Revista Tekhnê
2012, Vol. 9, 21–32

ISSN 1692-8407

Implementación de la transformada FFT sobre una FPGA orientada a su aplicación en convertidores electrónicos de potencia

Implementation of FFT transform on a FPGA oriented towards its application in power electronic converters

José I. Riaño
Universidad Distrital Francisco José de Caldas
jirianor@correo.udistrital.edu.co

César E. Ladino
Universidad Distrital Francisco José de Caldas
celadinoo@correo.udistrital.edu.co

Fredy H. Martínez
Universidad Distrital Francisco José de Caldas
fhmartinezs@udistrital.edu.co

La evolución de los esquemas de control para los convertidores electrónicos de potencia exige capacidades de procesamiento cada vez más altas, con altos consumos de tiempo y de recursos. En cuanto a compatibilidad, uno de los retos más importantes es el relacionado con distorsión armónica y el factor de potencia. En el diseño de sistemas electrónicos para medición y corrección, uno de los algoritmos más importantes es el de la FFT (*Fast Fourier Transform*). La FFT es un eficiente algoritmo que permite calcular la transformada de Fourier discreta (TFD) y su inversa. Dadas las características de la transformada, es posible analizar una señal vista desde su espectro en frecuencia. Este artículo documenta el diseño de una FFT embebida en una FPGA (*Field Programable Gate Array*) para el análisis de contenido armónico en señales de corriente de baja potencia.

Palabras clave: contenido armónico, convertidor de potencia, FFT, FPGA

The evolution of control schemes for electronic power converters requires increasingly higher processing capabilities with high consumption of time and resources. For compatibility, one of the most important challenges is related to harmonic distortion and power factor. In the design of electronic systems for measuring and correcting, one of the most important algorithms is the FFT (Fast Fourier Transform). The FFT is an efficient algorithm to compute the discrete Fourier transform (TFD) and its inverse. Given the characteristics of the transform, it is possible to analyze a signal viewed from its frequency spectrum. This article reports an FFT design embedded in a FPGA (Field Programmable Gate Array) for the analysis of harmonic signals contained in low power current.

Keywords: harmonic content, FFT, FPGA, power converter

Introducción

Es indudable la apertura en desarrollo que se dió a partir de la aparición y demostración de las series y la transformada de Fourier (Barbu, Kaminsky, y Trahan, 2005; Donnelly, 2006; Jutamulia, Song, y Zhang, 2007). En varias de sus aplicaciones, especialmente en la ingeniería, ha permitido actualmente desarrollar un gran número de soluciones a distintos problemas y exigencias que se imponen en el análisis de señales electrónicas. Por esta razón es completamente justificable dar por lo menos un vistazo general a esta herramienta y más precisamente a una de las posibles aplicaciones.

El cálculo o procesamiento de la transformada de Fourier de forma manual resulta un tanto engorroso e ineficiente en términos de tiempo y recursos, incluso cuando esta se implementa por software utilizando el algoritmo FFT (*Fast Fourier Transform*) (Bian, Zhou, y Li, 2008; Yanhui Liu, Nie, y Liu, 2008) dados los requisitos particulares de las aplicaciones. Por tanto, dada la importancia y capacidad de análisis de la estrategia, surge la necesidad de realizar dicho cálculo de forma eficiente, optando entonces por implementaciones dedicadas embebidas en un determinado hardware (Hori, 2008; Kauker, Sanftmann, Frey, y Ertl, 2010; Prasad, Ray, y Dhar, 2010; van der Byl, Wilkinson, y Inggs, 2011).

Dadas las características de la transformada, es posible analizar una señal vista desde su espectro en frecuencia, lo que arrojará la información precisa para determinar su factor de calidad y posteriormente ser corregido de ser necesario. Es esta investigación, se busca una implementación eficiente y rápida de la FFT para su utilización en equipo de medida y corrección activa del factor de potencia, aplicaciones en las cuales es importante el análisis del contenido armónico de las corrientes (Vásquez y Martínez, 2011). Para ello se opta en una implementación paralela dedicada sobre una FPGA (Calderón y Parra, 2010), la cual es alimentada directamente por las muestras de corriente desde un microcontrolador.

Especificación y diseño

Perfil del diseño

La finalidad principal del prototipo sujeto a diseño, es brindar una herramienta de trabajo capaz de permitir la visualización y el análisis del contenido armónico de una señal de corriente tomando como base de análisis el estándar IEC 61000-3-2 (McGranaghan y Beaulieu, 2006; Patil, Patil, y Kyatanavar, 2010), que específica los límites para las emisiones de corriente armónica en equipos monofásicos de hasta 16 A, para su posterior aplicación en convertidores electrónicos de potencia. Dichos convertidores pueden ser equipos de laboratorio como fuentes de voltaje, reguladores de voltaje, rectificadores, cargadores de baterías, entre otros.

Luego de muchas pruebas de laboratorio con diferentes algoritmos y configuraciones (con respecto a su utilización final en análisis de contenido armónico), se optó por implementar el algoritmo *Radix-4 diezmado en tiempo de Coley-Tukey* (Villalba, Antelo, Bruguera, y Zapata, 1998; Weidong y Wanhammar, 1999), aprovechando el rendimiento frente a la utilización de recursos y el tiempo de transformación, así como la versatilidad con que se cuenta para especificar el tamaño y formato de la señal de entrada. El algoritmo Coley-Tukey, está basado principalmente en la premisa *Divide y vencerás*, el cual una vez toma una señal de entrada con *N* muestras, la descompone en otras cuatro subseñales de *N/4* y así sucesivamente hasta obtener una FFT de una señal de tamaño 4, reduciendo notablemente el número de sumas y multiplicaciones respecto a cualquier otro algoritmo, presentando una gran eficiencia a la hora de tratar el problema de estimación del contenido armónico de una señal.

En cuanto a la adquisición de datos de entrada, se cuenta con la posibilidad de introducir cualquier tipo de señal digitalizada. Como también a la salida obtener de manera gráfica las componentes armónicas de dicha señal en un PC, para cualquier tipo de tratamiento o informe necesario.

Descripción del prototipo

El prototipo de desarrollo básicamente trata de un sistema para visualizar las componentes armónicas de una señal determinada. Está basado principalmente en un bloque FFT, el cual una vez calcula la transformación, entrega dos vectores, parte real e imaginaria de dicha señal. Estos vectores pasan a través de un bloque específico para obtener su magnitud. Los bloques están implementados en una FPGA Spartam 3XCS400 la cual finalmente se comunica a un computador equipado con un software a medida para capturar y visualizar datos desde el puerto serial mediante un bloque RS232.

El prototipo diseñado posee las siguientes características:

- Tensión de alimentación de 5.5 V DC.
- Tensión de funcionamiento de 3.3 V DC.
- Tamaño de transformación 128 puntos.
- Capacidad de visualizar hasta el armónico de orden 51.
- Microcontrolador que proporcionara la señal a tratar (a futuro se podrá implementar un convertidor análogo digital dedicado para acondicionar la señal de entrada).
- Velocidad de transmisión de datos al computador a 115200 Baudios.

