del modulador, ya sea por acción de la señal ***Seguro*** o por enviar el dato de parada, el mismo puede reiniciarse de la misma forma: enviando un

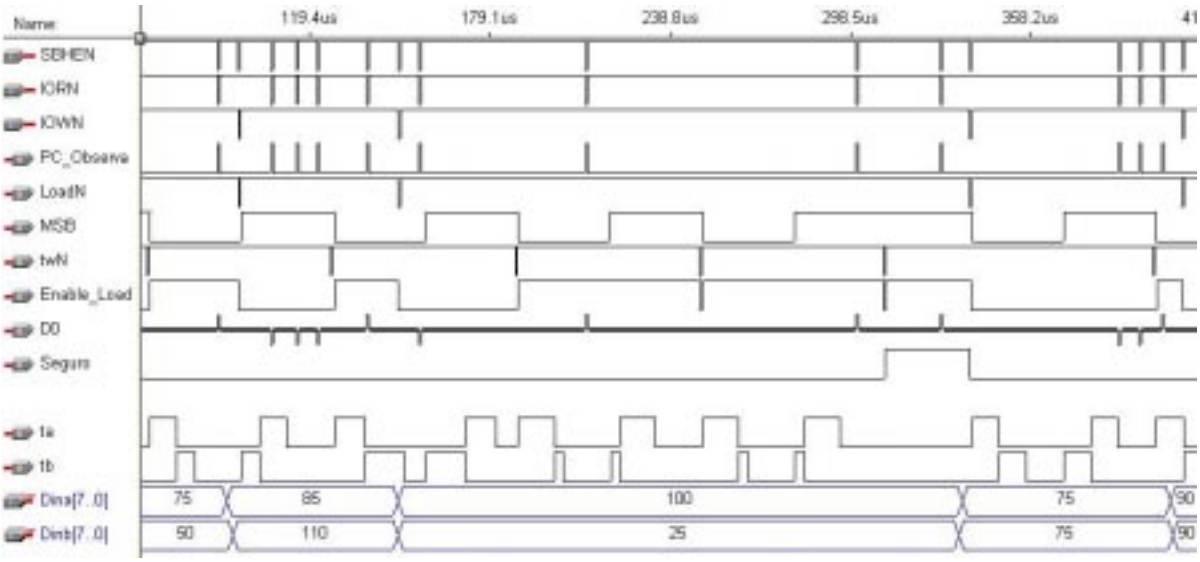


Figura 4.14: Funcionamiento de la etapa de entrada (hand-shake), obtenida con el simulador de Max+Plus II.

ciclo de escritura al modulador, como se muestra en la Fig. 4.14 en la parte en que la señal ***Seguro*** se pone en cero.

En la Fig.4.15 se muestra el funcionamiento del modulador lineal, implementando una señal de referencia de solo tres muestras por sector, con el fin de observar el comportamiento del mismo en los seis sectores.

Mientras la señal ***ta*** (***tb***) está en uno, el modulador se encuentra en el estado ta (tb) de la máquina secuencial, en donde se implementa el tiempo ta (tb). Cuando las señales ***ta*** y ***tb***

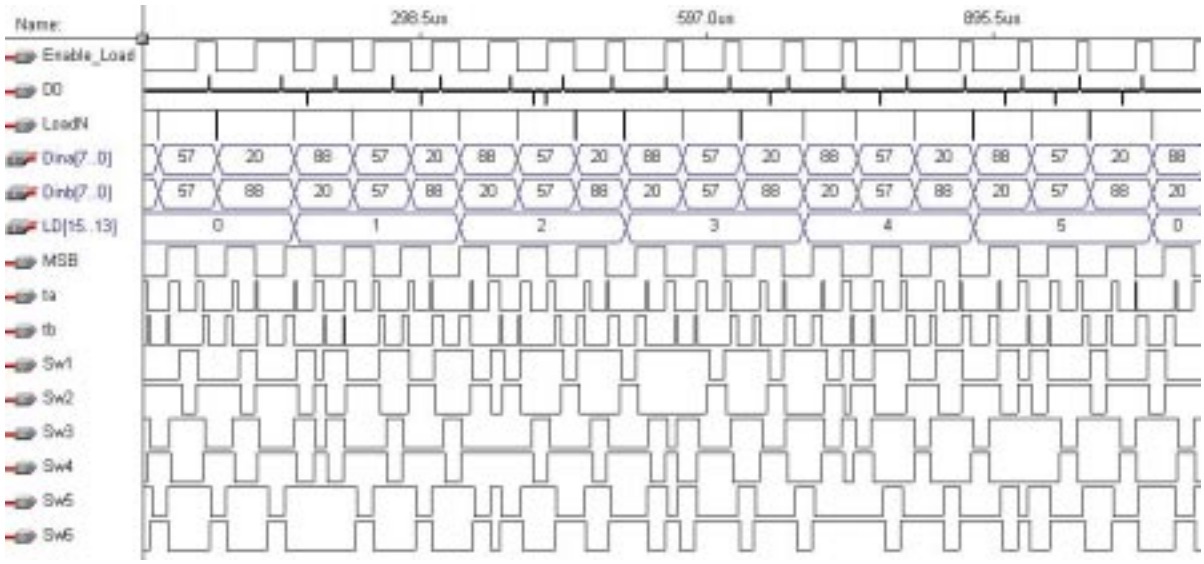


Figura 4.15: Funcionamiento del modulador lineal en los seis sectores, para una referencia de 908.76Hz (tres muestras por sector).

están en cero, el modulador implementa el tiempo t0, y se aplica el vector nulo correspondiente a las mínimas conmutaciones (Fig.4.15). Cuando las señales ***Sw1*** a ***Sw6***, que son las señales de salida del modulador, están en cero activan la llave correspondiente del inversor de potencia. Estas incluyen un tiempo de seguridad T_d = 2μseg. Las señales ***LD[15..13]*** representan al sector en donde se encuentra el vector de referencia, de acuerdo a la Tabla 4.1. Las señales ***Dina[7..0]*** y ***Dinb[7..0]***, son las salidas de las memorias ta y tb respectivamente.

El diseño del modulador completo dentro del CPLD requiere de 106 celdas lógicas de un total de 128. Se utilizan 56 de 64 pines de entrada-salida disponibles en el CPLD, para conectarla con la PC, las memorias, el inversor y algunos dispositivos adicionales para efectuar las pruebas funcionales. El porcentaje de utilización es del 86%, el cual está cerca del limite recomendado por ALTERA para el dispositivo utilizado.

4.4. Resultados experimentales

El modulador vectorial lineal se implementó en la placa de demostración de ALTERA (UP1) utilizando el dispositivo MAX7128SLC84-7, siendo este un dispositivo EEPROM de la familia MAX. Con la finalidad de probar el modulador vectorial lineal, se lo conecto al Bus ISA de la PC y a un inversor de potencia. En el ambiente de la PC se generaron diferentes vectores de referencia, los cuales giran a diferentes velocidades manteniendo la relación V/f constante. En las siguientes figuras se muestran los resultados experimentales de la parte digital y de potencia.

Los resultados experimentales de las señales de comunicación con la PC (Hand-shake) se muestran en la Fig. 4.16, en donde el CH1 (trazo superior) tiene la señal ***LoadN***, el CH2 (trazo del medio) la señal ***MSB*** y el CH3 (trazo inferior) la señal ***Enable_Load***. Observar que en el primer periodo de la señal ***MSB*** no se carga ningún dato (la señal ***LoadN*** permanece en alto), con el fin de mostrar que la señal ***Enable_load*** se mantiene en alto hasta un tiempo antes del final del subciclo en donde cambia de estado pasando a cero. En el segundo periodo de ***MSB*** se produce una carga de datos en el pulso de la señal ***LoadN***, entonces la señal ***Enable_load*** pasa al estado cero inhibiendo la carga de un nuevo dato. Al finalizar el subciclo del segundo periodo de MSB, la señal ***Enable_load*** pasa al estado alto habilitando la carga de un nuevo dato. Luego de un tiempo, la PC lee ese estado y al estar habilitada produce la carga de un nuevo dato.

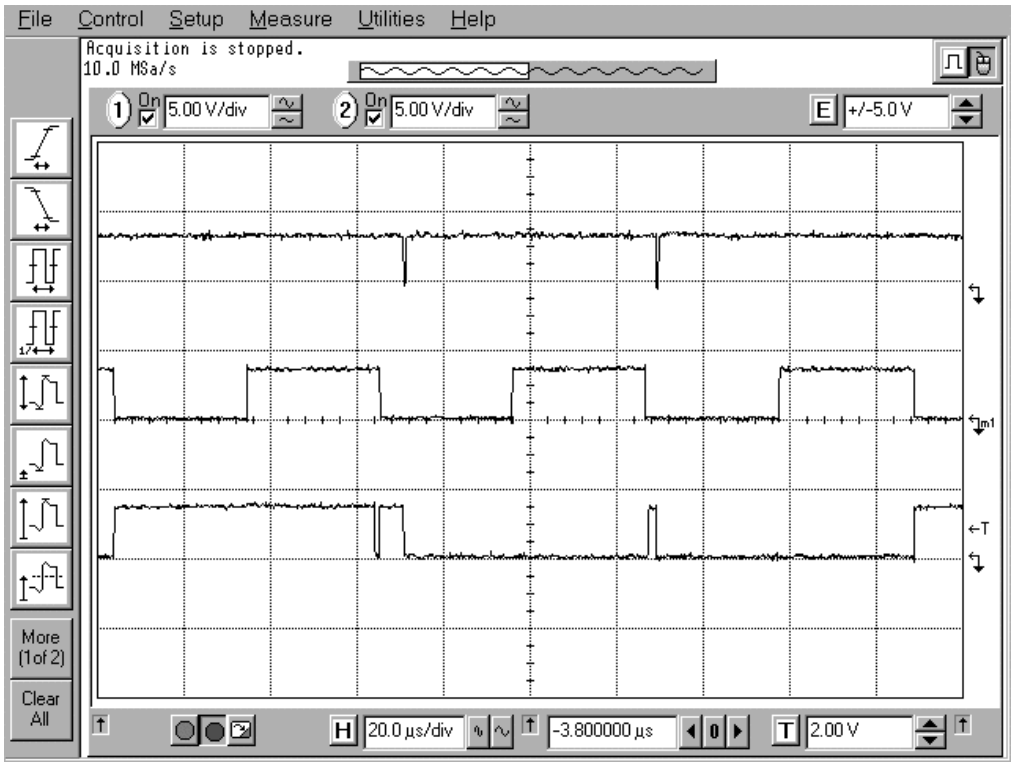


Figura 4.16: Hand shake entre el Modulador y la PC.

CH1 (arriba): ***LoadN***

CH2 (centro): ***MSB***

CH3 (abajo): ***Enable_Load***

En la Fig. 4.17 se muestran las siguientes señales del Bus ISA, ***Clk*** (Clock del Bus), ***SBHEN***, ***IOWN*** y ***LoadN***. Puede observarse un ciclo de escritura de datos, mostrándose indirectamente la transferencia de datos en 16 bits con la activación de la señal ***SBHEN*** y la generación de la señal ***LoadN*** que produce la retención de los datos en el registro Latch IN.

La excitación de las llaves del inversor poseen el tiempo de seguridad (T_D), el cual se implementa en el modulador vectorial lineal. En la Fig. 4.18 se muestran las señales de excitación para una columna del inversor, en donde puede apreciarse que el encendido de cada transistor esta retardado 2.83 µseg respecto del apagado del otro transistor.

En la Fig 4.19 se muestran las tensiones de columna y las corrientes de carga, para una referencia de 60Hz con un índice de modulación m = 0.86. En la Fig. 4.19.a se muestran las tensiones para dos columnas del inversor. Se midió el período de conmutación resultando igual a 61.28 µseg, el cual corresponde a una frecuencia de conmutación de 16.3 Khz. La Fig. 4.19.b muestra las corrientes de cada fase. Es evidente la diferencia de fase de 120º, y la buena forma sinusoidal de la corriente.

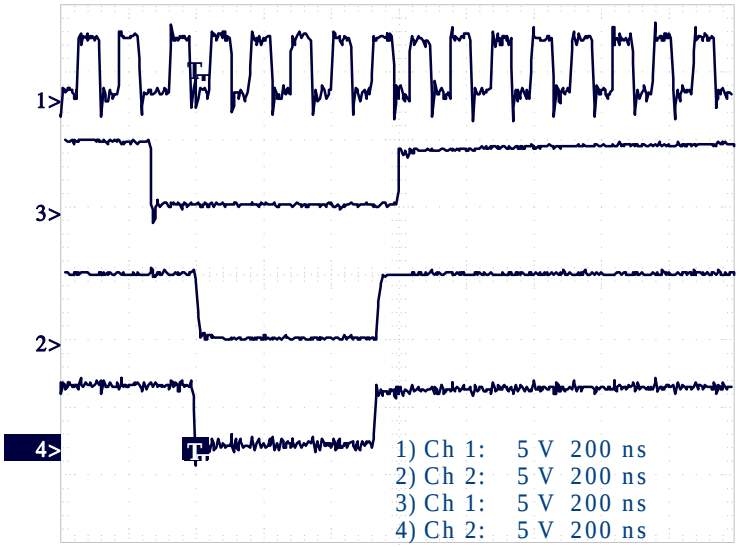


Figura 4.17: Señales del bus ISA.

CH1: ***Clk*** *- Clock Bus (8.375 Mhz)*

CH2: ***IOWN***

CH3: ***SBHEN***

CH4: ***LoadN***

En la Fig 4.20 se muestran las tensiones de columna y corrientes de carga, para una referencia de 10 Hz y m=0.15. Puede observarse como se mantiene la calidad de las formas de onda.

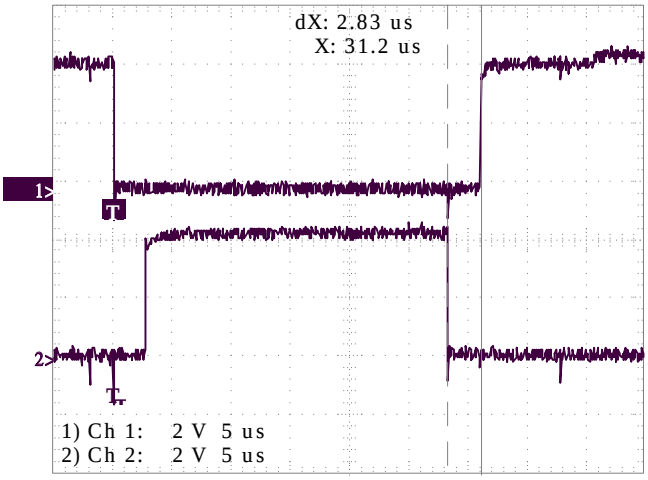
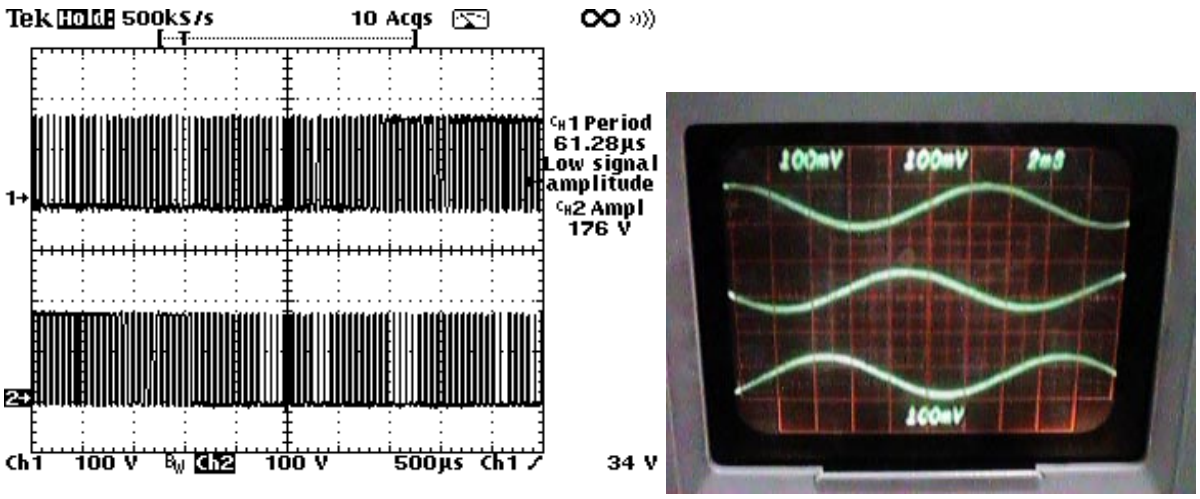


Figura 4.18: Señales de excitación de una columna del inversor.

CH1: IGBT superior

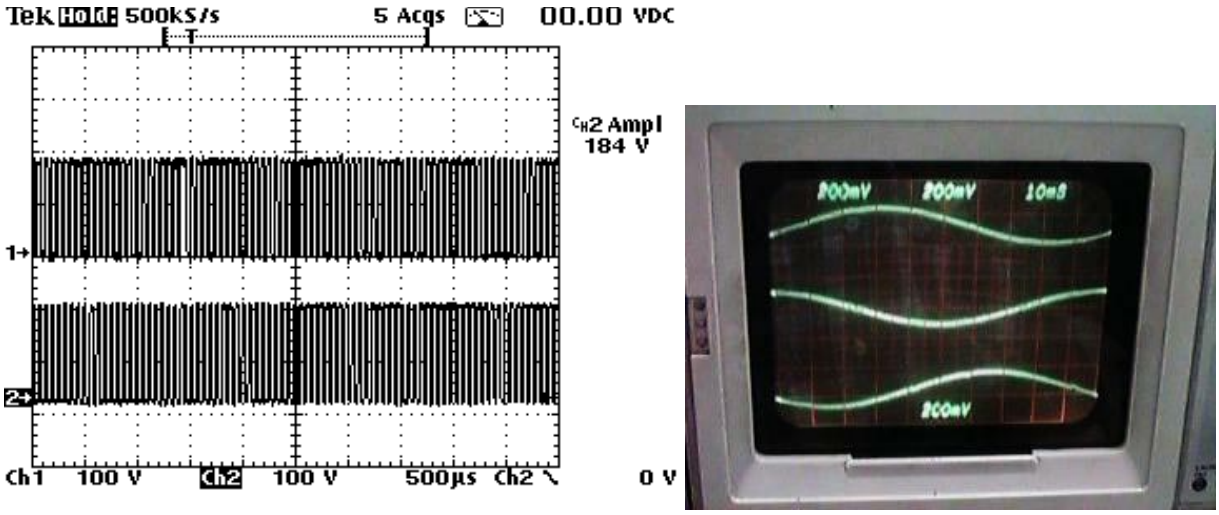
CH2: IGBT inferior



a) *b)*

Figura 4.19: Mediciones para una referencia de f_0 =60Hz y m = 0,86.

a) Tensión de dos columnas. b) Corrientes de estator.



a) *b)*

Figura 4.20: Mediciones para una referencia de f_0 =10Hz y m=0,15

a) Tensión de dos columnas. b) Corrientes de estator.

4.5. Resumen

En este capítulo se dieron los puntos básicos a tener en cuenta para el diseño de un modulador vectorial, se desarrolló la implementación de un modulador vectorial lineal.

El modulador vectorial lineal se implementa utilizando dos EPROM y un FPLD de Altera. En las memorias se almacenan los tiempos de activación de las llaves del inversor, y en el FPLD se implementa la lógica del algoritmo del modulador junto con las etapas de comunicación con la PC y del Inversor de Potencia, incluyendo la implementación del tiempo de seguridad T_D. El modulador vectorial lineal encuentra aplicaciones tanto en filtros activos como en driver para el control de motores. Éste aprovecha el 90% del bus de continua y no incluye compensación de tiempo muerto.

Se muestran resultados de simulación digital mostrando el funcionamiento del mismo y resultados experimentales que ilustran su buen desempeño.

5. SOBREMODULACIÓN

5.1. Introducción

La prestación de los métodos de modulación de ancho de pulso está caracterizada por el índice de modulación, la frecuencia de conmutación, y la distorsión armónica. El índice de modulación (m) está normalizado con la tensión fundamental del modo de operación six-step, como se definió en el punto en 2.2.3 y que se repite aquí por comodidad:

$$m = u_{j01} / (2V_{cc}/\pi) \qquad j = R, S, T \tag{5.1}$$

donde u_{j01} es la tensión fundamental de una de las columnas del inversor generada por el modulador y $2V_{cc}/\pi$ es la tensión fundamental de salida para el funcionamiento en six-step. Por lo tanto $0 \leq m \leq 1$, donde el máximo índice de modulación se alcanza solo en operación six-step y es por definición igual a uno.

La relación entre la tensión fundamental de la salida y la tensión de referencia en un inversor con PWM es lineal, hasta que la amplitud de la tensión de referencia alcanza un determinado límite [Hol 93a]. En el caso de la modulación sinusoidal (SPWM) la tensión fundamental de salida está dada por la expresión $V_{ref}/V_{tri}*V_{cc}/2$ [Bow 75b], entonces cuando la amplitud de la referencia de tensión (V_{ref}) supera a la amplitud de la triangular (V_{tri}) empieza a desaparecer la intersección de la moduladora con la portadora triangular como se muestra en la Fig. 5.1, por lo que comienzan a perderse pulsos de tensión. A pesar de que en

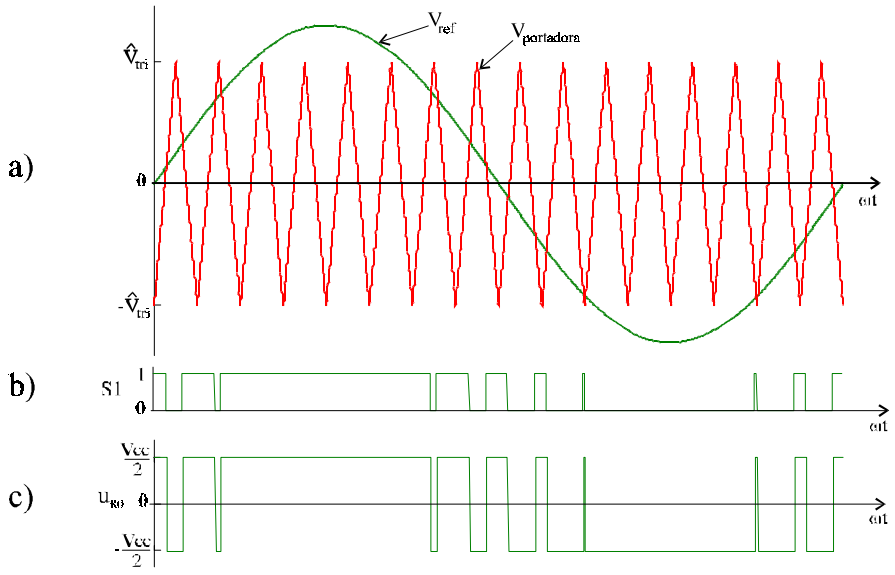


Figura 5.1: Sobremodulación en SPWM: a) Tensión de la portadora triangular y la moduladora, b) Señal de la llave S1 del inversor, c) Tensión de columna

esos intervalos la llave superior (S1) opera en el 100% del ciclo de trabajo no puede igualar la tensión de referencia. Cuando la tensión de referencia supera la amplitud de la triangular se pierde la linealidad entre ésta y la fundamental de salida y también se generan armónicas de baja frecuencia. Este efecto aumenta hasta saturar en el modo six-step. La misma condición ocurre en el semi período negativo. Se denomina sobremodulación al rango de operación nolineal del modulador; y la zona o región de sobremodulación es el rango de tensión que existe entre el comienzo de la nolinealidad y el funcionamiento en six-step [Hol 94c].

El método de modulación sinusoidal alcanza el límite máximo de linealidad cuando la amplitud de la tensión fundamental es $V_{cc}/2$ (Fig 5.1); a partir de (5.1) el máximo índice de modulación resulta $m_{subarmónico} = 0.785$. El índice de modulación puede incrementarse inyectando armónicas triples (componentes de secuencia cero) [Buj 75]. La inserción de armónicas triples modifica la tensión de columna, haciéndola más plana y con su valor pico menor que la amplitud de la componente fundamental de tensión de salida del inversor. Inyectando el 25% de armónicas triples, también se logra minimizar la distorsión armónica [Bow 97].

Utilizando la modulación vectorial, como se vio en el capítulo 2, el intervalo de tiempo t_0 decrece con el aumento del índice de modulación. Este tiempo es igual a cero ($t_0 = 0$) en (2.13), cuando la trayectoria circular del vector de referencia de tensión se hace tangente a los lados del hexágono. Por lo tanto, la relación lineal entre el vector de referencia de tensión y la componente fundamental de tensión de salida del inversor termina en este punto. En este caso, se tiene el módulo máximo del vector de referencia igual a $V_{cc}/\sqrt{3}$, que corresponde a un índice de modulación $m_{lineal\text{-}max} = 0.907$ [Hol 93a]. Se concluye que el uso de la modulación vectorial incrementa de manera natural, el aprovechamiento del Bus de continua en aproximadamente el 15% respecto de la modulación sinusoidal, como se vio en el punto 2.3.

La zona gris en la Fig. 5.2.a corresponde a la operación lineal del modulador. En esa figura también se muestra, con línea llena la componente fundamental de tensión de salida en el modo six-step y en línea de trazos el hexágono. Puede observarse que la zona lineal se extiende hasta la circunferencia inscripta dentro del hexágono. Cuando el vector de referencia de tensión se mueva sobre dicha circunferencia se tendrá el índice de modulación $m = m_{lineal\text{-}max} = 0.907$, siendo menor al dado en el modo six-step. Puede aprovecharse la zona comprendida entre el hexágono y la circunferencia inscripta en él (Fig. 5.2.a), haciendo que el vector de salida promedio se modifique a medida que avanza el vector de referencia de

tensión. Por lo tanto, cuando el índice de modulación supera a $m_{lineal\text{-}max} = 0.907$, se está en la zona de sobremodulación.

En la modulación vectorial, la sobremodulación está dividida en dos modos de operación. El modo I comprende la zona que existe entre la circunferencia inscripta en el hexágono y el hexágono en el plano complejo (Fig. 5.2.b). En este modo, la componente fundamental de la tensión de salida del inversor se mueve a la misma velocidad que el vector de referencia de tensión. Si bien, el vector modificado se desplaza por una trayectoria continua, su módulo no es constante (Fig. 5.2.b); por ello se está operando en la zona no lineal del modulador. El modo II presenta una trayectoria discontinua formada por segmentos de los lados del hexágono y periodos en los vértices del mismo (Fig. 5.2.c). A medida que aumenta el índice de modulación, el tamaño de los segmentos disminuye y el vector modificado permanece más tiempo en los vértices del hexágono. Cuando m = 1 la longitud de los segmentos es nula y el vector modificado está todo el tiempo en los vértices del hexágono, alcanzándose la operación del modo six-step. La trayectoria del vector de salida promedio (vector modificado) es discontinua, saltando desde un vértice del hexágono al segmento del lado, para luego volver a saltar al siguiente vértice.

La operación en modo I provee una reducida distorsión armónica total [Hol 93a], aunque está basada en una señal de referencia distorsionada. En el modo II la distorsión armónica se incrementa hasta alcanzar la distorsión que produce el funcionamiento en six-step [Hol 93a].

En este capítulo se analizará la sobremodulación de la modulación vectorial y se dará un esquema para su implementación.

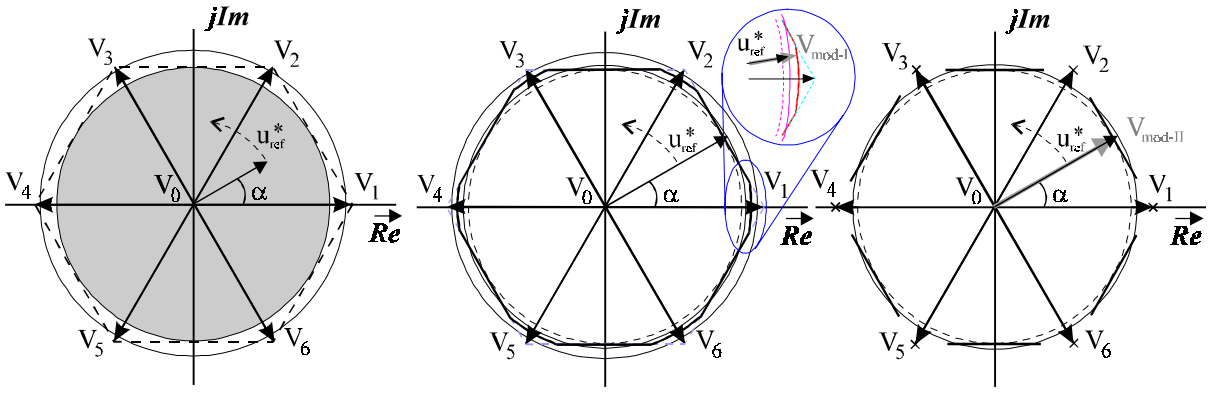


Figura 5.2: Modulación vectorial: a) Límite entre zona lineal y sobremodulación, b) Modo I de sobremodulación, c) Modo II de sobremodulación.

5.2. Análisis de los modos de sobremodulación de la modulación vectorial

5.2.1. Sobremodulación Modo I (0.907 < m ≤ 0.955)

El modo I de sobremodulación, comprende la zona que existe entre la circunferencia inscripta dentro del hexágono y el hexágono en el plano complejo (Fig. 5.3). Por tal motivo, el vector de salida promedio se modifica siguiendo una trayectoria continua que cambia entre circular y lineal. La Fig. 5.3.a muestra la trayectoria en el plano complejo del vector modificado, y la Fig. 5.3.b muestra la componente real en función de la fase. El ángulo α_{NL} es el ángulo de referencia medido desde el eje real hasta la intersección de la circunferencia con el lado del hexágono. Este sirve para calcular el módulo del vector espacial modificado $V_{mod\text{-}I}$, cuando se mueve por el tramo circular. Para una dada tensión de salida del inversor se tendrá una dada trayectoria continua en el plano complejo, la cual estará recorrida por el vector modificado. Esta trayectoria estará compuesta por tramos lineales y tramos circulares, donde el límite entre éstos queda definido por el ángulo α_{NL}.

La forma de onda de la tensión en función de la fase, se divide en siete segmentos (Fig. 5.3.b). El primer segmento denominado f_1, corresponde a la proyección del vector modificado que se mueve por el arco de la circunferencia en el intervalo $0 \leq \theta < \alpha_{NL}$, como muestra la Fig. 5.3.a. Durante ese y todos los demás segmentos en el cual el vector

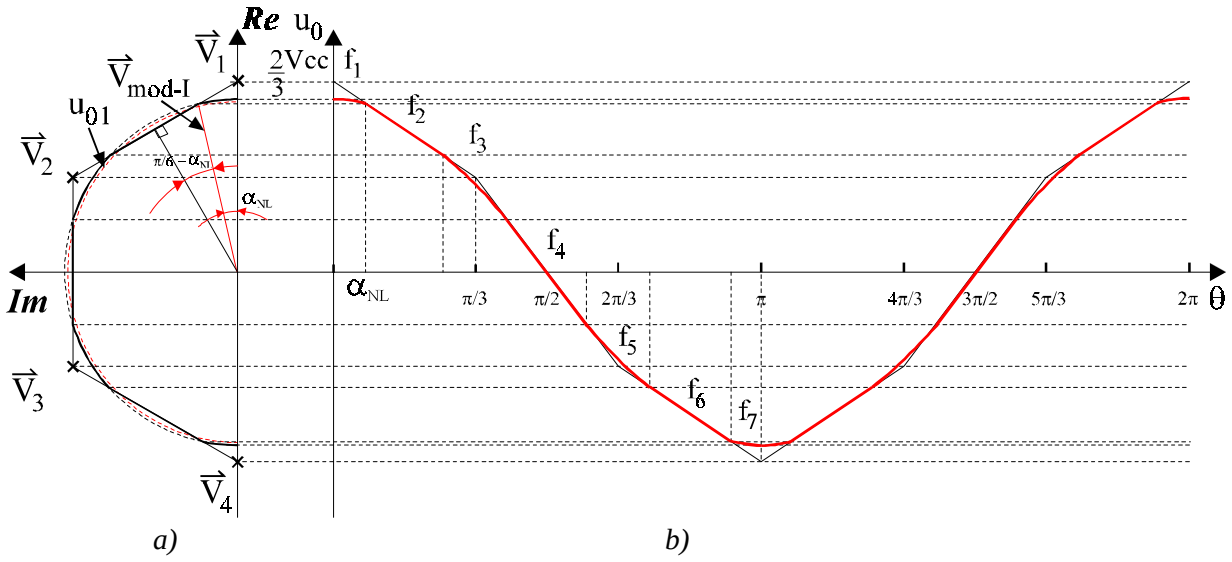


Figura 5.3: Modo I de sobremodulación: a) representación en el plano complejo, b) Tensión de salida promedio en función de la fase.

modificado se mueve por un arco de circunferencia, los tiempos de activación de las llaves se calculan con (2.13). Los segmentos f_2, f_4 y f_6 son la proyección del vector modificado cuando éste se mueve sobre el hexágono. En estos intervalos los vectores nulos no se utilizan, por lo tanto el tiempo $t_0 = 0$ y los tiempos de los vectores activos se calculan con (2.10). Aquí se repiten las ecuaciones de los tiempos para mayor comodidad.

$$\begin{cases} t_a = \sqrt{3}\ \dfrac{|\vec{V}|}{V_{cc}} \Delta t \sin(\pi/3 - \alpha) \\ t_b = \sqrt{3}\ \dfrac{|\vec{V}|}{V_{cc}} \Delta t \sin(\alpha) \qquad \text{donde} \quad 0 \le \alpha < \alpha_{NL}\,; \quad \dfrac{\pi}{3} - \alpha_{NL} \le \alpha < \dfrac{\pi}{3} \\ t_0 = \Delta t - t_a - t_b \end{cases} \tag{5.2.a}$$

$$\begin{cases} t_a = \Delta t\ \dfrac{sin(\frac{\pi}{3} - \alpha)}{sin(\frac{\pi}{3} + \alpha)} \qquad \text{donde} \qquad \alpha_{NL} \le \alpha < \dfrac{\pi}{3} - \alpha_{NL} \\ t_b = \Delta t - t_a \end{cases} \tag{5.2.b}$$

Las ecuaciones en cada segmento son:

$$f_j = \vec{V}_{mod-I}\, cos(\theta) \quad j = 1,3,5,7 \qquad \begin{aligned} f_1&: \ 0 \le \theta < \alpha_{NL} \\ f_3&: \ \tfrac{\pi}{3} - \alpha_{NL} \le \theta < \tfrac{\pi}{3} + \alpha_{NL} \\ f_5&: \ \tfrac{2\pi}{3} - \alpha_{NL} \le \theta < \tfrac{2\pi}{3} + \alpha_{NL} \\ f_7&: \ \pi - \alpha_{NL} \le \theta < \pi \end{aligned} \tag{5.3.a}$$

$$f_2 = -\frac{Vcc}{\pi}\theta + \frac{2Vcc}{3} \qquad \alpha_{NL} \le \theta < \frac{\pi}{3} - \alpha_{NL} \tag{5.3.b}$$

$$f_4 = -\frac{2Vcc}{\pi}\theta + Vcc \qquad \frac{\pi}{3} + \alpha_{NL} \le \theta < \frac{2\pi}{3} - \alpha_{NL} \tag{5.3.c}$$

$$f_6 = -\frac{Vcc}{\pi}\theta + \frac{Vcc}{3} \qquad \frac{2\pi}{3} + \alpha_{NL} \le \theta < \pi - \alpha_{NL} \tag{5.3.d}$$

donde $\theta = \omega t$ y ω es la velocidad angular de la componente fundamental del vector de referencia de tensión. La ecuación (5.4) da el valor del módulo del vector modificado cuando se mueve por el arco de circunferencia en la zona de los vértices (Fig. 5.3.a). Este valor debe reemplazar a $|\vec{V}|$ en (5.2.a) para calcular los tiempos t_a, t_b y t_0.

$$\vec{V}_{mod-I} = \frac{Vcc}{\sqrt{3}} \frac{1}{cos(\frac{\pi}{6} - \alpha_{NL})} \tag{5.4}$$

Haciendo el desarrollo en serie de Fourier de la (5.3) y utilizando (5.4), se obtienen las componentes de baja frecuencia de la tensión de salida del inversor; pues en (5.3) nos se tuvo en cuenta la acción de conmutación sino el valor promedio en cada ciclo. La componente fundamental de la tensión de salida del inversor para el modo I resulta,

$$u_{o1} = \frac{2}{\pi} Vcc \left[\frac{\sqrt{3}\,\alpha_{NL}}{cos(\alpha_{NL} - \frac{\pi}{6})} + \left(\frac{3}{\pi} + \frac{3\sqrt{3}}{\pi} \alpha_{NL} \right) cos(\alpha_{NL}) + \left(\frac{3}{\pi} \alpha_{NL} - 2 - \frac{3\sqrt{3}}{\pi} \right) sin(\alpha_{NL}) \right] \quad (5.5)$$

La (5.5) es función del ángulo α_{NL} y su valor es máximo cuando $\alpha_{NL} = 0$ y el vector modificado se desplaza sobre el hexágono. El valor mínimo es cuando $\alpha_{NL} = \pi/6$ y el vector modificado coincide con el de referencia y se mueve por la circunferencia inscripta en el hexágono. En la Fig. 5.4 se muestra la componente fundamental de la tensión de salida y el índice de modulación en función del ángulo α_{NL}. De (5.5) puede despejarse el índice de modulación máximo para el modo I,

$$m = m_{Modo\ I} = \frac{3}{\pi} = 0.955 \qquad \text{cuando} \qquad \alpha_{NL} = 0 \quad (5.6)$$

Como la referencia es el vector de tensión, la forma de proceder para calcular los tiempos es a la inversa de cómo muestra la Fig. 5.4. O sea que se ingresa con el vector de referencia de tensión (o el índice de modulación) y se obtiene el ángulo α_{NL}. Con este valor puede calcularse (5.4), y los tiempos de activación de las llaves del inversor para el tramo de trayectoria circular, quedando también definidos los tiempos para el tramo lineal.

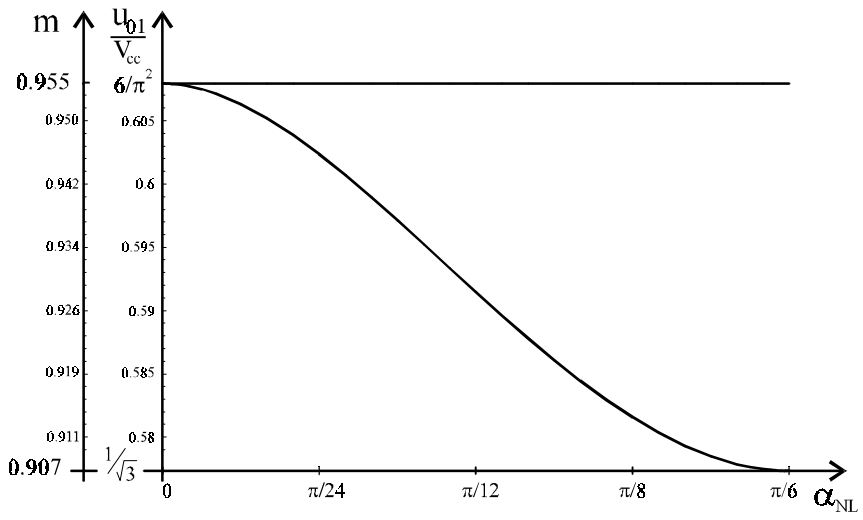


Figura 5.4: Índice de modulación en función del ángulo α_{NL}, para el modo I de sobremodulación.

5.2.2. Sobremodulación Modo II (0.955 < m ≤ 1)

Para obtener una componente fundamental en la tensión de salida mayor que la dada por el modo I, el vector modificado se mantiene durante un intervalo de tiempo en el vértice del hexágono, para luego saltar al hexágono y desplazarse por el mismo. Este modo de operación se denomina como modo II de sobremodulación.

En el modo I, el vector modificado y la componente fundamental de tensión de salida del inversor se mueven con igual velocidad. En cambio, en el modo II si bien la componente fundamental de la tensión de salida y el vector de referencia se mueven con igual velocidad, el vector modificado se mueve a un ritmo discontinuo. En la Fig. 5.5 se muestran las distintas ubicaciones entre el vector modificado y la componente fundamental de tensión de salida del inversor. Cuando el vector modificado se encuentra en un vértice, la componente fundamental que gira a velocidad constante empieza a adelantarse al mismo (Fig. 5.5.a). En la Fig. 5.5.b la componente fundamental y el vector modificado se desplazan a igual velocidad, hasta que el vector modificado salta al vértice sucesivo adelantándose al vector de tensión fundamental (Fig. 5.5.c).

En el modo II de sobremodulación, a diferencia del modo I, el vector modificado se mueve por trayectorias discontinuas (Fig. 5.6). El ángulo de retención α_H controla el intervalo de tiempo en que el vector modificado permanece en el vértice del hexágono. Entonces a medida que α_H aumenta, crece la componente fundamental de tensión de salida del inversor. La Fig. 5.6 muestra el funcionamiento del modo II en sobremodulación. En Fig. 5.6.a se detalla una trayectoria del vector modificado en el plano complejo, y en Fig. 5.6.b se muestra la componente real de la tensión en función de la fase. Para el análisis de este modo de operación se procederá de la misma manera que en el modo I, es decir que se buscará la componente fundamental de la tensión de la Fig. 5.6.b.

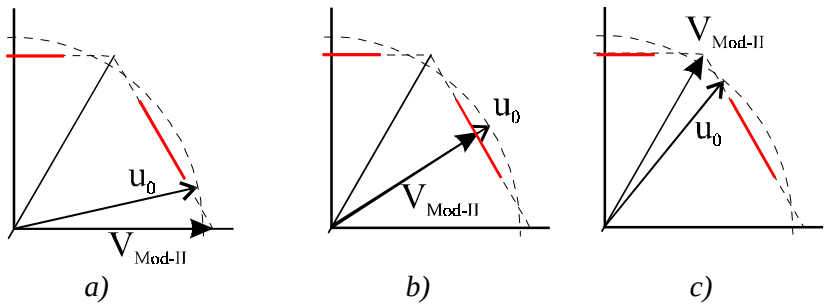
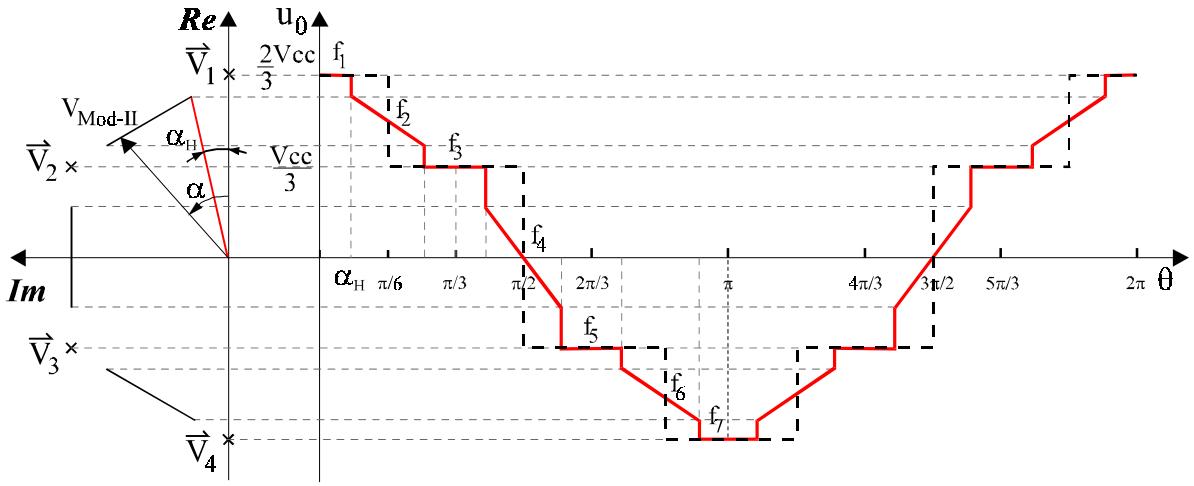


Figura 5.5: Descripción del modo II de sobremodulación



a) *b)*

Figura 5.6: Modo II de sobremodulación: a) representación en el plano complejo, b) Tensión de salida promedio en función de la fase.

La componente real del vector modificado en función del tiempo, se divide en siete segmentos (f_1...f_7). Los segmentos impares corresponden a la proyección del vector modificado cuando éste está detenido en un vértice del hexágono. Los segmentos pares son la proyección cuando el vector modificado se mueve por el hexágono. Cuando el vector modificado está detenido en un vértice, el tiempo en que se aplica el vector espacial correspondiente al vértice es igual al subciclo Δt. Éste se repite mientras el vector modificado permanezca en el vértice del hexágono. Por otro lado, cuando el vector modificado se mueve por el lado del hexágono, el tiempo en que se aplican los vectores espaciales activos están dados por (5.2.b).

De la Fig. 5.6.b, las expresiones de los siete segmentos resultan:

$$f_1 = \frac{2}{3}Vcc \qquad 0 \le \theta < \alpha_H \qquad (5.7.a)$$

$$f_2 = -\frac{Vcc}{\pi}\theta + \frac{2Vcc}{3} \qquad \alpha_H \le \theta < \frac{\pi}{3} - \alpha_H \qquad (5.7.b)$$

$$f_3 = \frac{Vcc}{3} \qquad \frac{\pi}{3} - \alpha_H \le \theta < \frac{\pi}{3} + \alpha_H \qquad (5.7.c)$$

$$f_4 = -\frac{2Vcc}{\pi}\theta + Vcc \qquad \frac{\pi}{3} + \alpha_H \le \theta < \frac{2\pi}{3} - \alpha_H \qquad (5.7.d)$$

$$f_5 = -\frac{Vcc}{3} \qquad \frac{2\pi}{3} - \alpha_H \le \theta < \frac{2\pi}{3} + \alpha_H \qquad (5.7.e)$$

$$f_6 = -\frac{Vcc}{\pi}\theta + \frac{Vcc}{3} \qquad \frac{2\pi}{3} + \alpha_H \le \theta < \pi - \alpha_H \qquad (5.7.f)$$

$$f_7 = -\frac{2}{3}Vcc \qquad \pi - \alpha_H \le \theta < \pi \qquad (5.7.g)$$

Desarrollando en serie de Fourier las (5.7) y tomando la componente fundamental, se obtiene la tensión de salida del inversor para el modo II, la cual es función del ángulo de retención α_H.

$$u_{01} = Vcc\frac{6}{\pi^2}\left[\left(1+\sqrt{3}\alpha_H\right)\cos(\alpha_H) - \left(\sqrt{3}-\alpha_H\right)\sin(\alpha_H)\right] \qquad (5.8)$$

La máxima tensión de salida ocurre cuando $\alpha_H = \pi/6$ y coincide con la tensión del modo six-step, dando el índice de modulación m = 1. Cuando $\alpha_H = 0$, el vector modificado se mueve de manera continua por el hexágono, dando el índice de modulación $m_{MODO\ I} = 0.955$ que corresponde a la máxima tensión de salida que se puede obtener en el modo I. En la Fig. 5.7 se muestra la variación del índice de modulación y de la tensión de salida en función de α_H. Con el modo I se llega al 95% del aprovechamiento del bus de continua. Para cubrir el 5% restante, el modulador debe poder operar en el modo II de sobremodulación; el cual requiere que pueda alcanzar el modo six-step.

Como ocurre en el modo I, debido a que la referencia es la tensión de salida, para calcular los tiempos de activación de las llaves se debe calcular el ángulo de retención (α_H); por lo que debe procederse de manera inversa a como muestra la Fig. 5.7. Se ingresa con el vector de referencia de tensión o el índice de modulación y se obtiene α_H.

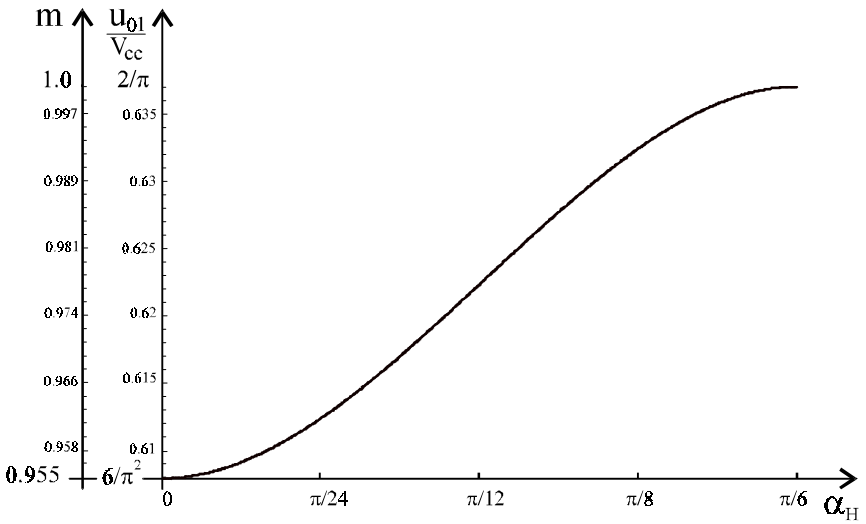


Figura 5.7: Índice de modulación en función del ángulo de retención α_H, para el modo II de sobremodulación.

5.3. Implementación de la sobremodulación

En este punto se desarrollará la incorporación de la sobremodulación al modulador vectorial lineal presentado en el capítulo 4. El principio básico del modulador es el control de los intervalos de tiempo t_a, t_b y t_0 en que se aplican diferentes vectores espaciales. En la Fig. 5.8 se muestra un diagrama de bloques con las entradas y salidas del modulador vectorial genérico. Las entradas corresponden a la tensión de columna pico ($\overline{V}_{ref}$) normalizada con la tensión del bus de continua (V_{cc}), el ángulo (α) que determina la posición del vector de referencia en el sector genérico ($0° \le \alpha < 60°$), y la información del sector particular. Para el modulador lineal del capítulo 4, los datos de las ROMs están dados por (2.13) y la máxima tensión de salida normalizada con V_{cc} es de $1/\sqrt{3} = 0.57735$. Por lo tanto no se puede alcanzar el máximo índice de modulación (m = 1), correspondiente al valor de tensión de salida normalizada $2/\pi = 0.6366$.

Para que el modulador alcance el índice de modulación unitario, se le incorporará el funcionamiento en la zona de sobremodulación. Entonces la entrada $V_{refN} = \overline{V}_{ref}/V_{cc}$ tomará valores comprendidos entre 0 y $2/\pi$, permitiendo que m varíe entre 0 y 1. El esquema del modulador lineal presentado en el capítulo 2 (Fig. 2.9) sigue siendo válido para la operación en sobremodulación, solo debe alterarse la máquina secuencial y los intervalos de tiempo que se guardarán en las ROMs.

