# Implementación de la transformada rápida de Fourier con NIOS II y presentación de la misma en monitor VGA

Cristóbal Andrés Candel Layana [(1)], José Israel Mayorga Bayas [(2)], Ing. Ronald Ponguillo [(3)]
Facultad de Ingeniería de Electricidad y Computación [(1) (2) (3)]
Escuela Superior Politécnica del Litoral (ESPOL) [(1) (2) (3)]
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador
ccandel@espol.edu.ec [(1)], jimayorg@espol.edu.ec [(2)], rponguil@espol.edu.ec [(3)]

## Resumen

*El proyecto consiste en la implementación del algoritmo de la Transformada Rápida de Fourier (FFT – Fast Fourier Transform) utilizando la tarjeta de desarrollo educativo ALTERA DE2, basada en un dispositivo FPGA CYCLONE II de Altera, la misma que consta de memorias embebidas y un PROCESADOR NIOS II SOFTCORE, que ayuda en el procesamiento y presentación de los datos mediante la utilización de un interfaz de monitor VGA.*

*Para una exitosa realización de este proyecto se aplicó tres etapas. La primera etapa está basada en un programa de interfaz gráfica desarrollado en el GUIDE de MATLAB, que permite al usuario generar señales básicas (seno, coseno, etc.) y configurar sus características (amplitud y frecuencia), además de generar las modulaciones analógicas AM (Amplitude Modulation) y FM (Frequency Modulation) y enviar a la tarjeta DE2 de Altera mediante comunicación serial la señal muestreada. La segunda etapa consiste en la recepción de estos datos muestreados en la tarjeta DE2 de Altera y el cálculo de la transformada rápida de Fourier mediante un algoritmo implementado en lenguaje C.. Finalmente la tercera etapa permite mostrar la transformada en un monitor VGA.*

**Palabras Claves:** *FFT, ALTERA DE2, FPGA CYCLONE II, PROCESADOR NIOS II SOFTCORE, VGA, GUIDE de MATLAB*

## Abstract

*This project consists on the implementation of the algorithm for Fast Fourier Transform (FFT - Fast Fourier Transform) using the ALTERA DE2 educational development board, based on FPGA Altera Cyclone II, this consists of embedded memories and NIOS II PROCESSOR SOFTCORE, which aids in the processing and presentation of data through the use of a VGA monitor interface.*

*For a successful implementation of this project it is divided into three stages: The first stage is based on a graphical interface program developed in MATLAB GUIDE, which allows the user to generate basic signals (sine, cosine, etc.), and set its characteristics (amplitude and frequency) and generate analog modulations AM (Amplitude Modulation) and FM (Frequency Modulation) and send to the Altera DE2 board via serial communication the sampled signal. The second stage consists in the reception of these sampled data DE2 Altera card and the calculation of the fast Fourier transform by an algorithm implemented in C language. Finally, the third stage to display the transform on a VGA monitor.*

**Keywords***: FFT, ALTERA DE2, FPGA CYCLONE II, PROCESADOR NIOS II SOFTCORE, VGA, GUIDE de MATLAB*

## 1. Introducción

El procesamiento digital de señales ocupa un papel importante en la elaboración de sistemas modernos de comunicación, para esto se hace uso de la transformada rápida de Fourier aplicada a señales digitalizadas, un claro ejemplo es el de un dispositivo wi-fi, el mismo que hace uso de la modulación digital OFDM mediante la implementación del algoritmo de la transformada de rápida de Fourier. Esta es una de las tantas aplicaciones que tiene el algoritmo de FFT, es por esta razón que en este proyecto se ha planteado la realización de un procesador de FFT en el procesador NIOS II de ALTERA, esta vez se hizo a manera de analizador espectral, es decir, una señal a ser procesada mediante un dispositivo de entrada de datos y un dispositivo de salida de datos que es un monitor VGA.

## 2. Generalidades

Se definen objetivos planteados y los correspondientes alcances y limitaciones del proyecto.

### 2.1. Objetivos

**2.1.1. Objetivos Generales.** El objetivo principal del proyecto es desarrollar un código en NIOS II capaz de calcular la transformada de Fourier de una señal, mediante el algoritmo de la FFT.

**2.1.2. Objetivos Específicos.** Son:

- Uso y manejo de la tarjeta educativa DE2 ALTERA.
- Estudiar la eficiencia e importancia de los distintos algoritmos de la transformada rápida de Fourier.
- Utilizar la interfaz VGA de la tarjeta DE2 para la presentación de los datos obtenidos como resultado del procesamiento de la señal de entrada mediante el algoritmo de la transformada rápida de Fourier en un monitor VGA.
- Analizar la manera más eficiente de adquirir los datos de la señal de entrada al sistema. Estas alternativas pueden ser utilizar una interfaz física con un convertidor analógico – digital (ADC) o el desarrollo de una interfaz gráfica en MATLAB GUIDE que se comunica serialmente con la tarjeta DE2.
- Implementar el proyecto y obtener, a partir de los resultados, las conclusiones y recomendaciones acerca de las pruebas realizadas.

### 2.2. Alcance y Limitaciones del Proyecto

Entre los alcances del proyecto se tiene:

❖ Implementación de un algoritmo para la codificación PCM de los datos generados y muestreados que serán enviados a la tarjeta DE2.

❖ Comunicación serial de los datos de la señal de entrada entre el computador y la tarjeta DE2.

❖ Lectura de los datos de la señal de entrada por medio de un código en lenguaje C, los mismos que llegan codificados en PCM.

❖ Decodificación PCM de los datos recibidos en la tarjeta DE2 antes de su procesamiento.

❖ Procesamiento de los datos mediante el algoritmo FFT Radix–2 Cooley – Tukey.

❖ Los datos obtenidos del procesamiento con el algoritmo FFT son finalmente mostrados en un monitor conectado a la interfaz VGA de la tarjeta DE2 Altera.

Entre las limitaciones se tiene las siguientes:

❖ Debido al elevado costo de un ADC de alta velocidad hemos prescindido de su uso, ya que el objetivo de nuestro proyecto es con fines educativos y no nos limita a utilizar otros métodos alternativos de adquisición de datos.

❖ Al utilizar comunicación serial para adquirir los datos de la señal nos hemos encontrado con limitaciones en tiempo, ya que para obtener una mayor resolución de señal necesitamos enviar mayor cantidad de datos.

❖ La resolución de la señal va ligada a la frecuencia de muestreo de la misma, que en nuestro caso es proporcional a la frecuencia de la señal de entrada. Esto nos obliga a establecer rangos máximos de frecuencia de la onda que se envía.

❖ Debido a las características del controlador VGA de la tarjeta DE2 de Altera, la presentación de la transformada de Fourier de la señal será de baja resolución, es decir, para obtener una buena apreciación de la señal no hemos dibujado la totalidad de los puntos, ya que este controlador tiene una resolución de imagen máxima de 320x240.