El diagrama de bloques para este sistema (Fig. 1) consta de cinco bloques principales:

- Bloque Entrada de datos.
- Bloque FFT.
- Bloque Magnitud.
- Bloque Envió de datos.
- Bloque Visualización.

Especificación y simulación

Bloque de entrada de datos. La señal de entrada se compone de una onda senoidal con frecuencia fundamental de 60 Hz sumada con otras senoidales de diferentes frecuencias múltiplos de 60 (esto corresponde a la corriente de entrada de un equipo monofásico con característica no lineal), muestreada para obtener *N* datos (128). Es decir, cada dato está separado 130,2 μs uno del otro para completar un periodo (ecuación 1).

Fecha recepción del manuscrito: Octubre 17, 2012
Fecha aceptación del manuscrito: Diciembre 10, 2012

José I. Riaño, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas; César E. Ladino, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas; Fredy H. Martínez, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas.

Esta investigación fue financiada por: Universidad Distrital Francisco José de Caldas.

Correspondencia en relación con el artículo debe ser enviada a: José I. Riaño. Email: jirianor@correo.udistrital.edu.co

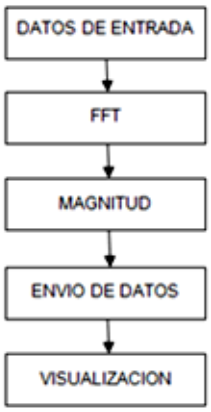

Figura 1. Diagrama de bloques del sistema.

$$t = \frac{T}{N} = \frac{0{,}0166}{128} = 130{,}2\,\mu s \qquad (1)$$

Donde t es el tiempo de separación entre un dato y el otro, T es el periodo de una señal senoidal de 60 Hz y N el número de muestras tomadas.

Una vez tomadas estas muestras, pasan por un proceso de digitalización con peso de 8 bits, y almacenadas, todo en un microcontrolador PIC16F87XA de Microchip. Estos datos ingresan directamente a la FPGA al bloque FFT. A continuación se presentan algunas simulaciones de los comportamientos de las señales en Matlab (Fig. 2 y 3).

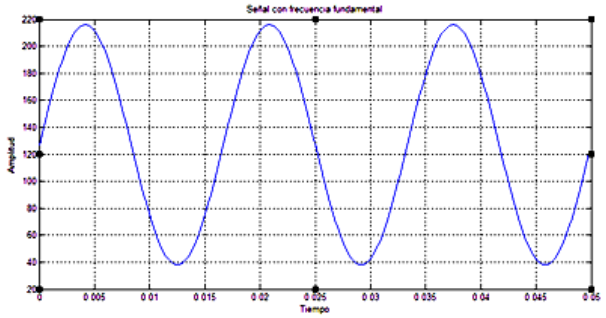

Figura 2. Señal frecuencia fundamental pura.

Para efectos de simulación práctica, se diseñó un bloque en código VHDL el cual contiene la señal digitalizada (Fig. 4).

Las simulaciones equivalentes obtenidas del bloque en Modelsim de Xilinx se muestran en las Fig. 5 y 6:

Bloque FFT. El bloque de la transformada rápida de Fourier (FFT), es un núcleo de propiedad intelectual de Xilinx, utiliza el algoritmo Coley-Tukey con el método de decimación en tiempo (DIT), para las arquitecturas Radix-4 y Radix-2, mientras que para la arquitectura *pipelined streaming* utiliza el método de decimación en frecuencia (DIF).

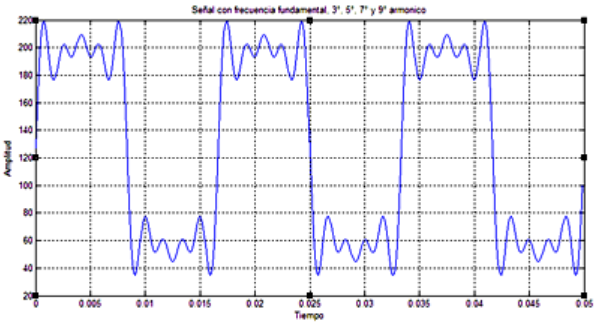

Figura 3. Señal frecuencia fundamental, 3º, 5º, 7º y 9º armónico.

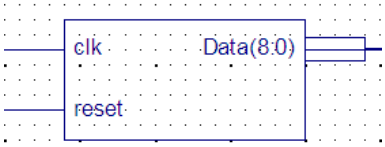

Figura 4. Esquemático del bloque de entrada.

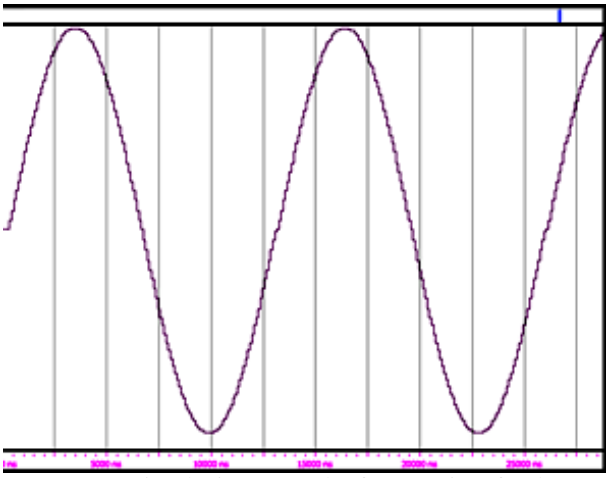

Figura 5. Simulación señal frecuencia fundamental (Modelsim).

El bloque FFT calcula un núcleo DFT o IDFT para un vector de entrada (Bx) de N valores complejos donde N puede ser $2m$, m = 3 a 16, y Bx está en el rango de 8 a 24 bits.

Este algoritmo puede ser implementado en FPGAs Xilinx de la familia Virtex y Spartam. Posee las siguientes características específicas:

- Posibilidad de obtener la FFT o la inversa IFFT.
- Transformar tamaños desde 8 a 65536 puntos.
- Datos de entrada con rango de 8 a 24 bits.
- Tres tipos de aritmética: (1) Sin escala (mejor precisión) de punto fijo. (2) Escala de punto fijo, y (3) Bloque de punto flotante.
- El redondeo puede ser truncado o redondeo convergente.

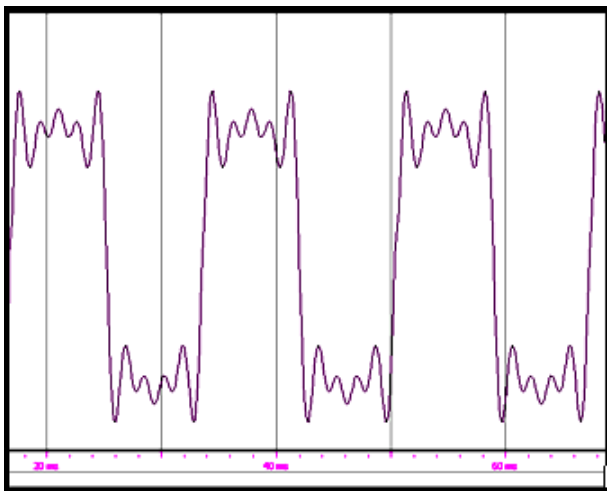

Figura 6. Simulación señal frecuencia fundamental, 3º, 5º, 7º y 9º armónico (Modelsim).