En el modulador lineal, las ROMs están direccionadas por el módulo y el ángulo, comprendido en el primer sector, del vector de referencia. En este caso el módulo del vector de referencia coincide con la tensión de salida. Al incorporar sobremodulación el módulo del vector espacial de referencia no coincide con la componente fundamental de tensión de salida, como se vio en las secciones anteriores. La dirección de las ROMs se forma con la tensión de columna pico normalizada (V_{refN}) y con el ángulo (α). La parte alta de la dirección

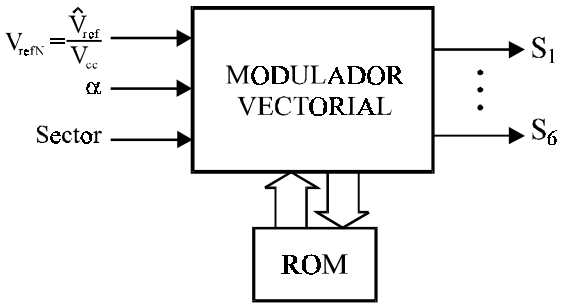


Figura 5.8: Señales de entradas y salidas del modulador vectorial genérico

está compuesta por la tensión V_{refN} y la parte baja por el ángulo α. Los datos en la ROMs se agrupan en segmentos, donde cada segmento corresponde a un valor de V_{refN}. Dentro de cada segmento, los tiempos t_a y t_b se ubican de acuerdo al valor del ángulo. La dirección más baja dentro del segmento, corresponde a $\alpha = 0°$ y la más alta a $\alpha = 60°$. En la Fig. 5.9 se muestra de forma genérica, como se guardarán los intervalos de tiempo en cada segmento de memoria. En la zona lineal ($0 \leq V_{refN} < 1/\sqrt{3}$) se utiliza (5.2.a) para el cálculo de los intervalos de tiempo, como muestra la Fig. 5.9.a. En la zona de sobremodulación del Modo I (Fig. 5.9.b), para los valores del ángulo α menores a α_{NL} y mayores a $60°-\alpha_{NL}$ se utiliza (5.2.a), para los demás valores de α se utiliza (5.2.b). Para el Modo II cuando los valores de α están comprendidos entre α_H y $60°- \alpha_H$ se utiliza (5.2.b) para el cálculo de t_a y t_b; si $\alpha < \alpha_H$ entonces $t_a = \Delta t$ y $t_b = 0$, y si $\alpha > 60°- \alpha_H$ entonces $t_a = 0$ y $t_b = \Delta t$ (Fig. 5.9.c). En la Fig. 5.9.d se muestra el segmento correspondiente a la operación en six-step, que es el límite de la operación en el Modo II de sobremodulación.

La máquina secuencial se encarga de controlar el encendido y apagado de los contadores que implementan los intervalos de tiempo t_a y t_b, como en el caso del modulador lineal (Fig. 2.10). Cuando se implementa la sobremodulación la máquina secuencial debe

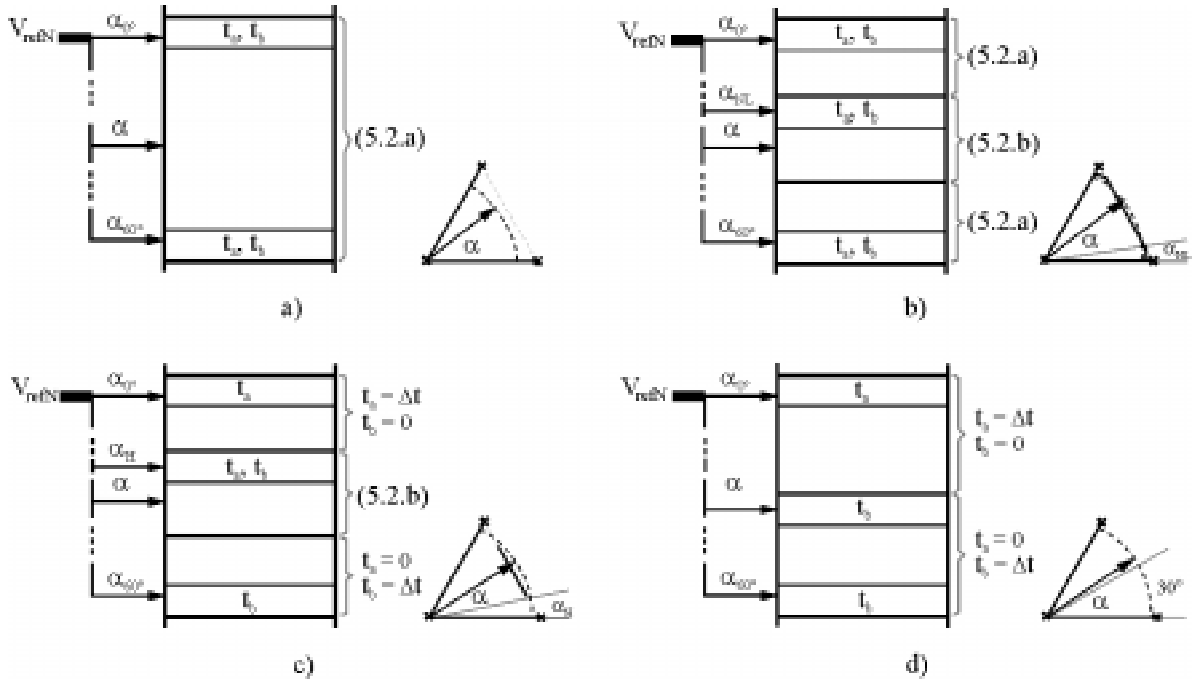


Figura 5.9: Descripción genérica de los tiempos almacenados en cada segmento de las ROMs, para cada modo de operación. a) Modo Lineal, b) Sobremodulación Modo I, c) Sobremodulación Modo II, y d) Six-Step

poder implementar las siguientes combinaciones de secuencias:

Modo Lineal: $...|\underbrace{t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-}_{0^o\le\alpha<60^o}|...|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|...$

Modo I: $...|\underbrace{t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-}_{0^o\le\alpha<\alpha_{NL}}|\underbrace{t_a\ t_b\ t_b\ t_a}_{\alpha_{NL}\le\alpha<60^o-\alpha_{NL}}|...|t_a\ t_b\ t_b\ t_a||\underbrace{t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-}_{60^o-\alpha_{NL}\le\alpha<60^o}|...$

Modo II: $...\underbrace{|t_a\ t_a|}_{0^o\le\alpha<\alpha_H}...|t_a\ t_a||\underbrace{t_a\ t_b\ t_b\ t_a}_{\alpha_H\le\alpha<60^o-\alpha_H}|...|t_a\ t_b\ t_b\ t_a|\ \underbrace{|t_b\ t_b|}_{60^o-\alpha_H\le\alpha<60^o}...|t_b\ t_b||t_a\ t_a|...$

En la Fig. 5.10 se muestra un esquema de la máquina secuencial, con los estados y las señales que permiten cambiar de un estado a otro en sincronismo con el clock. Cada estado define una salida de la máquina secuencial; los estados S0 y S3 tienen como salida los vectores nulos y los estados S1 y S2 corresponde a los vectores espaciales genéricos V_a y V_b respectivamente.

Cuando el modulador se pone en marcha, la máquina secuencial se posiciona en el estado S0. Para implementar una muestra del vector de referencia que se encuentra en la zona lineal, la máquina secuencial pasa al estado S1 por medio de 6 implementándose el tiempo t_a. Cuando t_a finaliza la señal 1 se activa y la máquina secuencial pasa al estado S2, activándose el contador que implementa al intervalo de tiempo t_b. Cuando el contador finaliza se activa la señal 2 y la máquina secuencial pasa al estado S3. Al terminar la implementación del tiempo t_0^+, se activa la señal 3 y la máquina secuencial vuelve al estado S2 iniciándose un nuevo subciclo con la implementación de t_b. Cuando el contador que implementa a t_b finaliza se

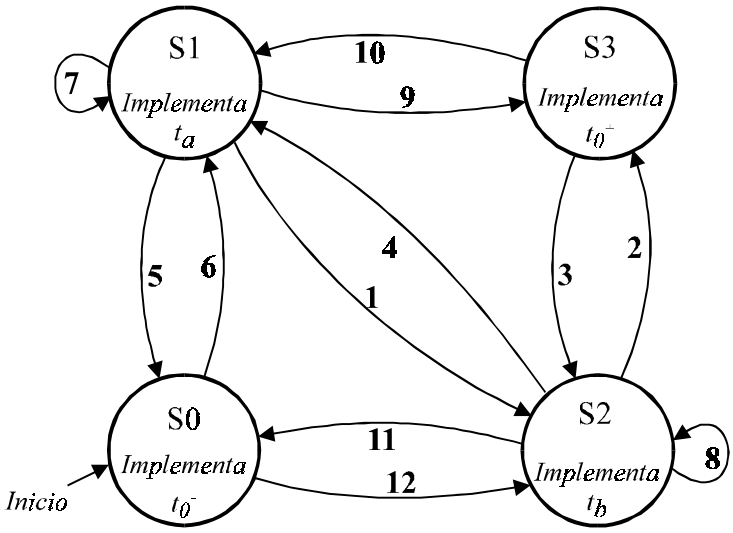


Figura 5.10: Esquema de la máquina secuencial para el modulador vectorial con implementación de los dos modos de sobremodulación.

activa la señal 4, y la máquina secuencial pasa al estado S1. Cuando se termina de implementar t_a en el estado S1, se activa la señal 5 y la máquina secuencial pasa al estado S0. La muestra del vector de referencia termina de ser implementada cuando finaliza el intervalo de tiempo t_0^-, completándose la secuencia.

La máquina secuencial verifica a que zona pertenece la siguiente muestra del vector de referencia cargada en los contadores, si ésta sigue en la zona lineal entonces se repite la secuencia anterior. En cambio, si el valor de la muestra del vector de referencia esta sobre el hexágono, la secuencia a implementar es la del Modo I de sobremodulación. En ésta los vectores nulos no se aplican, por lo tanto los estados de la máquina secuencial válidos son S1 y S2 y las señales validas para pasar de un estado a otro son 1, 8, 4 y 7 (Fig. 5.10). Si la muestra anterior estaba en la zona lineal, entonces la máquina secuencial pasa al estado S1 por medio de la señal 6. Cuando termina de implementarse t_a, la máquina secuencial pasa a S2 por medio de la activación de la señal 1. Como se dijo anteriormente, en el estado S2 se implementa t_b, al finalizar su implementación se activa la señal 8 y la máquina secuencial no cambia de estado volviendo a implementar el intervalo de tiempo t_b. Cuando termina la implementación de t_b, la máquina secuencial activa la señal 4 para pasar al estado S1 y finalizar la secuencia con la implementación de t_a. La máquina secuencial verifica a que zona pertenece la siguiente muestra, implementando la secuencia correspondiente.

Si la muestra a implementar pertenece al modo six-step, límite superior del Modo II de sobremodulación, entonces durante todo el periodo de conmutación la máquina secuencial debe permanecer en el mismo estado ya sea S1 ó S2.

Existen dos casos de secuencias particulares que corresponden a los extremos en donde t_b ó t_a son iguales a cero y $t_0 \neq 0$. Esta situación esta representada por los pasos 9 y 10 o 5 y 6 para $t_b = 0$, y para $t_a = 0$ los pasos son 2 y 3 o 11 y 12 según cual sea el vector activo V_b.

5.4. Resumen

En la modulación vectorial, el máximo valor de la componente fundamental de la tensión de salida en la región lineal es de $Vcc/\sqrt{3}$; cubriendo el 90% del aprovechamiento del bus de continua. Este valor es aproximadamente un 15% más que el del método de modulación sinusoidal.

El 10% restante se logra con la sobremodulación, la cual se divide en dos modos de funcionamiento. El análisis de los modos se basa en la representación en serie de Fourier de los vectores modificados, en donde se realizó una transformación gráfica entre el vector modificado en el plano complejo y la tensión de fase en el dominio del tiempo. Los ángulos α_{NL} y α_H correspondientes al modo I y al modo II respectivamente, están expresados numéricamente en función de la componente fundamental de tensión de salida. Las ecuaciones que los describen son de tipo recurrente. Estos datos pueden ser escritos en memorias, o pueden ser linealizados a tramos para implementaciones en tiempo real.

Tanto en el modo I como en el modo II, los ángulos α_{NL} y α_H actúan como limitadores de región para el cálculo de los tiempos de activación de las llaves del inversor. Cuando el ángulo del vector modificado está comprendido fuera de ellos, el cálculo de los tiempos se realiza utilizando (5.2.b). En cambio, en el modo I cuando el vector modificado se encuentra dentro del intervalo de α_{NL} debe utilizarse (5.2.a) y para el modo II dentro del intervalo α_H el tiempo de activación de las llaves del inversor coincide con el tiempo del subciclo.

Para que el modulador funcione en las zonas de sobremodulación, debe ser capaz de implementar las muestras que están afuera de la circunferencia inscripta en el hexágono del plano complejo y además llegar a funcionar en el modo six-step.

La implementación de la sobremodulación en el modulador vectorial, solo requiere la modificación de la máquina secuencial y de los intervalos de tiempo que se guardan en las ROMs; manteniéndose el esquema presentado en el capítulo 2 para el modulador lineal.
Debido a que los cálculos de los tiempos se realizan off-line, puede implementarse un modulador vectorial que funcione desde la zona lineal hasta el six-step. Si para una determinada aplicación se requiere una baja distorsión, entonces se deberá limitar la señal de referencia para que el modulador funcione solo en la zona lineal aprovechando solo el 90% del bus de continua. En cambio, si se desea aprovechar al 100% el bus de continua, entonces no será necesario limitar la señal de referencia, pero debe tenerse en cuenta el aumento de la distorsión armónica. Sin embargo, limitar la señal de referencia significa que se pierde resolución en la amplitud de la tensión que se puede obtener. Para aplicaciones que requieran de una alta resolución en la amplitud y una baja distorsión, puede ser conveniente desarrollar el modulador para que opere solo en la zona lineal.

6. TIEMPO MUERTO

6.1. Introducción

La mayoría de las técnicas de modulación propuestas hasta ahora, están basadas en la suposición de que las llaves del inversor operen de manera ideal. Esto significa que el inversor trabaja precisa y rápidamente como las señales del PWM. Es decir que las llaves conmuten en tiempo cero y respondan sin retardo a la señal de comando. En la práctica, las llaves del inversor son transistores, los cuales tienen tiempos de conmutación finitos, donde el tiempo de apagado es de mayor importancia.

Dado que el tiempo de apagado (t_{OFF}) es mayor que el de encendido (t_{ON}), debe introducirse un tiempo de seguridad para evitar un cortocircuito en la columna del inversor. Este tiempo de seguridad se denomina comúnmente tiempo muerto (T_d) [Moh 89][Hol 94c] [Hol 97]. Si bien el tiempo muerto garantiza una operación segura, también afecta adversamente la prestación del inversor. Mientras transcurre el tiempo muerto, las dos llave están abiertas y la tensión en la columna esta impuesta por la carga. La forma de onda de salida es distinta de la generada por el modulador; repitiéndose una y otra vez para cada instante de conmutación. Esto se conoce como el efecto de tiempo muerto.

El desarrollo de dispositivos de conmutación cada vez más rápidos, no necesariamente mejora la situación. El uso de éstos implica elevar la frecuencia de conmutación, y lo que realmente interesa es la relación entre el tiempo muerto y el período de conmutación y éste no se modifica sustancialmente. Por consiguiente, más allá de los dispositivos de conmutación a utilizar, es importante una comprensión profunda del efecto de tiempo muerto para poder mejorar la perfomance del inversor de potencia con PWM.

6.2. Análisis del efecto de tiempo muerto

Los dispositivos semiconductores utilizados como interruptores electrónicos no son ideales, éstos reaccionan con cierto retardo a las señales de control de encendido y apagado. El tiempo de retardo depende del tipo de semiconductor, sus rangos de corrientes y tensiones, la temperatura del dispositivo y de la magnitud de la corriente que debe ser conmutada.

En la Figura 6.1 se muestran los diferentes tiempos que determinan los tiempos de conmutación de los IGBT. Los tiempos de conmutación t_{ON} y t_{OFF} quedan definidos como sigue:

Tiempo de encendido $t_{ON} = t_{d(on)} + t_r$

Tiempo de apagado es $t_{OFF} = t_{d(off)} + t_f$

A modo de ejemplo se muestran los valores típicos para cuatro semipuentes de IGBT de distintas potencias:

Características de Conmutación	BSM50GB60DL *(600V, 50A)*	BSM75GB60DL *(600V, 75A)*	BSM50GB120DL *(1200V, 50A)*	BSM75GB120DL *(1200V, 75A)*
Tiempo de retardo de encendido $\mathbf{t_{d(on)}}$	100 nseg	150 nseg	90 nseg	100 nseg
Tiempo de subida $\mathbf{t_r}$	50 nseg	60 nseg	75 nseg	50 nseg
Tiempo de retardo de apagado $\mathbf{t_{d(off)}}$	300 nseg ***(62%)***	450 nseg ***(64%)***	470 nseg ***(66%)***	650 nseg ***(76%)***
Tiempo de bajada $\mathbf{t_f}$	35 nseg	40 nseg	70 nseg	50 nseg

El tiempo de retardo de apagado $t_{d(off)}$, es el tiempo de almacenamiento (T_{st}) que dan como dato algunos fabricantes y es el más grande de todos los tiempos que caracterizan la conmutación. De aquí en más se considerará el tiempo T_{st}, y se despreciarán los otros.

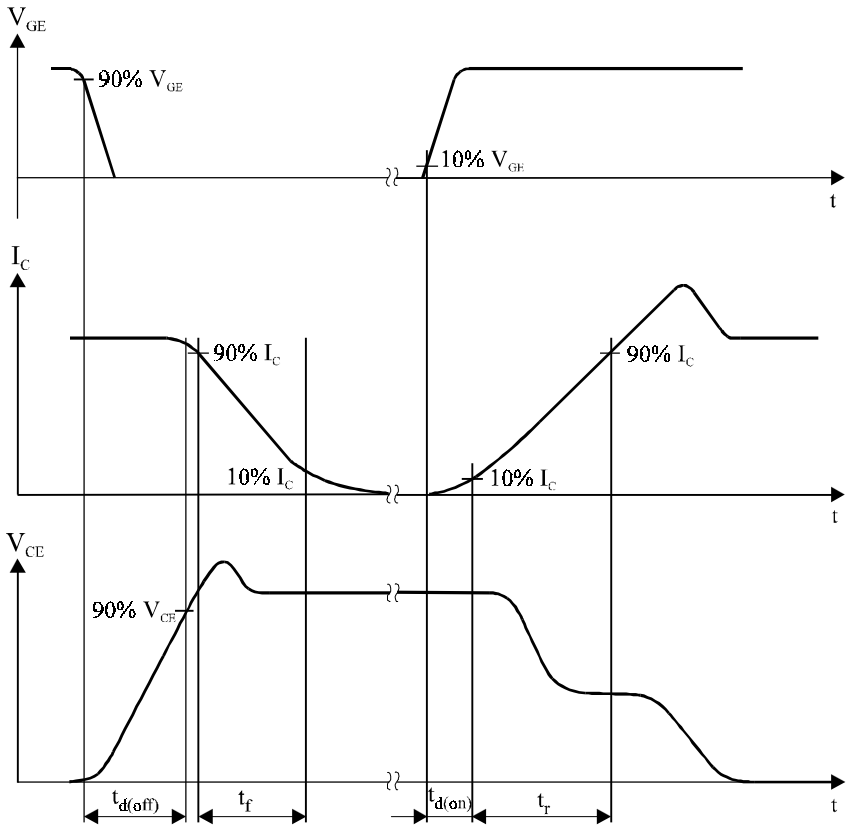


Figura 6.1: Tiempos de conmutación de los IGBT

Para evitar cortocircuitos en las columnas del inversor, el circuito de control de las señales de excitación del inversor debe introducir un tiempo de retardo T_d, en las señales de activación de los dispositivos (Fig. 6.2). El tiempo de retardo T_d, queda determinado por el máximo valor del tiempo de almacenamiento T_{st} más un intervalo de tiempo adicional por seguridad, que comúnmente se toma igual a 3 ó 4 veces T_{st}. Se considera particularmente T_{st} porque abarca más del 60% de la suma de $t_{ON} + t_{OFF}$.

Se puede observar en el circuito de la Fig. 6.2, que en el intervalo de tiempo T_d los interruptores quedan abiertos, por lo que el terminal R queda flotando y su tensión estará determinada por la carga. Si la carga es inductiva, la corriente i_R circulará por los diodos de rueda libre D_1 o D_2 dependiendo del signo de la misma. Con el signo de i_R se tienen dos situaciones diferentes para la tensión V_{R0}, como se muestra en la Fig. 6.3. Suponiendo que $i_R>0$ (de acuerdo a la convención de la Fig. 6.2); cuando la señal del modulador (PWM) pasa a alto, la señal G_2 de la base del transistor T_2, se desactiva y la señal G_1 de T_1 se retrasa en T_d antes de activarse. Como la corriente es positiva, durante el intervalo de tiempo (T_d) sigue conduciendo D_2 hasta que se activa G_1; por lo tanto la tensión de salida V_{R0} se mantiene negativa. Cuando la señal PWM cambia a cero inmediatamente G_1 pasa también a cero, pero el apagado de T_1 es retrasado debido al tiempo de almacenamiento $T_{st}<T_d$. Consecuentemente, el tiempo en que V_{R0} es mayor que cero, es menor que el tiempo que PWM = 1 en una diferencia igual a T_d - T_{st}. Un efecto similar ocurre cuando la corriente es negativa. En la Fig. 6.3 se observa para el caso de $i_R<0$, que el tiempo en que $V_{R0} > 0$ es

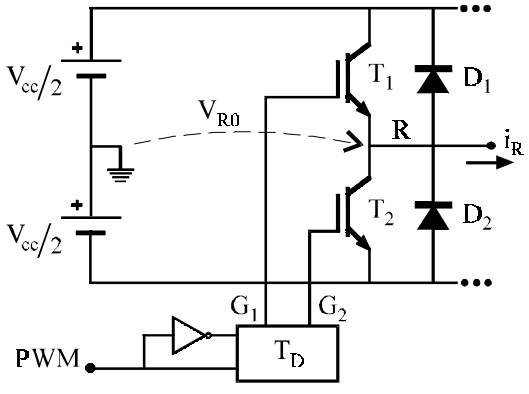


Figura 6.2: Configuración básica de una columna del inversor

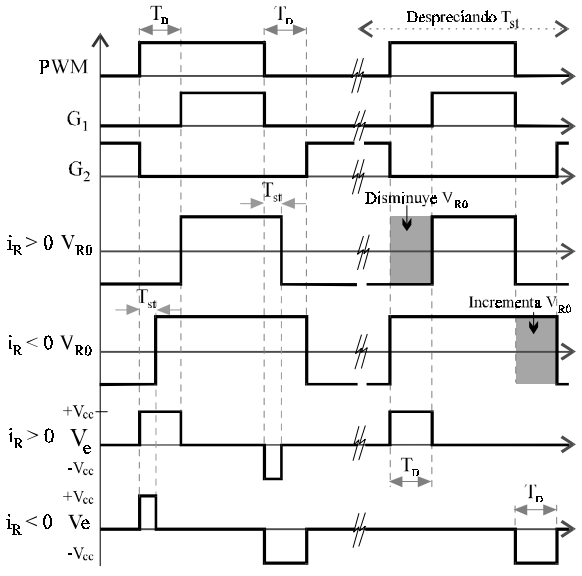


Figura 6.3: Señales de excitación y salida del inversor.

mayor que el tiempo en que la señal de control PWM = 1 en un valor dado por la diferencia T_d - T_{st}. Por lo tanto el ciclo de trabajo real de la columna del inversor no coincide con el de la señal de control PWM. Éste está incrementado o disminuido dependiendo del signo de la corriente. Este efecto se refleja en una disminución de la tensión eficaz de salida.

Para cuantificar el efecto del tiempo muerto, se efectúan dos hipótesis simplificativas:

a) el tiempo de almacenamiento T_{st} es despreciable frente a T_d

b) la frecuencia de conmutación f_s es suficientemente mayor que la frecuencia fundamental (f_o) de la señal modulante

Al despreciar el tiempo T_{st}, el pulso de error $V_e = PWM_{ideal} - V_{R0}$ (Fig. 6.3), tiene una duración de T_d y una amplitud igual a $\pm V_{cc}$. La señal PWM_{ideal} está en fase con la señal PWM, y tiene una amplitud de $\pm V_{cc}/2$. El valor medio de V_e en cada período de conmutación es:

$$V_{e_{AVG}} = \pm T_d \, f_s \, V_{cc} \tag{6.1}$$

La (6.1) es negativa mientras la corriente es positiva y positiva cuando la corriente es negativa. La Ve_{AVG} es una onda cuadrada en contrafase con la corriente de carga. Aplicando Fourier podemos obtener los k-ésimos armónicos de Ve_{AVG}:

$$V_{e_k} = \frac{4}{\pi k} T_d \, f_s V_{cc} \qquad \text{donde k: 1, 3, 5, ...} \tag{6.2}$$

La (6.2) con k = 1, da la disminución de la tensión fundamental de columna del inversor. En la Fig. 6.4 se muestra la tensión de salida con el efecto de tiempo muerto, junto con la tensión de referencia, la corriente de carga y la tensión promedio de error con su componente fundamental. Se puede observar que para valores positivos de la corriente, la tensión de salida disminuye y lo contrario ocurre cuando la corriente es negativa.

Una vez seleccionados T_d, f_s y V_{cc} la tensión de error Ve_1 queda definida y es constante. Por lo tanto su efecto será más notable para valores bajos del índice de modulación. Otro efecto importante tiene relación con los armónicos. Debido a que la tensión de salida no puede ser igual que la señal de control, aparecen componentes armónicas de baja frecuencia, no deseadas en la tensión de salida del inversor.

A continuación, se muestran simulaciones realizadas con PSPICE. Se utilizó un inversor trifásico con dispositivos de potencia IGBT, tiempo de seguridad T_d = 5 μseg,

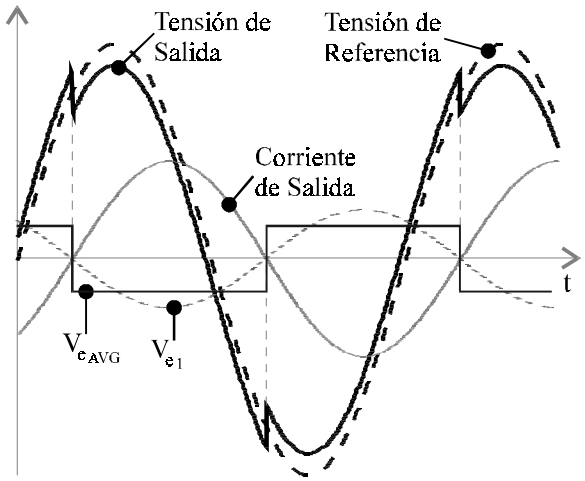


Figura 6.4: Tensión de salida con efecto de tiempo muerto

tensión del bus de continua de 540V y un índice de modulación de 0.5. El modulador se realizó utilizando modulación sinusoidal con frecuencia de conmutación de 16.7 KHz y una referencia de 500Hz. La elección de la alta frecuencia de referencia se realizó para disminuir el tiempo total de la simulación. Se utilizó un filtro de 7^{th} orden con frecuencia de corte en 3 KHz, para ver la tensión promedio en la columna del inversor.

En la Fig. 6.5 puede apreciarse la distorsión de baja frecuencia en la corriente y su bajo valor de salida, para el inversor sin compensación. También se muestra la tensión de la columna filtrada, en donde se observa la variación de la tensión en el cruce por cero de la corriente, confirmando lo analizado previamente. Comparando las Fig. 6.4 y Fig. 6.5, se verifica el efecto del tiempo muerto y que éste coincide con el análisis aproximado realizado.

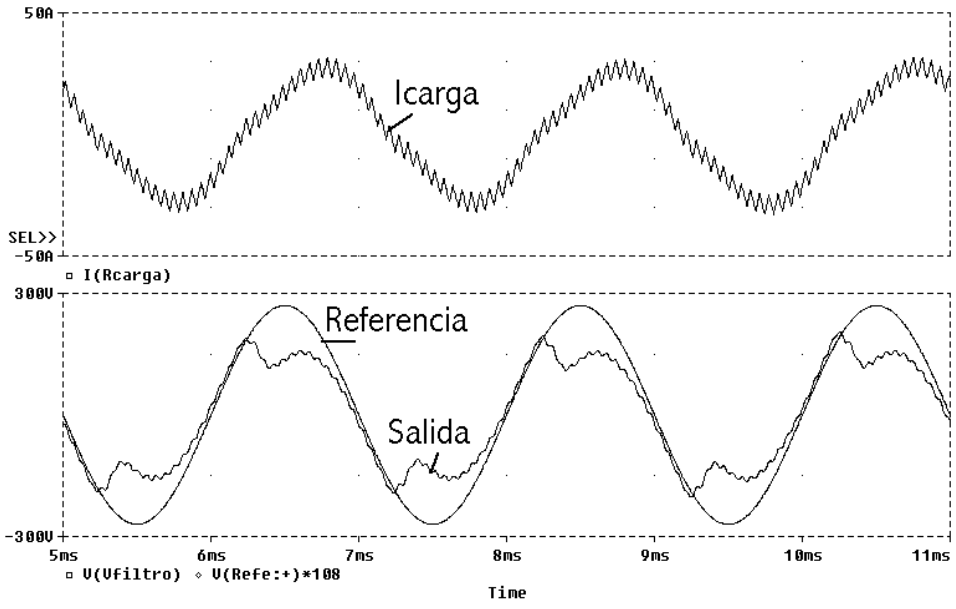


Figura 6.5: Inversor sin compensar.

6.3. Métodos de corrección

Si el modulador y el inversor forman parte de un lazo de control de corriente, la distorsión en la corriente causada por el efecto de tiempo muerto es compensada por el lazo de control. Esto minimiza o disminuye la necesidad de implementar una compensación de tiempo muerto. Por el contrario el uso de un compensador es muy importante cuando no existe un lazo de corriente rápido o cuando el par de la máquina alimentada por el inversor debe ser muy suave [Hol 97]. La compensación del tiempo muerto puede hacerse mediante distintos métodos [Ton 00].

Anteriormente se mostró que la existencia del tiempo muerto provoca caída de tensión y distorsión. Este efecto puede disminuirse haciendo pequeño el producto de T_d por f_s. Tambien se mostró que T_{st} aumenta cuando aumenta la potencia a manejar, por lo que aumenta T_d. Por lo tanto, T_d tiene un limite inferior dado por la potencia, y f_s por las componentes armónicas producto de la modulación. Esto significa que existe un límite inferior para el producto T_d x f_s. Para resolver este problema, el circuito de corrección debe compensar el tiempo muerto sin cambiar el producto T_d x f_s.

6.3.1. Modificación de la referencia

En sistemas donde el PWM se realiza por medio de la comparación de la referencia con la portadora, como ser el método subarmónico, la compensación del efecto de tiempo muerto es muy simple [Jeo 91]. Éste se realiza modificando la referencia de acuerdo al signo de la corriente de fase; de manera tal de compensar el efecto perjudicial del tiempo muerto. En la Fig. 6.6 se muestra un posible circuito para esta implementación. El comparador detecta el signo de la corriente, siendo su salida una onda cuadrada con amplitud igual a la

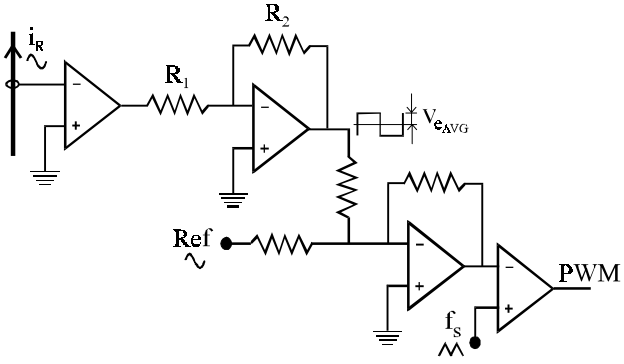


Figura 6.6: Compensación modificando la referencia

dada por (6.1). Cuando la corriente es positiva, la referencia se hace más positiva; y cuando la corriente es negativa la referencia se hace más negativa. Así se modifica la referencia en modo complementario al error que se ve en la Fig. 6.4. Es decir una referencia distorsionada más el error introducido por el tiempo muerto, dan una salida sinusoidal.

Este método también puede aplicarse a moduladores que utilizan muestreo regular, haciendo su implementación en el software del microprocesador [Leg 97]. Tanto en la implementación por hardware como por software, se requiere de sensores de corriente en almenos dos fases. La compensación no será completa, debido a que no tiene en cuenta el tiempo de almacenamiento (T_{st}) de los dispositivos de conmutación. Este método modifica las referencias a las tres fases. Por lo tanto no es aplicable a moduladores con vector espacial.

En la Fig. 6.7 se muestra la corriente de carga, la tensión de salida y la tensión de referencia, para la compensación del efecto de tiempo muerto con modificación de la tensión de referencia. El inversor utilizado es el mismo que el del punto anterior. Comparando con la Fig. 6.5, puede observarse que la corriente no presenta distorsión de baja frecuencia. Además los saltos de tensión son atenuados. Por lo tanto, éste método corrige muy bien el efecto de tiempo muerto.

6.3.2. Compensación con circuito lógico

Muchas estrategias de control de PWM, se basan en el cálculo off-line de los tiempos del PWM. En estos casos se puede utilizar un circuito lógico para compensar el efecto del tiempo muerto [Jeo 91]. Éste provee a los interruptores de la columna el tiempo de

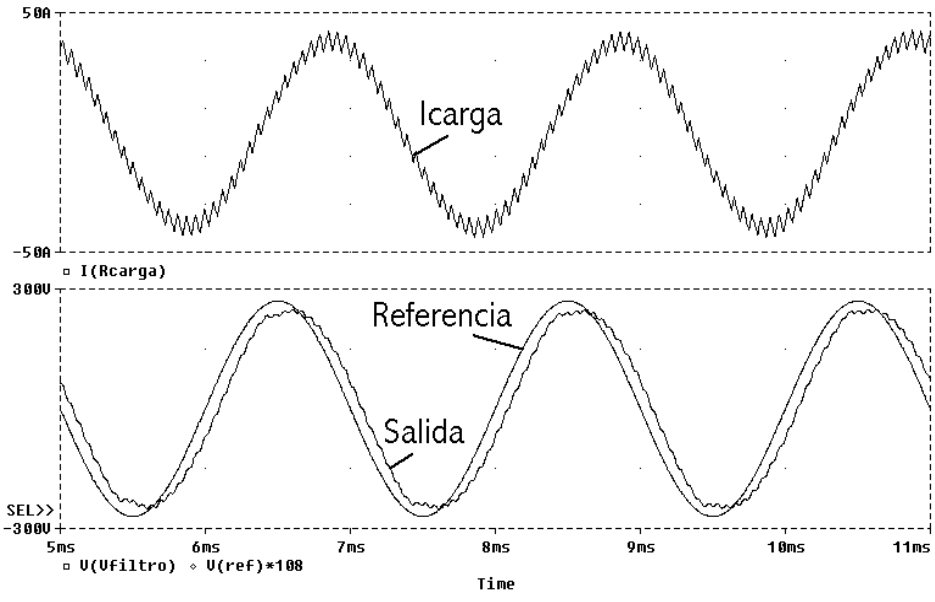


Figura 6.7: Inversor compensado con cambio de la tensión de referencia.

retardo T_d, y además compensa el efecto no deseado del tiempo muerto. Aquí existe una diferencia conceptual con respecto al método anterior, y es que el método de modificación de la referencia cambia la amplitud en cada fase, en cambio este método modifica los tiempos de activación de las llaves del inversor.

En la mitad derecha de la Fig. 6.3 despreciando T_{st}, se puede observar lo siguiente:

a) para $i_R>0$ la forma de onda de la tensión de salida, sigue la señal de excitación G_1,

b) para $i_R<0$ la tensión de salida sigue en forma inversa a G_2.

Por lo tanto, la tensión de salida queda determinada por una sola de las señales de excitación, que dependerá del signo de la corriente.

De lo expresado anteriormente, se deduce que el circuito lógico del compensador debe lograr que el ancho de pulso de la señal de excitación que determina la tensión de salida, sea igual al ancho de pulso de la señal de PWM en la Fig. 6.3. La señal de excitación restante se ajustará para cumplir con el tiempo de seguridad T_d. En la Fig. 6.8 se muestran las formas de onda para implementar este método. Las señales P1 y P2 son iguales a la PWM pero retrasadas en T_d y $2T_d$ respectivamente. Con estas tres señales y con la información del signo de la corriente i_R, se generan las señales excitación de los dispositivos de conmutación del inversor (G1 y G2).

Puede observarse en la Fig. 6.8 para $i_R>0$, que el ancho del pulso de la señal de excitación G1 es igual al ancho del PWM, mientras que G2 se ajusta para cumplir con el tiempo de seguridad T_d. Por el contrario, para $i_R<0$ el ancho del PWM es igual a la señal de

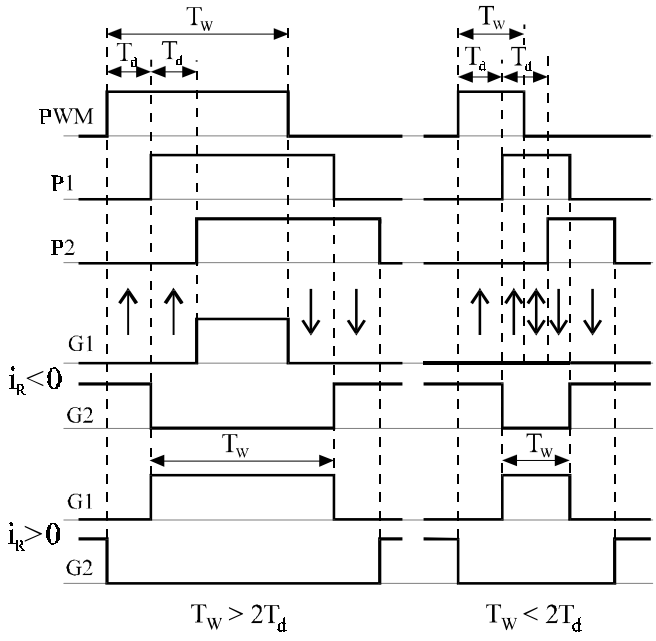


Figura 6.8: Señales para implementar la compensación con lógica

excitación G2 negada, y la señal G1 es quien se ajusta para cumplir con T_d. Cuando la duración de la señal PWM alta es menor que dos veces el tiempo de seguridad ($T_W < 2T_d$) e $i_R<0$, la señal G1 no se activa en absoluto. Las ecuaciones lógicas para generar las señales de excitación de una columna G1 y G2 son,

$$\begin{aligned} G_1 &= I.P_1 + \overline{I}.PWM.P_1.P_2 \\ G_2 &= \overline{I}.\overline{P_1} + I.\overline{PWM}.\overline{P_1}.\overline{P_2} \end{aligned} \tag{6.3}$$

las otras dos columnas tienen las mismas ecuaciones con sus respectivas señales PWM.

Debido a que las señales de excitación de las llaves del inversor son función del signo de la corriente, debe evitarse que la corriente cambie de signo en los lugares marcados con flechas de la Fig. 6.8. De esta manera se puede prevenir un posible cortocircuito de columna. Por ejemplo, en la Fig. 6.8 para el caso en que (PWM,P1,P2) = (1,0,0), se observa que si la corriente cambia de negativa a positiva en ese intervalo, la señal G2 cambia de 1 a 0 instantáneamente en el cruce por cero de la corriente. Poco después G1 pasa de 0 a 1, por lo que puede producirse un cortocircuito debido a que no se cumple con el retardo de T_d. El cambio de signo de la corriente se soluciona colocando un Flip-Flop, que mantiene el valor del signo de la corriente en los intervalos deseados. La ecuación lógica para obtener la señal de habilitación (Enable) del Flip-Flop es,

$$Enable = PWM.P_1.P_2 + \overline{PWM}.\overline{P_1}.\overline{P_2} \tag{6.4}$$

En la Fig. 6.9 se muestra un circuito lógico que implementa este método.

Como en el método anterior, requiere de sensores de corriente y la compensación no es completa; ya que siempre adiciona un tiempo constante sin conocer el tiempo de almacenamiento real de los dispositivos de potencia.

La implementación digital de este método, puede aplicarse a moduladores con modulación vectorial. Tomando como entrada la señal de PWM vectorial y generando a la salida las señales de excitación con el tiempo de seguridad T_d y la compensación del tiempo muerto.

En la Fig. 6.10 puede observarse como se corrige la corriente utilizando compensación con circuito lógico. Los saltos en la tensión de salida cuando la corriente cruza

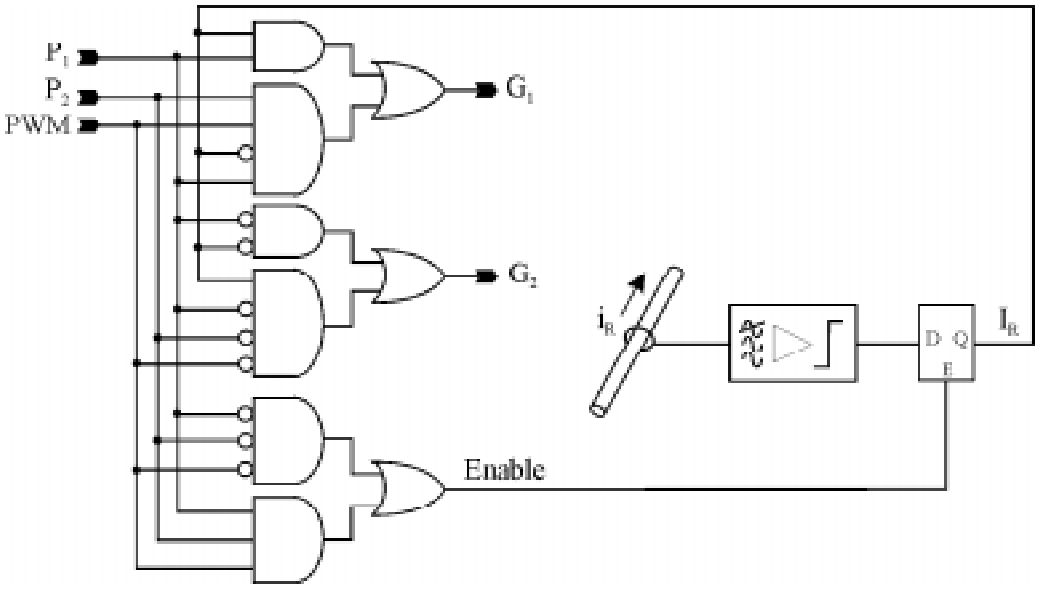


Figura 6.9: Circuito para implementar el método de compensación con circuito lógico

por cero, se ven atenuados. Estos no se anulan debido a la discretización del tiempo de la lógica empleada.

En la Fig. 6.11 se muestran las señales de control, cuando la corriente de carga cruza por cero. Cuando la corriente es mayor que cero, puede observarse que la señal G1 esta retrasada T_d respecto de PWM y el ancho del pulso G1 coincide con el de PWM. En cambio, cuando i<0 G2 negada mantiene el ancho de PWM. Esto verifica el correcto funcionamiento del método de compensación utilizando lógica (ver Fig. 6.8). Comparando la señal PWM con G1 ó G2, se puede apreciar que sólo se introduce un retardo igual a T_d en las señales de excitación.

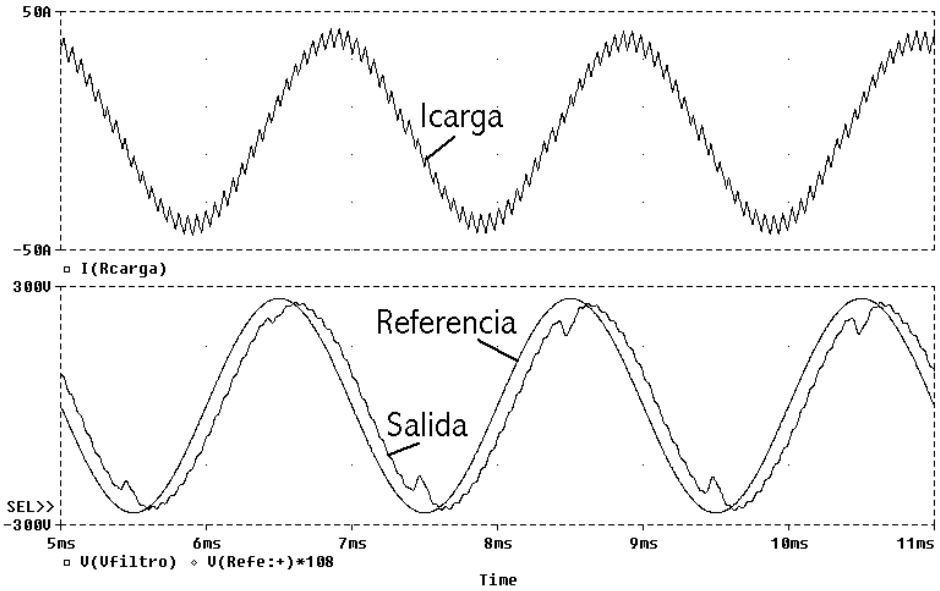


Figura 6.10: Inversor compensado con circuito lógico.

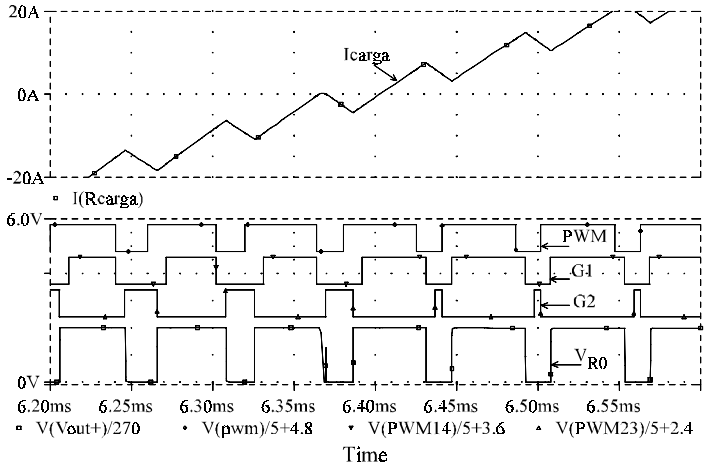


Figura 6.11: Compensación utilizando lógica: Corrientes de carga y señales de control.

6.3.3. Compensación sin medida de corriente

En los casos anteriores la compensación de tiempo muerto se realiza a lazo abierto. En ambos casos los valores que se agregan son constantes, siendo en el primero una tensión y en el segundo un tiempo. Por lo tanto estos métodos tienden a sobrecompensar el efecto de tiempo muerto. Una manera de evitar la sobrecompensación, es midiendo las diferencias de tiempo entre la tensión de la columna y la señal del PWM, para luego agregar o quitar dicha desviación en la conmutación siguiente [Mur 87a]. De esta manera se logrará compensar el efecto de tiempo muerto. Este método requiere la utilización de un circuito de corrección (Fig. 6.12) para medir y cancelar las desviaciones de la tensión de salida debidas al efecto de tiempo muerto, trabajando como un control de lazo cerrado. El modo de funcionamiento, está basado en la medición de la desviación de la tensión de salida con respecto a la señal de

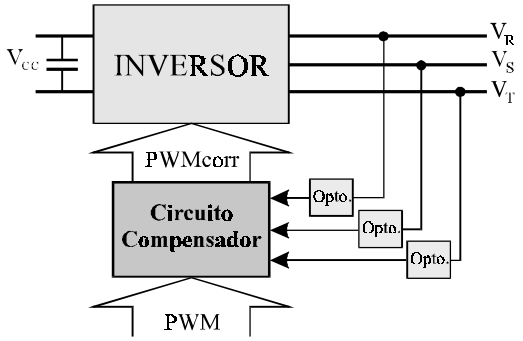


Figura 6.12: Diagrama en bloques del inversor con compensación del tiempo muerto.

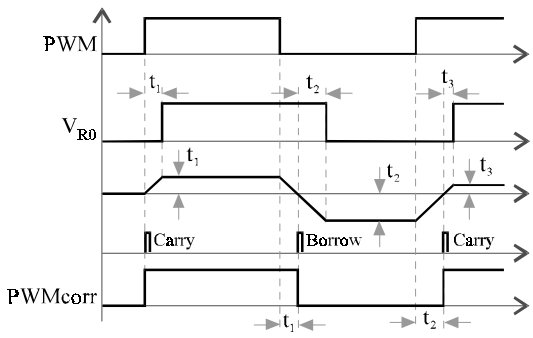


Figura 6.13: Corrección del tiempo muerto, midiendo el tiempo de desviación.

control, que se realiza por medio de un contador. La idea básica es que el contador cuente siempre que los estados lógicos de las señales de PWM y la tensión de columna difieran. La corrección se lleva a cabo en el período de conmutación posterior a la medición.

En la Fig. 6.13 se ilustra el diagrama de tiempos que caracteriza a esta compensación. Se parte de una condición en que la corriente es positiva ($i_R > 0$) y que las señales PWM y PWMcorr cambian de 0 a 1. La tensión de salida V_{R0} cambiará después de un tiempo t_1, que está fijado por el tiempo de almacenamiento real del dispositivo de conmutación. El circuito de corrección mide t_1 con un contador Up/Down. Luego, cuando la señal PWM cambia de 1 a 0, el circuito de corrección adiciona t_1 a la nueva señal de control PWMcorr; compensándose la desviación producida. Los cambios en la señal PWMcorr se realizan con las señales de salida del contador. Cuando el contador cruza por cero con pendiente positiva pone en alto PWMcorr, cuando cruza por cero con pendiente negativa pone en bajo a PWMcorr. Supongamos que cuando PWMcorr se desactiva pasando a cero, la tensión V_{R0} no conmuta permaneciendo en alto durante un tiempo t_2. Este tiempo es medido por el contador. Luego, en la siguiente conmutación de PWM el contador se activa contando de manera ascendente durante t_2. Cuando llega a cero causa la conmutación de la señal PWMcorr, produciéndose la compensación de la desviación de tiempo t_2. La compensación del tiempo t_3 se realiza de la misma forma a la de t_1. Así las siguientes desviaciones de tiempo, se compensan una después de la otra en los siguientes instantes de conmutación. Por lo tanto, ocurre una compensación completa del efecto del tiempo muerto.

