IMPLEMENTACIÓN DE TÉCNICAS SPWM EN FPGA DE CÓDIGO ABIERTO Y USO DE VERILOG

IMPLEMENTATION OF SPWM TECHNIQUES IN OPEN-SOURCE FPGA AND USE OF VERILOG

Benjamín Chavarría Domínguez
Tecnológico Nacional de México / CENIDET, México
d18ce092@cenidet.tecnm.mx

Jesús Aguayo Alquicira
Tecnológico Nacional de México / CENIDET, México
jesus.aa@cenidet.tecnm.mx

Susana Estefany de León Aldaco
Tecnológico Nacional de México / CENIDET, México
susana.da@cenidet.tecnm.mx

Fernando Chavarría Domínguez
Universidad Veracruzana / Campus Coatzacoalcos, México
fchavarria@uv.mx

Ricardo Eliú Lozoya Ponce
Tecnológico Nacional de México / IT de Chihuahua, México
ricardo.lp@chihuahua.tecnm.mx

Recepción: 22/noviembre/2022 **Aceptación:** 22/diciembre/2022

Resumen

Los CHB-MLI son una topología con numerosos interruptores debido a sus múltiples puentes H. Esta característica permite dividir la potencia procesada por su carga en varias secciones. Por lo cual, los CHB-MLI pueden procesar potencias más elevadas con interruptores menos robustos, comparado a otros inversores. Sin embargo, es necesaria una forma fácil de crear los pulsos de comando y la generación de estos no debe presentar retardos o desfaces que afecten la forma de onda resultante.

Este trabajo expone la simulación de cuatro variantes de técnicas SPWM desde Simulink como una alternativa para la creación de numerosos pulsos de comando. Se emplea una FPGA capaz generar pulsos en el orden de los megahertz sin retardos o desfaces. Se detallan las consideraciones para programar los pulsos en

un entorno de código abierto basado en Verilog. A diferencia de trabajos previos basados en IDEs de Xilinx y Altera, sujetos a licencia.

Palabras Clave: FPGA, Verilog, inversores multinivel, SPWM.

Abstract

CHB-MLIs are a topology with numerous switches due to their multiple H-bridges. This feature allows the power processed by its load to be divided into multiple sections. Therefore, the CHB-MLI can process higher powers with less robust switches, compared to other inverters. However, an easy way to create the command pulses is necessary and the generation of these must not present delays or gaps that affect the resulting waveform.

This work exposes the simulation of four variants of SPWM techniques from Simulink as an alternative for the creation of numerous command pulses. An FPGA capable of generating pulses in the order of megahertz without delays or gaps is used. Considerations for programming pulses in an open source environment based on Verilog are detailed. Unlike previous works based on Xilinx and Altera IDEs, subject to license.

Keywords: *FPGA, Verilog, multilevel inverter, SPWM.*

1. Introducción

En el rubro del procesamiento de energía eléctrica, dominado por la electrónica de potencia, existen cuatro procesos de conversión de corriente, figura 1.

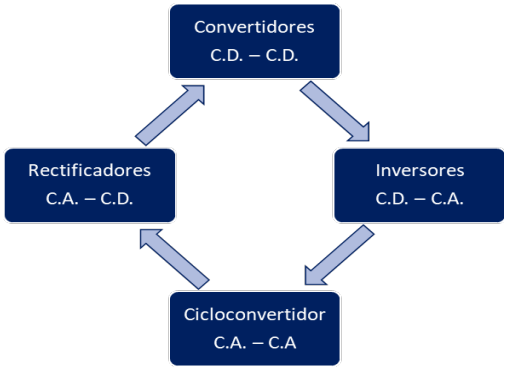
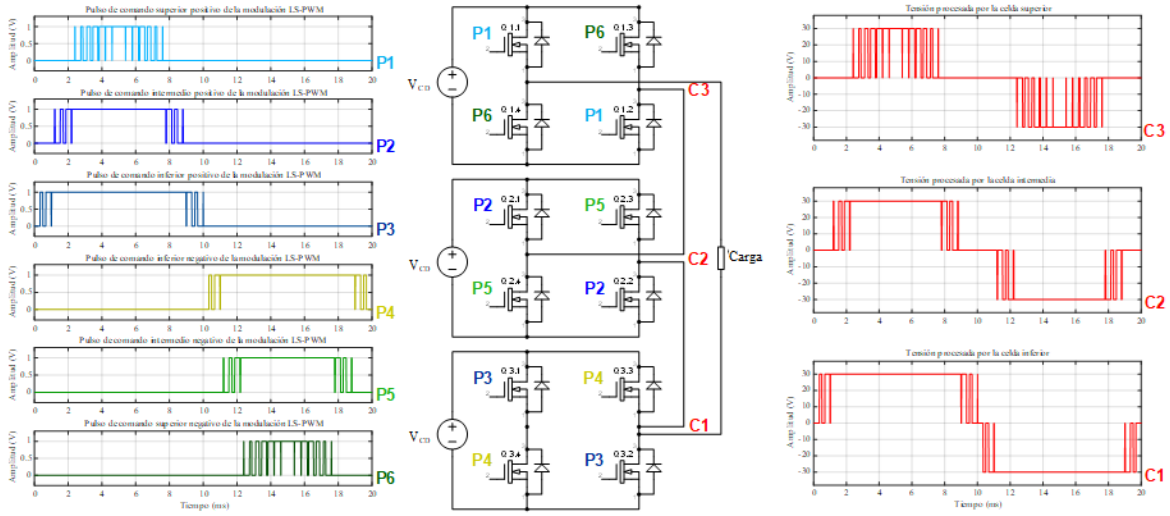


Figura 1 Relación entre la conversión de corriente y la topología asociada.

