IMPLEMENTACIÓN DE TÉCNICAS DE GENERACIÓN DE MODULACIÓN POR ANCHO DE PULSO (PWM) UTILIZANDO DISPOSITIVOS DE LÓGICA PROGRAMABLE (FIELD-PROGRAMMABLE GATE ARRAY) PARA EL CONTROL DE INVERSORES TRIFÁSICOS

CRISTÓBAL GARCÍA PÉREZ
CRISTIAN EDWIN ARBOLEDA VALENCIA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERIA ELECTRICA, ELECTRONICA Y SISTEMAS DE COMPUTACIÓN
PROGRAMA DE INGENIERIA ELETRICA
PEREIRA
2007

IMPLEMENTACIÓN DE TÉCNICAS DE GENERACIÓN DE MODULACIÓN POR ANCHO DE PULSO (PWM) UTILIZANDO DISPOSITIVOS DE LÓGICA PROGRAMABLE (FIELD-PROGRAMMABLE GATE ARRAY) PARA EL CONTROL DE INVERSORES TRIFÁSICOS

CRISTÓBAL GARCÍA PÉREZ
CRISTIAN EDWIN ARBOLEDA VALENCIA

Trabajo de grado para optar el titulo de
Ingeniero Eléctrico

Directores
ALFONSO ALZATE GÓMEZ
Ingeniero Eléctrico
JOSE A. JARAMILLO VILLEGAS
Ingeniero Electrónico

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERIA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS DE COMPUTACIÓN
PROGRAMA DE INGENIERIA ELETRICA
PEREIRA
2007

AGRADECIMIENTOS

A nuestras Familias por todo el apoyo y cariño brindado durante estos años de estudio.

Los autores agradecen a COLCIENCIAS por el patrocinio de este proyecto gracias al contrato cod. 1110-08017738 CT-RC-362-2005, a la UNIVERSIDAD TECNOLÓGICA DE PEREIRA a través del grupo de investigación de electrónica de potencia.

CONTENIDO

Pág.

INTRODUCCIÓN

Las FPGAs (Field-programmable gate array) [1,5] son una tecnología de implementación de circuitos electrónicos, que debido a su gran flexibilidad en el momento de la programación además de su alta velocidad de operación y su ejecución de procesos en paralelo, la convierten en una opción atractiva al momento de implementar un sistema electrónico de control. Las FPGAs son programadas a través de un lenguaje de descripción de hardware denominado VHDL (Very High Description Language). Aunque el software es desarrollado individualmente por cada fabricante de FPGA, el lenguaje VHDL es un estándar IEEE facilitando su estudio y aplicación sin importar el producto que se adquiera.

Estas características hacen de la FPGA un dispositivo ideal para el desarrollo de algoritmos de control. La generación de señales PWM para el control de puentes inversores trifásicos, es una aplicación que se estudia a lo largo de este trabajo de grado. El uso del lenguaje VHDL para describir los algoritmos de una manera óptima es necesario, para su posterior implementación en la FPGA. Es importante desarrollar un método de descripción adecuado que aproveche las capacidades del dispositivo, con el fin de establecer un orden de operación óptimo y una dependencia de las señales de entrada con jerarquía superior.

Después de la definición de los algoritmos a desarrollar, se inició un proceso de aprendizaje del lenguaje de descripción de hardware, al tiempo que se desarrollo conocimiento en el uso del software, además de cómo implementarlo en la FPGA, para identificar y aplicar las ventajas de esta tecnología en la generación de señales PWM. Llevar estos algoritmos al lenguaje de descripción de hardware VHDL, implica un proceso de comprensión e

identificación de las propiedades deterministicas de cada uno, con el fin de llevar de manera optima estos algoritmos a VHDL, ya que esta tecnología opera con sistemas totalmente digitales, haciendo necesario la digitalización de todas la señales análogas que se deseen utilizar.

Luego de implementar estos algoritmos se debe construir una interfaz adecuada a la salida de la tarjeta, con el fin de aislar el dispositivo de control de la etapa de potencia, para después evaluar la calidad de la señal de salida del inversor trifásico, esto a través de la visualización de estas señales en un osciloscopio.

En este proyecto se uso la tarjeta de entrenamiento Spartan 3iE XC3S500E [13], la cual maneja el software xilinx 8.2i [12] de diseño, simulación e implementación para FPGA.

1. PROGRAMACIÓN EN VHDL

Como lo expresan las siglas VHDL (*Very high speed hardware description lenguaje)* [9] este lenguaje de programación esta directamente orientado a la descripción de hardware, no obstante este posee características propias de los lenguajes de programación de software clásicos (C, PASCAL, JAVA,...).

El concepto de tipo de datos es una característica que VHDL comparte con los lenguajes de programación clásicos, además de incluir la posibilidad de que el usuario defina los propios tipos de datos según las necesidades del programador.

Una de las cualidades mas importantes de VHDL es el manejo del control de flujo a través de las sentencias condicionales (if, case) y las sentencias de control iterativas (for, while), lo que hace mas sencilla la descripción de los procesos que se desean implementar. Aunque no se este haciendo directamente una descripción del hardware, se debe tener en cuenta lo que se desea desarrollar, con el fin de aprovechar al máximo la FPGA.

Otra de las ventajas de la programación en VHDL, es la creación de bibliotecas de diseño que contienen funciones, operadores y tipos de datos, con el fin de que el usuario defina un conjunto determinado para su trabajo.

En este capitulo se describirán algunas de las principales características del lenguaje de programación VHDL, que se empleó en el desarrollo de este proyecto.

1.1 MODELADO DEL HARDWARE

El lenguaje VHDL posee tres características enfocadas a describir el hardware: Bloque de estructura, Bloque de concurrencia y Bloque de tiempo. A continuación se describirán sus funciones y partes.

1.1.1 Bloque de estructura

Aquí se hace referencia al diseño digital, teniendo en cuenta sus partes y la forma en que estas se interconectan, además estas partes se modelan de forma estructural a través de sus componentes o de forma funcional usando algoritmos.

Para describir cualquier elemento en VHDL se deben definir las entradas y las salidas (puertos) del dispositivo (entidad), además de las funciones que realiza internamente con estas señales de entrada, para mostrar un resultado determinado en las salidas (arquitectura).

Una característica importante que maneja VHDL es la de usar un elemento ya definido como parte de uno mas grande, a través del concepto de componente. Para llevar a cabo esto se debe definir el elemento que se va a usar y conectar su interfaz de terminales a los puntos adecuados para el diseño que se esta realizando, es decir se asemeja a un montaje de electrónica en el que se interconectan elementos básicos para formar un sistema mas complicado, cabe agregar que no importa la complejidad del elemento este puede usarse como componente de otro.

1.1.2 Bloque de concurrencia

Con el fin de modelar el paralelismo en un sistema, VHDL posee un elemento denominado proceso, Un proceso puede definirse como un programa, se compone de sentencias y define datos para su uso exclusivo, puede usar subprogramas, es decir, describe un comportamiento a través de un código secuencial, con la salvedad que todos los procesos del elemento VHDL se ejecutaran en forma paralela. Un proceso se ubica en la arquitectura del elemento que se esta programando.

Para que los procesos que conforman un sistema de VHDL puedan actuar de forma sincronizada entre ellos, se emplea un elemento denominado señal. Cada proceso tiene un grupo de señales a las que es sensible, estas son determinadas por la persona que realiza el proyecto, ser sensible quiere decir que el proceso se ejecutara si se presenta algún cambio en las señales, este comportamiento se ejecuta desde el principio de manera cíclica.

1.1.3 Bloque de tiempo

Como el modelado en VHDL se orienta también hacia la simulación, es necesario definir el concepto de tiempo para llevar acabo este procedimiento. Una simulación esta ligada a eventos, es decir, posee una lista de cambios en las entradas y salidas del proyecto en VHDL, y de sus señales internas a lo largo del tiempo de simulación. La función del simulador consiste en calcular las consecuencias de cada cambio en la lista de eventos actual, además de agregarlos a la lista de eventos futuros, esto se realiza llevando a cabo los procesos del proyecto VHDL que se esta simulando.

Este proceso termina al completar el tiempo de simulación establecido por el usuario o cuando los eventos son nulos.

Cabe aclarar que las estructuras de simulación son denominadas bancos de trabajo, los cuales poseen algunas diferencias en la sintaxis con los proyectos normales de VHDL, además de contener sentencias de tiempo exclusivas.

1.2 SINTAXIS BÁSICA

1.2.1 La entidad

Al declararse la entidad se define la visión externa del dispositivo, su función es determinar la interfaz del dispositivo que se va a implementar con su entorno, dado que la entidad es usada como declaración del componente en otro diseño. La sintaxis básica de la entidad es:

ENTITY *{nombre del dispositivo}* IS
PORT (
    {Lista de puertos de entrada}: IN *{tipo dato}*;
    {Lista de puertos bidireccionales}: INOUT *{tipo dato}*;
    {Lista de puertos de salida}: OUT *{tipo dato}*;
    {Lista de puertos de salida}: BUFFER *{tipo dato}*;
    );
END *{nombre del dispositivo}*;

El nombre del dispositivo servirá como referencia para su futura aplicación como componente. La palabra *port* representa el lugar donde se declaran el tipo de puerto y el tipo de dato, a fin de determinar el tipo de información que maneja el puerto, además del nombre que identifica el puerto.

En conclusión se trabaja el dispositivo como una caja negra de la cual solo se conocen las características de sus entradas y sus salidas.

1.2.2 La arquitectura

Aquí se define la funcionalidad de la entidad, las operaciones de las entradas sobre las salidas del dispositivo en todo momento. La conexión de la arquitectura con la entidad, se realiza a través de la inclusión del nombre de la entidad en la sintaxis. La sintaxis básica de la arquitectura es:

ARCHITECTURE *{nombre de la arquitectura}* OF *{nombre de la entidad}* IS
{Zona de declaración}
BEGIN
{Sentencias concurrentes}
END *{nombre de la arquitectura}* ;

La zona de declaración se usa para definir elementos internos a usarse en la descripción del dispositivo, tales como constantes, variables o señales internas, como también el tipo de sentencias concurrentes existentes.

Según el tipo de sentencias concurrentes que se estén usando para describir el hardware, se modela la arquitectura en alguno de los siguientes estilos:

- Estilo algorítmico: Aquí se describe la funcionalidad del dispositivo mediante un algoritmo secuencial, de manera similar a la programación clásica (C, PASCAL, etc.). Un ejemplo de este estilo es:

Multiplexor de dos bits:

```
ENTITY Mux IS
PORT (
        a: IN STD_LOGIC;
        b: IN STD_LOGIC;
        ctrl: IN STD_LOGIC;
```

```
        z: OUT STD_LOGIC;
        );
END Mux;

ARCHITECTURE Algoritmo OF Mux IS
BEGIN
PROCESS (a, b, ctrl)
BEGIN
        IF ctrl = 0 THEN
        z <= a;
        ELSE
        z <= b;
        END IF;
END PROCESS;
END Algoritmo
```

- Estilo flujo de datos: Refleja la funcionalidad de un dispositivo mediante un grupo de ecuaciones ejecutadas concurrentemente, las cuales reflejan el flujo que van a seguir los datos entre módulos encargados de implementar las operaciones. En este estilo existe una correspondencia directa entre el código y su implementación en hardware. Un ejemplo de este estilo es:

Multiplexor de dos bits:

```
ENTITY Mux IS
PORT (
        a: IN STD_LOGIC;
        b: IN STD_LOGIC;
        ctrl: IN STD_LOGIC;
        z: OUT STD_LOGIC;
        );
END Mux;
```

```
ARCHITECTURE Flujo_datos OF Mux IS
SIGNAL ctrl_n, n1, n2: STD_LOGIC := '0';
BEGIN
ctrl_n <= NOT ctrl;
n1 <= ctrl_n AND a;
n2 <= ctrl AND b;
z <= (n1 OR n2);
END Flujo_datos;
```

- Estilo estructural: Consiste en un grupo de componentes elaborados previamente interconectados mediante señales. Aquí no se incluye ningún tipo de funcionalidad, pero se incluirá la definición del componente usado, además de interconectar sus terminales de interfaz. Un ejemplo de este estilo es:

Multiplexor de dos bits:

```
ENTITY Mux IS
PORT (
        a: IN STD_LOGIC;
        b: IN STD_LOGIC;
        ctrl: IN STD_LOGIC;
        z: OUT STD_LOGIC;
        );
END Mux;

ARCHITECTURE Estructural OF Mux IS
SIGNAL ctrl_n, n1, n2: STD_LOGIC:= '0';

COMPONENT inv
PORT (y : in  STD_LOGIC;
```

```
            z : OUT STD_LOGIC);
END COMPONENT;

COMPONENT AND1
PORT (x : in  STD_LOGIC;
            y : in  STD_LOGIC;
            z : OUT STD_LOGIC);
END COMPONENT;

COMPONENT OR1
PORT (x : in  STD_LOGIC;
            y : in  STD_LOGIC;
            z : OUT STD_LOGIC);
END COMPONENT;
 BEGIN

U0: INV PORT MAP (
          y => ctrl,
          z => ctrl_n);

U1: AND1 PORT MAP (
          x => ctrl_n,
          y => a,
          z => n1);

U2: AND1 PORT MAP (
          x => ctrl,
          y => b,
          z => n2);

U3: OR1 PORT MAP (
          x => n1,
```

```
        y => n2,
        z => z);

END Estructural;
```

Es importante saber que es posible combinar los tres tipos de descripción de la arquitectura, con el fin de mejorar el programa o facilitar el desarrollo de un proyecto.

1.3 OBJETOS, TIPOS DE DATOS Y OPERACIONES

Un tipo de dato determina el conjunto de valores posibles que el objeto contiene, además de las operaciones que se pueden realizar con el. En general las operaciones entre tipos de datos no son permitidas, a no ser que se incluya una conversión de tipo de datos a los operandos. La lista de objetos, tipos de datos y operaciones usadas en el desarrollo de este trabajo son:

1.3.1 Objetos

Un objeto VHDL es un elemento al cual se le asigna un valor de un tipo determinado. Todos los objetos deben ser declarados antes de ser usados, la declaración consiste en determinar el identificador, el tipo de dato y su valor inicial.

1.3.1.1 Constantes

Como su nombre lo indica este objeto mantiene siempre su valor inicial. Su estructura de declaración y un ejemplo del mismo se muestra a continuación:

constant {*identificador*} : {*tipo*} := {*expresión*};
constant pi: integer:= 25736;.

La declaración de constantes facilita la legibilidad del código al sustituir la declaración de literales, además de facilitar la actualización de su valor.

1.3.1.2 Variables

Estas pueden cambiar su valor según la secuencia algorítmica a la que estén asociadas, aunque no tienen una analogía directa con el hardware, se usan en los bloques de concurrencia, ya que su aplicación no es global, sino que se limita al proceso donde fue declarada. Su estructura de declaración y un ejemplo del mismo se muestran a continuación:

variable {*identificador*}: {*tipo*} := {*expresión*};
variable contador: std_logic_vector(19 downto 0):= "00000000000000000000";

1.3.1.3 Señales

Este objeto tiene como característica su analogía directa con el hardware, ya que se puede considerar como una abstracción de una conexión física o un bus, por lo cual esta ligado a la interconexión de componentes de un circuito y para sincronizar la ejecución y suspensión de procesos.

Su estructura de declaración y un ejemplo del mismo se muestran a continuación:

signal {*identificador*} : {*tipo*} := {*expresión*};
signal sector_aux : std_logic_vector (2 downto 0):= "000";

1.3.2 Tipos de datos

Este determina el conjunto de valores que puede asumir un objeto y las operaciones que se pueden realizar con el. Además de los tipos de datos predefinidos, también es posible crear nuevos tipos de datos con características determinadas por el programador.

Los tipos de datos predefinidos son:

CHARACTER: Formado por los 128 caracteres ASCII.

BOOLEAN: Definido sobre los valores (FALSE, TRUE).

BIT: Formado por los caracteres '0' y '1'

INTEGER y REAL: Con un rango que depende de cada herramienta.

TIME: Definido para especificar unidades de tiempo.

NATURAL: desde 0 al máximo valor INTEGER.

POSITIVE: desde 1 al máximo valor INTEGER.

WIDTH: desde 0 al máximo valor INTEGER

1.3.3 Operadores

En la tabla 1 se muestran los distintos operadores de VHDL:

Tabla1 Operadores predefinidos en VHDL

Operación	Descripción	Tipo de operandos	Resultados
**	Potencia	integer op integer	integer
		real op integer	real
not	Negación	bit, boolean, vector bits	Igual tipo operando
*	Multiplicación	integer op integer	integer
		real op real	real
		físico op integer	físico
		físico op real	físico
/	División	integer op integer	integer
		real op real	real
		físico op integer	físico
		físico op real	físico
		físico op físico	integer
mod	módulo	integer op integer	Igual tipo operando
rem	resto	integer op integer	Igual tipo operando
+	Suma	numérico op numérico	Igual tipo operando
-	Resta	numérico op numérico	Igual tipo operando
=	Igual que	Todo tipo op todo tipo	boolean
/=	Diferente de	Todo tipo op todo tipo	boolean
<	Menor que	Todo tipo op todo tipo	boolean
>	Mayor que	Todo tipo op todo tipo	boolean
<=	Menor o igual que	Todo tipo op todo tipo	boolean
>=	Mayor o igual que	Todo tipo op todo tipo	boolean
and	Y lógica	bit,bolean op bit,boolean	Igual tipo operando
Or	O lógica	bit,bolean op bit,boolean	Igual tipo operando
nand	Y lógica negada	bit,bolean op bit,boolean	Igual tipo operando
nor	O lógica negada	bit,bolean op bit,boolean	Igual tipo operando

1.4 SENTENCIAS SECUENCIALES

Las sentencias secuenciales son aquellas que nos permiten describir o modelar la funcionalidad de un componente. Ahora se describirán las sentencias secuénciales usadas a lo largo de este trabajo de grado.

1.4.1 Sentencia “wait”

Esta sentencia propia de bancos de prueba, indica en qué punto del flujo debe suspenderse la ejecución de un proceso, además puede fijar en qué condiciones debe reactivarse el proceso.

La primera forma de usar esta sentencia es la de truncar el proceso, con el fin de que este se ejecute solamente una vez. Su uso mas común es determinar las señales que van entrar a comprobar el funcionamiento del dispositivo. Un ejemplo de estructura es:

```
PROCESS
    BEGIN
      a <= not a;
      wait;
END PROCESS;
```

La segunda forma de usar esta sentencia es establecer que señales serán las que ejecuten el proceso. Esto condiciona cualquier proceso a algún cambio en las señales indicadas en la sentencia “wait on”; al usar esta sentencia con la terminación “wait untill” se convierte en un detector de eventos en la señal booleana a la cual se aplica la sentencia, haciendo que el proceso se reactive cuando exista un evento y una condición booleana determinada por el usuario. Un ejemplo de estructura es:

```
PROCESS
    BEGIN
        x <= a or b;
        wait on a,b;
        wait until clk = ‘1’;
END PROCESS;
```

La sentencia “wait for” es la ultima forma de usar la sentencia “wait”, y esta permite detener un proceso en un tiempo determinado, el cual se especifica con un dato de tipo “Time”, así es posible generar señales de reloj y secuencias de prueba complicadas. Un ejemplo de estructura es:

```
PROCESS
    BEGIN
      wait for 10 ns;
      clk <= not clk;
END PROCESS;
```

1.4.2 Sentencia “if”

Esta sentencia permite condicionar las tareas que se van a ejecutar en función de alguna condición determinada, esta sentencia tiene un uso similar al de cualquier otro lenguaje de programación. Es necesario tener en cuenta las características de las señales que se usan como condición, con el fin de que se tomen todos sus valores posibles, evitando así conflictos. Además se pueden crear condiciones múltiples al aplicar la sentencia “elsif”, o al iniciar condiciones internas creando otra estructura “if”. Un ejemplo de estructura es:

```
PROCESS (clk)
BEGIN
  if rising_edge(clk) then
     if aux = '1' then
        if estado >= 5 then
              estado <= 0;
        elsif en = '1' then
              estado <= estado +1;
        end if;
```

```
      end if;
  end if;
END PROCESS;
```

1.4.3 Sentencia “case”

Esta sentencia se usa para determinar que grupo de sentencias deben ejecutarse entre un grupo de posibilidades, según el rango de valores de una expresión que se usa para seleccionar, siendo esta expresión la que determina el número de posibilidades disponibles para usar.