• Tres tipos de arquitectura dependiendo el tamaño y el tiempo de transformación: (1) Pipelined streaming I/O: Permite el tratamiento de los datos continuos. (2) Radix-4 burst I/O: Carga y descarga de datos por separado, utilizando un enfoque iterativo. Es menor en tamaño que el Pipelined streaming, pero ya tiene un tiempo de transformación establecido. Y (3) Radix-2 burst I/O: Carga y descarga de datos por separado, al igual que el Radix-4, menor en tamaño pero el tiempo de transformación es más largo (Fig. 7).

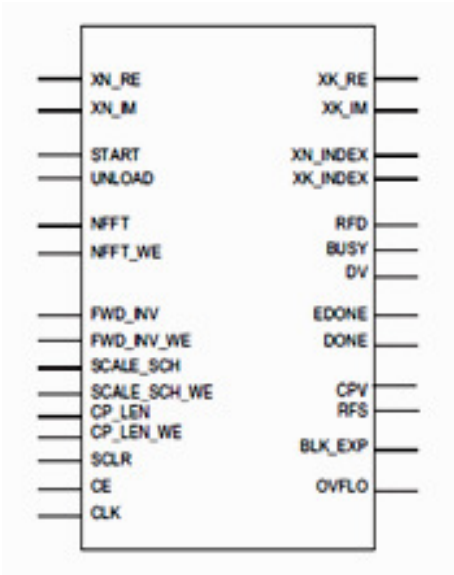

Figura 7. Esquemático núcleo FFT IP CORE.

Bloque Magnitud. El bloque Magnitud proviene de una de las funciones del núcleo CORDIC (*Coordinate Rotational Digital Computer*), el cual implementa un algoritmo generalizado capaz de resolver los siguientes tipos de ecuaciones:

- Conversión de coordenadas rectangular a polar.
- Conversión de coordenadas polar a rectangular.
- Ecuaciones trigonométricas.
- Ecuaciones hiperbólicas.
- Raíz cuadrada.

En este caso, solo se considera la conversión de coordenadas rectangular a polar, ya que solo se utiliza este bloque para obtener la magnitud de las componentes sinusoidales a la salida del bloque FFT.

Los resultados de la conversión son la magnitud X, y la fase θ del vector de entrada (X, Y). Las ecuaciones del núcleo CORDIC para la conversión de coordenadas son (ecuaciones 2 y 3):

$$M = \sqrt{(X^2 + Y^2)} \tag{2}$$

$$\theta' = tan^{-1}\frac{X}{Y} \tag{3}$$

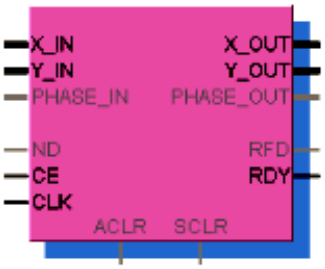

Figura 8. Esquemático bloque Magnitud.

Bloque Envío de datos. Una vez se tienen los datos a la salida del bloque Magnitud, es necesario enviarlos a un computador para visualizarlos, esto se logra mediante un bloque llamado Envió de datos, el cual fue diseñado mediante lenguaje VHDL. Este cumple con las siguientes especificaciones:

• Enviar 15 bits (un bit de inicio, ocho bits del dato, un bit de parada y cinco bits de espera entre un dato y otro).

• Velocidad de transmisión de 115.200 Baudios.

El reloj utilizado con la FPGA es de 50 MHz, es decir que cada ciclo de reloj será de 20 ns (ecuación 4).

$$P_r = \frac{1}{F_r} = \frac{1}{50 \times 10^6} = 20 \text{ ns} \tag{4}$$

Donde P_r es el periodo del reloj y F_r es la frecuencia del reloj de la FPGA.

Los datos de entrada están separados cada 130.2 μs, y es necesario enviar 15 bits, por lo cual se deben enviar a cada 8.68 μs (ecuación 5).

$$T_e = \frac{t}{N_b} = \frac{130\,\mu\text{s}}{15} = 8{,}68\,\mu\text{s} \approx 8{,}7\,\mu\text{s} \tag{5}$$

Donde T_e es el tiempo de envío de cada dato, t es el tiempo de separación entre un dato y el otro, y N_b es el número de bits (Fig. 9).

Figura 9. Esquemático bloque Envío de datos.

Bloque Visualización. Por medio del software REALTERM se realiza la captura de los datos enviados por el puerto serial desde la FPGA ya programada, y son guardados en un archivo de Excel para posteriormente manipulación por parte del usuario (visualización, análisis, etc). Esto se realiza después de programar el microntrolador con la señal de entrada, y realizar el respectivo diseño del esquemático final a programar en la FPGA.

Configuración del esquemático de simulación

Para asuntos de simulación, se diseñó un esquema en el cual se interconecta cada uno de los bloques anteriores con sus distintas configuraciones, a fin de tener una idea del funcionamiento del prototipo antes de proceder a programarlo tanto en la FPGA como en el microcontrolador (Fig. 10).

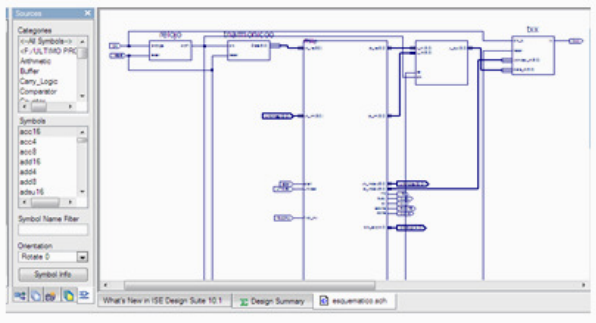

Figura 10. Esquemático de simulación.

Implementación y prototipo final

En esta sección se presenta cuáles son los parámetros finalmente seleccionados para la implementación y programación del prototipo. También se muestra el diagrama de conexión de cada uno de los bloques y la utilización de recursos de la FPGA.

Bloque de entrada de datos

La señal de entrada no es más que una tabla con 128 datos digitalizados con peso de 8 bits, procesada por el microcontrolador a partir de la lectura de corriente. Cuenta con la posibilidad de intercambiar la tabla por otra almacenada internamente, en la cual se tiene una configuración conocida de la señal de corriente de entrada, es decir con componentes armónicas previamente estudiadas.

Esta tabla esta descrita en código VHDL. Básicamente es un contador que en cada pulso coloca un dato de la tabla a la entrada de la FPGA.

