PROCESAMIENTO DIGITAL DE SEÑALES DE AUDIO SOBRE ZEDBOARD.

Lorena Riaño Molina

Ingeniería Electrónica

Director(es) del trabajo de grado: Luis Fernando Castaño Londoño

INSTITUTO TECNOLÓGICO METROPOLITANO

24 octubre 2016

RESUMEN

En esta sección se incluye el objeto del trabajo, la metodología utilizada para su desarrollo y los principales resultados y conclusiones encontradas. No debe incluir referencias y deberá tener una longitud máxima de 300 palabras.

Palabras clave:Palabras técnicas significativas que identifiquen el trabajo desarrollado.

En este trabajo de grado se presenta una aplicación de procesamiento de audio sobre FPGA. La implementación del proyecto se realiza en un sistema de desarrollo ZedBoard. Se emplea un diagrama de bloques para la descripción del sistema en el entorno *Vivado Design Suite* de *Xilinx.* La señal de audio se ingresa a través de la línea de entrada de la ZedBoard. La conversión de la señal es realizada por un CODEC de audio que se comunica con la FPGA a través de un bus I2C. El control del CODEC es ejecutado a través de un programa en lenguaje C desarrollado sobre el *Software Development Kit* (SDK) de *Xilinx*. Esta aplicación es ejecutada por un sistema procesamiento ZYNQ-7000 en uno de los núcleos ARM Cortex-A9 de la SoC FPGA. Sobre la señal digitalizada se aplica la transformada rápida de Fourier (FFT) desarrollado en lenguaje C. Los resultados obtenidos son comparados con la simulación realizada en MATLAB.

Palabras clave: FFT, FPGA, VHDL, Procesamiento de audio, Almacenamiento de datos.

RECONOCIMIENTOS

En primera instancia quiero agradecer a Dios y a la mujer que me dio la vida, mi madre que es el pilar de mi vida y mi ejemplo a seguir. Igualmente quiero agradecer a la Institución Universitaria ITM ya que gracias a sus docentes, instalaciones y demás recursos que me brindaron, logré trascender personal y profesionalmente. Agradezco al laboratorio de Sistemas de Control y Robótica por la posibilidad de desarrollar este trabajo de grado bajo la modalidad de Producto en Laboratorio de Investigación. Particularmente la asesoría del docente Luis Fernando Castaño por guiarme en todos los complejos procesos que se presentaron al largo de este trabajo y al docente David Márquez por aportar una visión crítica y práctica. No quiero olvidar a mis compañeros de carrera, gracias a todos ellos por acompañarme en esta experiencia que es estudiar una ingeniería. Finalmente agradezco a mis compañeros de trabajo, familiares y amigos por su apoyo en este proceso de aprendizaje y que contribuyeron para que yo lograra culminar mis estudios de pregrado.

ACRÓNIMOS

SDK Software Development Kit

NCO Numeric Controller Oscillator

DSP Digital Signal Processor

DAC Digital Analog Converter

ADC Analog Digital Converter

DAQ Data Acquisition

FPGA Field Programmable Gate Array

VHSIC Very High Speed Integrated Circuit

VHDL VHSIC Hardware Description Language

HDL Hardware Description Language

FFT Fast Fourier Transform

EDA Electronic Design Automation

TABLA DE CONTENIDO

TABLA DE CONTENIDO5

1. INTRODUCCIÓN6
2. MARCO TEÓRICO8
3. METODOLOGÍA18

3.1 ADQUISICIÓN DE AUDIO A TRAVÉS DE LA TARJETA ZEDBOARD19

3.2 ALMACENAMIENTO DE AUDIO EN FORMA DE TEXTO A TRAVÉS DE LA ZEDBOARD67

3.3PROCESAMIENTO DE LA TRANSFORMADA RAPIDA DE FOURIER.74

3.4 PROCESAMIENTO DE AUDIO A TRAVÉS DE MATLAB81

4. RESULTADOS Y DISCUSIÓN90
5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO94

96

97

1.INTRODUCCIÓN

El objetivo de este proyecto es desarrollar una aplicación que permita la adquisición de un audio, almacenamiento de datos en SD y procesamiento de señales. La implementación se realiza empleando el sistema de desarrollo ZedBoard, el cual contiene una FPGA de la familia Zynq7000 de Xilinx. Se realiza la adquisición de señales por medio de la interfaz de audio teniendo como base la guía "ADC/DAC and Digital Audio Processing" suministrada por la página https://embeddedcentric.com/. Para leer y almacenar los datos de audio adquiridos en la tarjeta SD de la ZedBoard también se toma como referencia esta página. Se emplea un algoritmo en lenguaje C para obtener la FFT de la señal adquirida. Este resultado obtenido es almacenado en una memoria SD. Estos valores son empleados para graficar el diagrama de magnitud de la señal en el dominio de la frecuencia en MATLAB. Se realiza la comparación entre los resultados obtenidos con la FPGA y MATLAB.
Las aportaciones de Cooley y Tukey (1965) de un algoritmo eficiente para el cálculo de las transformadas de Fourier aceleró el uso del computador digital. Muchas aplicaciones desarrolladas requerían del análisis espectral de la señal y con las nuevas transformadas rápidas se redujo en varios órdenes de magnitud de tiempo de cómputo.

En este trabajo de grado realizado en el laboratorio de Sistemas de Control y Robótica bajo la modalidad de producto en laboratorio de investigación, se presenta la implementación de un algoritmo para obtener la transformada rápida de Fourier (FFT) de señales de audio sobre FPGA.

El desarrollo de este trabajo sirve como insumo en el laboratorio de Sistemas de Control y Robótica, para el desarrollo de trabajos de grado y proyectos de investigación que requieran el uso de sistemas basados en FPGA para la adquisición y procesamiento digital

de señales. Igualmente permite divulgar a los estudiantes del Departamento de Electrónica y Telecomunicaciones del ITM, el uso de FPGA en el procesamiento digital de señales audio. Este beneficia particularmente a los estudiantes de la asignatura Diseño Digital y de Proyecto de Grado que deseen trabajar en esta área. El trabajo es desarrollado empleando los recursos disponibles en el laboratorio de Sistemas de Control y Robótica y de Microelectrónica y Nanotecnología del ITM.

En el capítulo 2 se presenta el marco teórico abordando los conceptos básicos para el desarrollo del presente trabajo de grado. En el capítulo 3 se presenta la metodología utilizada para llevar a cabo la adquisición, almacenamiento y procesamiento de señales de audio en la ZedBoard. En el capítulo 4 se presentan los resultados obtenidos en la ejecución de este trabajo. El capítulo 5 muestra las conclusiones y recomendaciones. Por último, se citan las referencias y apéndices que dan soporte a este trabajo.

2.MARCO TEÓRICO

En este proyecto de grado se hace uso de la herramienta de *Electronic Design Automation (EDA)*, las cuales buscan crear soluciones, a las altas exigencias de diseño electrónico en sistemas embebidos, especialmente cuando se enfoca en el paradigma de co-diseño de hardware y software. Con este proyecto se introduce al uso de la tecnología *Zynq-7000* de *Xilinx*. Esta familia de dispositivos incorpora un microprocesador ARM de 32 bits y un FPGA para el diseño de sistemas embebidos.

ADQUISICIÓN DE DATOS (DAQ):

La Adquisición de Datos es el proceso utilizado para recopilar y documentar información y posteriormente analizar un fenómeno o como en este caso el sonido. Un sistema DAQ está compuesto de cinco partes:

1. Sensores.
2. Acondicionamiento de Señales.
3. Convertidores ADC/DAC.
4. Controlador de Software y la aplicación.

Figura 1. Partes de un sistema DAQ

Sensores:

Son los dispositivos encargados de medir la variable física y convertir a una señal eléctrica medible que pueda ser interpretada por un dispositivo electrónico. Para la medición de las

variables físicas se establece un comportamiento del sistema que determina el funcionamiento del sistema. Las señales que transmiten los sensores pueden ser digitales o analógicas. En este trabajo de grado la señal que representa el sonido es eléctrica y analógica. Esta señal proviene de cualquier dispositivo que reproduzca audio.

Convertidores ADC y DAC:

Los convertidores ADC sirven para transformar la señal eléctrica analógica en un formato que pueda ser procesado por un dispositivo electrónico programable, señal digital. Por el contrario, un convertidor DAC sirve para realizar el proceso inverso al ADC, es decir, parte de una señal digital y la convierte a analógica. Una señal analógica es continua en el tiempo, puede tomar cualquier valor y tiene infinitos valores, mientras que una señal digital discreta no es continua en el tiempo y tiene valores finitos. El proceso de digitalización de una señal analógica consiste en la toma de una muestra de la señal cada determinado tiempo, motivo por el cual en el proceso de digitalización se pierde información de la señal. Para una correcta digitalización se debe tener en cuenta la resolución, rango y tasa de muestreo y así garantizar una correcta representación digital de la señal analógica.