La sentencia “case” se ejecuta cuando algún rango de valores especificados coincide con el valor actual de la expresión de selección, ignorando las demás.

Los valores para la selección nunca deben presentar intersecciones entre opciones distintas de la sentencia case. Además la unión de todos los valores especificados deben cubrir todos los valores posibles de la señal de selección. Un ejemplo de la estructura “case” es:

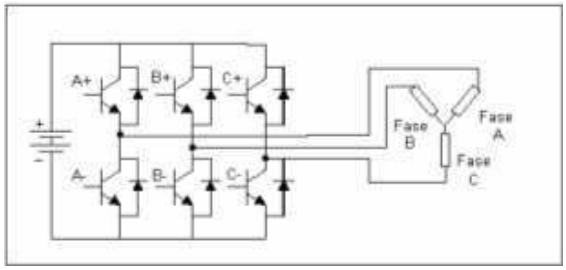
```
PROCESS
BEGIN
  case estado is
        when 0 => salidas <=  "101";
        when 1 => salidas <=  "100";
        when 2 => salidas <=  "110";
    when others =>  salidas <= "111";
  end case;
END PROCESS;
```

2. ALGORITMOS DE GENERACIÓN DE PWM

2.1 MODULACIÓN “SEIS PASOS” (PWM SEIS PASOS)

Para comprender la metodología de la modulación seis pasos, considérese el inversor trifásico mostrado en la Figura 1. Nótese que mientras el transistor A+ está encendido, el transistor A- está apagado y viceversa. Adoptando una notación específica es posible describir el funcionamiento del sistema. Por ejemplo, el estado cuando los transistores A+, B- y C- están encendidos (y por supuesto A-, B+ y C+ apagados) se puede representar por la notación (+, -, -). El estado cuando los transistores A-, B+, C- están encendidos es por lo tanto (-, +, -).

Figura 1. Inversor trifásico de voltaje con carga en estrella balanceada.



Usando esta notación considérese la siguiente secuencia de estados:
(+, -, -) , (+, +, -) , (-, +, -) , (-, +, +) , (-, -, +) , (+, -, +)

La modulación seis pasos en un puente inversor trifásico puede tener 8 estados; estos estados constan de dos vectores nulos y seis vectores activos.

Por medio del análisis de Fourier la magnitud del voltaje de salida en los terminales del inversor está dada por:

$$V_{\max(seis\ .pasos)} = \frac{4}{\pi} \cdot \frac{V_d}{2} \quad (1)$$

Siendo Vd el voltaje del bus de D.C. Es importante definir el índice de modulación como la relación entre el pico de la salida deseada y la fundamental máxima de salida en modulación seis pasos [3]:

$$m = \frac{|\ V_{Ref}\ |}{V_{Max(seis\cdot pasos)}} \quad (2)$$

2.2 MODULACIÓN “PWM CONVENCIONAL” (SPWM)

La modulación por ancho de pulso (PWM) controla el voltaje promedio de salida en un período lo suficientemente pequeño, llamado período de conmutación, mediante la producción de pulsos de ciclo de trabajo variable.

Una onda triangular de alta frecuencia, llamada portadora, es comparada con una señal senoidal que representa la salida deseada y denominada señal de referencia. Cuando la portadora es menor que la referencia, un comparador produce una salida en alto, si es llevada a una rama de transistores de un inversor, activa el transistor superior y desactiva el inferior por medio de una compuerta negadora. En el caso contrario, cuando la portadora sea mayor que la referencia, la salida del comparador será una señal en bajo, desactivando el transistor superior y activando el inferior.

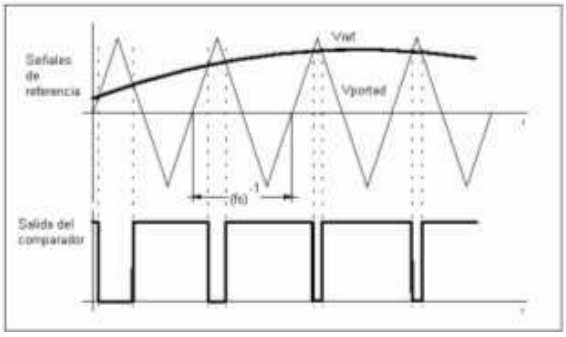
La magnitud del índice de modulación varía linealmente con la fracción:

$$m_a = \frac{V\max\_REFERENCIA}{V\max\_PORTADORA} \quad (3)$$

De tal manera es posible producir un voltaje senoidal, cuya magnitud y frecuencia dependen de la variación de la magnitud y frecuencia de la señal de referencia respectivamente. Sin embargo el voltaje de fase máximo posible alcanzado con este método es la mitad del voltaje de alimentación Vd del puente inversor dado que m_a no puede ser superior a la unidad.

$$V_{\max(pwm)} = \frac{V_d}{2} \quad (4)$$

Figura 2. Generación de una señal SPWM



Para lograr el control de un inversor trifásico se necesitarían tres dispositivos que generen tres señales PWM desfasadas 120 grados para formar cada una de las fases. El método de SPWM tiene desventajas como la elevada distorsión armónica y el hecho de que no es posible utilizar completamente la capacidad de la fuente del inversor, produciendo voltajes de Línea-Línea de aproximadamente el 86% del voltaje de D.C.

El índice de modulación SPWM es dado por la relación de las ecuaciones (4) y (1):

$$m = \frac{V_d/2}{2V_d/\pi} = 0.785 \quad (5)$$

Resultando que sólo el 78.5% de la capacidad del inversor es aprovechada [3].

2.3 MODULACIÓN POR ESPACIO VECTORIAL (SVPWM)

La modulación por espacio vectorial [3,8] se caracteriza en que se sustituye todo el sistema trifásico por un solo vector en el que la frecuencia queda reflejada en su velocidad de giro con el paso del tiempo, lo cual permite su uso para estudiar tanto los regímenes estacionarios como los dinámicos en dichos sistemas.

Se consideran tres instantes de tiempo en los que se obtienen los valores instantáneos de las tensiones de fase. Dados los voltajes (Va0, Vb0, Vc0), las componentes del vector (V_α , V_β) se pueden obtener de la transformación:

$$\vec{V} = V_\alpha + jV_\beta = \frac{2}{3}\left(V_{A0}\vec{a}^0 + V_{B0}\vec{a}^1 + V_{C0}\vec{a}^2\right) \quad (6)$$

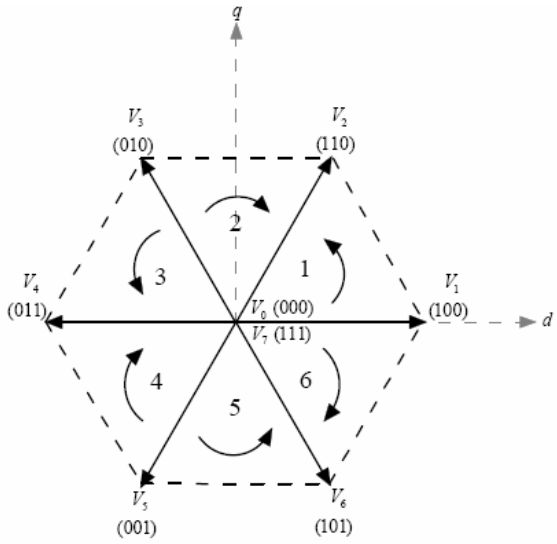
Como se vio en la modulación seis pasos un puente inversor trifásico puede tener 8 estados. Estos estados constan de dos vectores nulos y seis vectores activos, por lo tanto SVPWM busca aproximar el vector rotatorio de referencia en cada ciclo, mediante la conmutación entre los dos estados activos más cercanos a éste y los estados nulos.

Es posible demostrar la existencia de dos vectores nulos para los estados (+, +, +) y (-, -, -), y seis vectores no nulos para los demás estados. Luego, es fácil notar que los seis vectores no nulos, llamados activos, pueden ser representados por vectores espaciales de la siguiente forma:

$$V_k = \frac{2}{3} V_d e^{j(k+1)\frac{\pi}{3}} \quad (7)$$

Donde k es un entero entre 1 y 6, y que forman un hexágono regular dividido en sectores.

Figura 3. Esquema de conmutación indicando el sentido de inicio de cada sector



El hexágono de la Figura 3 muestra lo que sucede en los seis sectores, donde las flechas curveadas indican el sentido en que comienza la secuencia, y también representa el rango de voltajes que se pueden obtener por medio de un vector espacial. Mediante el proceso de SVPWM es posible lograr cualquier voltaje confinado dentro del hexágono.

La técnica de SVPWM continua se basa en el hecho de que cada vector dentro del hexágono, puede ser representado como una combinación de los dos vectores espaciales activos adyacentes y los vectores espaciales nulos 0 y 7. Con el fin de obtener mejor desempeño armónico y menor frecuencia de conmutación para los componentes de potencia, la secuencia es programada de manera que la transición de un sector al siguiente, se da mediante la conmutación de sólo una de las ramas del inversor. Tal condición se cumple si la secuencia comienza con uno de los estados nulos y los polos del inversor son conmutados, hasta que el siguiente estado nulo se alcance. Para completar el ciclo, la secuencia es reversada, terminando con la primera condición nula.

La columna vertebral de SVPWM es la combinación adecuada de los estados activos y nulos para cada ciclo de modulación.

El requisito de mínimas conmutaciones por ciclo es alcanzado si en cada sector impar la secuencia de vectores aplicados es:

$$V_0\ V_k\ V_{k+1}\ V_7\ V_{k+1}\ V_k\ V_0$$

Mientras que en sectores pares, los vectores son aplicados en orden inverso, o sea:

$$V_0\ V_{k+1}\ V_k\ V_7\ V_k\ V_{k+1}\ V_0$$

Si se asume que T_k denota la mitad del tiempo activo del vector y que T_0 es la mitad del tiempo del estado nulo, los tiempos activos pueden ser evaluados mediante las siguientes ecuaciones:

$$\int_0^{\frac{T_s}{2}} V_{Ref}\,dt = \int_0^{\frac{T_0}{2}} V_0\,dt + \int_{\frac{T_0}{2}}^{\frac{T_0}{2}+T_k} V_k\,dt + \int_{\frac{T_0}{2}+T_k}^{\frac{T_0}{2}+T_k+T_{k+1}} V_{k+1}\,dt + \int_{\frac{T_0}{2}+T_k+T_{k+1}}^{\frac{T_s}{2}} V_7\,dt \quad (8)$$

$$T_0 + T_k + T_{k+1} = \frac{T_s}{2} \quad (9)$$

Definiendo un vector espacial promedio en un período de conmutación Ts y asumiendo que Ts es lo suficientemente pequeño, el vector promedio puede ser considerado aproximadamente constante durante este intervalo. La ecuación (8) se reduce a:

$$V_{Ref} \cdot \frac{T_s}{2} = V_k \cdot T_k + V_{k+1} \cdot T_{k+1} \quad (10)$$

Separando esta ecuación vectorial en sus componentes reales e imaginarios, de (7) se obtiene:

$$\begin{pmatrix} V_\alpha \\ V_\beta \end{pmatrix} \frac{T_s}{2} = \frac{2}{3} V_d \left(T_k \begin{pmatrix} \cos\frac{(k-1)\pi}{3} \\ \sin\frac{(k-1)\pi}{3} \end{pmatrix} + T_{k+1} \begin{pmatrix} \cos\frac{k\pi}{3} \\ \sin\frac{k\pi}{3} \end{pmatrix} \right) = \frac{2}{3} V_d \begin{bmatrix} \cos\frac{(k-1)\pi}{3} & \cos\frac{k\pi}{3} \\ \sin\frac{(k-1)\pi}{3} & \sin\frac{k\pi}{3} \end{bmatrix} \begin{pmatrix} T_k \\ T_{k+1} \end{pmatrix} \quad (11)$$

Donde k se determina a partir del argumento del vector de referencia tal que:

$$\frac{(k+1)\pi}{3} \leq \arg\begin{pmatrix} V_\alpha \\ V_\beta \end{pmatrix} \leq \frac{k\pi}{3} \quad (12)$$

Resolviendo el sistema planteado en (11) se obtiene:

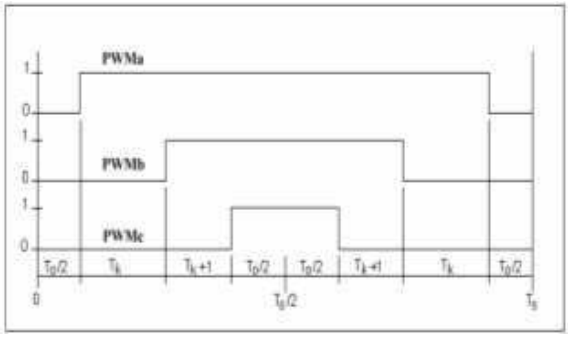
$$\begin{pmatrix} T_k \\ T_{k+1} \end{pmatrix} = \frac{\sqrt{3}}{2} \frac{T_s}{V_d} \cdot \begin{bmatrix} \sin\frac{k\pi}{3} & -\cos\frac{k\pi}{3} \\ -\sin\frac{(k-1)\pi}{3} & \cos\frac{(k-1)\pi}{3} \end{bmatrix} \cdot \begin{pmatrix} V_\alpha \\ V_\beta \end{pmatrix} \quad (13)$$

El tiempo total de estado nulo T_0 puede dividirse arbitrariamente entre los dos estados nulos, sin embargo resulta más sencillo dividirlo en partes iguales para ambos vectores; y despejando de (9), se obtiene:

$$T_0 = \frac{T_s}{2} - \left(T_k + T_{k+1}\right) \quad (14)$$

A manera de ejemplo se muestra el esquema para la conmutación en el sector 1 en la figura 4.

Figura 4. Señales PWM para vector de referencia en sector 1



Asumiendo que se desea producir un sistema de voltajes de fase balanceados, se hace necesario que la trayectoria del vector espacial correspondiente sea circular. Haciendo que:

$$\mathbf{V}_{\mathbf{Ref}} = |\,\mathbf{V}_{\mathbf{Ref}}\,|\,\mathbf{e}^{\,j\cdot\omega\cdot t} = |\,V_{Ref}\,|\cdot\left(\cos(\omega\cdot t)+\sin(\omega\cdot t)\right) \quad (15)$$

donde $|\,V_{Ref}\,|$ es la magnitud y ω es la frecuencia angular de los voltajes de fase deseados, de (13) se obtiene :

$$\begin{pmatrix} T_k \\ T_{k+1} \end{pmatrix} = \frac{\sqrt{3}}{2}\frac{|\,V_{ref}\,|}{V_d}\cdot T_s \cdot \begin{bmatrix} \sin\frac{k\pi}{3} & -\cos\frac{k\pi}{3} \\ -\sin\frac{(k-1)\pi}{3} & \cos\frac{(k-1)\pi}{3} \end{bmatrix} \cdot \begin{pmatrix} \cos(\omega\cdot t) \\ \sin(\omega\cdot t) \end{pmatrix} \quad (16)$$

y dado que cuando $0 \le \omega\cdot t \le \pi/3$, el vector de referencia cae en el sector 1, por lo que la ecuación (16) se reduce a [3]:

$$\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} = \frac{\sqrt{3}}{2}\frac{|\,V_{Ref}\,|}{V_d}\,T_s \begin{pmatrix} \sin\left(\frac{\pi}{3}\right)-\omega\cdot t \\ \sin(\omega\cdot t) \end{pmatrix} \quad (17)$$

Los valores promedio de los voltajes de fase del inversor son los siguientes:

$$V_{A0}(\omega \cdot t) = \frac{V_d}{2T_s}\left(-\frac{T_0}{2} + T_1 + T_2 + T_0 + T_2 + T_1 - \frac{T_0}{2}\right) = \frac{\sqrt{3}}{2}\frac{|\vec{V}_{Ref}|}{V_d} V_d \cdot \cos\left(\omega \cdot t - \frac{\pi}{6}\right) \quad (18)$$

$$V_{B0}(\omega \cdot t) = \frac{V_d}{2T_s}\left(-\frac{T_0}{2} - T_1 + T_2 + T_0 + T_2 - T_1 - \frac{T_0}{2}\right) = \frac{3}{2}\frac{|\vec{V}_{Ref}|}{V_d} V_d \cdot \sin\left(\omega \cdot t - \frac{\pi}{6}\right) \quad (19)$$

$$V_{C0}(\omega \cdot t) = -V_{A0}(\omega \cdot t) \quad (20)$$

Resolviendo análogamente la ecuación (16) para los demás sectores se obtiene:

$$V_{A0}(\omega \cdot t) = \frac{\sqrt{3}}{2} |V_{Ref}| \cos\left(\omega \cdot t - \frac{\pi}{6}\right) \quad (21)$$

$$0 \le \omega \cdot t \le \frac{\pi}{3}$$

$$V_{A0}(\omega \cdot t) = \frac{3}{2} |V_{Ref}| \cos(\omega \cdot t) \quad (22)$$

$$\frac{\pi}{3} \le \omega \cdot t \le \frac{2\pi}{3}$$

$$V_{A0}(\omega \cdot t) = \frac{\sqrt{3}}{2} |V_{Ref}| \cos\left(\omega \cdot t + \frac{\pi}{6}\right) \quad (23)$$

$$\frac{2\pi}{3} \le \omega \cdot t \le \pi$$

$$V_{A0}(\omega \cdot t) = \frac{\sqrt{3}}{2} |V_{Ref}| \cos\left(\omega \cdot t - \frac{\pi}{6}\right) \quad (24)$$

$$\pi \le \omega \cdot t \le \frac{4\pi}{3}$$

$$V_{A0}(\omega \cdot t) = \frac{3}{2} |V_{Ref}| \cos(\omega \cdot t) \quad (25)$$

$$\frac{4\pi}{3} \le \omega \cdot t \le \frac{5\pi}{3}$$

$$V_{AO}(\omega \cdot t) = \frac{\sqrt{3}}{2} | V_{Ref} | \cos\left(\omega \cdot t + \frac{\pi}{6}\right) \quad (26)$$

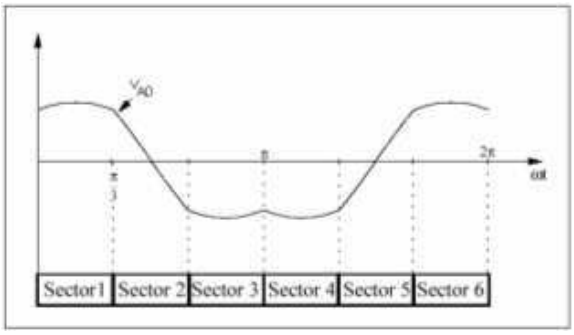
$$\frac{5\pi}{3} \le \omega \cdot t \le 2\pi$$

$$V_{BO}(\omega \cdot t) = V_{AO}\left(\omega \cdot t - \frac{2\pi}{3}\right) \quad (27)$$

$$V_{CO}(\omega \cdot t) = V_{AO}\left(\omega \cdot t - \frac{4\pi}{3}\right) \quad (28)$$

$V_{AO}(\omega \cdot t)$ es mostrado en la Figura 5.