Como muestra la figura 1, para cada tipo de proceso existe una topología asignada, en el caso de la conversión de CD a CA se denominan inversores. Se han propuesto diferentes variantes de la topología para su desarrollo y construcción, siendo los inversores basados en puentes H de los más empleados.

Por su parte, existen variantes de inversores de puentes según su aplicación, por ejemplo: para motores de tres fases se utilizan inversores de puente H trifásicos. Mientras que para reducir el contenido armónico se usan los CHB-MLI (del inglés: *Cascaded H-Bridge Multilevel Inverter*).

Uno de los inconvenientes que presentan los CHB-MLI radica en la forma de controlar la conversión de corriente en sus múltiples celdas. Es necesario generar una gran cantidad de pulsos de comando de manera fiable y sin tiempos de retardo. Comúnmente la generación de estos pulsos de comando es mediante el uso de técnicas de modulación, como es el caso de las técnicas SPWM (del inglés: *Sinusoidal Pulse Width Modulation*), figura 2.



a) Pulsos de comando *b) CHB-MLI de 7 niveles* *c) Tensiones por celda*

Figura 2 Técnica de modulación SPWM (tipo LS-PWM) aplicada a un CHB-MLI.

En este trabajo se aborda una metodología para la programación de los pulsos de comando dentro de una FPGA (del inglés: *Field-Programmable Gate Array*). Estos dispositivos digitales poseen la capacidad de operar con mucha precisión la ejecución de múltiples trenes de pulsos sin presentar retardos o desfaces.

Revisión de trabajos previos

Este apartado expone algunos artículos que sirven de antecedente a las consideraciones y procedimientos de este trabajo. La tabla 1 enlista algunas de las características relevantes de los artículos que resultan de mayor interés.

Tabla 1 Revisión de trabajos enfocados a implementar técnicas SPWM en FPGA.

Núm.	Título	Año	Revista / Conferencia	- Versión de FPGA - Lenguaje	Modulación
1	*FPGA-based High-Definition SPWM Generation with Harmonic Mitigation Property for Voltage Source* [Sarker, 2021]	2021	*IEEE: Transactions on Industrial Informatics*	- Xilinx, Spartan 3 - VHDL	SPWM, simple monofásica
2	*Research and Implementation of Natural Sampling SPWM Digital Method for Three-Level Inverter of Photovoltaic Power Generation System Based on FPGA* [Zhan, 2019]	2019	*IEEE: Access*	- Altera cyclone III - VHDL	SPWM, simple
3	*A Digitally Controlled Low-EMI SPWM Generation Method for Inverter Applications* [Liou, 2014]	2014	*IEEE: Transactions on Industrial Informatics*	- Altera, Cyclone II - Verilog	SPWM simple monofásica
4	*Development of an FPGA-Based SPWM Generator for High Switching Frequency DC/AC Inverters* [Lakka, 2014]	2014	*IEEE: Transactions on Power Electronics*	- Xilinx, Virtex 5 - VHDL	SPWM simple monofásica
5	*Generation of a multilevel SPWM technique of 3, 9 and 21 levels with FPGAs* [Salgado-Herrera, 2013]	2013	*IEEE: 2013 North American Power Symposium (NAPS)*	- Xilinx, Spartan 3E - VHDL	SPWM, tipo LS-PWM monofásica

Existen más trabajos relacionados a la implementación en FPGA de técnicas de modulación para inversores multinivel. Pero los 5 cinco artículos de la tabla 1 permiten ver la tendencia general en los modelos de FPGAs utilizadas y sus fabricantes, así como el lenguaje de mayor uso. En lo referente a técnicas de modulación, por lo general se emplean versiones sencillas de técnicas SPWM que constan de una sola señal portadora y moduladora. Los trabajos listados emplean IDEs de las empresas Xilinx y Altera para escribir código, quedando sujetos al uso de licencias y sus tarjetas de desarrollo oficiales.

Tomando en cuenta lo anterior, el objetivo de este trabajo consiste en presentar una metodología y consideraciones para el correcto ajuste de una serie de técnicas SPWM, que involucran el uso de múltiples portadoras. Es decir, modulaciones más complejas que las abordadas por la mayoría de trabajos enlistados. También se propone un esquema donde se desarrolla código escrito en lenguaje Verilog (lenguaje usado por la minoría de los trabajos listados) junto con programación de bloques visuales. Y se emplea una FPGA compatible con el software Icestudio de código abierto.

Tarjeta de desarrollo TinyFPGA BX

La figura 3 muestra un esquema de la tarjeta de desarrollo TinyFPGA BX, la ubicación de algunas secciones importantes, la distribución y nombre de los pines asignados por la tarjeta de desarrollo y por su FPGA.