1. Resolución:

Es el número de niveles binarios discretos que se utilizan para representar la señal analógica como lo muestra la Figura 2.

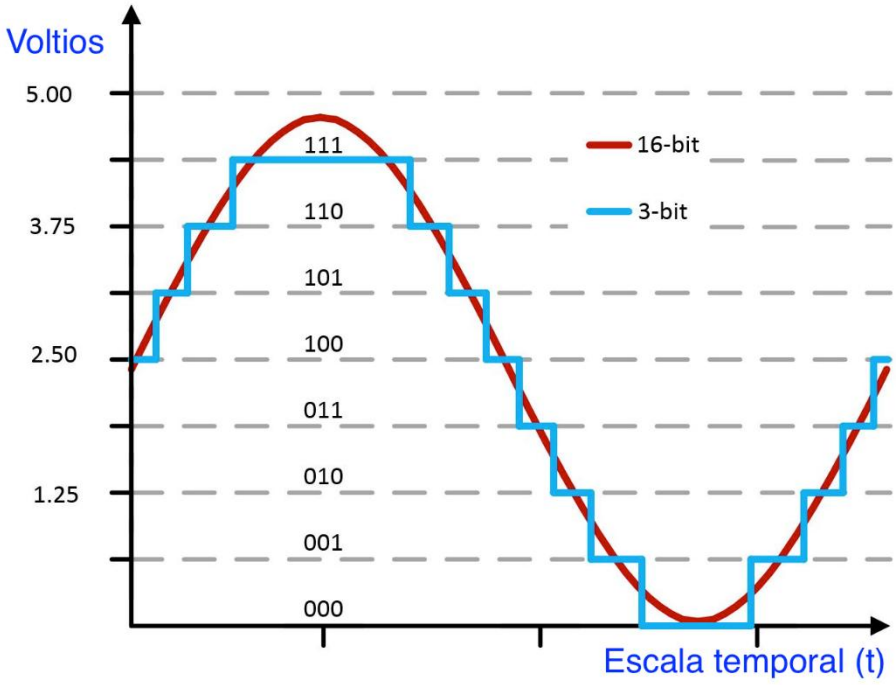

Figura 2.Niveles binarios para resolución de 3 Bits y 16 Bits.

Tomado de: https://soloarduino.blogspot.com.co/2015_09_01_archive.html.

A mayor resolución binaria, es mejor la representación de la señal original. Teniendo como ejemplo una señal sinusoidal analógica que se quiere digitalizar, en la Figura 2 se observa que para una digitalización de resolución de 3 bit existen 8 niveles de representación de voltaje, lo que genera una señal escalonada y para una digitalización de resolución de 16 bit existen 65536 niveles lo que permite una representación más fiel de la señal original.

2. **Rango:**

Son los niveles de voltaje que puede representar una señal digital. Este parámetro está dado por el fabricante de la tarjeta de adquisición de datos.

3. Tasa de muestreo:

Es la frecuencia con la que se toma una muestra de la señal original. Este parámetro es el más importante para una correcta representación digital de la señal original. La frecuencia de muestreo depende la calidad de la señal digitalizada, a mayor frecuencia de muestreo la señal digital es más parecida a la real. Sin embargo, una señal digital discreta nunca es igual a la señal analógica que intenta representar, esto se debe a que en el proceso de digitalización se pierde información, pero con una correcta frecuencia de muestreo, como lo describe el teorema de Nyquist. Lo que se desea es una representación digitalcon precisión de la forma de la señal original.

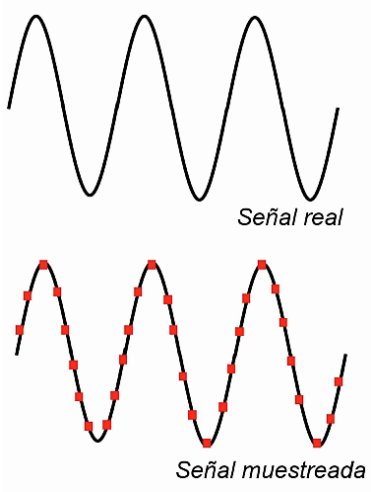

Figura 3.Proceso de muestreo de una señal analógica.

Tomada de: NI.com, 2016

Según el teorema de Nyquist, para replicar una señal analógica, la frecuencia de muestreo debe ser por lo menos igual o superior a dos veces la máxima frecuencia de la señal a muestrear. En la Figura 4 se observa la reconstrucción de una señal analógica a diferentes frecuencias de muestreo. A pesar de que muestrear a dos veces la frecuencia máxima de la señal analógica es suficiente para replicar la señal, se observa que una frecuencia de

muestreo de cinco o más veces la máxima frecuencia, la señal que se obtiene es similar a la original.

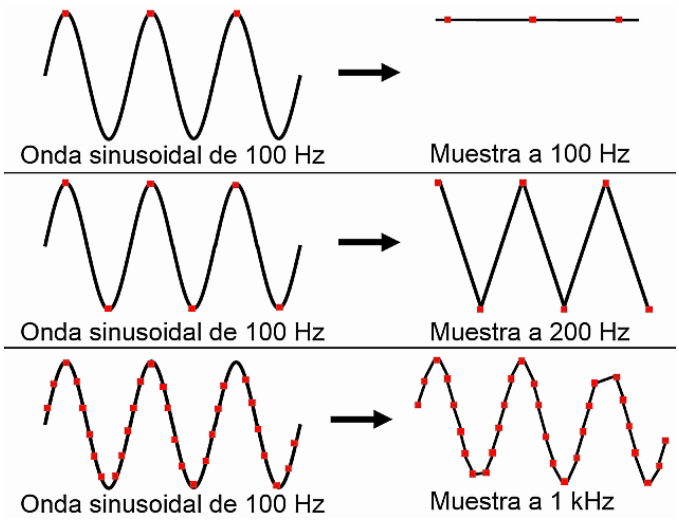

Figura 4.Señal sinusoidal de 100Hz muestreada a una frecuencia de 100Hz, 200Hz y 1KHz. Tomada de: NI.com, 2016.

Procesador de señales:

Son los dispositivos lógicos programables que permiten modificar y procesar los diferentes tipos de señales muestreadas. En este caso se empleará la tarjeta ZedBoard con el procesador Zynq-7000 de Xilinx. Los sistemas de procesamiento de datos poseen periféricos de entrada y salida que facilitan la interpretación y exportación de los datos.