Figura 5. Voltaje de fase para SVPWM ideal



Evaluando los voltajes de línea se tiene:

$$V_{AB}(\omega \cdot t) = V_{AO}(\omega \cdot t) - V_{BO}(\omega \cdot t) = \sqrt{3} | V_{Ref} | \sin\left(\omega \cdot t + \frac{\pi}{3}\right) \quad (29)$$

$$V_{BC}(\omega \cdot t) = V_{AB}\left(\omega \cdot t - \frac{2\pi}{3}\right) \quad (30)$$

$$V_{CA}(\omega \cdot t) = V_{AB}\left(\omega \cdot t - \frac{4\pi}{3}\right) \quad (31)$$

para $0 \le \omega \cdot t \le 2\pi$

Si SVPWM ha de producir un sistema trifásico balanceado de voltajes senoidales de magnitud Vref y velocidad angular ω dados por:

$$V_{AN} = V_{Ref} \cos(\omega \cdot t) \quad (32)$$

el vector de espacio de referencia viene dado por:

$$V_{Ref} = V_{Ref}\left[\cos(\omega \cdot t) + j \sin(\omega \cdot t)\right] = V_{Ref} e^{j \cdot \omega t} \quad (33)$$

Por lo tanto el vector espacial de referencia describe una trayectoria circular de radio V_{Ref} a una velocidad angular ω en el plano complejo. El máximo voltaje alcanzado utilizando la técnica de SVPWM, corresponde al radio del máximo círculo que puede ser inscrito en el hexágono. Tal círculo es tangencial a los puntos medios de las líneas que conectan los fines de los vectores correspondientes a los estados activos. Así, el voltaje máximo de fase viene dado por:

$$| V_{Ref} |_{\max} = \frac{2}{3} V_d \frac{\sqrt{3}}{2} = \frac{\sqrt{3}}{3} V_d \quad (34)$$

De acuerdo con la definición de índice de modulación, el máximo índice de modulación correspondiente es:

$$m_{\max(cont)} = \frac{| V_{Ref} |_{\max}}{V_{\max(seis \cdot pasos)}} = \frac{\frac{\sqrt{3}}{3} V_d}{\frac{2}{\pi} V_d} = \frac{\pi}{2\sqrt{3}} = 0.906 \quad (35)$$

El máximo valor de salida que puede ser obtenido con SVPWM es de cerca del 90.6% de la capacidad del inversor, lo que representa un incremento del 15% en comparación con la modulación SPWM. De la ecuación (1), el voltaje de línea máximo es:

$$V_{AB\max} = \sqrt{3} \mid V_{Ref} \mid_{\max} = V_d \quad (36)$$

3. DESCRIPCIÓN DE MÉTODOS DE GENERACIÓN DE PWM EN VHDL

Para la implementación de los algoritmos planteados para generar señales de PWM que controlen un inversor trifásico [4, 11], se deben convertir sus características principales a un entorno digital, con el fin de poder adaptar a la descripción de hardware su implementación en VHDL. A continuación se mostraran las funciones básicas de los componentes, como por ejemplo el formato de los datos de entrada y salida de cada modulo. Los módulos funcionales se identificaran con el nombre de la entidad correspondiente a cada uno.

3.1 DISPOSITIVO SEIS PASOS

Este es el método más básico y se compone de un solo modulo en VHDL. El código de este modulo esta incluido en el anexo 7. La representación esquemática y el diagrama de flujo están incluidos en los anexos 17 y 21.

3.1.1 Entradas y salidas

Las entradas al dispositivo son las siguientes:

Clk: Es la señal compuesta por un tren de pulsos que se usa para sincronizar el sistema, es decir es la señal encargada de que los sistemas secuénciales

conmuten. Este reloj tiene un valor de 50MHz y esta señal es usada en todos los dispositivos.

En: Este terminal es el encargado de habilitar la ejecución del proceso. Si en el terminal se aplica un uno lógico, el proceso trabaja normalmente, pero cuando se aplica un cero lógico la señal “Clk” es bloqueada, y el sistema se deshabilita.

Periodo: Tiene como fin determinar la frecuencia de las señales de salida del inversor. Esta entrada se calcula de la siguiente manera:

$$Periodo = \frac{F_M}{F * 6} \quad (37)$$

En donde:
F_M: Frecuencia del reloj.
F: Frecuencia deseada en el sistema trifásico a la de salida del inversor.

El resultado se transforma en un número binario y es llevado a los terminales de la entrada “periodo”.

Las salidas del dispositivo son:

Salidas: Conjunto de seis terminales los cuales son el resultado del proceso interno, según los datos de entrada que son aplicados al dispositivo. Estos corresponden a las seis entradas del inversor.

3.1.2 Modulo “PWMseispasos"

Esta compuesto por un contador de tiempo, el cual mide el tiempo en base al periodo del reloj de entrada. Cuando el contador iguala el valor de la entrada “periodo”, habilita el contador de estado, que representa los seis pasos del

método que se esta implementando. La salida del contador de estados se lleva a un decodificador, dándole el valor de los seis vectores activos a la señal de salida “Salidas”.

3.2 DISPOSITIVO SVPWM

Este dispositivo esta compuesto por tres módulos, uno es el encargado de las operaciones aritméticas, otro se ocupa de la medición de tiempos y asignación de las salidas correspondientes [15, 16], y por ultimo se incluye un IP CORE denominado “CORDIC” [12] para el cálculo del seno. En los anexos 8,9 y 10 se muestran los tres módulos que componen este dispositivo. La representación esquemática y el diagrama de flujo están incluidos en los anexos 18 y 22.

3.2.1 Entradas y salidas

Las entradas al dispositivo y al modulo “calculos” son las siguientes:

Clk: Esta entrada se comporta de forma similar a la descrita en el dispositivo seis pasos.

Reset: Tiene como función inicializar todas las señales internas de cada uno de los módulos al aplicarse un uno lógico.

Enable: Esta entrada se comporta de forma similar a la denominada “En”, descrita en el dispositivo seis pasos.

Amplitud: Esta entrada es calculada de la siguiente manera:

$$Amplitud = \frac{\left|\vec{V}ref\right|}{\frac{2}{3}Vcc} * 2^{14} \qquad (38)$$

En donde:

V_{ref}: Magnitud del voltaje de referencia.

V_{cc}: Valor del voltaje continuo en la barra positiva del inversor trifásico.

El resultado se transforma en un número binario y es llevado a los terminales de la entrada “amplitud”.

Para evitar obtener valores de tiempo negativo, se determino el valor máximo de esta entrada mediante un código en Matlab, obteniendo un valor máximo de 0.86603, en el anexo 6, se incluye el programa con su respectivo resultado.

Periodo: Tiene como fin determinar la frecuencia de las señales de salida del inversor. Esta entrada se calcula de la siguiente manera:

$$Periodo = \frac{2^{18}}{F * 6} \qquad (39)$$

En donde:

F: Frecuencia deseada para el sistema trifásico a la salida del inversor.

El resultado se transforma en un número binario y es llevado a los terminales de la entrada “periodo”.

Desfase: Corresponde al ángulo de referencia, para el inicio del sistema en un determinado momento, es decir si la fase A del sistema inicia en 30 grados las fases B y C estarán a 120 y 240 grados de desfase de la fase A. El valor del desfase se calcula de la siguiente manera:

(40)

$$Desfase = \theta * 2^{13}$$

En donde:
Θ: Magnitud del ángulo en radianes.

El resultado se transforma en un número binario y es llevado a los terminales de la entrada “Desfase”.

Las salidas del dispositivo son:

Salida: Combinación de los estados de conmutación según el método modulación por espacio vectorial.

3.2.2 Modulo “seno”

Este se encarga del cálculo del seno. Es importante agregar que este IP CORE “CORDIC” [12], calcula funciones trigonometricas, convierte coordenadas rectangulares a polares, calcula funciones hiperbólicas y calcula la raíz cuadrada.

Las entradas al modulo “seno” son las siguientes:

SCLR y *CE:* Cumplen la función de “Reset” y “Enable” respectivamente.

PHASE_IN: Determina el angulo de entrada en radianes, este valor se ingresa en el formato 2QN de 16 bits, el primer BIT representa el signo, el segundo y tercer BIT tienen el valor de 2^1 y 2^0, del cuarto BIT en adelante tiene un valor de 2^{-1}, 2^{-2}, 2^{-3},…,2^{-13}, con el fin de representar fracciones, por ejemplo:

3.5 equivale a 0111000000000000.
-Pi equivale a 1110010010001000.

Estos números binarios se obtienen al multiplicar el número decimal, por dos elevado a una potencia igual al número de BIT que representan números entre cero y uno, en este caso trece.

Es decir:

$3.5 * 2^{13} = 28672$ equivale a 0111000000000000 en binario.
$Pi * 2^{13} = 25736$ equivale a 0110010010001000 en binario.

Las salidas del modulo son:

YOUT: Entrega el resultado del seno. El formato de los datos de salida es 1QN, este trabaja el primer BIT como el signo, el segundo tiene un valor de 2^0 y del tercer BIT en adelante tiene el valor de 2^{-1}, 2^{-2}, 2^{-3},…,2^{-14}, esto entrega números de -1 a 1. Un ejemplo de este formato es:

0.781 equivale a 0011000111111100.
-0.586 equivale a 1010010110000001.

RDY: Determina con uno lógico cuando el modulo esta listo para entregar un resultado, este tiempo depende de la velocidad del reloj.

El modulo “seno” calcula el valor del seno en las ecuaciones de tiempos activos y tiempos nulos para el primer cuadrante. Este posee un retraso de calculo inicial propio, este retraso es de aproximadamente 598ns. La señal “RDY” del modulo “seno” se conecta al terminal “En” del modulo “SVPWM”, con el fin de garantizar que el proceso se inicie en el instante que los cálculos hayan sido realizados.

3.2.3 Modulo “Cálculos”

Este modulo posee un bloque que toma el valor de la entrada “desfase”, e identifica el sector inicial para enviarlo al modulo “SVPWM”, y los ángulos en el primer sector para enviarlos al modulo “seno”.

Un bloque de multiplicadores toma las entradas “periodo”, “amplitud” y las señales de salida del modulo “seno” (“sen1” y “sen2”), para calcular los tiempos de “T1”. “T2” y “T00”, según las formulas de tiempos activos y tiempos nulos para el primer cuadrante, a estos multiplicadores se les añaden registros, con el fin de sincronizar los cálculos, ya que el sistema debe calcular secuencialmente estos valores, para que el sistema trabaje eficientemente al realizar cada operación en paralelo.

3.2.4 Modulo “SVPWM”

Este modulo posee las siguientes entradas y salidas análogas al modulo “cálculos”:

Reloj: Esta entrada se comporta de forma similar a la descrita en el dispositivo seis pasos.

Reset y *en*: Estas entradas se comporta de forma similar a la descrita en el dispositivo SVPWM.

Salida: Resultado de los estados de conmutación según el método modulación por espacio vectorial.

Entradas exclusivas del modulo “SVPWM”:

T00, T1 y *T2*: Reciben los datos “T00”, “T1” y “T2” del bloque “cálculos”.

Sector: Indica al modulo en que sector debe comenzar el proceso.

Este modulo posee un bloque compuesto de sumadores el cual toma los valores “T00”, “T1” y “T2”, para calcular los limites de tiempo de los estados nulos y activos.

Un bloque contador de tiempo lleva el valor de la entrada “periodo” a tiempo real, con base en 3.800ns, ya que el último BIT de la entrada “periodo” tiene un valor de 2^{-18} segundos. Este contador tiene como punto máximo el tiempo determinado por la señal “periodo”, el bloque contador de sectores también es controlado por el bloque contador de tiempo. El contador de sectores es el encargado de determinar, el sector que esta operando en ese momento.

Finalmente se encuentra un bloque el cual posee un total de seis estados, el contador de estados determina el sector que se va a desarrollar, cada uno de estos estados esta compuesto por comparadores, los cuales comparan el valor de los límites de tiempo de los estados nulos y activos, con el valor del contador de tiempo, para finalmente asignarle un valor adecuado a la salida del dispositivo, de acuerdo al método de modulación por espacio vectorial.

3.3 DISPOSITIVOS SPWM y SPWMA

Es importante mencionar que estos dispositivos trabajan con la mitad de un ciclo, para no incluir el BIT de signo y además optimizar componentes internos. Los módulos que conforman cada una de las dos versiones están incluidos en los anexos 11, 12, 13 y 14. La representación esquemática y el diagrama de flujo están incluidos en los anexos 19, 20, 23 y 24.

3.3.1 SPWM

Este dispositivo posee un generador de ondas senoidales trifásicas digitalizadas, además del generador de onda triangular.

3.3.1.1 Entradas y salidas

Las entradas al dispositivo y al modulo “fases” son las siguientes

Clk, rst y *en*: Estas corresponden a las entradas “Clk”, “reset” y “enable” explicadas para el dispositvo “SVPWM”

Periodo: Tiene como fin determinar la frecuencia de las señales de salida del inversor. Esta entrada se calcula de la siguiente manera:

$$Periodo = \frac{F_M}{F * pi * 2^{14}} \quad (41)$$

En donde:
F_M: Frecuencia del reloj de entrada al sistema.
Pi: Constante igual a 3.1416.
F: Frecuencia deseada para el sistema trifásico a la salida del inversor.

El resultado se transforma en un número binario y es llevado a los terminales de la entrada “periodo”.

Amp: Representa el aumento discreto en la amplitud de la onda triangular, va en pasos de 0.625%, este aumento es representado en 8 estados digitales.

Veces: Entrada que define la relación entre el periodo de la onda triangular y de la onda seno, este cociente puede tomar un valor de 15 o 21.

Las salidas del dispositivo son:

PWM_F1, PWM_F1N: estas dos salidas entregan el tren de pulsos resultante de aplicar el método de “SPWM” a la fase 1 del sistema trifásico, PWM_F1N resulta de negar la salida PWM_F1.

PWM_F2, PWM_F2N, PWM_F3, PWM_F3N: Estas salidas cumplen la función de las salidas “PWM_F1” y “PWM_F1N”, con respecto a las fases 2 y 3 del sistema trifásico.

3.3.1.2 Modulo “seno”

Este es el mismo modulo descrito para el dispositivo “SVPWM” y se encarga del cálculo en tiempo real del seno, este resultado es llevado al bloqué de comparación “SPWM”.

3.3.1.3 Modulo “SPWM”

Este modulo compara el valor obtenido del modulo seno y el valor de la onda triangular. Las entradas de este modulo son:

Clk, rst y en: Estas corresponden a las entradas “Clk”, “reset” y “enable” explicadas para el dispositivo “SVPWM”

TRI: Entrada de la onda triangular.

Theta: Tiene como objetivo llevar el Angulo a la entrada del modulo “seno”.

Las salidas de este modulo son:

Rdy: Determina con un uno lógico cuando el modulo esta listo para entregar un resultado, este tiempo depende de la velocidad del reloj.

PWM: Resultante de la comparación de las ondas triangular y seno.

Este modulo se compone de un bloque que compara la salida del modulo “seno” con el valor de la entrada “TRI” y le asigna un uno lógico a la salida “PWM”, cuando el valor en la salida del modulo “seno” sea mayor al de la variable “TRI”, y en caso contrario le asigna un cero lógico.

3.3.1.4 Modulo “Fases”

Definidas ya las entradas y salidas de este modulo podemos pasar a describir su función, el primer bloque se trata de una estructura decodificadora, que le da un valor determinado a la señal “amplitud”, según el valor en la entrada “Amp”.

Un segundo bloque identifica la magnitud del periodo de la onda triangular, realizando el siguiente procedimiento:

El periodo de la onda seno se definió en la ecuación (41) y el periodo de la onda triangular se define como:

$$PeriodoT = \frac{F_M}{F * veces * 2^{13} - 1} \quad (42)$$

En donde:

PeriodoT: Periodo de la onda triangular

F_M: Frecuencia del reloj de entrada al sistema.

Pi: Constante igual a 3.1416.

F: Frecuencia deseada para el sistema trifásico a la salida del inversor.

Veces: tiene un valor de 15 o 21 según lo que se quiera aplicar.

Ahora haciendo un cociente entre las ecuaciones (41) y (42) se obtiene:

$$\frac{veces}{pi} * 0.499938964844 = \frac{Periodo}{PeriodoT} \quad (43)$$

$$PeriodoT = \frac{Periodo * pi}{veces} * 2.00024417043 \quad (44)$$

Reemplazando *veces* en la ecuación (44) se obtiene:

$$PeriodoT = periodo * 0.299235828153 \text{ Para 21.} \quad (45)$$

$$PeriodoT = periodo * 0.418930159414 \text{ Para 15.} \quad (46)$$

El calculo del periodo de la onda triangular, se realiza tomando el valor de la entrada “periodo” aplicada a las ecuaciones (45) y (46). La entrada “veces” define cual de las ecuaciones se va a emplear.

Se establece un bloque contador de tiempo, para controlar la frecuencia de los contadores de ángulo para cada fase y el generador de onda triangular. La salida de los tres bloques encargados de contar los ángulos del sistema trifásico, se envían a los módulos “SPWM” definidos como “fase1”, “fase2” y “fase3”. El desfase se establece según el valor del ángulo de inicio de conteo, además de declarar la constante “pi” como limite de conteo común, ya que se trabajara con medio ciclo de la onda.

El generador de onda triangular consiste, en un contador ascendente-descendente, que se multiplica por la señal “amplitud”, siendo este resultado el que se conecta a la entrada “TRI” del modulo “SPWM”.

Tres bloques se encargan de negar medio ciclo de las salidas de los módulos “SPWM”, actuando cada vez que su contador de angulo sea igual a “pi”, y conectándolas a las salidas “PWM_F1”, “PWM_F2” y “PWM_F3”, además de crear las salidas “PWM_F1N”, “PWM_F2N” y “PWM_F3N” que conforman la negación de las tres primeras salidas.

3.3.2 SPWMA

Como la relación entre la onda triangular y la onda senoidal se fijo en 15 y 21, se desarrollo un código fuente en Matlab para obtener los puntos de corte, donde la magnitud de la onda senoidal es mayor a la onda triangular, para luego crear una tabla. Estos resultados se llevan a un entorno digital, con el fin de realizar un cambio en la amplitud de la onda triangular. El cambio se fijo hasta un 50% de la magnitud, este código fuente se incluye en los anexos 1, 2 y 3.

3.3.2.1 Entradas y salidas

Las siguientes entradas y salidas corresponden a las del dispositivo “SPWM” en todas sus características, variando solamente en el uso interno que les da el dispositivo. Las entradas de este modulo son:

Clk, en, periodo y las salidas *PWM_F1, PWM_F1N, PWM_F2, PWM_F2N, PWM_F3 y PWM_F3N*.

Pero posee entradas propias como:

Up: Incrementa la magnitud de la onda triangular.

Down: Se encarga de disminuir la amplitud de la onda triangular cuando esta se encuentra por encima de la onda seno.

Div: Se encargada de seleccionar un cociente de 15 o 21 entre los periodos de la onda triangular y de la onda senoidal.

3.3.2.2 Módulo “PWM”

Las entradas de este modulo son:

Up, Down y Div: Estas corresponden a las entradas explicadas para el modulo “SPWMA”.

Periodo: Recibe el valor del angulo del modulo “SPWMA”

Las salidas de este modulo son:

Sal: Resultante de la comparación de las ondas triangular y seno.

Este modulo trabaja principalmente con los puntos de corte definidos en el programa Matlab, un bloque contiene la tabla de los puntos de corte, los cuales se llevan al formato 2QN, creando los limites de comparación indicados para cada método.