Bloque FFT

Se optó por implementar una FFT de 128 puntos y longitud de palabra a 8 bits, ya que el bloque no presenta gran diferencia (en cuanto a tiempo de transformación) con implementaciones de 256, 512 o 1024. Por otra parte una implementación con 12 bits de entrada, proporciona más precisión pero mayor uso de *bonde* IOBs (pines de entrada/salida), representando mayor consumo de recursos del FPGA.

La configuración de los parámetros del bloque FFT se realizó de acuerdo a lo especificado en el *datasheet*. Este bloque cuenta con una interfaz gráfica, la cual permite seleccionar cada parámetro según sea su necesidad.

En la primera ventana que aparece se selecciona: el tamaño de la transformada, el reloj que utiliza, y el tipo de arquitectura para procesar la mariposa (Fig. 11).

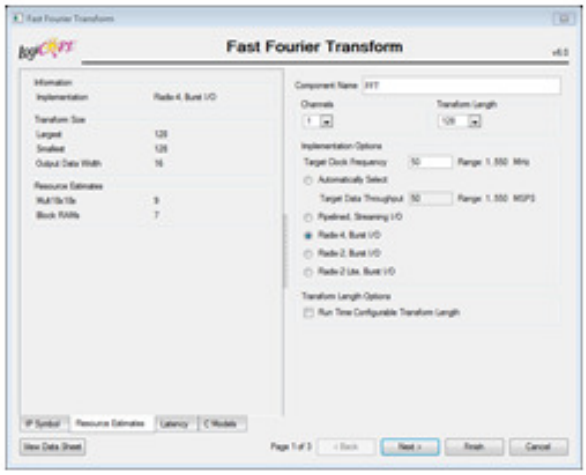

Figura 11. Configuración del bloque FFT paso 1.

La configuración usada para el tamaño de la transformada es de 128 muestras, con las cuales se podrá calcular hasta el armónico 64, y para las cuales está programado el bloque de entrada. El número de canales se fijó en 1, este parámetro indica cuantas transformadas se llevarán a cabo al interior del bloque. El reloj de la transformada se estableció en 50 MHz

para obtener una perfecta sincronización con el reloj interno de la FPGA.

Se eligió la arquitectura Radix-4 Burst I/O, a pesar que implica mayor uso de multiplicadores. Los recursos de la tarjeta son suficientes para esta implementación, además cuenta con una excelente respuesta en tiempo comparado con el Radix-2 Burst I/O que utiliza el doble de tiempo para la misma transformación. Una arquitectura en *Pipelined Streaming* utiliza una gran cantidad de multiplicadores, con lo que reduciría considerablemente los recursos para el resto del sistema así que sería inoficiosa su implementación en esta FPGA.

En la segunda ventana de configuración (Fig. 12) se establece el tipo de dato para la entrada y salida del bloque, la longitud de palabra para cada muestra, las opciones de escalado y redondeo, los pines adicionales y el orden de la salida.

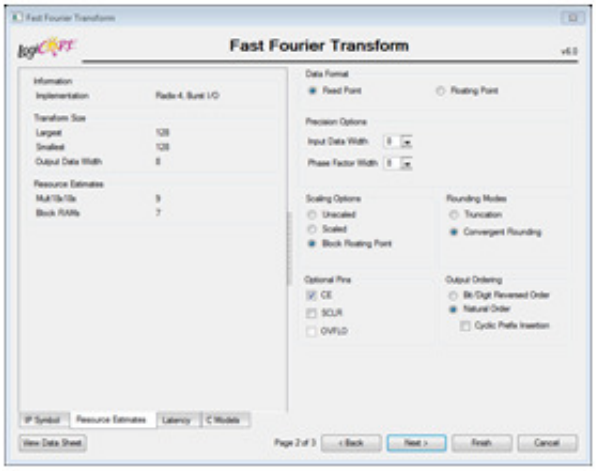

Figura 12. Configuración del bloque FFT paso 2.

En esta implementación se seleccionó el tipo de dato como punto fijo, con longitud de palabra de 8 bits, ya que las muestras procedentes del microcontrolador están en este formato. En las opciones de escalonamiento se decidió elegir el bloque de punto flotante teniendo en cuenta que al resultado se debe multiplicar por el exponente de escalamiento, que puede ser un factor de 1, 2, 4 u 8. Se descartó la opción de escala debido a que es necesario programar el factor de escalamiento en cada etapa, si la escala es insuficiente se producirá un desbordamiento, ya que la salida de la mariposa crecerá más allá del rango. Además, esta escala se aplica a cada etapa del algoritmo, lo que hace más lento el procesamiento de la FFT. Por otro lado, al no escoger escala la longitud de palabra de la salida aumenta considerablemente, por lo cual se hace necesario mayor cantidad de puertos en la tarjeta y por ende mayor utilización de recursos hardware.

Para la opción de redondeo se eligió el convergente, debido a que este hace un redondeo hacia abajo mientras que el truncamiento hace un recorte del exceso de bits, lo que se traduce en pérdida de información disminuyendo la precisión del sistema. Se habilito en la opción de pines opcionales el CE o habilitación de operación de la FFT, con el fin de evitar que esta realice operaciones en instantes no válidos.

Para el caso del ordenamiento de la salida, se eligió en orden natural para no alterar la disposición con la que viene la señal a lo largo del sistema.

En la última ventana de configuración del núcleo FFT se tomó el tipo de memoria RAM que se utiliza para almacenar las muestras seleccionando el método de optimización de resultados de la FFT (Fig. 13).

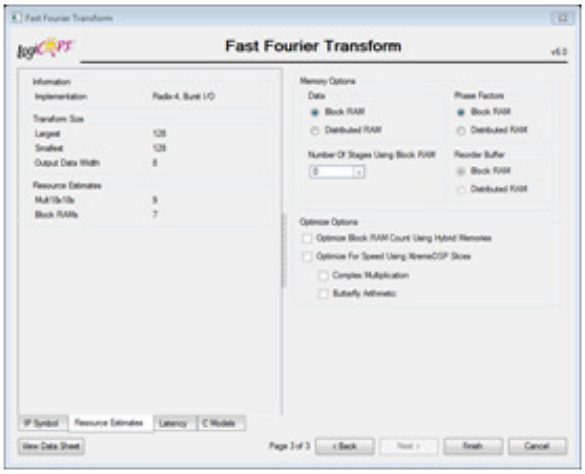

Figura 13. Configuración del bloque FFT paso 3.

En el tipo de memoria se eligió la de bloques de memoria RAM, ya que ocupa menos área del FPGA y su síntesis requiere menor tiempo.

Bloque Magnitud

Al igual que la configuración del bloque FFT, la configuración del bloque magnitud se realizó por medio de una interfaz gráfica al momento de generar el núcleo.

Como este núcleo está diseñado para cumplir varias funciones, es necesario seleccionar en la primera ventana la opción *traslate* la cual convierte coordenadas rectangulares a polares para obtener la magnitud, así mismo se selecciona el tipo de arquitectura en paralelo ya que de esta forma llegan los datos al bloque (Fig. 14).