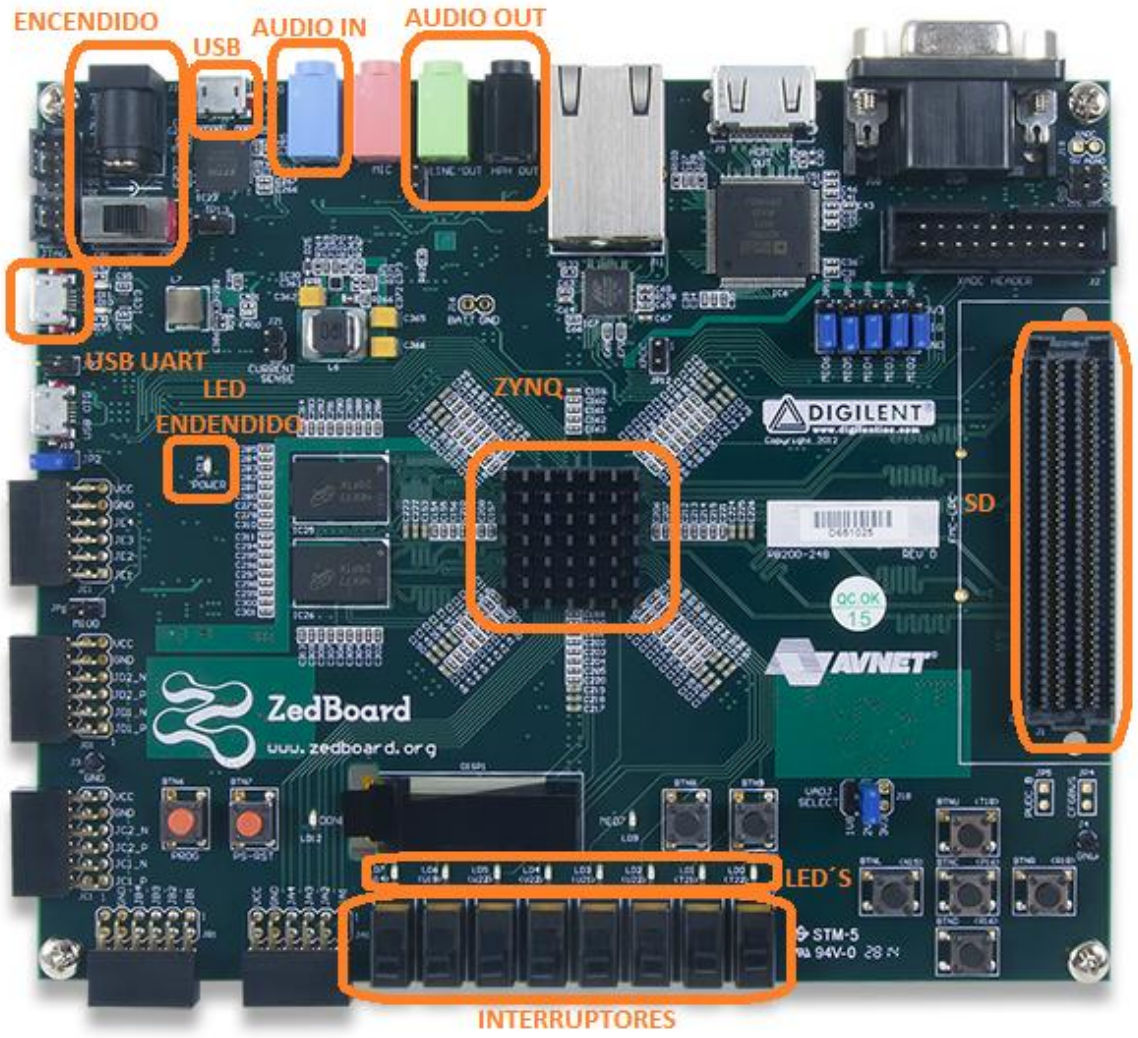

Figura 5. Tarjeta ZedBoard.

PROCESAMIENTO DIGITAL DE SEÑALES (DSP)

Consiste en la manipulación matemática de una señal en el dominio del tiempo discreto para modificarla o mejorarla en algún aspecto. El procesamiento de las señales se hace mediante un sistema basado en un procesador que posee un juego de instrucciones, un hardware y un software optimizado para aplicaciones que requieran operaciones numéricas a muy alta velocidad tal como las características de la ZedBoard.

Se puede trabajar con señales analógicas, pero es un sistema digital, por lo tanto, necesitará un conversor analógico/digital a su entrada y digital/analógico en la salida. Como todo sistema basado en procesador programable necesita una memoria donde almacenar los datos con los que trabaja y el programa que ejecuta. En la Figura 6se observa el diagrama de un sistema de procesamiento digital de señales.

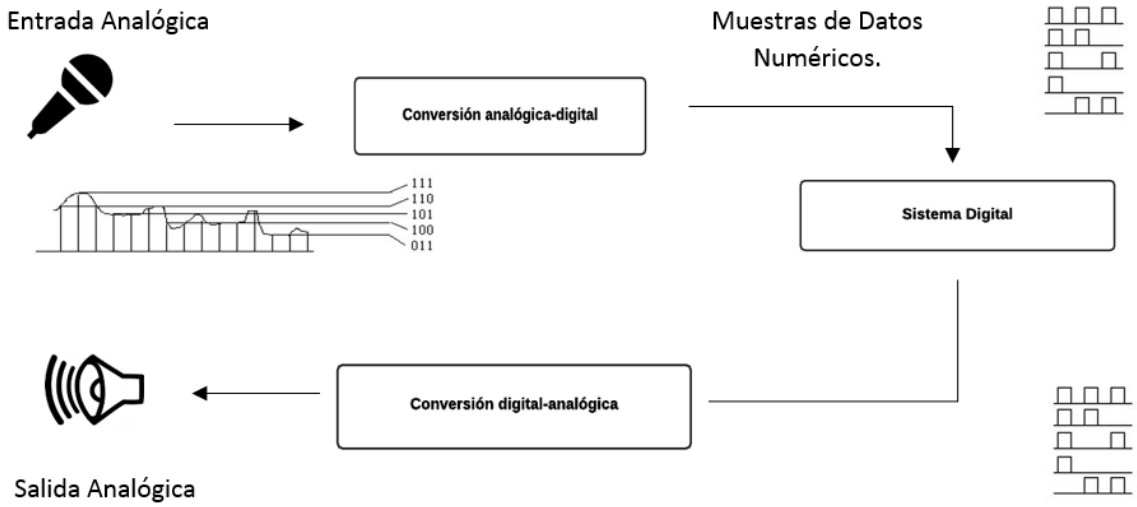

Figura 6. Procesamiento Digital de Señales.

Ventajas del procesamiento digital de señales:

4. El sistema se puede programar y modificar fácilmente dando un alto grado de flexibilidad en el diseño.
5. Se puede atenuar el ruido, ya que la señal al estar digitalizada no sufre alteraciones por otros componentes o etapas del sistema.

6. Un sistema digital funciona igual en toda su vida útil. A diferencia de los sistemas analógicos que los componentes activos y pasivos varían sus tolerancias con el paso del tiempo logrando así cambios en el comportamiento del diseño.
7. Se puede replicar los diseños fácilmente porque la fabricación de dispositivos es similar y se puede transferir de uno a otro los algoritmos de programación.
8. Procesamiento avanzado y reprogramable.
9. Baja sensibilidad a condiciones ambientales, tolerancia y estado de los componentes.

Desventajas del procesamiento digital de señales:

10. Perdida de información por muestreo.
11. Error de redondeo por cuantificación.
12. Velocidad de adquisición de datos y procesamiento.
13. Para señales analógicas de muy alta frecuencia se requieren conversores ADC de muy alta tasa de muestro ya que se requiere mínimo el doble de frecuencia de la señal original.
14. En algunos casos requiere un procesador de muy alta velocidad que pueda realizar el tratamiento de la señal en muy poco tiempo.
15. El diseño es más complejo ya que se requiere de hardware y software para su implementación.

La transformada de Fourier es una de las principales herramientas utilizadas en el procesamiento de señales, ya que es un algoritmo que permite a un procesador digital hacer el cálculo de la transformada discreta de Fourier de forma práctica, haciendo una mejora en la carga computacional y el tiempo de procesamiento. Dado que con este

cálculo se puede realizar el proceso inverso, se concluye que la FFT tiene la flexibilidad de pasar de un dominio cualquiera a otro.

La serie de Fourier es una función periódica con periodo T_0 puede ser expresada como una serie de Fourier, como lo muestra la siguiente formula:

$$y(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n * cos(2\pi n f_0 t) + b_n sin(2\pi n f_o t)] \, (1)$$

Siendo $f_0 = \frac{1}{T_0}$ la frecuencia fundamental.

Luego de definir los coeficientes, se obtienen las variables complejas para desarrollar las identidades necesarias para reemplazar los valores y factorizar. Se agregan valores negativos de n en las ecuaciones y luego se reemplaza en la siguiente ecuación.

$$y(t) = \frac{a_0}{2} + \frac{1}{2}\sum_{n=1}^{\infty}(a_n - jb_n)e^{j2\pi n f_0 t} + \frac{1}{2}\sum_{n=1}^{\infty}(a_n + jb_n)e^{-j2\pi n f_0 t} \, (2)$$

Continuando con el reemplazo de los valores se obtiene la serie Fourier en forma exponencial, donde:

$$a_n = \frac{1}{2}(a_n - jb_n) yn = \pm 1, \pm 2, \pm 3, \ldots$$

$$se\ obtiene a_n = \frac{1}{T_O}\int_{-T_O/2}^{T_O/2} y(t)e^{-j2\pi n f_0 t} dt\ donde n = \pm 1, \pm 2, \pm 3, \ldots \, (3)$$

Correspondiente a la serie de Fourier en forma de coeficientes complejos; esta forma es la que más se usa en análisis. El algoritmo básico para la FFT es desarrollado por Cooley-Tukey en 1965. El cálculo de la transformada discreta de Fourier reduce el número de

sumas y multiplicaciones respecto al algoritmo original. La transformada discreta de Fourier (DFT) es una de las técnicas más utilizadas para convertir señales del dominio del tiempo al dominio de la frecuencia. La transformada discreta de Fourier puede ser representada mediante la ecuación 4 (Salomon Bochner, Komaravolu Chandrasekharan).