Además este módulo contiene bloques sumadores, a los cuales llegan las entradas “up”, “down” y “div”, usando las dos primeras señales como control

del cambio en la amplitud de la señal triangular, la entrada “div” selecciona el valor del cociente entre los periodos de las ondas triangular y senoidal.

Finalmente este módulo esta compuesto por un bloque de comparadores, el cual determina cuando la salida “sal” tendrá un valor de uno o cero lógico, comparando la entrada “periodo” con los intervalos creados mediante las tablas de los puntos de corte.

3.3.2.3 Modulo “SPWMA”

Esta compuesto por un contador de tiempo, el cual mide el tiempo en base al periodo del reloj de entrada. Cuando el contador iguala el valor de la entrada “periodo”, habilita el contador de angulo que tiene como limite la constante “pi” según el formato 2QN, este contador controla los bloques contadores de ángulo, que representan el sistema trifásico, que se aplican a la entrada “periodo” del modulo “PWM”, los bloques negadores que toman la salida “sal” del modulo “PWM” se comportan igual que los expuestos en el modulo “fases” del dispositivo “SPWM”

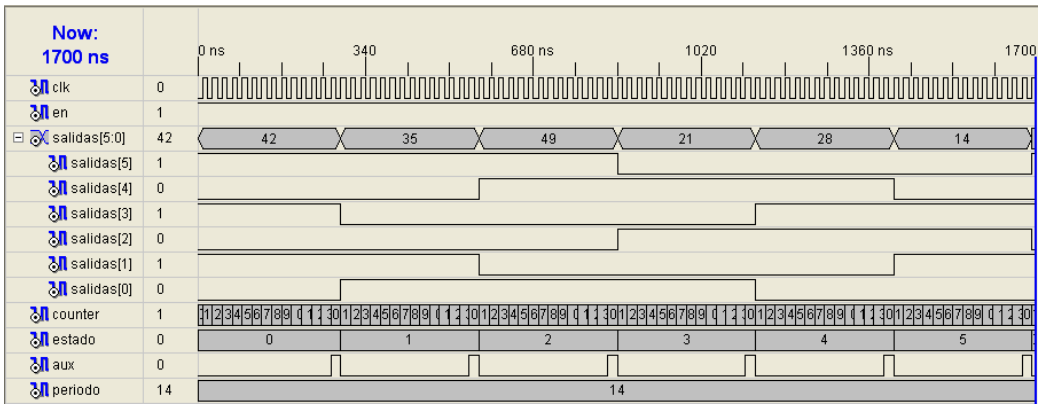
4. RESULTADOS OBTENIDOS

4.1 DISPOSITIVO SEIS PASOS

4.1.1 Simulación

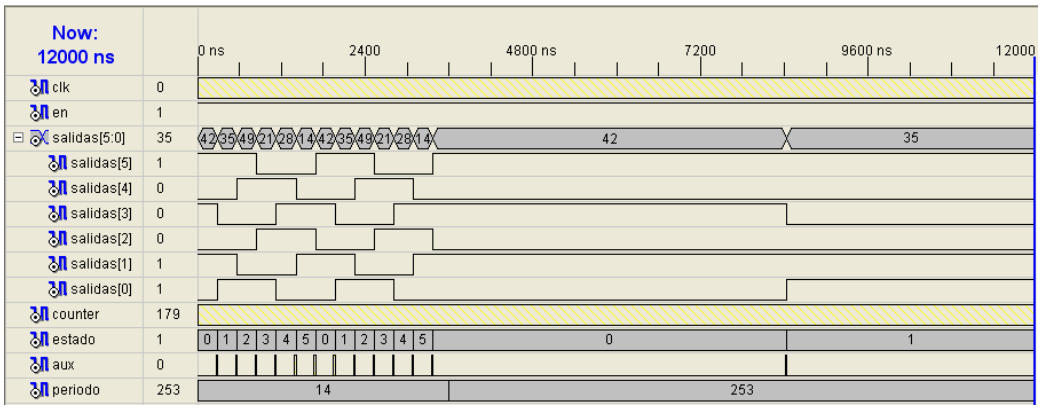
Los resultados de la simulación en el software de xilinx fueron:

Figura 6. *Simulación para una señal con un periodo de 1680 nS*



En la simulación mostrada en la figura 6, se observa que cada vez que el contador de tiempo (“counter”) cuenta catorce estados, la señal “aux” toma un valor de uno lógico y el contador de estados (“estado”) conmuta, cambiando en la salida (“salidas”) el vector activo correspondiente. Dado que la entrada “periodo” posee un valor de catorce. Los tres últimos BIT que componen la salida (“salidas”), son equivalentes a los tres primeros BIT negados.

Figura 7. *Simulación de cambio en la entrada “periodo” durante la operación*



En la figura 7 se observa un cambio en la entrada “periodo” iniciando en un valor de 14 y cambiando a un valor de 253, lo que produce un cambio en el limite del contador de tiempo “counter” y aumentando el periodo de las señales de salida (“Salidas”). Este cambio va de un periodo de 1680 nS a uno de 30360 nS.

4.1.2 Salida de pulsos

Las salidas de la FPGA obtenidas mediante un osciloscopio fueron las siguientes:

En la figura 8 se observa la salida de la FPGA para una de las ramas del inversor.

En la figura 9 se realizó un acercamiento a la señal de salida de la FPGA, con el fin de visualizar que no existen rebotes o traslapes en las salidas.

Figura 8. *Salidas de una misma rama en los terminales de la FPGA*

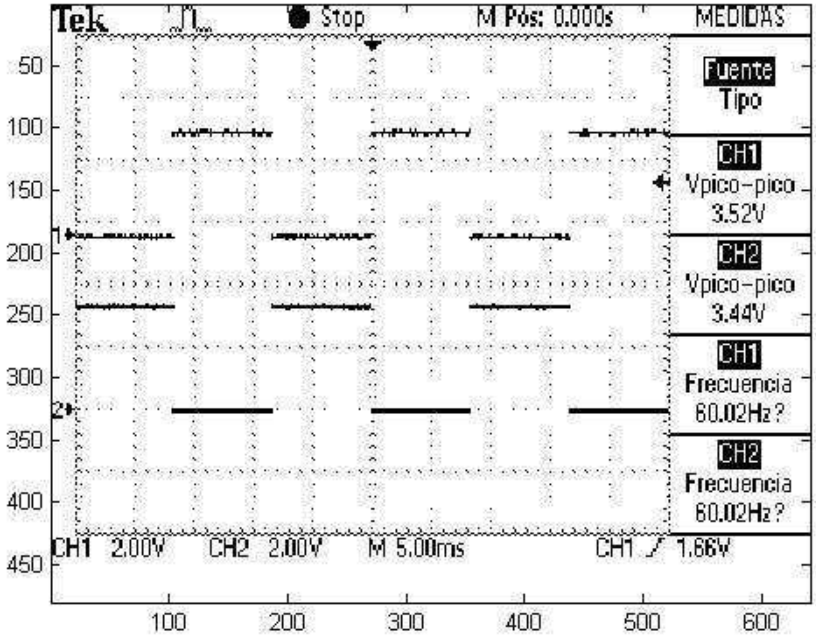
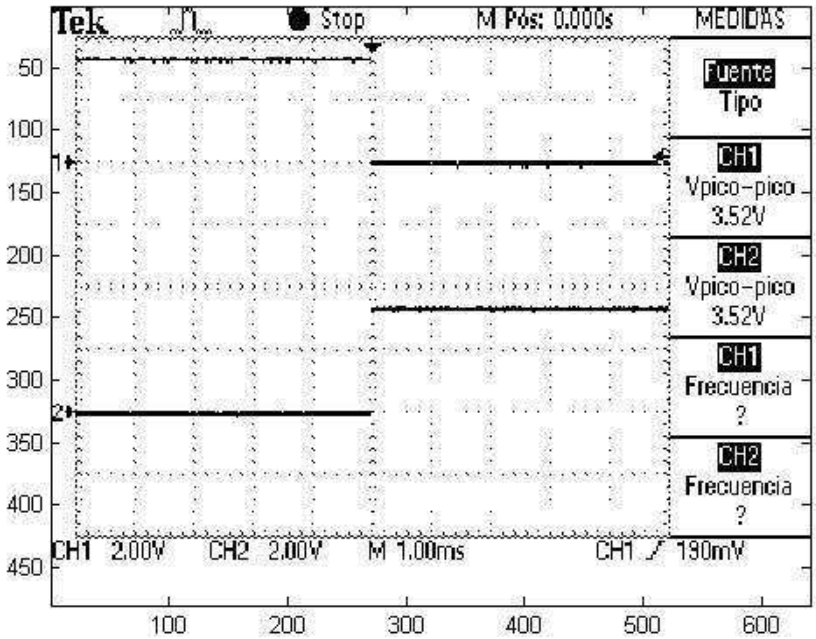
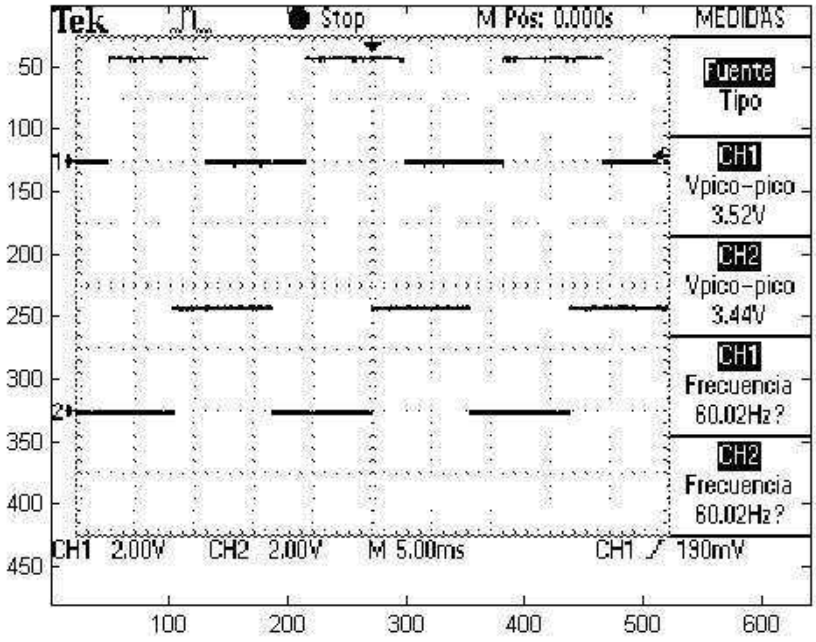


Figura 9. *Acercamiento en los pulsos de salida*



En la figura 10 se puede visualizar el desfase entre dos ramas diferentes del inversor.

Figura 10. *Salidas de la FPGA para dos ramas diferentes del inversor*



4.1.3 Salida inversor

En la figura 11 se observa el diagrama de conexión al inversor trifásico, usado para todos los dispositivos. Se conecta la FPGA con la tarjeta de interfaz, la cual aísla la FPGA del inversor, teniendo en cuenta que la carga, consta de una resistencia 1000 ohmios con un condensador en paralelo de 10μF y en serie con una inductancia de 100mH. Se desea visualizar la señal de salida en la resistencia.

En las figuras 12 y 13 se observan las salidas del inversor, las cuales presentan deformaciones a 120 Hz, pero estas disminuyen al aumentar la

frecuencia a 160 Hz, sin embargo la forma de la onda no es la más apropiada, ya que posee un aspecto de diente de sierra.

Figura 11. *Diagrama de conexión al inversor trifásico*

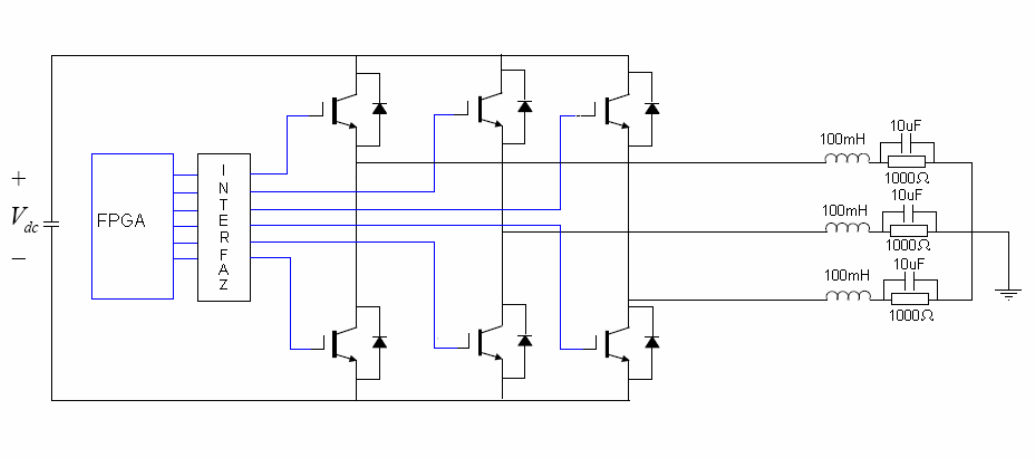


Figura 12. *Onda en la resistencia de carga a 120 Hz con carga trifásica de 100mH, 10 uF y 1000 ohmios*

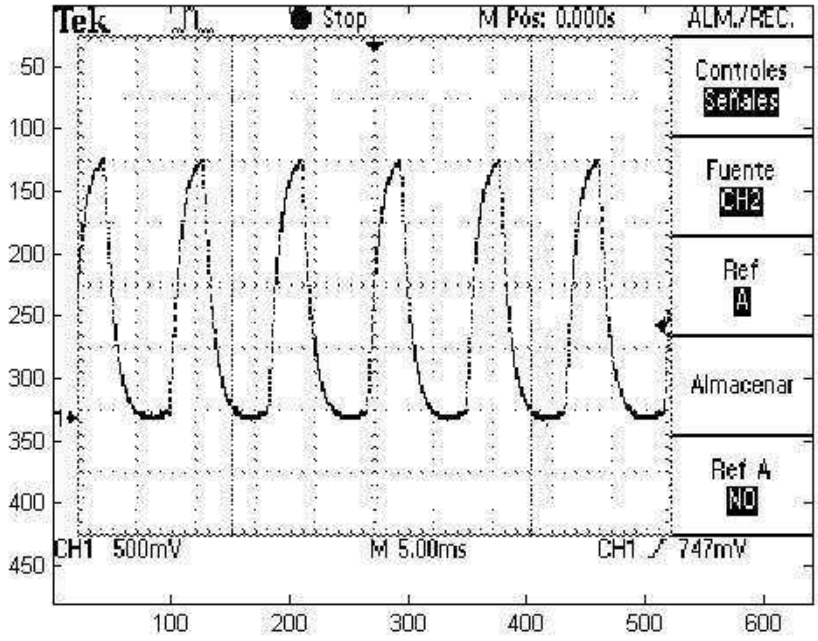
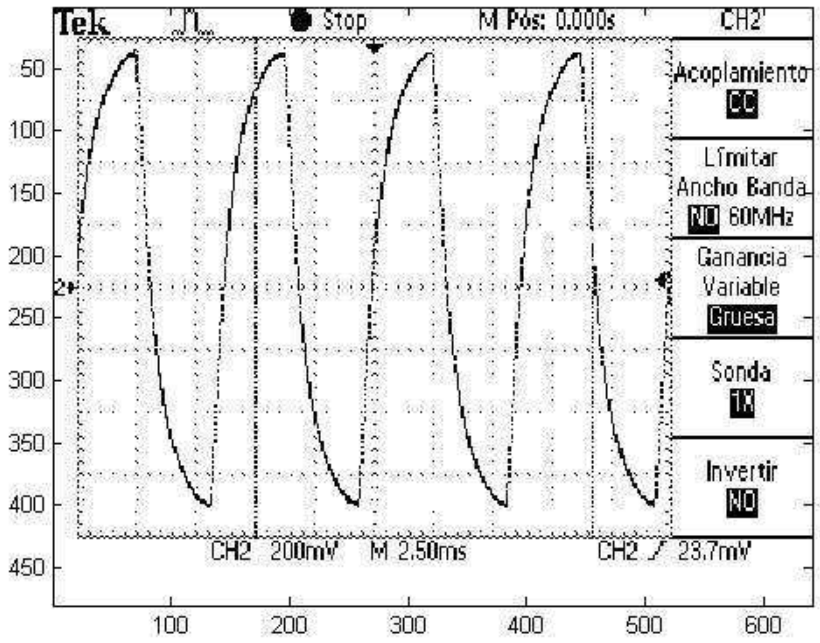


Figura 13. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios*



4.2 DISPOSITIVO SVPWM

4.2.1 Simulación

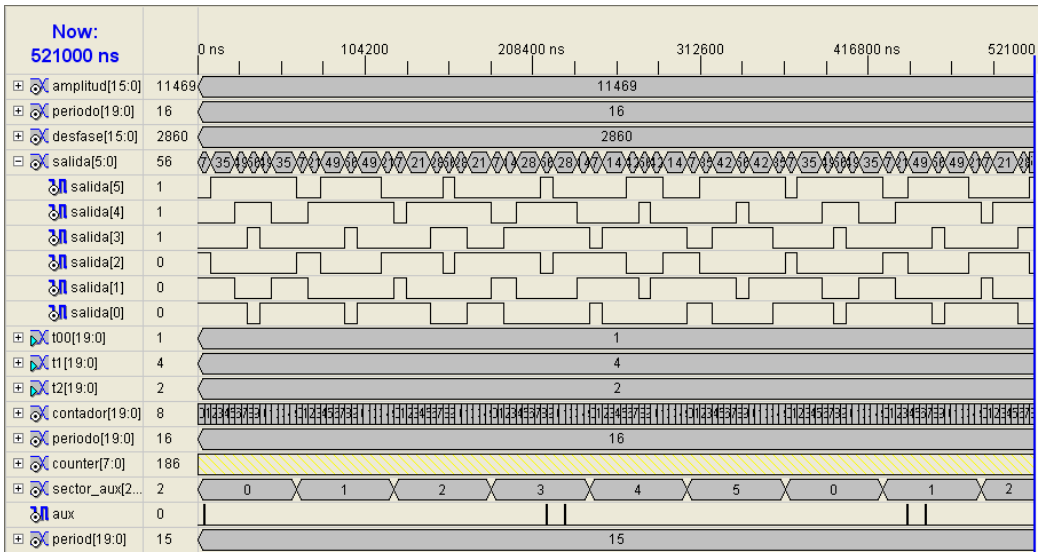
En la figura 14 se aprecia el valor de la entrada “periodo”, “amplitud” y “desfase” para obtener una salida a 2730 Hz. Además una amplitud de 0.7 y un desfase de 20 grados. También se observa la forma de las salidas y el número del sector inicial. En este caso el sector inicial (“sector_aux”) es igual a cero. El valor de los tiempos “T00”, “T1” y “T2”, las señales “contador”, “counter”, “sector” y “aux” que se encargan de la medición de tiempos y control de los sectores de salida.