## 3. Marco Teórico

### 3.1. Tarjeta de desarrollo Altera DE2

La tarjeta de desarrollo de Altera DE2 mostrada en la figura 1, fue diseña con fines educativos, creada por profesores, dirigida para profesores, investigadores y estudiantes.

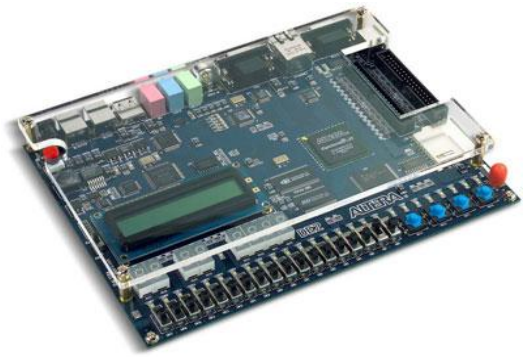


**Figura 1.** Tarjeta DE2 de ALTERA [1]

Adecuada para realizar un amplio número de ejercicios, desde tareas simples para ilustrar conceptos fundamentales de cursos de lógica digital, hasta la implementación de diseños avanzados. Todas estas características convierten a la tarjeta DE2 de Altera en una herramienta indispensable en los laboratorios de las universidades del mundo. [6]

Los periféricos incluidos en la tarjeta DE2 son [7]:

- FPGA Altera Cyclone II 2C35 con 35000 Les.
- Altera Serial Configuration device (EPCS16) para Cyclone II 2C35.
- USB Blaster para programar y usar la tarjeta.
- 8 Mbyte (1M x 4 x 16) SDRAM.
- 1 Mbyte Flash Memory.

- 18 switches DPDT.
- 9 LEDs verdes.
- 18 LEDs rojos.
- Oscilador de 50 MHz para señal de reloj.
- Conector VGA.
- Controlador USB.
- Transceiver RS-232.

### 3.2. FPGA

Una FPGA es un dispositivo semiconductor que contiene bloques lógicos, cuya interconexión y funcionalidad se puede programar.

En la figura 2 muestra una FPGA de la familia Cyclone II, esta es la usada en la tarjeta De2 de Altera.

Los principales beneficios de los FPGAs son:

- Rendimiento.
- Bajo precio.
- Confiabilidad.
- Mantenimiento a largo plazo.


**Figura 2.** Cyclone II FPGA [2]

Hoy las FPGAs están presentes en campos tan diversos como la electrónica de consumo, la investigación científica, entretenimiento, etc.

### 3.3. Procesador embebido NIOS II

El procesador NIOS II de Altera es un procesador soft – core, que a diferencia de los procesadores fijos prefabricados, es descrito mediante un código en HDL y luego cargado sobre una FPGA.

NIOS II es un procesador muy eficiente debido a las bondades que proporciona, su característica principal es que posee un grupo de registros de propósito general de 32 bits.

La figura 3 muestra un sistema elemental con NIOS II, que está compuesto por: el núcleo procesador NIOS II, memoria interna de programa y de datos, periféricos integrados e interfaces para memoria externa y/o entrada/salida.

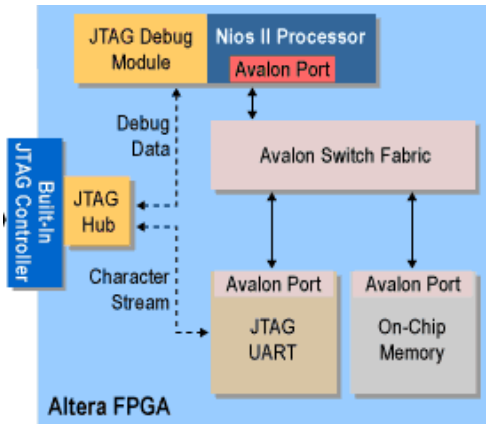



**Figura 3.** Sistema Elemental con NIOS II System [3]

Existen tres versiones del procesador Nios II, las cuales tienen la misma arquitectura de instrucciones de 32 bits:

- NIOS II /f (rápido).
- NIOS II /s (estándar).
- NIOS II /e (económico).

### 3.4. Transformada de Fourier

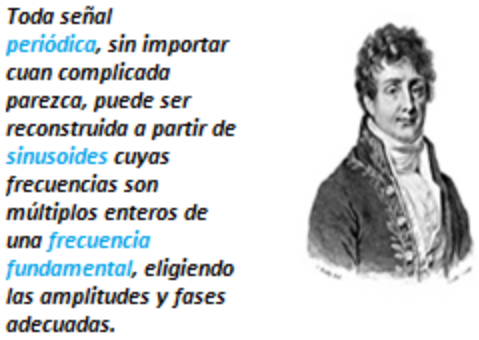



**Figura 4.** Matemático francés Joseph Fourier (1768-1830) [4]

La figura 4 muestra al Matemático francés Joseph Fourier, conocido por sus diversos trabajos en el área de las Matemáticas, de sus trabajos se conoce que la transformada de Fourier.

La transformada de Fourier es una particularización de la transformada de Laplace con $s = j\omega$, siendo $\omega = 2\pi f$, y se define como:

$$x(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t}\, dt \qquad (3.1)$$

Y su inversa se define como:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} x(\omega)e^{j\omega t}\, d\omega \qquad (3.2)$$

**3.4.1. Transformada Discreta de Fourier.** La Transformada Discreta de Fourier (DFT del inglés Discrete Fourier Transform) es el equivalente discreto de la Transformada de Fourier donde se ha transformado la variable continua 't' por la variable discreta *'nTs'* siendo *'Ts'* el periodo de muestreo.

Recordemos el par de ecuaciones de la DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n)W^{nk}\ ;\ k = 0,1,\dots,N-1 \qquad (3.3)$$

Donde las constantes *'W'* son conocidas como factores twiddle y definidas como:

$$W = e^{-j2\pi/N} \qquad (3.4)$$

Observar que *'W'* es una función de longitud *N*, por ello, también suele expresarse como $W_N$.

El inconveniente de realizar unos algoritmos que implementen tal cual estas fórmulas es la cantidad de tiempo requerido para computar la salida. Esto es debido a que los índices *k* y *n* deben variar de *0* a *N-1* para conseguir el rango de salida completo y, por tanto, se deben realizar $N^2$ operaciones.

**3.4.2. Transformada Rápida de Fourier (FFT).** La FFT aprovecha la periodicidad y simetría del factor twiddle *'W'* para el cálculo del Transformada Discreta de Fourier. La periodicidad de *'W'* implica:

$$W^k = W^{k+N} \qquad (3.5)$$

y su simetría implica:

$$W^k = -W^{k+N/2} \qquad (3.6)$$

La FFT descompone la DFT de *N* puntos en transformadas más pequeñas. Una DFT de *N* puntos es descompuesta en dos DFT's de *(N/2)* puntos. Cada DFT de *(N/2)* puntos se descompone a su vez en dos DFT's de *(N/4)* puntos y así sucesivamente. Al final de la descomposición se obtienen *(N/2)* DFT´s de 2 puntos cada una. La transformada más pequeña viene determinada por la base de la FFT.