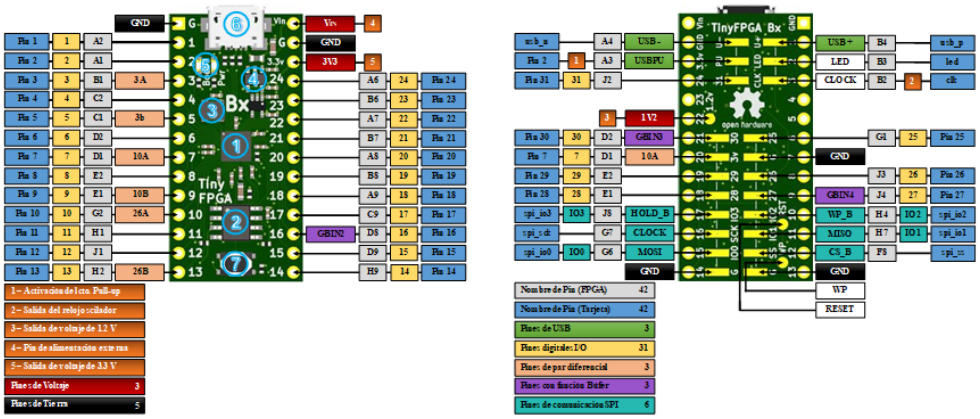


Figura 3 Distribución de pines y elementos en la tarjeta TinyFPGA BX.

En la parte frontal de la tarjeta de desarrollo se señalan en un círculo y número azul los componentes más importantes de la tarjeta de desarrollo:

- FPGA: Ubicación de la FPGA iCE40 LP/HX1K en la tarjeta de desarrollo.
- Memoria flash SPI: Almacena el diseño digital y otros datos importantes de la programación mientras la FPGA se encuentre sin energía eléctrica.
- Reloj oscilador: Ubicación del oscilador dentro de la tarjeta de desarrollo, este reloj de 16 MHz es empleado como oscilador principal por la FPGA.
- Regulador de tensión: Este dispositivo recibe suministro eléctrico del puerto USB y pin VIN, dicha tensión es regulado para la FPGA en valores de 1.3 V utilizado por los Bloques lógicos y 3.3 V utilizado por los Bloques I/O.
- Indicadores LED: Señalan el estado de la FPGA y suministro eléctrico.
- Conexión USB: Posee un gestor de arranque para la programación de la FPGA al cargar un nuevo diseño digital o para regrabar en la FPGA el diseño almacenado en la memoria SPI al energizar la tarjeta de desarrollo.
- Botón de reinicio: Permite activar manualmente la ejecución del gestor de arranque, reestableciendo el estado de la FPGA.

FPGA iCE40

La figura 4 presenta la arquitectura interna de la FPGA iCE40 LP/HX1K que se encuentra en la tarjeta de desarrollo Tiny FPGA BX.

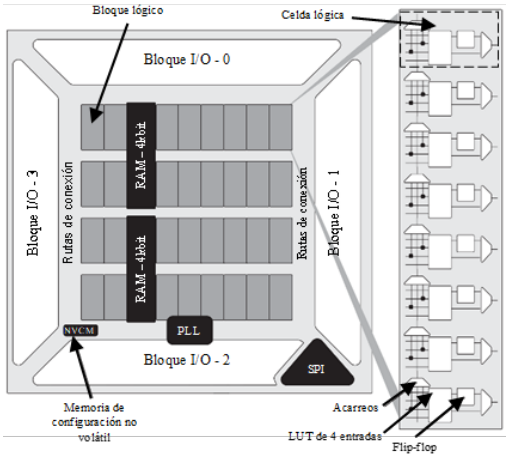


Figura 4 Arquitectura interna de la FPGA iCE40 LP/HX1K.

Se identifican las secciones de Bloques I/O, los Bloques lógicos formados por ocho celdas lógicas y las Rutas de conexión. También se observan elementos de función específica como un bloque para generar dos PLL, memorias de tipo volátil y no volátil, además de un bloque de comunicación SPI.

Lenguaje Verilog

El código consta de dos subsecciones, figura 5. El submódulo 1 define y nombra los puertos de entrada y salida, mientras que el submódulo 2 brinda la descripción del comportamiento y estructura del circuito [LaMeres, 2019a].

```
1  module Mux_2_To_1 (input i_Select,
2                     input i_Data1,
3                     input i_Data2,
4                     output o_Data);
5
6  assign o_Data = i_Select ? i_Data1 : i_Data2;
7
8  endmodule // Mux_2_To_1
```

Submódulo 1

Submódulo 2

Módulo

Figura 5 Código desarrollado en lenguaje Verilog para un MUX de 2 entradas y 1 salida.

Métodos

Parámetros considerados previos a la programación

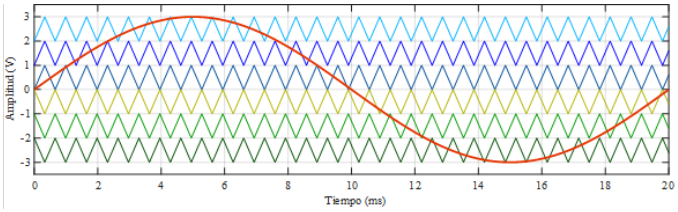
Los requerimientos y características que debe tener el circuito digital desarrollado en la FPGA se presentan en los siguientes puntos:

- Un circuito digital que genere los pulsos de conmutación resultantes de las técnicas de modulación a frecuencias superiores de los 2.5 kHz.
- Cada pulso de comando generado requiere una versión complementaria con un grado de desface, para la correcta conmutación de los puentes H.
- La ejecución de todos los pulsos de comando con sus complementarios debe de estar sincronizada y no se pueden presentar desfases imprevistos.
- Todos los pulsos de comando deben ser una secuencia de estados altos y bajos dentro de un periodo de 20 ms o 50 Hz.