Las señales “T00”, “T1” y “T2” responden al cálculo del periodo ya que:

$$PERIODO = 4*T00 + 2*T1 + 2*T2 \quad (48)$$

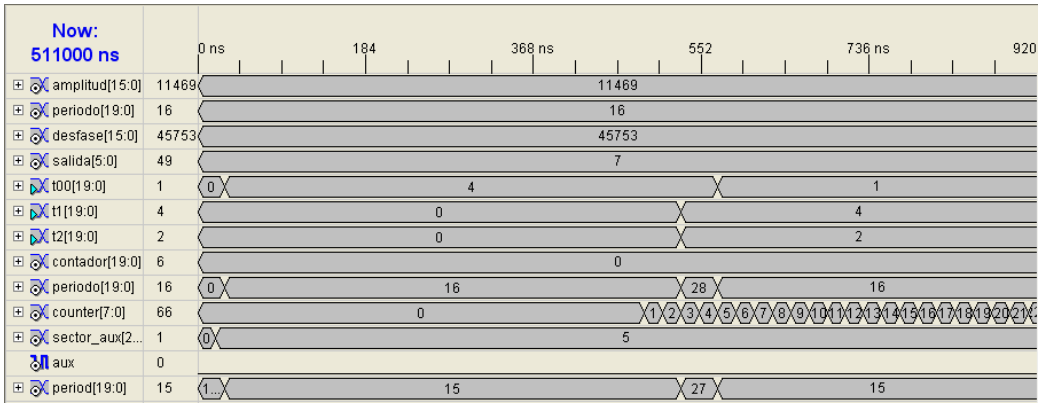
$$PERIODO = 4*1 + 2*4 + 2*2 = 16 \quad (49)$$

Figura 14. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 20 grados*



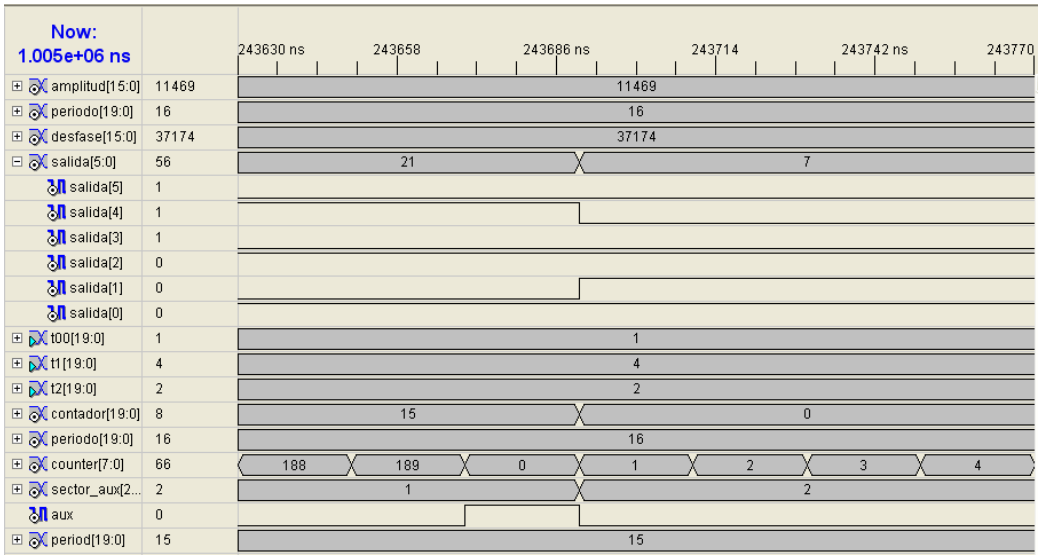
En la figura 15 se observa el inicio de la simulación, el contador (“counter”) inicia su conteo en el instante en que se ha calculado las señales internas “sen” y “sen1”, se inicia el calculo de “T1” y “T2” para luego entregar el valor de “T00”, este proceso se demora aproximadamente 576 nS, pero solo representa un retraso en el inicio del proceso, ya que las salidas inician su operación en el instante que los cálculos sean realizados. Al generar señales de salidas superiores a 11MHz presentaría dificultades, pero esto se soluciona aumentando la velocidad del reloj o disminuyendo el tamaño de las entradas y de las señales internas.

Figura 15. *Simulación del Inicio del proceso*



Para ver un acercamiento de la señal “aux” y el cambio de la señal “sector”, se hizo un acercamiento en la simulación teniendo que el contador (“counter”), realiza un conteo de 3800 nS, según lo explicado en la sección 3.2.4, esto se puede observar en la figura 16.

Figura 16. *Simulación con acercamiento en el auxiliar y cambio de sector*



En las figuras 17, 18, 19, 20 y 21 se observan los resultados para diferentes desfases, lo cual produce un sector de inicio diferente.

Figura 17. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 110 grados*

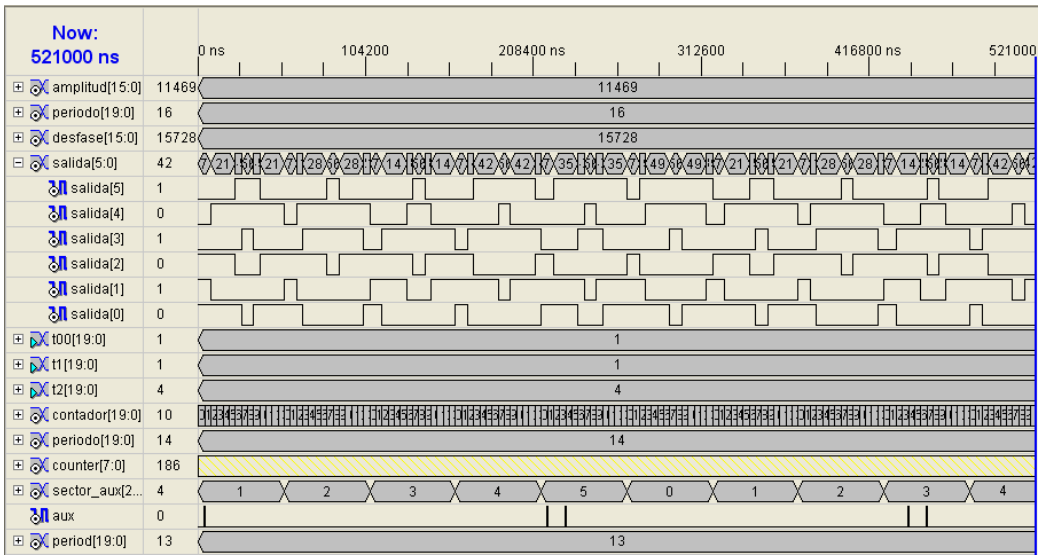


Figura 18. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 160 grados*

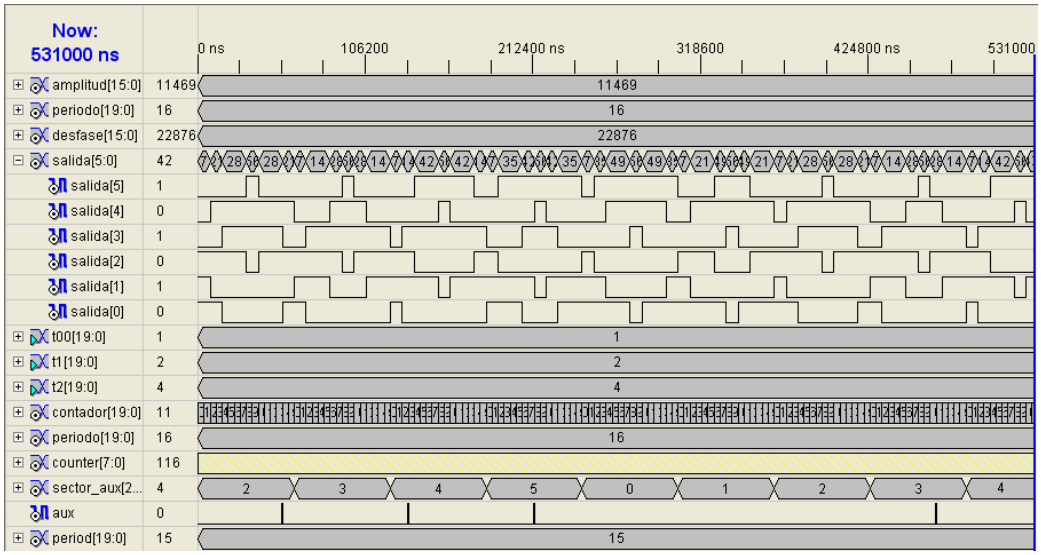


Figura 19. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 220 grados*

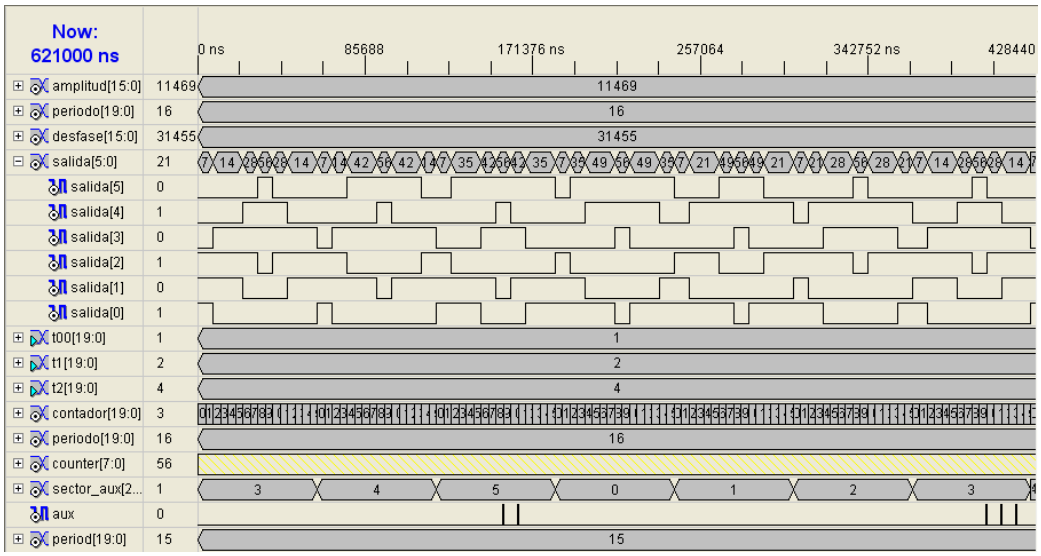


Figura 20. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 260 grados*

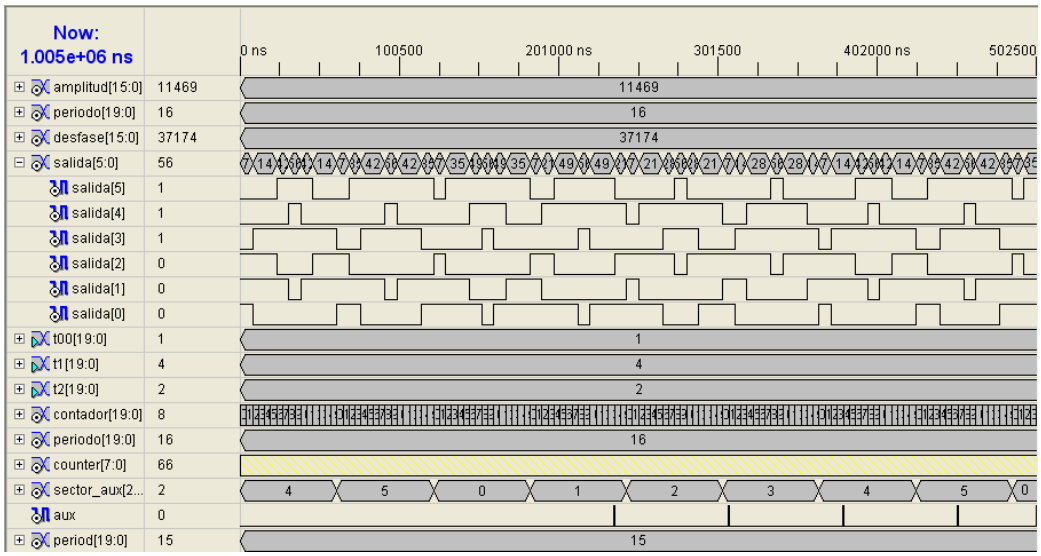
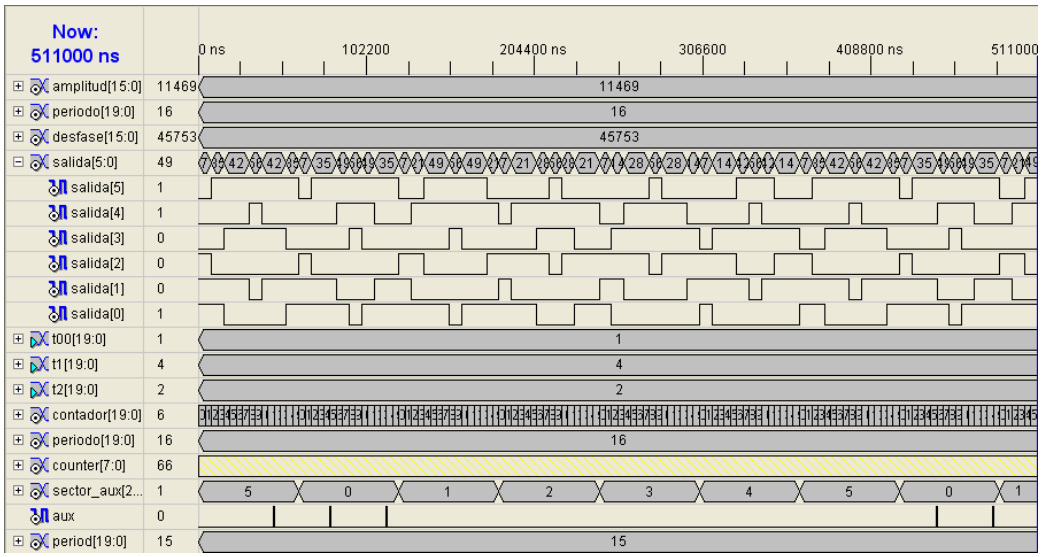


Figura 21. *Simulación para una salida con periodo de 36.62 mS, amplitud de 0.7 y desfase de 320 grados*



4.2.2 Salida de pulsos

En la figura 22 se observa la salida de la FPGA para una de las ramas del inversor.

En la figura 23 se realizó un acercamiento a la señal de salida de la FPGA, con el fin de visualizar que no existen rebotes o traslapes en las salidas.

En la figura 24 se puede visualizar el desfase entre dos ramas diferentes del inversor.

Figura 22. *Salida de la FPGA para una misma rama del inversor*

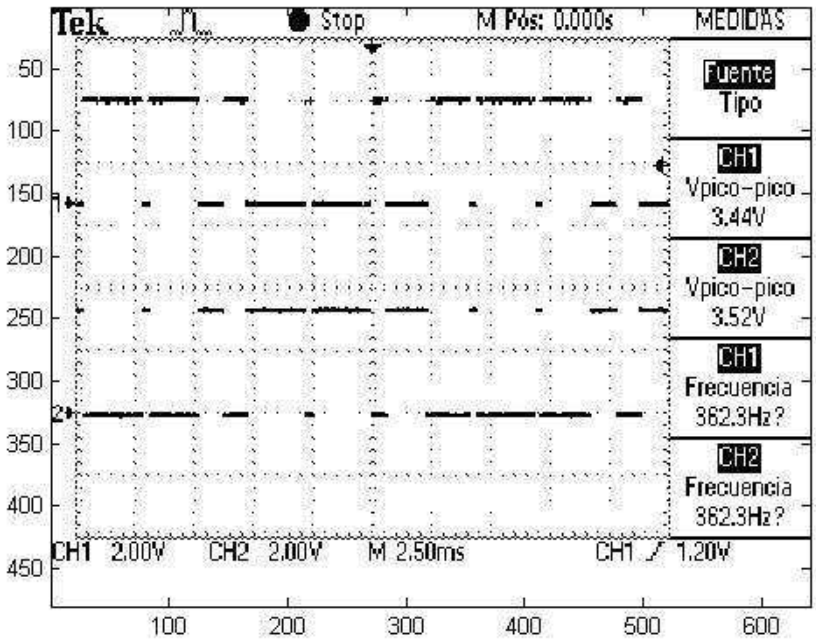


Figura 23. *Salida ampliada de La FPGA para una misma rama del inversor*

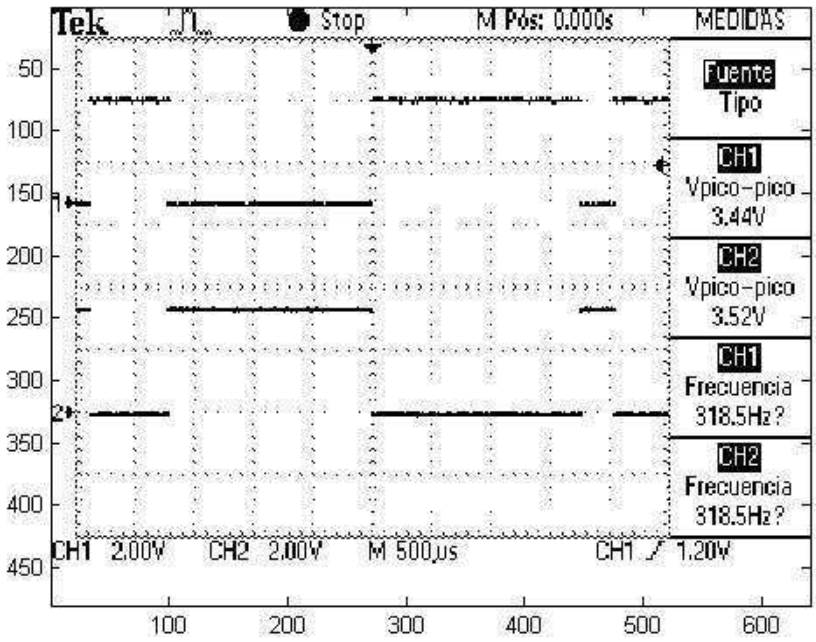
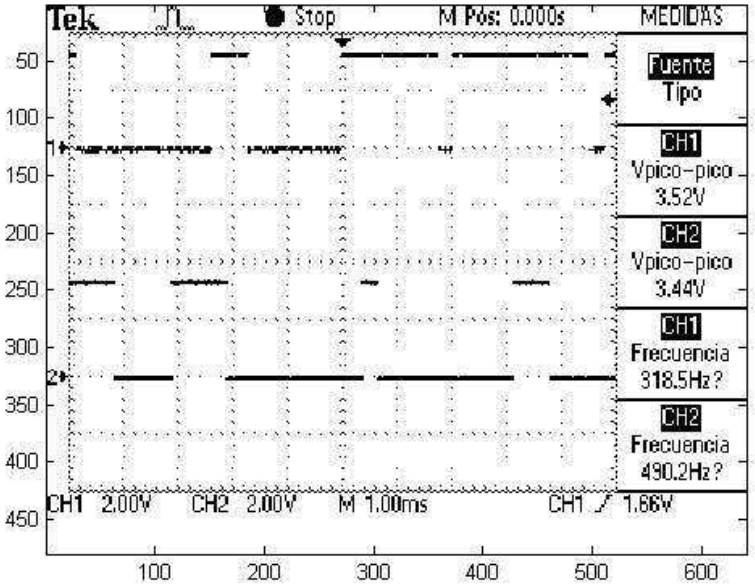


Figura 24. *Salidas de la FPGA para las ramas restantes del inversor*



4.2.3 Salida inversor

Figura 25. *Onda en la resistencia de carga a 120 Hz de 100 mH, 10 uF y 1000 ohmios*