**3.4.3. Algoritmo Cooley-Tukey (radix-2).** El algoritmo propuesto por Cooley y Tukey en 1965 permite realizar el cálculo de la DFT de una forma más eficiente haciendo uso del enfoque divide y vencerás. En las siguientes subsecciones se presentarán dos versiones del algoritmo radix-2, una implementación específica del algoritmo, orientada al cálculo eficiente de secuencias de datos de longitud $2^p$, donde *P* es un número natural.

**3.4.4. FFT Radix-2 DIF.** El algoritmo radix-2 DIF (Decimation In Frequency) se obtiene al aplicar el enfoque divide y vencerás en la definición de la DFT. De esta forma es posible dividir la sumatoria en dos sumatorias de longitud *N/2* de la siguiente forma:

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n){W_N}^{kn} + \sum_{n=N/2}^{N-1} x(n){W_N}^{kn} \qquad (3.7)$$

donde

$$W_N = e^{j2/N} \qquad (3.8)$$

y *WN* es llamado factor de fase o factor Twiddle como se mencionó anteriormente.

El algoritmo propuesto puede utilizarse de forma recursiva reduciendo a la mitad cada vez el número de puntos de la transformada hasta reducirse a una DFT de 2 puntos. El hecho que se obtengan los valores de la transformada para frecuencias pares e impares de forma separada a partir de la secuencia de datos ordenada conlleva al uso de un algoritmo de ordenamiento de los datos resultantes, es esta característica la que le da su nombre al algoritmo.

En la figura 5 se muestra una representación gráfica del algoritmo para *N = 8*, puede observarse como a partir de una secuencia de datos ordenada se realizan una serie de operaciones mariposa para finalmente obtener los valores de DFT en orden de bit invertido.

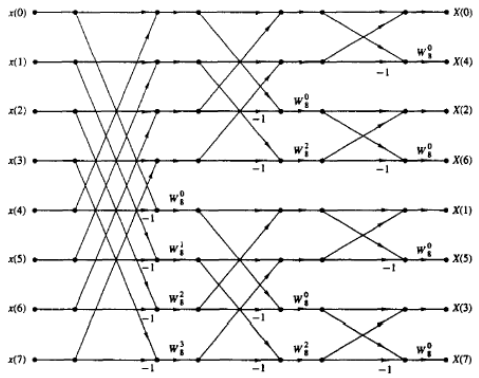



**Figura 5.** Representación gráfica del algoritmo DIF radix-2 [5]

**3.4.5. FFT Radix-2 DIT.** El algoritmo radix-2 DIT (Decimation In Time) es similar conceptualmente al DIF discutido anteriormente, con la diferencia que en este caso se aplica el concepto divide y vencerás a la secuencia de datos de entrada. De esta forma dada una secuencia de datos *x(n)* con *xp(n) = x(2n')* y *xi(n) = x(2n' + 1)* las correspondientes secuencias de elementos pares e impares de *x(n)*, respectivamente, donde *n' = 0, 1,…,N/2 - 1* se tiene que:

$$X(k) = \sum_{n=0}^{(N/2)-1} x_p(n) {W_{N/2}}^{kn} + {W_N}^k \sum_{n=0}^{(N/_2)-1} x_i(n) {W_{N/2}}^{kn} \quad (3.9)$$

En la ecuación 3.9 se puede observar que cada sumatoria corresponde a la transformada discreta de la función par/impar, por lo cual puede reescribirse como:

$$X(k) = X_p(k) + {W_N}^k X_i(k) \; ; k = 0,1, \dots, \frac{N}{2} - 1 \quad (3.10)$$

Como es posible observar en la ecuación 3.10, el cálculo de la FFT DIT tal y como se ha expresado solamente suministra los valores de DFT para *N/2* muestras de datos, sin embargo, considerando que $X_p(k)$ y $X_i(k)$ tienen periodo *N/2* y utilizando la propiedad del factor de fase ${W_N}^{k+N/2} = -{W_N}^k$ se tiene que:

$$X(k + N/2\,) = X_p(k) - {W_N}^k X_i(k) \; ; \; k = 0,1,2, \dots, \frac{N}{2} - 1 \quad (3.11)$$

De esta forma las ecuaciones 3.10 y 3.11 suministran la transformada discreta de todos los *N* valores de entrada.

En la figura 6 se muestra un gráfico del algoritmo aplicado a una secuencia de *N = 8* muestras, como es posible observar, los valores de entrada son reordenados mientras que la salida en frecuencia se obtiene automáticamente en orden, es por esta razón que se le da el nombre DIT al algoritmo.

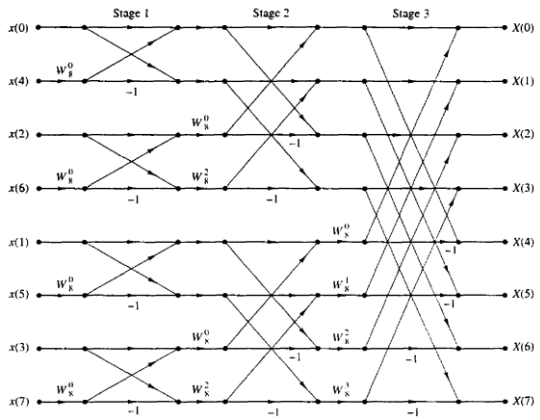



**Figura 6.** Representación gráfica del algoritmo DIT radix-2 [5]

## 4. Diseño e Implementación

En esta sección se explican los detalles del diseño del proyecto mediante la elaboración de un diagrama de bloques que muestra el funcionamiento básico del proyecto.

### 4.1. Diagrama de Bloques

La figura 7 muestra el diagrama de bloques del sistema, el cual consta de tres etapas fundamentales:

**Software Generador de Funciones.-** Esta etapa consiste de un software desarrollado en MATLAB utilizando la interfaz gráfica del mismo.

**Máquina desarrollada en la tarjeta DE2.-** En esta etapa se describen todos los bloques de hardware utilizados en este proyecto.

**Procesamiento y presentación de los datos en un Monitor VGA.-** En esta etapa se hace la presentación de los datos procesados con el algoritmo FFT implementado en el procesador de la tarjeta.

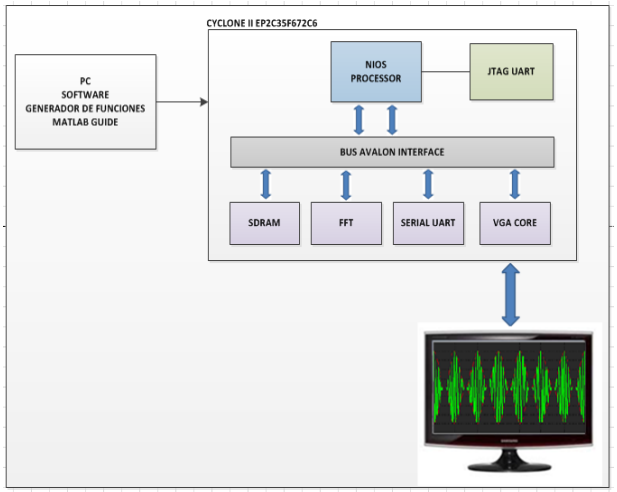



**Figura 7.** Diagrama de bloques del sistema

A continuación se explica cada etapa con mayor detalle.