$$X(k) = \sum_{n=0}^{N-1} X(n) W_N^{kn} (4)$$

En donde $X(k)$son las muestras de la señal en el dominio del tiempo, $X(n)$ son las muestras obtenidas en el dominio de la frecuencia, N es el número de puntos y $W_N$son los factores de giro. Existen básicamente dos tipos de algoritmos para la FFT, el de diezmado en tiempo y diezmado en frecuencia. Básicamente el algoritmo FFT toma el de la DFT y lo separa en dos partes, uno con índices pares y otro con impares. Como se ve en la ecuación 5

$$X(k) = \sum_{n=0}^{(N/2)-1} X_1(m) W_{N/2}^{mk} + W_N^k \sum_{n=0}^{(N/2)-1} X_2(m) W_{N/2}^{mk} (5)$$

3.METODOLOGÍA

En este trabajo la adquisición y procesamiento de la señal de audio es realizado empleando un sistema de desarrollo ZedBoard. La fuente de la señal de audio es un dispositivo móvil o PC. El audio ingresa por la línea de entrada de la ZedBoard, que corresponde al Jack de color azul. La digitalización de la señal se hace por medio del chip*ADAU1761*, que es un circuito integrado que posee dos ADC de 24 bits cada uno y dos conversores DAC de 24 bits. Este chip integrado tiene la capacidad de trabajar a una frecuencia de muestreo desde 8KHz hasta 96KHz. El procesamiento de la señal se hace por medio del núcleo ARM Cortex-A9. La interface de tarjeta SD se encarga de almacenar los datos procesados por el núcleo. La memoria SD es una memoria externa no volátil que se usa para almacenamiento de datos. Adicionalmente, si los datos se quieren interpretar en otro software y por medio de un computador, se hace uso de los datos almacenados en la memoria SD.

El entorno de desarrollo usado para realizar la aplicación es el Vivado Design Suite 2015.3. Este software es necesario para configurar los conversores ADC y DAC, el procesador, la tarjeta SD y los drivers necesarios para que la tarjeta ZedBoard funcione correctamente. Por último, se comparan los resultados con el entorno de trabajo MATLAB.

Para alcanzar el objetivo principal de este trabajo de grado se deben seguir los lineamientos en el orden descrito a continuación:

16. Adquirir el audio por medio de la tarjeta ZedBoard.
17. Almacenar el audio en formato de texto en la tarjeta SD de la ZedBoard.
18. Cargar el texto de audio desde la tarjeta SD, ejecutar el código de FFT y almacenar en forma de texto la FFT de la señal de audio en la tarjeta SD a través de la ZedBoard.
19. Graficar y comparar los resultados obtenidos por medio de la tarjeta ZedBoard y los obtenidos por MATLAB.

3.1 ADQUISICIÓN DE AUDIO A TRAVÉS DE LA TARJETA ZEDBOARD

La adquisición del audio se realiza basándose en el ejemplo de la página llamado "ADC/DAC and Digital Audio Processing".

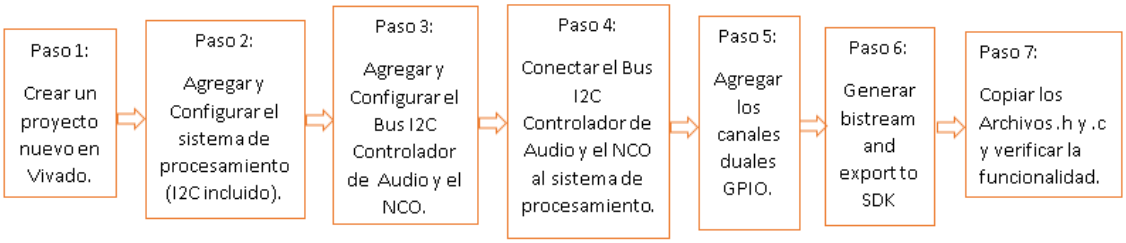

Figura 7. Diagrama de Bloques del Paso a Paso.

20. Descargar los dos archivos de la página GitHub como se observa en la Figura 8*zed_audio_ctrl*: IP del controlador responsable del intercambio de datos con el ADAU1761 Codec a través del protocolo I2S.
21. *xilinx_com_hls_nco_1_0:* Controlador IP responsable de generar ondas sinusoidales a una frecuencia deseada.

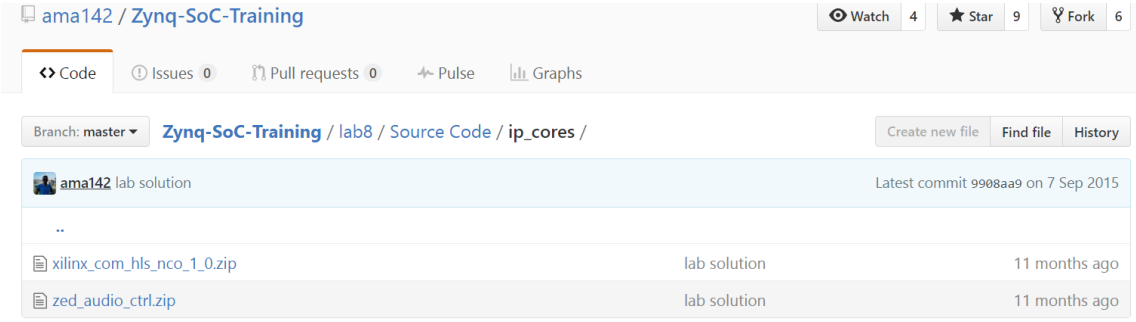

Figura 8. IP Cores para descargar.

Estos controladores IP Cores son desarrollados por Louise Crockett, Ross Elliot, Martin Enderwitz, Bob Stewart, David Northcote del equipo de Zynq.Guardar en un directorio

pueda consultar con facilidad y que no contenga ningún espacio. Tal como "C: \ ip_cores". Después de eso, extraer los IP en el mismo directorio, como se observa en la Figura 9.

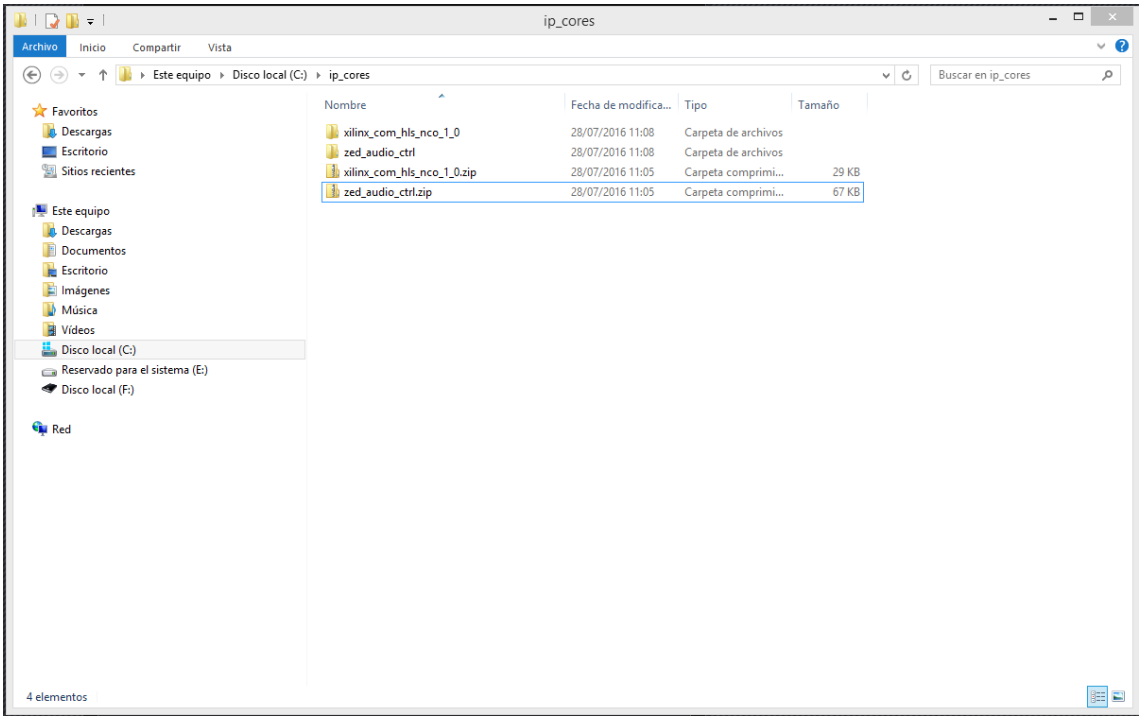

Figura 9.Carpetas descargadas IP Cores

1. Crear un proyecto nuevo en Vivado:

Clic en *File-> New Project*, para abrir *New Project Wizard* y clic *Next*. O también como se observa en el panel principal se encuentra el icono *Create New Proyect* como lo muestra la Figura 10 a continuación:

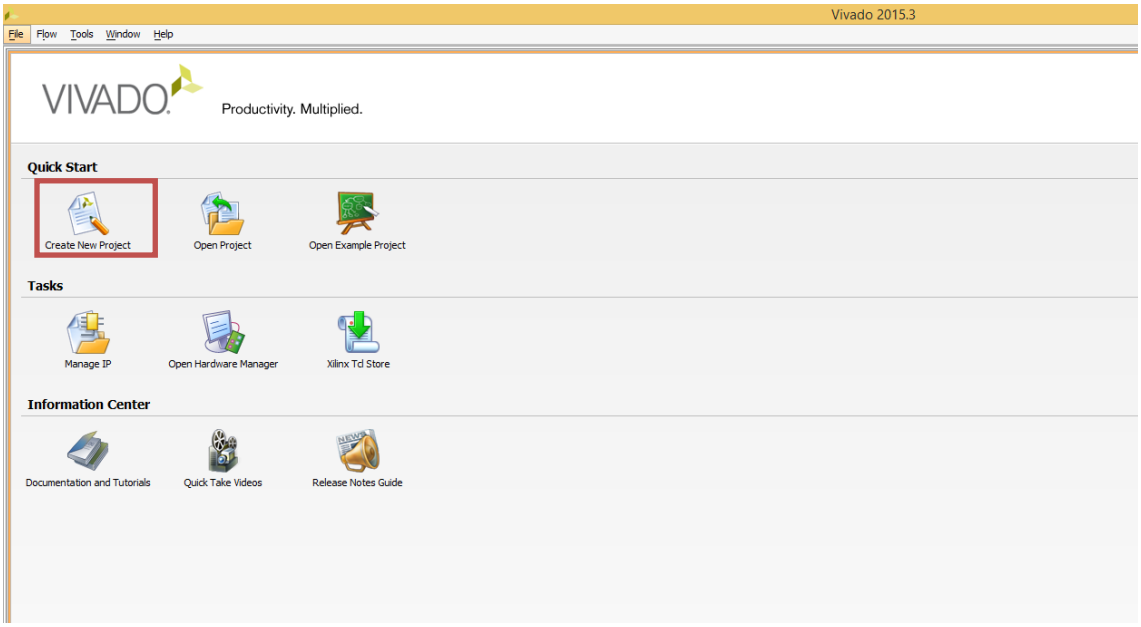

Figura 10.Ventana Inicial de Vivado.

2. Al parecer la ventana emergente como se muestra en la Figura 11. Click en Next.

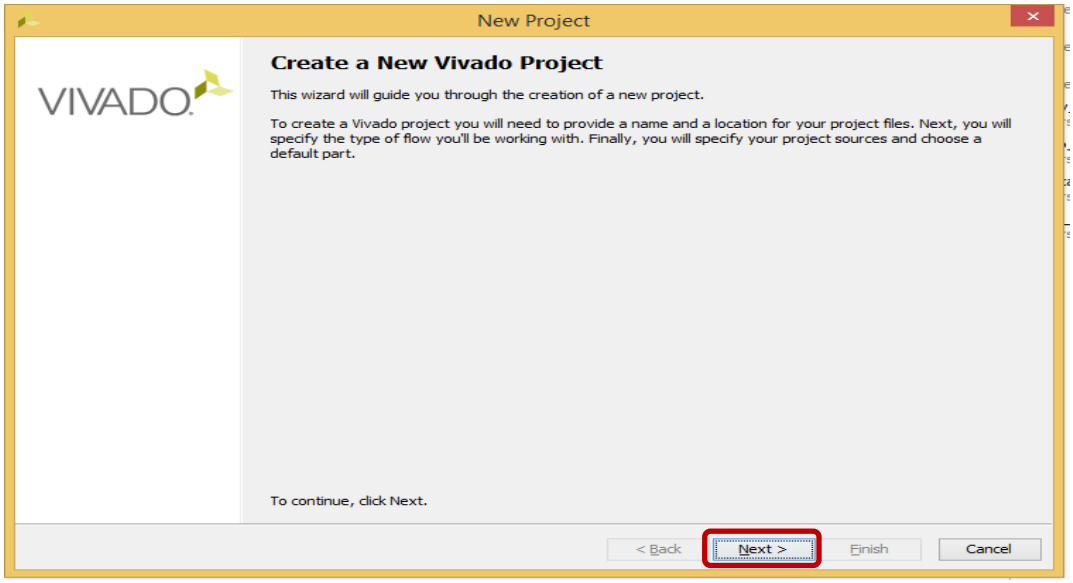

Figura 11.Ventana Siguiente.

En la siguiente ventana digite el nombre del proyecto, en este caso se llama 'App Audio'. Especificar el directorio en el que almacenar los archivos del proyecto

"C: /AppAudio /". Crear la carpeta en la unidad C. Garantiza no tener problemas a la hora de ejecutar el proyecto. Dejar marcado la casilla *Create Proyect Subdirectory* como se observa en la Figura 12.

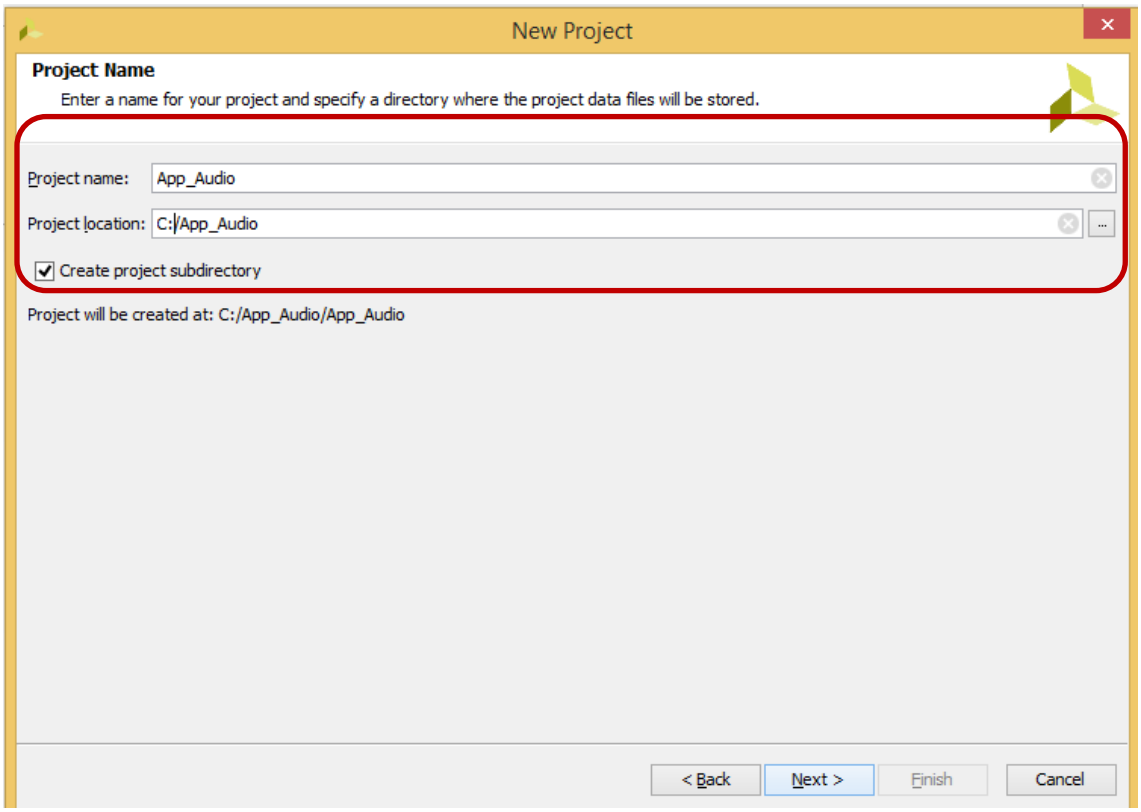

Figura 12.Creación del Proyecto.

3. Especifique el tipo de proyecto como se muestra en la Figura 13. Utilizar la selección por defecto *RTL Project*. Eligiendo esta opción tendremos la flexibilidad de añadir / modificar los códigos fuente más adelante.