La segunda ventana (Fig. 15) muestra las opciones que se tienen para el formato de la magnitud y la fase, así como también los pines opcionales con los que cuenta este bloque.

Por último, en la tercera ventana (Fig. 16) se configura el tipo de redondeo y la longitud de palabra, que continua siendo 8 bits para todo el sistema.

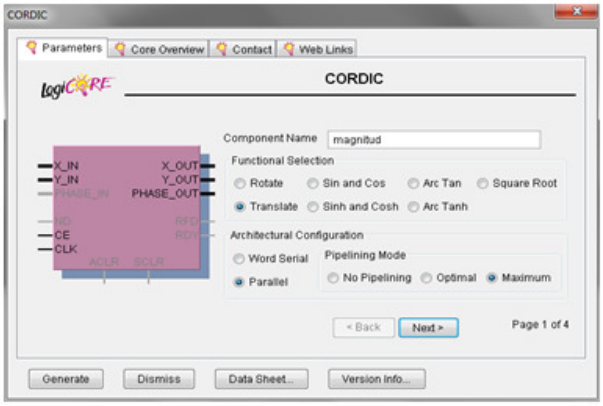

Figura 14. Configuración del bloque magnitud paso 1.

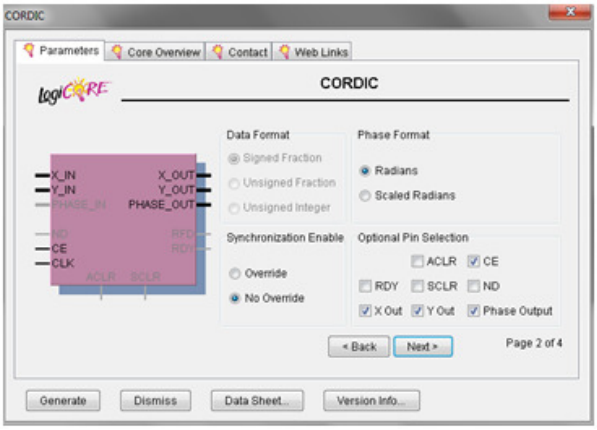

Figura 15. Configuración del bloque magnitud paso 2.

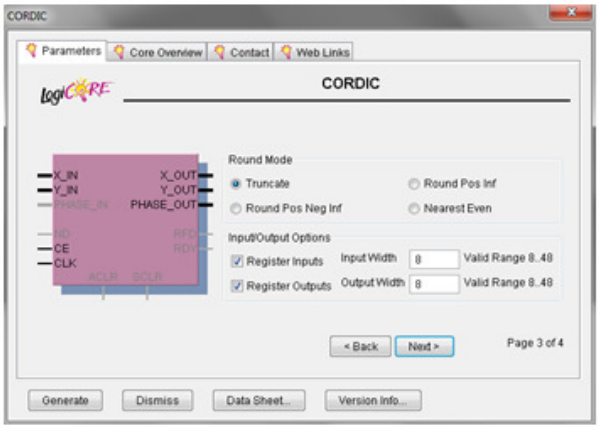

Figura 16. Configuración del bloque magnitud paso 3.

Bloque de envío de datos

De la misma manera que el bloque de entrada, este fue creado en lenguaje VHDL, tomando las secuencias de una máquina de estados, la cual hace el control y la comunicación de las señales que están a la salida de la FPGA, las cuales se enviaran al computador donde posteriormente son visualizadas.

El código se basa principalmente en un contador de 0 a 15 (*conta_baudios*) que comienza su cuenta justo cuando está en alto la señal de control que viene del bloque FFT (*xk_index*), a la salida (*tx*) se tiene un proceso de selección de la señal *conta_baudios* que está predeterminada con un bit de inicio, ocho bits de datos, un bit de parada y seis bits de espera entre un dato y el siguiente.

Bloque Visualización

Para el bloque de visualización se configuro el menú *Port* en el cual se especifica la velocidad de transmisión a (115.200 Baudios), además si se cuenta con paridad, bits de parada y cantidad de bits del dato a visualizar. También se puede configurar la presentación deseada del dato en la ventana del *RealTerm* (Fig. 17 y 18).

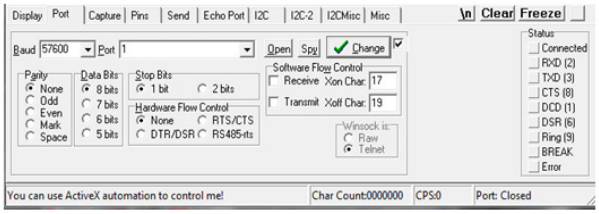

Figura 17. Configuración para el bloque visualización.

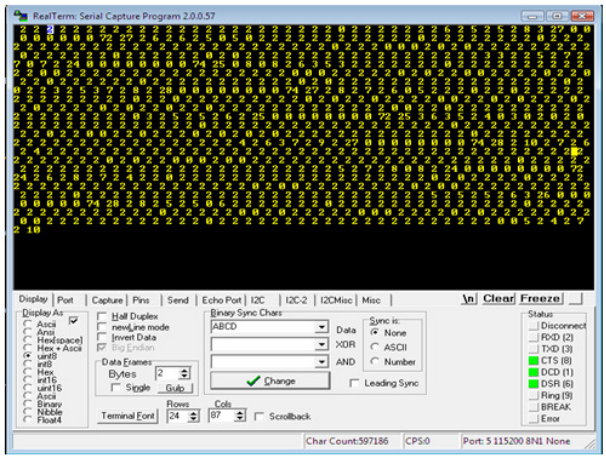

Figura 18. Resultados obtenido en el bloque visualización.

En el menú *Display* se configura el formato en que se desea ver el dato, ya sea binario, entero o hexadecimal, y por último en el menú *Capture* se indica en que archivo se guardaran los datos tomados, en este caso se almacenan en un archivo formato Excel para análisis posterior.

Prototipo de laboratorio

Obtenidos todos los bloques con sus respectivas especificaciones, se procede a crear un proyecto en el

ISE Project Navigator de Xilinx, interconectándolos en un esquemático, para su posterior síntesis e implementación (Fig. 19).

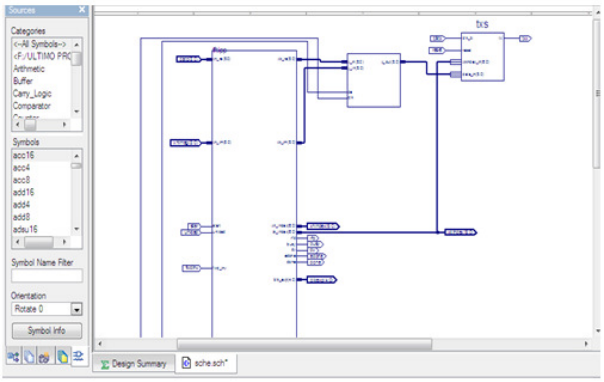

Figura 19. Esquemático de programación en la FPGA.