### 4.2. Software Generador de Funciones

El software generador de funciones utilizado en el proyecto está desarrollado en la interfaz gráfica de usuario de MATLAB.

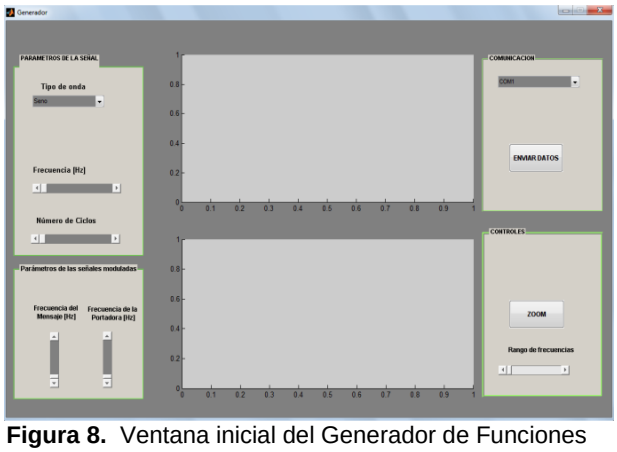



**Figura 8.** Ventana inicial del Generador de Funciones

En la figura 8 observamos la ventana inicial del Software Generador de Funciones, el mismo que consta de cuatro paneles de configuración de parámetros y dos ventanas de gráficos.

El Panel “Parámetros de la Señal” observado en la figura 8 consta de los siguientes elementos:

- Menú Pop – up “Tipo de onda”.- Permite seleccionar el tipo de onda que se desea enviar a través de la interfaz serial hacia la tarjeta DE2.
- Slider “Frecuencia”.- La frecuencia a fijar está en los rangos de 1 Hz hasta 1000 Hz.

En el Panel “Parámetros de las Señales Moduladas”, observado en la figura 8, se puede configurar las características de las señales moduladas en AM y FM, se ha establecido como señal de mensaje una onda coseno debido a que el alcance de este proyecto tiene fines educativos.

- Slider “Frecuencia del Mensaje”.
- Slider “Frecuencia de la Portadora”.

Las frecuencias a fijar en estos dos sliders están en los rangos de 1 Hz hasta 1000 Hz.

Las ventanas de gráficos del generador de funciones muestran la señal en el tiempo y su correspondiente transformada de Fourier, esto se lo hace con el fin de tener una idea de lo que la máquina implementada en la tarjeta DE2 debe realizar.

El Panel “Controles” observado en la figura 8 consta de los siguientes elementos:

- Botón “ZOOM ON”.- Permite realizar un ZOOM en la onda que se seleccione con el puntero del mouse.
- Slider “Rango de Frecuencias”.- Permite configurar la frecuencia inicial y final que se graficaría en el espectro de frecuencias de cualquier señal generada, el valor por defecto de este rango es *(2*f)/2000*. Donde *f* es la frecuencia de la señal en el dominio del tiempo.

El Panel “Comunicación” observado en la figura 8 consta de los siguientes elementos:

- Menú Pop – up “Puerto de Comunicación Serial”.- Permite seleccionar el puerto de comunicación entre el computador y la tarjeta DE 2.
- Botón “ENVIAR DATOS”.- Activa la transmisión serial de la onda generada.

Al presionar el botón “ENVIAR DATOS”, el generador de funciones ejecuta códigos de envío de datos a través del puerto serial hacia la tarjeta. A continuación se muestra parte del código del Generador de Funciones.

En el código mostrado en la figura 9 contiene la información de cómo se ha configurado el puerto serial desde Matlab, es decir, esta es la configuración del Generador de Funciones para la comunicación entre el computador y la Tarjeta DE2. Después de abrir el puerto de comunicación MATLAB genera una codificación PCM de 12 bits de resolución con 1024 muestras que van desde valores de 0 hasta 4095. Estos 12 bits de valores PCM se lo ha dividido en dos paquetes de datos de un byte cada uno, el primer byte enviado corresponde a los 5 bits más significativos de cada muestra generada, mientras que el segundo byte enviado contiene los siguientes 7 bits menos significativos de la muestra. Este procedimiento se ejecuta para las 1024 muestras a ser enviadas.

```
%--- Código para la Comunicación Serial ---%

%--- Parametrización del puerto de Comunicación Serial
    SerDE2 = serial(puerto);
set(SerDE2,'BaudRate',115200);
set(SerDE2,'DataBits',8);
set(SerDE2,'Parity','odd');
set(SerDE2,'StopBits',1);
set(SerDE2,'FlowControl','none');
fopen(SerDE2);
%*-*-*-*-*-*-*
```

**Figura 9.** Código en MATLAB para configurar el Puerto Serial

Para cerrar el puerto que se ha utilizado en el computador, se usa las líneas mostradas en la figura 10, este procedimiento es necesario y se realiza con el fin de evitar problemas con el funcionamiento de los puertos de comunicación usados en futuros envíos de datos.

```
%--- Se Cierra el puerto COM al finalizar ---%
fclose(SerDE2);
delete(SerDE2)
clear SerDE2
disp('STOP')
%*-*-*-*-*-*-*
```

**Figura 10.** Código en Matlab para cerrar el puerto Serial

## 4.3. Maquina desarrollada en la tarjeta DE2

Para implementar sistemas basados en el procesador Nios II hacemos uso del Software de Altera SOPC Builder. Cualquier sistema básico requiere de componentes funcionales tales como memorias, puertos de entrada/salida, interfaces de comunicación, etc.

En la figura 11 se puede observar las diferentes componentes de hardware de nuestro sistema, tomados de la máquina media de University Program de Altera que proporciona soporte completo para introducir a los estudiantes a la tecnología digital. [8]