Este método opera como un control de lazo cerrado, y a diferencia de los anteriores no mide corriente. Por lo tanto no necesita de sensores de corriente, sólo utiliza optoacopladores para medir la tensión de las columnas. Estos se conectan en paralelo a la llave inferior de la columna del inversor, por lo que no requieren de elevada aislación. El único requisito es que deben ser lo suficientemente rápidos como para seguir las transiciones de la tensión de columna. La implementación de este método, si bien es más compleja, no tiene mayor peso en diseños basados en lógica programable como lo son las FPLD. Puede aplicarse a moduladores sinusoidales como a moduladores con modulación vectorial.

En la Fig. 6.14 se muestra la compensación considerando el tiempo de almacenamiento de los dispositivos de potencia. La corriente no presenta distorsión de baja frecuencia, y el incremento en su valor pico pone en evidencia el aumento en la componente fundamental de la tensión de salida.

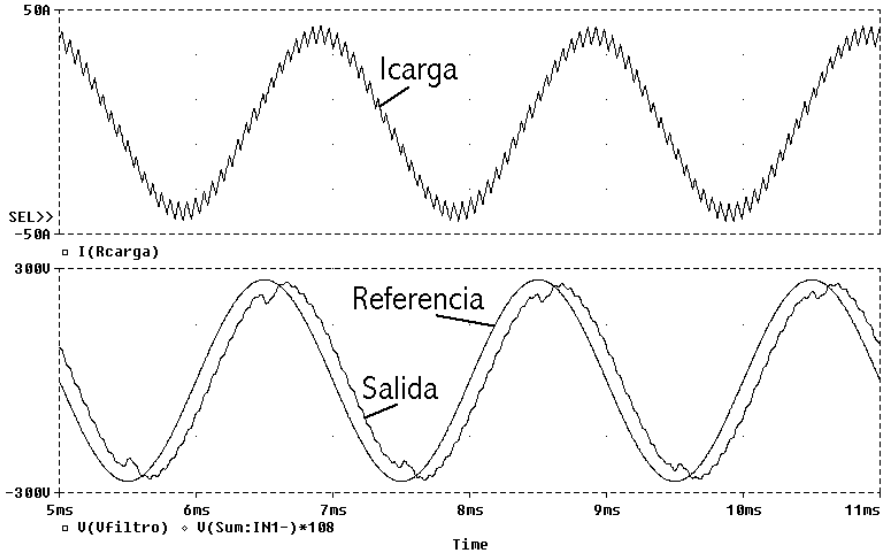


Figura 6.14: Inversor compensado con contador.

Los saltos en la tensión de salida, si bien no se anulan debido a la discretización del tiempo, se ven más atenuados que en el método anterior utilizando lógica.

En la Fig. 6.15 se muestra en un gráfico la corriente de carga, y en el otro las señales de excitación G1 y G2, la señal de PWM, la señal de control corregida PWMcorr y la tensión de la columna del inversor. En los distintos signos de la corriente de carga, se puede observar como se modifica la señal de PWM dando como resultado a PWMcorr, compensando el efecto de tiempo muerto. Después del cruce por cero de la corriente, puede observarse que el primer pulso bajo de PWM tiene el mismo ancho que la señal PWMcorr, debido a que la conmutación se realiza con corriente cercana a cero.

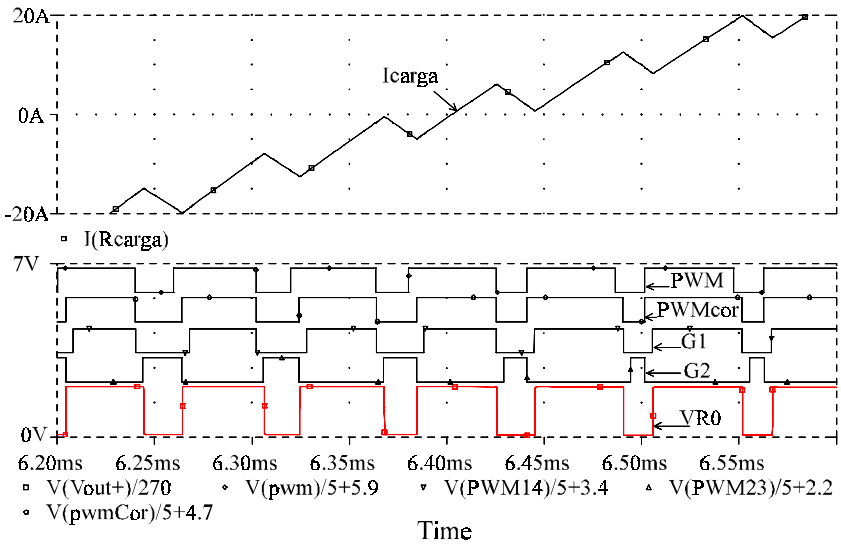


Figura 6.15: Compensación sin medir corriente: Gráfico superior Corriente de carga, y Gráfico inferior señales de control y tensión de columna.

6.4. Implementación de la compensación del efecto de tiempo muerto

El método de modificación de la referencia es una buena elección para moduladores analógicos, midiendo el signo de la corriente, pero no es apto para la modulación vectorial. El segundo método se puede aplicar al modulador vectorial, pero necesita conocer el signo de la corriente por lo que deben adicionarse los respectivos sensores de corriente que son costosos. El tercer método, necesita medir las transiciones de las tensiones de columna (u_{R0}, u_{S0} y u_{T0}), sin necesidad de conocer el signo de la corriente. Para ello se emplean optoacopladores cuyo principal requisito es la velocidad, ya que debe seguir fielmente la tensión de columna. Por lo tanto, el método de compensación sin medición de corriente resulta más apropiado para la implementación de la compensación del efecto de tiempo muerto dentro del modulador vectorial, ya que la complejidad del circuito lógico se ve disminuida utilizando dispositivos lógicos programables.

En la Fig. 6.16 se muestra el circuito que se emplea para obtener la tensión de columna acondicionada a niveles de lógica TTL (V_{R0TTL}). Éste debe repetirse en las otras dos columnas.

El circuito que se muestra en la Fig. 6.17, es el empleado para implementar el método de compensación sin medición de la corriente. Se debe disponer de tres circuitos, uno para cada columna. Si bien es más complejo que el del método utilizando lógica, al momento de la implementación en dispositivos lógicos programables este factor no tiene peso.

El contador es de 8 bits con dos entradas de reloj (cuenta ascendente (UP) y cuenta descendente (DOWN)) y una señal de carga de datos ($\overline{LOAD}$). CARRY y BORROW son las señales de salida del contador, éstas manejan la señal de modulación de ancho de pulso

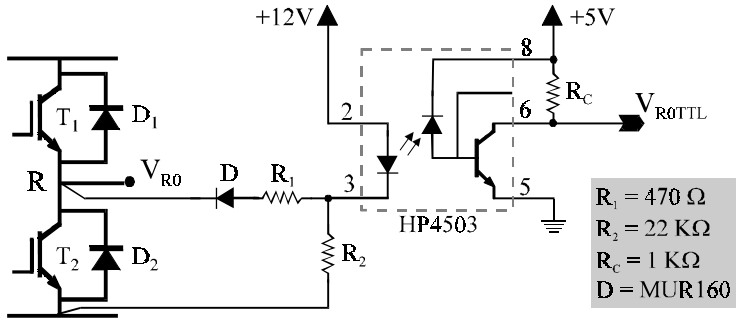


Figura 6.16: Circuito para reproducir digitalmente la tensión de columna

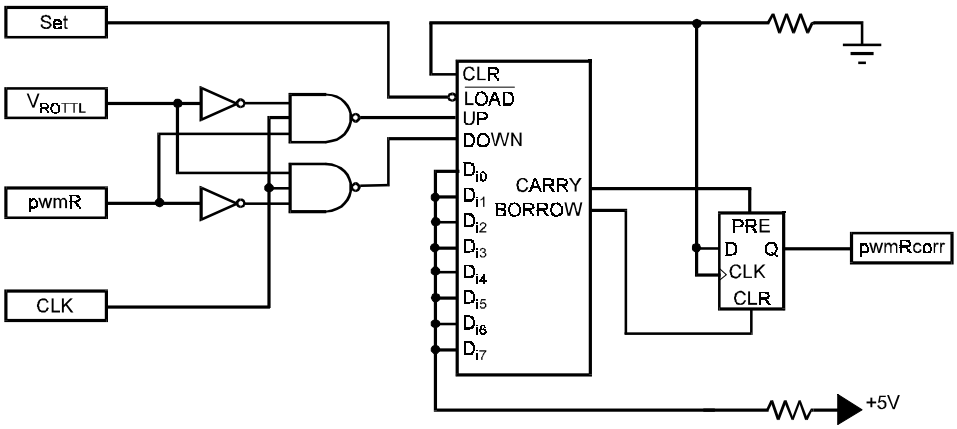
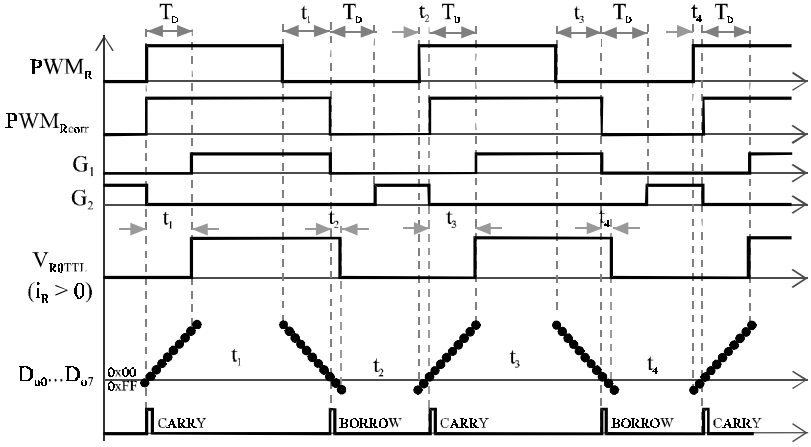


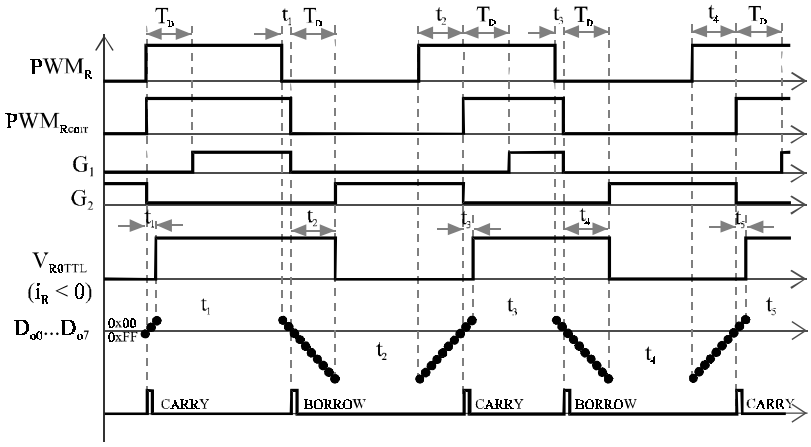
Figura 6.17: Circuito para implementar el método de compensación sin medir corriente para la columna R.

corregido (PWMcorr) a través de un Flip-Flop con Clear y Preset.

El circuito de la Fig. 6.17 funciona de la siguiente manera: cuando la tensión de la columna R en niveles TTL (V_{R0TTL}) es cero y la señal pwmR es uno, la señal de clock (CLK) ingresa al contador por medio de la entrada UP generando la cuenta ascendente en el



a)



b)

Figura 6.18: Señales involucradas en la columna R: a) $i_R > 0$; b) $i_R < 0$

contador, cuando ésta pasa por cero se activa la señal CARRY, la cual pone en estado lógico alto a la señal pwmRcorr. El contador termina la cuenta ascendente cuando la señal V_{R0TTL} se pone en uno. Cuando pwmR está en cero y V_{R0TTL} está en uno, la señal CLK ingresa al contador por la entrada DOWN iniciando la cuenta regresiva del contador. Cuando la cuenta descendente cruza por cero finaliza la compensación del tiempo almacenado en el contador y se activa la señal BORROW, la cual pone a pwmRcorr en cero. La cuenta descendente finaliza cuando V_{R0TTL} alcanza el mismo estado lógico que pwmR.

El contador tiene inicialmente los datos de entrada todos en uno (0xFF en hexadecimal), para iniciar el funcionamiento del sistema.

En la Fig. 6.18 se muestran las señales del circuito de la Fig. 6.17 y las señales involucradas en la columna R, la Fig. 6.18.a corresponde a la corriente positiva y la Fig. 6.18.b a la corriente negativa.

6.5. Resumen

El tiempo de retardo impuesto a la señal de control de PWM, tiene un efecto perjudicial en la operación del inversor. Éste causa una disminución en la componente fundamental e incrementa los armónicos de bajo orden. Por tal motivo, el efecto de tiempo muerto es más notable para índices de modulación bajos. Éste está estrechamente relacionado con la fase (no la amplitud) de la corriente de salida y puede ser evaluado promediando los pulsos de desviación.

Se analizaron los siguientes métodos:

- modificación de la referencia: es un método que puede aplicarse a moduladores sinusoidales, es simple de implementar pero requiere de sensores de corriente. Es ideal para moduladores analógicos.
- utilizando circuitos lógicos: éste método tiende a sobrecompensar el efecto debido a que siempre agrega un tiempo fijo igual a T_d. Puede aplicarse a moduladores que utilizan memorias y a moduladores sinusoidales. Es fácil de implementar pero requiere de sensores de corriente.

- midiendo el tiempo de desviación: puede aplicarse a cualquier modulador. No requiere sensores de corriente, lo cuál permite una implementación de bajo costo. Como elemento de medición utiliza optoacopladores.

Se eligió el tercer método para ser implementado en el modulador por los siguientes motivos,

a) no necesita sensores de corriente

b) la mayor complejidad del circuito digital no es de importancia significativa en una implementación con dispositivos lógicos programables (FPLD)

7 MODULADOR VECTORIAL UNIVERSAL

7.1 Introducción

La mayoría de las aplicaciones de potencia se pueden dividir en tres bloques, compuestos por la carga, la etapa de potencia y el control de la carga. En éstas aplicaciones la modulación vectorial es parte del algoritmo que controla la carga, lo cual puede complicar el diseño del mismo. Resulta interesante, tanto para la implementación práctica como teórica, tener la modulación de ancho de pulso independiente del algoritmo de control formando ésta un bloque más del sistema.

En la Fig. 7.1 puede observarse un sistema completo, en el cual el modulador vectorial constituye un bloque del sistema. De esta manera, se logra independizar el algoritmo de control de la carga de la modulación vectorial de ancho de pulso.

El objetivo es desarrollar un modulador vectorial universal para aplicaciones de laboratorio. Para ello, debe cumplir con las siguientes condiciones:

a) Disponibilidad completa de la componente fundamental de tensión de salida del inversor, es decir una variación del índice de modulación desde cero hasta uno.
b) Posibilidad de compensar el efecto de tiempo muerto.
c) Mínimas conmutaciones.
d) Flexibilidad con el inversor.

La implementación del PWM por medio de la técnica de modulación vectorial (SVM), está basado en el cálculo “off-line” de los tiempos de activación de las llaves. De este modo

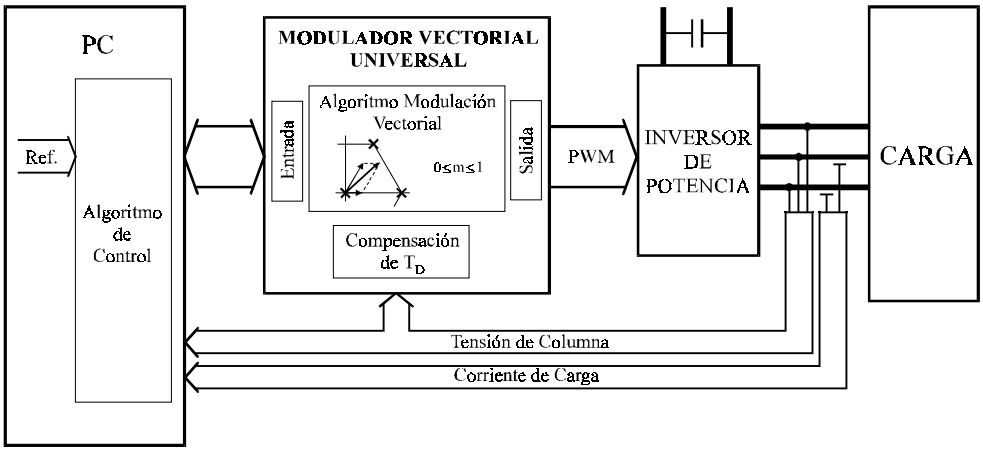


Figura 7.1: Diagrama en bloques de un sistema de potencia para laboratorio.

se puede cerrar el lazo de realimentación con un controlador que no pierda tiempo calculando los tiempos t_a, t_b y t_0, sino que busque los valores en una tabla previamente almacenada en una EPROM.

El diseño propuesto para la implementación del modulador vectorial universal, está basado en el diseño del modulador vectorial lineal visto en el capítulo 4, incorporando sobremodulación y compensación de tiempo muerto. Por lo tanto, el modulador podrá funcionar tanto en la zona lineal como en sobremodulación, cumpliendo con el punto a) del modulador vectorial universal. La posibilidad de compensar el tiempo muerto y de seleccionar distintos tiempos de seguridad T_D, cumple con los puntos b) y d) del modulador vectorial universal. El punto c) de mínimas conmutaciones, se cumple con la selección adecuada de la secuencia de vectores espaciales como se vio en el capítulo 2. Estas incorporaciones tienen como implicancias:

a) cambio de los tiempos t_a y t_b almacenados, como se vio en el capítulo 5
b) modificación en la máquina secuencial
c) incorporación del bloque de compensación con funcionamiento casi independiente

El modulador vectorial universal tiene como entrada de referencia las componentes del vector espacial, en formato polar. Este vector espacial es la salida del algoritmo de control de la carga, y debe ser sintetizado por el modulador. La salida del modulador es la señal de PWM, que ingresa al inversor trifásico de potencia directamente como la señal de excitación de los 6 interruptores. El modulador también posee entradas que provienen de la carga, como ser la tensión de columna del Inversor utilizada para la compensación del tiempo muerto. Las demás entradas y salidas sirven para la comunicación del modulador con la PC.

En la Fig. 7.2 se presenta un diagrama en bloques del modulador propuesto, el cuál consta básicamente de cuatro etapas:

- *Entrada:* usada para la comunicación entre el modulador vectorial universal y la PC, a través del bus ISA. Esta etapa es la misma que se desarrolló en el punto 4.3.1 para el modulador vectorial lineal y puede ser fácilmente adaptada para comunicar al modulador por medio de otro bus como el bus PCI, ó con otro sistema, como ser un DSP. Consta de dos bloques, uno es un registro (Latch IN) en donde se retiene al nuevo vector espacial a implementar y el otro es el bloque de decodificación que genera la señal de habilitación del modulador vectorial universal. A través de esta etapa, el algoritmo de modulación

vectorial recibe el vector espacial de salida del algoritmo de control en coordenadas polares.

- *Algoritmo de Modulación Vectorial:* Esta es la etapa principal del Modulador Vectorial Universal. En él se genera el algoritmo de modulación vectorial, pudiendo trabajar desde la zona lineal hasta la sobremodulación alcanzando el funcionamiento de Six-step, manteniendo el número de conmutaciones en un mínimo. Está compuesto por cuatro bloques. El registro de entrada (Latch II), en donde se almacenan los tiempos de los vectores espaciales activos (t_a, t_b) que deben ser sintetizados por los contadores. Estos tiempos se calculan "off-line" y se guardan en memorias EPROM. El bloque de los contadores, se encarga de la implementación del período de conmutación (T_S) y de los tiempos en que deben aplicarse los vectores espaciales activos (t_a y t_b). La máquina secuencial se ocupa principalmente de decidir que contador va a actuar y cual no, implementando de esta forma la secuencia de vectores espaciales deseada. Esta secuencia se selecciona para que el Inversor de Potencia tenga mínimas conmutaciones, como se vio en el punto 2.4. El bloque de salida es la tabla de estados, en donde se almacenan los estados de los ocho vectores espaciales; siendo manejada por la máquina secuencial y el Latch II.

- *Compensación del tiempo muerto:* esta etapa es independiente del algoritmo de modulación vectorial y se encarga de compensar el efecto de tiempo muerto. Las entradas

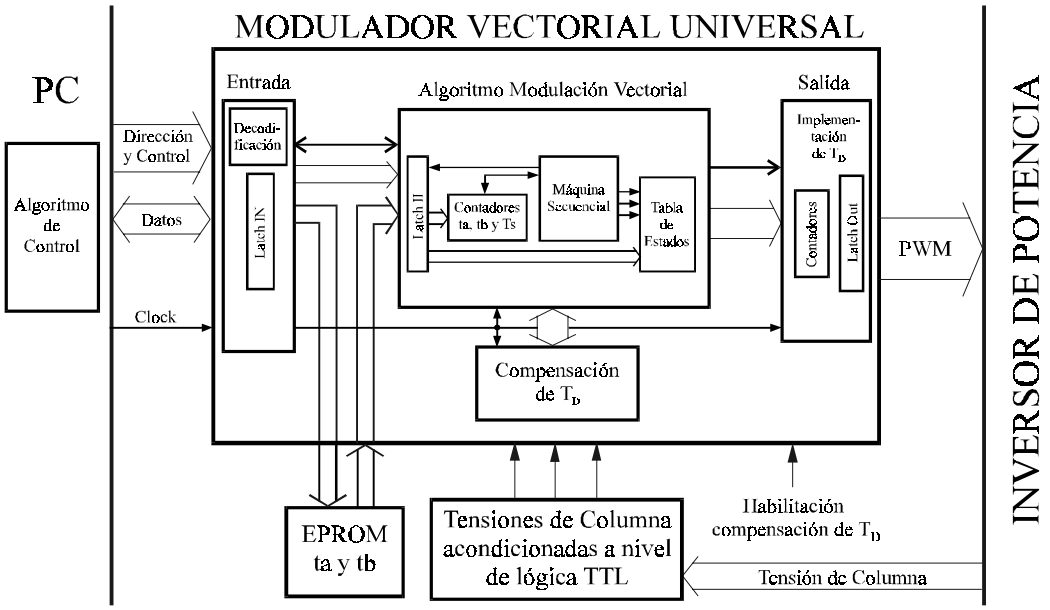


Figura 7.2: Diagrama en bloques del modulador propuesto.

a esta etapa son las tensiones de columna, previamente acondicionadas a los niveles de tensión de lógica TTL, y el PWM de salida del algoritmo de modulación vectorial. La salida es el PWM al cuál se le agrega el tiempo de seguridad T_D. Esta etapa puede habilitarse o no, por medio de un switch. En caso de que no se utilice, el PWM de salida del algoritmo de modulación vectorial ingresa directamente a la etapa de salida. El método de compensación que se utilizará es el de compensación sin medida de corriente, descripto en el capítulo 6.

- *Salida:* acondiciona las señales de excitación de las llaves del inversor, que provienen de la etapa del algoritmo de modulación vectorial ó de la etapa de compensación de T_D. Es una etapa importante debido a que genera los retardos necesarios para que no se produzcan cortocircuitos en las columnas del inversor. La implementación del tiempo de seguridad se realiza con registros de desplazamiento, de esta manera se puede seleccionar la magnitud de dicho tiempo, adaptándose al inversor de potencia y cumpliendo con la condición d) del modulador vectorial universal.

En este capítulo se darán los detalles para la implementación del modulador vectorial universal en un FPLD, los cuales fueron presentados en el congreso internacional IECON 2001 [Ton 01].

7.2 Diseño del circuito de control de PWM en el FPLD

El diseño del modulador se realiza para que cumpla con las condiciones dadas en el punto 7.1, de manera tal de obtener un modulador universal. Por lo tanto, debe poder trabajar en sobremodulación llegando a alcanzar el modo de funcionamiento six-step. También debe proveer una compensación de tiempo muerto, y entregar las señales de excitación de las llaves del inversor con el respectivo tiempo de seguridad. Esta posibilidad tendrá variantes tales como elegir un determinado tiempo de seguridad acorde al inversor que se utilice, o la posibilidad de que las señales de excitación no posean dicho tiempo de seguridad si es que lo tiene el inversor.

En los siguientes puntos se darán los detalles del diseño de la etapa del Algoritmo de Modulación Vectorial y de la etapa de compensación de T_D (Fig. 7.2). Las etapas de entrada

y de salida son las mismas que se describieron para el modulador vectorial lineal en el capítulo 4, por lo que se omitirán en este capítulo.

Para la implementación del modulador vectorial universal se utilizará un FPLD y dos EPROMs, además de los accesorios necesarios para el correcto funcionamiento del mismo. Al igual que en el modulador lineal la elección de almacenar los tiempos en memorias externas se hizo para disminuir el tamaño del FPLD empleado, pero se podrían incluir tranquilamente en él.

7.2.1 Etapa del Algoritmo de Modulación Vectorial

Esta es la etapa principal del Modulador Vectorial Universal. Se encarga de generar las señales de excitación de las llaves del inversor, siguiendo la secuencia previamente definida de la aplicación sucesiva de los vectores espaciales. Esta etapa es básicamente igual a la del modulador lineal, descripto en el capítulo 4. Lo único que se modifica es la implementación del bloque correspondiente a la máquina secuencial (SM), como se mostró en el punto 5.3, para que pueda trabajar desde la zona lineal hasta el six-step. Consecuentemente deben modificarse los datos de los intervalos de tiempo que se guardan en las EPROMs. En la Fig. 7.3 se muestra un diagrama de bloques del Algoritmo de Modulación Vectorial.

En el punto 5.3 se detalló el funcionamiento ideal de la máquina secuencial del Modulador Vectorial Universal. En este punto se describirá con detalles su implementación práctica. Para ello se presentarán nuevamente todas las posibles secuencias de vectores espaciales que debe implementar. Las secuencias en Modo Lineal, Modo I, Modo II y las secuencias particulares son:

Modo Lineal: $\ldots\underbrace{|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|}_{0^o\le\alpha<60^o}\ldots|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|\ldots$

Modo I Sobremodulación: $\ldots\underbrace{|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|}_{0^o\le\alpha<\alpha_{NL}}\underbrace{|t_a\ t_b\ t_b\ t_a|}_{\alpha_{NL}\le\alpha<60^o-\alpha_{NL}}\ldots|t_a\ t_b\ t_b\ t_a|\underbrace{|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|}_{60^o-\alpha_{NL}\le\alpha<60^o}\ldots$

Modo II Sobremodulación: $\ldots\underbrace{|t_a\ t_a|}_{0^o\le\alpha<\alpha_H}\ldots|t_a\ t_a|\underbrace{|t_a\ t_b\ t_b\ t_a|}_{\alpha_H\le\alpha<60^o-\alpha_H}\ldots|t_a\ t_b\ t_b\ t_a|\ \underbrace{|t_b\ t_b|}_{60^o-\alpha_H\le\alpha<60^o}\ \ldots|t_b\ t_b||t_a\ t_a|\ldots$

Secuencias particulares: $....|t_a \;\; t_0^+ \;\; t_a \;\; t_0^-|...$

$....|t_b \;\; t_0^+ \;\; t_b \;\; t_0^-|...$

En la Fig. 7.4 se muestran todos los estados de la máquina secuencial del Modulador Universal junto con las señales y en la Tabla 7.1 se dan los significados de dichas señales, que componen al Algoritmo de Modulación Vectorial.

Tabla 7.1: significados de las señales y salidas de los estados de la SM

Señales y Estados	**Descripción**
MSB	***MSB*** = 0 Subciclo impar, ***MSB*** = 1 Subciclo par
PLa	***PLa*** = 1 carga t_a en Ct_a, ***PLa*** = 0 permite la cuenta de Ct_a
PLb	***PLb*** = 1 carga t_b en Ct_b, ***PLb*** = 0 permite la cuenta de Ct_b
FCa	***FCa*** = 1 el contador llegó a cero, ***FCa*** = 0 salida de Ct_a distinta de cero
FCb	***FCb*** = 1 el contador llegó a cero, ***FCb*** = 0 salida de Ct_b distinta de cero
SumaAB	***SumaAB*** = 1 la suma de $t_a + t_b = \Delta t$, sino $t_a + t_b < \Delta t$
Cero_A	***Cero_A*** = 1 el tiempo $t_a = 0$, ***Cero_A*** = 0 $t_a > 0$
Cero_B	***Cero_B*** = 1 el tiempo $t_b = 0$, ***Cero_B*** = 0 $t_b > 0$
Max_A	***Max_A*** = 1 el tiempo $t_a = \Delta t$, ***Max_A*** = 0 $t_a < \Delta t$
Max_B	***Max_B*** = 1 el tiempo $t_b = \Delta t$, ***Max_B*** = 0 $t_b < \Delta t$
L_IIDina	Dato de entrada del contador Ct_a
L_IIDinb	Dato de entrada del contador Ct_b
MR	Reset del contador C: ***MR*** = 0 cuenta, ***MR*** = 1 detenido
S0	Implementación de t_0^- Contadores Ct_a y Ct_b detenidos, ***PLa*** = ***PLb*** = 1
S1	Implementación de t_a Cuenta descendente de Ct_a, ***PLa*** = 0, y ***PLb*** = 1
S2	Implementación de t_b Cuenta descendente de Ct_b, ***PLa*** = 1, y ***PLb*** = 0
S3	Implementación de t_0^+ Contadores Ct_a y Ct_b detenidos, ***PLa*** = ***PLb*** = 1
S4	Implementación de t_a Contador Ct_a Inactivo ***PLa*** = ***PLb*** = 1
S5	Implementación de t_b Contador Ct_b Inactivo ***PLa*** = ***PLb*** = 1

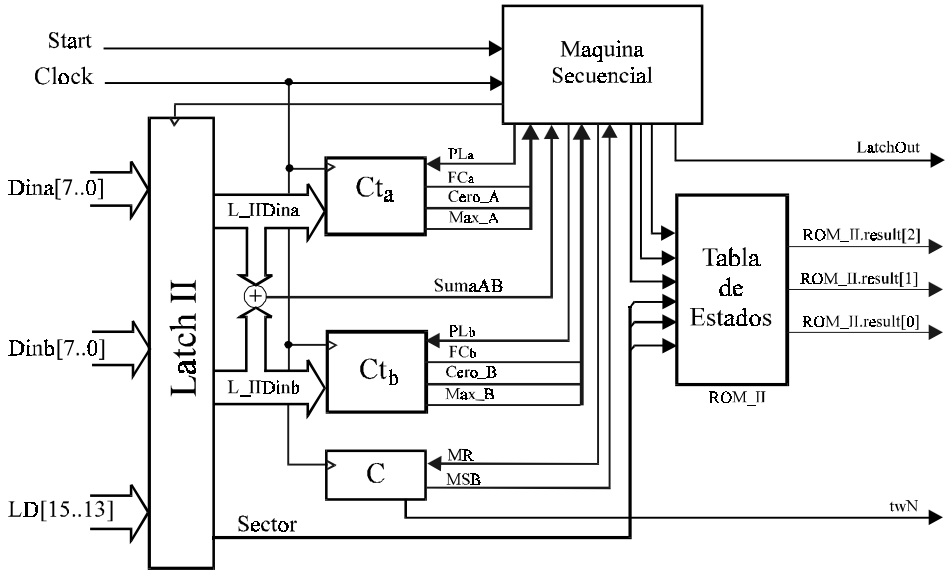


Figura 7.3: Diagrama en bloque de Algoritmo de modulación Vectorial.

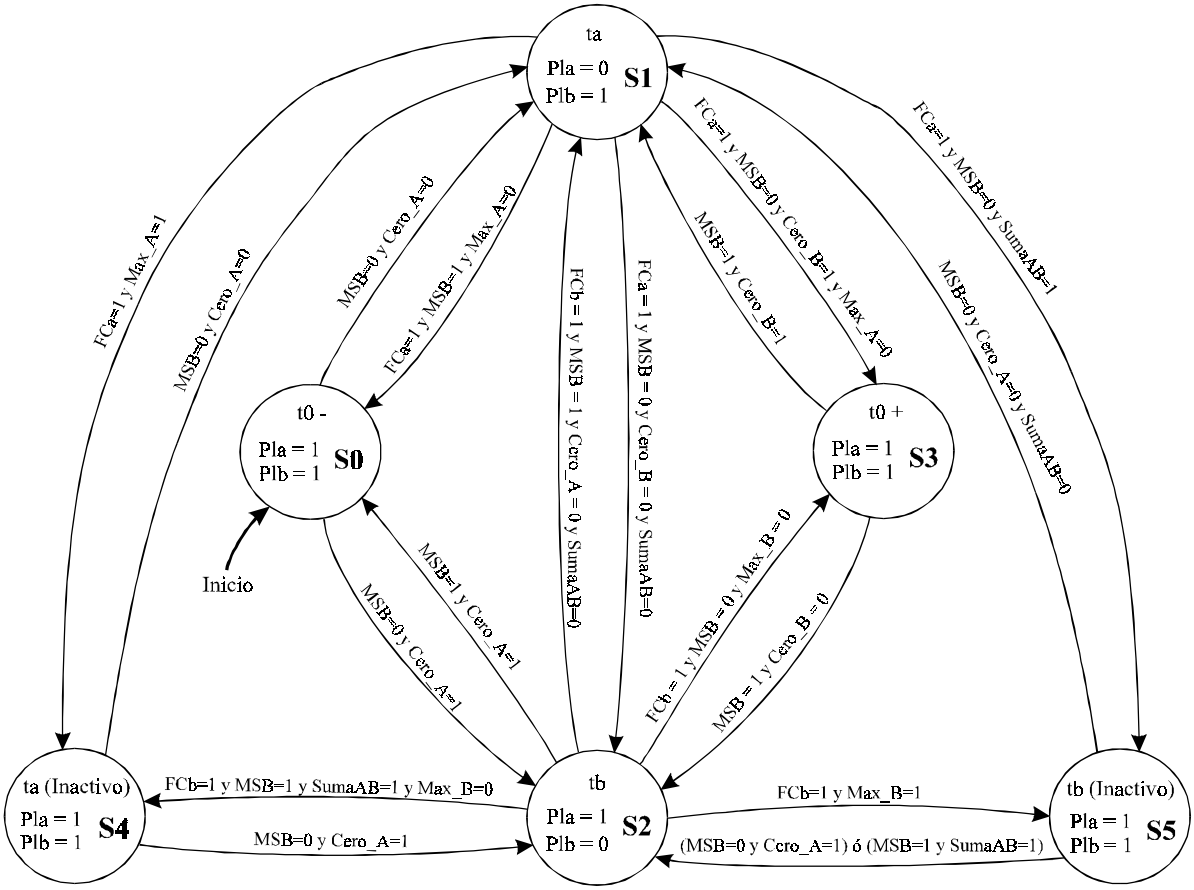


Figura 7.4: Esquema de la máquina secuencial para el modulador vectorial con implementación de los dos modos de sobremodulación.

Con respecto a la máquina secuencial descripta en 5.3, se agregaron dos nuevos estados como muestra la Fig. 7.4, estos permiten que se simplifique el diseño de la misma. Los nuevos estados S4 y S5 se utilizan para implementar los tiempos activos t_a y t_b respectivamente sin que los contadores Ct_a y Ct_b actúen.

7.2.1.1 Funcionamiento de la máquina secuencial en el modo lineal

Primero se supondrá que la SM está operando en el modo lineal. Entonces debe poder implementar las secuencias: $|t_a\ t_0^+\ t_a\ t_0^-|$, $|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|$, $|t_a\ t_b\ t_b\ t_a|$, y $|t_b\ t_0^+\ t_b\ t_0^-|$.

En el inicio la máquina secuencial se encuentra en el estado denominado S0. En el momento en que la SM arranca, se cargan los tiempos t_a y t_b en los contadores Ct_a y Ct_b respectivamente (Fig. 7.3). Suponiendo que t_a y t_b son distintos de cero y además que el subciclo es impar (***MSB***=0), la SM salta al estado S1. En S1 se implementa el tiempo t_a. Cuando termina de implementarse el intervalo de tiempo t_a, la señal ***FCa*** se pone en uno cumpliéndose la condición necesaria para que la SM salte al estado S2. En este nuevo estado se implementa el tiempo t_b a través del contador Ct_b. Cuando termina de contar Ct_b, la señal ***FCb*** se pone en uno, por lo que la SM salta al estado S3. En S3 los contadores están detenidos, teniendo cargados los intervalos de tiempo t_a y t_b implementados en el subciclo impar, en este estado se implementa el intervalo de tiempo t_0^+. Cuando comienza el subciclo par (***MSB***=1), se dan las condiciones para que la SM salte al estado S2 ya que t_b se supuso distinto de cero. En S2 se implementa nuevamente t_b, y cuando este termina de implementarse se cumplen las condiciones para que la SM salte al estado S1. Pues, las señales ***FCb***=1, ***MSB***=1, ***Cero_A***=0 y ***SumaAB***=0. En el nuevo estado S1, se vuelve a implementar t_a. Cuando aparece la señal de fin de cuenta ***FCa***, la SM salta al estado S0 en el cual se implementa el intervalo de tiempo t_0^-.

Cuando termine el subciclo par, la máquina secuencial saltará al estado S1 si el intervalo de tiempo t_a es distinto de cero, o sino saltará a S2 si $t_a = 0$.

Los estados S0 y S3 son iguales desde el punto de vista del manejo de los contadores Ct_a y Ct_b, los dos mantienen a los contadores detenidos y cargando los tiempos, pero se diferencian en la selección del vector nulo que se debe aplicar para que las conmutaciones sean mínimas. Hasta aquí se completo la implementación de una muestra siguiendo la secuencia $|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|$**.**

7.2.1.2 Funcionamiento de la máquina secuencial en secuencias particulares

Supóngase que la muestra a implementar tiene como valores de los intervalos de tiempo $t_b = 0$ y $0<t_a<\Delta t$, y que la SM se encuentra en alguno de los estados finales (S0, S4 ó S5).

En el comienzo de la implementación de la muestra, en el subciclo impar, se dan las condiciones para que salte al estado S1, desde cualquiera de los estados finales S0, S4 ó S5. Una vez que el contador Ct_a termina de implementar al intervalo de tiempo t_a, aparece la señal ***FCa***, haciendo que la SM salte al estado S3 en el cual se implementa t_0^+. Cuando termina el subciclo impar la SM tiene las condiciones necesarias para saltar al estado S1, volviéndose a implementar el tiempo t_a. Cuando la señal ***FCa*** se pone en uno nuevamente, la SM salta al estado S0 ya que se esta en el subciclo par (***MSB***=1). Cuando termina el intervalo de tiempo t_0^-, finaliza la implementación de la muestra cumpliendo la secuencia $|t_a\ t_0^+\ t_a\ t_0^-|$.

Supóngase ahora que la muestra a implementar tiene como valores de los intervalos de tiempo a $t_a = 0$ y $0<t_b<\Delta t$,, y que la SM se encuentra en los estados S0, S4 ó S5.

Al inicio de la implementación de la muestra en el subciclo impar, se dan las condiciones para que la SM salte al estado S2 implementándose t_b. Una vez que el contador Ct_b termina de contar, se activa la señal ***FCb*** haciendo que la SM salte al estado S3 en el cual se implementa t_0^+. Cuando termina el subciclo impar, la SM vuelve al estado S2 implementándose nuevamente t_b. Cuando ***FCb*** se pone en uno, la SM salta al estado S0 ya que ***MSB***=1. Cuando termina el intervalo de tiempo t_0^-, finaliza la implementación de la muestra con la secuencia $|t_b\ t_0^+\ t_b\ t_0^-|$, y la SM permanece en el estado S0.

7.2.1.3 Funcionamiento de la máquina secuencial en el modo I y II de sobremodulación

Ahora se explicará la implementación de la secuencia $|t_a\ t_b\ t_b\ t_a|$, que corresponde a muestras pertenecientes al hexágono que forman los vectores espaciales. Por lo tanto, $t_a+t_b=\Delta t$ con lo que la señal ***SumaAB*** se activa poniéndose en uno. Supongamos que la máquina secuencial se encuentra en alguno de los estados finales S0, S4 ó S5. Al inicio del subciclo impar las señales de la SM cumplirán las condiciones necesarias para que salte al estado S1. Una vez transcurrido el intervalo t_a, la señal ***FCa*** se activa produciendo el cambio

de estado de la SM a S5. En este estado se implementa t_b pero sin que actúe el contador Ct_b, simplemente se espera a que finalice el subciclo impar. Cuando éste finaliza, la SM salta al estado S2 para implementar nuevamente a t_b pero ahora haciendo uso de Ct_b. Al finalizar la implementación de t_b la SM salta al estado S4, en donde se implementa t_a. En S4 ocurre lo mismo que en S5, pero para el intervalo de tiempo t_a. Así se espera a que finalice el subciclo par, implementándose t_a sin hacer uso del contador Ct_a. Por lo tanto la SM permanece en el estado S4 hasta que comience la implementación de una nueva muestra.

7.2.1.4 Funcionamiento de la máquina secuencial en el modo six-step

Por último se detallará el funcionamiento de la máquina secuencial en el modo six-step. Por lo tanto debe poder implementar las secuencias $|t_a\ t_a|$, $|t_b\ t_b|$. Supóngase que $t_a = \Delta t$ y $t_b = 0$, entonces las señales ***Max_A*** y ***Cero_B*** se activarán y ***Max_B***, ***Cero_A*** seguirán inactivas. Cuando comience el subciclo impar la SM saltará a S1 desde el estado final en que se encuentre S0, S4 ó S5. Cuando transcurre el intervalo t_a, la SM salta a S4 en donde permanece hasta que termine el subciclo par, completándose la implementación inactiva de t_a en el subciclo par. La SM salta al estado S1 si el nuevo dato t_a es distinto de cero, y en el caso que fuera igual al tiempo de subciclo se repite el proceso descripto. En cambio, si $t_a = 0$ la SM salta a S2.

Supóngase ahora que $t_a = 0$ y $t_b = \Delta t$, por lo que solo las señales ***Max_B*** y ***Cero_A*** se activarán. Ni bien sea la señal ***MSB*** = 0, la máquina secuencial salta a S2 de manera independiente del estado inicial en que se encuantre. Una vez finalizada la implementación de t_b, la SM salta a S5. Como se mencionó anteriormente, éste estado se comporta igual que el estado S4, ya que la implementación de t_b se realiza de forma inactiva o sea sin utilizar el contador Ct_b. Para que la SM salga de este estado, debe empezar un nuevo subciclo impar. Dependiendo del nuevo valor de t_a, la SM irá a S1 ó a S2. Si $t_a = 0$ la SM salta a S2 y si el valor de t_b coincide con el tiempo de subciclo, se repite el proceso descripto implementándose nuevamente la secuencia $|t_b\ t_b|$. En cambio se $t_a = \Delta t$ y $t_b = 0$ la SM salta al estado S1 implementándose la secuencia $|t_a\ t_a|$, cumpliendo con la operación en six-step.

7.2.2 Etapa de Compensación de T_D

Esta etapa corresponde a la compensación del tiempo muerto y es independiente del algoritmo de modulación vectorial. El funcionamiento de la misma corresponde al método de compensación sin medida de la corriente descripto en el punto 6.3.3. La elección de éste método se debe a que no necesita de sensores de corriente y las desviaciones de tiempo son compensadas eficientemente, debido a que las mismas son previamente medidas. Si bien la complejidad del circuito digital es mayor que en los otros métodos, esta no tiene una importancia significativa cuando se implementa en dispositivos lógicos programable (FPLD).

En la Fig. 7.5 se muestra un diagrama en bloque de la etapa de compensación del efecto tiempo muerto. Las entradas a esta etapa son las tensiones de columna (***VR0TTL***, ***VS0TTL*** y ***VT0TTL***), previamente acondicionadas a los niveles de tensión de lógica TTL, y el PWM de salida del algoritmo de modulación vectorial (***pwmS1***, ***pwmS3*** y ***pwmS5***). La salida es el PWM al cuál se le debe agregar el tiempo de seguridad T_D (***Sw1_Corr***, ***Sw3_Corr*** y ***Sw5_Corr***). También posee una entrada para habilitar la etapa (***Enable_Comp_Td***). En caso de que ésta etapa no se utilice, el PWM de salida del algoritmo de modulación vectorial ingresa directamente a la etapa de salida.

Para la implementación de la compensación del efecto del tiempo muerto en el FPLD, se siguió con los lineamientos dados en el punto 6.4. El tiempo de seguridad T_D, para el inversor de potencia empleado, es de 2 μseg y la frecuencia de clock es de 8.375MHz; por lo tanto para medir desviaciones de tiempo de ese orden se necesita de un contador de por lo menos 5 bits.

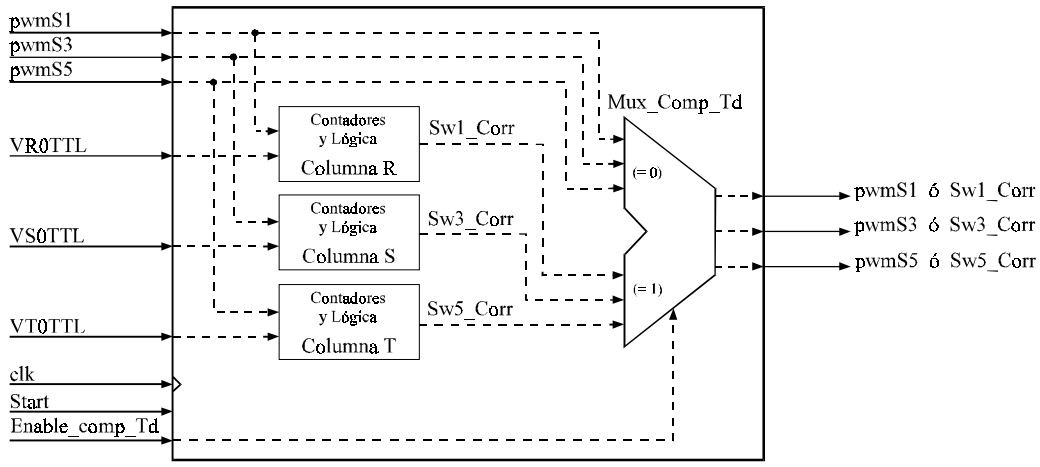


Figura 7.5: Diagrama en bloques de la etapa de compensación del efecto de tiempo muerto.

Para cada columna del Inversor R, S y T (Fig. 2.1) se utilizaron dos contadores de 4bits (74LS193), tomados de la librería que provee el software de Altera. Estos se eligieron sobre la base de que poseen las líneas de salida ***Carry*** y ***Borrow***, como se requiere en 6.4. La señal de ***Carry*** pone al PWM corregido (***Sw1_Corr***, ***Sw3_Corr*** ó ***Sw5_Corr***) en uno y la señal de ***Borrow*** lo pone en cero. La carga del contador con el dato FF_{Hex} se debe a que la señal de ***Carry*** aparece cuando el contador pasa de la cuenta FF_{Hex} a 00_{Hex} y es necesario para que el modulador arranque. La iniciación de estos contadores se realiza con la señal ***Start***.

En la Tabla 7.2 se muestran las condiciones lógicas que deben tener las señales de entrada a la etapa de compensación, para que los contadores funcionen de la manera esperada. Estas se muestran para la columna R, valiendo también para las otras dos columnas del Inversor con solo cambiar la letra "R" por "S" ó "T".

La etapa de compensación de T_D trabaja siempre que tenga todas las entradas. La señal de habilitación (***Enable_Comp_Td***) actúa sobre un multiplexer llamado Mux_Comp_Td, el cual permite seleccionar las señales del PWM de salida del Algoritmo de Modulación (***pwmS1***, ***pwmS3*** y ***pwmS5***) ó las señales de PWM corregido (***Sw1_Corr***, ***Sw3_Corr*** y ***Sw5_Corr***). La salida de este multiplexer se conecta con la entrada del bloque de implementación del tiempo de seguridad en la etapa de Salida.

Tabla 7.2: Condiciones lógicas para las señales de entrada de clock ascendente y descendente.

VR0TTL	**PwmS1**	**up_R**	**Down_R**	Estado contador
0 ($-V_{cc}/2$)	*0*	*1*	*1*	*Detenido*
0 ($-V_{cc}/2$)	*1*	**clk**	*1*	*Cuenta ascendente*
1 ($+V_{cc}/2$)	*0*	*1*	**clk**	*Cuenta descendente*
1 ($+V_{cc}/2$)	*1*	*1*	*1*	*Detenido*

7.3 Resultados de simulación digital

El modulador vectorial universal se implementó en la placa de demostración de ALTERA (UP1) utilizando el dispositivo EPF10K20RC240-4, siendo este un CPLD SRAM de la familia FLEX. El mismo puede ser implementado en otro dispositivo más pequeño de la familia FLEX, como ser un EPF10K10.

Al igual que en la implementación del modulador vectorial lineal del capítulo 4, se utilizan las herramientas de software MAX+plus II, provisto por ALTERA. El diseño se realizó en AHDL, se compiló y simuló utilizando MAX+Plus II. En esta sección se presentarán las simulaciones digitales correspondientes al funcionamiento en sobremodulación y compensación de tiempo muerto. La etapa de entrada es la misma que se utilizó para el modulador lineal y se detalla en el capítulo 4. Para las simulaciones digitales, que incluyen los retardos del dispositivo EPF10K20, se eligió una señal de frecuencia elevada para poder analizar con detalle el funcionamiento del modulador.

En el apéndice A se encuentran los diagramas esquemáticos de los circuitos implementados en el FPLD. En el apéndice B está el listado del diseño en AHDL del modulador vectorial universal. El apéndice C tiene los datos de los tiempo t_a y t_b almacenados en las respectivas memorias. El apéndice D posee el programa en C utilizado para la comunicación del modulador con la PC.