Comprobando que los procesos de síntesis e implementación son exitosos, se programa el archivo generado con extensión bit en la FPGA, y se realizan las pruebas pertinentes para verificar el comportamiento real del sistema.

Recursos utilizados

A continuación se presentan los resultados obtenidos para la implementación a 128 puntos de longitud y 8 bits de peso (Fig. 20).

Como se puede observar, la configuración seleccionada utilizó siete de los 16 bloques de memoria RAM, y nueve de los 16 multiplicadores, como las demás configuraciones, la diferencia radica en el consumo de *flip flops*, la ocupación de *slices* y el número de pines de entrada/salida IOBs, ya que obviamente la configuración a 12 bits de entrada consume más de estos.

A continuación se presentan fotografías del montaje del prototipo (Fig. 21 y 22).

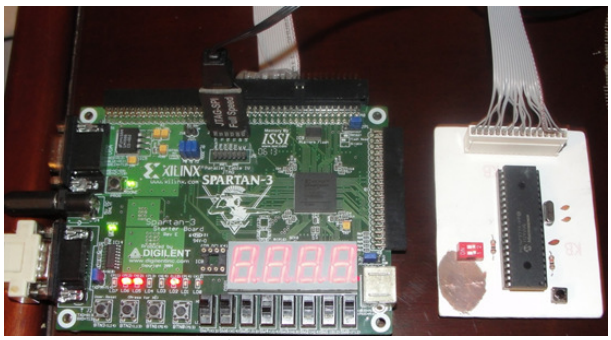

Figura 21. Foto montaje (1).

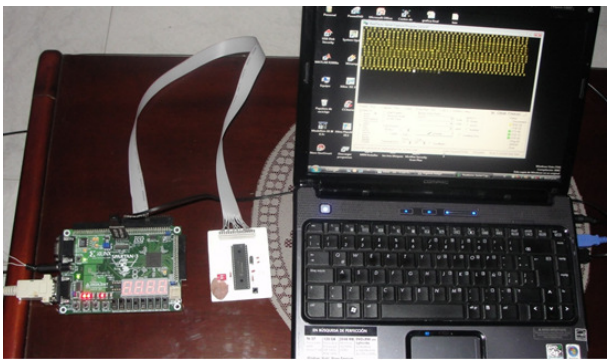

Figura 22. Foto montaje (2).

Evaluación de desempeño

Con el fin de corroborar el diseño y el comportamiento real del sistema, se efectuaron pruebas técnicas a la salida del prototipo. A continuación se muestran de manera gráfica los resultados obtenidos de las mismas, como también los comentarios generados por estas.

La ejecución de estas pruebas consistió en tomar los resultados obtenidos de la implementación y compararlos con los resultados teóricos proporcionados por Matlab.

Prueba y resultados obtenidos para una señal senoidal con frecuencia fundamental de 60 Hz

La primer prueba que se realizó se basó en aplicar a la entrada del prototipo una señal senoidal pura, con amplitud de 0.7 y frecuencia fundamental de 60 Hz, y compararla con la señal obtenida a la salida por Matlab para este tipo de estímulo (Fig. 23 y 24).

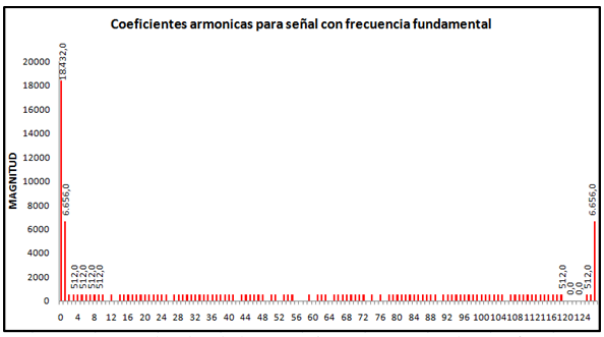

Figura 23. Resultado del prototipo para señal con frecuencia fundamental.

Como se puede observar, tanto en la señal de salida del prototipo como en la señal de salida de Matlab solo existe un pico que pertenece a la frecuencia fundamental.

Para visualizar el porcentaje de las componentes armónicas basta con dividir el valor del coeficiente por sí mismo y multiplicarlo por 100, no se nota diferencia alguna

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2,024	7,168	28%	
Number of 4 input LUTs	1,872	7,168	26%	
Logic Distribution				
Number of occupied Slices	1,509	3,584	42%	
Number of Slices containing only related logic	1,509	1,509	100%	
Number of Slices containing unrelated logic	0	1,509	0%	
Total Number of 4 input LUTs	2,019	7,168	28%	
Number used as logic	1,767			
Number used as a route-thru	147			
Number used as Shift registers	105			
Number of bonded IOBs	44	173	25%	
Number of RAMB16s	7	16	43%	
Number of MULT18X18s	9	16	56%	
Number of BUFGMUXs	2	8	25%	

Figura 20. Recursos utilizados de la FPGA XC3S400 para la configuración 128 puntos 8 bits.

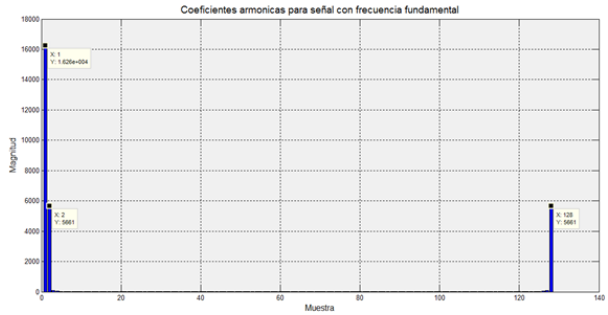

Figura 24. Resultado de Matlab para señal con frecuencia fundamental.

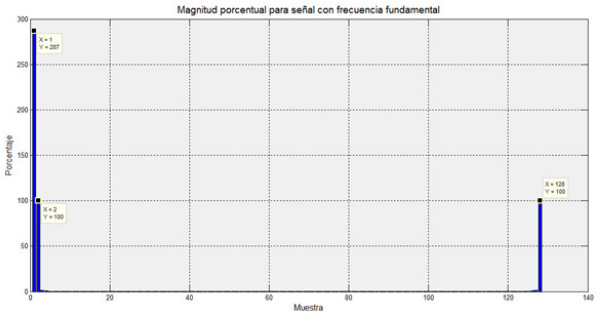

Figura 26. Porcentaje obtenido en Matlab para señal con frecuencia fundamental.

entre Matlab y el prototipo, ya que solo hay una componente (Fig. 25 y 26).

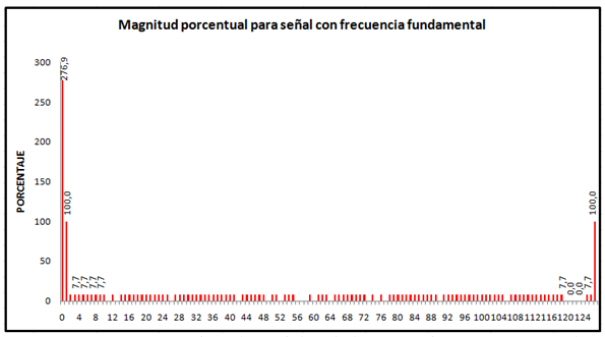

Figura 25. Porcentaje obtenido del prototipo para señal con frecuencia fundamental.