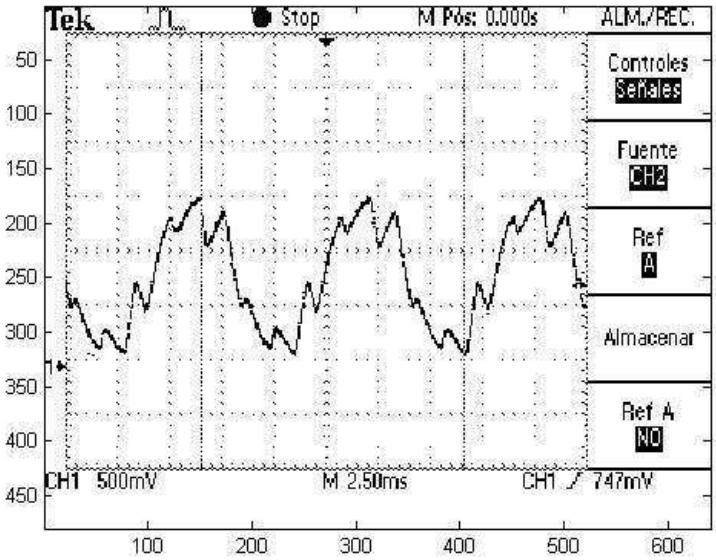


Figura 26. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios*

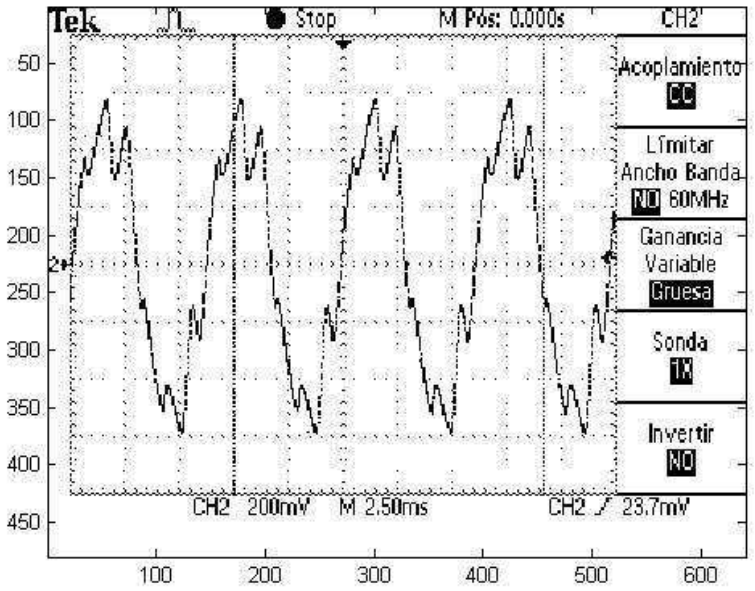
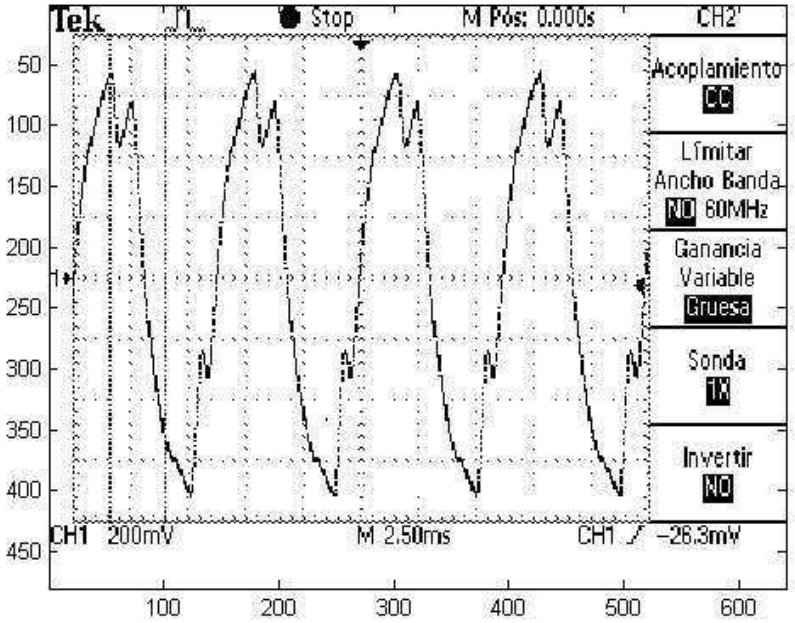


Figura 27. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios con una amplitud máxima*



Se utilizó el montaje de la figura 11, las salidas resultantes son mostradas en las figuras 25 y 26, para frecuencias de 120Hz y 160Hz, las salidas presentan una gran deformación.

La alta deformación de la señal es clara aunque se lleve la amplitud a su valor máximo como en la figura 27, pero se observa que esta mantiene un adecuado valor de frecuencia.

4.3 DISPOSITIVO SPWM

4.3.1 SPWM

4.3.1.1 Simulación

En la simulación de la figura 28, se observan las salidas para un periodo de 16.471 mS y una relación de periodos de 15, las señales negadoras “neg1”, “neg2” y “neg3”, el valor digital de entrada “periodo” y la señal “period1”, que determinan el periodo de la onda seno y la onda triangular respectivamente. El generador de onda triangular (“ram”), en el cual la mitad del periodo es igual a cero, ya que este dispositivo trabaja con medio ciclo de la onda senoidal. El grupo de salidas (“pwm_f1”, “pwm_f2”, “pwm_f3”, “pwm_f1n”, “pwm_f2n” y “pwm_f3n”) muestra una forma acorde al método de modulación pwm convencional.

En la figura 29 se observa la simulación para un periodo de 8.2355 mS y una relación de periodos de 21.

Figura 28. *Simulación para una salida con periodo de 16.471 mS, con una relación de periodos de 15 con amplitud aumentada*

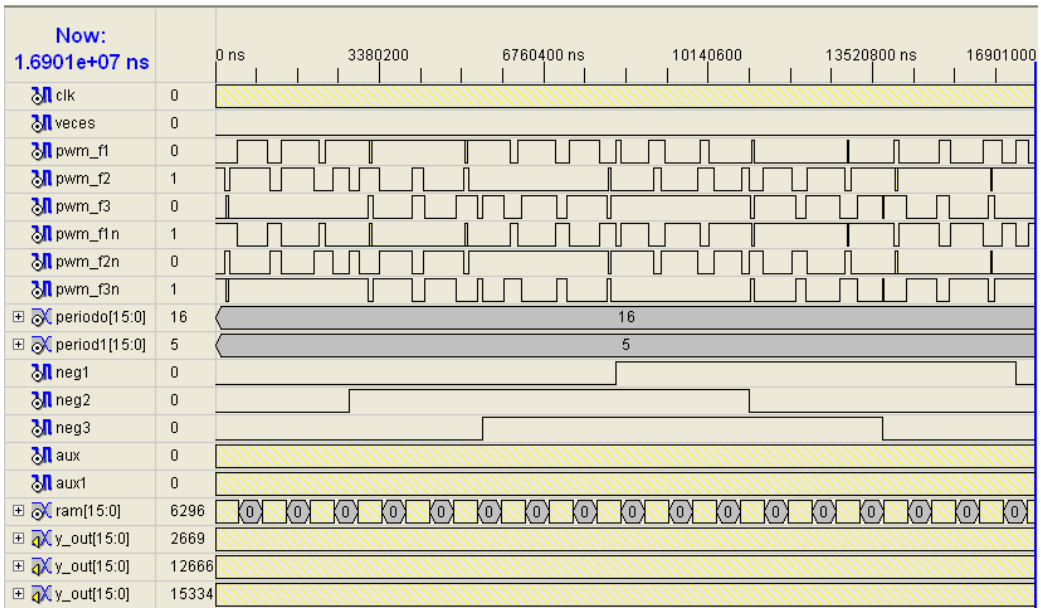
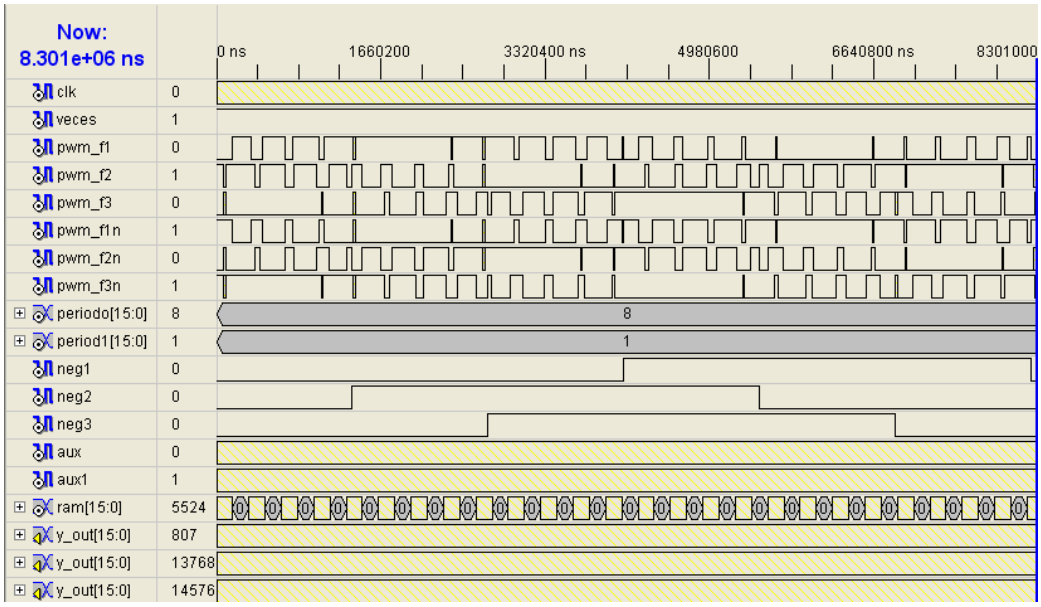


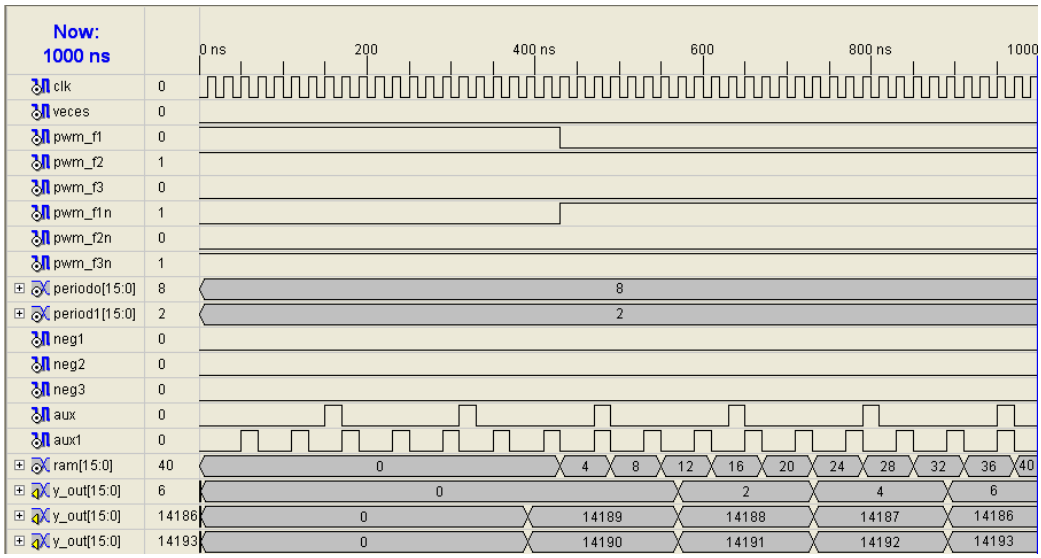
Figura 29. *Simulación para una salida con periodo de 8.2355 mS, con una relación de periodos de 21 con amplitud aumentada*



En la figura 30 se observa el inicio de la simulación, el generador de onda triangular “ram”, empieza su operación en el instante en que se ha iniciado el calculo de las ondas seno (“y_out”), este proceso se demora aproximadamente 280 nS, pero solo representa un retraso en el inicio del proceso, ya que las salidas inician su operación en el instante que los cálculos sean realizados.

También se observa la diferencia entre la frecuencia de las señales “aux” y “aux1”, lo que representa la diferencia entre la frecuencia de la onda seno y la onda triangular.

Figura 30. *Simulación para una salida con periodo de 8.2355 mS, con una relación de periodos de 21 detalle inicio*



4.3.1.2 Salida de pulsos

En la figura 31 se observa la salida de la FPGA para una de las ramas del inversor.

En la figura 32 se realizó un acercamiento a la señal de salida de la FPGA, con el fin de visualizar que no existen rebotes o traslapes en las salidas.

En la figura 33 se puede visualizar el desfase entre dos ramas diferentes del inversor.

Figura 31. *Salida de la FPGA para una misma rama del inversor*

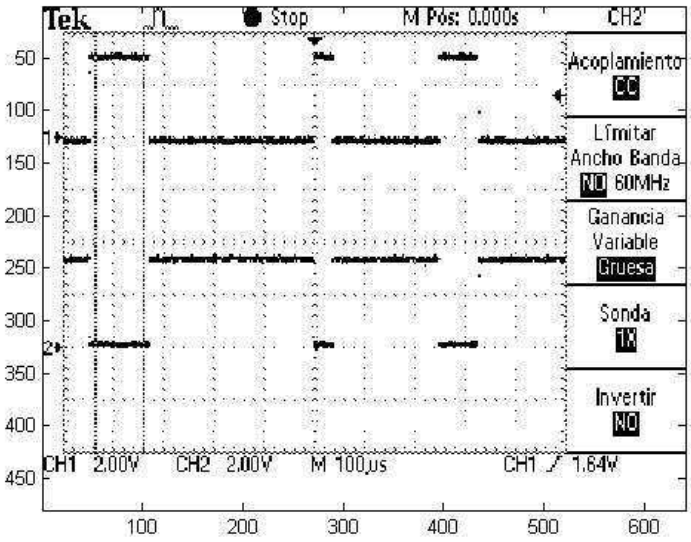


Figura 32. *Salida ampliada de la FPGA para una misma rama del inversor*

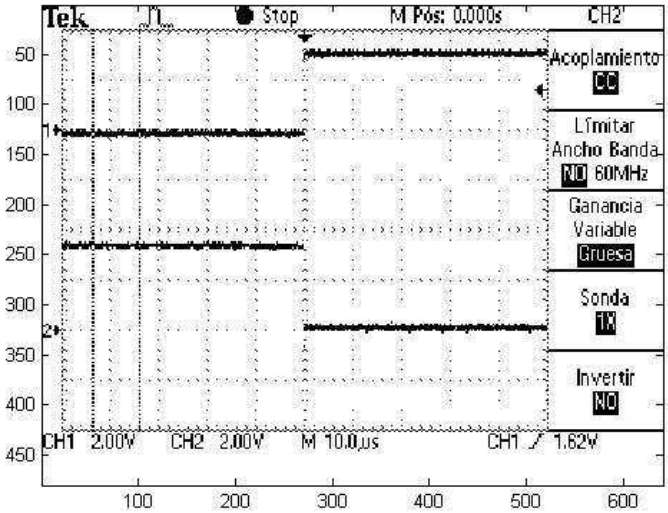
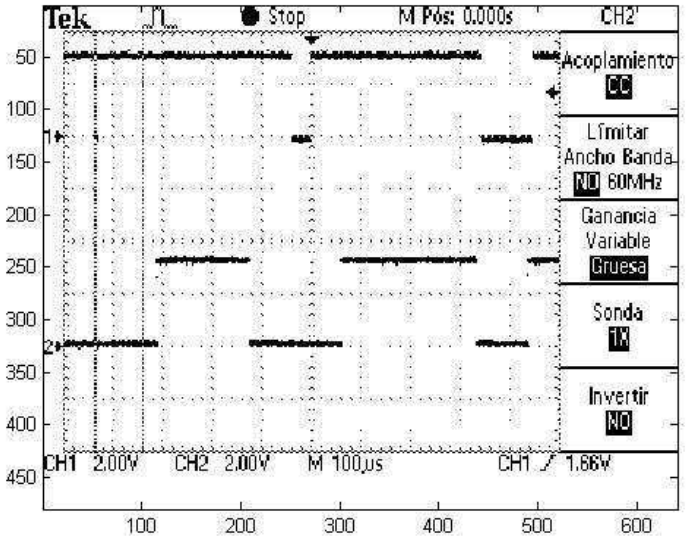


Figura 33. *Salidas de la FPGA para las ramas restantes del inversor*



4.3.1.3 Salida inversor

En las figuras 34, 35, 36 y 37 se observan las formas de onda en la salida del inversor, luego de aplicar los pulsos de la FPGA, conservando el montaje de la figura 11.

Las ondas resultantes presentan gran deformación, a pesar que la relación entre los periodos de la onda triangular y la onda seno vario, y se aumento la amplitud de la onda triangular.

Figura 34. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15*

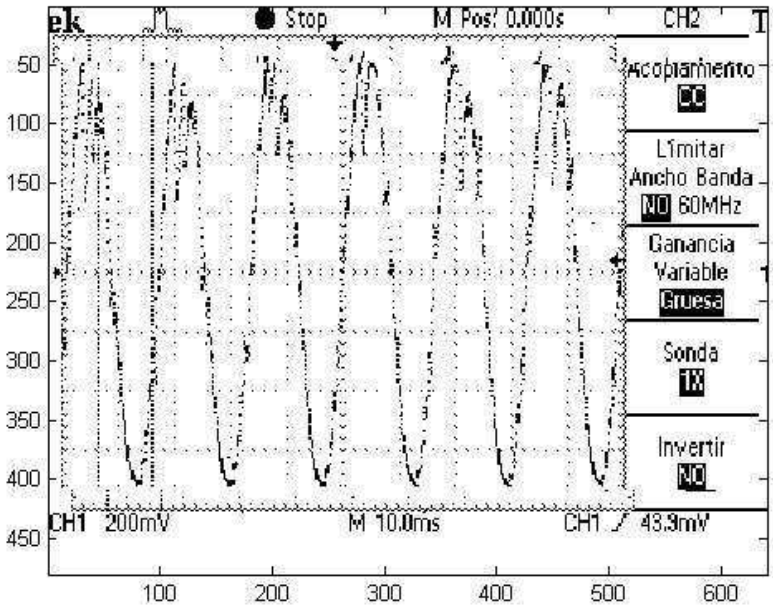


Figura 35. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15 con la amplitud aumentada 50%*