En la Fig. 7.6 se muestran los resultados obtenidos de una simulación digital, para el modulador vectorial universal funcionando en el modo I de sobremodulación con un índice de modulación de 0,935. Las señales ***Dina[7..0]*** y ***Dinb[7..0]***, compuestas por 8 líneas cada una, tienen los datos de los tiempos t_a y t_b respectivamente; en donde el número significa la cantidad de ciclos de clock que debe contar cada contador. Las señales ***LD[15..13]***

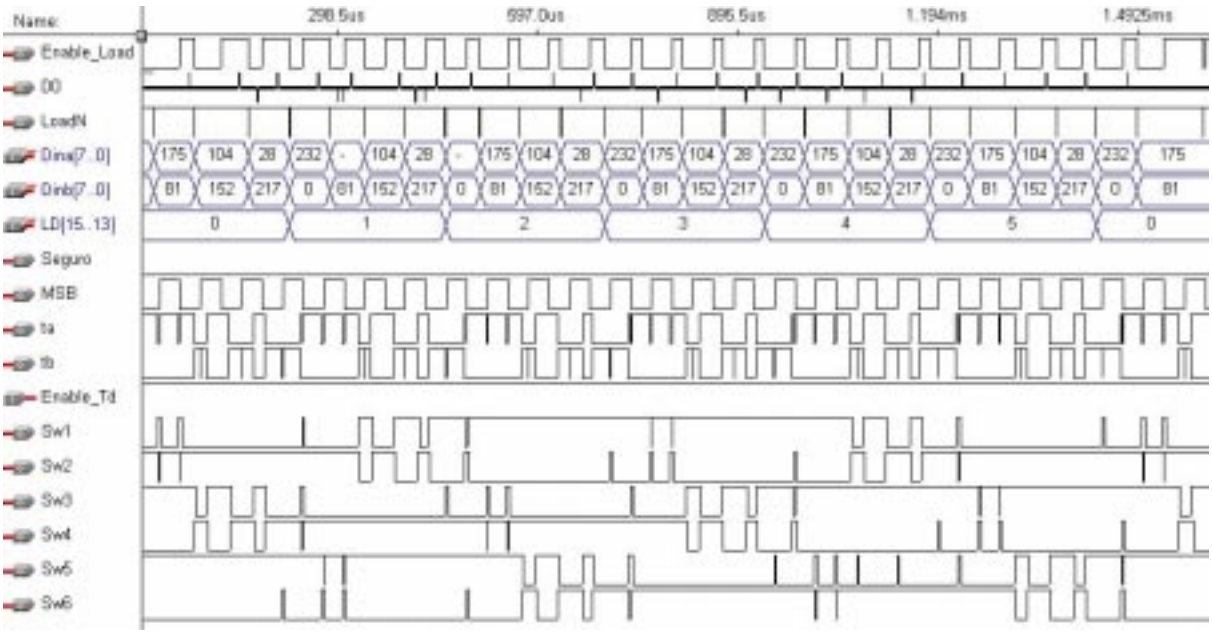


Figura 7.6: Resultados de simulación digital para el Modulador Vectorial Universal funcionando en el modo I de sobremodulación, con m = 0,935 y f = 681,576Hz (4 muestras por sector).

representan al sector, como se vio en el capítulo 4 (Tabla 4.1). Un período de ***MSB*** corresponde a un período de conmutación del modulador. Cuando la señal ***ta*** o ***tb*** está en uno, el modulador se encuentra implementando el intervalo de tiempo t_a o t_b respectivo; pero cuando ambas están en cero, el modulador implementa el intervalo de tiempo correspondiente a los vectores nulos. Las señales ***SW1*** a ***SW6***, correspondientes a las señales de activación de las llaves del inversor, son activas en cero. Las señales ***Enable_Load***, ***D0*** y ***LoadN*** muestran el funcionamiento de la etapa de entrada, y la carga de los datos. La señal ***Enable_Td*** esta en uno porque la etapa de salida implementa el tiempo de seguridad. La señal ***Seguro*** permanece en cero, ya que no se repite la implementación de la misma muestra. Trazando una línea vertical imaginaria en cada sector de la Fig. 7.6 y moviéndolo en la implementación de cada muestra, pueden verificarse los estados de las llaves correspondientes a los vectores espaciales de cada sector, verificando el correcto funcionamiento del modulador en cada sector.

En la Fig 7.7, que es una ampliación de la Fig. 7.6 sobre los sectores 1 y 2, puede apreciarse con detalles el funcionamiento en el modo I de sobremodulación. La primer muestra t_a=232 y t_b=0, corresponde a la implementación de la secuencia particular $|t_a\, t_0^+\, t_a\, t_0^-|$, es por ello que durante todo el período de ***MSB*** la señal ***tb*** permanece en cero. Durante la implementación de t_a = 232, la señal ***ta*** permanece en uno y el estado de las señales ***SW1*** a ***SW6*** corresponde al vector espacial $V_1 = (S_1\ S_4\ S_6)$ (Fig. 2.2); vale recordar que las llaves S1

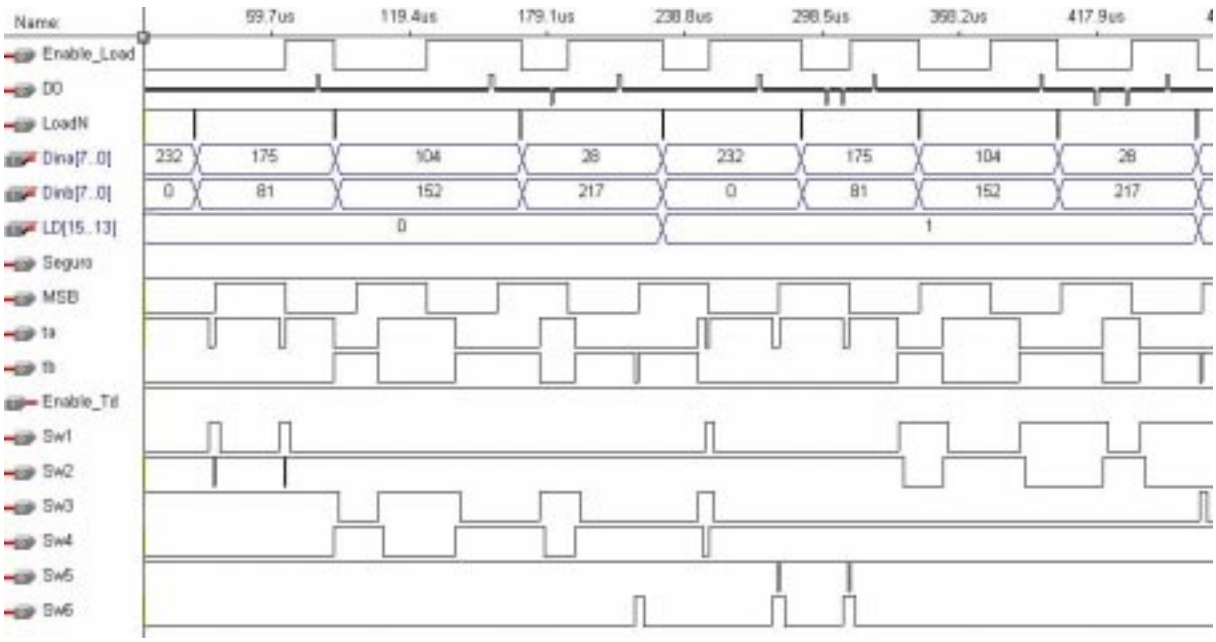


Figura 7.7: Ampliación de la Fig.7.6.

a S6 son activas en cero. Al finalizar la implementación de t_a se implementa el tiempo t_0^+, hasta completar los 256 períodos de reloj que terminan cuando ***MSB*** pasa a uno. Aquí se observa el cambio de estado en ***SW1*** y ***SW2*** debido al vector nulo $V_7 = (S_2\ S_4\ S_6)$. Cuando ***MSB***=1, se repite el primer subciclo. En el quinto período de ***MSB***, el cual corresponde a la implementación en el sector 2 de la muestra en cuestión, se verifica que en la implementación de t_a el vector espacial es $V_2 = (S_1\ S_3\ S_6)$. La implementación de la segunda y tercer muestra (t_a=175, t_b=81 y t_a=104, t_b=152), verifican el funcionamiento del modulador para implementar la secuencia $|t_a\ t_b\ t_b\ t_a|$ que pertenece al hexágono. El funcionamiento del modulador en la zona lineal, se verifica al implementar la cuarta muestra (t_a=28, t_b=217), que corresponde a la secuencia $|t_a\ t_b\ t_0^+\ t_b\ t_a\ t_0^-|$. En las conmutaciones de las señales ***SW***, se pueden observar los retardos debido al tiempo de seguridad T_d.

En la Fig. 7.8 se muestran los resultados obtenidos de una simulación digital, para el modulador vectorial universal funcionando en el modo II de sobremodulación con un índice de modulación de 0,98. En esta figura se muestra un poco más de un período de la señal de referencia, y puede observarse como las tres fases están 120º desfasadas una de la otra. Esta figura verifica el funcionamiento en el modo II de sobremodulación.

La Fig. 7.9 es una ampliación de la Fig. 7.8, y muestra con más detalles el funcionamiento en los sectores 1, 2 y parte del sector 3 para el modo II de sobremodulación.

Figura 7.8: Resultados de simulación digital para el Modulador Vectorial Universal funcionando en el modo II de sobremodulación, con m = 0,98 y f = 681,576Hz (4 muestras por sector).

La primer muestra t_a=255 y t_b=0, corresponde a la implementación de la secuencia particular $|t_a\ t_a|$. Debido a que se está en el sector I, el vector espacial activo aplicado es $V_1 = (S_1\ S_4\ S_6)$; en cambio para la quinta y novena muestra los vectores espaciales serán $V_2 = (S_1\ S_3\ S_6)$ y $V_3 = (S_2\ S_3\ S_6)$ correspondientes a los sectores II y III respectivamente. La segunda y tercer muestra (t_a=175, t_b=81 y t_a=104, t_b=152), corresponden a la secuencia $|t_a\ t_b\ t_b\ t_a|$ que pertenece al hexágono. Y la cuarta muestra (t_a=0, t_b=255) corresponde a la implementación de la secuencia particular $|t_b\ t_b|$, y es donde el vector modificado salta del hexágono al vector espacial correspondiente.

La Fig. 7.10 muestra los resultados obtenidos de una simulación digital, para el modulador vectorial universal funcionando en el modo Six-Step (m = 1). En esta figura se muestran los seis sectores, completando un poco más de un período de la señal de referencia, y verificando el desfasaje de 120º entre las tres señales de excitación de las llaves del inversor.

La Fig 7.11 muestra el funcionamiento del modulador con la compensación del efecto de tiempo muerto; en ella se muestran las señales de las tres columnas del inversor necesarias para verificar su funcionamiento, siendo:

pwmS1, ***pwmS3***, ***pwmS5***: las señales de PWM de salida del algoritmo de modulación vectorial, o sea la salida de la Tabla de estados

Sw1_Corr, Sw3_Corr, Sw5_Corr: las señales de PWM de salida del algoritmo de modulación vectorial con compensación del efecto de tiempo muerto

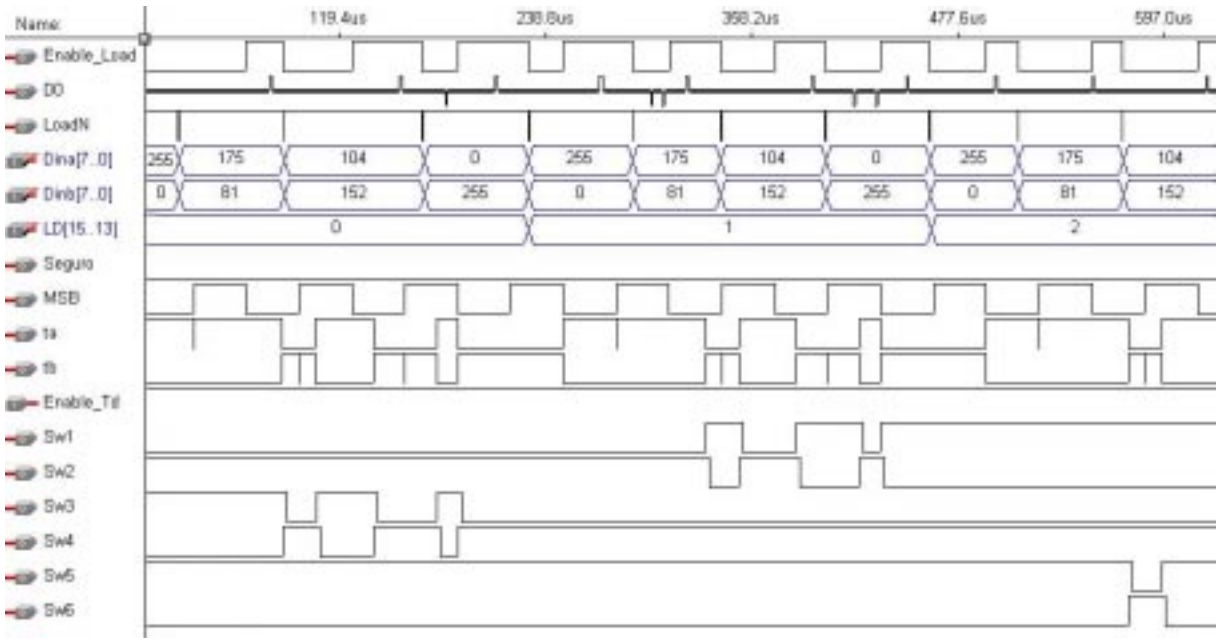


Figura 7.9: Ampliación de la Fig.7.8.

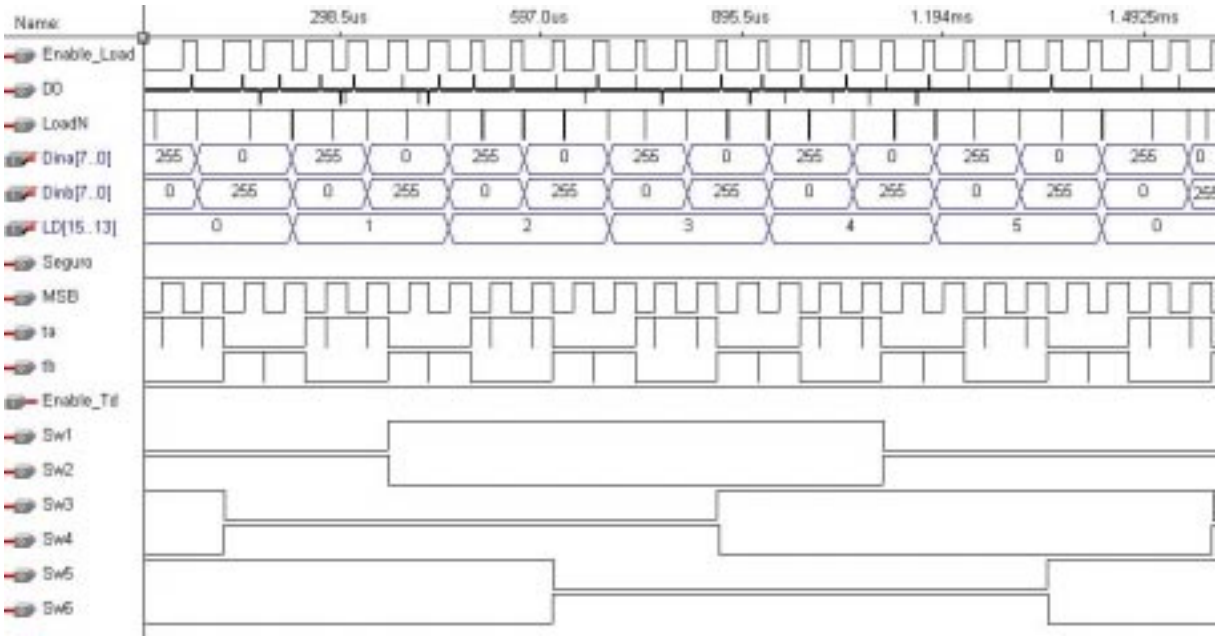


Figura 7.10: Resultados de simulación digital para el Modulador Vectorial Universal funcionando en el modo Six-Step de sobremodulación, m = 1 y f = 681,576Hz (4 muestras por sector).

VR0TTL, VS0TTL, VT0TTL: las tensiones de columna acondicionadas a niveles TTL. Estas señales se generan manualmente como una señal de test al simulador. El tiempo de encendido y apagado de cada llave se estima, teniendo la precaución de que estos no superen el tiempo de seguridad T_D.

Sw1, Sw2: las señales de excitación de las llaves del inversor para la columna R, con T_d

Sw3, Sw4: las señales de excitación de las llaves del inversor para la columna S, con T_d

Sw5, Sw6: las señales de excitación de las llaves del inversor para la columna T, con T_d

Carry_R, Carry_S, Carry_T: las señales de carry para cada columna

Borrow_R, Borrow_S, Borrow_T: las señales de borrow para cada columna

En la Fig. 7.11 las señales de carry y borrow muestran como ponen en uno y cero respectivamente, a la señal compensada de PWM. En esta figura también puede observarse el tiempo de seguridad en cada columna, el cual es para esta simulación de 3 μseg.

Cuando ***pwmS1*** se pone en "1" la tensión de la columna (***VR0TTL***) sigue en "0", debido a que $i_R > 0$, hasta que se cierra la llave S1 del Inversor de potencia. Este intervalo de tiempo t1 se compensa a partir de cuando la señal ***pwmS1*** se pone en "0", y la señal ***Sw1_Corr*** permanece en "1" durante el tiempo t1. Esto puede verificarse en la Fig. 7.11, en donde esta acotado el tiempo t1 en la columna R. Lo mismo ocurre para la columna T, ya que también tiene corriente positiva ($i_T > 0$).

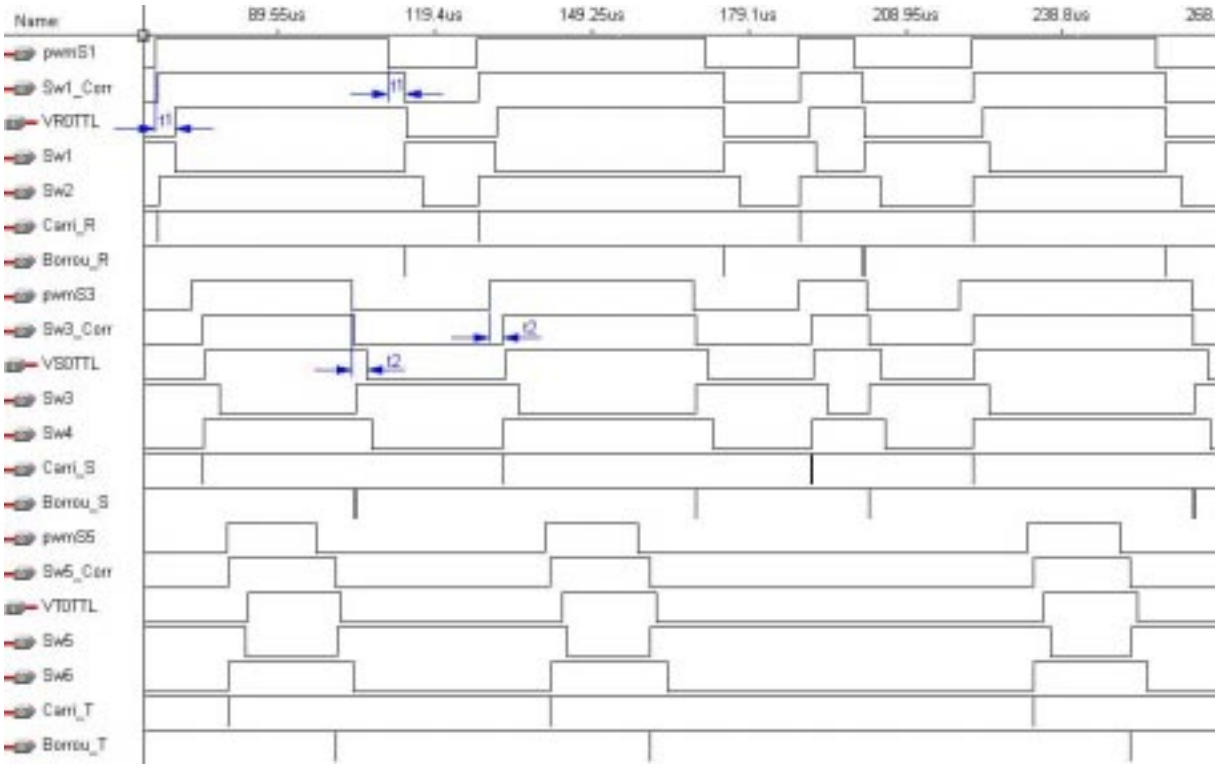


Figura 7.11: Resultados de simulación digital para el Modulador Vectorial Universal funcionando con compensación del efecto de tiempo muerto.

En la columna S la corriente es negativa ($i_S < 0$), entonces cuando ***pwmS3*** pasa a "0" la tensión de la columna (***VS0TTL***) permanece en "1" durante el intervalo de tiempo t2, hasta que se cierra la llave S4 del Inversor de potencia. El intervalo de tiempo t2 se compensa a partir de cuando la señal ***pwmS3*** se pone en "1", dejando la señal ***Sw3_Corr*** en "0" durante el tiempo t2, como muestra la Fig. 7.11.

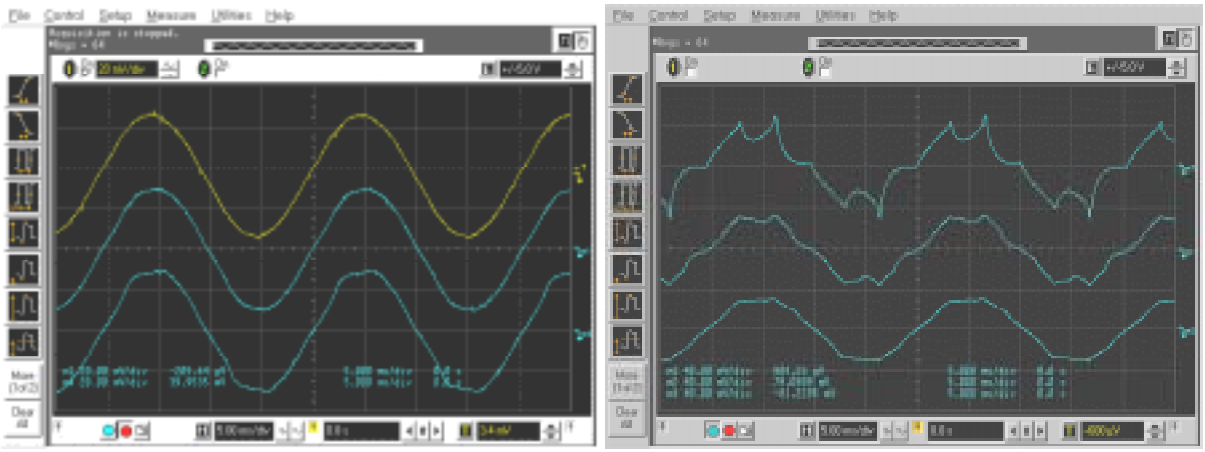
7.4 Resultados experimentales

El modulador vectorial universal se implementó en la placa de demostración de ALTERA (UP1) utilizando el dispositivo EPF10K20RC240-4 de la familia FLEX. A los efectos de poder evaluar el funcionamiento total del modulador, incluida la compensación de tiempo muerto, se implementó un inversor conectado a una carga R-L (5Ω, 38,6mH) en configuración estrella.

En el ambiente de la PC se generaron distintos vectores de referencia para probar el funcionamiento del modulador en los distintos escenarios: Modulación lineal, sobremodulación en Modo I, Modo II y Six-Step, con y sin compensación de tiempo muerto.

La Fig. 7.12 muestra la forma de onda de la corriente de carga para la transición del funcionamiento del modulador desde el Modo Lineal al Six-step, pasando por el Modo I y Modo II de sobremodulación, con una señal de referencia de 50Hz. En la Fig.7.12.a, empezando desde arriba, se muestra la corriente para el límite de la modulación lineal y las dos corrientes de abajo corresponden al Modo I de sobremodulación. La escala empleada en esta figura es de 20mV/Div. En la Fig.7.12.b, empezando desde abajo, se muestra la corriente para el límite máximo del Modo I de sobremodulación, pasando por el Modo II y alcanzando el Modo de funcionamiento Six-Step. En esta figura la escala es de 40mV/Div.

Para mostrar el funcionamiento del modulador vectorial universal implementando la compensación del tiempo muerto, se generó una señal de referencia con un bajo índice de modulación (m = 0,393). En la Fig. 7.13 se muestra la distorsión en la corriente de carga debida al efecto del tiempo muerto, como se vio en el Capítulo 6; y la corriente de carga con la compensación de dicho efecto. Por un lado en el cruce por cero, se observa claramente la distorsión cuando no está compensado el tiempo muerto y como se corrige al compensar. Además en la semionda positiva se observa claramente como aumenta la amplitud de la corriente al compensar el tiempo muerto, tal como se previó en el Capítulo 6. La corriente de carga con el efecto de tiempo muerto compensado tiene una mayor amplitud, debido a que la



a) *b)*

Figura 7.12: Corriente de carga para 50Hz: Transición del modo lineal al modo Six-Step.

a) Transición del Modo lineal al Modo I de sobremodulación (1mV = 22mA)

b) Transición del Modo II de sobremodulación al Six-Step (1mV = 22mA)

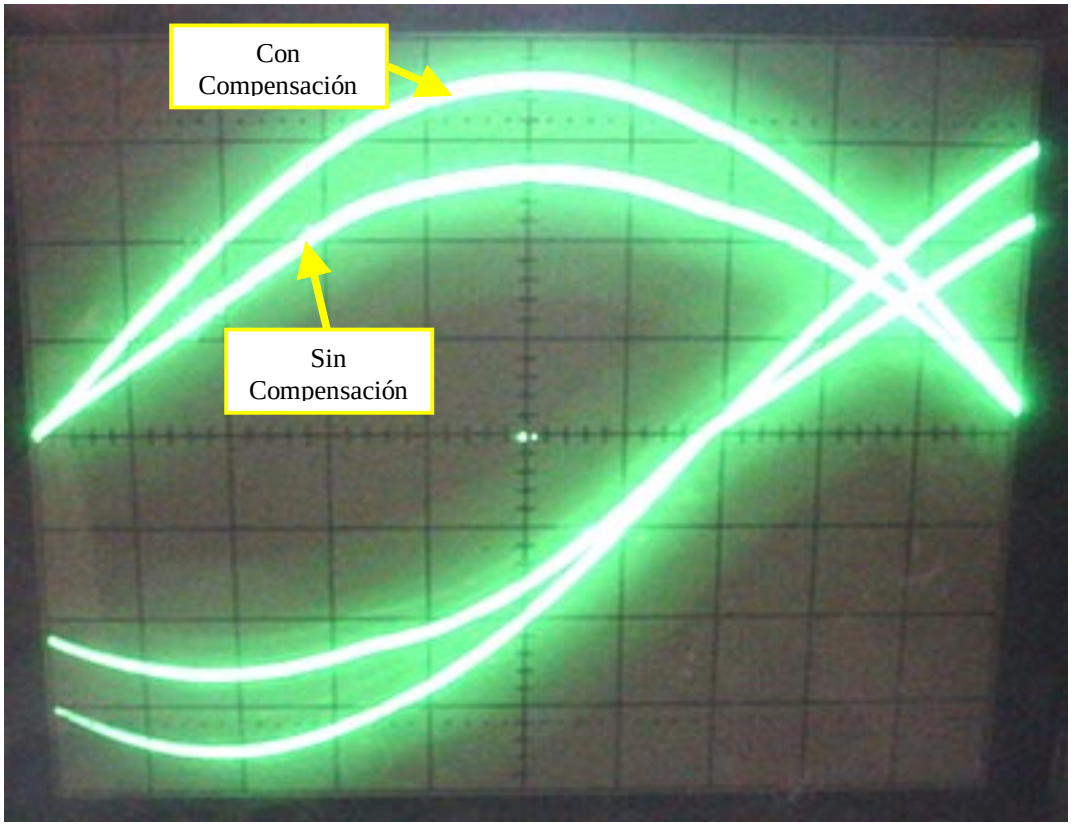


Figura 7.13: Corriente de carga para el funcionamiento del modulador sin y con compensación del tiempo muerto.

compensación le permite ganar la "caída de tensión" que le produce el efecto del tiempo muerto, tal como se vio en 6.2. De esta manera se comprueba experimentalmente el buen funcionamiento del modulador vectorial universal implementando la compensación del tiempo muerto.

Otra forma de analizar el funcionamiento del Modulador Vectorial Universal es haciendo uso del locus de corriente. Para esto se utilizó el integrado AD2S100AP de Analog Device, en donde se conectaron las salidas en cuadratura del integrado a las entradas X e Y del osciloscopio. Las señales de referencia que se utilizaron son: m = 0.393, m = 0.785, m = 0.907 en el Modo Lineal, y m = para el Six-Step; todas con frecuencia de 20 Hz. Como se vio en el Capítulo 5, m = 0.785 corresponde al máximo índice de modulación sinusoidal (SPWM) y m = 0,907 corresponde al máximo índice de modulación vectorial (SVM).

En la Fig 7.14 se muestra el locus de corriente para el funcionamiento del modulador sin compensación de tiempo muerto, con las 4 señales de referencia anteriores. El hexágono corresponde al funcionamiento en Six-Step, pero se visualizan las líneas debido al trazo del

fósforo del osciloscopio ya que los vectores solo están en los vértices del mismo. El círculo más grande corresponde al límite entre modo lineal (m = 0.907), el círculo que le sigue corresponde al máximo índice obtenible con modulación sinusoidal (m = 0.785). La diferencia entre estos círculos es lo que se gana utilizando SVM en vez de SPWM. El círculo más grande debería ser tangente al hexágono, pero no lo es justamente por la caída de tensión generada por el tiempo muerto. El trazo más pequeño corresponde al 50% del máximo índice en SPWM (m = 0.393), puede verse que no es un círculo perfecto ya que las corrientes tienen distorsión debido al efecto de tiempo muerto.

En la Fig 7.15 se muestra el locus de corriente para el funcionamiento del modulador con compensación de tiempo muerto. Aquí el hexágono permanece sin alteración y el círculo correspondiente a m = 0.907 se hace tangente al hexágono como indica la teoría en el capítulo 2. En el trazo del interior correspondiente a m = 0.393 se puede observar que es un círculo de mayor diámetro y sin distorsión, debido a la compensación del efecto de tiempo muerto.

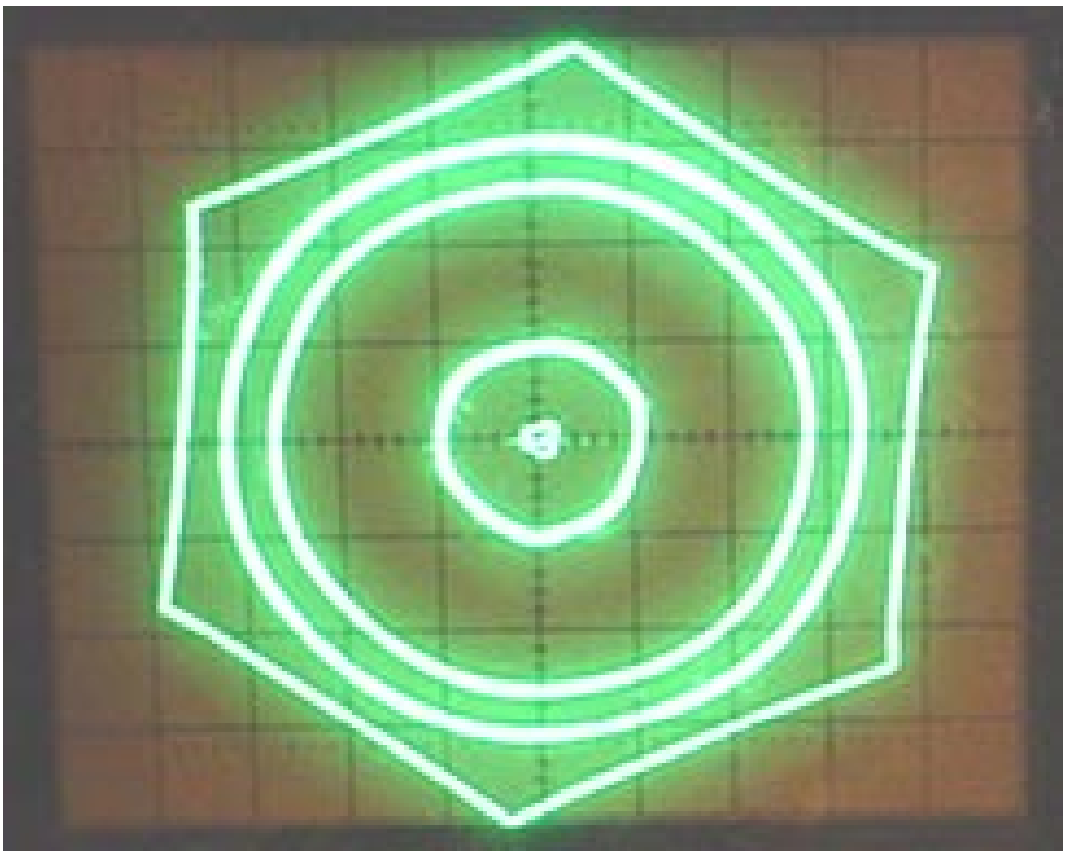


Figura 7.14: Locus de la corriente de carga para el funcionamiento del modulador sin compensación del tiempo muerto.

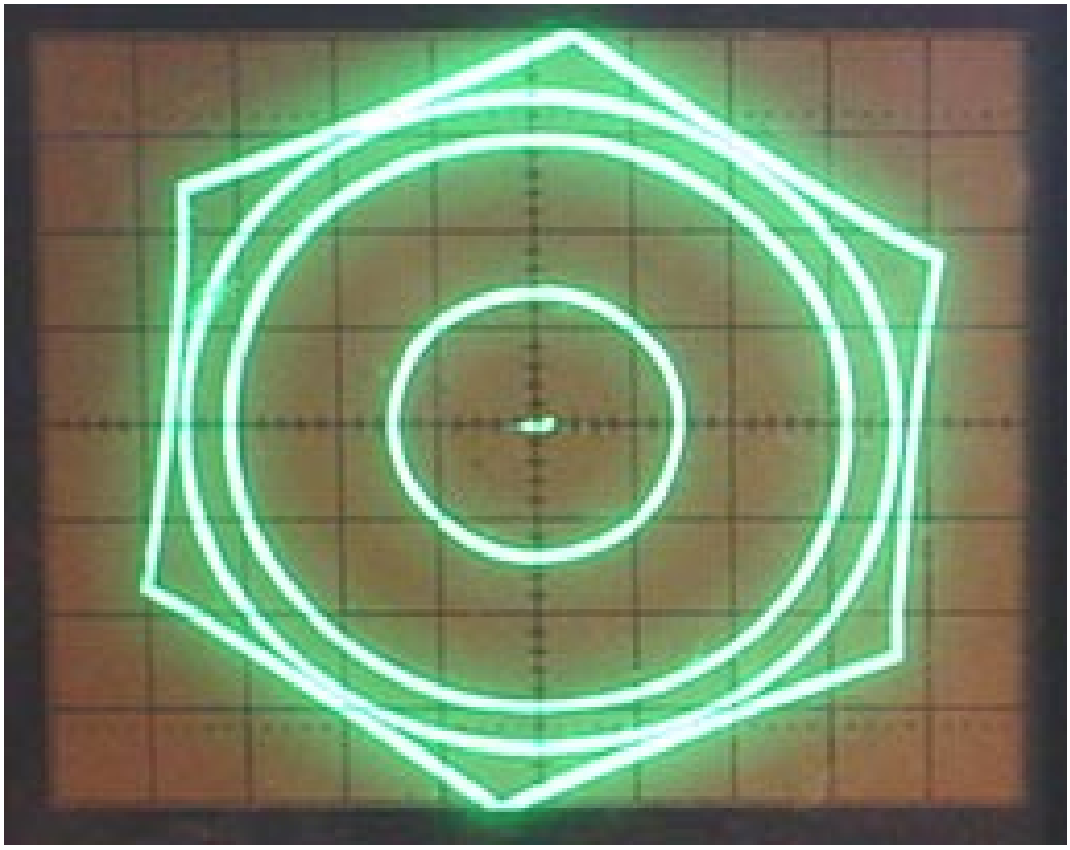


Figura 7.15: Locus de la corriente de carga para el funcionamiento del modulador con compensación del tiempo muerto.

7.5 Resumen

En este capítulo se realizó el diseño de modular universal que fuera objetivo de la presente Tesis. En base al diseño del modulador lineal presentado en el Capítulo 4 se modificó la máquina secuencial para tener en cuenta los distintos modos de sobremodulación. Esto también condujo a modificar los tiempos grabados en las memorias. También se incorporó una compensación de tiempo muerto en forma casi independiente del modulador, la cual puede ser incorporada o no según el deseo del usuario.

El modulador vectorial universal se implementó utilizando el FPDL EPF10K20RC240-4 de ALTERA y dos EPROMs. En las memorias se almacenaron los tiempos de activación de las llaves del inversor, y en el FPLD se implementó la lógica del algoritmo del modulador junto con las etapas de comunicación con la PC y del Inversor de Potencia, incluyendo la compensación de tiempo muerto y la implementación del tiempo de

seguridad T_D. El modulador vectorial aquí presentado cumple con todos los objetivos propuestos:

a) Variación del índice de modulación desde cero hasta uno con plena utilización de la tensión continua disponible.
b) Posibilidad de compensar el efecto de tiempo muerto evitando las distorsiones para bajos índices de modulación.
c) Mínimas conmutaciones.
d) Flexibilidad para adaptarse a distintos inversores.

Asimismo se presentaron resultados experimentales que muestran el buen funcionamiento del modulador.

8. CONCLUSIONES

8.1. Conclusiones del trabajo realizado

En este trabajo se propuso e implementó un modulador vectorial universal para inversores trifásicos de potencia, que funciona con el principio de la modulación del vector espacial.

Primero se analizó detalladamente la teoría del vector espacial y su aplicación a la modulación vectorial de ancho de pulso en inversores trifásicos de tensión, obteniéndose los tiempos de conmutación de las llaves del inversor de potencia. Se comparó la modulación vectorial con la modulación sinusoidal, mostrándose las ventajas de la primera en la utilización de la tensión de la tensión continua. Como conclusión del análisis se presentó un diagrama en bloques para una posible implementación del modulador. Luego se estudiaron algunas familias de dispositivos lógicos programables de distintos fabricantes (Altera, Xilinx y Atmel) con el fin de elegir los dispositivos adecuados para la implementación buscada. Entonces se prosiguió con la implementación de un modulador vectorial lineal, el cual se implementó con un FPLD de Altera, el EPM7128S, y dos memorias EPROMs. En las memorias se almacenan los tiempos de activación de las llaves del inversor, y en el FPLD se implementa la lógica del algoritmo del modulador junto con las etapas de comunicación con la PC y con el Inversor de Potencia, incluyendo la implementación del tiempo de seguridad T_D. El diseño del modulador completo utiliza el 86% del dispositivo elegido, el cual está cerca del limite máximo recomendado. El modulador lineal presentado aquí está especialmente indicado en aplicaciones en las que no son críticos: ni el uso completo de la tensión continua (por ejemplo inversores de baja tensión), ni la distorsión debida al tiempo muerto en la excitación de las llaves de potencia (cuando no se usan índices de modulación bajos, por ejemplo motores que no deban girar a muy baja velocidad). Los resultados mostrados en el capítulo 4 son ampliamente satisfactorios.

Satisfechos con los resultados obtenidos, se continuó con el análisis para obtener una transición suave del modo sinusoidal (lineal) hasta el de onda cuadrada para un aprovechamiento completo de la tensión continua. Se analizaron los métodos de sobremodulación observando que pueden dividirse en tres partes: a) Modo I, en el cual el vector de referencia se mueve parte sobre la circunferencia y parte sobre el hexágono, vale

para $0,90 < m < 0,95$. b) Modo II, en el cual el vector de referencia se mueve sobre el hexágono o permanece en los vectores espaciales, vale para $0,95 < m < 0,999$. c) Modo six-step, en el cual el vector de referencia se queda sobre los vértices del hexágono, vale para $m = 1$. Siempre buscando superar las limitaciones del modulador lineal desarrollado, se analizó el efecto que produce el tiempo de retardo en la excitación de las llaves impuesto a la señal de control de PWM para evitar el cortocircuito de cada columna y su posible compensación. El tiempo muerto generado en la excitación produce una disminución en la componente fundamental e incrementa los armónicos de bajo orden y está estrechamente relacionado con la fase de la corriente de salida. Se analizaron distintos métodos para mitigar sus efectos y se propuso compensarlo midiendo el tiempo de desviación entre la señal de PWM y la tensión real de la columna. Este método no requiere de sensores de corriente, puede aplicarse a cualquier modulador y compensa eficientemente toda desviación del tiempo de la señal de control de PWM, ya que realiza una medición de la misma. Como elemento de medición utiliza optoacopladores para sensar las tensiones de cada columna.

Finalmente se desarrolló un modulador vectorial universal que puede trabajar desde la zona lineal al six-step sin interrupción, aprovechando el 100% de la tensión continua. En él se implementa una compensación del efecto de tiempo muerto que aumenta significativamente la amplitud de la señal de salida y mejora su distorsión armónica cuando se trabaja con índices de modulación bajos. El modulador está diseñado para comunicarse con una PC utilizando el bus ISA y entrega a la salida las seis señales de control de las llaves del inversor con los tiempos de seguridad incluidos. La implementación del modulador vectorial universal se realizó en un dispositivo lógico programable de la familia FLEX de Altera (EPF10K20RC240-4). Para el diseño se utilizó el paquete de software Max+Plus II, en donde se utilizó la herramienta AHDL. El uso del HDL como herramienta de diseño digital es muy poderosa, permitiendo realizar hardware mientras se escribe un programa. Esto permite realizar cambios en el hardware una vez armado el prototipo, acelerando el tiempo de puesta en marcha. Antes de la programación del dispositivo, se realizaron varios vectores de test que sirvieron para probar el funcionamiento del modulador en las peores condiciones y en los distintos modos de operación. Una vez programado el dispositivo, se obtuvieron los resultados experimentales mostrados en el capítulo 7. Éstos evidencian el correcto funcionamiento del modulador vectorial universal trabajando tanto en la zona lineal como en los tres modos de sobremodulación: Modo I, Modo II y Six-Step. La compensación del

efecto del tiempo muerto también se verificó experimentalmente; para ello se obtuvieron las gráficas de los locus de corriente, en donde se pudo apreciar muy bien la compensación de dicho efecto. En síntesis se obtuvo un modulador con una implementación muy simple libre de ajustes, sólo se incorporan algunas llaves para darle mayor flexibilidad en su conexión con la PC y el inversor de potencia.

8.2. Propuestas de Trabajos futuros

En relación con el trabajo realizado quedan abiertas distintas líneas tanto en lo que respecta a la implementación digital, como a su aplicación a otros convertidores de potencia.

En lo referente a los aspectos digitales se puede pensar en modificar la interfaz para obtener las señales de referencia y también en lograr un avance en el empleo de distintos dispositivos. Si bien el modulador presentado se diseñó con una interfaz al Bus ISA de la PC, el desarrollo presentado puede cambiarse fácilmente para lograr una interfaz con cualquier otro sistema digital. Esto se logra simplemente reprogramando la etapa de decodificación dentro del FPLD. Por ejemplo en aplicaciones industriales reales se puede desarrollar una interfaz a alguno de los Buses industriales. También pensando en producciones en mayor escala se puede pensar en desarrollar un circuito con máscara como paso intermedio a la producción de un integrado “custom”.

En lo que respecta a los aspectos de electrónica de potencia, hoy en día está recibiendo mucha atención el empleo de técnicas de modulación vectorial aplicadas al control de convertidores multiniveles. Su implementación es algo más complicada pues al aumentar los niveles de conmutación también se multiplican las combinaciones posibles de vectores. De todos modos es un área abierta en la que se podría investigar la factibilidad de lograr un desarrollo similar al presentado en esta Tesis.

Referencias

[Alt 95] ALTERA, "CPLDs vs. FPGAs Comparing High-Capacity Programmable Logic", *Product Information Bulletin 18*, February 1995, ver.1.

[Alt 97a] ALTERA, "Max+Plus II Getting Started Manual", *User Guide*, Septiembre 1997, ver.8.1.

[Alt 97b] ALTERA, "University Program Design Laboratorie Package", *User Guide*, Agosto 1997, ver.1.

[Alt 98a] ALTERA, "MAX 7000 Programmable Logic Device Family", *Data Sheet*, July 1998, ver.5.03.

[Alt 98b] ALTERA, "FLEX 10K Embedded Programmable Logic Family", *Data Sheet*, May 1998, ver.3.11.

[Bak 00] A. R. Bakhshai, G. Joos, P. K. Jain and H. in, "Incorporating the Overmodulation Range in Space Vector Pattern Generators Using a Classification Algorithm", *IEEE Trans. on Power Electronics*, Vol. 15, Nº 1, January 2000.

[Bos 97] B. K. Bose, *Power Electronics and Variable Frequency Drives*, IEEE PRESS, New York 1997.

[Bow 75a] S.R. Bowes and B. M. Bird, "Novel approach to the analysis and synthesis of modulation processes in power convertors", *Proceedings of IEE*, vol. 122 N° 5, pp. 507-513, May 1975.

[Bow 75b] S. R. Bowes, "New sinusoidal pulsewith-modulated invertor", *Proceedings of IEE*, vol. 122 N° 11, pp. 1279-1285, November 1975.

[Bow 97] S. R. Bowes and Y. Lai, "The Relationship Between Space-Vector Modulation and Regular-Sampled PWM", *IEEE Trans. on Industrial Electronics*, Vol. 44, Nº 5, pp. 670-679, October 1997.

[Buj 75] G. Buja and G. Indri, "Improvement of pulse width modulation techniques", *Archiv für Elektrotechnik*, vol. 57, pp. 281-289, 1975.

[Buj 77] G. S. Buja and G. B. Indri, "Optimal pulsewidth modulation for feeding AC motors", *IEEE Trans. on Industry Applications*, vol. IA-13, pp. 38-44, January/February 1977.

[Buj 80] G. Buja, "Optimum output waveforms in PWM inverters", *IEEE Transactions on Industry Applications*, vol. IA-16, pp. 830-836, Nov.-Dec. 1980.

[Buj 82] G. Buja and P. Fiorini, "Microcomputer control of PWM inverters", *IEEE Transactions on Industrial Electronics*, vol. IE-29, pp. 212-216, Aug. 1982.

[Del 91] Del Valle J., Rodriguez P. y Contreras T., "Técnicas de modulación vectorial en inversores trifásicos", *Brazilian Power Electronics Conference (COBEP'91)*, Florianópolis, BRASIL, Dec. 1991, pp 231-238.

[Ham 00] J. O. Hamblen and M. D. Furman, *Rapid Prototyping of Digital Design – A Tutorial Approach*, Kluwer Academic Publishers, 2000.

[Han 92] P. G. Handley, and J. T. L. Boys, "Resolution Corrected Modulation: the practical realization of ideal PWM waveform", *IEE Proceedings-B*, Vol. 139, Nº 4, pp. 402-408, July 1992.

[Hol 87] J. Holtz, P. Lammert and W. Lotzkat, "High-Speed Drive System with Ultrasonic MOSFET PWM Inverter and Single-Chip Microprocessor Control", *IEEE Trans. on Industry Applications*, Vol. 23, Nº 6, pp. 1010-1015, Nov/Dec 1987.

[Hol 91] J. Holtz and E. Bube, "Field-Oriented Asynchronous Pulse-Width Modulation for High-Performance ac Machine Drives Operating at Low Switching Frequency", *IEEE Trans. on Industry Applications*, Vol. 27, Nº 3, pp. 574-580, May-June 1991.

[Hol 92] J. Holtz and B. Beyer, "Optimal synchronous pulsewidth modulation with a trajectory tracking scheme for high dynamic performance", *IEEE Applied Power Electronics. Conf. Rec*., pp. 734-741, Boston, Massachusetts, 1992.

[Hol 93a] J. Holtz, W. Lotzkat, and A. Khambadkone, "On Continuous Control of PWM Inverters in the Overmodulation Range Including the Six-Step Mode", *IEEE Trans. on Power Electronics*, Vol. 8, Nº 4, pp. 546-553, October 1993.

[Hol 93b] J. Holtz and B. Beyer, "Optimal Synchronous Pulsewidth Modulation with a Traectory-Tracking Scheme for High-Dynamic Perfomance", *IEEE Trans. on Industry Applications*, Vol. 29, pp. 1098-1105, Nov/Dec 1993.

[Hol 94a] J. Holtz and B. Beyer, "Fast current trajectory tracking control based on synchronous optimal pulsewidth modulation", *IEEE Industrial Application Soc. Conf. Records*, pp. 734-741, Denver, Colorado, 1994.

[Hol 94b] J. Holtz and B. Beyer, "The trajectory tracking approach – a new method for minimum distortion PWM in dynamic high-power drives", *IEEE Trans. on Industry Applications*, pp. 1048-1057, July/August, 1994.

[Hol 94c] J. Holtz, "Pulse Width Modulation for Electronic Power Conversion", *IEEE Proceedings*, Vol. 82 N° 8, Aug. 1994, pp. 1194-1214.

[Hol 97] J. Holtz, "Pulse Width Modulation for Electronic Power Conversion". En: *Power Electronics and Variable Frequency Drives*, (B. K. Bose) pp. 154-174. IEEE PRESS, 1997.

[Hou 84] J. A. Houldsworth and D. A. Grant. "The use of harmonic distortion to increase the output voltage of a three-phase PWM inverter". *IEEE Trans. on Industry Applications*, vol. IA-16, pp. 1224-1228, September/October 1984.

[Hub 95] L. Huber and D. Borojevic, "Space Vector modulates Three-Phase to Three-Phase Matrix Converter with input power Factor Correction", *IEEE Trans. on Industry Applications*, Vol. 31, pp. 1234-1246, Nov/Dec 1995.

[Jeo 91] Jeong Seung-Gi and Park Min-Ho, "The Analysis and Compensation of Dead-Time Effects in PWM Inverters", *IEEE Trans. on Industry Electronics*, Vol. 38, Nº 2, pp. 108-114, April 1991.

[Kaz 91] M. Kazmierkowski and W. Sulkowski, "A Novel Vector Control Scheme for Transistor PWM Inverter-Fed Induction Motor Drive", *IEEE Trans. on Industrial Electronics*, Vol. 38, N°1, pp. 41-47, Feb. 1991.

[Kog 89] T. Koga, H. Hayashi, E. Kijima and Y. Ohmori, "Instantaneous Spatial Vector Controlled Motor Drive System Using DSP's and SI-Thyristors", *European Power Electronics Conf. (EPE'89)*, Aachen, Germany, pp. 539-544, August 1989.

[Leg 97] D. Leggate and R. J. Kerkman, "Pulse-Based Dead-Time Compesator for PWM Voltage Inverters", *IEEE Trans. on Industrial Electronics*, Vol. 44, Nº 2, pp.191-197, April 1997.

[Maz 81] M. Mazzucchelli, L. Puglisi and G. Sciutto, "PWM systems in Power Converters: An Extension of the "Subharmonic" Method", *IEEE Transactions on Industrial Electronics and Control Instrumentation*, vol. IE-28, pp. 315-322, November 1981.