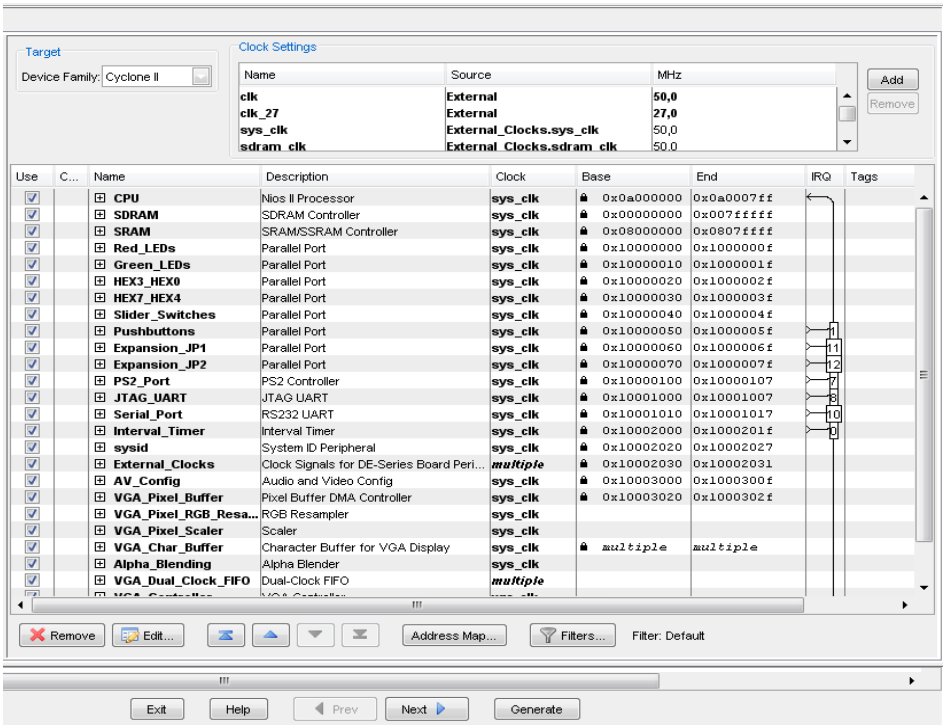



**Figura 11.** Sistema en SOPC Builder

Las partes de nuestro sistema desarrollado en SOPC Builder son:

- **Procesador NIOS II.** Unidad de Procesador Central (CPU).
- **JTAG UART.** Interfaz de comunicación con el computador.
- **RS232 UART.** Puerto para la comunicación serial entre el computador y la tarjeta DE2 de Altera.

Se escogió este protocolo de comunicación debido a que la tarjeta DE2 cuenta con el puerto necesario.

Como se muestra en la figura 12, los parámetros escogidos para el controlador RS232 son los siguientes:

**Baud Rate:** 115200 bps.
**Parity:** "Odd".
**Data Bits:** 8 bits.
**Stop Bits:** 1 bit de parade.

- **VGA Controller.** Controlador que se encarga del manejo de los diferentes parámetros de la interfaz VGA.

VGA (Video Graphics Array) es un estándar de pantallas de video. El controlador VGA genera la sincronización de señales y salidas seriales de datos de pixeles. La implementación en SOPC Builder del controlador se muestra en la figura 13.

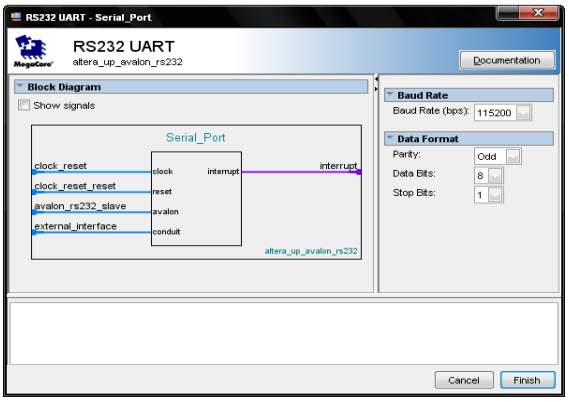



**Figura 12.** Configuración del puerto RS232 en SOPC Builder

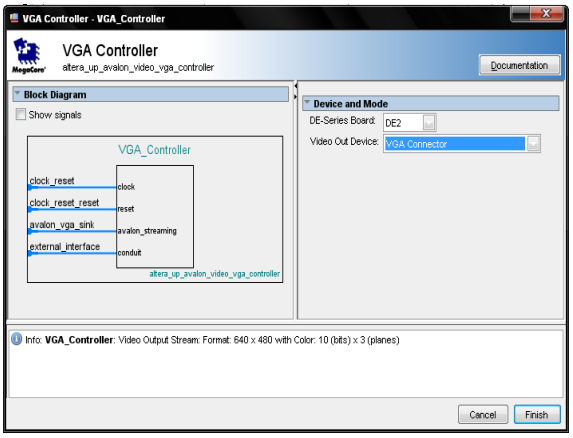



**Figura 13.** Controlador VGA

En la figura 14 vemos que la resolución utilizada es de 320 x 240 debido a que esta es la máxima resolución soportada por el controlador usado en este proyecto.

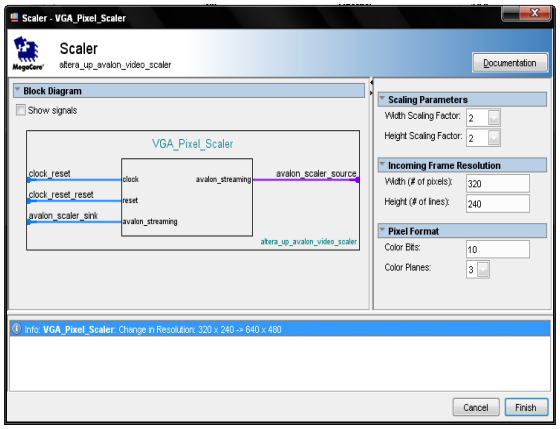



**Figura 14.** VGA Pixel Scaler

El factor escalador utilizado es de 2, para obtener la impresión de resolución de 640 x 480, es decir que se multiplican el ancho y alto de los pixeles con el fin de aparentar una mayor resolución de pantalla.

El color de cada pixel se representa por medio de un número de 16 bits, en donde los 5 primeros (bits del 0 al 4) representan el tono azul, los seis siguientes (bits del 5 al 10) representan el tono verde y los cinco últimos (bits del 11 al 15) representan el tono rojo del pixel.

## 4.4. Procesamiento y presentación de los datos en un Monitor VGA

Esta etapa del proyecto usa una programación en lenguaje C en NIOS II. Siendo las funciones más importantes las descritas a continuación:

**Decodificación de datos.** Una vez que los datos de muestra de señal sean enviados a la tarjeta DE2 está los recibe utilizando el protocolo de comunicación serial RS232, una vez recibidos los datos NIOS II procede a realizar la decodificación de los mismos ya que previo a su envío desde Matlab se los codificó en PCM con 12 bits de resolución.

La figura 15 muestra un fragmento del código desarrollado en lenguaje correspondiente a la función decodificarDatos, se procede a realizar una decodificación de la codificación hecha en MATLAB. Los datos enviados desde la interfaz gráfica se codifican asignando una escala del 0 al 4095 según el valor real del dato. En el código principal simplemente se hace el proceso inverso, dividiendo los datos para 4095, lo cual nos deja datos entre 0 y 1. Luego restamos 0.5 para tener datos entre -0.5 y 0.5 y finalmente multiplicamos por 2 para obtener datos entre -1 y 1.

```
/*************************************************************
 * Decodificar Datos
 *************************************************************/
        void decodificarDatos(double real[], double sig[], int rsig[])
        {

                int n = 1024, i;
                for (i = 0; i < n; i++)
                {
                        real[i] = 2 * ((real[i]/4095) - 0.5);
                        sig[i] = real[i] * -32;
                }
                entero (1024, sig, rsig);
        }
```

**Figura 15.** Función decodificarDatos

**Función FFT2.** La Figura 16 muestra la función *fft2* encargada de calcular la transformada rápida de Fourier. Los parámetros que recibe son el número de datos *n*, el exponente de base dos del número de datos *m*, el arreglo en donde se encuentran los datos muestreados de la señal en el tiempo, que además será el arreglo en el que se almacenará la parte real de la transformada de Fourier *x[ ]* y el arreglo en donde se almacena la parte imaginaria de la transformada de Fourier *y[ ]*. Es importante tener en cuenta que esta función puede calcular la transformada de datos complejos, en este caso tanto la matriz *x* como la matriz y estarán con datos inicialmente, que luego serán reemplazados por la parte real e imaginaria de la transformada. En nuestro caso solo utilizamos datos reales, por lo que es necesario encerar la matriz imaginaria antes de utilizar la función para evitar errores en el cálculo.