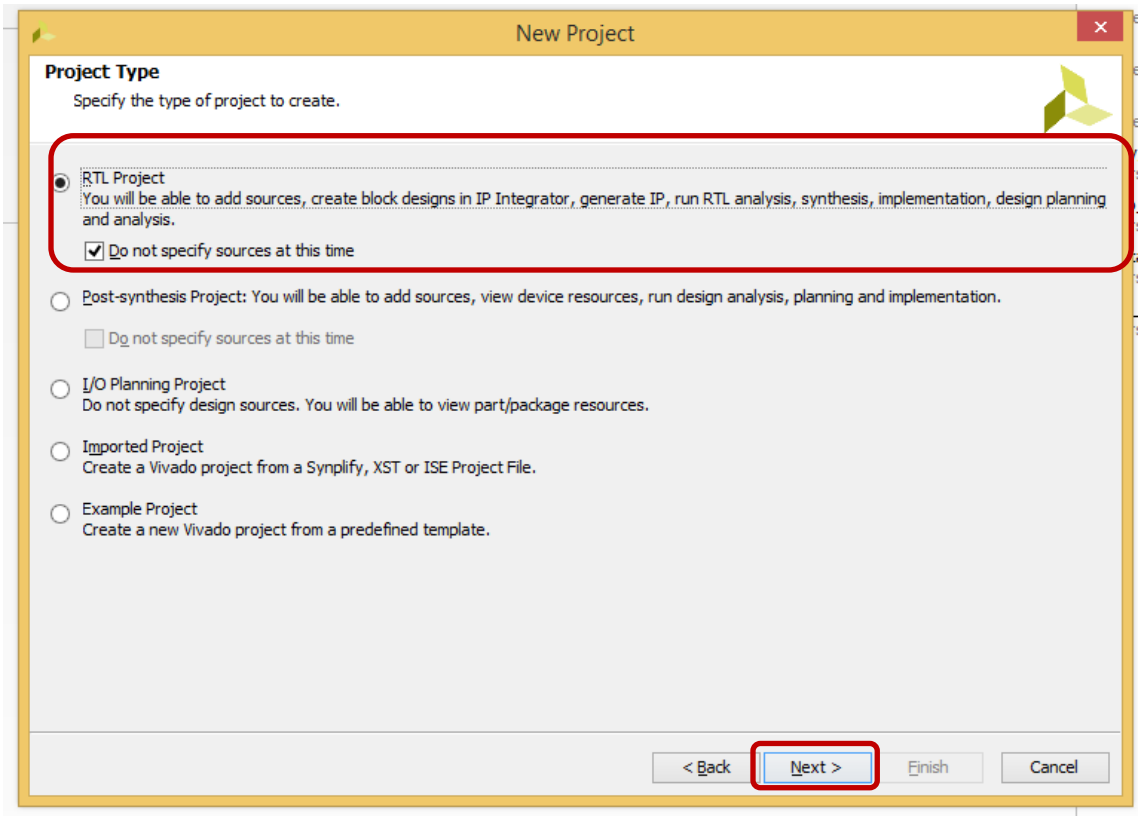

Figura 13.Selección Tipo de Proyecto.

4. Seleccione la Board para la implementación del proyecto el proyecto. Para este trabajo utilizaremos la *Zedboard Zynq Evaluation and Development Kit* Versión D. Haga clic en *Next*. Como se observa en la Figura 14.

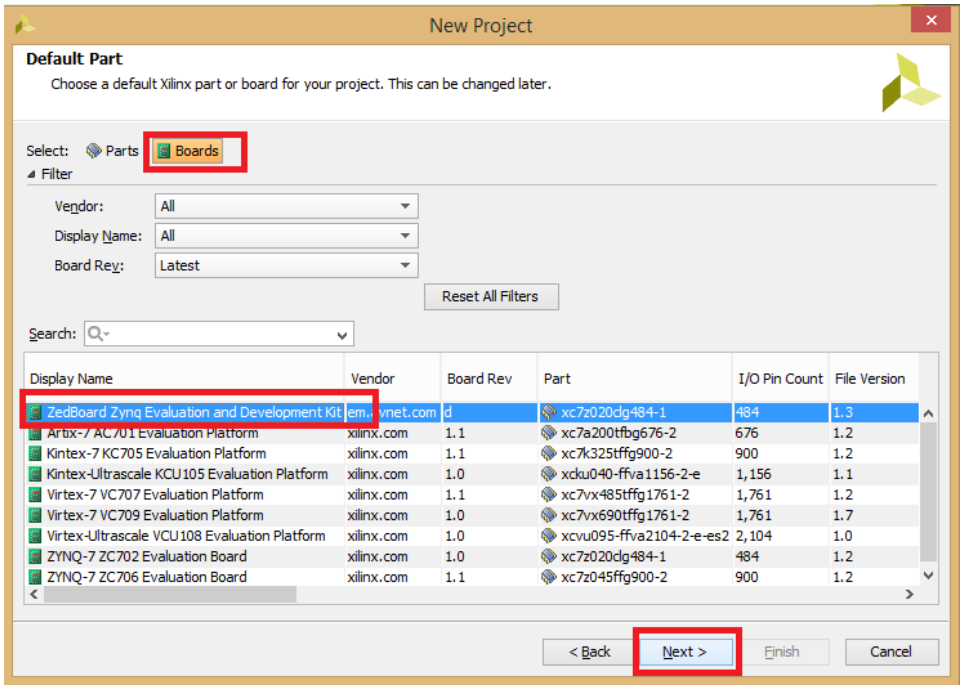

Figura 14.Elección de Board

5. Clic en *Next*, la ventana emergente de la Figura15 se mostrara en pantalla, especificando los componentes y configuraciones del proyecto y Clic en *Finish.*

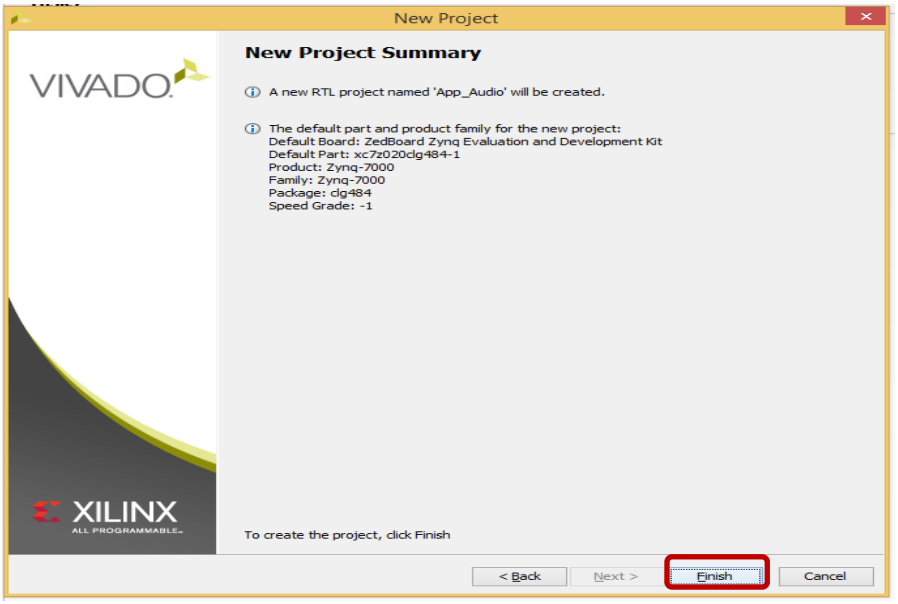

Figura 15.Configuración del Proyecto

6. Crear un proyecto embebido con el "*IP Integrator*":

   En la pantalla principal del proyecto creado en Vivado ubicar el menú "*Flow Navigator*",disponible en la parte superior izquierda. Luego el submenú "*IP Integrador*" y se da clic en "*Create Block Design*". Así se crea el bloque necesario para el diseño.

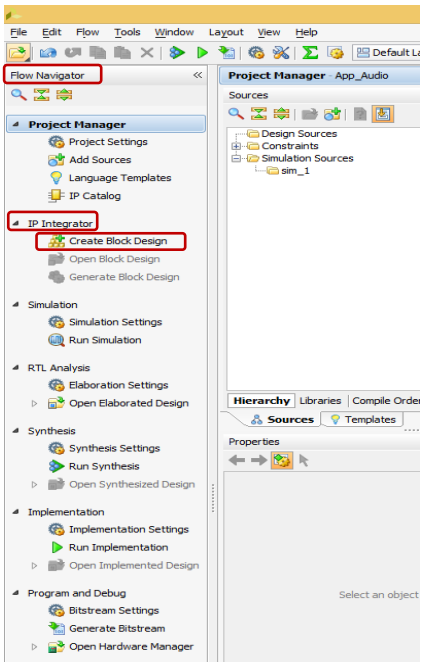

Figura 16.Pantalla Principal parte superior izquierda del Proyecto.