[Moh 89] N. Mohan, T. Undeland and W. Robbinson, *Power Electronics: Converters, Applications, and Design,* 2nd Ed., New York: Wiley, 1995.

[Mon 98] E. Monmasson, K. Tazi and J.P. Louis, "Description of an Entirely Re-Programmable Hardware System Dedicated to the Control of a Three Phase PWM Inverter", *Power Electronics and Motion Control Conf. (PEMC '98),* Praga, República Checa, pp. 2.179-2.184, Agosto 1998.

[Mur 87a] Y. Murai, T. Watanabe and H. Iwasaki, "Waveform Distortion and Correction Circuit for PWM Inverters with Switching Lag-Times", *IEEE Trans. on Industry Applications*, Vol. 23, Nº 5, pp. 881-886, September/October 1987.

[Mur 87b] Y. Murai, K. Ohashi and I. Hosono, "New PWM Method for Fully Digitized Inverters", *IEEE Trans. on Industry Applications*, Vol. 23, Nº 5, pp. 887-893, Sep/Oct 1987.

[Mur 92] Y. Murai, Y. Gohshi and I. Hosono, "High-Frequency Split-Zerovector PWM with Harmonic Reduction for Induction Motor Drive", *IEEE Trans on Industry Applications*, Vol. 28, Nº 1, pp. 105-112, Jan/Feb 1992.

[Oga 89] S. Ogasawara, H. Akagi and A. Nabae, "A Novel PWM Scheme of Voltage Source Inverters Based on Space Vector Theory", *European Power Electronics Conf. (EPE'89)*, Aachen, Germany, pp. 1197-1202, August 1989.

[Pat 73] H. S. Patel and R. G. Hoft, "Generalized techniques of harmonic elimination and voltage control in thyristor inverters: Part I-harmonic elimination", *IEEE Trans. on Industry Applications*, vol. IA-9, pp. 310-317, May/June 1973.

[Pat 74] H. S. Patel and R. G. Hoft, "Generalized techniques of harmonic elimination and voltage control techniques", *IEEE Trans. on Industry Applications*, vol. IA-10, pp. 666-673, September/October 1974.

[Pfa 82] G. Pfaff, A. Weschta, and A. Wick. "Design and experimental results of a brushless AC servo-drive". *IEEE Industrial Application Soc. Conf. Records*, pp. 692-697, San Fransisco, California, 1982.

[Sal 00] Z. Salcic and A. Smailagic, *Digital System Design and Prototyping Using Field Programmable Logic and Hardware Description Languages - Second Edition*, Kluwer Academic Publishers, 2000.

[Sch 64] A. Schönung and H. Stemmler. "Static frequency changers with subharmonic control in conjunction with reversable variable speed AC drives". *Brown Boveri Review*, pp. 555-577, September 1964.

[Sch 98] W. Schumacher, "Field Oriented Control With ASIC's, High Performance Drives For Mechatronics", *Power Electronics and Motion Control Conf. (PEMC '98)*, Praga, República Checa, pp K5.1-K5.8., Agosto 1998.

[Ska 96] K. Skahill, *VHDL for Programmable Logic*, Addison-Wesley, 1996.

[Ton 98a] M. A. Tonelli, "Modulación de Ancho de Pulso (PWM) con Control Vectorial", *Trabajo final de grado,* Depto. Electrotecnia, Fac. Ingeniería, UNLP, Agosto 1998.

[Ton 98b] M. Tonelli y M. I. Valla, "Modulador vectorial para inversores de potencia controlado desde una PC", *Anales del XVI Congreso Argentino de Control Automático, (AADECA'98)*, 17 al 21 de agosto de 1998, pp 188-193.

[Ton 99a] M. Tonelli, M. I. Valla y P. E. Battaiotto, "Space Vector Modulator With a Low Cost FPGA", *The 5th Brazilian Power Electronics Conference (COBEP '99)*, Foz do Iguazu, Brasil, Septiembre, 1999, pp 209-214.

[Ton 99b] M. Tonelli, "Space Vector Modulator for Power Inverters", *IEEE, Second Place in the 1999 Myron Zucker Industry Applications Student Design Award*, Phoenix, Arizona, USA, October 1999.

[Ton 00] M. Tonelli y M. I. Valla, "Análisis Crítico de la Compensación de Tiempo Muerto en Inversores de Potencia", *Anales del XVII Congreso Argentino de Control Automático, (AADECA'00)*, Septiembre de 2000, pp. 63-68.

[Ton 01] M. Tonelli, M. I. Valla and P. E. Battaiotto, "FPGA Implementation of an Universal Space Vector Modulator", *The 27th Annual Conference of the IEEE Industrial Electronics Society (IECON '01)*, Denver, Colorado, USA, November 2001, pp 1172-1177.

[Tur 63] F. G. Turnbull, "Selected harmonic reduction in static DC-AC", *IEEE Trans. Commun. Electron*, pp. 374-378, July 1963.

[Tzo 97] Tzou Ying-Yu and Hsu Hau-Jean, "FPGA Realization of Space-Vector PWM Control IC for Three-Phase PWM Inverters", *IEEE Trans. on Power Electronics*, Vol. 12, Nº 12, pp. 953-963, Nov 1997.

[Van 86] H. Van Der Broeck, H. Skudelny, and G. Stanke. "Analysis and realization of a pulse width modulator based on voltage space vectors". *IEEE Industrial Application Soc Conf. Records*, pp. 244-251, Denver, USA, 1986.

[Zac 85a] F. C. Zach and H. Ertl, "Efficiency optimal control for AC drives with PWM inverters", *IEEE Trans. on Industry Applications*, Vol. 21, pp. 987-1000, July/August 1985.

[Zac 85b] F. C. Zach, R. Martinez, S. Keplinger, and A. Seiser, "Dynamically optimal switching patterns for PWM inverters drives for minimization of the torque and speed ripples", *IEEE Trans. on Industry Applications*, Vol. 21, pp. 975-986, July/August 1985.

DIAGRAMAS ESQUEMÁTICOS

En las siguientes Tablas se muestran las conexiones entre la placa de Altera (UP1), el inversor trifásico, y la placa de medición de las tensiones de columna del Inversor.

Los circuitos esquemáticos del modulador vectorial universal se presentan a continuación de las Tablas de conexionado.

Revision: 1.0 13-Oct-2002 c:\max2work\mauricio\tesis\svm_10k.rpt

**** SALIDAS ****

Name	FLEX Pin	Primitive	UP1 Conector-A	UP1 Conector-B	UP1 Conector-C	Flex Switch, Push buttom y Digitos	Conector J1 (20 pins)	Conector J2 (6 pins)
Borrou_R	61	OUTPUT	25	---	---	---	---	---
Borrou_S	63	OUTPUT	27	---	---	---	---	---
Borrou_T	65	OUTPUT	29	---	---	---	---	---
Carri_R	70	OUTPUT	33	---	---	---	---	---
Carri_S	67	OUTPUT	31	---	---	---	---	---
Carri_T	72	OUTPUT	35	---	---	---	---	---
D0	162	TRI	---	55	---	---	---	---
D1a	6	OUTPUT	---	---	---	Digit 1 - Segment a	---	---
D1b	7	OUTPUT	---	---	---	Digit 1 - Segment b	---	---
D1c	8	OUTPUT	---	---	---	Digit 1 - Segment c	---	---
D1d	9	OUTPUT	---	---	---	Digit 1 - Segment d	---	---
D1e	11	OUTPUT	---	---	---	Digit 1 - Segment e	---	---
D1f	12	OUTPUT	---	---	---	Digit 1 - Segment f	---	---
D1g	13	OUTPUT	---	---	---	Digit 1 - Segment g	---	---
D1p	14	OUTPUT	---	---	---	Digit 1 - Segment p	---	---
D2a	17	OUTPUT	---	---	---	Digit 2 - Segment a	---	---
D2b	18	OUTPUT	---	---	---	Digit 2 - Segment b	---	---
D2c	19	OUTPUT	---	---	---	Digit 2 - Segment c	---	---
D2d	20	OUTPUT	---	---	---	Digit 2 - Segment d	---	---
D2e	21	OUTPUT	---	---	---	Digit 2 - Segment e	---	---
D2f	23	OUTPUT	---	---	---	Digit 2 - Segment f	---	---
D2g	24	OUTPUT	---	---	---	Digit 2 - Segment g	---	---
D2p	25	OUTPUT	---	---	---	Digit 2 - Segment p	---	---
Enable_Load	94	OUTPUT	50	---	---	---	---	---
LoadN	163	OUTPUT	---	56	---	---	---	---
MSB	101	OUTPUT	56	---	---	---	---	---
PC_Observa	84	OUTPUT	46	---	---	---	---	---
Seguro	175	OUTPUT	---	---	15	---	---	---
Start	80	OUTPUT	42	---	---	---	---	---
StopN	82	OUTPUT	44	---	---	---	---	---
Sw_testR	48	OUTPUT	17	---	---	---	---	---
Sw_testS	51	OUTPUT	20	---	---	---	---	---
Sw_testT	45	OUTPUT	15	---	---	---	---	---
Sw1	134	OUTPUT	---	33	---	---	14	---
Sw1_Com_Td	46	OUTPUT	16	---	---	---	---	---
Sw2	136	OUTPUT	---	34	---	---	13	---
Sw3	132	OUTPUT	---	31	---	---	12	---
Sw3_Com_Td	50	OUTPUT	19	---	---	---	---	---
Sw4	133	OUTPUT	---	32	---	---	11	---
Sw5	129	OUTPUT	---	29	---	---	10	---
Sw5_Com_Td	49	OUTPUT	18	---	---	---	---	---
Sw6	131	OUTPUT	---	30	---	---	9	---
ta	99	OUTPUT	54	---	---	---	---	---
tb	97	OUTPUT	52	---	---	---	---	---
twN	87	OUTPUT	48	---	---	---	---	---

**** ENTRADAS ****

Name	FLEX Pin	Primitive	UP1 Conector-A	UP1 Conector-B	UP1 Conector-C	Flex Switch, Push buttom y Digitos	Conector J1 (20 pins)	Conector J2 (6 pins)
Ap1	36	INPUT	---	---	---	Flex SW 5	---	---
Ap2	35	INPUT	---	---	---	Flex SW 6	---	---
Ap3	34	INPUT	---	---	---	Flex SW 7	---	---
Ap4	33	INPUT	---	---	---	Flex SW 8	---	---
A0	200	INPUT	---	---	33	---	---	---
a1	201	INPUT	---	---	34	---	---	---
a2	198	INPUT	---	---	31	---	---	---
a3	199	INPUT	---	---	32	---	---	---
A4	195	INPUT	---	---	29	---	---	---
A5	196	INPUT	---	---	30	---	---	---
A6	193	INPUT	---	---	27	---	---	---
A7	194	INPUT	---	---	28	---	---	---
A8	191	INPUT	---	---	25	---	---	---
A9	192	INPUT	---	---	26	---	---	---
clk	211	INPUT	---	---	14	---	---	---
Dina0	139	INPUT	---	37	---	---	---	---
Dina1	142	INPUT	---	39	---	---	---	---
Dina2	144	INPUT	---	41	---	---	---	---
Dina3	147	INPUT	---	43	---	---	---	---
Dina4	148	INPUT	---	44	---	---	---	---
Dina5	146	INPUT	---	42	---	---	---	---
Dina6	143	INPUT	---	40	---	---	---	---
Dina7	141	INPUT	---	38	---	---	---	---
Dinb0	152	INPUT	---	47	---	---	---	---
Dinb1	154	INPUT	---	49	---	---	---	---
Dinb2	157	INPUT	---	51	---	---	---	---
Dinb3	159	INPUT	---	53	---	---	---	---
Dinb4	161	INPUT	---	54	---	---	---	---
Dinb5	158	INPUT	---	52	---	---	---	---
Dinb6	156	INPUT	---	50	---	---	---	---
Dinb7	153	INPUT	---	48	---	---	---	---
Enable_Comp_Td	41	INPUT	---	---	---	Flex SW 1	---	---
Enable_Td	39	INPUT	---	---	---	Flex SW 3	---	---
IORN	183	INPUT	---	---	18	---	---	---
IOWN	181	INPUT	---	---	16	---	---	---
LD13	202	INPUT	---	---	35	---	---	---
LD14	204	INPUT	---	---	37	---	---	---
LD15	207	INPUT	---	---	39	---	---	---
ResetDrv	209	INPUT	---	---	12	---	---	---
SBHEN	217	INPUT	---	---	43	---	---	---
Sw_StopN	29	INPUT	---	---	---	Flex PB 2	---	---
VR0TTL	111	INPUT	---	17	---	---	---	6
VS0TTL	114	INPUT	---	19	---	---	---	4
VT0TTL	116	INPUT	---	21	---	---	---	2
VCC	Vcc	INPUT	---	5	---	---	---	5
GND	Gnd	INPUT	---	6	---	---	---	1
VCC	Vcc	INPUT	---	59	---	---	20	---
GND	Gnd	INPUT	---	60	---	---	18	---

LISTADO DEL PROGRAMA DEL MODULADOR UNIVERSAL EN AHDL

En este apéndice se presenta el listado del código en AHDL utilizado para la implementación del modulador vectorial universal en la placa de Altera (UP1). Este código pertenece al archivo SVM_10K.tdf.

MAX+plus II 7.21 File: SVM_10K.TDF

```
--
--  TESIS DE MAESTRIA - LEICI - UNLP - ARGENTINA - 10/1999:01/2002
--
-- Prototipo: Modulador Vectorial Universal
--
-- Diseño: Mauricio A. Tonelli & Maria Ines Valla
--
-- Fecha ultima revisión: 21-01-2003
-- Fecha ultima revisión: 18-01-2003
-- Fecha ultima revisión: 16-01-2003
-- Fecha ultima revisión: 12-01-2003
-- Fecha ultima revisión: 15-04-2002
-- Dispositivo a utilizar en UP1: EPF10K20RC240-4
--
% Notas:
1) Modulador completo, CON implementacion de los tiempos muertos
   y la interfaz de lectura escritura con el BUS ISA.
      * Direccion default de escritura: 31E
      * Direccion default de lectura:   30E
        asignaciones:
                        Ap1    FLEX_SWITCH-5 (pin 36) se compara con A1
                        Ap2    FLEX_SWITCH-6 (pin 35) se compara con A2
                        Ap3    FLEX_SWITCH-7 (pin 34) se compara con A3
      (no usado)   Ap4    FLEX_SWITCH-8 (pin 33) se compara con A4
2) La Secuencia a utilizar es:  Va Vb V0 Vb Va V0
                                Sector I  : V1 V2 V0 V2 V1 V7
                                Sector II : V2 V3 V7 V3 V2 V0
                                Sector III: V3 V4 V0 V4 V3 V7
                                Sector IV : V4 V5 V7 V5 V4 V0
                                Sector V  : V5 V6 V0 V6 V5 V7
                                Sector VI : V6 V1 V7 V1 V6 V0
Aclaracion: En los cambios de sector ocurriran dos conmutaciones.
3) Este modulador se detiene por SOFTWARE, enviando LD13=LD14=LD15=1 ó HARDWARE por
   medio del switch Sw_StopN que esta asignado al FLEX_PB2 (pin 29).
4) (23-05-2001)
    Maquina secuencial con 6 estados, cambio en los nombres de los estados.
    Cta y Ctb siempre cuentan en sentido descendente.
5) (24-05-2001)
      * Senal Enable_Load implementada con una maquina secuencial de 3 estados
6) (01-06-2001)
      * Implementacion de compensación del tiempo muerto.
      * Habilitacion/Desabilitacion Implementacion de Tiempo de seguridad Td,
        asignado al FLEX_SWITCH-3 (pin 39).
      * Habilitacion/Desabilitacion Implementacion de Compensacion del Tiempo muerto.
        asignado al FLEX_SWITCH-1 (pin 41).
7) (27-11-2001)
      * Agregado de optoacopladores en la etapa de salida.
        (OJO que los optoacopladores son inversores) por eso se negó la salida.
8) (15-04-2002)
      * Agregado de la etapa de detección de la repetición de la muestra.
        con esta etapa se evita que el modulador implemente indefinidamente la misma muestra,
         protegiendo la carga. Esto podia pasar si por alguna razon el software perdia la
         comunicacion con el modulador, por Ej: con un Ctrl-Break.
9) (05-05-2002)
      * Agregado de los Flip-Flop para hacer sincronico el clear del shiftregister de la
        etapa de implementacion del tiempo de seguridad Td.
```

```
10) (12-01-2003)
      * Agregado del comparador "SumaAB_MAXIMA", para detectar cuando la suma de A+B es 255.
        En la version anterior se utilizaba la salida .cout pero esta nunca se ponia en 1
        debido a que la suma da como maxino ff o sea 255.
11) (16-01-2003)
      * Modificaciones en la maq. sec. en los estados s0, s1 y s2. Agregado de señales.
        NOTA: la modificacion se anula por problemas de timing.
12) (18-01-2003)
      * Modificaciones en la MS, modificacion en cambio de estados: agregado salto de S5 a S4
        y eliminacion salto de S5 a S1.
13) (21-01-2003)
      * Modificaciones en la MS, incorporacion del nuevo estado S6.
       * La MS salta:
      *            de S1 (ta) a S6 (tbinew) y de S6 a S2 (tb).
      * se elimina salto de S1 a S5 (tbi) y de S5 a S4.
      * se incorpora salto de S5 a S1
%
INCLUDE "lpm_compare.inc";
INCLUDE "lpm_add_sub.inc";
INCLUDE "lpm_counter.inc";
INCLUDE "lpm_shiftreg.inc";
INCLUDE "lpm_mux.inc";
INCLUDE "lpm_ltch.inc";   -- FUNCTION lpm_ltch (clk, data[7..0])  RETURNS (lq[7..0])
INCLUDE "74193.inc";      -- Compensacion de Td - M.T. 01-06-01

SUBDESIGN svm_10K
(
      A[9..0], Ap[4..1]   : INPUT;
    SBHEN, IOWN, IORN     : INPUT;
    ResetDrv              : INPUT;
      LD13, LD14, LD15    : INPUT;
    clk                   : INPUT;
    Dina[7..0]           : INPUT;
    Dinb[7..0]           : INPUT;
    Sw_StopN             : INPUT; --Pulsador que en cero, detiene al modulador.
    Enable_Td            : INPUT; --Jumper habilita Implementacion de Td = 2useg - MT 01-06-01
    Enable_Comp_Td       : INPUT; --Jumper habilita Compensacion de Td - MT 01-06-01
    VR0TTL               : INPUT; -- Usada para compensacion de Td
    VS0TTL               : INPUT; -- Usada para compensacion de Td
    VT0TTL               : INPUT; -- Usada para compensacion de Td

    LoadN           : OUTPUT;
    D0              : OUTPUT;
    MSB             : OUTPUT; -- Para verificar funcionamiento.
    ta, tb          : OUTPUT; -- Para verificar funcionamiento.
    Enable_Load     : OUTPUT; -- Para verificar funcionamiento.
    PC_Observa      : OUTPUT; -- Para verificar funcionamiento.
    twN             : OUTPUT; -- Para verificar funcionamiento.
--    PLa, PLb      : OUTPUT; -- Para verificar funcionamiento.
--    LatchOut      : OUTPUT; -- Para verificar funcionamiento en la medida del retardo.
    Sw[6..1]        : OUTPUT;
    Sw_testR        : OUTPUT; -- Para verificar funcionamiento.
    Sw_testS        : OUTPUT; -- Para verificar funcionamiento.
    Sw_testT        : OUTPUT; -- Para verificar funcionamiento.
      Sw1_Com_Td    : OUTPUT; -- Para verificar funcionamiento.
      Sw3_Com_Td    : OUTPUT; -- Para verificar funcionamiento.
      Sw5_Com_Td    : OUTPUT; -- Para verificar funcionamiento.
      StopN         : OUTPUT; -- Para verificar funcionamiento.
      Start         : OUTPUT; -- Para verificar funcionamiento.
      Seguro        : OUTPUT; -- Para verificar funcionamiento.
      Carri_R       : OUTPUT; -- Para verificar funcionamiento.
      Borrou_R      : OUTPUT; -- Para verificar funcionamiento.
    Carri_S         : OUTPUT; -- Para verificar funcionamiento.
    Borrou_S        : OUTPUT; -- Para verificar funcionamiento.
    Carri_T         : OUTPUT; -- Para verificar funcionamiento.
    Borrou_T        : OUTPUT; -- Para verificar funcionamiento.

-- Salidas adicionales para los "chiches"
    D1a, D1b, D1c, D1d    : OUTPUT;
```

```
    D1e, D1f, D1g, D1p    : OUTPUT;
    D2a, D2b, D2c, D2d    : OUTPUT;
    D2e, D2f, D2g, D2p    : OUTPUT;
)

VARIABLE
-- Variables de la Maquina secuencial y Contadores Cta, Ctb
    ss  : MACHINE
--MT 21-01-03                 WITH STATES (s0, s1, s2, s3, s4, s5);
                WITH STATES (s0, s1, s2, s3, s4, s5, s6);
    Cta          : lpm_counter WITH (LPM_WIDTH = 8, LPM_DIRECTION = "DOWN");
    Ctb          : lpm_counter WITH (LPM_WIDTH = 8, LPM_DIRECTION = "DOWN");
    Cta_Q[7..0]  : NODE;
    Ctb_Q[7..0]  : NODE;
    FCa, FCb     : Node; --FC (Final Count) es cuando el contador llego a cero.
    PLa, PLb     : Node; --PL sirve para cargar el contador, es una senal activa en alto.
      SumaAB    : lpm_add_sub WITH (LPM_WIDTH = 8, LPM_DIRECTION = "ADD");
      Cero_A    : lpm_compare WITH (LPM_WIDTH = 8, ONE_INPUT_IS_CONSTANT = "YES");
      Cero_B    : lpm_compare WITH (LPM_WIDTH = 8, ONE_INPUT_IS_CONSTANT = "YES");
      Max_A     : lpm_compare WITH (LPM_WIDTH = 8, ONE_INPUT_IS_CONSTANT = "YES");
      Max_B     : lpm_compare WITH (LPM_WIDTH = 8, ONE_INPUT_IS_CONSTANT = "YES");
SumaAB_MAXIMA   : lpm_compare WITH (LPM_WIDTH = 8, ONE_INPUT_IS_CONSTANT = "YES");

 -- Variables decodificador direccion, contador C y Encendido/Apagado.
          C     : lpm_counter WITH (LPM_WIDTH = 9);
      sal_and   : NODE;
   sal_and_r    : NODE;
        comp    : NODE;
      FF_pru    : DFF;
    FF_Start    : JKFF;     -- CLRN J  K     |     Q
--                           --  H   L  L     |     Qo*
--                           --  H   H  L     |     H   (usado asi)
--                           --  H   L  H     |     L
--                           --  H   H  H     |     Toggle
--                           -- * Qo = level of Q before Clock pulse (Low to High)
    Ena       : MACHINE                    -- Maquina Secuencial para generar la señal de
                WITH STATES (e0, e1, e2);  -- habilitación Enable_Load.
-- Variables para evitar repetir mas de tres veces una muestra, cuando la PC no carga una
-- muestra nueva. COn esto se evita aplicar una señal continua a la carga.
-- MT 15-04-02
--    Seguro   : NODE;
      FF_segur_a   : DFF;
      FF_segur_b   : DFF;
      FF_segur_c   : DFF;

-- Variables de la ROM de salida (LUT) y de los Vectores Espaciales.
    Rom_II          : lpm_mux WITH (LPM_WIDTH=3, LPM_SIZE=64, LPM_WIDTHS=6);
    Rsw1, Rsw3, Rsw5 : Node;
      L_out[0..2]   : DFF;
      retardo1      : Node;
      retardo2      : Node;
      retardo3      : Node;
      LatchOut      : Node;
      Null          : Node;

-- Variables Lacth II.
      L_IIa, L_IIb    : lpm_ltch;
      L_II3[15..13]   : DFF;
      L_IIDina[7..0]  : NODE;
      L_IIDinb[7..0]  : NODE;
      L_IILD[15..13]  : NODE;

--*** Variables para Td = 2 useg:
--* Usado en SEIS lineas.
--    SftReg_SW1    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
--    SftReg_SW2    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
--    SftReg_SW3    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
--    SftReg_SW4    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
--    SftReg_SW5    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
```

```
--    SftReg_SW6    : lpm_shiftreg WITH (LPM_WIDTH = 18, LPM_DIRECTION = "Left");
--*** Variables para Td = 3 useg: (Para observar compensacion de tiempo muerto MT 28-12-01).
--* Usado en SEIS lineas.
    SftReg_SW1      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");
    SftReg_SW2      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");
    SftReg_SW3      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");
    SftReg_SW4      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");
    SftReg_SW5      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");
    SftReg_SW6      : lpm_shiftreg WITH (LPM_WIDTH = 27, LPM_DIRECTION = "Left");

--*Flip-Flop adicionales para clear sincronico del shiftregister de implementacion de Td.
      FF_TdSw1      : DFF;
      FF_TdSw2      : DFF;
      FF_TdSw3      : DFF;
      FF_TdSw4      : DFF;
      FF_TdSw5      : DFF;
      FF_TdSw6      : DFF;

      Sw1_Td        : Node;
      Sw2_Td        : Node;
      Sw3_Td        : Node;
      Sw4_Td        : Node;
      Sw5_Td        : Node;
      Sw6_Td        : Node;
-- Contadores para implementacion de compensacion de Td - M.T. 01-06-01
      C_td_R1       : 74193;  -- Cuenta fase R
      C_td_R2       : 74193;
        C_td_S1     : 74193;  -- Cuenta fase S
        C_td_S2     : 74193;
          C_td_T1   : 74193;  -- Cuenta fase T
          C_td_T2   : 74193;
      ff_R          : DFF;
       ff_S         : DFF;
        ff_T        : DFF;
--    Sw1_Com_Td    : Node;
      Sw2_Com_Td    : Node;
--    Sw3_Com_Td    : Node;
      Sw4_Com_Td    : Node;
--    Sw5_Com_Td    : Node;
      Sw6_Com_Td    : Node;
      up_R          : Node;
       up_S         : Node;
        up_T        : Node;
      down_R        : Node;
       down_S       : Node;
        down_T      : Node;
--    Carri_R       : Node;
--     Carri_S      : Node;
--      Carri_T     : Node;
--    Borrou_R      : Node;
--     Borrou_S     : Node;
--      Borrou_T    : Node;

-- Multiplexer para Habilitar o desabilitar compensacion de Td - M.T. 01-06-01
      Mux_Comp_Td   : lpm_mux WITH (LPM_WIDTH=6, LPM_SIZE=2, LPM_WIDTHS=1);

-- Multiplexer para Habilitar o desabilitar implementacion de Td - M.T. 01-06-01
       Mux_Td       : lpm_mux WITH (LPM_WIDTH=6, LPM_SIZE=2, LPM_WIDTHS=1);


BEGIN
-- Bloque decodificador de direccion.
--    sal_and = !A0 & !A5 & !A6 & !A7 & A8 & A9;
      sal_and = !A0 & !A5 & !A6 & !A7 & A8 & A9 & A4;
      sal_and_r = !A0 & !A5 & !A6 & !A7 & A8 & A9 & !A4;
--       comp = (A[4..1] == Ap[4..1]);
       comp = (A[3..1] == Ap[3..1]); -- To allows one switch for Read and Write MT 26-11-01
      LoadN = !(!IOWN & (!ResetDrv & !SBHEN & sal_and & comp));
--    PC_Observa = !IORN & !ResetDrv & sal_and & comp;
```

```
      PC_Observa = !SBHEN & !ResetDrv & sal_and_r & comp;
% Switch selector de direccion del Modulador en el BUS ISA.
      Address Write|  A4  Ap3 Ap2 Ap1        Address Read |  A4  Ap3 Ap2 Ap1
     (Default) 31E |  1    1   1   1       (Default) 30E  |  0   1    1   1
               31C |  1    1   1   0                 30C  |  0   1    1   0
               31A |  1    1   0   1                 30A  |  0   1    0   1
               318 |  1    1   0   0                 308  |  0   1    0   0
               316 |  1    0   1   1                 306  |  0   0    1   1
               314 |  1    0   1   0                 304  |  0   0    1   0
               312 |  1    0   1   1                 302  |  0   0    1   1
               310 |  1    0   1   0                 300  |  0   0    1   0
%
-- Bloque generador de la senal Start y control del contador C.
         FF_Start.J = VCC;
         FF_Start.K = GND;
       FF_Start.CLK = LoadN;  -- LoadN porque los FF son activos con el flanco de subida.
      FF_Start.CLRN = StopN;  -- Pone al FF en cero.
--      FF_pru.d = !ResetDrv AND !(LD15 AND LD14 AND LD13) AND Sw_StopN;
      FF_pru.d   = !ResetDrv AND !(LD15 AND LD14 AND LD13);
      FF_pru.clk = clk;
           StopN = !Seguro AND FF_pru.q;
           Start = FF_Start.Q;

-- Implementacion de la señal de seguridad "Seguro", para no repetir muestras indefinidamente.
--    FF_segur_a.clk = twN;
--    FF_segur_b.clk = twN;
--    FF_segur_c.clk = twN;
      FF_segur_a.clk = Enable_Load;
      FF_segur_b.clk = Enable_Load;
      FF_segur_c.clk = Enable_Load;
      FF_segur_a.clrn = LoadN;
      FF_segur_b.clrn = LoadN;
      FF_segur_c.clrn = LoadN;

      FF_segur_a.d = VCC;
      FF_segur_b.d = FF_segur_a.q;
      FF_segur_c.d = FF_segur_b.q;

      Seguro = FF_segur_c.q;  -- "1" SVM repitio muestra mas de dos veces   (func. abortado)
                              -- "0" SVM NO repitio muestra mas de dos veces (func. normal)

-- Implementacion del contador Binario C.
    C.clock = clk;
    C.aset = !Start;  -- C.aset=1 Detiene funcionamiento Contador C (Donde MSB queda en 1).
MSB = C.q8;           -- Cuenta 512 periodos de reloj.
twN = !(C.q8 & C.q7 & C.q6 & C.q5 & C.q4 & C.q3 AND !C.q2);  -- tw = 4*Tclk (aprox. 480 nseg)

-- Implementacion de la senal de habilitacion de datos.
-- Usando una maquina secuencial de 3 estados "Ena"
      Ena.clk = clk;
      Ena.reset = Seguro;
      CASE Ena IS
            WHEN e0 => Enable_Load = VCC; --Habilita la carga de un nuevo dato.
                  IF (!LoadN) THEN
                        Ena = e1;
                  ELSIF (!twN) THEN
                        Ena = e2;
                  ELSE
                        Ena = e0;
                  END IF;
            WHEN e1 => Enable_Load = GND; -- Desabilita la carga de un nuevo dato.
                  IF (!twN) THEN
                        Ena = e2;
                  ELSE
                        Ena = e1;
                  END IF;
            WHEN e2 => Enable_Load = GND; -- Desabilita la carga de un nuevo dato.
                  IF (!MSB) THEN
                        Ena = e0;
```

```
                    ELSE
                          Ena = e2;
                    END IF;
      END CASE;
 D0 = TRI(Enable_Load, PC_Observa);  -- Señal que va al pin D0 del Bus de Datos

-- Implementacion del Lacth II.
               L_IIa.clk = !MSB;
               L_IIb.clk = !MSB;
             L_II3[].clk = !MSB;
      L_IIa.data[7..0] = Dina[7..0];
      L_IIb.data[7..0] = Dinb[7..0];
       L_II3[15..13].d = (LD15, LD14, LD13);
L_IIDina[7..0] = L_IIa.lq[7..0];
L_IIDinb[7..0] = L_IIb.lq[7..0];
L_IILD[15..13] = L_II3[15..13].q;

-- ***************************************************
--     INICIO DE LA     Maquina secuencial

-- Variables auxiliares.
      SumaAB.dataa[] = L_IIDina[];
      SumaAB.datab[] = L_IIDinb[];
      --Salida:  SumaAB.cout  que se pone en 1 cuando: A + B = 256.
      Cero_A.dataa[] = L_IIDina[];
      Cero_A.datab[] = B"00000000";
      --Salida:  Cero_A.aeb que se pone en 1 cuando: A == 0.
      Cero_B.dataa[] = L_IIDinb[];
      Cero_B.datab[] = B"00000000";
      --Salida:  Cero_B.aeb que se pone en 1 cuando: B == 0.
      Max_A.dataa[] = L_IIDina[];
      Max_A.datab[] = B"11111111";
      --Salida:  Max_A.aeb que se pone en 1 cuando: A == 255.
      Max_B.dataa[] = L_IIDinb[];
      Max_B.datab[] = B"11111111";
      --Salida:  Max_B.aeb que se pone en 1 cuando: B == 255.

      SumaAB_MAXIMA.dataa[] = SumaAB.result[];
      SumaAB_MAXIMA.datab[] = B"11111111";
      --Salida:  SumaAB_MAXIMA.aeb que se pone en 1 cuando: SumaAB == 255.

       ss.clk   = !clk;
       ss.reset = !Start;  -- ss.reset=1 => Reset Maq. Sec. => 1er estado definido
                           -- en WITH STATE, o sea s0.
CASE ss IS

      WHEN s0 =>      -- t0-. Espero a que termine el Subciclo implementando t0-.
                  ta = GND; -- to Testing
                  tb = GND; -- to Testing
             PLa = VCC;          -- Cargo dato ta
             PLb = VCC;          -- Cargo dato tb
--MT 16-01-03      IF (!MSB AND !Cero_A.aeb) THEN
--MT 18-01-03      IF (!MSB AND (!Cero_A.aeb OR Max_A.aeb)) THEN
             IF (!MSB AND !Cero_A.aeb) THEN
                  ss = s1;
--MT 18-01-03      ELSIF (!MSB AND Cero_A.aeb) THEN
             ELSIF (!MSB AND Cero_A.aeb AND !Cero_B.aeb) THEN

                  ss = s2;
             ELSE
                  ss = s0;
             END IF;

      WHEN s1 =>       -- ta. Implementa ta.
                   ta = VCC; -- to Testing
                   tb = GND; -- to Testing
             PLa = GND;    -- Permito que cuente el contador ta
             PLb = VCC;          -- Cargo dato tb
             IF (!MSB AND FCa AND !Cero_B.aeb AND !SumaAB_MAXIMA.aeb) THEN
```

```
                  ss = s2;
            ELSIF (MSB AND FCa AND !Max_A.aeb) THEN
                  ss = s0;
--MT 16-01-03  ELSIF (FCa AND Max_A.aeb) THEN
--MT 18-01-03  ELSIF (FCa AND !MSB AND SumaAB_MAXIMA.aeb AND Max_A.aeb AND Cero_B.aeb) THEN
            ELSIF (MSB AND SumaAB_MAXIMA.aeb AND Max_A.aeb) THEN
                  ss = s4;
            ELSIF (!MSB AND FCa AND Cero_B.aeb AND !Max_A.aeb) THEN
                  ss = s3;
--MT 16-01-03       ELSIF (FCa AND !MSB AND SumaAB_MAXIMA.aeb) THEN
            ELSIF (!MSB AND FCa AND SumaAB_MAXIMA.aeb AND !Max_A.aeb AND !Cero_B.aeb) THEN
--MT 21-01-03             ss = s5;
                  ss = s6;
            ELSE
                  ss = s1;
            END IF;

      WHEN s2 =>        -- tb. Implementa tb.
                 tb = VCC; -- to Testing
                   ta = GND; -- to Testing
            PLb = GND;      -- Permito que cuente el contador tb
            PLa = VCC;      -- Cargo dato ta
            IF (!MSB AND FCb AND !Max_B.aeb) THEN
                  ss = s3;
--MT 16-01-03       ELSIF (FCb AND Max_B.aeb) THEN
--MT 18-01-03       ELSIF (FCb AND Max_B.aeb AND Cero_A.aeb AND !MSB) THEN
            ELSIF (MSB AND SumaAB_MAXIMA.aeb AND Max_B.aeb  AND Cero_A.aeb) THEN
                  ss = s5;
            ELSIF (MSB AND FCb AND !SumaAB_MAXIMA.aeb AND !Cero_A.aeb) THEN
                  ss = s1;
--MT 18-01-03       ELSIF (MSB AND FCb AND Cero_A.aeb) THEN
            ELSIF (MSB AND FCb AND Cero_A.aeb AND !SumaAB_MAXIMA.aeb) THEN
                  ss = s0;
--MT 18-01-03       ELSIF (MSB AND FCb AND SumaAB_MAXIMA.aeb AND !Max_B.aeb) THEN
            ELSIF (MSB AND FCb AND SumaAB_MAXIMA.aeb AND !Max_B.aeb AND !Cero_A.aeb) THEN
                  ss = s4;
            ELSE
                  ss = s2;
            END IF;

      WHEN s3 =>      -- t0+. Espero a que termine el Subciclo implementando t0+.
                  ta = GND; -- to Testing
                  tb = GND; -- to Testing
            PLb = VCC;
            PLa = VCC;
            IF (MSB AND Cero_B.aeb) THEN
                  ss = s1;
            ELSIF (MSB AND !Cero_B.aeb) THEN
                  ss = s2;
            ELSE
                  ss = s3;
            END IF;

      WHEN s4 =>       -- ta(inactivo). Espero a que termine el Subciclo implementando ta.
                   ta = VCC; -- to Testing
                  tb = GND; -- to Testing
            PLb = VCC;
            PLa = VCC;
--MT 16-01-03       IF (!MSB AND Cero_A.aeb) THEN
--MT 18-01-03       IF (!MSB AND (Cero_A.aeb OR Max_B.aeb)) THEN
            IF (!MSB AND Cero_A.aeb AND !Cero_B.aeb) THEN
                  ss = s2;
            ELSIF (!MSB AND (!Cero_A.aeb OR Max_A.aeb)) THEN
                  ss = s1;
            ELSE
                  ss = s4;
            END IF;

      WHEN s5 =>       -- tb(inactivo). Espero a que termine el Subciclo implementando tb.
```

```
                    tb = VCC; -- to Testing
                    ta = GND; -- to Testing
              PLb = VCC;
              PLa = VCC;
--MT 16-01-03  IF ((!MSB AND Cero_A.aeb) OR (MSB AND SumaAB_MAXIMA.aeb)) THEN
--MT 18-01-03  IF ((!MSB AND Cero_A.aeb) OR (MSB AND SumaAB_MAXIMA.aeb AND !Max_B.aeb)) THEN
              IF (!MSB AND Cero_A.aeb AND Max_B.aeb) THEN
                    ss = s2;
--MT 16-01-03       ELSIF (!MSB AND !Cero_A.aeb AND !SumaAB_MAXIMA.aeb) THEN
--MT 18-01-03       ELSIF (!MSB AND !Cero_A.aeb AND !SumaAB_MAXIMA.aeb AND !Max_B.aeb) THEN
              ELSIF (!MSB AND !Cero_A.aeb AND !Max_B.aeb) THEN
--MT 18-01-03             ss = s1;
--MT 21-01-03             ss = s4;
                    ss = s1;
              ELSE
                    ss = s5;
              END IF;

--MT 21-01-03 Incorporacion del nuevo estado S6
       WHEN s6 =>    -- tb(inactivo nuevo). Espero a que termine el Subciclo implementando tb.
                    tb = VCC; -- to Testing
                    ta = GND; -- to Testing
              PLb = VCC;
              PLa = VCC;
              IF (MSB) THEN
                    ss = s2;
              ELSE
                    ss = s6;
              END IF;

   END CASE;

--     FIN DE LA Maq. Secuencial
--*******************************

-- Implementacion de los contadores Cta y Ctb.
            Cta.clock = clk;
            Cta.aload = PLa;
       Cta.data[7..0] = L_IIDina[7..0];
          Cta_Q[7..0] = Cta.q[7..0];
   FCa = !Cta_Q7 & !Cta_Q6 & !Cta_Q5 & !Cta_Q4 & !Cta_Q3 & !Cta_Q2 & !Cta_Q1 & !Cta_Q0;
            Ctb.clock = clk;
            Ctb.aload = PLb;
       Ctb.data[7..0] = L_IIDinb[7..0];
         Ctb_Q[7..0] = Ctb.q[7..0];
   FCb = !Ctb_Q7 & !Ctb_Q6 & !Ctb_Q5 & !Ctb_Q4 & !Ctb_Q3 & !Ctb_Q2 & !Ctb_Q1 & !Ctb_Q0;

-- Implementacion de la ROM de salida (contiene los estados de
-- los Vectores Espaciales).
--Orden de llaves:   S1   S3   S5    donde:   -------------------
--                    :    :    :            +|    S1    S3    S5
-- V0  1  1  1.                               Vcc    |-*   |-*   |-*
Rom_II.data[0][] = (VCC, VCC, VCC);   --      -|    S2    S4    S6
Rom_II.data[12][] = (VCC, VCC, VCC);  --       -------------------
Rom_II.data[16][] = (VCC, VCC, VCC);
Rom_II.data[28][] = (VCC, VCC, VCC);
Rom_II.data[32][] = (VCC, VCC, VCC);
Rom_II.data[44][] = (VCC, VCC, VCC);
-- V1  1  0  0.
Rom_II.data[1][] = (VCC, GND, GND);
Rom_II.data[5][] = (VCC, GND, GND);
Rom_II.data[42][] = (VCC, GND, GND);
Rom_II.data[46][] = (VCC, GND, GND);
-- V2  1  1  GND.
Rom_II.data[2][] = (VCC, VCC, GND);
Rom_II.data[6][] = (VCC, VCC, GND);
Rom_II.data[9][] = (VCC, VCC, GND);
Rom_II.data[13][] = (VCC, VCC, GND);
-- V3  GND  1  GND.
```

```
Rom_II.data[10][] = (GND, VCC, GND);
Rom_II.data[14][] = (GND, VCC, GND);
Rom_II.data[17][] = (GND, VCC, GND);
Rom_II.data[21][] = (GND, VCC, GND);
-- V4  GND  1  1.
Rom_II.data[18][] = (GND, VCC, VCC);
Rom_II.data[22][] = (GND, VCC, VCC);
Rom_II.data[25][] = (GND, VCC, VCC);
Rom_II.data[29][] = (GND, VCC, VCC);
-- V5  GND  GND  VCC.
Rom_II.data[26][] = (GND, GND, VCC);
Rom_II.data[30][] = (GND, GND, VCC);
Rom_II.data[33][] = (GND, GND, VCC);
Rom_II.data[37][] = (GND, GND, VCC);
-- V6  VCC  GND   VCC.
Rom_II.data[34][] = (VCC, GND, VCC);
Rom_II.data[38][] = (VCC, GND, VCC);
Rom_II.data[41][] = (VCC, GND, VCC);
Rom_II.data[45][] = (VCC, GND, VCC);
-- V7  gnd  gnd  gnd.
Rom_II.data[4][] = (GND, GND, GND);
Rom_II.data[8][] = (GND, GND, GND);
Rom_II.data[20][] = (GND, GND, GND);
Rom_II.data[24][] = (GND, GND, GND);
Rom_II.data[36][] = (GND, GND, GND);
Rom_II.data[40][] = (GND, GND, GND);

Rom_II.sel[] = (L_IILD15, L_IILD14, L_IILD13, Null, tb, ta); -- NEW Line, whit the change of
                                                             -- the address A2.
(Rsw1, Rsw3, Rsw5) = ROM_II.result[];

-- Organizacion de la ROM II:
--
-- Direccion: L_IILD15, L_IILD14, L_IILD13, Null, tb, ta  =  A5, A4, A3, A2, A1, A0
-- Salida:    (S1,  S3,  S5)
--
-- Rom_II.data[0][] = (vcc, vcc, vcc);  V0          |
-- Rom_II.data[1][] = (vcc, gnd, gnd);  V1          |
-- Rom_II.data[2][] = (vcc, vcc, gnd);  V2          |
-- don't care                                       |  Sector I
-- Rom_II.data[4][] = (gnd, gnd, gnd);  V7          |
-- Rom_II.data[5][] = (vcc, gnd, gnd);  V1          |
-- Rom_II.data[6][] = (vcc, vcc, gnd);  V2          |
-- don't care
-- Rom_II.data[8][] = (gnd, gnd, gnd);   V7         |
-- Rom_II.data[9][] = (vcc, vcc, gnd);   V2         |
-- Rom_II.data[10][] = (gnd, vcc, gnd);  V3         |
-- don't care                                       |  Sector II
-- Rom_II.data[12][] = (vcc, vcc, vcc);  V0         |
-- Rom_II.data[13][] = (vcc, vcc, gnd);  V2         |
-- Rom_II.data[14][] = (gnd, vcc, gnd);  V3         |
-- don't care
-- Rom_II.data[16][] = (vcc, vcc, vcc);  V0         |
-- Rom_II.data[17][] = (gnd, vcc, gnd);  V3         |
-- Rom_II.data[18][] = (gnd, vcc, vcc);  V4         |
-- don't care                                       |  Sector III
-- Rom_II.data[20][] = (gnd, gnd, gnd);  V7         |
-- Rom_II.data[21][] = (gnd, vcc, gnd);  V3         |
-- Rom_II.data[22][] = (gnd, vcc, vcc);  V4         |
-- don't care
-- Rom_II.data[24][] = (gnd, gnd, gnd);  V7         |
-- Rom_II.data[25][] = (gnd, vcc, vcc);  V4         |
-- Rom_II.data[26][] = (gnd, gnd, vcc);  V5         |
-- don't care                                       |  Sector IV
-- Rom_II.data[28][] = (vcc, vcc, vcc);  V0         |
-- Rom_II.data[29][] = (gnd, vcc, vcc);  V4         |
-- Rom_II.data[30][] = (gnd, gnd, vcc);  V5         |
-- don't care
-- Rom_II.data[32][] = (vcc, vcc, vcc);  V0         |
```

```
-- Rom_II.data[33][] = (gnd, gnd, vcc);  V5        |
-- Rom_II.data[34][] = (vcc, gnd, vcc);  V6        |
-- don't care                                                |  Sector V
-- Rom_II.data[36][] = (gnd, gnd, gnd);  V7        |
-- Rom_II.data[37][] = (gnd, gnd, vcc);  V5        |
-- Rom_II.data[38][] = (vcc, gnd, vcc);  V6        |
-- don't care
-- Rom_II.data[40][] = (gnd, gnd, gnd);  V7        |
-- Rom_II.data[41][] = (vcc, gnd, vcc);  V6        |
-- Rom_II.data[42][] = (vcc, gnd, gnd);  V1        |
-- don't care                                                |  Sector VI
-- Rom_II.data[44][] = (vcc, vcc, vcc);  V0        |
-- Rom_II.data[45][] = (vcc, gnd, vcc);  V6        |
-- Rom_II.data[46][] = (vcc, gnd, gnd);  V1        |
-- don't care

-- Acomodo la direccion A2 de la ROM II, para seleccionar adecuadamente el vector
-- nulo cuando tb o ta son cero.
            IF Cero_B.aeb THEN
                  Null = VCC;
            ELSE
                  Null = MSB;
            END IF;

-- Implementacion de la senal del LacthOut.
retardo1 = LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(ta)))))))); --Retardo aprox:20nseg.
retardo2 = LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(tb)))))))); 
retardo3 = LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(LCELL(!ta & !tb)))))))); 
LatchOut = (ta & retardo1) # (tb & retardo2) # ((!ta & !tb) & retardo3);

-- Implementacion del Latch de salida (L_Out) y de los estados de las llaves.
          L_out[].clk = LatchOut;
      L_out[0..2].d = (Rsw1, Rsw3, Rsw5);


--*********************************************************************
--* Implementacion de Tiempo de seguridad Td para SEIS lineas.
      SftReg_SW1.clock = clk;
      SftReg_SW2.clock = clk;
      SftReg_SW3.clock = clk;
      SftReg_SW4.clock = clk;
      SftReg_SW5.clock = clk;
      SftReg_SW6.clock = clk;
-- Bloque viejo, cuando no se implementaba la compensacion de Td.
%     SftReg_SW1.shiftin =  L_out0.q;
      SftReg_SW2.shiftin = !L_out0.q;
      SftReg_SW3.shiftin =  L_out1.q;
      SftReg_SW4.shiftin = !L_out1.q;
      SftReg_SW5.shiftin =  L_out2.q;
      SftReg_SW6.shiftin = !L_out2.q;
      SftReg_SW1.sclr = !L_out0.q;
      SftReg_SW2.sclr =  L_out0.q;
      SftReg_SW3.sclr = !L_out1.q;
      SftReg_SW4.sclr =  L_out1.q;
      SftReg_SW5.sclr = !L_out2.q;
      SftReg_SW6.sclr =  L_out2.q;
      Sw1_Td = SftReg_SW1.shiftout;
      Sw2_Td = SftReg_SW2.shiftout;
      Sw3_Td = SftReg_SW3.shiftout;
      Sw4_Td = SftReg_SW4.shiftout;
      Sw5_Td = SftReg_SW5.shiftout;
      Sw6_Td = SftReg_SW6.shiftout;
%
-- Las siguientes 18 lineas reemplazan a las 18 lineas anteriores. MT 01-06-01
      SftReg_SW1.shiftin = Mux_Comp_Td.result[0];
      SftReg_SW2.shiftin = Mux_Comp_Td.result[1];
      SftReg_SW3.shiftin = Mux_Comp_Td.result[2];
      SftReg_SW4.shiftin = Mux_Comp_Td.result[3];
      SftReg_SW5.shiftin = Mux_Comp_Td.result[4];
```

```
       SftReg_SW6.shiftin = Mux_Comp_Td.result[5];
%      SftReg_SW1.sclr =  !Mux_Comp_Td.result[0];
       SftReg_SW2.sclr =  !Mux_Comp_Td.result[1];
       SftReg_SW3.sclr =  !Mux_Comp_Td.result[2];
       SftReg_SW4.sclr =  !Mux_Comp_Td.result[3];
       SftReg_SW5.sclr =  !Mux_Comp_Td.result[4];
       SftReg_SW6.sclr =  !Mux_Comp_Td.result[5];
%
-- MT 05-05-02: Flip-Flop Adicionales para clear sincronico del shiftregister, se espera
-- evitar que glitches generen un cambio en la activacion de los SWx de 2 useg.
-- Estas lineas reemplazan a las 6 anteriores
    FF_TdSw1.clk = !clk;
    FF_TdSw2.clk = !clk;
    FF_TdSw3.clk = !clk;
    FF_TdSw4.clk = !clk;
    FF_TdSw5.clk = !clk;
    FF_TdSw6.clk = !clk;
    FF_TdSw1.d = Mux_Comp_Td.result[0];
    FF_TdSw2.d = Mux_Comp_Td.result[1];
    FF_TdSw3.d = Mux_Comp_Td.result[2];
    FF_TdSw4.d = Mux_Comp_Td.result[3];
    FF_TdSw5.d = Mux_Comp_Td.result[4];
    FF_TdSw6.d = Mux_Comp_Td.result[5];
       SftReg_SW1.sclr =  !FF_TdSw1.q;
       SftReg_SW2.sclr =  !FF_TdSw2.q;
       SftReg_SW3.sclr =  !FF_TdSw3.q;
       SftReg_SW4.sclr =  !FF_TdSw4.q;
       SftReg_SW5.sclr =  !FF_TdSw5.q;
       SftReg_SW6.sclr =  !FF_TdSw6.q;
-- Fin Flip-Flop adicionales

       Sw1_Td = SftReg_SW1.shiftout;
       Sw2_Td = SftReg_SW2.shiftout;
       Sw3_Td = SftReg_SW3.shiftout;
       Sw4_Td = SftReg_SW4.shiftout;
       Sw5_Td = SftReg_SW5.shiftout;
       Sw6_Td = SftReg_SW6.shiftout;

%*****************************************************************************
 Inicio Bloque para implementacion de la compensacion del tiempo muerto Td
%
       C_td_R1.ldn = Start;  -- Cuando el modulador esta detenido inicializo
       C_td_R2.ldn = Start;  -- a los contadores que se utilizan para medir el
       C_td_S1.ldn = Start;  -- tiempo de desviación, cargandoles el valor FF.
       C_td_S2.ldn = Start;
       C_td_T1.ldn = Start;
       C_td_T2.ldn = Start;
       C_td_R1.a = VCC;
       C_td_R1.b = VCC;
       C_td_R1.c = VCC;
       C_td_R1.d = VCC;
       C_td_R2.a = VCC;
       C_td_R2.b = VCC;
       C_td_R2.c = VCC;
       C_td_R2.d = VCC;

       C_td_S1.a = VCC;
       C_td_S1.b = VCC;
       C_td_S1.c = VCC;
       C_td_S1.d = VCC;
       C_td_S2.a = VCC;
       C_td_S2.b = VCC;
       C_td_S2.c = VCC;
       C_td_S2.d = VCC;

       C_td_T1.a = VCC;
       C_td_T1.b = VCC;
       C_td_T1.c = VCC;
       C_td_T1.d = VCC;
```

```
      C_td_T2.a = VCC;
      C_td_T2.b = VCC;
      C_td_T2.c = VCC;
      C_td_T2.d = VCC;

        up_R = !(!VR0TTL & L_out0.q & clk);
      down_R = !(VR0TTL & !L_out0.q & clk);
        up_S = !(!VS0TTL & L_out1.q & clk);
      down_S = !(VS0TTL & !L_out1.q & clk);
        up_T = !(!VT0TTL & L_out2.q & clk);
      down_T = !(VT0TTL & !L_out2.q & clk);

      C_td_R1.up = up_R;
      C_td_R1.dn = down_R;
      C_td_R2.up = C_td_R1.con;
      C_td_R2.dn = C_td_R1.bon;

      C_td_S1.up = up_S;
      C_td_S1.dn = down_S;
      C_td_S2.up = C_td_S1.con;
      C_td_S2.dn = C_td_S1.bon;

      C_td_T1.up = up_T;
      C_td_T1.dn = down_T;
      C_td_T2.up = C_td_T1.con;
      C_td_T2.dn = C_td_T1.bon;

       Carri_R = C_td_R2.con;   -- activa en cero
      Borrou_R = C_td_R2.bon;   -- activa en cero
       Carri_S = C_td_S2.con;   -- activa en cero
      Borrou_S = C_td_S2.bon;   -- activa en cero
       Carri_T = C_td_T2.con;   -- activa en cero
      Borrou_T = C_td_T2.bon;   -- activa en cero

      ff_R.prn  = Carri_R;
      ff_R.clrn = Borrou_R;
      ff_R.d    = GND;
      ff_R.clk  = GND;
             ff_S.prn = Carri_S;
             ff_S.clrn = Borrou_S;
             ff_S.d    = GND;


              ff_S.clk  = GND;
                    ff_T.prn  = Carri_T;
                    ff_T.clrn = Borrou_T;
                    ff_T.d    = GND;
                    ff_T.clk  = GND;

      Sw1_Com_Td = ff_R.q;
      Sw2_Com_Td = !ff_R.q;
      Sw3_Com_Td = ff_S.q;
      Sw4_Com_Td = !ff_S.q;
      Sw5_Com_Td = ff_T.q;
      Sw6_Com_Td = !ff_T.q;
--*
--*********************************************************************
-- Multiplexer de Compensacion de Td (Mux_Comp_Td)

      Mux_Comp_Td.data[0][0] = L_out0.q;
      Mux_Comp_Td.data[0][1] = !L_out0.q;
      Mux_Comp_Td.data[0][2] = L_out1.q;
      Mux_Comp_Td.data[0][3] = !L_out1.q;
      Mux_Comp_Td.data[0][4] = L_out2.q;
      Mux_Comp_Td.data[0][5] = !L_out2.q;

      Mux_Comp_Td.data[1][0] = Sw1_Com_Td;
      Mux_Comp_Td.data[1][1] = Sw2_Com_Td;
      Mux_Comp_Td.data[1][2] = Sw3_Com_Td;
```

```
      Mux_Comp_Td.data[1][3] = Sw4_Com_Td;
      Mux_Comp_Td.data[1][4] = Sw5_Com_Td;
      Mux_Comp_Td.data[1][5] = Sw6_Com_Td;

      Mux_Comp_Td.sel[0] = Enable_Comp_Td;

--*****************************************************************
-- Multiplexer de Salida (Mux_Td)

      Sw_testR = L_out0.q; -- Para verificar funcionamiento de compensacion de Td
      Sw_testS = L_out1.q; -- Para verificar funcionamiento de compensacion de Td
      Sw_testT = L_out2.q; -- Para verificar funcionamiento de compensacion de Td

      Mux_Td.data[0][0] = L_out0.q;
      Mux_Td.data[0][1] = !L_out0.q;
      Mux_Td.data[0][2] = L_out1.q;
      Mux_Td.data[0][3] = !L_out1.q;
      Mux_Td.data[0][4] = L_out2.q;
      Mux_Td.data[0][5] = !L_out2.q;

      Mux_Td.data[1][0] = Sw1_Td;
      Mux_Td.data[1][1] = Sw2_Td;
      Mux_Td.data[1][2] = Sw3_Td;
      Mux_Td.data[1][3] = Sw4_Td;
      Mux_Td.data[1][4] = Sw5_Td;
      Mux_Td.data[1][5] = Sw6_Td;

      Mux_Td.sel[0] = Enable_Td;


-------------------------------------------------------------------------------------------
--*          Llaves del Inversor
-------------------------------------------------------------------------------------------
-- Sin los multiplexer Mux_Td y sin la compensacion de Td. MT 28-12-01.
%     Sw1 = Tri(!Sw1_Td OR !Start, Vcc);
      Sw2 = Tri(!Sw2_Td OR !Start, Vcc);
      Sw3 = Tri(!Sw3_Td OR !Start, Vcc);
      Sw4 = Tri(!Sw4_Td OR !Start, Vcc);
      Sw5 = Tri(!Sw5_Td OR !Start, Vcc);
      Sw6 = Tri(!Sw6_Td OR !Start, Vcc);
%
-- Las salidas se niegan debido al agregado de los optoacopladores. MT 27-11-01
-- MT 27-12-01. En estado OFF, las salidas deben estar en "1" antes de los optoacopladores!
      Sw1 = Tri(!Mux_Td.result[0] OR !Start, VCC);
      Sw2 = Tri(!Mux_Td.result[1] OR !Start, VCC);
      Sw3 = Tri(!Mux_Td.result[2] OR !Start, VCC);
      Sw4 = Tri(!Mux_Td.result[3] OR !Start, VCC);
      Sw5 = Tri(!Mux_Td.result[4] OR !Start, VCC);
      Sw6 = Tri(!Mux_Td.result[5] OR !Start, VCC);
--*

--*****************************************************************
-- Fin del algoritmo SVM
--*****************************************************************

--*************************************************************************************
--****                    CHICHES DEL MODULADOR                                    ****
  TABLE
   StopN => D1a, D1b, D1c, D1d, D1e, D1f, D1g, D1p,   D2a, D2b, D2c, D2d, D2e, D2f, D2g, D2p;
     1   =>   0,   0,   1,   0,   0,   1,   0,   1,     1,   1,   0,   0,   0,   1,   1,   1;
            -- 2u encendido
     0   =>   0,   0,   0,   0,   0,   0,   1,   1,     0,   1,   1,   1,   0,   0,   0,   1;
            -- OF apagado
  END TABLE;
--****                    FIN CHICHES DEL MODULADOR                                ****
--*************************************************************************************

END; -- Del modulador.
```