Ahora debemos adecuar estos datos para que puedan ser graficados correctamente en el monitor VGA. Para ello multiplicamos cada dato por 32, lo que nos deja con datos entre -32 y 32 en la matriz *sig[ ]*. Estos datos aún son flotantes, por lo que utilizamos la función entero para llenar *rsig[ ]* con el entero de los datos del arreglo *sig[ ]*. El arreglo *rsig[ ]* es el que se utiliza para graficar la señal en el tiempo.

A continuación calculamos el diferencial de frecuencia *df*, este valor es de vital importancia para poder saber la frecuencia de la señal que ha sido enviada.

El arreglo *frec[ ]* es utilizado para almacenar las frecuencias que se muestran en la escala de frecuencia de la gráfica en el monitor VGA.

```
/*****************************************************
 * FFT2
 *****************************************************/
        void fft2 (int n, int m, double x[], double y[])
        {
                long i, i1, j, k, i2, l, l1, l2;
                double c1, c2, tx, ty, t1, t2, u1, u2, z;

                /* Do the bit reversal */
                i2 = n >> 1;
                j = 0;
                for (i = 0; i < n-1; i++)
                {
                        if (i < j)
                        {
                                tx = x[i];
                                ty = y[i];
                                x[i] = x[j];
                                y[i] = y[j];
                                x[j] = tx;
                                y[j] = ty;
                        }
                        k = i2;
                        while (k <= j)
                        {
                                j -= k;
                                k >>= 1;
                        }
                        j += k;
                }

                /* Compute the FFT */
                c1 = -1.0;
                c2 = 0.0;
                l2 = 1;
                for (l = 0; l < m; l++)
                {
                l1 = l2;
                l2 <<= 1;
                u1 = 1.0;
                u2 = 0.0;
                for (j = 0; j < l1; j++)
                {
                        for (i = j; i < n; i+=l2)
                        {
                                i1 = i + l1;
                                t1 = u1 * x[i1] - u2 * y[i1];
                                t2 = u1 * y[i1] + u2 * x[i1];
                                x[i1] = x[i] - t1;
                                y[i1] = y[i] - t2;
                                x[i] += t1;
                                y[i] += t2;
                        }
                        z =  u1 * c1 - u2 * c2;
                        u2 = u1 * c2 + u2 * c1;
                        u1 = z;
                }
                c2 = sqrt((1.0 - c1) / 2.0);
                c2 = -c2;
                c1 = sqrt((1.0 + c1) / 2.0);
        }
}
```

**Figura 16.** Función fft2

## 5. Pruebas y Resultados

**5.1. Envío exitoso de una onda generada y comparación numérica de los resultados mostrados en la pantalla del monitor VGA.** Para probar el funcionamiento del proyecto el Software Generador de Funciones desarrollado en Matlab ha enviado una señal modulada en amplitud "AM" con frecuencia de mensaje 50 Hz y de portadora 500 Hz tal como se muestra en la figura 17.

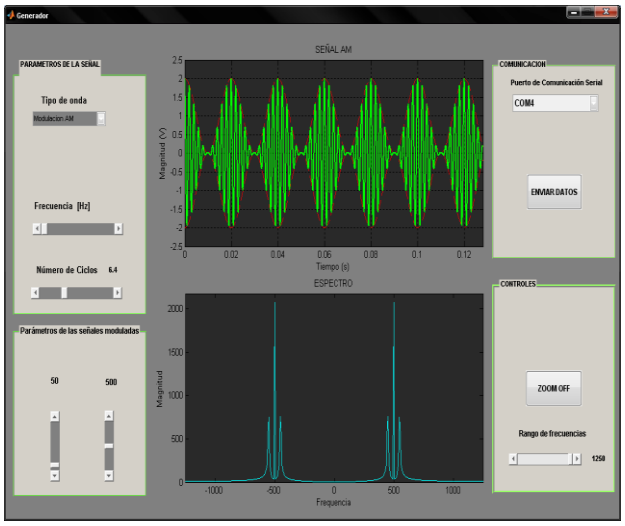



**Figura 17.** Generación de onda AM

La figura 18 muestra que la señal generada tiene un valor de magnitud de 2000 unidades para cuando la frecuencia es 500 Hz y -500Hz, mientras que en las bandas laterales se muestra una amplitud de aproximadamente 700 unidades para cuando la frecuencia es igual a 550Hz, en teoría estos valores de amplitud deberían ser números infinitos o números muy grandes, pero en la práctica el generador no gráfica estos valores teóricos debido a que la onda en el dominio del tiempo esta muestreada a una frecuencia de muestreo de:

$$fs = 64 * fp \qquad (4.1)$$

Donde fs es la frecuencia de muestreo de la onda AM generada y fp es la frecuencia de la portadora de esta señal.

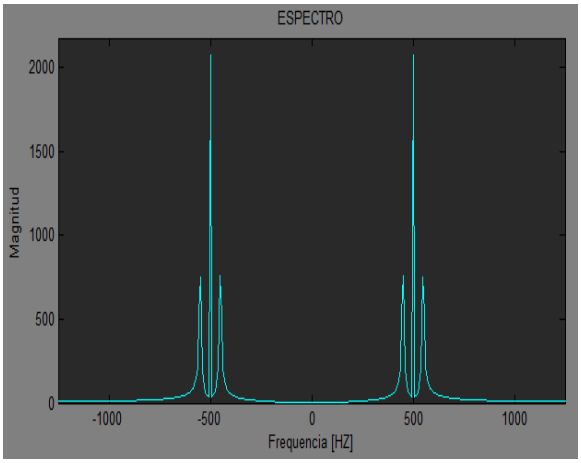



**Figura 18.** Espectro de frecuencias de la señal generada

Si aumentásemos la relación entre fs y fp de 64 veces a un número mayor, entonces obtendríamos una señal AM más pura debido a un aumento en la frecuencia de muestreo, lo que haría que los valores de impulsos mostrados en la figura 18 se acerquen más a los valores teóricos. Hacer esto no es muy práctico, ya que al aumentar la frecuencia de muestreo también aumentaría la cantidad de datos muestreados, es decir, tendríamos más de 1024 datos que transmitir lo que haría lenta la transmisión total de los datos quitándole así la versatilidad a nuestro proyecto en cuanto a lo que a velocidad de transmisión respecta.