7. Escriba un nombre para el módulo y clic en OK. En este caso se nombra: "DAQ_AUDIO".

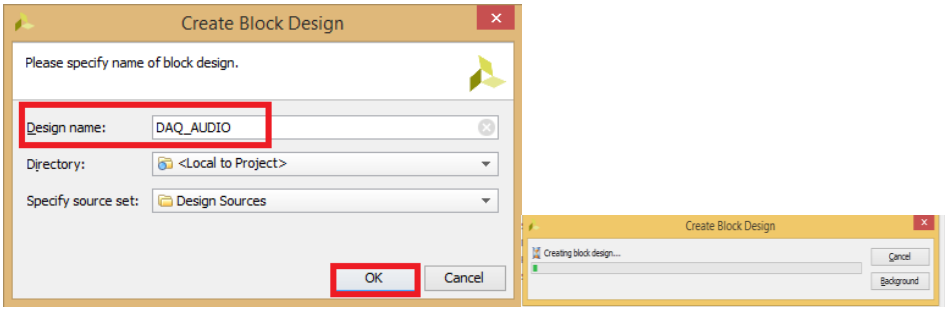

Figura 17.Creando y Nombrando el Bloque de Diseño

8. Clic en "*OK*" se le presenta una vista de diagrama de bloques en blanco en la interfaz gráfica de usuario Vivado. En el diagrama en blanco puede agregar y conectar los bloques de hardware que necesita en el diseño. Se añade el bloque *ZYNQ7 Processing System*. Mediante la adición de este bloque, se puede configurar uno de los núcleos del procesador ARM Cortex-A9 para su aplicación. Esto se realiza con el icono para agregar IP como se observa en la Figura 18. Aparecerá la ventana del catálogo que muestra todas las direcciones IP que se pueden agregar en este diseño. En este caso se agrega la llamada *ZYNQ7 Processing System* como lo muestra la Figura 18.

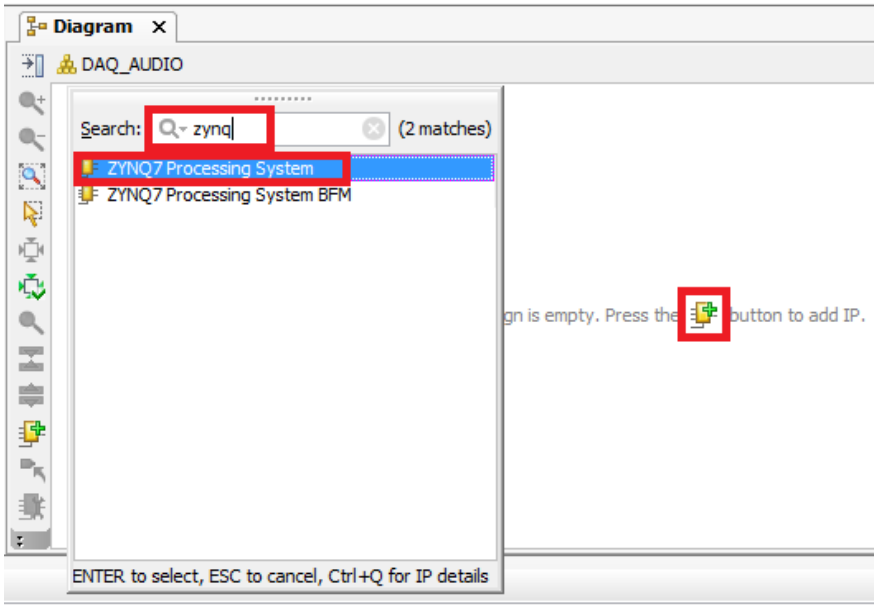

Figura 18.Catalogo IP

9. Crear un bloque llamado "*ZYNQ7 Proccesing System*", como se muestra en la Figura 19.

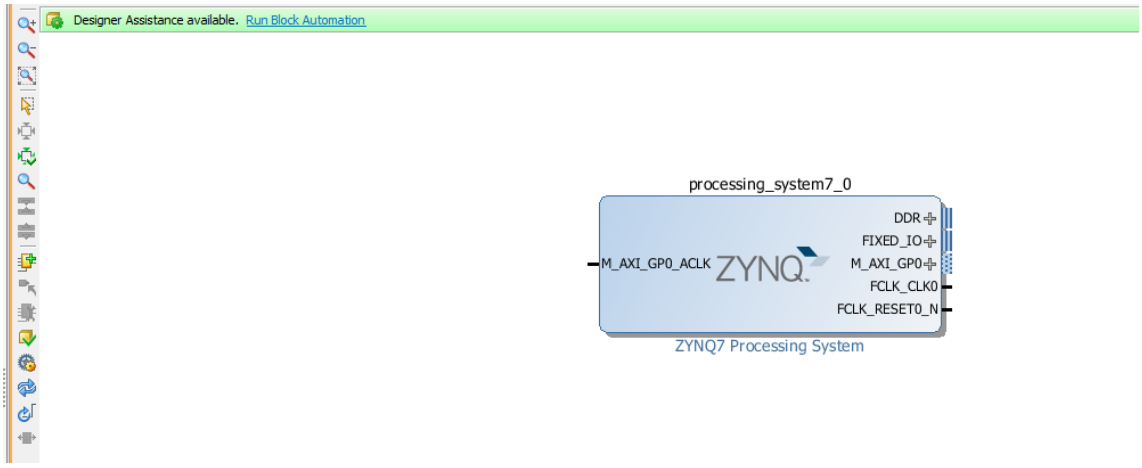

Figura 19.Bloque Creado

10. Clic en "*Run Block Automation*". En la barra de información verde donde muestra la Figura 20. En la ventana emergente dejar todas las opciones tal como están, si no márquelas como la Figura 21. Clic en "*OK".*