TABLAS CON EL CONTENIDO DE LAS EPROM ta Y tb

En las siguientes tablas se muestran los datos cuantizados de los tiempos ta y tb, los cuales se programan en las EPROM ta y tb respectivamente.

;*** Datos de tiempos ta a grabar en la **EPROM ta** ***
;*** (son los tiempos que debe contar cada Contador 74LS193) ***

;*** Parametros utilizados para generar el archivo: ***
;*** Bit para cuantificar el tiempo: 8 ***
;*** Frecuencia del clock en Hz: 8375000 ***
;*** Bit para m: 7 ***
;*** Bit para alfa: 6 ***

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
0	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**0**	**15**
16	**10**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**16**	**31**
32	**20**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**32**	**47**
48	**30**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**48**	**63**
64	**40**	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**1**	**0**	**15**
80	**50**	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1		**16**	**31**
96	**60**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		**32**	**47**
112	**70**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**48**	**63**
128	**80**	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	**2**	**0**	**15**
144	**90**	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2		**16**	**31**
160	**A0**	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1		**32**	**47**
176	**B0**	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0		**48**	**63**
192	**C0**	6	6	6	6	6	5	5	5	5	5	5	5	5	5	5	5	**3**	**0**	**15**
208	**D0**	5	5	5	4	4	4	4	4	4	4	4	4	4	4	3	3		**16**	**31**
224	**E0**	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2		**32**	**47**
240	**F0**	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0		**48**	**63**
256	**100**	8	8	8	7	7	7	7	7	7	7	7	7	7	7	6	6	**4**	**0**	**15**
272	**110**	6	6	6	6	6	6	6	5	5	5	5	5	5	5	5	4		**16**	**31**
288	**120**	4	4	4	4	4	4	4	3	3	3	3	3	3	3	2	2		**32**	**47**
304	**130**	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0		**48**	**63**
320	**140**	a	9	9	9	9	9	9	9	9	9	9	8	8	8	8	8	**5**	**0**	**15**
336	**150**	8	8	8	7	7	7	7	7	7	7	6	6	6	6	6	6		**16**	**31**
352	**160**	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3		**32**	**47**
368	**170**	3	3	2	2	2	2	2	1	1	1	1	1	1	0	0	0		**48**	**63**
384	**180**	c	b	b	b	b	b	b	b	b	a	a	a	a	a	a	a	**6**	**0**	**15**
400	**190**	9	9	9	9	9	9	8	8	8	8	8	7	7	7	7	7		**16**	**31**
416	**1A0**	7	6	6	6	6	6	5	5	5	5	5	4	4	4	4	3		**32**	**47**
432	**1B0**	3	3	3	3	2	2	2	2	2	1	1	1	1	0	0	0		**48**	**63**
448	**1C0**	d	d	d	d	d	d	d	c	c	c	c	c	c	b	b	b	**7**	**0**	**15**
464	**1D0**	b	b	b	a	a	a	a	a	9	9	9	9	9	8	8	8		**16**	**31**
480	**1E0**	8	7	7	7	7	6	6	6	6	6	5	5	5	5	4	4		**32**	**47**
496	**1F0**	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0	0		**48**	**63**
512	**200**	f	f	f	f	f	f	e	e	e	e	e	d	d	d	d	d	**8**	**0**	**15**
528	**210**	c	c	c	c	c	b	b	b	b	a	a	a	a	9	9	9		**16**	**31**
544	**220**	9	8	8	8	8	7	7	7	7	6	6	6	6	5	5	5		**32**	**47**
560	**230**	4	4	4	4	3	3	3	2	2	2	1	1	1	1	0	0		**48**	**63**
576	**240**	11	11	11	11	11	10	10	10	10	10	f	f	f	f	e	e	**9**	**0**	**15**
592	**250**	e	e	e	d	d	d	d	c	c	c	b	b	b	b	a	a		**16**	**31**
608	**260**	a	a	9	9	9	8	8	8	7	7	7	7	6	6	6	5		**32**	**47**
624	**270**	5	5	4	4	4	3	3	3	2	2	2	1	1	1	0	0		**48**	**63**
640	**280**	13	13	13	13	12	12	12	12	12	11	11	11	11	10	10	10	**10**	**0**	**15**
656	**290**	10	f	f	f	f	e	e	e	d	d	d	c	c	c	c	b		**16**	**31**
672	**2A0**	b	b	a	a	a	9	9	9	8	8	8	7	7	7	6	6		**32**	**47**
688	**2B0**	5	5	5	4	4	4	3	3	3	2	2	1	1	1	0	0		**48**	**63**
704	**2C0**	15	15	15	14	14	14	14	14	13	13	13	13	12	12	12	11	**11**	**0**	**15**
720	**2D0**	11	11	11	10	10	10	f	f	f	e	e	e	d	d	d	c		**16**	**31**
736	**2E0**	c	c	b	b	b	a	a	9	9	9	8	8	8	7	7	6		**32**	**47**
752	**2F0**	6	6	5	5	4	4	4	3	3	2	2	2	1	1	0	0		**48**	**63**
768	**300**	17	17	17	16	16	16	16	15	15	15	14	14	14	14	13	13	**12**	**0**	**15**
784	**310**	13	12	12	12	11	11	11	10	10	10	f	f	f	e	e	d		**16**	**31**
800	**320**	d	d	c	c	c	b	b	a	a	a	9	9	8	8	7	7		**32**	**47**
816	**330**	7	6	6	5	5	4	4	4	3	3	2	2	1	1	0	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
832	**340**	19	19	18	18	18	18	17	17	17	17	16	16	16	15	15	15	**13**	**0**	**15**
848	**350**	14	14	14	13	13	13	12	12	11	11	11	10	10	f	f	f		**16**	**31**
864	**360**	e	e	d	d	c	c	c	b	b	a	a	9	9	8	8	8		**32**	**47**
880	**370**	7	7	6	6	5	5	4	4	3	3	2	2	1	1	0	0		**48**	**63**
896	**380**	1b	1b	1a	1a	1a	19	19	19	19	18	18	18	17	17	17	16	**14**	**0**	**15**
912	**390**	16	15	15	15	14	14	14	13	13	12	12	11	11	11	10	10		**16**	**31**
928	**3A0**	f	f	e	e	d	d	d	c	c	b	b	a	a	9	9	8		**32**	**47**
944	**3B0**	8	7	7	6	6	5	5	4	4	3	3	2	2	1	1	0		**48**	**63**
960	**3C0**	1d	1c	1c	1c	1c	1b	1b	1b	1a	1a	1a	19	19	19	18	18	**15**	**0**	**15**
976	**3D0**	17	17	17	16	16	15	15	14	14	14	13	13	12	12	11	11		**16**	**31**
992	**3E0**	10	10	f	f	e	e	d	d	c	c	b	b	a	a	9	9		**32**	**47**
1008	**3F0**	8	8	7	7	6	5	5	4	4	3	3	2	2	1	1	0		**48**	**63**
1024	**400**	1f	1e	1e	1e	1d	1d	1d	1c	1c	1c	1b	1b	1b	1a	1a	19	**16**	**0**	**15**
1040	**410**	19	19	18	18	17	17	16	16	15	15	14	14	13	13	12	12		**16**	**31**
1056	**420**	11	11	10	10	f	f	e	e	d	d	c	c	b	a	a	9		**32**	**47**
1072	**430**	9	8	8	7	6	6	5	5	4	4	3	2	2	1	1	0		**48**	**63**
1088	**440**	21	20	20	20	1f	1f	1f	1e	1e	1d	1d	1d	1c	1c	1b	1b	**17**	**0**	**15**
1104	**450**	1b	1a	1a	19	19	18	18	17	17	16	16	15	15	14	14	13		**16**	**31**
1120	**460**	13	12	11	11	10	10	f	f	e	d	d	c	c	b	a	a		**32**	**47**
1136	**470**	9	9	8	7	7	6	6	5	4	4	3	3	2	1	1	0		**48**	**63**
1152	**480**	23	22	22	21	21	21	20	20	20	1f	1f	1e	1e	1d	1d	1d	**18**	**0**	**15**
1168	**490**	1c	1c	1b	1b	1a	1a	19	19	18	18	17	16	16	15	15	14		**16**	**31**
1184	**4A0**	14	13	12	12	11	11	10	f	f	e	e	d	c	c	b	a		**32**	**47**
1200	**4B0**	a	9	9	8	7	7	6	5	5	4	3	3	2	1	1	0		**48**	**63**
1216	**4C0**	24	24	24	23	23	23	22	22	21	21	20	20	20	1f	1f	1e	**19**	**0**	**15**
1232	**4D0**	1e	1d	1d	1c	1c	1b	1b	1a	19	19	18	18	17	17	16	15		**16**	**31**
1248	**4E0**	15	14	14	13	12	12	11	10	10	f	e	e	d	c	c	b		**32**	**47**
1264	**4F0**	a	a	9	8	8	7	6	6	5	4	3	3	2	1	1	0		**48**	**63**
1280	**500**	26	26	26	25	25	24	24	24	23	23	22	22	21	21	20	20	**20**	**0**	**15**
1296	**510**	1f	1f	1e	1e	1d	1c	1c	1b	1b	1a	1a	19	18	18	17	16		**16**	**31**
1312	**520**	16	15	15	14	13	13	12	11	11	10	f	e	e	d	c	c		**32**	**47**
1328	**530**	b	a	9	9	8	7	7	6	5	4	4	3	2	1	1	0		**48**	**63**
1344	**540**	28	28	27	27	27	26	26	25	25	24	24	23	23	22	22	21	**21**	**0**	**15**
1360	**550**	21	20	20	1f	1e	1e	1d	1d	1c	1b	1b	1a	1a	19	18	18		**16**	**31**
1376	**560**	17	16	16	15	14	13	13	12	11	11	10	f	e	e	d	c		**32**	**47**
1392	**570**	b	b	a	9	8	8	7	6	5	5	4	3	2	2	1	0		**48**	**63**
1408	**580**	2a	2a	29	29	28	28	28	27	27	26	26	25	25	24	23	23	**22**	**0**	**15**
1424	**590**	22	22	21	21	20	1f	1f	1e	1d	1d	1c	1b	1b	1a	19	19		**16**	**31**
1440	**5A0**	18	17	17	16	15	14	14	13	12	11	11	10	f	e	e	d		**32**	**47**
1456	**5B0**	c	b	a	a	9	8	7	6	6	5	4	3	2	2	1	0		**48**	**63**
1472	**5C0**	2c	2c	2b	2b	2a	2a	29	29	28	28	27	27	26	26	25	24	**23**	**0**	**15**
1488	**5D0**	24	23	23	22	21	21	20	1f	1f	1e	1d	1d	1c	1b	1b	1a		**16**	**31**
1504	**5E0**	19	18	18	17	16	15	15	14	13	12	11	11	10	f	e	d		**32**	**47**
1520	**5F0**	d	c	b	a	9	8	8	7	6	5	4	3	3	2	1	0		**48**	**63**
1536	**600**	2e	2e	2d	2d	2c	2c	2b	2b	2a	2a	29	28	28	27	27	26	**24**	**0**	**15**
1552	**610**	25	25	24	23	23	22	21	21	20	1f	1f	1e	1d	1c	1c	1b		**16**	**31**
1568	**620**	1a	19	19	18	17	16	15	15	14	13	12	11	11	10	f	e		**32**	**47**
1584	**630**	d	c	b	b	a	9	8	7	6	5	4	4	3	2	1	0		**48**	**63**
1600	**640**	30	2f	2f	2e	2e	2d	2d	2c	2c	2b	2b	2a	2a	29	28	28	**25**	**0**	**15**
1616	**650**	27	26	26	25	24	24	23	22	21	21	20	1f	1e	1e	1d	1c		**16**	**31**
1632	**660**	1b	1a	1a	19	18	17	16	15	15	14	13	12	11	10	f	f		**32**	**47**
1648	**670**	e	d	c	b	a	9	8	7	6	6	5	4	3	2	1	0		**48**	**63**
1664	**680**	32	31	31	30	30	2f	2f	2e	2e	2d	2c	2c	2b	2b	2a	29	**26**	**0**	**15**
1680	**690**	29	28	27	26	26	25	24	24	23	22	21	20	20	1f	1e	1d		**16**	**31**
1696	**6A0**	1c	1c	1b	1a	19	18	17	16	15	15	14	13	12	11	10	f		**32**	**47**
1712	**6B0**	e	d	c	b	a	a	9	8	7	6	5	4	3	2	1	0		**48**	**63**
1728	**6C0**	34	33	33	32	32	31	31	30	2f	2f	2e	2d	2d	2c	2b	2b	**27**	**0**	**15**
1744	**6D0**	2a	29	29	28	27	26	26	25	24	23	22	22	21	20	1f	1e		**16**	**31**
1760	**6E0**	1d	1d	1c	1b	1a	19	18	17	16	15	14	14	13	12	11	10		**32**	**47**
1776	**6F0**	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
1792	**700**	36	35	35	34	34	33	32	32	31	30	30	2f	2e	2e	2d	2c	**28**	**0**	**15**
1808	**710**	2c	2b	2a	29	29	28	27	26	25	25	24	23	22	21	20	1f		**16**	**31**
1824	**720**	1f	1e	1d	1c	1b	1a	19	18	17	16	15	14	13	12	11	10		**32**	**47**
1840	**730**	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0		**48**	**63**
1856	**740**	38	37	37	36	35	35	34	34	33	32	32	31	30	2f	2f	2e	**29**	**0**	**15**
1872	**750**	2d	2c	2c	2b	2a	29	28	28	27	26	25	24	23	22	21	21		**16**	**31**
1888	**760**	20	1f	1e	1d	1c	1b	1a	19	18	17	16	15	14	13	12	11		**32**	**47**
1904	**770**	10	f	e	d	c	b	a	9	7	6	5	4	3	2	1	0		**48**	**63**
1920	**780**	3a	39	38	38	37	37	36	35	35	34	33	33	32	31	30	30	**30**	**0**	**15**
1936	**790**	2f	2e	2d	2c	2c	2b	2a	29	28	27	26	25	24	24	23	22		**16**	**31**
1952	**7A0**	21	20	1f	1e	1d	1c	1b	1a	19	18	17	16	15	14	13	11		**32**	**47**
1968	**7B0**	10	f	e	d	c	b	a	9	8	7	6	4	3	2	1	0		**48**	**63**
1984	**7C0**	3b	3b	3a	3a	39	38	38	37	36	36	35	34	33	33	32	31	**31**	**0**	**15**
2000	**7D0**	30	30	2f	2e	2d	2c	2b	2a	29	29	28	27	26	25	24	23		**16**	**31**
2016	**7E0**	22	21	20	1f	1e	1d	1c	1b	1a	19	17	16	15	14	13	12		**32**	**47**
2032	**7F0**	11	10	f	e	c	b	a	9	8	7	6	5	3	2	1	0		**48**	**63**
2048	**800**	3d	3d	3c	3c	3b	3a	3a	39	38	37	37	36	35	34	34	33	**32**	**0**	**15**
2064	**810**	32	31	30	2f	2e	2e	2d	2c	2b	2a	29	28	27	26	25	24		**16**	**31**
2080	**820**	23	22	21	20	1f	1e	1d	1c	1a	19	18	17	16	15	14	13		**32**	**47**
2096	**830**	11	10	f	e	d	c	b	9	8	7	6	5	4	2	1	0		**48**	**63**
2112	**840**	3f	3f	3e	3d	3d	3c	3b	3b	3a	39	38	38	37	36	35	34	**33**	**0**	**15**
2128	**850**	33	33	32	31	30	2f	2e	2d	2c	2b	2a	29	28	27	26	25		**16**	**31**
2144	**860**	24	23	22	21	20	1f	1d	1c	1b	1a	19	18	17	16	14	13		**32**	**47**
2160	**870**	12	11	10	e	d	c	b	a	8	7	6	5	4	2	1	0		**48**	**63**
2176	**880**	41	41	40	3f	3f	3e	3d	3c	3c	3b	3a	39	38	38	37	36	**34**	**0**	**15**
2192	**890**	35	34	33	32	31	30	2f	2e	2d	2c	2b	2a	29	28	27	26		**16**	**31**
2208	**8A0**	25	24	23	22	21	20	1e	1d	1c	1b	1a	19	17	16	15	14		**32**	**47**
2224	**8B0**	13	11	10	f	e	c	b	a	9	7	6	5	4	3	1	0		**48**	**63**
2240	**8C0**	43	42	42	41	40	40	3f	3e	3d	3d	3c	3b	3a	39	38	37	**35**	**0**	**15**
2256	**8D0**	37	36	35	34	33	32	31	30	2f	2e	2d	2c	2b	29	28	27		**16**	**31**
2272	**8E0**	26	25	24	23	22	20	1f	1e	1d	1c	1b	19	18	17	16	14		**32**	**47**
2288	**8F0**	13	12	11	f	e	d	c	a	9	8	6	5	4	3	1	0		**48**	**63**
2304	**900**	45	44	44	43	42	41	41	40	3f	3e	3d	3d	3c	3b	3a	39	**36**	**0**	**15**
2320	**910**	38	37	36	35	34	33	32	31	30	2f	2e	2d	2c	2b	2a	28		**16**	**31**
2336	**920**	27	26	25	24	23	21	20	1f	1e	1d	1b	1a	19	17	16	15		**32**	**47**
2352	**930**	14	12	11	10	e	d	c	b	9	8	7	5	4	3	1	0		**48**	**63**
2368	**940**	47	46	46	45	44	43	43	42	41	40	3f	3e	3d	3d	3c	3b	**37**	**0**	**15**
2384	**950**	3a	39	38	37	36	35	34	33	31	30	2f	2e	2d	2c	2b	2a		**16**	**31**
2400	**960**	28	27	26	25	24	22	21	20	1f	1d	1c	1b	19	18	17	16		**32**	**47**
2416	**970**	14	13	12	10	f	e	c	b	a	8	7	5	4	3	1	0		**48**	**63**
2432	**980**	49	48	47	47	46	45	44	43	43	42	41	40	3f	3e	3d	3c	**38**	**0**	**15**
2448	**990**	3b	3a	39	38	37	36	35	34	33	32	31	2f	2e	2d	2c	2b		**16**	**31**
2464	**9A0**	29	28	27	26	25	23	22	21	1f	1e	1d	1b	1a	19	17	16		**32**	**47**
2480	**9B0**	15	13	12	11	f	e	d	b	a	8	7	6	4	3	1	0		**48**	**63**
2496	**9C0**	4b	4a	49	49	48	47	46	45	44	44	43	42	41	40	3f	3e	**39**	**0**	**15**
2512	**9D0**	3d	3c	3b	3a	39	38	36	35	34	33	32	31	2f	2e	2d	2c		**16**	**31**
2528	**9E0**	2b	29	28	27	25	24	23	22	20	1f	1e	1c	1b	19	18	17		**32**	**47**
2544	**9F0**	15	14	13	11	10	e	d	b	a	9	7	6	4	3	1	0		**48**	**63**
2560	**A00**	4d	4c	4b	4a	4a	49	48	47	46	45	44	43	42	41	40	3f	**40**	**0**	**15**
2576	**A10**	3e	3d	3c	3b	3a	39	38	37	35	34	33	32	31	2f	2e	2d		**16**	**31**
2592	**A20**	2c	2a	29	28	26	25	24	22	21	20	1e	1d	1c	1a	19	17		**32**	**47**
2608	**A30**	16	14	13	12	10	f	d	c	a	9	7	6	4	3	1	0		**48**	**63**
2624	**A40**	4f	4e	4d	4c	4b	4b	4a	49	48	47	46	45	44	43	42	41	**41**	**0**	**15**
2640	**A50**	40	3f	3e	3d	3b	3a	39	38	37	36	34	33	32	31	2f	2e		**16**	**31**
2656	**A60**	2d	2b	2a	29	27	26	25	23	22	20	1f	1e	1c	1b	19	18		**32**	**47**
2672	**A70**	16	15	13	12	11	f	e	c	b	9	8	6	5	3	2	0		**48**	**63**
2688	**A80**	51	50	4f	4e	4d	4c	4c	4b	4a	49	48	47	46	45	44	43	**42**	**0**	**15**
2704	**A90**	41	40	3f	3e	3d	3c	3b	39	38	37	36	34	33	32	30	2f		**16**	**31**
2720	**AA0**	2e	2c	2b	2a	28	27	26	24	23	21	20	1e	1d	1b	1a	18		**32**	**47**
2736	**AB0**	17	15	14	12	11	f	e	c	b	9	8	6	5	3	2	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
2752	**AC0**	52	52	51	50	4f	4e	4d	4c	4b	4a	49	48	47	46	45	44	**43**	**0**	**15**
2768	**AD0**	43	42	41	40	3e	3d	3c	3b	39	38	37	36	34	33	32	30		**16**	**31**
2784	**AE0**	2f	2e	2c	2b	29	28	26	25	24	22	21	1f	1e	1c	1b	19		**32**	**47**
2800	**AF0**	17	16	14	13	11	10	e	d	b	9	8	6	5	3	2	0		**48**	**63**
2816	**B00**	54	54	53	52	51	50	4f	4e	4d	4c	4b	4a	49	48	47	46	**44**	**0**	**15**
2832	**B10**	45	43	42	41	40	3f	3d	3c	3b	3a	38	37	36	34	33	31		**16**	**31**
2848	**B20**	30	2f	2d	2c	2a	29	27	26	24	23	21	20	1e	1d	1b	1a		**32**	**47**
2864	**B30**	18	16	15	13	12	10	f	d	b	a	8	6	5	3	2	0		**48**	**63**
2880	**B40**	56	55	55	54	53	52	51	50	4f	4e	4d	4c	4b	4a	48	47	**45**	**0**	**15**
2896	**B50**	46	45	44	43	41	40	3f	3d	3c	3b	39	38	37	35	34	33		**16**	**31**
2912	**B60**	31	30	2e	2d	2b	2a	28	27	25	24	22	21	1f	1d	1c	1a		**32**	**47**
2928	**B70**	19	17	15	14	12	10	f	d	c	a	8	7	5	3	2	0		**48**	**63**
2944	**B80**	58	57	56	56	55	54	53	52	51	50	4f	4d	4c	4b	4a	49	**46**	**0**	**15**
2960	**B90**	48	47	45	44	43	41	40	3f	3d	3c	3b	39	38	37	35	34		**16**	**31**
2976	**BA0**	32	31	2f	2e	2c	2b	29	28	26	24	23	21	20	1e	1c	1b		**32**	**47**
2992	**BB0**	19	17	16	14	13	11	f	e	c	a	8	7	5	3	2	0		**48**	**63**
3008	**BC0**	5a	59	58	57	56	55	54	53	52	51	50	4f	4e	4d	4c	4a	**47**	**0**	**15**
3024	**BD0**	49	48	47	45	44	43	42	40	3f	3d	3c	3b	39	38	36	35		**16**	**31**
3040	**BE0**	33	32	30	2f	2d	2c	2a	28	27	25	24	22	20	1f	1d	1b		**32**	**47**
3056	**BF0**	1a	18	16	15	13	11	10	e	c	a	9	7	5	3	2	0		**48**	**63**
3072	**C00**	5c	5b	5a	59	58	57	56	55	54	53	52	51	50	4f	4d	4c	**48**	**0**	**15**
3088	**C10**	4b	4a	48	47	46	44	43	42	40	3f	3d	3c	3a	39	37	36		**16**	**31**
3104	**C20**	34	33	31	30	2e	2d	2b	29	28	26	24	23	21	1f	1e	1c		**32**	**47**
3120	**C30**	1a	19	17	15	13	12	10	e	c	b	9	7	5	4	2	0		**48**	**63**
3136	**C40**	5e	5d	5c	5b	5a	59	58	57	56	55	54	53	51	50	4f	4e	**49**	**0**	**15**
3152	**C50**	4c	4b	4a	48	47	46	44	43	42	40	3f	3d	3c	3a	39	37		**16**	**31**
3168	**C60**	35	34	32	31	2f	2d	2c	2a	28	27	25	23	22	20	1e	1d		**32**	**47**
3184	**C70**	1b	19	17	15	14	12	10	e	d	b	9	7	5	4	2	0		**48**	**63**
3200	**C80**	60	5f	5e	5d	5c	5b	5a	59	58	57	55	54	53	52	51	4f	**50**	**0**	**15**
3216	**C90**	4e	4d	4b	4a	49	47	46	44	43	41	40	3e	3d	3b	3a	38		**16**	**31**
3232	**CA0**	37	35	33	32	30	2e	2d	2b	29	28	26	24	22	21	1f	1d		**32**	**47**
3248	**CB0**	1b	1a	18	16	14	12	10	f	d	b	9	7	6	4	2	0		**48**	**63**
3264	**CC0**	62	61	60	5f	5e	5d	5c	5b	59	58	57	56	55	53	52	51	**51**	**0**	**15**
3280	**CD0**	50	4e	4d	4b	4a	49	47	46	44	43	41	40	3e	3c	3b	39		**16**	**31**
3296	**CE0**	38	36	34	33	31	2f	2e	2c	2a	28	27	25	23	21	1f	1e		**32**	**47**
3312	**CF0**	1c	1a	18	16	15	13	11	f	d	b	9	8	6	4	2	0		**48**	**63**
3328	**D00**	64	63	62	61	60	5f	5d	5c	5b	5a	59	58	56	55	54	52	**52**	**0**	**15**
3344	**D10**	51	50	4e	4d	4b	4a	49	47	46	44	42	41	3f	3e	3c	3a		**16**	**31**
3360	**D20**	39	37	35	34	32	30	2e	2d	2b	29	27	26	24	22	20	1e		**32**	**47**
3376	**D30**	1c	1b	19	17	15	13	11	f	d	b	a	8	6	4	2	0		**48**	**63**
3392	**D40**	66	65	64	63	61	60	5f	5e	5d	5c	5b	59	58	57	55	54	**53**	**0**	**15**
3408	**D50**	53	51	50	4e	4d	4b	4a	48	47	45	44	42	40	3f	3d	3c		**16**	**31**
3424	**D60**	3a	38	36	35	33	31	2f	2e	2c	2a	28	26	24	23	21	1f		**32**	**47**
3440	**D70**	1d	1b	19	17	15	13	11	10	e	c	a	8	6	4	2	0		**48**	**63**
3456	**D80**	68	67	65	64	63	62	61	60	5f	5d	5c	5b	5a	58	57	56	**54**	**0**	**15**
3472	**D90**	54	53	51	50	4e	4d	4b	4a	48	47	45	43	42	40	3e	3d		**16**	**31**
3488	**DA0**	3b	39	37	36	34	32	30	2e	2d	2b	29	27	25	23	21	1f		**32**	**47**
3504	**DB0**	1e	1c	1a	18	16	14	12	10	e	c	a	8	6	4	2	0		**48**	**63**
3520	**DC0**	69	68	67	66	65	64	63	62	60	5f	5e	5d	5b	5a	59	57	**55**	**0**	**15**
3536	**DD0**	56	54	53	51	50	4e	4d	4b	4a	48	46	45	43	41	3f	3e		**16**	**31**
3552	**DE0**	3c	3a	38	37	35	33	31	2f	2d	2c	2a	28	26	24	22	20		**32**	**47**
3568	**DF0**	1e	1c	1a	18	16	14	12	10	e	c	a	8	6	4	2	0		**48**	**63**
3584	**E00**	6b	6a	69	68	67	66	65	63	62	61	60	5e	5d	5c	5a	59	**56**	**0**	**15**
3600	**E10**	57	56	54	53	51	50	4e	4c	4b	49	48	46	44	42	41	3f		**16**	**31**
3616	**E20**	3d	3b	39	38	36	34	32	30	2e	2c	2a	28	27	25	23	21		**32**	**47**
3632	**E30**	1f	1d	1b	19	17	15	12	10	e	c	a	8	6	4	2	0		**48**	**63**
3648	**E40**	6d	6c	6b	6a	69	68	66	65	64	63	61	60	5f	5d	5c	5a	**57**	**0**	**15**
3664	**E50**	59	57	56	54	53	51	50	4e	4c	4b	49	47	45	44	42	40		**16**	**31**
3680	**E60**	3e	3c	3b	39	37	35	33	31	2f	2d	2b	29	27	25	23	21		**32**	**47**
3696	**E70**	1f	1d	1b	19	17	15	13	11	f	d	a	8	6	4	2	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
3712	**E80**	6f	6e	6d	6c	6b	69	68	67	66	64	63	62	60	5f	5d	5c	**58**	**0**	**15**
3728	**E90**	5a	59	57	56	54	53	51	4f	4e	4c	4a	48	47	45	43	41		**16**	**31**
3744	**EA0**	3f	3d	3c	3a	38	36	34	32	30	2e	2c	2a	28	26	24	22		**32**	**47**
3760	**EB0**	20	1e	1c	19	17	15	13	11	f	d	b	9	6	4	2	0		**48**	**63**
3776	**EC0**	71	70	6f	6e	6d	6b	6a	69	67	66	65	63	62	60	5f	5e	**59**	**0**	**15**
3792	**ED0**	5c	5a	59	57	56	54	52	51	4f	4d	4b	4a	48	46	44	42		**16**	**31**
3808	**EE0**	40	3e	3d	3b	39	37	35	33	31	2f	2d	2b	29	27	24	22		**32**	**47**
3824	**EF0**	20	1e	1c	1a	18	16	13	11	f	d	b	9	7	4	2	0		**48**	**63**
3840	**F00**	73	72	71	70	6e	6d	6c	6b	69	68	66	65	64	62	61	5f	**60**	**0**	**15**
3856	**F10**	5e	5c	5a	59	57	55	54	52	50	4e	4d	4b	49	47	45	43		**16**	**31**
3872	**F20**	41	40	3e	3c	3a	38	36	34	32	30	2d	2b	29	27	25	23		**32**	**47**
3888	**F30**	21	1f	1c	1a	18	16	14	12	f	d	b	9	7	4	2	0		**48**	**63**
3904	**F40**	75	74	73	71	70	6f	6e	6c	6b	6a	68	67	65	64	62	61	**61**	**0**	**15**
3920	**F50**	5f	5d	5c	5a	59	57	55	53	52	50	4e	4c	4a	48	46	44		**16**	**31**
3936	**F60**	43	41	3f	3d	3b	39	37	34	32	30	2e	2c	2a	28	26	23		**32**	**47**
3952	**F70**	21	1f	1d	1b	19	16	14	12	10	d	b	9	7	4	2	0		**48**	**63**
3968	**F80**	77	76	75	73	72	71	6f	6e	6d	6b	6a	68	67	65	64	62	**62**	**0**	**15**
3984	**F90**	61	5f	5d	5c	5a	58	56	55	53	51	4f	4d	4b	4a	48	46		**16**	**31**
4000	**FA0**	44	42	40	3e	3c	39	37	35	33	31	2f	2d	2b	28	26	24		**32**	**47**
4016	**FB0**	22	20	1d	1b	19	17	14	12	10	e	b	9	7	5	2	0		**48**	**63**
4032	**FC0**	79	78	76	75	74	73	71	70	6e	6d	6c	6a	69	67	65	64	**63**	**0**	**15**
4048	**FD0**	62	61	5f	5d	5b	5a	58	56	54	52	50	4f	4d	4b	49	47		**16**	**31**
4064	**FE0**	45	43	41	3f	3d	3a	38	36	34	32	30	2e	2b	29	27	25		**32**	**47**
4080	**FF0**	22	20	1e	1c	19	17	15	12	10	e	c	9	7	5	2	0		**48**	**63**
4096	**1000**	7b	7a	78	77	76	74	73	72	70	6f	6d	6c	6a	69	67	65	**64**	**0**	**15**
4112	**1010**	64	62	60	5f	5d	5b	59	57	56	54	52	50	4e	4c	4a	48		**16**	**31**
4128	**1020**	46	44	42	40	3d	3b	39	37	35	33	30	2e	2c	2a	28	25		**32**	**47**
4144	**1030**	23	21	1e	1c	1a	17	15	13	10	e	c	9	7	5	2	0		**48**	**63**
4160	**1040**	7d	7b	7a	79	78	76	75	73	72	71	6f	6d	6c	6a	69	67	**65**	**0**	**15**
4176	**1050**	65	64	62	60	5e	5d	5b	59	57	55	53	51	4f	4d	4b	49		**16**	**31**
4192	**1060**	47	45	43	41	3e	3c	3a	38	36	33	31	2f	2d	2a	28	26		**32**	**47**
4208	**1070**	24	21	1f	1d	1a	18	15	13	11	e	c	a	7	5	2	0		**48**	**63**
4224	**1080**	7f	7d	7c	7b	79	78	77	75	74	72	71	6f	6e	6c	6a	69	**66**	**0**	**15**
4240	**1090**	67	65	63	62	60	5e	5c	5a	58	56	54	52	50	4e	4c	4a		**16**	**31**
4256	**10A0**	48	46	44	42	3f	3d	3b	39	37	34	32	30	2d	2b	29	26		**32**	**47**
4272	**10B0**	24	22	1f	1d	1b	18	16	13	11	f	c	a	7	5	2	0		**48**	**63**
4288	**10C0**	80	7f	7e	7d	7b	7a	78	77	75	74	72	71	6f	6e	6c	6a	**67**	**0**	**15**
4304	**10D0**	68	67	65	63	61	5f	5d	5c	5a	58	56	54	52	4f	4d	4b		**16**	**31**
4320	**10E0**	49	47	45	43	40	3e	3c	3a	37	35	33	30	2e	2c	29	27		**32**	**47**
4336	**10F0**	25	22	20	1d	1b	19	16	14	11	f	c	a	7	5	2	0		**48**	**63**
4352	**1100**	82	81	80	7e	7d	7c	7a	79	77	76	74	73	71	6f	6e	6c	**68**	**0**	**15**
4368	**1110**	6a	68	66	65	63	61	5f	5d	5b	59	57	55	53	51	4f	4c		**16**	**31**
4384	**1120**	4a	48	46	44	41	3f	3d	3a	38	36	33	31	2f	2c	2a	28		**32**	**47**
4400	**1130**	25	23	20	1e	1b	19	16	14	11	f	c	a	8	5	3	0		**48**	**63**
4416	**1140**	84	83	82	80	7f	7e	7c	7b	79	77	76	74	73	71	6f	6d	**69**	**0**	**15**
4432	**1150**	6c	6a	68	66	64	62	60	5e	5c	5a	58	56	54	52	50	4d		**16**	**31**
4448	**1160**	4b	49	47	45	42	40	3e	3b	39	37	34	32	2f	2d	2b	28		**32**	**47**
4464	**1170**	26	23	21	1e	1c	19	17	14	12	f	d	a	8	5	3	0		**48**	**63**
4480	**1180**	86	85	84	82	81	7f	7e	7c	7b	79	78	76	74	72	71	6f	**70**	**0**	**15**
4496	**1190**	6d	6b	69	68	66	64	62	60	5e	5c	59	57	55	53	51	4f		**16**	**31**
4512	**11A0**	4c	4a	48	46	43	41	3f	3c	3a	37	35	33	30	2e	2b	29		**32**	**47**
4528	**11B0**	26	24	21	1f	1c	1a	17	15	12	f	d	a	8	5	3	0		**48**	**63**
4544	**11C0**	88	87	85	84	83	81	80	7e	7d	7b	79	78	76	74	72	71	**71**	**0**	**15**
4560	**11D0**	6f	6d	6b	69	67	65	63	61	5f	5d	5b	59	56	54	52	50		**16**	**31**
4576	**11E0**	4d	4b	49	47	44	42	3f	3d	3b	38	36	33	31	2e	2c	29		**32**	**47**
4592	**11F0**	27	24	22	1f	1d	1a	17	15	12	10	d	a	8	5	3	0		**48**	**63**
4608	**1200**	8a	89	87	86	84	83	81	80	7e	7d	7b	79	78	76	74	72	**72**	**0**	**15**
4624	**1210**	70	6e	6c	6a	68	66	64	62	60	5e	5c	5a	58	55	53	51		**16**	**31**
4640	**1220**	4f	4c	4a	48	45	43	40	3e	3b	39	37	34	32	2f	2c	2a		**32**	**47**
4656	**1230**	27	25	22	20	1d	1a	18	15	13	10	d	b	8	5	3	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
4672	**1240**	8c	8b	89	88	86	85	83	82	80	7e	7d	7b	79	77	76	74	**73**	**0**	**15**
4688	**1250**	72	70	6e	6c	6a	68	66	64	62	5f	5d	5b	59	57	54	52		**16**	**31**
4704	**1260**	50	4d	4b	49	46	44	41	3f	3c	3a	37	35	32	30	2d	2a		**32**	**47**
4720	**1270**	28	25	23	20	1d	1b	18	15	13	10	d	b	8	5	3	0		**48**	**63**
4736	**1280**	8e	8d	8b	8a	88	87	85	83	82	80	7e	7d	7b	79	77	75	**74**	**0**	**15**
4752	**1290**	73	71	6f	6d	6b	69	67	65	63	61	5f	5c	5a	58	55	53		**16**	**31**
4768	**12A0**	51	4e	4c	4a	47	45	42	40	3d	3b	38	35	33	30	2e	2b		**32**	**47**
4784	**12B0**	28	26	23	20	1e	1b	18	16	13	10	e	b	8	5	3	0		**48**	**63**
4800	**12C0**	90	8e	8d	8b	8a	88	87	85	84	82	80	7e	7d	7b	79	77	**75**	**0**	**15**
4816	**12D0**	75	73	71	6f	6d	6b	69	66	64	62	60	5e	5b	59	57	54		**16**	**31**
4832	**12E0**	52	4f	4d	4b	48	46	43	40	3e	3b	39	36	34	31	2e	2c		**32**	**47**
4848	**12F0**	29	26	24	21	1e	1b	19	16	13	11	e	b	8	6	3	0		**48**	**63**
4864	**1300**	92	90	8f	8d	8c	8a	89	87	85	84	82	80	7e	7c	7a	78	**76**	**0**	**15**
4880	**1310**	76	74	72	70	6e	6c	6a	68	66	63	61	5f	5c	5a	58	55		**16**	**31**
4896	**1320**	53	50	4e	4c	49	46	44	41	3f	3c	3a	37	34	32	2f	2c		**32**	**47**
4912	**1330**	2a	27	24	21	1f	1c	19	16	14	11	e	b	8	6	3	0		**48**	**63**
4928	**1340**	94	92	91	8f	8e	8c	8a	89	87	85	83	82	80	7e	7c	7a	**77**	**0**	**15**
4944	**1350**	78	76	74	72	70	6e	6b	69	67	65	62	60	5e	5b	59	56		**16**	**31**
4960	**1360**	54	52	4f	4d	4a	47	45	42	40	3d	3a	38	35	32	30	2d		**32**	**47**
4976	**1370**	2a	27	25	22	1f	1c	19	17	14	11	e	b	8	6	3	0		**48**	**63**
4992	**1380**	96	94	93	91	8f	8e	8c	8b	89	87	85	83	81	80	7e	7c	**78**	**0**	**15**
5008	**1390**	7a	78	75	73	71	6f	6d	6b	68	66	64	61	5f	5c	5a	58		**16**	**31**
5024	**13A0**	55	53	50	4e	4b	48	46	43	40	3e	3b	38	36	33	30	2d		**32**	**47**
5040	**13B0**	2b	28	25	22	1f	1d	1a	17	14	11	e	b	9	6	3	0		**48**	**63**
5056	**13C0**	97	96	94	93	91	90	8e	8c	8b	89	87	85	83	81	7f	7d	**79**	**0**	**15**
5072	**13D0**	7b	79	77	75	73	70	6e	6c	6a	67	65	63	60	5e	5b	59		**16**	**31**
5088	**13E0**	56	54	51	4e	4c	49	47	44	41	3f	3c	39	36	34	31	2e		**32**	**47**
5104	**13F0**	2b	28	26	23	20	1d	1a	17	14	11	f	c	9	6	3	0		**48**	**63**
5120	**1400**	99	98	96	95	93	92	90	8e	8c	8a	89	87	85	83	81	7f	**80**	**0**	**15**
5136	**1410**	7d	7b	78	76	74	72	70	6d	6b	69	66	64	61	5f	5c	5a		**16**	**31**
5152	**1420**	57	55	52	4f	4d	4a	48	45	42	3f	3d	3a	37	34	31	2f		**32**	**47**
5168	**1430**	2c	29	26	23	20	1d	1a	17	15	12	f	c	9	6	3	0		**48**	**63**
5184	**1440**	9b	9a	98	97	95	93	92	90	8e	8c	8a	88	86	84	82	80	**81**	**0**	**15**
5200	**1450**	7e	7c	7a	78	76	73	71	6f	6c	6a	67	65	63	60	5e	5b		**16**	**31**
5216	**1460**	58	56	53	50	4e	4b	48	46	43	40	3d	3b	38	35	32	2f		**32**	**47**
5232	**1470**	2c	29	26	24	21	1e	1b	18	15	12	f	c	9	6	3	0		**48**	**63**
5248	**1480**	9d	9c	9a	99	97	95	93	92	90	8e	8c	8a	88	86	84	82	**82**	**0**	**15**
5264	**1490**	80	7e	7b	79	77	75	72	70	6e	6b	69	66	64	61	5f	5c		**16**	**31**
5280	**14A0**	59	57	54	51	4f	4c	49	47	44	41	3e	3b	38	36	33	30		**32**	**47**
5296	**14B0**	2d	2a	27	24	21	1e	1b	18	15	12	f	c	9	6	3	0		**48**	**63**
5312	**14C0**	9f	9e	9c	9a	99	97	95	93	92	90	8e	8c	8a	88	86	84	**83**	**0**	**15**
5328	**14D0**	81	7f	7d	7b	78	76	74	71	6f	6d	6a	68	65	62	60	5d		**16**	**31**
5344	**14E0**	5b	58	55	52	50	4d	4a	47	45	42	3f	3c	39	36	33	30		**32**	**47**
5360	**14F0**	2d	2a	27	24	21	1e	1b	18	15	12	f	c	9	6	3	0		**48**	**63**
5376	**1500**	a1	9f	9e	9c	9b	99	97	95	93	91	8f	8d	8b	89	87	85	**84**	**0**	**15**
5392	**1510**	83	81	7e	7c	7a	78	75	73	70	6e	6b	69	66	64	61	5e		**16**	**31**
5408	**1520**	5c	59	56	53	51	4e	4b	48	45	43	40	3d	3a	37	34	31		**32**	**47**
5424	**1530**	2e	2b	28	25	22	1f	1c	19	16	13	f	c	9	6	3	0		**48**	**63**
5440	**1540**	a3	a1	a0	9e	9c	9b	99	97	95	93	91	8f	8d	8b	89	87	**85**	**0**	**15**
5456	**1550**	85	82	80	7e	7b	79	77	74	72	6f	6d	6a	67	65	62	5f		**16**	**31**
5472	**1560**	5d	5a	57	54	52	4f	4c	49	46	43	40	3d	3a	37	34	31		**32**	**47**
5488	**1570**	2e	2b	28	25	22	1f	1c	19	16	13	10	d	9	6	3	0		**48**	**63**
5504	**1580**	a5	a3	a2	a0	9e	9c	9b	99	97	95	93	91	8f	8d	8a	88	**86**	**0**	**15**
5520	**1590**	86	84	82	7f	7d	7a	78	75	73	70	6e	6b	69	66	63	61		**16**	**31**
5536	**15A0**	5e	5b	58	55	53	50	4d	4a	47	44	41	3e	3b	38	35	32		**32**	**47**
5552	**15B0**	2f	2c	29	26	23	20	1c	19	16	13	10	d	9	6	3	0		**48**	**63**
5568	**15C0**	a7	a5	a4	a2	a0	9e	9c	9b	99	97	95	93	90	8e	8c	8a	**87**	**0**	**15**
5584	**15D0**	88	85	83	81	7e	7c	79	77	74	72	6f	6d	6a	67	64	62		**16**	**31**
5600	**15E0**	5f	5c	59	56	54	51	4e	4b	48	45	42	3f	3c	39	36	33		**32**	**47**
5616	**15F0**	30	2c	29	26	23	20	1d	1a	16	13	10	d	a	6	3	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
5632	**1600**	a9	a7	a5	a4	a2	a0	9e	9c	9a	98	96	94	92	90	8e	8b	**88**	**0**	**15**
5648	**1610**	89	87	85	82	80	7d	7b	78	76	73	70	6e	6b	68	66	63		**16**	**31**
5664	**1620**	60	5d	5a	57	55	52	4f	4c	49	46	43	40	3d	39	36	33		**32**	**47**
5680	**1630**	30	2d	2a	27	23	20	1d	1a	17	13	10	d	a	6	3	0		**48**	**63**
5696	**1640**	ab	a9	a7	a6	a4	a2	a0	9e	9c	9a	98	96	94	92	8f	8d	**89**	**0**	**15**
5712	**1650**	8b	88	86	84	81	7f	7c	7a	77	74	72	6f	6c	6a	67	64		**16**	**31**
5728	**1660**	61	5e	5b	58	55	53	50	4d	4a	46	43	40	3d	3a	37	34		**32**	**47**
5744	**1670**	31	2d	2a	27	24	21	1d	1a	17	14	10	d	a	7	3	0		**48**	**63**
5760	**1680**	ad	ab	a9	a7	a6	a4	a2	a0	9e	9c	9a	98	95	93	91	8f	**90**	**0**	**15**
5776	**1690**	8c	8a	88	85	83	80	7e	7b	78	76	73	70	6d	6b	68	65		**16**	**31**
5792	**16A0**	62	5f	5c	59	56	53	50	4d	4a	47	44	41	3e	3b	38	34		**32**	**47**
5808	**16B0**	31	2e	2b	27	24	21	1e	1a	17	14	11	d	a	7	3	0		**48**	**63**
5824	**16C0**	ae	ad	ab	a9	a7	a6	a4	a2	a0	9e	9b	99	97	95	93	90	**91**	**0**	**15**
5840	**16D0**	8e	8b	89	87	84	82	7f	7c	7a	77	74	71	6f	6c	69	66		**16**	**31**
5856	**16E0**	63	60	5d	5a	57	54	51	4e	4b	48	45	42	3f	3b	38	35		**32**	**47**
5872	**16F0**	32	2e	2b	28	25	21	1e	1b	17	14	11	d	a	7	3	0		**48**	**63**
5888	**1700**	b0	af	ad	ab	a9	a7	a5	a3	a1	9f	9d	9b	99	96	94	92	**92**	**0**	**15**
5904	**1710**	8f	8d	8b	88	86	83	80	7e	7b	78	76	73	70	6d	6a	67		**16**	**31**
5920	**1720**	64	61	5e	5b	58	55	52	4f	4c	49	46	42	3f	3c	39	36		**32**	**47**
5936	**1730**	32	2f	2c	28	25	22	1e	1b	18	14	11	e	a	7	3	0		**48**	**63**
5952	**1740**	b2	b1	af	ad	ab	a9	a7	a5	a3	a1	9f	9d	9a	98	96	93	**93**	**0**	**15**
5968	**1750**	91	8f	8c	8a	87	84	82	7f	7c	7a	77	74	71	6e	6b	68		**16**	**31**
5984	**1760**	65	62	5f	5c	59	56	53	50	4d	4a	46	43	40	3d	39	36		**32**	**47**
6000	**1770**	33	2f	2c	29	25	22	1f	1b	18	15	11	e	a	7	3	0		**48**	**63**
6016	**1780**	b4	b2	b1	af	ad	ab	a9	a7	a5	a3	a1	9e	9c	9a	97	95	**94**	**0**	**15**
6032	**1790**	93	90	8e	8b	88	86	83	80	7e	7b	78	75	72	6f	6d	6a		**16**	**31**
6048	**17A0**	67	64	60	5d	5a	57	54	51	4e	4a	47	44	41	3d	3a	37		**32**	**47**
6064	**17B0**	33	30	2d	29	26	22	1f	1c	18	15	11	e	a	7	3	0		**48**	**63**
6080	**17C0**	b6	b4	b3	b1	af	ad	ab	a9	a7	a4	a2	a0	9e	9b	99	97	**95**	**0**	**15**
6096	**17D0**	94	92	8f	8c	8a	87	85	82	7f	7c	79	76	74	71	6e	6b		**16**	**31**
6112	**17E0**	68	65	62	5e	5b	58	55	52	4e	4b	48	45	41	3e	3b	37		**32**	**47**
6128	**17F0**	34	31	2d	2a	26	23	1f	1c	18	15	11	e	a	7	3	0		**48**	**63**
6144	**1800**	b8	b6	b4	b3	b1	af	ad	aa	a8	a6	a4	a2	9f	9d	9b	98	**96**	**0**	**15**
6160	**1810**	96	93	91	8e	8b	89	86	83	80	7e	7b	78	75	72	6f	6c		**16**	**31**
6176	**1820**	69	66	63	5f	5c	59	56	53	4f	4c	49	45	42	3f	3b	38		**32**	**47**
6192	**1830**	34	31	2e	2a	27	23	20	1c	19	15	12	e	b	7	4	0		**48**	**63**
6208	**1840**	ba	b8	b6	b4	b2	b0	ae	ac	aa	a8	a6	a3	a1	9f	9c	9a	**97**	**0**	**15**
6224	**1850**	97	95	92	8f	8d	8a	87	84	82	7f	7c	79	76	73	70	6d		**16**	**31**
6240	**1860**	6a	67	64	60	5d	5a	57	53	50	4d	49	46	43	3f	3c	38		**32**	**47**
6256	**1870**	35	32	2e	2b	27	24	20	1c	19	15	12	e	b	7	4	0		**48**	**63**
6272	**1880**	bc	ba	b8	b6	b4	b2	b0	ae	ac	aa	a7	a5	a3	a0	9e	9b	**98**	**0**	**15**
6288	**1890**	99	96	94	91	8e	8b	89	86	83	80	7d	7a	77	74	71	6e		**16**	**31**
6304	**18A0**	6b	68	65	61	5e	5b	58	54	51	4e	4a	47	43	40	3c	39		**32**	**47**
6320	**18B0**	36	32	2f	2b	27	24	20	1d	19	16	12	e	b	7	4	0		**48**	**63**
6336	**18C0**	be	bc	ba	b8	b6	b4	b2	b0	ae	ab	a9	a7	a4	a2	9f	9d	**99**	**0**	**15**
6352	**18D0**	9a	98	95	92	90	8d	8a	87	84	81	7e	7b	78	75	72	6f		**16**	**31**
6368	**18E0**	6c	69	66	62	5f	5c	58	55	52	4e	4b	48	44	41	3d	3a		**32**	**47**
6384	**18F0**	36	33	2f	2b	28	24	21	1d	19	16	12	f	b	7	4	0		**48**	**63**
6400	**1900**	c0	be	bc	ba	b8	b6	b4	b2	af	ad	ab	a8	a6	a4	a1	9e	**100**	**0**	**15**
6416	**1910**	9c	99	97	94	91	8e	8b	89	86	83	80	7d	7a	77	73	70		**16**	**31**
6432	**1920**	6d	6a	67	63	60	5d	59	56	53	4f	4c	48	45	41	3e	3a		**32**	**47**
6448	**1930**	37	33	2f	2c	28	25	21	1d	1a	16	12	f	b	7	4	0		**48**	**63**
6464	**1940**	c2	c0	be	bc	ba	b8	b6	b3	b1	af	ac	aa	a8	a5	a3	a0	**101**	**0**	**15**
6480	**1950**	9d	9b	98	95	93	90	8d	8a	87	84	81	7e	7b	78	75	71		**16**	**31**
6496	**1960**	6e	6b	68	64	61	5e	5a	57	53	50	4c	49	45	42	3e	3b		**32**	**47**
6512	**1970**	37	34	30	2c	29	25	21	1e	1a	16	13	f	b	7	4	0		**48**	**63**
6528	**1980**	c4	c2	c0	be	bc	ba	b7	b5	b3	b1	ae	ac	a9	a7	a4	a2	**102**	**0**	**15**
6544	**1990**	9f	9c	9a	97	94	91	8e	8b	88	85	82	7f	7c	79	76	73		**16**	**31**
6560	**19A0**	6f	6c	69	65	62	5f	5b	58	54	51	4d	4a	46	43	3f	3b		**32**	**47**
6576	**19B0**	38	34	30	2d	29	25	22	1e	1a	16	13	f	b	8	4	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
6592	**19C0**	c5	c4	c2	c0	bd	bb	b9	b7	b5	b2	b0	ad	ab	a8	a6	a3	**103**	**0**	**15**
6608	**19D0**	a1	9e	9b	98	95	93	90	8d	8a	87	84	80	7d	7a	77	74		**16**	**31**
6624	**19E0**	70	6d	6a	66	63	60	5c	59	55	52	4e	4a	47	43	40	3c		**32**	**47**
6640	**19F0**	38	35	31	2d	29	26	22	1e	1a	17	13	f	b	8	4	0		**48**	**63**
6656	**1A00**	c7	c5	c3	c1	bf	bd	bb	b9	b6	b4	b2	af	ad	aa	a7	a5	**104**	**0**	**15**
6672	**1A10**	a2	9f	9d	9a	97	94	91	8e	8b	88	85	82	7f	7b	78	75		**16**	**31**
6688	**1A20**	71	6e	6b	67	64	60	5d	59	56	52	4f	4b	48	44	40	3d		**32**	**47**
6704	**1A30**	39	35	31	2e	2a	26	22	1f	1b	17	13	f	b	8	4	0		**48**	**63**
6720	**1A40**	c9	c7	c5	c3	c1	bf	bd	ba	b8	b6	b3	b1	ae	ac	a9	a6	**105**	**0**	**15**
6736	**1A50**	a4	a1	9e	9b	98	95	92	8f	8c	89	86	83	80	7c	79	76		**16**	**31**
6752	**1A60**	73	6f	6c	68	65	61	5e	5a	57	53	50	4c	48	45	41	3d		**32**	**47**
6768	**1A70**	39	36	32	2e	2a	26	23	1f	1b	17	13	f	c	8	4	0		**48**	**63**
6784	**1A80**	cb	c9	c7	c5	c3	c1	bf	bc	ba	b7	b5	b3	b0	ad	ab	a8	**106**	**0**	**15**
6800	**1A90**	a5	a2	a0	9d	9a	97	94	91	8e	8b	87	84	81	7e	7a	77		**16**	**31**
6816	**1AA0**	74	70	6d	69	66	62	5f	5b	58	54	50	4d	49	45	41	3e		**32**	**47**
6832	**1AB0**	3a	36	32	2f	2b	27	23	1f	1b	17	13	10	c	8	4	0		**48**	**63**
6848	**1AC0**	cd	cb	c9	c7	c5	c3	c0	be	bc	b9	b7	b4	b2	af	ac	aa	**107**	**0**	**15**
6864	**1AD0**	a7	a4	a1	9e	9b	98	95	92	8f	8c	89	85	82	7f	7c	78		**16**	**31**
6880	**1AE0**	75	71	6e	6a	67	63	60	5c	58	55	51	4d	4a	46	42	3e		**32**	**47**
6896	**1AF0**	3a	37	33	2f	2b	27	23	1f	1c	18	14	10	c	8	4	0		**48**	**63**
6912	**1B00**	cf	cd	cb	c9	c7	c4	c2	c0	bd	bb	b8	b6	b3	b1	ae	ab	**108**	**0**	**15**
6928	**1B10**	a8	a6	a3	a0	9d	9a	97	94	90	8d	8a	87	83	80	7d	79		**16**	**31**
6944	**1B20**	76	72	6f	6b	68	64	61	5d	59	56	52	4e	4a	46	43	3f		**32**	**47**
6960	**1B30**	3b	37	33	2f	2b	28	24	20	1c	18	14	10	c	8	4	0		**48**	**63**
6976	**1B40**	d1	cf	cd	cb	c9	c6	c4	c2	bf	bd	ba	b8	b5	b2	b0	ad	**109**	**0**	**15**
6992	**1B50**	aa	a7	a4	a1	9e	9b	98	95	92	8e	8b	88	85	81	7e	7a		**16**	**31**
7008	**1B60**	77	73	70	6c	69	65	61	5e	5a	56	53	4f	4b	47	43	3f		**32**	**47**
7024	**1B70**	3c	38	34	30	2c	28	24	20	1c	18	14	10	c	8	4	0		**48**	**63**
7040	**1B80**	d3	d1	cf	cd	ca	c8	c6	c3	c1	be	bc	b9	b7	b4	b1	ae	**110**	**0**	**15**
7056	**1B90**	ab	a9	a6	a3	a0	9d	99	96	93	90	8d	89	86	82	7f	7c		**16**	**31**
7072	**1BA0**	78	74	71	6d	6a	66	62	5f	5b	57	53	4f	4c	48	44	40		**32**	**47**
7088	**1BB0**	3c	38	34	30	2c	28	24	20	1c	18	14	10	c	8	4	0		**48**	**63**
7104	**1BC0**	d5	d3	d1	ce	cc	ca	c8	c5	c3	c0	be	bb	b8	b6	b3	b0	**111**	**0**	**15**
7120	**1BD0**	ad	aa	a7	a4	a1	9e	9b	98	94	91	8e	8a	87	84	80	7d		**16**	**31**
7136	**1BE0**	79	76	72	6e	6b	67	63	5f	5c	58	54	50	4c	48	45	41		**32**	**47**
7152	**1BF0**	3d	39	35	31	2d	29	25	21	1d	18	14	10	c	8	4	0		**48**	**63**
7168	**1C00**	d7	d5	d3	d0	ce	cc	c9	c7	c4	c2	bf	bd	ba	b7	b4	b2	**112**	**0**	**15**
7184	**1C10**	af	ac	a9	a6	a3	9f	9c	99	96	92	8f	8c	88	85	81	7e		**16**	**31**
7200	**1C20**	7a	77	73	6f	6c	68	64	60	5d	59	55	51	4d	49	45	41		**32**	**47**
7216	**1C30**	3d	39	35	31	2d	29	25	21	1d	19	15	10	c	8	4	0		**48**	**63**
7232	**1C40**	d9	d7	d4	d2	d0	ce	cb	c9	c6	c4	c1	be	bc	b9	b6	b3	**113**	**0**	**15**
7248	**1C50**	b0	ad	aa	a7	a4	a1	9e	9a	97	94	90	8d	89	86	82	7f		**16**	**31**
7264	**1C60**	7b	78	74	70	6d	69	65	61	5d	59	56	52	4e	4a	46	42		**32**	**47**
7280	**1C70**	3e	3a	36	32	2d	29	25	21	1d	19	15	11	c	8	4	0		**48**	**63**
7296	**1C80**	db	d8	d6	d4	d2	cf	cd	ca	c8	c5	c3	c0	bd	ba	b8	b5	**114**	**0**	**15**
7312	**1C90**	b2	af	ac	a9	a5	a2	9f	9c	98	95	92	8e	8b	87	84	80		**16**	**31**
7328	**1CA0**	7c	79	75	71	6e	6a	66	62	5e	5a	56	52	4e	4a	46	42		**32**	**47**
7344	**1CB0**	3e	3a	36	32	2e	2a	26	21	1d	19	15	11	d	8	4	0		**48**	**63**
7360	**1CC0**	dc	da	d8	d6	d4	d1	cf	cc	ca	c7	c4	c2	bf	bc	b9	b6	**115**	**0**	**15**
7376	**1CD0**	b3	b0	ad	aa	a7	a4	a0	9d	9a	96	93	8f	8c	88	85	81		**16**	**31**
7392	**1CE0**	7d	7a	76	72	6e	6b	67	63	5f	5b	57	53	4f	4b	47	43		**32**	**47**
7408	**1CF0**	3f	3b	37	32	2e	2a	26	22	1e	19	15	11	d	8	4	0		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
7424	**1D00**	df	dd	da	d8	d6	d3	d1	ce	cc	c9	c6	c4	c1	be	bb	b8	**116**	**0**	**15**
7440	**1D10**	b5	b2	af	ac	a9	a5	a2	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7456	**1D20**	7e	7a	76	73	6f	6b	68	64	60	5c	58	54	50	4c	48	44		**32**	**47**
7472	**1D30**	3f	3b	37	33	2f	2b	26	22	1e	1a	15	11	d	9	4	0		**48**	**63**
7488	**1D40**	e1	df	dd	db	d8	d6	d3	d1	ce	cb	c9	c6	c3	c0	bd	ba	**117**	**0**	**15**
7504	**1D50**	b7	b4	b1	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7520	**1D60**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	44		**32**	**47**
7536	**1D70**	40	3c	38	34	2f	2b	27	23	1e	1a	16	11	d	9	4	0		**48**	**63**
7552	**1D80**	e5	e2	e0	de	db	d9	d6	d4	d1	ce	cc	c9	c6	c3	c0	bd	**118**	**0**	**15**
7568	**1D90**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7584	**1DA0**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7600	**1DB0**	41	3d	39	34	30	2c	27	23	1f	1a	16	12	d	9	4	0		**48**	**63**
7616	**1DC0**	e9	e6	e4	e2	df	dd	da	d7	d5	d2	cf	cc	c9	c6	c2	be	**119**	**0**	**15**
7632	**1DD0**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7648	**1DE0**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7664	**1DF0**	41	3d	39	35	31	2c	28	24	1f	1b	16	12	d	9	4	0		**48**	**63**
7680	**1E00**	ee	ec	e9	e7	e4	e2	df	dc	da	d6	d2	ce	ca	c6	c2	be	**120**	**0**	**15**
7696	**1E10**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7712	**1E20**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7728	**1E30**	41	3d	39	35	31	2d	29	24	20	1b	17	12	e	9	5	0		**48**	**63**
7744	**1E40**	f7	f4	f2	ef	ec	e8	e3	df	da	d6	d2	ce	ca	c6	c2	be	**121**	**0**	**15**
7760	**1E50**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7776	**1E60**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7792	**1E70**	41	3d	39	35	31	2d	29	25	20	1c	17	13	e	9	5	0		**48**	**63**
7808	**1E80**	ff	ff	ff	f1	ec	e8	e3	df	da	d6	d2	ce	ca	c6	c2	be	**122**	**0**	**15**
7824	**1E90**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7840	**1EA0**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7856	**1EB0**	41	3d	39	35	31	2d	29	25	20	1c	17	13	e	0	0	0		**48**	**63**
7872	**1EC0**	ff	ff	ff	ff	ff	ff	e3	df	da	d6	d2	ce	ca	c6	c2	be	**123**	**0**	**15**
7888	**1ED0**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7904	**1EE0**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7920	**1EF0**	41	3d	39	35	31	2d	29	25	20	1c	0	0	0	0	0	0		**48**	**63**
7936	**1F00**	ff	ff	ff	ff	ff	ff	ff	ff	ff	d6	d2	ce	ca	c6	c2	be	**124**	**0**	**15**
7952	**1F10**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
7968	**1F20**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
7984	**1F30**	41	3d	39	35	31	2d	29	0	0	0	0	0	0	0	0	0		**48**	**63**
8000	**1F40**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	c6	c2	be	**125**	**0**	**15**
8016	**1F50**	ba	b6	b2	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
8032	**1F60**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	4d	49	45		**32**	**47**
8048	**1F70**	41	3d	39	0	0	0	0	0	0	0	0	0	0	0	0	0		**48**	**63**
8064	**1F80**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	**126**	**0**	**15**
8080	**1F90**	ff	ff	ff	ae	aa	a6	a3	9f	9b	97	94	90	8c	89	85	81		**16**	**31**
8096	**1FA0**	7e	7a	76	73	6f	6b	68	64	60	5c	59	55	51	0	0	0		**32**	**47**
8112	**1FB0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**48**	**63**
8128	**1FC0**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	**127**	**0**	**15**
8144	**1FD0**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff		**16**	**31**
8160	**1FE0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**32**	**47**
8176	**1FF0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**48**	**63**