El resultado obtenido se observa en la figura 19.

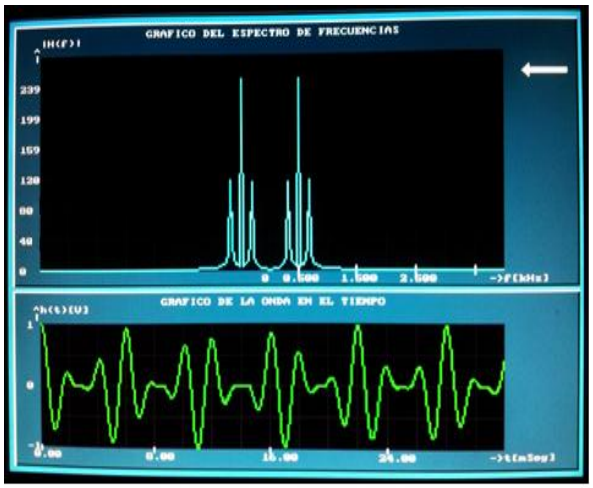



**Figura 19.** Señales graficadas en el monitor VGA

En la figura 19 vemos que la gráfica dibujada de color verde corresponde a una señal AM y la de color azul turquesa es su respectivo espectro de frecuencias que ha sido resultado de haber usado el algoritmo de la Transformada Rápida de Fourier (FFT).

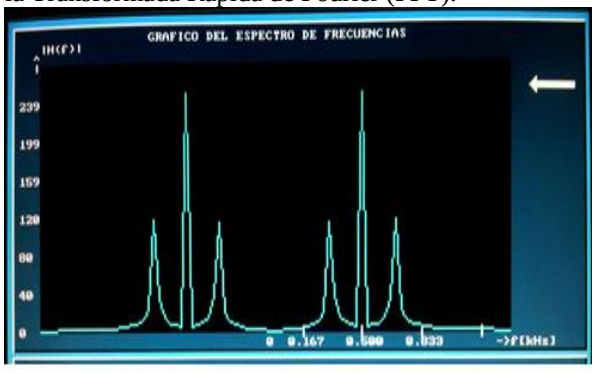



**Figura 20.** Espectro de frecuencias graficado en la Pantalla del monitor VGA

En la figura 20 vemos nuevamente el espectro de la señal, podemos observar que la amplitud para frecuencias de 500Hz y -500Hz son valores de aproximadamente 250 unidades, cuando por lo menos deberían ser de 2000 unidades como se plantea en la figura 18. Esta notoria diferencia es porque el proyecto realiza la transformada Rápida de Fourier de las 1024 muestras recibidas serialmente obteniendo así todos los puntos de frecuencias que teóricamente debería tener, pero con la limitante de que no se dispone de tantos pixeles de resolución VGA para graficar estos 1024 puntos obtenidos al aplicar el algoritmo FFT, procede a hacer una compresión de la señal en el dominio de la frecuencia (eje horizontal del espectro), lo que implica la perdida de ciertos datos, es decir, se pasa de tener 1024 datos a tan solo 256 datos a ser graficados en la pantalla del monitor VGA de una resolución de 320x240 pixeles.

## 5.2. Pruebas de rendimiento del procesador de la Tarjeta DE2 aplicadas a nuestro proyecto.

En esta prueba se toman datos del tiempo que demora la tarjeta DE2 en realizar un fragmento de código de nuestro proyecto. Para esto se hace uso del Interval Timer Core de Altera, configurado como contador descendente de 32 bits, desde SOPC BUILDER, que trabaja con la frecuencia del sistema, con el fin de tener datos de tiempo confiables y de gran precisión.

Se obtendrán resultados de tiempo de procesamiento para las distintas señales del generador de MATLAB, tanto en la decodificación de los datos recibidos como en el cálculo de la FFT.

```
*(TimerTimeoutL)=0xffff;
*(TimerTimeoutH)=0xffff;
*(TimerControl)=4;

decodificarDatos(real, sig, rsig);

*(TimerControl)=8;
*(TimerSnapshotL)=0;
*(TimerSnapshotH)=0;
numclow = 0xffff & *(TimerSnapshotL);
numchigh = 0xffff & *(TimerSnapshotH );
numclks = 0xffffffff & (numclow + (numchigh << 16));
printf("Tiempo de procesamiento de la decodificacion de los datos
recibidos: %f segundos\n", (float)((4294967295 - (float)(numclks)) / 50000000));
```

**Figura 21.** Fragmento de código 1 para pruebas de cálculo de capacidad de procesamiento

En la figura 21 se ha implementado las líneas de código para poner en marcha el conteo del Timer. El código muestra la función decodificarDatos. De esta manera mediremos el tiempo que le toma al procesador realizar la decodificación de los datos recibidos en código PCM.

La siguiente función seleccionada para medir la capacidad del procesador es fft2, la misma que se encarga de calcular la transformada rápida de Fourier de los datos decodificados. La figura 22 muestra el fragmento de código para llevar a cabo esta tarea.

```
*(TimerTimeoutL)=0xffff;
*(TimerTimeoutH)=0xffff;
*(TimerControl)=4;

//funcion para calcular la FFT
fft2 (n, exp, real, imag);

*(TimerControl)=8;
*(TimerSnapshotL)=0;
*(TimerSnapshotH)=0;
numclow = 0xffff & *(TimerSnapshotL);
numchigh = 0xffff & *(TimerSnapshotH );
numclks = 0xffffffff & (numclow + (numchigh << 16));
printf("Tiempo de procesamiento de la FFT de los datos:
%f segundos\n", (float)((4294967295 - (float)(numclks)) / 50000000));
```

**Figura 22.** Fragmento de código 2 para pruebas de cálculo de capacidad de procesamiento

Desde el Generador de Funciones de MATLAB se envían diferentes tipos de señales, para así realizar una comparación de los resultados obtenidos. Como se muestra en la tabla 4-1 en la primera prueba enviamos una señal sinusoidal de frecuencia 100 Hz., para una segunda prueba la señal enviada es una Diente de Sierra con frecuencia de 150 Hz., y como prueba final generamos una señal modulada en frecuencia, con portadora de 1000 Hz y frecuencia de mensaje de 100 Hz.