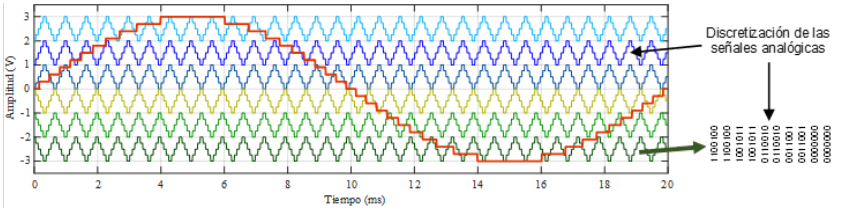
Métodos de discretización para las señales y pulsos de comando

Las técnicas de modulación SPWM consisten en la comparación de señales triangulares portadoras con respecto a una señal senoidal moduladora. Estas señales son de tipo analógico, por lo que no pueden ser procesadas dentro de una FPGA de forma directa. Son necesarias estrategias de discretización de señales analógicas que permitan el desarrollo del generador de pulsos de comando empleando dispositivos digitales. A continuación, se presentan dos métodos:

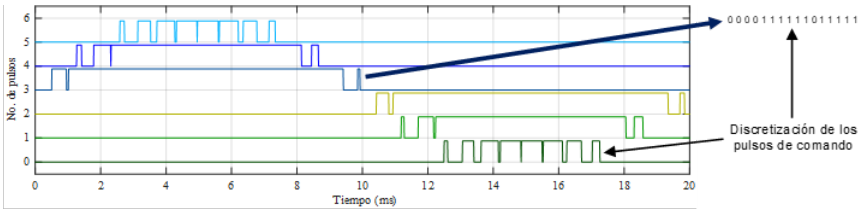
- **Discretización de las señales analógicas SPWM:** Las señales son muestreadas a partir de dos criterios. Primero se asigna una determinada lista de “palabras binarias” al ciclo de cada señal. El segundo consiste en representar mediante el valor de las palabras binarias la amplitud que la señal discretizada posee en un determinado momento, ver la figura 6b [Romli, 2008].
- **Discretización directa de los pulsos de comando:** La amplitud de los pulsos es representada con estados lógicos (1 y 0 binario), esta representación es una sucesión continua de estados que cubren un ciclo del pulso, figura 6c. Por lo tanto, solo se utiliza una palabra binaria para cada pulso [Afarulrazi, 2010].



a) Señal SPWM original



b) Discretización de las señales analógicas



c) Discretización directa de los pulsos de comando

Figura 6 Métodos de discretización tomando como ejemplo una técnica de modulación.

Parámetros de la simulación, discretización y divisor de frecuencia

Se eligió el método de discretización directa por su sencilles. Se consideran los datos presentados en la tabla 2 para efectuar dichos cálculos y las simulaciones.

Tabla 2 Características de la FPGA y parámetros de las modulaciones SPWM.

Características del FPGA	Parámetros de las modulaciones
- FPGA de la empresa Lattice basada en el modelo ICE40 LP/HX1K. - Compatibilidad con el IDE de Icestudio y lenguaje HDL Verilog. - Sistema embebido basado en la tarjeta de desarrollo TinyFPGA BX. - Reloj oscilador principal de 16 MHz. - 7,864 LUTs y 128 kb de RAM. - 31 salidas o entradas digitales.	- Moduladora de 50 Hz y 6 Vpp. - 6 portadoras de 2.5 kHz y 1 Vpp para las técnicas LS-PWM, PWM Rotativo y PWM Distribuido - 6 portadoras de 2.5 kHz y 6 Vpp para la técnica PS-PWM. - Índice de modulación en amplitud de 1 e índice de modulación en frecuencia de 50. - 1,000 muestras por ciclo. - Simulación de 20 ms para un ciclo de moduladora (equivalente a 50 Hz).

Verificación de los valores de muestreo

Para integrar los pulsos de comando a la FPGA, es necesario obtener de cada simulación seis vectores de datos correspondientes a los seis pulsos de comando, cada vector de datos almacena mil muestras del respectivo pulso de comando.

Una correcta discretización de los pulsos de comando considera que la división del periodo por ciclo de los pulsos entre el número de muestras para cada vector no de como resultado un número irracional. Esta operación es denominada como “Factor de no irracionalidad”. La ecuación 1 efectúa esta comprobación con el periodo de 20 ms y las mil muestras de los vectores de datos.

$$F_r = \frac{t_p}{N_m} = \frac{20\ ms}{1{,}000\ muestras} = 20\ \mu s/muestra \qquad (1)$$

Donde:

F_r: Factor de no irracionalidad.

t_p: Periodo por ciclo de los pulsos de comando.

N_m: Número de muestras de los vectores de datos.

También se verifica que la división del valor de frecuencia del oscilador entre el número de muestras de cada vector resulte en un valor no irracional y submúltiplo de la frecuencia del oscilador, se asigna el término “Factor submúltiplo” para esta operación. La ecuación 2 corrobora dicha operación considerando al reloj oscilador de 16 MHz de la tarjeta TinyFPGA BX y las mil muestras de cada vector.

$$F_s = \frac{f_o}{N_m} = \frac{16\ MHz}{1{,}000\ muestras} = 16\ KHz/muestra \qquad (2)$$

Donde:

F_s: Factor submúltiplo.

f_o: Frecuencia del oscilador en la tarjeta de desarrollo.

N_m: Número de muestras de los vectores de datos.

Cálculo del divisor de frecuencia

Los pulsos de comando generados por la FPGA deben tener una frecuencia por ciclo de 50 Hz, esto se consigue modificando la frecuencia base entregada por el oscilador en la tarjeta de desarrollo con un “Divisor de frecuencia”.