Referencias:

Six-Step
Modo II
Modo I
Modo Lineal

;*** Datos de tiempos tb a grabar en la **EPROM tb** ***
;*** (son los tiempos que debe contar cada Contador 74LS193) ***

;*** Parametros utilizados para generar el archivo: ***
;*** Bit para cuantificar el tiempo: 8 ***
;*** Frecuencia del clock en Hz: 8375000 ***
;*** Bit para m: 7 ***
;*** Bit para alfa: 6 ***

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	Hex																
0	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	**10**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	**20**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	**30**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64	**40**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
80	**50**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
96	**60**	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2
112	**70**	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
128	**80**	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
144	**90**	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
160	**A0**	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3
176	**B0**	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4
192	**C0**	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2
208	**D0**	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
224	**E0**	3	3	4	4	4	4	4	4	4	4	4	4	4	5	5	5
240	**F0**	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6
256	**100**	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2
272	**110**	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4
288	**120**	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6
304	**130**	6	6	7	7	7	7	7	7	7	7	7	7	7	8	8	8
320	**140**	0	0	0	1	1	1	1	1	1	2	2	2	2	2	3	3
336	**150**	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5
352	**160**	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8
368	**170**	8	8	8	8	8	9	9	9	9	9	9	9	9	9	9	a
384	**180**	0	0	0	1	1	1	1	2	2	2	2	2	3	3	3	3
400	**190**	3	4	4	4	4	5	5	5	5	5	6	6	6	6	6	7
416	**1A0**	7	7	7	7	7	8	8	8	8	8	9	9	9	9	9	9
432	**1B0**	a	a	a	a	a	a	a	b	b	b	b	b	b	b	b	c
448	**1C0**	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4
464	**1D0**	4	4	5	5	5	5	6	6	6	6	6	7	7	7	7	8
480	**1E0**	8	8	8	9	9	9	9	9	a	a	a	a	a	b	b	b
496	**1F0**	b	b	b	c	c	c	c	c	c	d	d	d	d	d	d	d
512	**200**	0	0	1	1	1	1	2	2	2	3	3	3	4	4	4	4
528	**210**	5	5	5	6	6	6	6	7	7	7	7	8	8	8	8	9
544	**220**	9	9	9	a	a	a	a	b	b	b	b	c	c	c	c	c
560	**230**	d	d	d	d	d	e	e	e	e	e	f	f	f	f	f	f
576	**240**	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5
592	**250**	5	6	6	6	7	7	7	7	8	8	8	9	9	9	a	a
608	**260**	a	a	b	b	b	b	c	c	c	d	d	d	d	e	e	e
624	**270**	e	e	f	f	f	f	10	10	10	10	10	11	11	11	11	11
640	**280**	0	0	1	1	1	2	2	3	3	3	4	4	4	5	5	5
656	**290**	6	6	7	7	7	8	8	8	9	9	9	a	a	a	b	b
672	**2A0**	b	c	c	c	c	d	d	d	e	e	e	f	f	f	f	10
688	**2B0**	10	10	10	11	11	11	11	12	12	12	12	12	13	13	13	13
704	**2C0**	0	0	1	1	2	2	2	3	3	4	4	4	5	5	6	6
720	**2D0**	6	7	7	8	8	8	9	9	9	a	a	b	b	b	c	c
736	**2E0**	c	d	d	d	e	e	e	f	f	f	10	10	10	11	11	11
752	**2F0**	11	12	12	12	13	13	13	13	14	14	14	14	14	15	15	15
768	**300**	0	0	1	1	2	2	3	3	4	4	4	5	5	6	6	7
784	**310**	7	7	8	8	9	9	a	a	a	b	b	c	c	c	d	d
800	**320**	d	e	e	f	f	f	10	10	10	11	11	11	12	12	12	13
816	**330**	13	13	14	14	14	14	15	15	15	16	16	16	16	17	17	17