Prueba y resultados obtenidos para una señal senoidal con frecuencia fundamental de 60 Hz y 2º armónico 120 Hz

La segunda prueba que se realizó fue diseñada para que a la entrada del prototipo se tuviese una señal senoidal con amplitud de 0.7 y frecuencia fundamental de 60 Hz, sumada con otra de amplitud 0.2 y frecuencia de 120 Hz, es decir, introduciendo el segundo armónico. La señal obtenida a la salida del prototipo fue comparada con lo proporcionado por Matlab para este tipo de estímulo (Fig. 27 y 28).

Se puede observar que tanto en la señal de salida del prototipo como en la señal de salida de Matlab, además del pico de la frecuencia fundamental aparece otro que pertenece al segundo armónico.

Al visualizar el porcentaje de las componentes armónicas se observa que al igual que en la prueba anterior, el porcentaje de la componente fundamental no presenta diferencia alguna entre el prototipo y lo transmitido por Matlab, pero la componente del segundo armónico muestra un decremento del 4 % (Fig. 29 y 30).

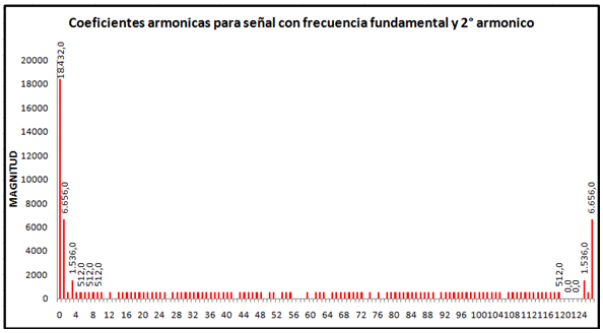

Figura 27. Resultado del prototipo para señal con frecuencia fundamental y segundo armónico.

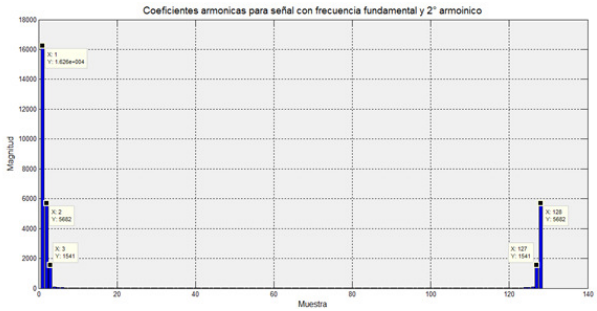

Figura 28. Resultado de Matlab para señal con frecuencia fundamental y segundo armónico.

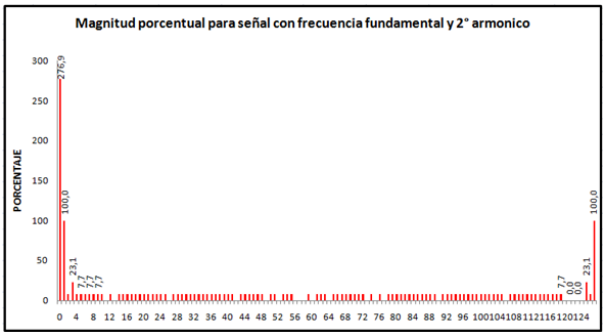

Figura 29. Porcentaje obtenido del prototipo para señal con frecuencia fundamental y segundo armónico.

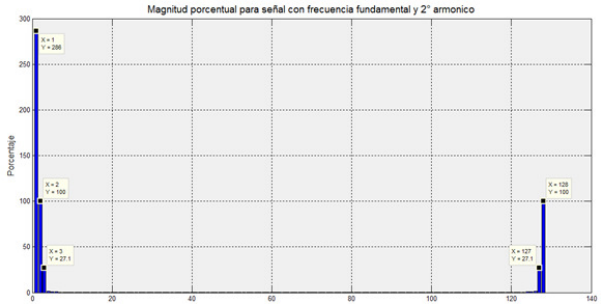

Figura 30. Porcentaje obtenido en Matlab para señal con frecuencia fundamental segundo armónico.

Dicho decremento es producido por la resolución en la cuantificación numérica, el sistema hace el redondeo en cada etapa de la FFT introduciendo pequeños desfases en cada una de estas. Se puede pensar a futuro, y en próximos prototipos mejorar dicha precisión.

Prueba y resultados obtenidos para una señal senoidal con frecuencia fundamental de 60 Hz más 3º, 5º, 7º y 9º armónico

Esta prueba se realizó con una señal de entrada compuesta por las siguientes sub-señales sumadas a la señal senoidal con amplitud de 0.7 y frecuencia fundamental de 60 Hz (Fig. 31 y 32): señal senoidal con amplitud de 0.2 y frecuencia de 180 Hz (tercer armónico), señal senoidal con amplitud de 0.18 y frecuencia de 300 Hz (quinto armónico), señal senoidal con amplitud de 0.15 y frecuencia de 420 Hz (séptimo armónico), señal senoidal con amplitud de 0.10 y frecuencia de 540 Hz (noveno armónico).

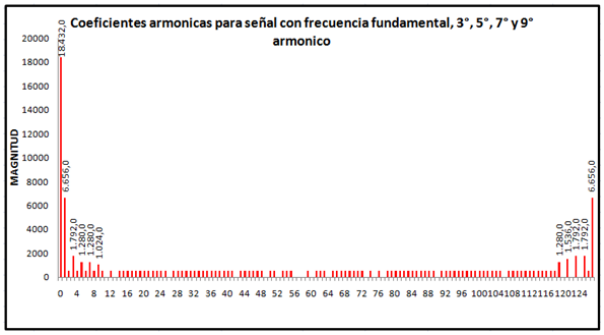

Figura 31. Resultado del prototipo para señal con frecuencia fundamental, 3º, 5º, 7º y 9º armónico.

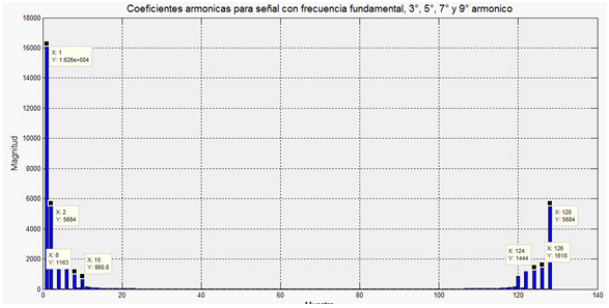

Figura 32. Resultado de Matlab para señal con frecuencia fundamental, 3º, 5º, 7º y 9º armónico.

En las gráficas se puede observar que tanto en la señal de salida del prototipo como en la de Matlab, además del pico de la frecuencia fundamental aparecen cuatro picos más que pertenecen al tercer, quinto, séptimo y noveno armónico respectivamente.