Para desarrollar el divisor de frecuencia es necesario calcular: El “Valor del divisor de frecuencia”, que se obtiene al multiplicar la frecuencia por ciclo de los pulsos de comando a la salida del FPGA con el número de muestras de los vectores de datos.

La ecuación 3 realiza esta operación considerando la frecuencia de 50 Hz y las mil muestras de los vectores de datos.

$$V_d = f_c * N_m = 50\ Hz * 1{,}000\ muestras = 50\ kHz \quad (3)$$

Donde:

V_d: Valor del divisor de frecuencia.

f_c: Frecuencia por ciclo para los pulsos de comando a la salida del FPGA.

N_m: Número de muestras.

El segundo cálculo denominado "Factor del divisor de frecuencia" es el número que se utiliza en la sección del código HDL llamada divisor de frecuencia y se obtiene mediante la división de la frecuencia del oscilador montado en la FPGA, sobre el valor del divisor de frecuencia obtenido previamente. La ecuación 4 realiza esta operación considerando los 16 MHz del reloj montado en la tarjeta de desarrollo y el resultado obtenido en la ecuación 3.

$$F_d = \frac{f_o}{V_d} = \frac{16\ MHz}{50\ kHz} = 320 \quad (4)$$

Donde:

F_d: Factor del divisor de frecuencia.

f_o: Frecuencia del oscilador en la tarjeta de desarrollo.

V_d: Valor del divisor de frecuencia.

Simulación y discretización de los pulsos de comando

Las simulaciones se desarrollaron en Simulink con los valores expuestos en la tabla 2. La figura 7 muestra los bloques empleados para la generación de las técnicas de modulación [SankaraKumar, 2018]. La señal senoidal moduladora es generada con el bloque "*Sine Wave*" y se conecta en la entrada de los seis bloques de comparación. En la otra entrada de los bloques de comparación se conecta una de las seis señales triangulares portadoras generadas con los bloques "*Repeating Sequence*". Los pulsos resultantes de las comparaciones se procesan con los bloques "*Data Type Conversion*". Estos bloques convierten los valores booleanos entregados por los bloques de comparación a un tipo de dato numérico "*single*" que

representa los estados alto y bajo como valores binarios de “1” y “0” respectivamente.

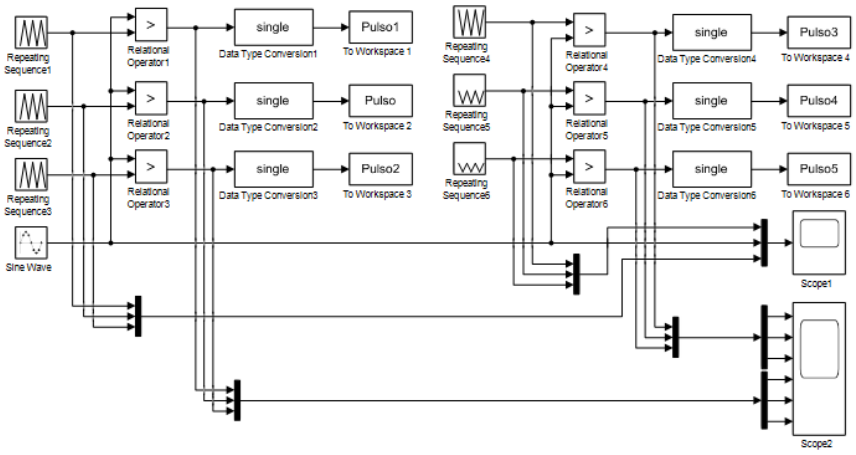


Figura 7 Bloques usados en Simulink.

El resultado de esta conversión es entregado a los bloques “*To Workspace*”, que almacenan como vectores de datos los valores binarios “1” y “0” que conforman a los pulsos de comando provenientes de las comparaciones.

Los parámetros internos de los bloques fueron ajustados según la técnica de simulada. Las técnicas SPWM convencional y PWM Distribuido emplean los mismos pulsos de comando, debido a que la técnica PWM Distribuido consiste en una técnica SPWM convencional que reasigna sus pulsos de comando después de terminar un periodo de 20 ms.

Tratamiento de los pulsos de comando discretizados

Los seis vectores de datos (con la información de los pulsos de comando discretizados gracias a los bloques “*Data Type Conversion*” y “*To Workspace*”) son visualizados con la ventana “*Workspace*” de Matlab. El vector se representa como una tabla de mil celdas donde se almacena sus valores binarios, ver la figura 8.

En una hoja de cálculo del programa Excel se despliega la columna con la información del vector de datos. Las columnas aledañas son ocupadas por un grupo de caracteres que al ser integrados con la información del vector de datos terminan por formar una lista de instrucciones que presentan la siguiente estructura: “10'd0:

S <= (valor binario del vector);". Esta nomenclatura es utilizada por el lenguaje Verilog como una instrucción, figura 8.