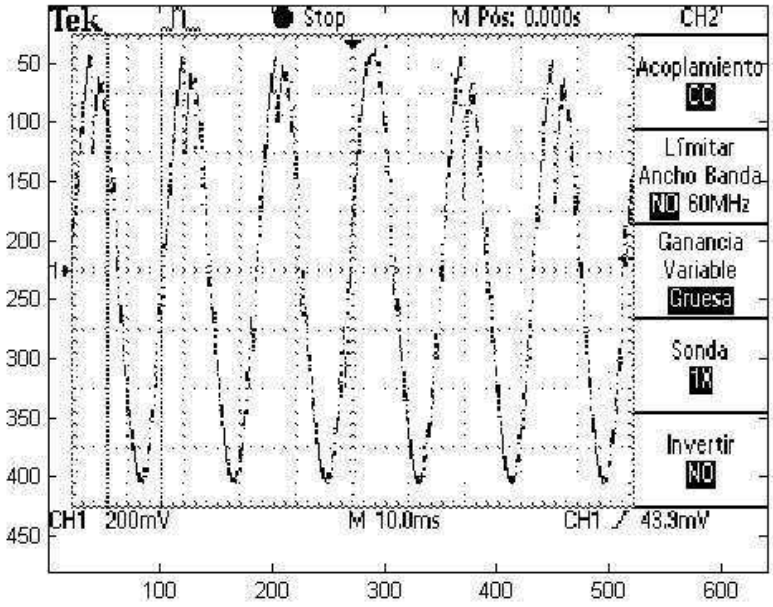


Figura 36. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 21*

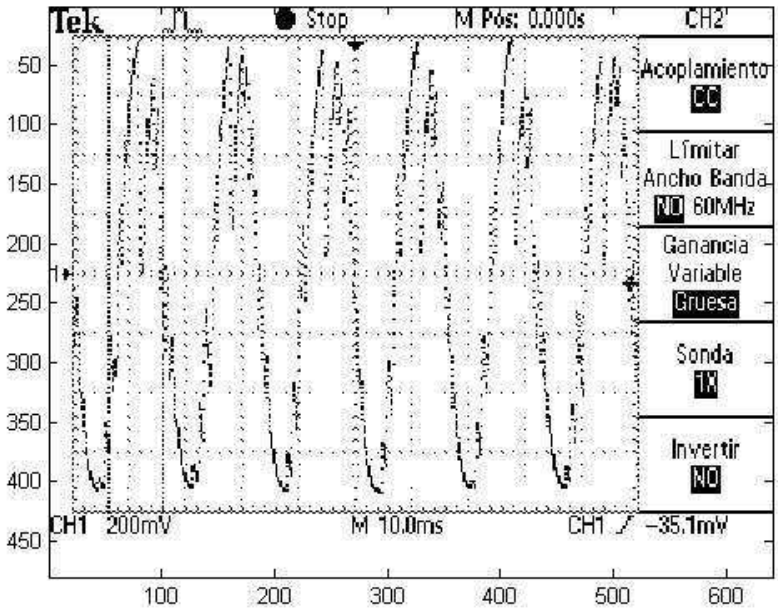
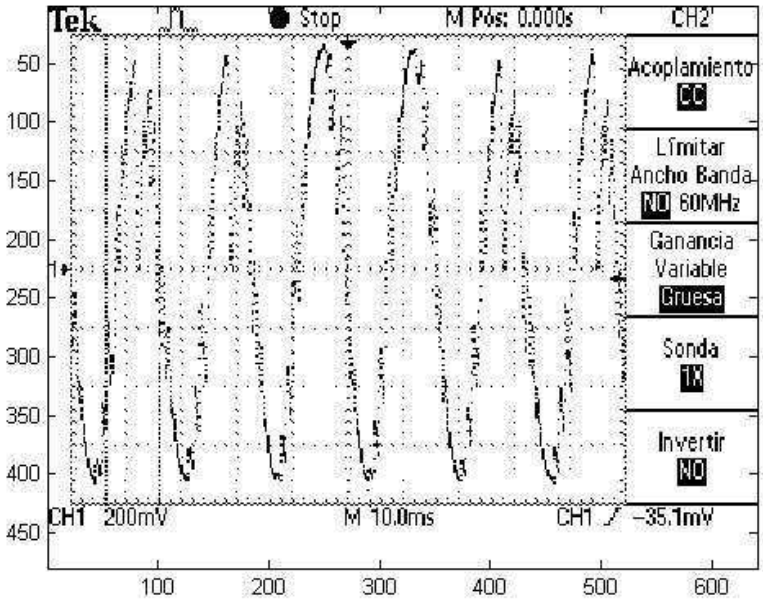


Figura 37. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 21 con la amplitud aumentada 50%*

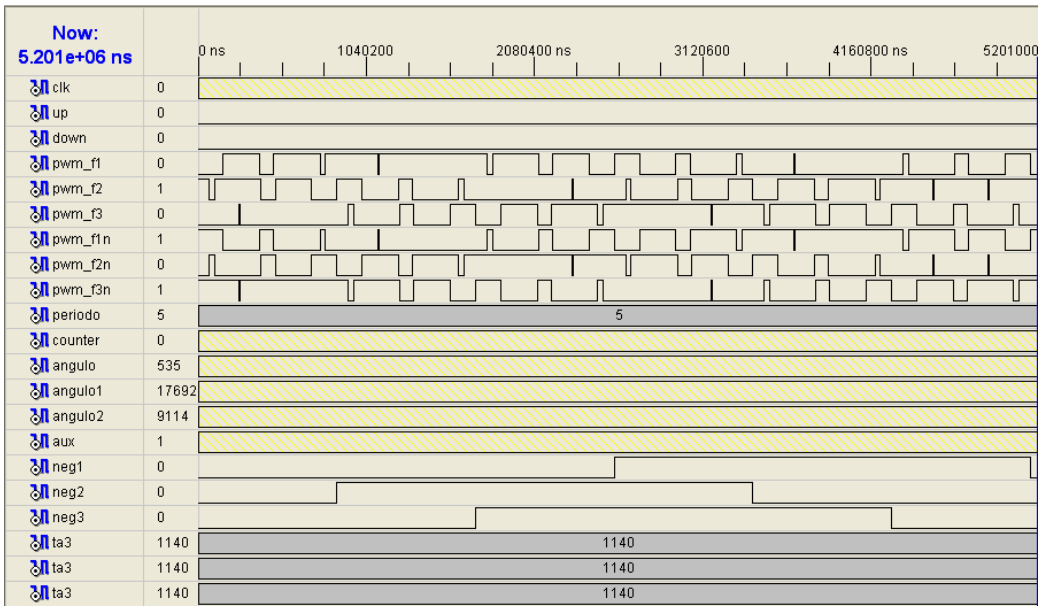


4.3.2 SPWMA

4.3.2.1 Simulación

En la figura 38 se trabajo con un periodo de 5.147 mS y un cociente de 15 entre el periodo de la onda triangular y de la onda seno, esta incluye el comportamiento de las señales “neg1”, “neg2” y “neg3”, las cuales se encargan de la negación del medio ciclo de la señal, el grupo de salidas (“pwm_f1”, “pwm_f2”, “pwm_f3”, “pwm_f1n”, “pwm_f2n” y “pwm_f3n”) muestra una forma acorde al método de modulación pwm convencional.

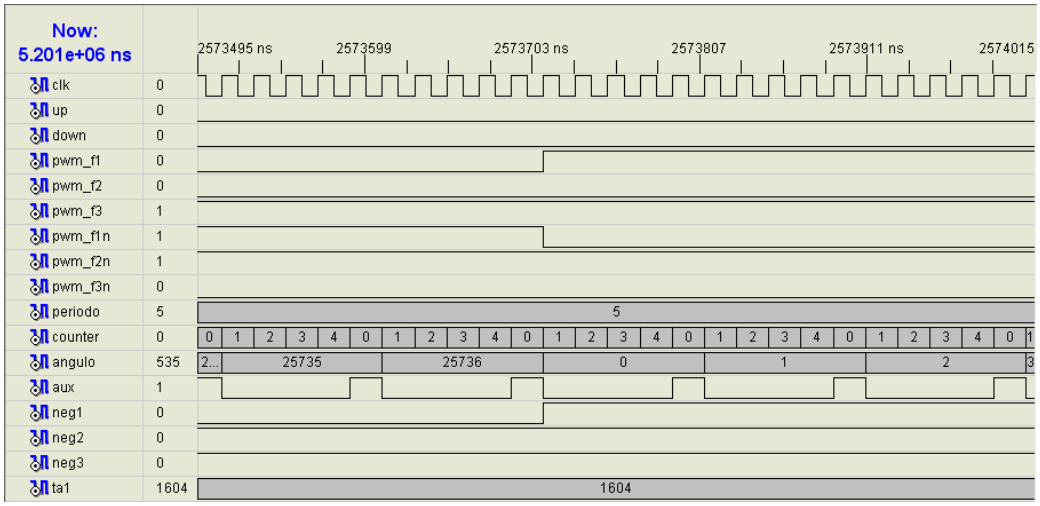
Figura 38. *Simulación para una salida con periodo de 5.147 mS, con una relación de periodos de 15*



En la figura 39 se observa como las señales “neg1”, “neg2” y “neg3”, niegan su valor cada vez que termina un ciclo, es decir cuando los contadores de (“angulo”, “angulo1” y “angulo2”) son iguales a la constante “pi”. El contador de

tiempo (“counter”) cambia de 0 al 4, representando los cinco estados de la entrada “periodo”. La señal “aux” habilita los cambios en el contador “angulo”, cuando pasa de su valor 25736 al 0, lo cual representa el final del medio ciclo, se produce un cambio en la señal “neg1”.

Figura 39. *Simulación con detalle en el auxiliar, negador y contador de angulo*



Para aumentar la amplitud de la onda triangular, se aplicar un uno lógico en la entrada “up”, con el fin de ver una disminución en la onda de salida del inversor, esto observa en la figura 40:

El aumento en la amplitud de la onda triangular hace que los pulsos de salida que se encuentran muy cercanos se hagan más definidos, como se ve en la figura 41.

En las figuras 42 y 43, se observa la salida para un cociente entre periodos de 21, esta se presenta como una opción para mejorar la calidad de la salida. También es claro la forma como los pulsos disminuyen su tamaño.

Figura 40. *Simulación con detalle en el aumento de la amplitud y la entrada up*

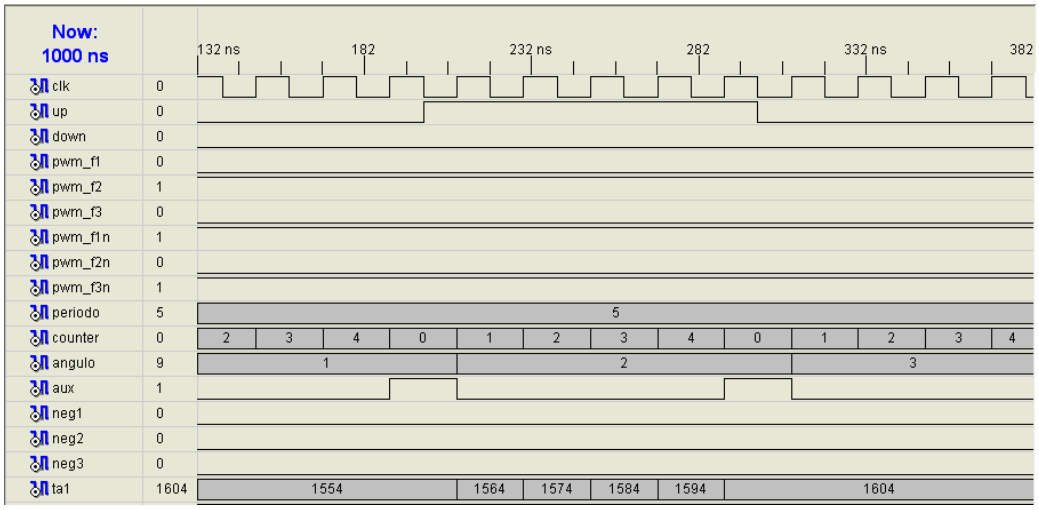


Figura 41. *Simulación para una salida con periodo de 5.147 mS, con una relación de periodos de 15 con amplitud aumentada 10%*

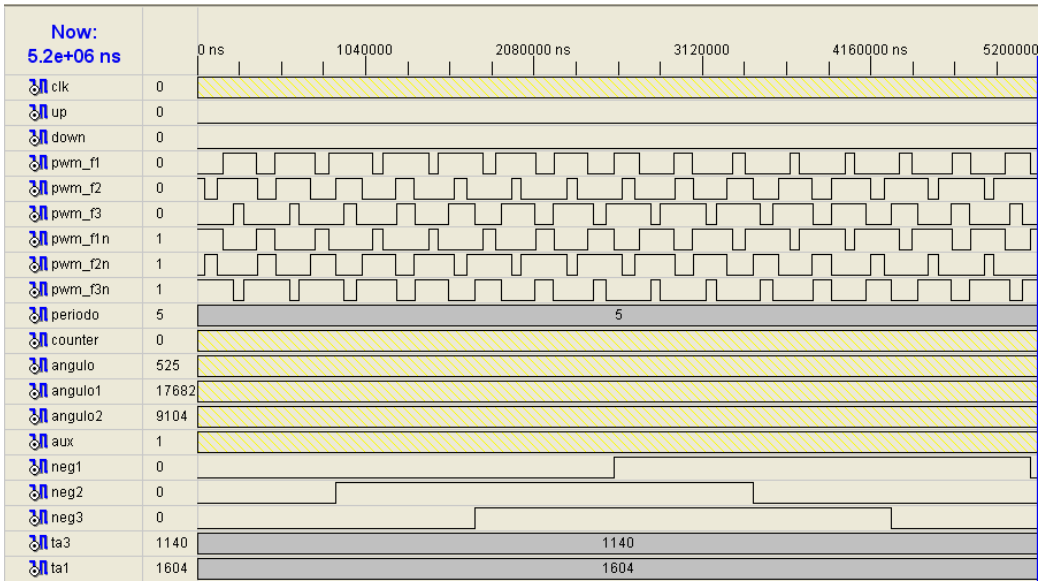


Figura 42. *Simulación para una salida con periodo de 5.147 mS, con una relación de periodos de 21*

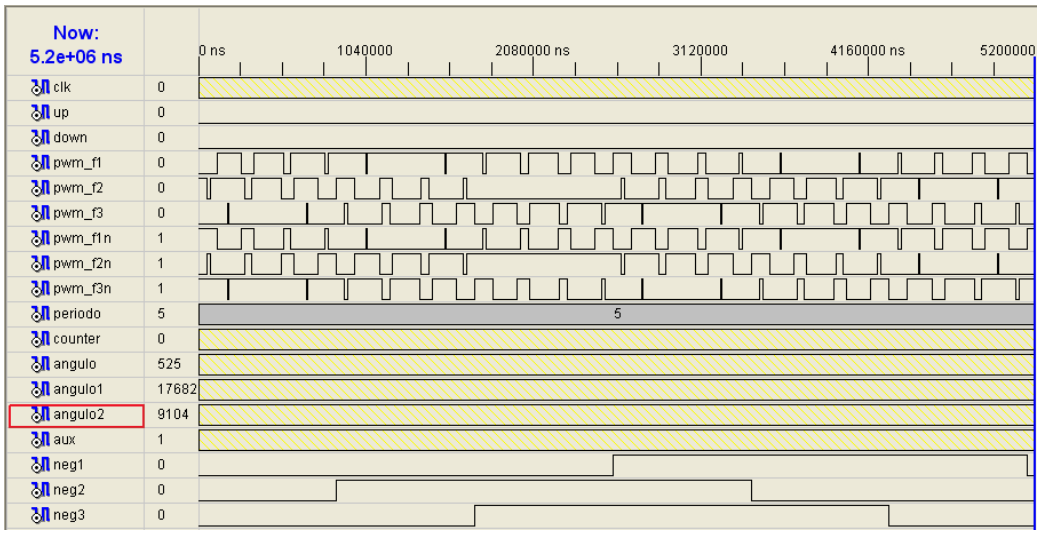
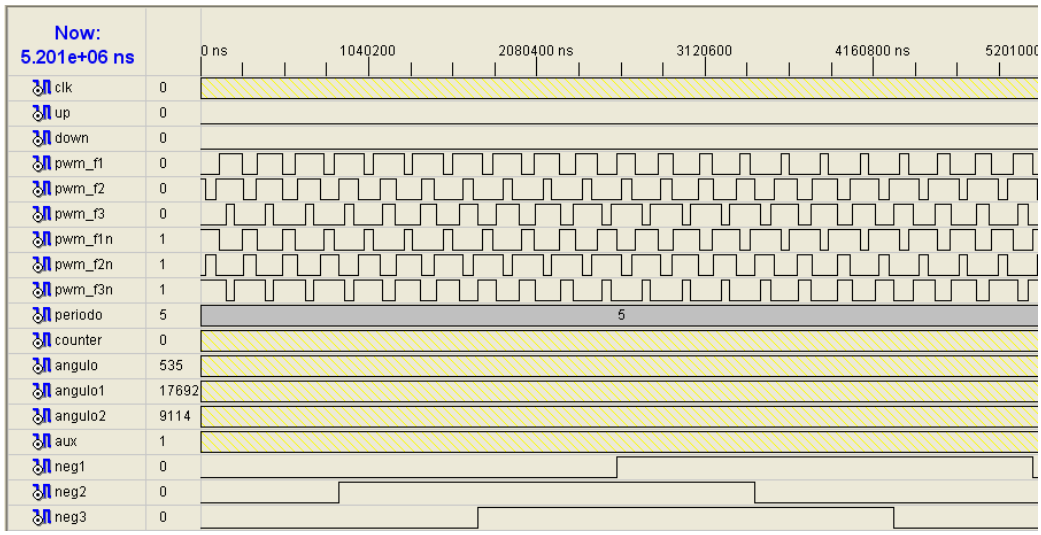


Figura 43. *Simulación para una salida con periodo de 5.147 mS, con una relación de periodos de 21 con amplitud aumentada 10%*



4.3.2.2 Salida de pulsos

En la figura 44 se observa la salida de la FPGA para una de las ramas del inversor.

En la figura 45 se realizó un acercamiento a la señal de salida de la FPGA, con el fin de visualizar que no existen rebotes o traslapes en las salidas.

En la figura 46 se puede visualizar el desfase entre dos ramas diferentes del inversor.

Figura 44. *Salida de la FPGA para una misma rama del inversor*

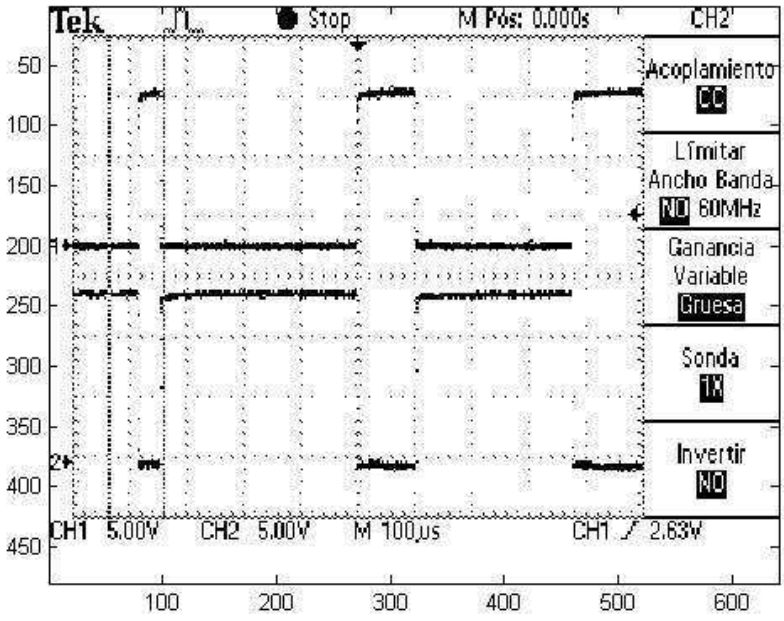


Figura 45. *Salida ampliada de la FPGA para una misma rama del inversor*

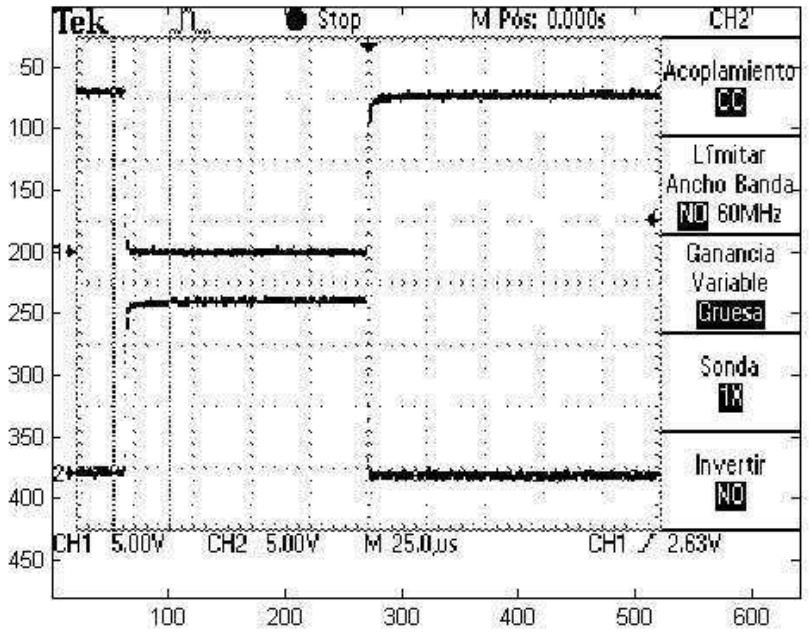
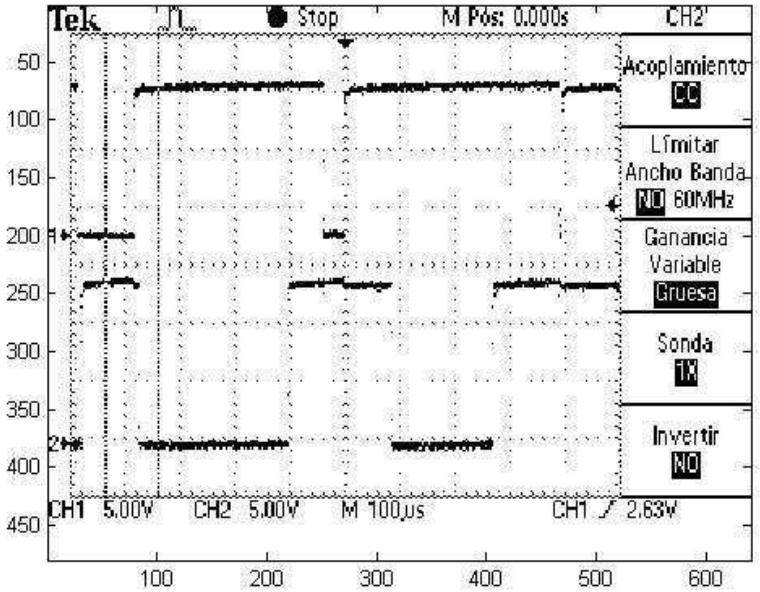


Figura 46. *Salidas de la FPGA para las ramas restantes del inversor*



4.3.2.3 Salida inversor

En las figuras 47, 48, 49 y 51 se realizan pruebas para 60 y 160 Hz con el fin de ver la forma de las señales de salida, conservando el montaje de la figura 11.