Módulo Cuantizado	Angulo Cuantizado	
0	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
1	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
2	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
3	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
4	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
5	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
6	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
7	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
8	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
9	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
10	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
11	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
12	**0**	**15**
	16	**31**
	32	**47**
	48	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
832	**340**	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	**13**	**0**	**15**
848	**350**	8	8	8	9	9	a	a	b	b	c	c	c	d	d	e	e		**16**	**31**
864	**360**	f	f	f	10	10	11	11	11	12	12	13	13	13	14	14	14		**32**	**47**
880	**370**	15	15	15	16	16	16	17	17	17	17	18	18	18	18	19	19		**48**	**63**
896	**380**	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	**14**	**0**	**15**
912	**390**	8	9	9	a	a	b	b	c	c	d	d	d	e	e	f	f		**16**	**31**
928	**3A0**	10	10	11	11	11	12	12	13	13	14	14	14	15	15	15	16		**32**	**47**
944	**3B0**	16	17	17	17	18	18	18	19	19	19	19	1a	1a	1a	1b	1b		**48**	**63**
960	**3C0**	0	1	1	2	2	3	3	4	4	5	5	6	7	7	8	8	**15**	**0**	**15**
976	**3D0**	9	9	a	a	b	b	c	c	d	d	e	e	f	f	10	10		**16**	**31**
992	**3E0**	11	11	12	12	13	13	14	14	14	15	15	16	16	17	17	17		**32**	**47**
1008	**3F0**	18	18	19	19	19	1a	1a	1a	1b	1b	1b	1c	1c	1c	1c	1d		**48**	**63**
1024	**400**	0	1	1	2	2	3	4	4	5	5	6	6	7	8	8	9	**16**	**0**	**15**
1040	**410**	9	a	a	b	c	c	d	d	e	e	f	f	10	10	11	11		**16**	**31**
1056	**420**	12	12	13	13	14	14	15	15	16	16	17	17	18	18	19	19		**32**	**47**
1072	**430**	19	1a	1a	1b	1b	1b	1c	1c	1c	1d	1d	1d	1e	1e	1e	1f		**48**	**63**
1088	**440**	0	1	1	2	3	3	4	4	5	6	6	7	7	8	9	9	**17**	**0**	**15**
1104	**450**	a	a	b	c	c	d	d	e	f	f	10	10	11	11	12	13		**16**	**31**
1120	**460**	13	14	14	15	15	16	16	17	17	18	18	19	19	1a	1a	1b		**32**	**47**
1136	**470**	1b	1b	1c	1c	1d	1d	1d	1e	1e	1f	1f	1f	20	20	20	21		**48**	**63**
1152	**480**	0	1	1	2	3	3	4	5	5	6	7	7	8	9	9	a	**18**	**0**	**15**
1168	**490**	a	b	c	c	d	e	e	f	f	10	11	11	12	12	13	14		**16**	**31**
1184	**4A0**	14	15	15	16	16	17	18	18	19	19	1a	1a	1b	1b	1c	1c		**32**	**47**
1200	**4B0**	1d	1d	1d	1e	1e	1f	1f	20	20	20	21	21	21	22	22	23		**48**	**63**
1216	**4C0**	0	1	1	2	3	3	4	5	6	6	7	8	8	9	a	a	**19**	**0**	**15**
1232	**4D0**	b	c	c	d	e	e	f	10	10	11	12	12	13	14	14	15		**16**	**31**
1248	**4E0**	15	16	17	17	18	18	19	19	1a	1b	1b	1c	1c	1d	1d	1e		**32**	**47**
1264	**4F0**	1e	1f	1f	20	20	20	21	21	22	22	23	23	23	24	24	24		**48**	**63**
1280	**500**	0	1	1	2	3	4	4	5	6	7	7	8	9	9	a	b	**20**	**0**	**15**
1296	**510**	c	c	d	e	e	f	10	11	11	12	13	13	14	15	15	16		**16**	**31**
1312	**520**	16	17	18	18	19	1a	1a	1b	1b	1c	1c	1d	1e	1e	1f	1f		**32**	**47**
1328	**530**	20	20	21	21	22	22	23	23	24	24	24	25	25	26	26	26		**48**	**63**
1344	**540**	0	1	2	2	3	4	5	5	6	7	8	8	9	a	b	b	**21**	**0**	**15**
1360	**550**	c	d	e	e	f	10	11	11	12	13	13	14	15	16	16	17		**16**	**31**
1376	**560**	18	18	19	1a	1a	1b	1b	1c	1d	1d	1e	1e	1f	20	20	21		**32**	**47**
1392	**570**	21	22	22	23	23	24	24	25	25	26	26	27	27	27	28	28		**48**	**63**
1408	**580**	0	1	2	2	3	4	5	6	6	7	8	9	a	a	b	c	**22**	**0**	**15**
1424	**590**	d	e	e	f	10	11	11	12	13	14	14	15	16	17	17	18		**16**	**31**
1440	**5A0**	19	19	1a	1b	1b	1c	1d	1d	1e	1f	1f	20	21	21	22	22		**32**	**47**
1456	**5B0**	23	23	24	25	25	26	26	27	27	28	28	28	29	29	2a	2a		**48**	**63**
1472	**5C0**	0	1	2	3	3	4	5	6	7	8	8	9	a	b	c	d	**23**	**0**	**15**
1488	**5D0**	d	e	f	10	11	11	12	13	14	15	15	16	17	18	18	19		**16**	**31**
1504	**5E0**	1a	1b	1b	1c	1d	1d	1e	1f	1f	20	21	21	22	23	23	24		**32**	**47**
1520	**5F0**	24	25	26	26	27	27	28	28	29	29	2a	2a	2b	2b	2c	2c		**48**	**63**
1536	**600**	0	1	2	3	4	4	5	6	7	8	9	a	b	b	c	d	**24**	**0**	**15**
1552	**610**	e	f	10	11	11	12	13	14	15	15	16	17	18	19	19	1a		**16**	**31**
1568	**620**	1b	1c	1c	1d	1e	1f	1f	20	21	21	22	23	23	24	25	25		**32**	**47**
1584	**630**	26	27	27	28	28	29	2a	2a	2b	2b	2c	2c	2d	2d	2e	2e		**48**	**63**
1600	**640**	0	1	2	3	4	5	6	6	7	8	9	a	b	c	d	e	**25**	**0**	**15**
1616	**650**	f	f	10	11	12	13	14	15	15	16	17	18	19	1a	1a	1b		**16**	**31**
1632	**660**	1c	1d	1e	1e	1f	20	21	21	22	23	24	24	25	26	26	27		**32**	**47**
1648	**670**	28	28	29	2a	2a	2b	2b	2c	2c	2d	2d	2e	2e	2f	2f	30		**48**	**63**
1664	**680**	0	1	2	3	4	5	6	7	8	9	a	a	b	c	d	e	**26**	**0**	**15**
1680	**690**	f	10	11	12	13	14	15	15	16	17	18	19	1a	1b	1c	1c		**16**	**31**
1696	**6A0**	1d	1e	1f	20	20	21	22	23	24	24	25	26	26	27	28	29		**32**	**47**
1712	**6B0**	29	2a	2b	2b	2c	2c	2d	2e	2e	2f	2f	30	30	31	31	32		**48**	**63**
1728	**6C0**	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	**27**	**0**	**15**
1744	**6D0**	10	11	12	13	14	14	15	16	17	18	19	1a	1b	1c	1d	1d		**16**	**31**
1760	**6E0**	1e	1f	20	21	22	22	23	24	25	26	26	27	28	29	29	2a		**32**	**47**
1776	**6F0**	2b	2b	2c	2d	2d	2e	2f	2f	30	31	31	32	32	33	33	34		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
1792	**700**	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	**28**	**0**	**15**
1808	**710**	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f		**16**	**31**
1824	**720**	1f	20	21	22	23	24	25	25	26	27	28	29	29	2a	2b	2c		**32**	**47**
1840	**730**	2c	2d	2e	2e	2f	30	30	31	32	32	33	34	34	35	35	36		**48**	**63**
1856	**740**	0	1	2	3	4	5	6	7	9	a	b	c	d	e	f	10	**29**	**0**	**15**
1872	**750**	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f	20		**16**	**31**
1888	**760**	21	21	22	23	24	25	26	27	28	28	29	2a	2b	2c	2c	2d		**32**	**47**
1904	**770**	2e	2f	2f	30	31	32	32	33	34	34	35	35	36	37	37	38		**48**	**63**
1920	**780**	0	1	2	3	4	6	7	8	9	a	b	c	d	e	f	10	**30**	**0**	**15**
1936	**790**	11	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f	20	21		**16**	**31**
1952	**7A0**	22	23	24	24	25	26	27	28	29	2a	2b	2c	2c	2d	2e	2f		**32**	**47**
1968	**7B0**	30	30	31	32	33	33	34	35	35	36	37	37	38	38	39	3a		**48**	**63**
1984	**7C0**	0	1	2	3	5	6	7	8	9	a	b	c	e	f	10	11	**31**	**0**	**15**
2000	**7D0**	12	13	14	15	16	17	19	1a	1b	1c	1d	1e	1f	20	21	22		**16**	**31**
2016	**7E0**	23	24	25	26	27	28	29	29	2a	2b	2c	2d	2e	2f	30	30		**32**	**47**
2032	**7F0**	31	32	33	33	34	35	36	36	37	38	38	39	3a	3a	3b	3b		**48**	**63**
2048	**800**	0	1	2	4	5	6	7	8	9	b	c	d	e	f	10	11	**32**	**0**	**15**
2064	**810**	13	14	15	16	17	18	19	1a	1c	1d	1e	1f	20	21	22	23		**16**	**31**
2080	**820**	24	25	26	27	28	29	2a	2b	2c	2d	2e	2e	2f	30	31	32		**32**	**47**
2096	**830**	33	34	34	35	36	37	37	38	39	3a	3a	3b	3c	3c	3d	3d		**48**	**63**
2112	**840**	0	1	2	4	5	6	7	8	a	b	c	d	e	10	11	12	**33**	**0**	**15**
2128	**850**	13	14	16	17	18	19	1a	1b	1c	1d	1f	20	21	22	23	24		**16**	**31**
2144	**860**	25	26	27	28	29	2a	2b	2c	2d	2e	2f	30	31	32	33	33		**32**	**47**
2160	**870**	34	35	36	37	38	38	39	3a	3b	3b	3c	3d	3d	3e	3f	3f		**48**	**63**
2176	**880**	0	1	3	4	5	6	7	9	a	b	c	e	f	10	11	13	**34**	**0**	**15**
2192	**890**	14	15	16	17	19	1a	1b	1c	1d	1e	20	21	22	23	24	25		**16**	**31**
2208	**8A0**	26	27	28	29	2a	2b	2c	2d	2e	2f	30	31	32	33	34	35		**32**	**47**
2224	**8B0**	36	37	38	38	39	3a	3b	3c	3c	3d	3e	3f	3f	40	41	41		**48**	**63**
2240	**8C0**	0	1	3	4	5	6	8	9	a	c	d	e	f	11	12	13	**35**	**0**	**15**
2256	**8D0**	14	16	17	18	19	1b	1c	1d	1e	1f	20	22	23	24	25	26		**16**	**31**
2272	**8E0**	27	28	29	2b	2c	2d	2e	2f	30	31	32	33	34	35	36	37		**32**	**47**
2288	**8F0**	37	38	39	3a	3b	3c	3d	3d	3e	3f	40	40	41	42	42	43		**48**	**63**
2304	**900**	0	1	3	4	5	7	8	9	b	c	d	e	10	11	12	14	**36**	**0**	**15**
2320	**910**	15	16	17	19	1a	1b	1d	1e	1f	20	21	23	24	25	26	27		**16**	**31**
2336	**920**	28	2a	2b	2c	2d	2e	2f	30	31	32	33	34	35	36	37	38		**32**	**47**
2352	**930**	39	3a	3b	3c	3d	3d	3e	3f	40	41	41	42	43	44	44	45		**48**	**63**
2368	**940**	0	1	3	4	5	7	8	a	b	c	e	f	10	12	13	14	**37**	**0**	**15**
2384	**950**	16	17	18	19	1b	1c	1d	1f	20	21	22	24	25	26	27	28		**16**	**31**
2400	**960**	2a	2b	2c	2d	2e	2f	30	31	33	34	35	36	37	38	39	3a		**32**	**47**
2416	**970**	3b	3c	3d	3d	3e	3f	40	41	42	43	43	44	45	46	46	47		**48**	**63**
2432	**980**	0	1	3	4	6	7	8	a	b	d	e	f	11	12	13	15	**38**	**0**	**15**
2448	**990**	16	17	19	1a	1b	1d	1e	1f	21	22	23	25	26	27	28	29		**16**	**31**
2464	**9A0**	2b	2c	2d	2e	2f	31	32	33	34	35	36	37	38	39	3a	3b		**32**	**47**
2480	**9B0**	3c	3d	3e	3f	40	41	42	43	43	44	45	46	47	47	48	49		**48**	**63**
2496	**9C0**	0	1	3	4	6	7	9	a	b	d	e	10	11	13	14	15	**39**	**0**	**15**
2512	**9D0**	17	18	19	1b	1c	1e	1f	20	22	23	24	25	27	28	29	2b		**16**	**31**
2528	**9E0**	2c	2d	2e	2f	31	32	33	34	35	36	38	39	3a	3b	3c	3d		**32**	**47**
2544	**9F0**	3e	3f	40	41	42	43	44	44	45	46	47	48	49	49	4a	4b		**48**	**63**
2560	**A00**	0	1	3	4	6	7	9	a	c	d	f	10	12	13	14	16	**40**	**0**	**15**
2576	**A10**	17	19	1a	1c	1d	1e	20	21	22	24	25	26	28	29	2a	2c		**16**	**31**
2592	**A20**	2d	2e	2f	31	32	33	34	35	37	38	39	3a	3b	3c	3d	3e		**32**	**47**
2608	**A30**	3f	40	41	42	43	44	45	46	47	48	49	4a	4a	4b	4c	4d		**48**	**63**
2624	**A40**	0	2	3	5	6	8	9	b	c	e	f	11	12	13	15	16	**41**	**0**	**15**
2640	**A50**	18	19	1b	1c	1e	1f	20	22	23	25	26	27	29	2a	2b	2d		**16**	**31**
2656	**A60**	2e	2f	31	32	33	34	36	37	38	39	3a	3b	3d	3e	3f	40		**32**	**47**
2672	**A70**	41	42	43	44	45	46	47	48	49	4a	4b	4b	4c	4d	4e	4f		**48**	**63**
2688	**A80**	0	2	3	5	6	8	9	b	c	e	f	11	12	14	15	17	**42**	**0**	**15**
2704	**A90**	18	1a	1b	1d	1e	20	21	23	24	26	27	28	2a	2b	2c	2e		**16**	**31**
2720	**AA0**	2f	30	32	33	34	36	37	38	39	3b	3c	3d	3e	3f	40	41		**32**	**47**
2736	**AB0**	43	44	45	46	47	48	49	4a	4b	4c	4c	4d	4e	4f	50	51		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
2752	**AC0**	0	2	3	5	6	8	9	b	d	e	10	11	13	14	16	17	**43**	**0**	**15**
2768	**AD0**	19	1b	1c	1e	1f	21	22	24	25	26	28	29	2b	2c	2e	2f		**16**	**31**
2784	**AE0**	30	32	33	34	36	37	38	39	3b	3c	3d	3e	40	41	42	43		**32**	**47**
2800	**AF0**	44	45	46	47	48	49	4a	4b	4c	4d	4e	4f	50	51	52	52		**48**	**63**
2816	**B00**	0	2	3	5	6	8	a	b	d	f	10	12	13	15	16	18	**44**	**0**	**15**
2832	**B10**	1a	1b	1d	1e	20	21	23	24	26	27	29	2a	2c	2d	2f	30		**16**	**31**
2848	**B20**	31	33	34	36	37	38	3a	3b	3c	3d	3f	40	41	42	43	45		**32**	**47**
2864	**B30**	46	47	48	49	4a	4b	4c	4d	4e	4f	50	51	52	53	54	54		**48**	**63**
2880	**B40**	0	2	3	5	7	8	a	c	d	f	10	12	14	15	17	19	**45**	**0**	**15**
2896	**B50**	1a	1c	1d	1f	21	22	24	25	27	28	2a	2b	2d	2e	30	31		**16**	**31**
2912	**B60**	33	34	35	37	38	39	3b	3c	3d	3f	40	41	43	44	45	46		**32**	**47**
2928	**B70**	47	48	4a	4b	4c	4d	4e	4f	50	51	52	53	54	55	55	56		**48**	**63**
2944	**B80**	0	2	3	5	7	8	a	c	e	f	11	13	14	16	17	19	**46**	**0**	**15**
2960	**B90**	1b	1c	1e	20	21	23	24	26	28	29	2b	2c	2e	2f	31	32		**16**	**31**
2976	**BA0**	34	35	37	38	39	3b	3c	3d	3f	40	41	43	44	45	47	48		**32**	**47**
2992	**BB0**	49	4a	4b	4c	4d	4f	50	51	52	53	54	55	56	56	57	58		**48**	**63**
3008	**BC0**	0	2	3	5	7	9	a	c	e	10	11	13	15	16	18	1a	**47**	**0**	**15**
3024	**BD0**	1b	1d	1f	20	22	24	25	27	28	2a	2c	2d	2f	30	32	33		**16**	**31**
3040	**BE0**	35	36	38	39	3b	3c	3d	3f	40	42	43	44	45	47	48	49		**32**	**47**
3056	**BF0**	4a	4c	4d	4e	4f	50	51	52	53	54	55	56	57	58	59	5a		**48**	**63**
3072	**C00**	0	2	4	5	7	9	b	c	e	10	12	13	15	17	19	1a	**48**	**0**	**15**
3088	**C10**	1c	1e	1f	21	23	24	26	28	29	2b	2d	2e	30	31	33	34		**16**	**31**
3104	**C20**	36	37	39	3a	3c	3d	3f	40	42	43	44	46	47	48	4a	4b		**32**	**47**
3120	**C30**	4c	4d	4f	50	51	52	53	54	55	56	57	58	59	5a	5b	5c		**48**	**63**
3136	**C40**	0	2	4	5	7	9	b	d	e	10	12	14	15	17	19	1b	**49**	**0**	**15**
3152	**C50**	1d	1e	20	22	23	25	27	28	2a	2c	2d	2f	31	32	34	35		**16**	**31**
3168	**C60**	37	39	3a	3c	3d	3f	40	42	43	44	46	47	48	4a	4b	4c		**32**	**47**
3184	**C70**	4e	4f	50	51	53	54	55	56	57	58	59	5a	5b	5c	5d	5e		**48**	**63**
3200	**C80**	0	2	4	6	7	9	b	d	f	10	12	14	16	18	1a	1b	**50**	**0**	**15**
3216	**C90**	1d	1f	21	22	24	26	28	29	2b	2d	2e	30	32	33	35	37		**16**	**31**
3232	**CA0**	38	3a	3b	3d	3e	40	41	43	44	46	47	49	4a	4b	4d	4e		**32**	**47**
3248	**CB0**	4f	51	52	53	54	55	57	58	59	5a	5b	5c	5d	5e	5f	60		**48**	**63**
3264	**CC0**	0	2	4	6	8	9	b	d	f	11	13	15	16	18	1a	1c	**51**	**0**	**15**
3280	**CD0**	1e	1f	21	23	25	27	28	2a	2c	2e	2f	31	33	34	36	38		**16**	**31**
3296	**CE0**	39	3b	3c	3e	40	41	43	44	46	47	49	4a	4b	4d	4e	50		**32**	**47**
3312	**CF0**	51	52	53	55	56	57	58	59	5b	5c	5d	5e	5f	60	61	62		**48**	**63**
3328	**D00**	0	2	4	6	8	a	b	d	f	11	13	15	17	19	1b	1c	**52**	**0**	**15**
3344	**D10**	1e	20	22	24	26	27	29	2b	2d	2e	30	32	34	35	37	39		**16**	**31**
3360	**D20**	3a	3c	3e	3f	41	42	44	46	47	49	4a	4b	4d	4e	50	51		**32**	**47**
3376	**D30**	52	54	55	56	58	59	5a	5b	5c	5d	5f	60	61	62	63	64		**48**	**63**
3392	**D40**	0	2	4	6	8	a	c	e	10	11	13	15	17	19	1b	1d	**53**	**0**	**15**
3408	**D50**	1f	21	23	24	26	28	2a	2c	2e	2f	31	33	35	36	38	3a		**16**	**31**
3424	**D60**	3c	3d	3f	40	42	44	45	47	48	4a	4b	4d	4e	50	51	53		**32**	**47**
3440	**D70**	54	55	57	58	59	5b	5c	5d	5e	5f	60	61	63	64	65	66		**48**	**63**
3456	**D80**	0	2	4	6	8	a	c	e	10	12	14	16	18	1a	1c	1e	**54**	**0**	**15**
3472	**D90**	1f	21	23	25	27	29	2b	2d	2e	30	32	34	36	37	39	3b		**16**	**31**
3488	**DA0**	3d	3e	40	42	43	45	47	48	4a	4b	4d	4e	50	51	53	54		**32**	**47**
3504	**DB0**	56	57	58	5a	5b	5c	5d	5f	60	61	62	63	64	65	67	68		**48**	**63**
3520	**DC0**	0	2	4	6	8	a	c	e	10	12	14	16	18	1a	1c	1e	**55**	**0**	**15**
3536	**DD0**	20	22	24	26	28	2a	2c	2d	2f	31	33	35	37	38	3a	3c		**16**	**31**
3552	**DE0**	3e	3f	41	43	45	46	48	4a	4b	4d	4e	50	51	53	54	56		**32**	**47**
3568	**DF0**	57	59	5a	5b	5d	5e	5f	60	62	63	64	65	66	67	68	69		**48**	**63**
3584	**E00**	0	2	4	6	8	a	c	e	10	12	15	17	19	1b	1d	1f	**56**	**0**	**15**
3600	**E10**	21	23	25	27	28	2a	2c	2e	30	32	34	36	38	39	3b	3d		**16**	**31**
3616	**E20**	3f	41	42	44	46	48	49	4b	4c	4e	50	51	53	54	56	57		**32**	**47**
3632	**E30**	59	5a	5c	5d	5e	60	61	62	63	65	66	67	68	69	6a	6b		**48**	**63**
3648	**E40**	0	2	4	6	8	a	d	f	11	13	15	17	19	1b	1d	1f	**57**	**0**	**15**
3664	**E50**	21	23	25	27	29	2b	2d	2f	31	33	35	37	39	3b	3c	3e		**16**	**31**
3680	**E60**	40	42	44	45	47	49	4b	4c	4e	50	51	53	54	56	57	59		**32**	**47**
3696	**E70**	5a	5c	5d	5f	60	61	63	64	65	66	68	69	6a	6b	6c	6d		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
3712	**E80**	0	2	4	6	9	b	d	f	11	13	15	17	19	1c	1e	20	**58**	**0**	**15**
3728	**E90**	22	24	26	28	2a	2c	2e	30	32	34	36	38	3a	3c	3d	3f		**16**	**31**
3744	**EA0**	41	43	45	47	48	4a	4c	4e	4f	51	53	54	56	57	59	5a		**32**	**47**
3760	**EB0**	5c	5d	5f	60	62	63	64	66	67	68	69	6b	6c	6d	6e	6f		**48**	**63**
3776	**EC0**	0	2	4	7	9	b	d	f	11	13	16	18	1a	1c	1e	20	**59**	**0**	**15**
3792	**ED0**	22	24	27	29	2b	2d	2f	31	33	35	37	39	3b	3d	3e	40		**16**	**31**
3808	**EE0**	42	44	46	48	4a	4b	4d	4f	51	52	54	56	57	59	5a	5c		**32**	**47**
3824	**EF0**	5e	5f	60	62	63	65	66	67	69	6a	6b	6d	6e	6f	70	71		**48**	**63**
3840	**F00**	0	2	4	7	9	b	d	f	12	14	16	18	1a	1c	1f	21	**60**	**0**	**15**
3856	**F10**	23	25	27	29	2b	2d	30	32	34	36	38	3a	3c	3e	40	41		**16**	**31**
3872	**F20**	43	45	47	49	4b	4d	4e	50	52	54	55	57	59	5a	5c	5e		**32**	**47**
3888	**F30**	5f	61	62	64	65	66	68	69	6b	6c	6d	6e	70	71	72	73		**48**	**63**
3904	**F40**	0	2	4	7	9	b	d	10	12	14	16	19	1b	1d	1f	21	**61**	**0**	**15**
3920	**F50**	23	26	28	2a	2c	2e	30	32	34	37	39	3b	3d	3f	41	43		**16**	**31**
3936	**F60**	44	46	48	4a	4c	4e	50	52	53	55	57	59	5a	5c	5d	5f		**32**	**47**
3952	**F70**	61	62	64	65	67	68	6a	6b	6c	6e	6f	70	71	73	74	75		**48**	**63**
3968	**F80**	0	2	5	7	9	b	e	10	12	14	17	19	1b	1d	20	22	**62**	**0**	**15**
3984	**F90**	24	26	28	2b	2d	2f	31	33	35	37	39	3c	3e	40	42	44		**16**	**31**
4000	**FA0**	46	48	4a	4b	4d	4f	51	53	55	56	58	5a	5c	5d	5f	61		**32**	**47**
4016	**FB0**	62	64	65	67	68	6a	6b	6d	6e	6f	71	72	73	75	76	77		**48**	**63**
4032	**FC0**	0	2	5	7	9	c	e	10	12	15	17	19	1c	1e	20	22	**63**	**0**	**15**
4048	**FD0**	25	27	29	2b	2e	30	32	34	36	38	3a	3d	3f	41	43	45		**16**	**31**
4064	**FE0**	47	49	4b	4d	4f	50	52	54	56	58	5a	5b	5d	5f	61	62		**32**	**47**
4080	**FF0**	64	65	67	69	6a	6c	6d	6e	70	71	73	74	75	76	78	79		**48**	**63**
4096	**1000**	0	2	5	7	9	c	e	10	13	15	17	1a	1c	1e	21	23	**64**	**0**	**15**
4112	**1010**	25	28	2a	2c	2e	30	33	35	37	39	3b	3d	40	42	44	46		**16**	**31**
4128	**1020**	48	4a	4c	4e	50	52	54	56	57	59	5b	5d	5f	60	62	64		**32**	**47**
4144	**1030**	65	67	69	6a	6c	6d	6f	70	72	73	74	76	77	78	7a	7b		**48**	**63**
4160	**1040**	0	2	5	7	a	c	e	11	13	15	18	1a	1d	1f	21	24	**65**	**0**	**15**
4176	**1050**	26	28	2a	2d	2f	31	33	36	38	3a	3c	3e	41	43	45	47		**16**	**31**
4192	**1060**	49	4b	4d	4f	51	53	55	57	59	5b	5d	5e	60	62	64	65		**32**	**47**
4208	**1070**	67	69	6a	6c	6d	6f	71	72	73	75	76	78	79	7a	7b	7d		**48**	**63**
4224	**1080**	0	2	5	7	a	c	f	11	13	16	18	1b	1d	1f	22	24	**66**	**0**	**15**
4240	**1090**	26	29	2b	2d	30	32	34	37	39	3b	3d	3f	42	44	46	48		**16**	**31**
4256	**10A0**	4a	4c	4e	50	52	54	56	58	5a	5c	5e	60	62	63	65	67		**32**	**47**
4272	**10B0**	69	6a	6c	6e	6f	71	72	74	75	77	78	79	7b	7c	7d	7f		**48**	**63**
4288	**10C0**	0	2	5	7	a	c	f	11	14	16	19	1b	1d	20	22	25	**67**	**0**	**15**
4304	**10D0**	27	29	2c	2e	30	33	35	37	3a	3c	3e	40	43	45	47	49		**16**	**31**
4320	**10E0**	4b	4d	4f	52	54	56	58	5a	5c	5d	5f	61	63	65	67	68		**32**	**47**
4336	**10F0**	6a	6c	6e	6f	71	72	74	75	77	78	7a	7b	7d	7e	7f	80		**48**	**63**
4352	**1100**	0	3	5	8	a	c	f	11	14	16	19	1b	1e	20	23	25	**68**	**0**	**15**
4368	**1110**	28	2a	2c	2f	31	33	36	38	3a	3d	3f	41	44	46	48	4a		**16**	**31**
4384	**1120**	4c	4f	51	53	55	57	59	5b	5d	5f	61	63	65	66	68	6a		**32**	**47**
4400	**1130**	6c	6e	6f	71	73	74	76	77	79	7a	7c	7d	7e	80	81	82		**48**	**63**
4416	**1140**	0	3	5	8	a	d	f	12	14	17	19	1c	1e	21	23	26	**69**	**0**	**15**
4432	**1150**	28	2b	2d	2f	32	34	37	39	3b	3e	40	42	45	47	49	4b		**16**	**31**
4448	**1160**	4d	50	52	54	56	58	5a	5c	5e	60	62	64	66	68	6a	6c		**32**	**47**
4464	**1170**	6d	6f	71	73	74	76	77	79	7b	7c	7e	7f	80	82	83	84		**48**	**63**
4480	**1180**	0	3	5	8	a	d	f	12	15	17	1a	1c	1f	21	24	26	**70**	**0**	**15**
4496	**1190**	29	2b	2e	30	33	35	37	3a	3c	3f	41	43	46	48	4a	4c		**16**	**31**
4512	**11A0**	4f	51	53	55	57	59	5c	5e	60	62	64	66	68	69	6b	6d		**32**	**47**
4528	**11B0**	6f	71	72	74	76	78	79	7b	7c	7e	7f	81	82	84	85	86		**48**	**63**
4544	**11C0**	0	3	5	8	a	d	10	12	15	17	1a	1d	1f	22	24	27	**71**	**0**	**15**
4560	**11D0**	29	2c	2e	31	33	36	38	3b	3d	3f	42	44	47	49	4b	4d		**16**	**31**
4576	**11E0**	50	52	54	56	59	5b	5d	5f	61	63	65	67	69	6b	6d	6f		**32**	**47**
4592	**11F0**	71	72	74	76	78	79	7b	7d	7e	80	81	83	84	85	87	88		**48**	**63**
4608	**1200**	0	3	5	8	b	d	10	13	15	18	1a	1d	20	22	25	27	**72**	**0**	**15**
4624	**1210**	2a	2c	2f	32	34	37	39	3b	3e	40	43	45	48	4a	4c	4f		**16**	**31**
4640	**1220**	51	53	55	58	5a	5c	5e	60	62	64	66	68	6a	6c	6e	70		**32**	**47**
4656	**1230**	72	74	76	78	79	7b	7d	7e	80	81	83	84	86	87	89	8a		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
4672	**1240**	0	3	5	8	b	d	10	13	15	18	1b	1d	20	23	25	28	**73**	**0**	**15**
4688	**1250**	2a	2d	30	32	35	37	3a	3c	3f	41	44	46	49	4b	4d	50		**16**	**31**
4704	**1260**	52	54	57	59	5b	5d	5f	62	64	66	68	6a	6c	6e	70	72		**32**	**47**
4720	**1270**	74	76	77	79	7b	7d	7e	80	82	83	85	86	88	89	8b	8c		**48**	**63**
4736	**1280**	0	3	5	8	b	e	10	13	16	18	1b	1e	20	23	26	28	**74**	**0**	**15**
4752	**1290**	2b	2e	30	33	35	38	3b	3d	40	42	45	47	4a	4c	4e	51		**16**	**31**
4768	**12A0**	53	55	58	5a	5c	5f	61	63	65	67	69	6b	6d	6f	71	73		**32**	**47**
4784	**12B0**	75	77	79	7b	7d	7e	80	82	83	85	87	88	8a	8b	8d	8e		**48**	**63**
4800	**12C0**	0	3	6	8	b	e	11	13	16	19	1b	1e	21	24	26	29	**75**	**0**	**15**
4816	**12D0**	2c	2e	31	34	36	39	3b	3e	40	43	46	48	4b	4d	4f	52		**16**	**31**
4832	**12E0**	54	57	59	5b	5e	60	62	64	66	69	6b	6d	6f	71	73	75		**32**	**47**
4848	**12F0**	77	79	7b	7d	7e	80	82	84	85	87	88	8a	8b	8d	8e	90		**48**	**63**
4864	**1300**	0	3	6	8	b	e	11	14	16	19	1c	1f	21	24	27	2a	**76**	**0**	**15**
4880	**1310**	2c	2f	32	34	37	3a	3c	3f	41	44	46	49	4c	4e	50	53		**16**	**31**
4896	**1320**	55	58	5a	5c	5f	61	63	66	68	6a	6c	6e	70	72	74	76		**32**	**47**
4912	**1330**	78	7a	7c	7e	80	82	84	85	87	89	8a	8c	8d	8f	90	92		**48**	**63**
4928	**1340**	0	3	6	8	b	e	11	14	17	19	1c	1f	22	25	27	2a	**77**	**0**	**15**
4944	**1350**	2d	30	32	35	38	3a	3d	40	42	45	47	4a	4d	4f	52	54		**16**	**31**
4960	**1360**	56	59	5b	5e	60	62	65	67	69	6b	6e	70	72	74	76	78		**32**	**47**
4976	**1370**	7a	7c	7e	80	82	83	85	87	89	8a	8c	8e	8f	91	92	94		**48**	**63**
4992	**1380**	0	3	6	9	b	e	11	14	17	1a	1d	1f	22	25	28	2b	**78**	**0**	**15**
5008	**1390**	2d	30	33	36	38	3b	3e	40	43	46	48	4b	4e	50	53	55		**16**	**31**
5024	**13A0**	58	5a	5c	5f	61	64	66	68	6b	6d	6f	71	73	75	78	7a		**32**	**47**
5040	**13B0**	7c	7e	80	81	83	85	87	89	8b	8c	8e	8f	91	93	94	96		**48**	**63**
5056	**13C0**	0	3	6	9	c	f	11	14	17	1a	1d	20	23	26	28	2b	**79**	**0**	**15**
5072	**13D0**	2e	31	34	36	39	3c	3f	41	44	47	49	4c	4e	51	54	56		**16**	**31**
5088	**13E0**	59	5b	5e	60	63	65	67	6a	6c	6e	70	73	75	77	79	7b		**32**	**47**
5104	**13F0**	7d	7f	81	83	85	87	89	8b	8c	8e	90	91	93	94	96	97		**48**	**63**
5120	**1400**	0	3	6	9	c	f	12	15	17	1a	1d	20	23	26	29	2c	**80**	**0**	**15**
5136	**1410**	2f	31	34	37	3a	3d	3f	42	45	48	4a	4d	4f	52	55	57		**16**	**31**
5152	**1420**	5a	5c	5f	61	64	66	69	6b	6d	70	72	74	76	78	7b	7d		**32**	**47**
5168	**1430**	7f	81	83	85	87	89	8a	8c	8e	90	92	93	95	96	98	99		**48**	**63**
5184	**1440**	0	3	6	9	c	f	12	15	18	1b	1e	21	24	26	29	2c	**81**	**0**	**15**
5200	**1450**	2f	32	35	38	3b	3d	40	43	46	48	4b	4e	50	53	56	58		**16**	**31**
5216	**1460**	5b	5e	60	63	65	67	6a	6c	6f	71	73	76	78	7a	7c	7e		**32**	**47**
5232	**1470**	80	82	84	86	88	8a	8c	8e	90	92	93	95	97	98	9a	9b		**48**	**63**
5248	**1480**	0	3	6	9	c	f	12	15	18	1b	1e	21	24	27	2a	2d	**82**	**0**	**15**
5264	**1490**	30	33	36	38	3b	3e	41	44	47	49	4c	4f	51	54	57	59		**16**	**31**
5280	**14A0**	5c	5f	61	64	66	69	6b	6e	70	72	75	77	79	7b	7e	80		**32**	**47**
5296	**14B0**	82	84	86	88	8a	8c	8e	90	92	93	95	97	99	9a	9c	9d		**48**	**63**
5312	**14C0**	0	3	6	9	c	f	12	15	18	1b	1e	21	24	27	2a	2d	**83**	**0**	**15**
5328	**14D0**	30	33	36	39	3c	3f	42	45	47	4a	4d	50	52	55	58	5b		**16**	**31**
5344	**14E0**	5d	60	62	65	68	6a	6d	6f	71	74	76	78	7b	7d	7f	81		**32**	**47**
5360	**14F0**	84	86	88	8a	8c	8e	90	92	93	95	97	99	9a	9c	9e	9f		**48**	**63**
5376	**1500**	0	3	6	9	c	f	13	16	19	1c	1f	22	25	28	2b	2e	**84**	**0**	**15**
5392	**1510**	31	34	37	3a	3d	40	43	45	48	4b	4e	51	53	56	59	5c		**16**	**31**
5408	**1520**	5e	61	64	66	69	6b	6e	70	73	75	78	7a	7c	7e	81	83		**32**	**47**
5424	**1530**	85	87	89	8b	8d	8f	91	93	95	97	99	9b	9c	9e	9f	a1		**48**	**63**
5440	**1540**	0	3	6	9	d	10	13	16	19	1c	1f	22	25	28	2b	2e	**85**	**0**	**15**
5456	**1550**	31	34	37	3a	3d	40	43	46	49	4c	4f	52	54	57	5a	5d		**16**	**31**
5472	**1560**	5f	62	65	67	6a	6d	6f	72	74	77	79	7b	7e	80	82	85		**32**	**47**
5488	**1570**	87	89	8b	8d	8f	91	93	95	97	99	9b	9c	9e	a0	a1	a3		**48**	**63**
5504	**1580**	0	3	6	9	d	10	13	16	19	1c	20	23	26	29	2c	2f	**86**	**0**	**15**
5520	**1590**	32	35	38	3b	3e	41	44	47	4a	4d	50	53	55	58	5b	5e		**16**	**31**
5536	**15A0**	61	63	66	69	6b	6e	70	73	75	78	7a	7d	7f	82	84	86		**32**	**47**
5552	**15B0**	88	8a	8d	8f	91	93	95	97	99	9b	9c	9e	a0	a2	a3	a5		**48**	**63**
5568	**15C0**	0	3	6	a	d	10	13	16	1a	1d	20	23	26	29	2c	30	**87**	**0**	**15**
5584	**15D0**	33	36	39	3c	3f	42	45	48	4b	4e	51	54	56	59	5c	5f		**16**	**31**
5600	**15E0**	62	64	67	6a	6d	6f	72	74	77	79	7c	7e	81	83	85	88		**32**	**47**
5616	**15F0**	8a	8c	8e	90	93	95	97	99	9b	9c	9e	a0	a2	a4	a5	a7		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
5632	**1600**	0	3	6	a	d	10	13	17	1a	1d	20	23	27	2a	2d	30	**88**	**0**	**15**
5648	**1610**	33	36	39	3d	40	43	46	49	4c	4f	52	55	57	5a	5d	60		**16**	**31**
5664	**1620**	63	66	68	6b	6e	70	73	76	78	7b	7d	80	82	85	87	89		**32**	**47**
5680	**1630**	8b	8e	90	92	94	96	98	9a	9c	9e	a0	a2	a4	a5	a7	a9		**48**	**63**
5696	**1640**	0	3	7	a	d	10	14	17	1a	1d	21	24	27	2a	2d	31	**89**	**0**	**15**
5712	**1650**	34	37	3a	3d	40	43	46	4a	4d	50	53	55	58	5b	5e	61		**16**	**31**
5728	**1660**	64	67	6a	6c	6f	72	74	77	7a	7c	7f	81	84	86	88	8b		**32**	**47**
5744	**1670**	8d	8f	92	94	96	98	9a	9c	9e	a0	a2	a4	a6	a7	a9	ab		**48**	**63**
5760	**1680**	0	3	7	a	d	11	14	17	1a	1e	21	24	27	2b	2e	31	**90**	**0**	**15**
5776	**1690**	34	38	3b	3e	41	44	47	4a	4d	50	53	56	59	5c	5f	62		**16**	**31**
5792	**16A0**	65	68	6b	6d	70	73	76	78	7b	7e	80	83	85	88	8a	8c		**32**	**47**
5808	**16B0**	8f	91	93	95	98	9a	9c	9e	a0	a2	a4	a6	a7	a9	ab	ad		**48**	**63**
5824	**16C0**	0	3	7	a	d	11	14	17	1b	1e	21	25	28	2b	2e	32	**91**	**0**	**15**
5840	**16D0**	35	38	3b	3f	42	45	48	4b	4e	51	54	57	5a	5d	60	63		**16**	**31**
5856	**16E0**	66	69	6c	6f	71	74	77	7a	7c	7f	82	84	87	89	8b	8e		**32**	**47**
5872	**16F0**	90	93	95	97	99	9b	9e	a0	a2	a4	a6	a7	a9	ab	ad	ae		**48**	**63**
5888	**1700**	0	3	7	a	e	11	14	18	1b	1e	22	25	28	2c	2f	32	**92**	**0**	**15**
5904	**1710**	36	39	3c	3f	42	46	49	4c	4f	52	55	58	5b	5e	61	64		**16**	**31**
5920	**1720**	67	6a	6d	70	73	76	78	7b	7e	80	83	86	88	8b	8d	8f		**32**	**47**
5936	**1730**	92	94	96	99	9b	9d	9f	a1	a3	a5	a7	a9	ab	ad	af	b0		**48**	**63**
5952	**1740**	0	3	7	a	e	11	15	18	1b	1f	22	25	29	2c	2f	33	**93**	**0**	**15**
5968	**1750**	36	39	3d	40	43	46	4a	4d	50	53	56	59	5c	5f	62	65		**16**	**31**
5984	**1760**	68	6b	6e	71	74	77	7a	7c	7f	82	84	87	8a	8c	8f	91		**32**	**47**
6000	**1770**	93	96	98	9a	9d	9f	a1	a3	a5	a7	a9	ab	ad	af	b1	b2		**48**	**63**
6016	**1780**	0	3	7	a	e	11	15	18	1c	1f	22	26	29	2d	30	33	**94**	**0**	**15**
6032	**1790**	37	3a	3d	41	44	47	4a	4e	51	54	57	5a	5d	60	64	67		**16**	**31**
6048	**17A0**	6a	6d	6f	72	75	78	7b	7e	80	83	86	88	8b	8e	90	93		**32**	**47**
6064	**17B0**	95	97	9a	9c	9e	a1	a3	a5	a7	a9	ab	ad	af	b1	b2	b4		**48**	**63**
6080	**17C0**	0	3	7	a	e	11	15	18	1c	1f	23	26	2a	2d	31	34	**95**	**0**	**15**
6096	**17D0**	37	3b	3e	41	45	48	4b	4e	52	55	58	5b	5e	62	65	68		**16**	**31**
6112	**17E0**	6b	6e	71	74	76	79	7c	7f	82	85	87	8a	8c	8f	92	94		**32**	**47**
6128	**17F0**	97	99	9b	9e	a0	a2	a4	a7	a9	ab	ad	af	b1	b3	b4	b6		**48**	**63**
6144	**1800**	0	4	7	b	e	12	15	19	1c	20	23	27	2a	2e	31	34	**96**	**0**	**15**
6160	**1810**	38	3b	3f	42	45	49	4c	4f	53	56	59	5c	5f	63	66	69		**16**	**31**
6176	**1820**	6c	6f	72	75	78	7b	7e	80	83	86	89	8b	8e	91	93	96		**32**	**47**
6192	**1830**	98	9b	9d	9f	a2	a4	a6	a8	aa	ad	af	b1	b3	b4	b6	b8		**48**	**63**
6208	**1840**	0	4	7	b	e	12	15	19	1c	20	24	27	2b	2e	32	35	**97**	**0**	**15**
6224	**1850**	38	3c	3f	43	46	49	4d	50	53	57	5a	5d	60	64	67	6a		**16**	**31**
6240	**1860**	6d	70	73	76	79	7c	7f	82	84	87	8a	8d	8f	92	95	97		**32**	**47**
6256	**1870**	9a	9c	9f	a1	a3	a6	a8	aa	ac	ae	b0	b2	b4	b6	b8	ba		**48**	**63**
6272	**1880**	0	4	7	b	e	12	16	19	1d	20	24	27	2b	2f	32	36	**98**	**0**	**15**
6288	**1890**	39	3c	40	43	47	4a	4e	51	54	58	5b	5e	61	65	68	6b		**16**	**31**
6304	**18A0**	6e	71	74	77	7a	7d	80	83	86	89	8b	8e	91	94	96	99		**32**	**47**
6320	**18B0**	9b	9e	a0	a3	a5	a7	aa	ac	ae	b0	b2	b4	b6	b8	ba	bc		**48**	**63**
6336	**18C0**	0	4	7	b	f	12	16	19	1d	21	24	28	2b	2f	33	36	**99**	**0**	**15**
6352	**18D0**	3a	3d	41	44	48	4b	4e	52	55	58	5c	5f	62	66	69	6c		**16**	**31**
6368	**18E0**	6f	72	75	78	7b	7e	81	84	87	8a	8d	90	92	95	98	9a		**32**	**47**
6384	**18F0**	9d	9f	a2	a4	a7	a9	ab	ae	b0	b2	b4	b6	b8	ba	bc	be		**48**	**63**
6400	**1900**	0	4	7	b	f	12	16	1a	1d	21	25	28	2c	2f	33	37	**100**	**0**	**15**
6416	**1910**	3a	3e	41	45	48	4c	4f	53	56	59	5d	60	63	67	6a	6d		**16**	**31**
6432	**1920**	70	73	77	7a	7d	80	83	86	89	8b	8e	91	94	97	99	9c		**32**	**47**
6448	**1930**	9e	a1	a4	a6	a8	ab	ad	af	b2	b4	b6	b8	ba	bc	be	c0		**48**	**63**
6464	**1940**	0	4	7	b	f	13	16	1a	1e	21	25	29	2c	30	34	37	**101**	**0**	**15**
6480	**1950**	3b	3e	42	45	49	4c	50	53	57	5a	5e	61	64	68	6b	6e		**16**	**31**
6496	**1960**	71	75	78	7b	7e	81	84	87	8a	8d	90	93	95	98	9b	9d		**32**	**47**
6512	**1970**	a0	a3	a5	a8	aa	ac	af	b1	b3	b6	b8	ba	bc	be	c0	c2		**48**	**63**
6528	**1980**	0	4	8	b	f	13	16	1a	1e	22	25	29	2d	30	34	38	**102**	**0**	**15**
6544	**1990**	3b	3f	43	46	4a	4d	51	54	58	5b	5f	62	65	69	6c	6f		**16**	**31**
6560	**19A0**	73	76	79	7c	7f	82	85	88	8b	8e	91	94	97	9a	9c	9f		**32**	**47**
6576	**19B0**	a2	a4	a7	a9	ac	ae	b1	b3	b5	b7	ba	bc	be	c0	c2	c4		**48**	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	**Hex**																
6592	**19C0**	0	4	8	b	f	13	17	1a	1e	22	26	29	2d	31	35	38
6608	**19D0**	3c	40	43	47	4a	4e	52	55	59	5c	60	63	66	6a	6d	70
6624	**19E0**	74	77	7a	7d	80	84	87	8a	8d	90	93	95	98	9b	9e	a1
6640	**19F0**	a3	a6	a8	ab	ad	b0	b2	b5	b7	b9	bb	bd	c0	c2	c4	c5
6656	**1A00**	0	4	8	b	f	13	17	1b	1f	22	26	2a	2e	31	35	39
6672	**1A10**	3d	40	44	48	4b	4f	52	56	59	5d	60	64	67	6b	6e	71
6688	**1A20**	75	78	7b	7f	82	85	88	8b	8e	91	94	97	9a	9d	9f	a2
6704	**1A30**	a5	a7	aa	ad	af	b2	b4	b6	b9	bb	bd	bf	c1	c3	c5	c7
6720	**1A40**	0	4	8	c	f	13	17	1b	1f	23	26	2a	2e	32	36	39
6736	**1A50**	3d	41	45	48	4c	50	53	57	5a	5e	61	65	68	6c	6f	73
6752	**1A60**	76	79	7c	80	83	86	89	8c	8f	92	95	98	9b	9e	a1	a4
6768	**1A70**	a6	a9	ac	ae	b1	b3	b6	b8	ba	bd	bf	c1	c3	c5	c7	c9
6784	**1A80**	0	4	8	c	10	13	17	1b	1f	23	27	2b	2f	32	36	3a
6800	**1A90**	3e	41	45	49	4d	50	54	58	5b	5f	62	66	69	6d	70	74
6816	**1AA0**	77	7a	7e	81	84	87	8b	8e	91	94	97	9a	9d	a0	a2	a5
6832	**1AB0**	a8	ab	ad	b0	b3	b5	b7	ba	bc	bf	c1	c3	c5	c7	c9	cb
6848	**1AC0**	0	4	8	c	10	14	18	1c	1f	23	27	2b	2f	33	37	3a
6864	**1AD0**	3e	42	46	4a	4d	51	55	58	5c	60	63	67	6a	6e	71	75
6880	**1AE0**	78	7c	7f	82	85	89	8c	8f	92	95	98	9b	9e	a1	a4	a7
6896	**1AF0**	aa	ac	af	b2	b4	b7	b9	bc	be	c0	c3	c5	c7	c9	cb	cd
6912	**1B00**	0	4	8	c	10	14	18	1c	20	24	28	2b	2f	33	37	3b
6928	**1B10**	3f	43	46	4a	4e	52	56	59	5d	61	64	68	6b	6f	72	76
6944	**1B20**	79	7d	80	83	87	8a	8d	90	94	97	9a	9d	a0	a3	a6	a8
6960	**1B30**	ab	ae	b1	b3	b6	b8	bb	bd	c0	c2	c4	c7	c9	cb	cd	cf
6976	**1B40**	0	4	8	c	10	14	18	1c	20	24	28	2c	30	34	38	3c
6992	**1B50**	3f	43	47	4b	4f	53	56	5a	5e	61	65	69	6c	70	73	77
7008	**1B60**	7a	7e	81	85	88	8b	8e	92	95	98	9b	9e	a1	a4	a7	aa
7024	**1B70**	ad	b0	b2	b5	b8	ba	bd	bf	c2	c4	c6	c9	cb	cd	cf	d1
7040	**1B80**	0	4	8	c	10	14	18	1c	20	24	28	2c	30	34	38	3c
7056	**1B90**	40	44	48	4c	4f	53	57	5b	5f	62	66	6a	6d	71	74	78
7072	**1BA0**	7c	7f	82	86	89	8d	90	93	96	99	9d	a0	a3	a6	a9	ab
7088	**1BB0**	ae	b1	b4	b7	b9	bc	be	c1	c3	c6	c8	ca	cd	cf	d1	d3
7104	**1BC0**	0	4	8	c	10	14	18	1d	21	25	29	2d	31	35	39	3d
7120	**1BD0**	41	45	48	4c	50	54	58	5c	5f	63	67	6b	6e	72	76	79
7136	**1BE0**	7d	80	84	87	8a	8e	91	94	98	9b	9e	a1	a4	a7	aa	ad
7152	**1BF0**	b0	b3	b6	b8	bb	be	c0	c3	c5	c8	ca	cc	ce	d1	d3	d5
7168	**1C00**	0	4	8	c	10	15	19	1d	21	25	29	2d	31	35	39	3d
7184	**1C10**	41	45	49	4d	51	55	59	5d	60	64	68	6c	6f	73	77	7a
7200	**1C20**	7e	81	85	88	8c	8f	92	96	99	9c	9f	a3	a6	a9	ac	af
7216	**1C30**	b2	b4	b7	ba	bd	bf	c2	c4	c7	c9	cc	ce	d0	d3	d5	d7
7232	**1C40**	0	4	8	c	11	15	19	1d	21	25	29	2d	32	36	3a	3e
7248	**1C50**	42	46	4a	4e	52	56	59	5d	61	65	69	6d	70	74	78	7b
7264	**1C60**	7f	82	86	89	8d	90	94	97	9a	9e	a1	a4	a7	aa	ad	b0
7280	**1C70**	b3	b6	b9	bc	be	c1	c4	c6	c9	cb	ce	d0	d2	d4	d7	d9
7296	**1C80**	0	4	8	d	11	15	19	1d	21	26	2a	2e	32	36	3a	3e
7312	**1C90**	42	46	4a	4e	52	56	5a	5e	62	66	6a	6e	71	75	79	7c
7328	**1CA0**	80	84	87	8b	8e	92	95	98	9c	9f	a2	a5	a9	ac	af	b2
7344	**1CB0**	b5	b8	ba	bd	c0	c3	c5	c8	ca	cd	cf	d2	d4	d6	d8	db
7360	**1CC0**	0	4	8	d	11	15	19	1e	22	26	2a	2e	32	37	3b	3f
7376	**1CD0**	43	47	4b	4f	53	57	5b	5f	63	67	6b	6e	72	76	7a	7d
7392	**1CE0**	81	85	88	8c	8f	93	96	9a	9d	a0	a4	a7	aa	ad	b0	b3
7408	**1CF0**	b6	b9	bc	bf	c2	c4	c7	ca	cc	cf	d1	d4	d6	d8	da	dc

Módulo Cuantizado	Angulo Cuantizado	
103	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
104	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
105	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
106	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
107	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
108	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
109	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
110	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
111	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
112	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
113	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
114	**0**	**15**
	16	**31**
	32	**47**
	48	**63**
115	**0**	**15**
	16	**31**
	32	**47**
	48	**63**

Address		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Módulo Cuantizado	Angulo Cuantizado	
Dec	Hex																			
7424	**1D00**	0	4	9	d	11	15	1a	1e	22	26	2b	2f	33	37	3b	3f	**116**	**0**	**15**
7440	**1D10**	44	48	4c	50	54	58	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7456	**1D20**	81	85	89	8c	90	94	97	9b	9f	a2	a5	a9	ac	af	b2	b5		**32**	**47**
7472	**1D30**	b8	bb	be	c1	c4	c6	c9	cc	ce	d1	d3	d6	d8	da	dd	df		**48**	**63**
7488	**1D40**	0	4	9	d	11	16	1a	1e	23	27	2b	2f	34	38	3c	40	**117**	**0**	**15**
7504	**1D50**	44	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7520	**1D60**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b1	b4	b7		**32**	**47**
7536	**1D70**	ba	bd	c0	c3	c6	c9	cb	ce	d1	d3	d6	d8	db	dd	df	e1		**48**	**63**
7552	**1D80**	0	4	9	d	12	16	1a	1f	23	27	2c	30	34	39	3d	41	**118**	**0**	**15**
7568	**1D90**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7584	**1DA0**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7600	**1DB0**	bd	c0	c3	c6	c9	cc	ce	d1	d4	d6	d9	db	de	e0	e2	e5		**48**	**63**
7616	**1DC0**	0	4	9	d	12	16	1b	1f	24	28	2c	31	35	39	3d	41	**119**	**0**	**15**
7632	**1DD0**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7648	**1DE0**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7664	**1DF0**	be	c2	c6	c9	cc	cf	d2	d5	d7	da	dd	df	e2	e4	e6	e9		**48**	**63**
7680	**1E00**	0	5	9	e	12	17	1b	20	24	29	2d	31	35	39	3d	41	**120**	**0**	**15**
7696	**1E10**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7712	**1E20**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7728	**1E30**	be	c2	c6	ca	ce	d2	d6	da	dc	df	e2	e4	e7	e9	ec	ee		**48**	**63**
7744	**1E40**	0	5	9	e	13	17	1c	20	25	29	2d	31	35	39	3d	41	**121**	**0**	**15**
7760	**1E50**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7776	**1E60**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7792	**1E70**	be	c2	c6	ca	ce	d2	d6	da	df	e3	e8	ec	ef	f2	f4	f7		**48**	**63**
7808	**1E80**	0	0	0	e	13	17	1c	20	25	29	2d	31	35	39	3d	41	**122**	**0**	**15**
7824	**1E90**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7840	**1EA0**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7856	**1EB0**	be	c2	c6	ca	ce	d2	d6	da	df	e3	e8	ec	f1	ff	ff	ff		**48**	**63**
7872	**1EC0**	0	0	0	0	0	0	1c	20	25	29	2d	31	35	39	3d	41	**123**	**0**	**15**
7888	**1ED0**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7904	**1EE0**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7920	**1EF0**	be	c2	c6	ca	ce	d2	d6	da	df	e3	ff	ff	ff	ff	ff	ff		**48**	**63**
7936	**1F00**	0	0	0	0	0	0	0	0	0	29	2d	31	35	39	3d	41	**124**	**0**	**15**
7952	**1F10**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
7968	**1F20**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
7984	**1F30**	be	c2	c6	ca	ce	d2	d6	ff	ff	ff	ff	ff	ff	ff	ff	ff		**48**	**63**
8000	**1F40**	0	0	0	0	0	0	0	0	0	0	0	0	0	39	3d	41	**125**	**0**	**15**
8016	**1F50**	45	49	4d	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
8032	**1F60**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	b2	b6	ba		**32**	**47**
8048	**1F70**	be	c2	c6	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff		**48**	**63**
8064	**1F80**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**126**	**0**	**15**
8080	**1F90**	0	0	0	51	55	59	5c	60	64	68	6b	6f	73	76	7a	7e		**16**	**31**
8096	**1FA0**	81	85	89	8c	90	94	97	9b	9f	a3	a6	aa	ae	ff	ff	ff		**32**	**47**
8112	**1FB0**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff		**48**	**63**
8128	**1FC0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**127**	**0**	**15**
8144	**1FD0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		**16**	**31**
8160	**1FE0**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff		**32**	**47**
8176	**1FF0**	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff		**48**	**63**

Referencias:

Six-Step
Modo II
Modo I
Modo Lineal

LISTADO DEL PROYECTO EN C, PARA LA COMUNICACIÓN DEL MODULADOR CON LA PC

A continuación se presentan los listados de los programas en lenguaje “C”, utilizados en el proyecto “Usvm_v1” del modulador vectorial universal. El programa esta compuesto del archivo principal “”Usvm_v1.prj” y de dos archivos en C: “utilmenu.c” y “mdldr.c”. El primer archivo en C tiene las funciones gráficas y el menú. El segundo archivo en C, tiene el cuerpo principal del programa y las funciones de lectura-escritura del Bus ISA, permitiendo la comunicación entre la PC y el modulador.

Con este programa se puede probar al modulador de dos maneras diferentes: manualmente o utilizando un archivo. La forma manual requiere la carga de una muestra del vector de referencia. La segunda opción requiere primero la generación de todas las muestras del vector de referencia guardadas en un archivo, y luego cargar dicho archivo en el programa. Los datos de las muestras se arman teniendo en cuanta los lineamientos del capitulo 4 ítem 4.3.

```
/*********************************************************************
                                  Mdldr.c
*********************************************************************/

#include <string.h>
#include <stdio.h>
#include <fcntl.h>
#include <io.h>
#include <conio.h>
#include <dos.h>
#include <GRAPHICS.H>

// Definiciones
#define SALIR 1
#define TESTMANUAL 2
#define TESTARCHIVO 3
#define MENU 4
#define RUN 5
#define STOP 6
// Definicion de ventanas en el menu
#define MENU_WIN       window(2,21,27,23)
#define STATUS_WIN     window(29,21,79,23)
#define MAIN_WIN       window(2,4,79,19)
#define TITLE_WIN      window(1,1,79,3)

void frame(void);
void Imprime_menu_principal(void);
void Imprime_menu_secundario(int);
void Imprime_menu_run(void);
void Imprime_menu_stop(void);
int Cargar_archivo(void);  // Devuelve la cantidad de muestras que tiene el archivo
int Enviar_dato_ISA(int);
void Detener_Modulador(void);

   int habilitacion = 1;
   int posicionx = 10;   // Variables para posicionar los mensajes en la pantalla
   int posiciony = 5;
   int portw = 0x31e; /*  Direccion Puerto Escritura por default */
   int portr = 0x30e; /*  Direccion Puerto Lectura por default */
   unsigned long Vector_datos_archivo[6000];
   int Sample_Qty;
   int Sample_Adrs;
   int Aux,i,ii;

int main(void)
{
   int estado, accion = 0;
   char ch = 0;
   int Cantidad_Muestras_archivo = 0;
   int sample = 0; //variable auxiliar para enviar los datos del archivo al Bus ISA
   int err;

   clrscr();
   frame();
   Imprime_menu_principal();
   estado = MENU;
   do
    {
     accion=0; ch=0; Cantidad_Muestras_archivo=0; sample=0;
     while (estado == MENU)
        {
         clrscr();
         frame();
         Imprime_menu_principal();
         ch = getch();
         if (ch=='q'||ch=='Q')  estado = SALIR;
         if (ch=='t'||ch=='T')  estado = TESTMANUAL;
         if (ch=='c'||ch=='C')  estado = TESTARCHIVO;
        } //end while estado==MENU
```

```
//-----------------------------
//         Test Manual
//-----------------------------
     if (estado == TESTMANUAL)
      {
       clrscr();
       frame();
       Imprime_menu_secundario(TESTMANUAL);
       accion = 0;

       MAIN_WIN;
       gotoxy(2,1);
       cprintf("Ingrese la cantidad de muestras a implementar (Max. 20): ");
       scanf("%d",&Sample_Qty);
       for (i=0; i<Sample_Qty; i++)
       {
        gotoxy(2,2);
        cprintf("Ingrese la muestra (direccion en decimal) Nro %d : ",i);
        scanf("%d",&Sample_Adrs);
        Vector_datos_archivo[i] = Sample_Adrs;
       } //end for i

       gotoxy(2,4);
       printf(" Cantidad de muestras a implementar: %d ",Sample_Qty);
       gotoxy(2,5);
       printf(" Muestras (en DECIMAL) ingresadas: ");
       gotoxy(2,6);
       for (i=0; i<Sample_Qty; i++)
       {
        printf("%d ",Vector_datos_archivo[i]);
/*      if (Vector_datos_archivo[i] > 8191)
         {
          textcolor(EGA_RED + BLINK);
          cprintf(" Fail!");
          gotoxy(20, 18);
          textcolor(EGA_RED);
          textbackground(0);
          gotoxy(2, 10);
          cprintf(" **** Muestra %d: %d mayor a 8191 **** ", i, Vector_datos_archivo[i]);
          break;
         }
*/     }
       ii=0;
       while (estado == TESTMANUAL)
       {
        habilitacion = 1;
        if (kbhit())
          {
            ch = getch();
            if (ch=='r'||ch=='R')
              {
                accion = RUN;
                Imprime_menu_run();
                outport(portw, 0);  //ENVIAR SI O SI para que arranque el modulador.
              }
            if (ch=='m'||ch=='M')  accion = MENU;
          } //end if (kbhit())
        switch(accion)
             {
              case RUN:
                 {
                  do
                   {if (ii == Sample_Qty) ii=0;    //inicializo ii para enviar de nuevo los
datos
                    if ( Enviar_dato_ISA(ii) < 0 )  //Envio dato al bus ISA
                      { accion = 0;
                        STATUS_WIN;
                        clrscr();
                        gotoxy(10, 2);
```

```
                    cprintf("ERROR: No habilita para escribir");
                    MAIN_WIN;
                   }
                  ii++;
                  } while (!kbhit() || (accion == 0));
                 accion = STOP;
                 estado = TESTMANUAL;
                 break;
                }
            case STOP:
                {
                 Detener_Modulador();
                   //     sample = 0;
                 Imprime_menu_stop();
                 estado = TESTMANUAL;
                 accion = 0;
                 MAIN_WIN;
                 break;
                }
            case MENU:
                {
                 estado = MENU;
                 break;
                }
             } // end switch(accion)
       } //end while(estado==TESTMANUAL)
      } //end if(estado == TESTMANUAL)



//-----------------------------
//     Test con Archivo
//-----------------------------
     if (estado == TESTARCHIVO)
      {
       frame();
       Imprime_menu_secundario(TESTARCHIVO);
       Cantidad_Muestras_archivo = Cargar_archivo();
       if (Cantidad_Muestras_archivo < 0)
       {
        STATUS_WIN;
        textcolor(EGA_RED);
        gotoxy(10, 1);
        cprintf("     Error en archivo      ");
        gotoxy(5, 2);
        cprintf("Presione una tecla para continuar...");
        while (!kbhit()) /* do nothing */ ;
        estado = MENU;
        MAIN_WIN;
       }
       accion = 0;
       while (estado == TESTARCHIVO)
       {
        habilitacion = 1;
        if (kbhit())
          {
           ch = getch();
           if (ch=='r'||ch=='R')
             {
              accion = RUN;
              Imprime_menu_run();
              outport(portw, 0);  //ENVIAR SI O SI para que
                                  // arranque el modulador.
             }
           if (ch=='m'||ch=='M')  accion = MENU;
          } // end kbhit
        switch(accion)
             {
```

```
            case RUN:
                {
                 do
                  {
                   if ( Enviar_dato_ISA(sample) < 0 )  //Envio dato al bus ISA
                   { accion = 0;
                     STATUS_WIN;
                     clrscr();
                     gotoxy(10, 2);
                     cprintf("ERROR: No habilita para escribir");
                     MAIN_WIN;
                    }
                   sample++;
                   if (sample == Cantidad_Muestras_archivo) sample = 0;
                   estado = TESTARCHIVO;
                  } while (!kbhit() || (accion == 0));
                 accion = STOP;
                 estado = TESTARCHIVO;
                 break;
                }
            case STOP:
                {
                 Detener_Modulador();
                 sample = 0;
                 Imprime_menu_stop();
                 estado = TESTARCHIVO;
                 accion = 0;
                 MAIN_WIN;
                 break;
                }
            case MENU:
                {
                 estado = MENU;
                 break;
                }
             } // end switch(accion)
       } //end while(estado==TESTARCHIVO)
      } //end if(estado==TESTARCHIVO)

   }while( estado != SALIR );  //estado = SALIR entonces salgo del programa
   return;
} //end de main



//=========================================================================
//        Funcion para cargar un archivo en un vector
// La funcion devuelve la cantidad de elementos que se cargaron en
// el vector (mayor que cero). Para el modulador son las cantidades
// de muestras que tiene el archivo.
// Codigo de errores:
//                -1: No puede abrir el archivo
//                -2: Cantidad de muestras mayor a 6000
//  Rev. 1.0:   emision    MT-31/08/2001
int Cargar_archivo(void)
{
   FILE *Puntero_leo;
   int dato_leido=0, i=0, ii=0;
   char name[30];
   int Cantidad_de_datos = 0;

   posicionx = 1;
   posiciony = 1;
   MAIN_WIN;
   textcolor(EGA_CYAN);
   gotoxy(posicionx, posiciony);
   cprintf("Ingrese el nombre del archivo a leer que puede tener hasta 6000");
   gotoxy(posicionx, posiciony+1);
   cprintf("muestras en HEXADECIMAL:   ");
```

```
   scanf("%20s",&name);
   gotoxy(posicionx, posiciony+3);
   cprintf("Leyendo archivo...    ");
   if((Puntero_leo = fopen(name,"r"))==NULL)
     {
      cprintf("Fail!");
      gotoxy(20, 18);
      textcolor(EGA_RED + BLINK);
      textbackground(0);
      cprintf(" **** No puedo abrir el archivo :  %s **** ",name);
      return -1;
     }
   cprintf("Ok!");
/*************************************************************
 * Cargo las muestras del archivo en el vector dato_salida_BUS *
 *************************************************************/
   gotoxy(posicionx, posiciony+4);
   cprintf("Cargando muestras en vector de salida...  ");
   while (!feof(Puntero_leo))
     {
      fscanf(Puntero_leo, "%x", &dato_leido);
      Vector_datos_archivo[i] = dato_leido;
      i++;
     }
   fclose(Puntero_leo);
   Cantidad_de_datos  = i-1; //Cantidad de muestras que tiene el archivo.
   cprintf("Ok!");
/*************************************************************
 *  Cuento las muestras a implementar.                       *
 *************************************************************/
   gotoxy(posicionx, posiciony+5);
   cprintf("Contando muestras en el vector de salida...  ");
   printf("%d  ", Cantidad_de_datos);
   if (i > 5999)
     {
      cprintf("Fail!");
      gotoxy(20, 18);
      textcolor(EGA_RED + BLINK);
      textbackground(0);
      cprintf(" **** Cantidad de muestras mayor a 6000 **** ");
      return -2;
     }
   cprintf("Ok!");  // Ok si las muestra son menores a 6000
/*************************************************************
 *  Imprimo las muestras a implementar.                      *
 *************************************************************/
   gotoxy(posicionx, posiciony+6);
   for (ii=0; ii<Cantidad_de_datos; ii++)
     {
      cprintf("%x ", Vector_datos_archivo[ii]);
/*      if (Vector_datos_archivo[ii] > 0x1fff)
       {
       textcolor(EGA_RED + BLINK);
       cprintf("Fail!");
       gotoxy(20, 18);
       textcolor(EGA_RED);
       textbackground(0);
       gotoxy(posicionx, posiciony+8);
       cprintf(" **** Muestra %d: %x mayor a 0x1fff **** ", ii, Vector_datos_archivo[ii]);
       return -2;
        }
*/   }
   cprintf("\n\rListo para empezar...   ");
   cprintf("Ok!");
   return Cantidad_de_datos;
} //end Cargar_archivo
```

```
//=====================================================================
//           Funcion para enviar un dato al Bus ISA.
// La funcion envia un dato a la direccion dada por "portw" del Bus
// ISA.
// El primer dato se envia sin leer el puerto, ya que la se¤al "habilitacion"
// se inicializa en 1. Una vez enviado el dato, ,sta se pasa a 0. Para
// enviar los datos siguientes, la lectura del puerto debe pasar a 1.
//
// Codigo de errores:
//  -1 : Escritura no permitida, senal habilitacion = 0 (despues de 10 lecturas)
//   0 : Dato enviado correctamente
//
//  Rev. 1.0: emision    MT-31/08/2001
//  Rev. 2.0: Incorporacion de errores  MT-12/01/2003
int Enviar_dato_ISA(int j)
{
  unsigned int xx;
/*********************************
 Mando los datos al BUS de la PC.
*********************************/
// int habilitacion = 1; //OJO!!! habilitacion=1 no puede definirse aca porque
                  //esta funcion se llama para cada dato.
    xx = 0;
    MAIN_WIN;
    while(habilitacion == 0) //Espero que el modulador habilite para poder
       {                 //escribir un dato.
        habilitacion = 1 & inport(portr); //Con 1 enmascaro el dato leido del
                                  //puerto, dejando que pase solo el
        //printf("%x: %d \n", inport(portr), habilitacion); // OJO, la impresion en pantalla le
                                                            // toma 0,4mseg => detiene al modulador
        xx = xx + 1;
        if (xx > 120)  return -1;
       }                          //dato menos significativo D0.

    outport(portw, Vector_datos_archivo[j]);
    habilitacion = 0;

//****SOLO PARA PROBAR EL BUCLE DE ENVIO DE DATOS****
//   cprintf("\n\r       Enviando Muestra Nro %d : %d", j, Vector_datos_archivo[j]);
//   delay(3);
    return 0;
} // end Enviar_dato_ISA




//=====================================================================
//    Funcion para detener el funcionamiento del modulador
// La funcion envia el dato "C000" a la direccion dada por "portw"
// del Bus ISA.
// El dato C000Hex = 57344Dec pone en "1" a las lineas LD13, LD14 y LD15
// del Bus de Datos. En esa condicion se detiene el funcionamiento del
// contador "C", deteniendo al modulador.
//  Rev. 1.0:   emision      MT-31/08/2001
void Detener_Modulador(void)
{
//    int stop = 49152;    //Dato de parada C000 (Poner LD15=LD14=1)
    int stop = 57344;    //Dato de parada E000 (Poner LD15=LD14=LD13=1)
    outport(portw, stop); //Detengo el funcionamiento del modulador
}//end Detener_Modulador
```

```
/*********************************************************************
                                  Utilmenu.c
*********************************************************************/

#include <GRAPHICS.H>
#include <conio.h>
#include <stdio.h>

#define SALIR 1
#define TESTMANUAL 2
#define TESTARCHIVO 3
#define MENU 4

//Definicion de ventanas
#define MENU_WIN     window(2,21,27,23)
#define STATUS_WIN    window(29,21,79,23)
#define MAIN_WIN      window(2,4,79,19)
#define TITLE_WIN     window(1,1,79,3)


void frame(void)
{
   window(1,1,80,25); //Maximo tama¤o del monitor
   textbackground(BLUE);
   textcolor(CYAN);
   clrscr();
   textbackground(BLUE);
   gotoxy(1, 1);
    printf("╔══════════════════════════════════════════════════════════════════════════════╗");
    printf("║                                                                              ║");
// TITLE_WIN
    printf("╠══════════════════════════════════════════════════════════════════════════════╣");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
// MAIN_WIN
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("║                                                                              ║");
    printf("╠═══════════════════════════╦══════════════════════════════════════════════════╣");
    printf("║                           ║                                                  ║");
    printf("║                           ║                                                  ║");
// MENU_WIN  and STATUS_WIN
    printf("╚═══════════════════════════╩══════════════════════════════════════════════════╝");
          //12345678901234567890123456789012345678901234567890123456789012345678901234567890
          //         10        20        30        40        50        60        70        80

   //textcolor(MAGENTA);
   TITLE_WIN;
   textcolor(EGA_LIGHTMAGENTA);
   gotoxy(6, 2);
   cprintf(" LEICI-UNLP ");
   gotoxy(22, 2);
   cprintf(" TEST UNIVERSAL SPACE VECTOR MODULATOR ");
   gotoxy(65, 2);
   cprintf(" Ver 2.0 ");
}
```

```
void Imprime_menu_principal(void)
{
    int posicionx;
    int posiciony;

    MENU_WIN;
    posicionx = 2;
    posiciony = 1;

    textcolor(EGA_CYAN);
    gotoxy(posicionx,posiciony);
    cprintf(" : Test Manual");
    gotoxy(posicionx,posiciony+1);
    cprintf(" : Cargar Archivo");
    gotoxy(posicionx,posiciony+2);
    cprintf(" : Salir");

    textcolor(EGA_YELLOW);
    gotoxy(posicionx,posiciony);
    cprintf("T");
    gotoxy(posicionx,posiciony+1);
    cprintf("C");
    gotoxy(posicionx,posiciony+2);
    cprintf("Q");
    MAIN_WIN;
}


void Imprime_menu_secundario(int opcion)
{
    int posicionx;
    int posiciony;

    MENU_WIN;
    clrscr();
    posicionx = 2;
    posiciony = 1;
    textcolor(EGA_CYAN);
    gotoxy(posicionx,posiciony);
    cprintf("R: RUN");
    gotoxy(posicionx,posiciony+1);
    cprintf("M: Menu principal");

    textcolor(EGA_YELLOW);
    gotoxy(posicionx,posiciony);
    cprintf("R");
    gotoxy(posicionx,posiciony+1);
    cprintf("M");

    STATUS_WIN;
    clrscr();
    gotoxy(9,2);
    if (opcion == TESTMANUAL) cprintf("        Test Manual        ");
            else            cprintf(" Test con archivo definido ");
    MAIN_WIN;
}


void Imprime_menu_run(void)
{
    int posicionx;
    int posiciony;

    MENU_WIN;
    clrscr();
    posicionx = 2;
    posiciony = 1;
    textcolor(EGA_YELLOW);
    gotoxy(posicionx,posiciony);
```

```
    cprintf("Presione una tecla");
    gotoxy(posicionx,posiciony+1);
    cprintf("para parar ...");

    STATUS_WIN;
    clrscr();
    textcolor(EGA_CYAN);
    gotoxy(10,2);
    cprintf("Test corriendo ...");
    MAIN_WIN;
}


void Imprime_menu_stop(void)
{
    int posicionx;
    int posiciony;

    MENU_WIN;
    clrscr();
    posicionx = 2;
    posiciony = 1;
    textcolor(EGA_CYAN);
    gotoxy(posicionx,posiciony);
    cprintf("R: RUN");
    gotoxy(posicionx,posiciony+1);
    cprintf("M: Menu principal");

    textcolor(EGA_YELLOW);
    gotoxy(posicionx,posiciony);
    cprintf("R");
    gotoxy(posicionx,posiciony+1);
    cprintf("M");

    STATUS_WIN;
    clrscr();
    textcolor(EGA_RED);
    gotoxy(10,2);
    cprintf("Test detenido !!!");
    MAIN_WIN;
}
```