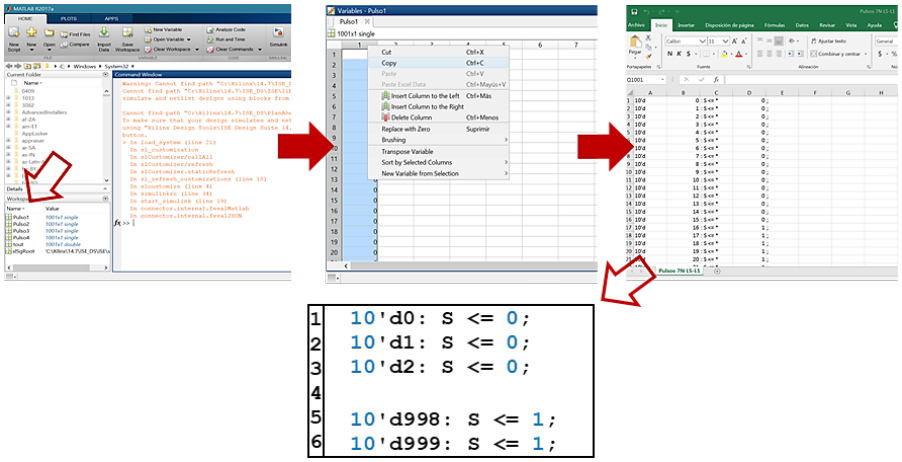


Figura 8 Pasos del tratamiento de los pulsos de comando discretizados.

El procedimiento antes descrito es aplicado a los seis vectores de datos de cada una de las cuatro técnicas de modulación estudiadas en este trabajo.

Al desarrollar las listas de instrucciones dentro la hoja de cálculo de Excel, se crean los registros que se emplearán en el entorno de programación de Icestudio. Estas instrucciones (mil por cada vector) describen al compilador la forma de los pulsos de comando que serán sintetizados e implementados en la FPGA iCE40 mediante circuitos lógicos.

Programación del generador de pulsos de comando

Descripción del IDE de Icestudio

La figura 9 presenta la interfaz del entorno de programación de Icestudio, los recuadros señalan las distintas secciones que la conforman. En el menú de componentes se dispone de los elementos para descripción de hardware (bloques para escribir código, bloques de configuración de puertos, compuertas lógicas, entre otros). Mientras que en el submenú "Herramientas" se encuentra las funciones de sintetizado y programación de la FPGA. En la parte inferior de la interfaz se presentan datos del código desarrollado como: El nombre del proyecto, tipo y cantidad de elementos lógicos necesarios para implementar, además del modelo de

FPGA para la que se realizó el sintetizado del código. En la figura 9 también se desarrolla un circuito combinacional mediante dos métodos: El primer método emplea un bloque para la escritura de código en lenguaje Verilog, mediante la técnica de "modelo estructural" se describen las conexiones de los elementos que integran al circuito. Los códigos en Verilog se constituyen dentro de un módulo dividido en dos submódulos, figura 5. En este caso el bloque define internamente la apertura y cierre del módulo, así como la definición de entradas y salidas (tarea realizada en el submódulo 1). El segundo método emplea simbología de compuertas lógicas para desarrollar el mismo circuito combinacional descrito en el bloque del primer método.

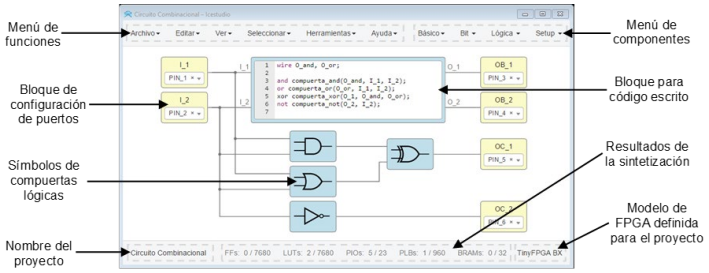


Figura 9 Entorno de programación de Icestudio.

Esquema de programación del generador de pulsos de comando

Se desarrollaron cuatro proyectos para programar los pulsos de comando de las técnicas de modulación estudiadas. La estructura de programación utilizada se basa en un esquema de bloques, figura 10.

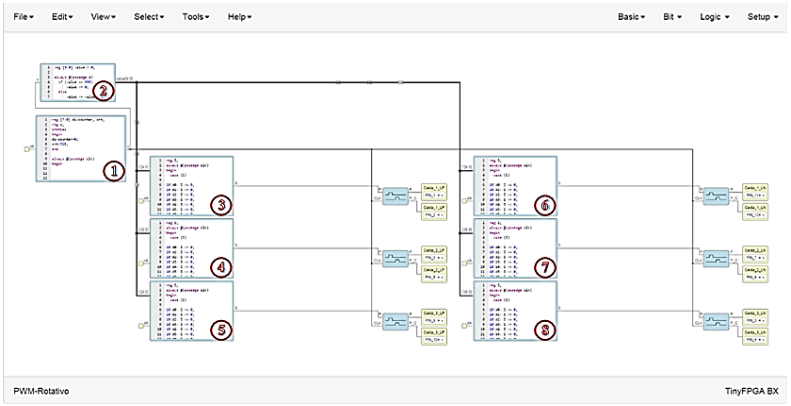


Figura 10 Esquema de programación para las técnicas SPWM, PWM Rotativo y PS-PWM.

La técnica PWM Distribuido consiste en una técnica SPWM que reasigna sus pulsos de comando después de terminar un periodo de 20 ms. Para cumplir esta función se agregan unos bloques extra al esquema de programación inicial, figura 11.