En las figuras 50 y 52 se incluye la salida para una onda triangular mayor a la onda seno.

Figura 47. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15*

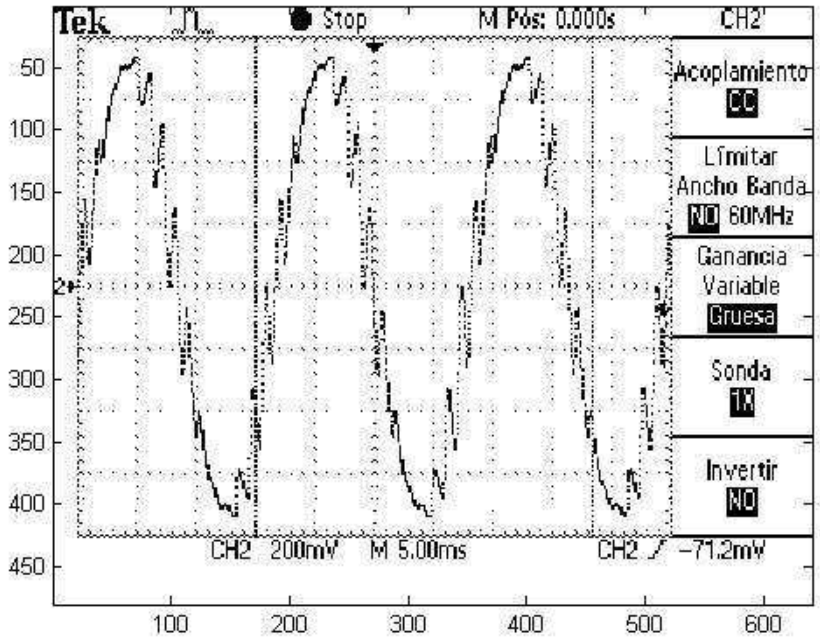


Figura 48. *Onda en la resistencia de carga a 60 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 21*

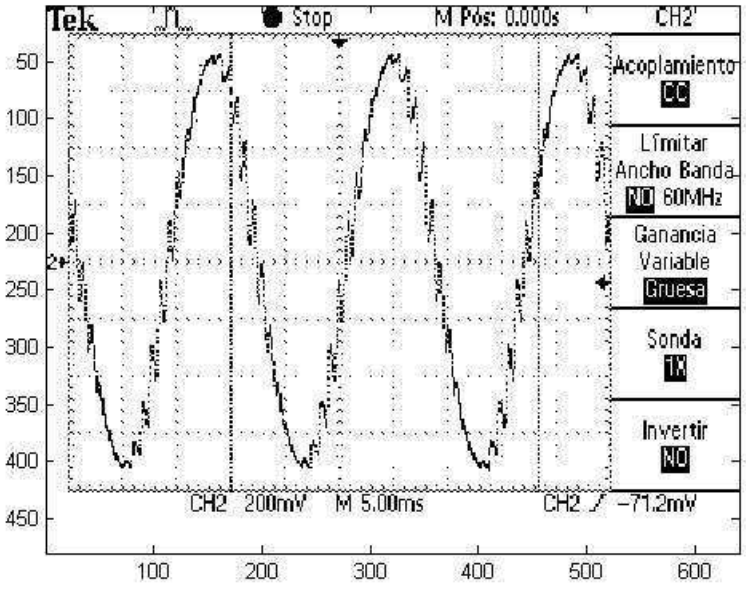


Figura 49. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15*

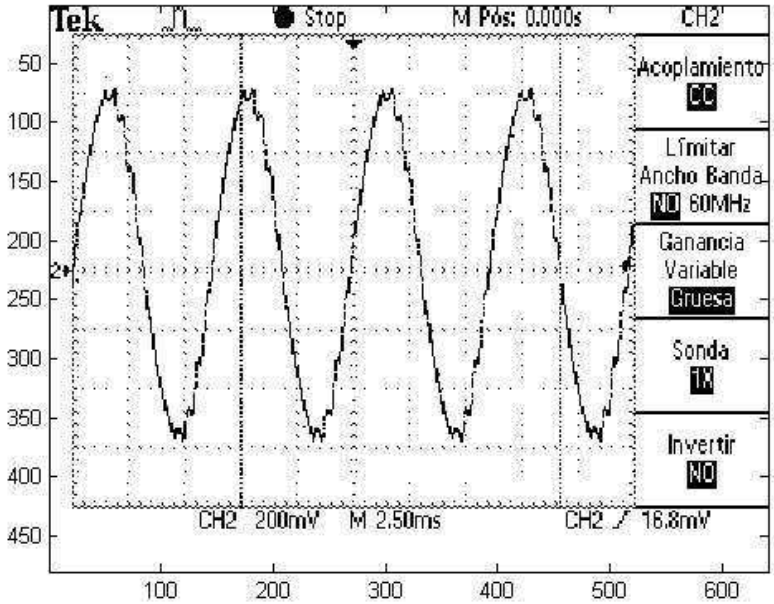


Figura 50. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15 con amplitud de la onda triangular aumentada*

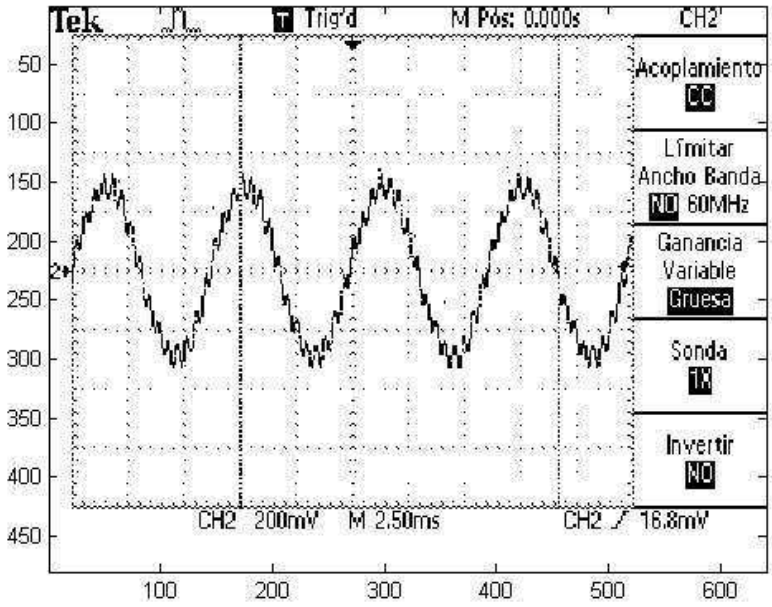


Figura 51. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 21*

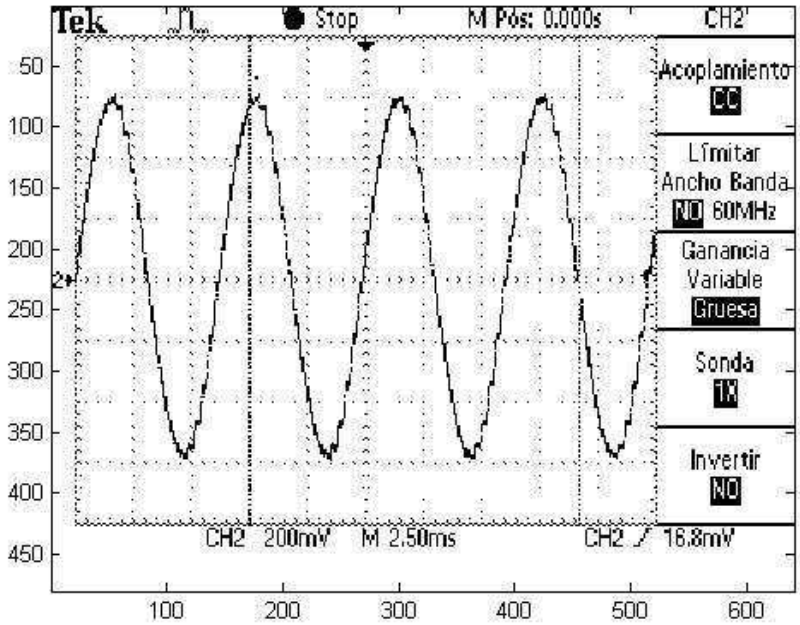
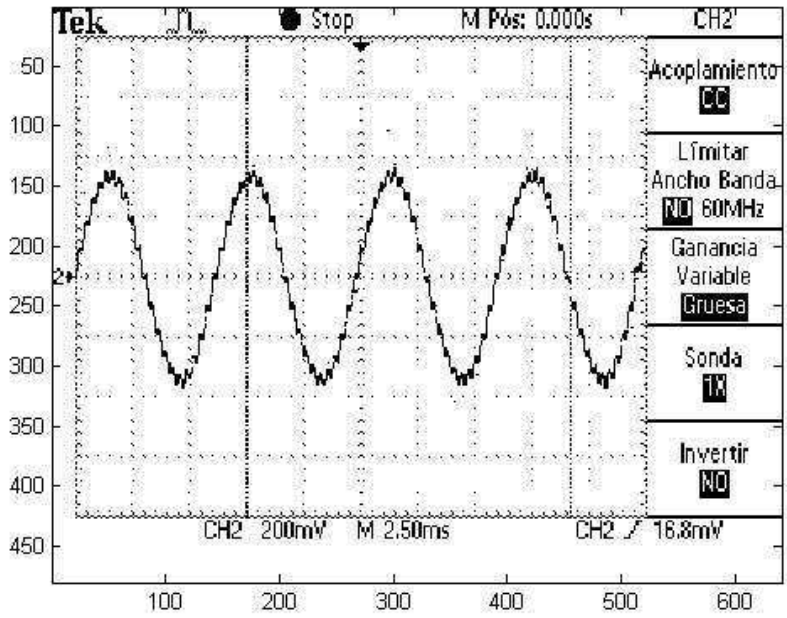


Figura 52. *Onda en la resistencia de carga a 160 Hz de 100 mH, 10 uF y 1000 ohmios con una relación de 15 con amplitud de la onda triangular aumentada*



4.4 COMPARACIÓN DE LA CALIDAD DE LAS SEÑALES DE SALIDA

Al momento de comparar las salidas obtenidas en cada dispositivo, la salida del dispositivo “SPWMA” es mejor, en comparación con los demás dispositivos, esto después de observar las figuras 13, 27, 37 y 52, ya que estas muestran la mejor forma de onda obtenida a la salida del inversor. Mostrando que la opción más adecuada es implementar un sistema “SPWMA”, sin embargo es importante aclarar que el dispositivo “SPWM” puede ser más práctico a la hora sensar una señal externa, ya que basta con reemplazar el modulo encargado de la generación de onda seno (“seno”), por una señal externa digitalizada, realizando los respectivos ajustes en el tamaño de las entradas y las señales internas, el dispositivo tiene una aplicación en sistemas de control.

4.5 TARJETA INTERFAZ Y DE POTENCIA

El inversor consta de dos etapas: una de potencia y la otra de control. En la etapa de potencia se disponen los IGBTs con sus conexiones de disparo, mientras que en la de control se localizan todos los componentes para activar y desactivar los IGBTs.

Al puente inversor llegan las señales de control provenientes de la FPGA por medio de dispositivos de interfaz y de voltajes que garantizan un aislamiento de seguridad entre los dispositivos de potencia y los de control.

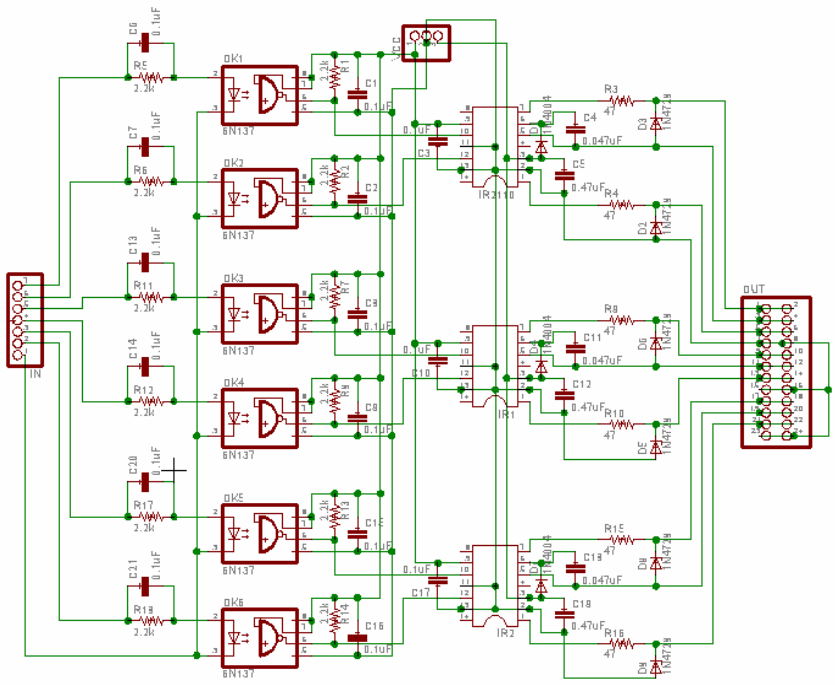
El inversor fue facilitado por el grupo de Electrónica de Potencia de la Universidad Tecnológica de Pereira, consta de semiconductores de potencia del tipo IGBT. De manera especifica se utilizo un modulo integrado fabricado por Internacional Rectifier [7] (modelo IRG4PC50FD), con los que se construyo cada rama del inversor. Estos IGBTs soportan 39A y un voltaje de 600V entre colector y emisor. Como el inversor es de tres fases, se utilizaron 6 de estos para su construcción.

Para que los IGBTs conduzcan es necesario generar una señal de voltaje de alrededor de 15V entre la puerta y el emisor. Por lo tanto, al cambiar de estado los IGBTs generan tierras flotantes en las fuentes de disparo, lo que hace necesaria la implementación de un circuito de disparo que sea capaz de generar los 15V independientemente para cada uno.

Para solucionar este problema de excitación adecuada se utilizó el circuito integrado IR2110 [6] de International Rectifier. Este circuito es capaz de excitar directamente un par de IGBTs conectados en configuración de medio puente. Por lo tanto, por cada rama se utilizo uno de estos circuitos, sumando un total de tres en el inversor. En el Anexo 15 se encuentran las hojas de datos técnicos de este circuito integrado.

En la figura 54 se muestra un diagrama esquemático del circuito de disparo. Este aislamiento tiene por objeto mantener el lado de potencia del inversor aislado del circuito de control. El elemento que genera el aislamiento es el optoacoplador modelo 6N137 [6], el cual se ha colocado para cada IGBT del inversor. Para mayor información, en el Anexo 16 se agregan las hojas de datos técnicos de este circuito integrado.

Figura 54. Diagrama esquemático del Circuito de Disparo



BIBLIOGRAFÍA

[1]. Clarke, G., Crabb, G., Swan, R., Vernon, P., Wyatt, A., "Re-configurable Computing", Group Project Report, Nottingham Trent University, May 1998.

[2]. C.S. Langensiepen, "Graphite - A Scaleable Language For Training Simulator Design" To appear in: *Proceedings of the European Simulation Symposium ESS'98*.

[3]. Garces U. Francisco Jose, Diez, A. Dario, Quintero, V. Cesar Augusto. Revista Energia y Computación. Volumen 21, 8a. 2002. <Disponible en la internet: http://energiaycomputacion.univalle.edu.co/edicion21/revista21_8a.html>

[4]. Gonzalez B. Danny A., Trujillo R. Cesar L., Lopez C. Hans I., "Descripción VHDL de un generador de Señales SPWM Hexafasicas". <Disponible en la internet :http://www.iberchip.org/IX/Articles/PAP-064.pdf>

[5]. Hauser, J., Wawrzynek, J., "GARP: A MIPS Processor with a Reconfigurable Coprocessor", *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, 12-21, April, 1997.

[6]. Infineon, < Disponible en la internet: http://www.infineon.com>

[7]. International Rectifier, < Disponible en la internet: http://www.irf.com>

[8]. L. H. R. Alfonso Alzate., *Electrónica de Potencia*., Universidad Tecnológica de Pereira, 1991

[9]. Olloz. S., Villar. E., Torroja, Y. Teres, .VHDL LENGUAJE ESTÁNDAR DE DISEÑO ELECTRÓNICO. McGraw-Hill, 1998.

[10]. Volder, J., .*The CORDIC Trigonometric Computing Technique*. IRE Trans. Electronic Computing, Vol.EC-8, Sept. 1959, pp330-334.

[11]. WEI Liu, Distributed Modular Controller Architecture for High Power Converter Appications. Tesis de grado. North Carolina State University. Raleigh, 2005. 109p. <Disponible en la internet : http://www.lib.ncsu.edu/theses/available/etd-12272005-224919/unrestricted/etd.pdf>

[12]. XILINX, LOGICORE *CORDICv 3.0 Product specification*. DS249 May 21, 2004 pdf.

[13]. XILINX, Spartan-3E Starter Kit Board User guide, UG230 (v1.0) March 9, 2006 pdf.

[14]. Xilinx Inc., < Disponible en la internet: http://www.xilinx.com.>

[15]. Yadong Liu., Kazuo Yamazaki Makoto Fujishima.,Jiangang Liang., “Modeling and Cosimulation of FPGA-Based SVPWM Control for PMSM”. Industrial Electronics Society, Vol.6-10, Nov. 2005, pp1538-1543.

[16]. Zhaoyong Zhou., Tiecai Li., Takahashi, T., Ho, E.,” Design of a universal space vector PWM controller based on FPGA" . Applied Power Electronics Conference and Exposition, Vol 3 - 2004, Nov 2-6. 2004 pp1698 - 1702