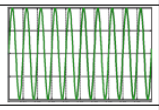
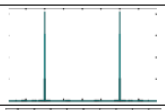
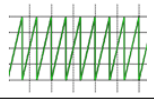
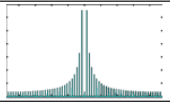
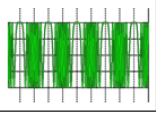
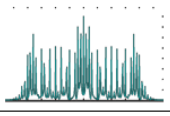
| Señal | Frecuencia (Hz) | Gráfico | Espectro | Tiempo (s) |
|---|---|---|---|---|
| Seno | 100 |  |  | Decod 0.258555<br>FFT2 1.668567 |
| Diente de Sierra | 150 |  |  | Decod 0.258621<br>FFT 1.668787 |
| FM | $f_c$ 1000<br>$f_m$ 100 |  |  | Decod 0.258796<br>FFT 2.309258 |

**Tabla 1.** Resultados del tiempo de procesamiento

Para cada una de las señales transmitidas a la tarjeta los resultados arrojados por el Timer se muestran en la tabla 1 en la columna Tiempo *(s)*, se puede apreciar que el tiempo de procesamiento para una señal sinusoidal de la decodificación de datos recibidos es de 0.258555 segundos y el del cálculo de la FFT es de 1.6668567 segundos. El resultado se debe a la cantidad de datos que se necesitan decodificar y además las transformaciones de tipo de dato necesarias para realizar un correcto cálculo de la transformada de Fourier.

En la tabla 2 se muestran los resultados resumidos de los tiempos de procesamiento obtenidos en las pruebas con la función *decodificarDatos.*

**Tabla 2.** Tiempos procesamiento función decodificarDatos

| Señal | # Líneas de Código | # Lazos | # Iteraciones del lazo | Tiempo (s) |
|---|---|---|---|---|
| Seno | 9 | 2 | 1024 | 0.258555 |
| Diente de sierra | 9 | 2 | 1024 | 0.258621 |
| FM | 9 | 2 | 1024 | 0.258796 |

Podemos observar como para las tres señales los tiempos son muy similares, ya que esta función solo se encarga de decodificar los datos recibidos y dejarlos listos para procesamientos posteriores. Cabe recalcar que para las distintas señales enviadas, esta función emplea la misma cantidad de líneas de código, lazos e iteraciones de lazos.

En la tabla 3 se muestran los resultados resumidos de los tiempos de procesamiento obtenidos en las pruebas con la función *FFT2*.

**Tabla 3.** Tiempos de procesamiento para la función FFT2

| Señal | # Líneas de Código | # Lazos | # Iteraciones del lazo | Tiempo (s) |
|---|---|---|---|---|
| Seno | 36 | 5 | 1024/variable | 1.666.856 |
| Diente de sierra | 36 | 5 | 1024/variable | 1.668.787 |
| FM | 36 | 5 | 1024/variable | 2.309.258 |

Podemos observar como para las dos primeras señales los tiempos son similares, exceptuando el caso de la señal modulada en frecuencia, la cual demora más. Este resultado es de esperarse debido a que el espectro de una señal FM contiene una mayor cantidad de bandas en comparación a los espectros de las señales seno y diente de sierra, lo que provoca que la tarjeta utilice una mayor cantidad de recursos para procesar estos datos. Este cambio en el tiempo también se debe a los lazos de la función que son de duración variable dependiendo de los datos a procesar.

## 6. Conclusiones

- Se pudo desarrollar un código en NIOS II que facilitó el procesamiento y visualización de una señal en un monitor VGA, usando el algoritmo de la Transformada Rápida de Fourier (FFT) para la comparación de resultados en las distintas etapas del proyecto.
- Para poder demostrar el funcionamiento de este proyecto fue necesario desarrollar en MATLAB una aplicación para la generación y envío serial de una señal muestreada hacia la tarjeta DE2, debido

a los costos que implicaría el usar un convertidor Analógico – Digital (ADC) para adquirir físicamente una señal continua de tiempo. El previo uso de esta aplicación fue de mucha ayuda, ya que nos proporcionó los datos que se debería obtener al procesar y graficar en un monitor VGA los datos de la señal generada, lo que facilitó las pruebas y el análisis de los resultados obtenidos.

- El uso de comunicación serial para transmitir los datos generados desde el computador a la tarjeta DE2 supuso un incremento en el tiempo de procesamiento y limitó la cantidad de datos que se utilizaron para calcular la transformada de Fourier, dando como resultado una diferencia entre los espectros obtenidos en MATLAB con los graficados por la tarjeta en el monitor VGA.
- La velocidad de procesamiento del algoritmo para el cálculo de la FFT es relativamente lento, llegando a demorar cerca de un minuto y medio en calcular la transformada de Fourier para señales modulas. Sin embargo, para fines educativos es aceptable.
- El desarrollo de nuestro proyecto en lenguaje C para NIOS II trajo limitaciones principalmente en la velocidad de procesamiento, ya que de haber utilizado más descripción de hardware, estos tiempos se hubiesen reducido considerablemente. Las soluciones en hardware resultan siempre más eficientes que las basadas en software.

## 7. Recomendaciones

- Se recomienda configurar de manera correcta los diferentes parámetros de señal en el software generador de funciones desarrollado en MATLAB, sobre todo el parámetro “Puerto de Comunicación Serial” ubicado en el panel “COMUNICACIÓN” de nuestro generador de funciones.
- Como futura mejora, se recomienda el uso de un convertidor analógico – digital, el cual proporcionaría una adquisición física y en tiempo real de una onda adquirida de un generador de funciones o un acoplador para una antena que receptaría una señal del ambiente. Lo que implicaría también un incremento en la velocidad con la que la tarjeta DE2 adquiere los datos a procesar.
- El uso de un puerto paralelo mejoraría la velocidad de la transmisión de datos en lugar de comunicación serial, ya que esta envía una mayor cantidad de datos por unidad de tiempo.
- Para una mayor velocidad de procesamiento de los datos, se recomienda usar un Core desarrollado en lenguaje de descripción de hardware, ya que este tipo de soluciones proporcionan mejores tiempos de respuesta comparados a los tiempos obtenidos en funciones netamente de software, desarrolladas para ejecutarse en el procesador NIOS II.
- La resolución del monitor VGA se puede mejorar si se utiliza un Core más poderoso, que configure cantidad de pixeles mayores a las que se usan en nuestro proyecto, ya que la tarjeta DE2 soporta una resolución 1600x1200 mientras que con el Core VGA usado solo se llega a 640 x 320.
- El uso de un Core PLL proporcionaría una mayor velocidad de procesamiento al aumentar la frecuencia del reloj del sistema, teniendo en cuenta que como valor máximo se tiene 200 MHz para esta tarjeta específica.

## 8. Referencias


[1] Loomis John, Altera's Development and Education (DE2) Board, http://www.johnloomis.org/altera/

[2] ALTERA, Cyclone II FPGA, http://www.digchip.com/datasheets/parts/datasheet/033/EP2C35F672C6.php

[3] ALTERA, JTAG Debug Module, http://www.altera.com/devices/processor/nios2/benefits/ni2-jtag-debug.html,

[4] The University of Adelaide, Jean Baptiste Joseph Fourier, http://ebooks.adelaide.edu.au/f/fourier/joseph/index.html

[5] CMLAB, Fast Fourier Transform (FFT), http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html

[6] ALTERA, DE2 Development and Education Board, http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html

[7] ALTERA, DE2 Development and Education Board – Getting started Guide, http://users.ece.gatech.edu/~hamblen/DE2/DE2%20Reference%20Manual.pdf

[8] MathWorks, Matlab – The language of Technical Computing, http://www.mathworks.com/products/matlab