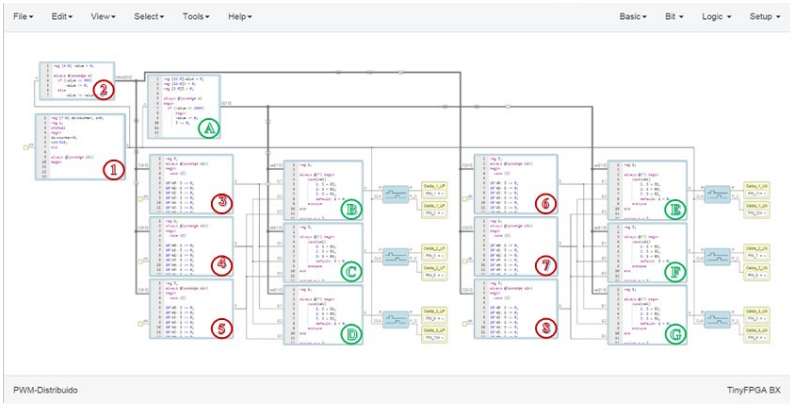


Figura 11 Esquema de programación para la técnica PWM Distribuido.

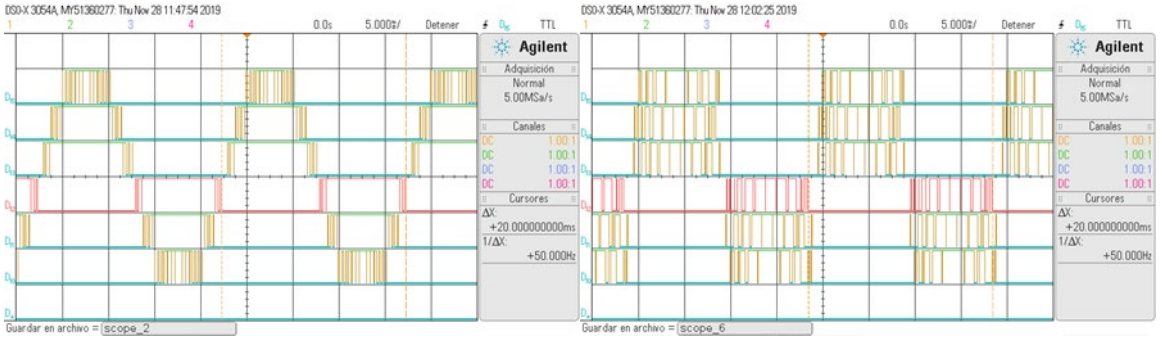
A continuación, se describen los bloques de los esquemas de programación:

- **Divisor de frecuencia:** El bloque “1” cumple la función de ajustar la frecuencia de 16 MHz del oscilador principal de la tarjeta de desarrollo, a una frecuencia de 50 kHz. El ajuste de frecuencia es necesario para las listas de instrucciones que almacena en la FPGA la información de los pulsos de comando.
- **Memorias de instrucciones:** Los bloques del “3” al “8” son denominados “Memorias”, ya que almacenan las listas de instrucciones. Las listas se ejecutan de forma paralela y cada una de ellas posee mil espacios de memoria donde se encuentran los estados binarios resultantes de la discretización de los pulsos de comando.
- **Contador:** El bloque “2” desarrolla un contador de 10 bits necesario para generar mil pasos durante un periodo de 20 ms (o 50 Hz). Cada paso acciona consecutivamente un espacio de memoria en cada uno de los bloques de memoria, esto permite la ejecución en paralelo de los seis pulsos de comando almacenados. El contador se desborda y reinicia al llegar al número “999” en binario ya que el conteo inicia desde el número “0” en binario.

- **Generadores de pulsos complementarios:** Los bloques azules con un símbolo de dos señales desfasadas, generan un pulso de comando complementario al pulso entregado por el bloque de “Memoria”. También generan un tiempo muerto entre el pulso del bloque de memoria y el nuevo pulso, esto evita estados de corto circuito en los interruptores del puente H.
- **Pines de salida:** Los bloques de tonalidad amarilla definen los pines de la tarjeta de desarrollo por donde se emiten los pulsos de comando entregados por los bloques de “Memoria” o por los bloques “Generadores de pulso”.
- **Etapa de Multiplexores:** Corresponde a la sección especial utilizada solo para la técnica PWM Distribuido y se encuentra constituida por los bloques señalados con letras en color verde. El bloque “A” es un contador de 12 bits corriendo de “0” a “2999” en binario. Este contador cada 1000 pasos ordena una reasignación de posición a los bloques “B” a “G”. La reasignación de posiciones permite multiplexar la información entregada por los bloques de “Memoria”, de esta manera los pulsos son redistribuidos al finalizar cada periodo de 20 ms.

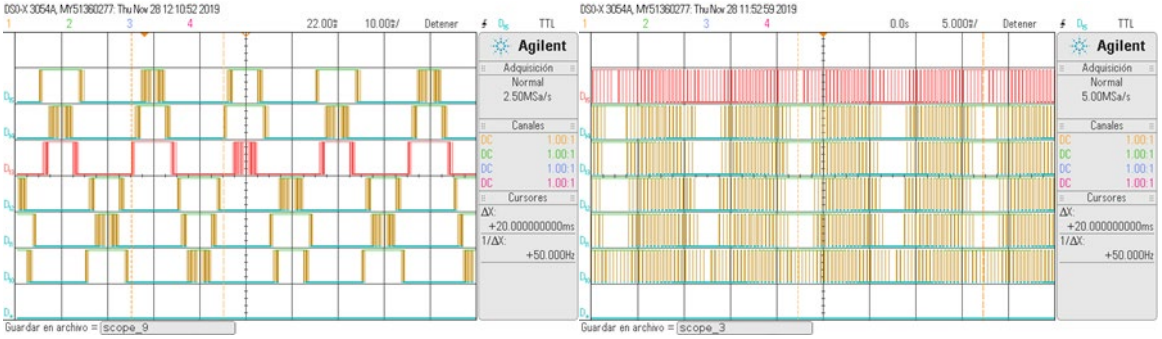
3. Resultados

Después de implementar el código descrito en el apartado anterior, se verificó su correcta operación. Con el analizador digital del osciloscopio Agilent modelo DSO-X 3054A se midieron los pines de la tarjeta de desarrollo asignados para la emisión de los pulsos comando. Las figuras 12 y 13 presentan las lecturas obtenidas.