Al comparar el porcentaje de las componentes armónicas en esta prueba se observa que al igual que en las pruebas

anteriores el porcentaje de la componente fundamental no presenta diferencia alguna en el prototipo comparado con lo entregado por Matlab, sin embargo para la componente del tercer armónico muestra un decremento del 1.6 %, para la componente del quinto armónico muestra un decremento del 6.2 %, para la componente del séptimo armónico muestra un decremento del 1.3 % y para la componente del noveno armónico muestra un incremento del 0.3 % (Fig. 33 y 34).

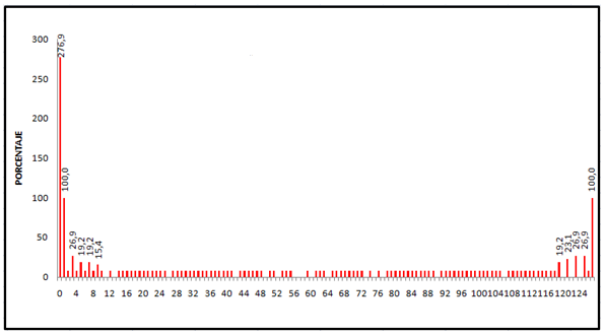

Figura 33. Porcentaje obtenido del prototipo para señal con frecuencia fundamental, 3º, 5º, 7º y 9º armónico.

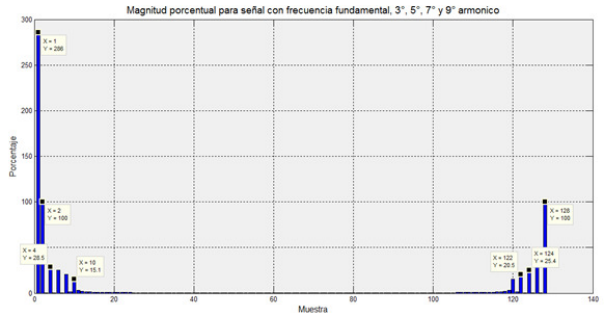

Figura 34. Porcentaje obtenido en Matlab para señal con frecuencia fundamental, 3º, 5º, 7º y 9º armónico.

Conclusiones

El prototipo presenta una alto desempeño, ya que el mayor desfase en cuanto a porcentaje es de apenas el 6.2 % con respecto al comportamiento teórico de la FFT proporcionado por Matlab, lo cual brinda confiabilidad a la hora de visualizar las componentes armónicas que son en gran parte el objetivo principal de este proyecto.

Para mejorar el desempeño del prototipo existe la posibilidad de no hacer el escalonamiento en el bloque FFT, pero la longitud de palabra a la salida aumentaría considerablemente lo cual haría necesario mayor cantidad de puertos en la tarjeta, y por ende mayor utilización de recursos hardware.

Los IP Core son una herramienta muy importante a la hora de implementar cualquier diseño, ya que proporcionan gran flexibilidad en la configuración de los parámetros necesarios reduciendo el tiempo de implementación y optimizando los recursos de la FPGA.

Se optó por disponer del formato en punto fijo, ya que desde la entrada las muestras provenientes del microcontrolador están representadas en este formato y por comodidad se continuó trabajando durante todo el sistema con este tipo de formato, advirtiendo que en cada etapa de procesamiento se introduce un mínimo error provocado por el redondeo convergente que se realiza.

Referencias

Barbu, M., Kaminsky, E. J., y Trahan, R. E. (2005). Fractional Fourier transform for sonar signal processing. En *Oceans, 2005. proceedings of mts/ieee* (p. 1630-1635). doi: 10.1109/OCEANS.2005.1639989

Bian, Y., Zhou, Y., y Li, C. (2008). Some discrete Fourier kind transforms based on flos. En *Ieee international joint conference on neural networks ijcnn 2008 (ieee world congress on computational intelligence)* (p. 78-82). doi: 10.1109/IJCNN.2008.4633770

Calderón, J., y Parra, I. (2010). Software para el diseño e implementación de controladores difusos en microcontroladores. *Revista Tekhnê*, *7*(1), 3-12.

Donnelly, D. (2006). The fast Fourier transform for experimentalists. part vi. chirp of a bat. *Computing in Science & Engineering*, *8*(2), 72-78. doi: 10.1109/MCSE.2006.33

Hori, T. (2008). Reduction in sampling-point numbers for 2-D discrete Fourier transform used in harmonic balance method. , *55*(9), 2664-2672. doi: 10.1109/TCSI.2008.923165

Jutamulia, S., Song, F., y Zhang, Y. (2007). Fourier transforms in optical information processing. En *Conference on lasers and electro-optics - pacific rim, 2007. cleo/pacific rim 2007* (p. 1-2). doi: 10.1109/CLEOPR.2007.4391680

Kauker, Sanftmann, H., Frey, S., y Ertl, T. (2010). Memory saving discrete Fourier transform on GPUs. En *2010 ieee 10th international conference on computer and information technology (cit)* (p. 1152-1157). doi: 10.1109/CIT.2010.209

McGranaghan, M., y Beaulieu, G. (2006). Update on IEC 61000-3-6: Harmonic emission limits for customers connected to MV, hv, and ehv. En *2005/2006 ieee pes transmission and distribution conference and exhibition* (p. 1158-1161). doi: 10.1109/TDC.2006.1668668

Patil, Patil, S. P., y Kyatanavar, D. N. (2010). Single-phase, wide line voltage range, buck power factor corrector with lower harmonic contents in compliance with IEC standard 61000-3-2. En *2010 joint international*

conference on power electronics, drives and energy systems (pedes) & 2010 power india (p. 1-8). doi: 10.1109/PEDES.2010.5712546

Prasad, Ray, K. C., y Dhar, A. S. (2010). FPGA implementation of discrete fractional Fourier transform. En *2010 international conference on signal processing and communications (spcom)* (p. 1-5). doi: 10.1109/SPCOM.2010.5560491

van der Byl, Wilkinson, R. H., y Inggs, M. R. (2011). Recursive Fourier transform hardware. En *2011 ieee radar conference (radar)* (p. 746-750). doi: 10.1109/RADAR.2011.5960637

Vásquez, M., y Martínez, F. H. (2011). Diseño y construcción de control híbrido difuso-deslizante para convertidor DC/DC tipo Boost. *Revista Tekhnê*, *8*(1), 31-40.

Villalba, J., Antelo, E., Bruguera, J. D., y Zapata, E. L. (1998). Radix-4 vectoring cordic algorithm and architectures. *Journal of VLSI signal processing systems for signal, image, and video technology*, *19*(2), 127-147.

Weidong, L., y Wanhammar, L. (1999). Efficient radix-4 and radix-8 butterfly elements. En *Norchip 99* (p. 1-6).

Yanhui Liu, D., Nie, Z., y Liu, Q. H. (2008). Difft: A fast and accurate algorithm for Fourier transform integrals of discontinuous functions. , *18*(11), 716-718. doi: 10.1109/LMWC.2008.2005162