a) Técnica SPWM tipo LS-PWM *b) Técnica PWM Rotativo*

Figura 12 Lecturas de los pulsos de comando generados por la FPGA.



a) Señal de tensión resultante *b) Contenido armónico*

Figura 13 Lecturas de los pulsos de comando generados por la FPGA.

Empleando los cursores del osciloscopio se corroboró que los pulsos de comando resultantes tienen un periodo de 20 ms, equivalente a una frecuencia de 50 Hz. Dicho periodo se encuentra en concordancia con el valor definido en la tabla 2, también permite comprobar la correcta operación del divisor de frecuencia. Cabe destacar que, en las cuatro lecturas, solo se visualizan los pulsos de comando originales, sin sus respectivas contrapartes complementarias.

4. Discusión

Antes de elegir un método de discretización, es necesario tomar en cuenta las siguientes observaciones para cada uno de las propuestas planteadas:

La técnica de discretización de las señales analógicas SPWM permite desarrollar un circuito digital capaz de variar las características de las señales discretizadas. Brindando la posibilidad de modular los pulsos resultantes, una función que sería de gran utilidad para sistemas de control de lazo cerrado. Debido a que esta característica brinda la capacidad de reajustar y modificar la respuesta a la salida del CHB-MLI en función de la acción de control.

Es necesario que la discretización de las señales SPWM posean una cantidad de muestras adecuada que le permita tener una representación cercana a su forma analógica. De otro modo los pulsos generados por esta discretización no corresponderán a su equivalente analógico, lo que puede ocasionar la generación de una señal deficiente a la salida del CHB-MLI. La discretización directa de los pulsos de comando solo requiere de una palabra binaria para representar los

estados altos y bajos, el número de componentes lógicos utilizados en su implementación dentro de una FPGA es menor que los necesarios para la discretización de señales analógicas. Sin embargo, al guardar directamente los pulsos generados por una configuración fija de la modulación SPWM, estos pulsos no se pueden usar por sistemas de control de lazo cerrado.

5. Conclusiones

La discretización de señales debe ser lo más cercano posible a las formas de onda originales para evitar distorsiones. Es necesario seleccionar adecuadamente el número de muestras que conformaran a cada señal discretizada, estas deben ser múltiplos entre sí. También se debe cuidar el redondeo de valores decimales al discretizar. Al momento de definir las frecuencias de las señales discretizadas dentro del FPGA se tiene que prever que los valores de división del reloj principal para los sudrelojes de las señales discretizadas (divisores de frecuencia) sean submúltiplos exactos del valor del reloj fundamental.

6. Bibliografía y Referencias

[1] Afarulrazi, A. B., Zarafi, M., Utomo, W. M. & Zar, A. FPGA implementation of Unipolar SPWM for single phase inverter. IEEE: 2010 International Conference on Computer Applications, pp. 671-676, 2010.

[2] Lakka, M., Koutroulis, E. & Dollas, A. Development of an FPGA-Based SPWM Generator for High Switching Frequency DC/AC Inverters. IEEE Transactions on Power Electronics, vol. 29, no. 1, pp. 356-365, 2014.

[3] LaMeres, B. J. Introduction to Logic Circuits & Logic Design with Verilog. Springer International Publishing. Montana, USA. 2019.

[4] Liou, W. R., Villaruza, H. M., Yeh, M. L. & Roblin, P. A Digitally Controlled Low-EMI SPWM Generation Method for Inverter Applications. IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pp. 73-83, 2014.

[5] Romli, M. S. N., Idris, Z., Saparon, A. & Hamzah, M. K. An area-efficient Sinusoidal Pulse Width Modulation (SPWM) technique for Single Phase

Matrix Converter (SPMC). IEEE: 2008 3rd IEEE Conference on Industrial Electronics and Applications, pp. 1163-1168, 2008.

[6] Salgado-Herrera, N. M., Medina-Ríos, A., Ramos-Paz, A. & Rodríguez-Rodríguez, J. R. Generation of a multilevel SPWM technique of 3, 9 and 21 levels with FPGAs. IEEE: 2013 North American Power Symposium (NAPS), pp. 1-5, 2013.

[7] Sankarakumar, S., Iruthayarajan, M. W., Sivakumar, T. & Chokkalingarn, S. Performance Analysis of Multicarrier Sine PWM Based Cascaded H-Bridge Multi Level Inverter. 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1018–1023, 2018.

[8] Sarker, R., Datta, A. & Debnath, S. FPGA-Based High-Definition SPWM Generation With Harmonic Mitigation Property for Voltage Source Inverter Applications. IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 1352-1362, 2021.

[9] Zhang, Z. & Wang, P. Research and Implementation of Natural Sampling SPWM Digital Method for Three-Level Inverter of Photovoltaic Power Generation System Based on FPGA. IEEE Access, vol. 7, pp. 49-58, 2019.