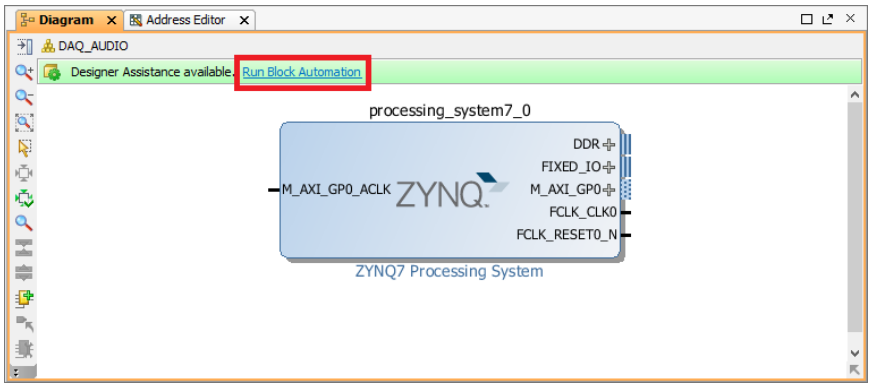

Figura 20.Bloque Processing System

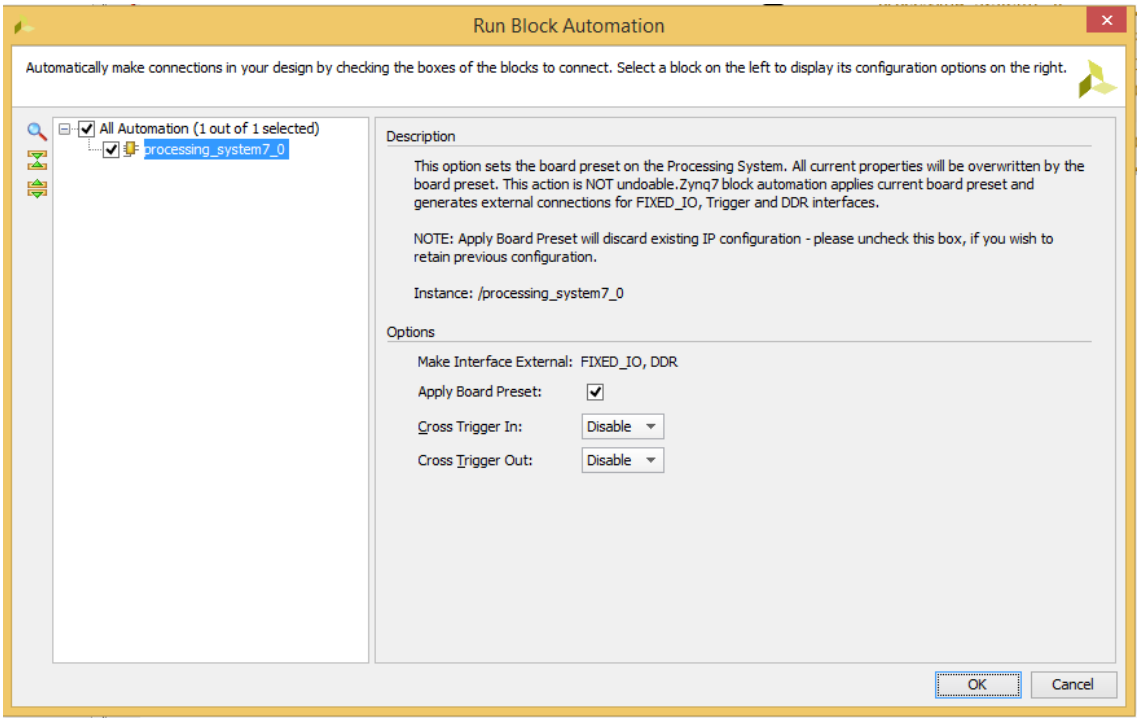

Figura 21.Run Block Automation

11. Al darle OK. El bloque se ve como la Figura 22.

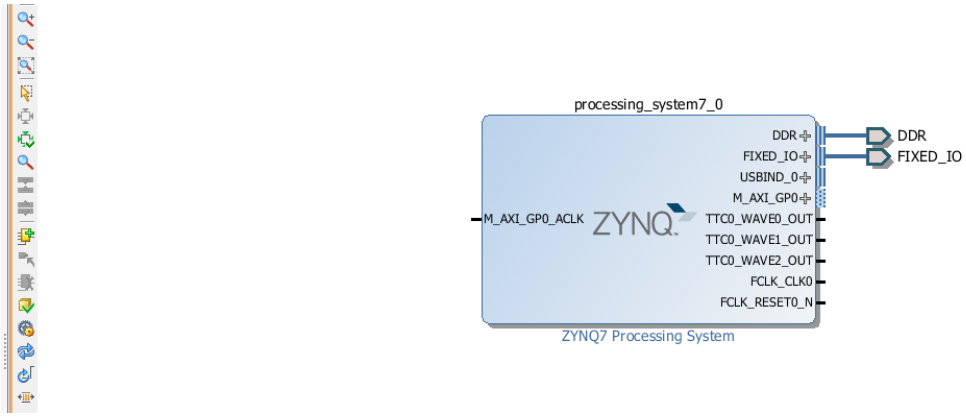

Figura 22. Bloque Processing System con los puertos de salida.

12. Doble clic al bloque *"ZYNQ7 Proccessing System"*.Se ingresa en la configuración y Clic en *"MIO Configuration"*, desplegar las opciones de *"I/O Peripherals"* y seleccionar las opciones como se muestra en la siguiente Figura 23.

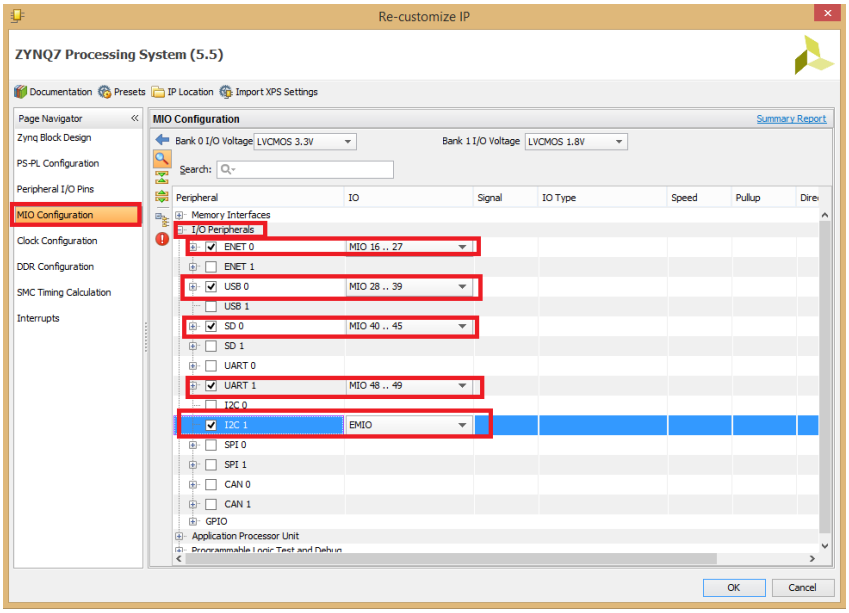

Figura 23.Re-Customize IP - MIO Configuration.

13. Clic en *"Clock Configuration"* y desplegar las opciones de *"PL Fabric Clocks"*.Seleccionar FCLK_CLK1 y ponerlo a 10 Mhz, como se observa en la Figura 24.

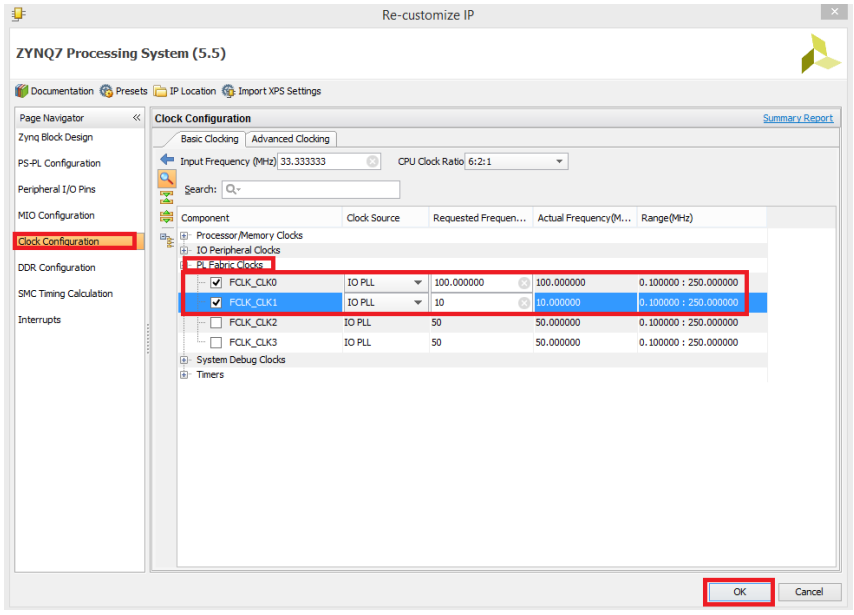

Figura 24.Re-Customize IP- Clock Configuration.

14. Este es el reloj para configurar el ADAU1761 Codec. Clic en *ok* para cerrar la ventana *IP Customize*. El bloque *"ZYNQ7 Proccessing System*" se modifica. Hacer clic derecho en el pin *"IIC_1"* y seleccionar *"Make External".* Hacer lo mismo con el pin "FCLK_CLK1". Él se ve como la Figura 25:

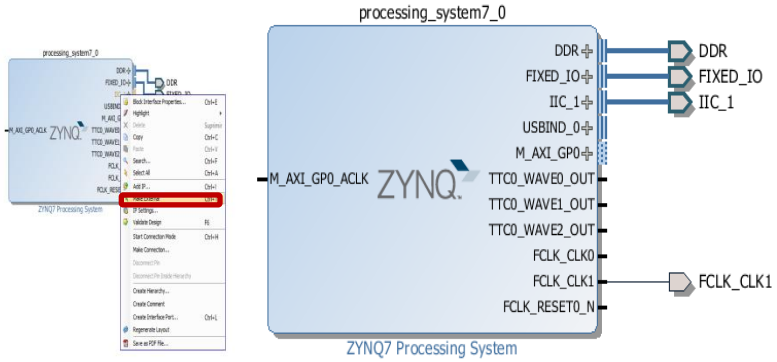

Figura 25.Modificaciones al Bloque ZYNQ7

15. Ahora se deben agregar los archivos de las carpetas que se guardaron previamente en la ruta "C:\Ip_cores". Para esto seguir los pasos:

    1) En*"Flow Navigator",* luego en el menú *"Project Manager"*se hace clic sobre *"Project Setting".*

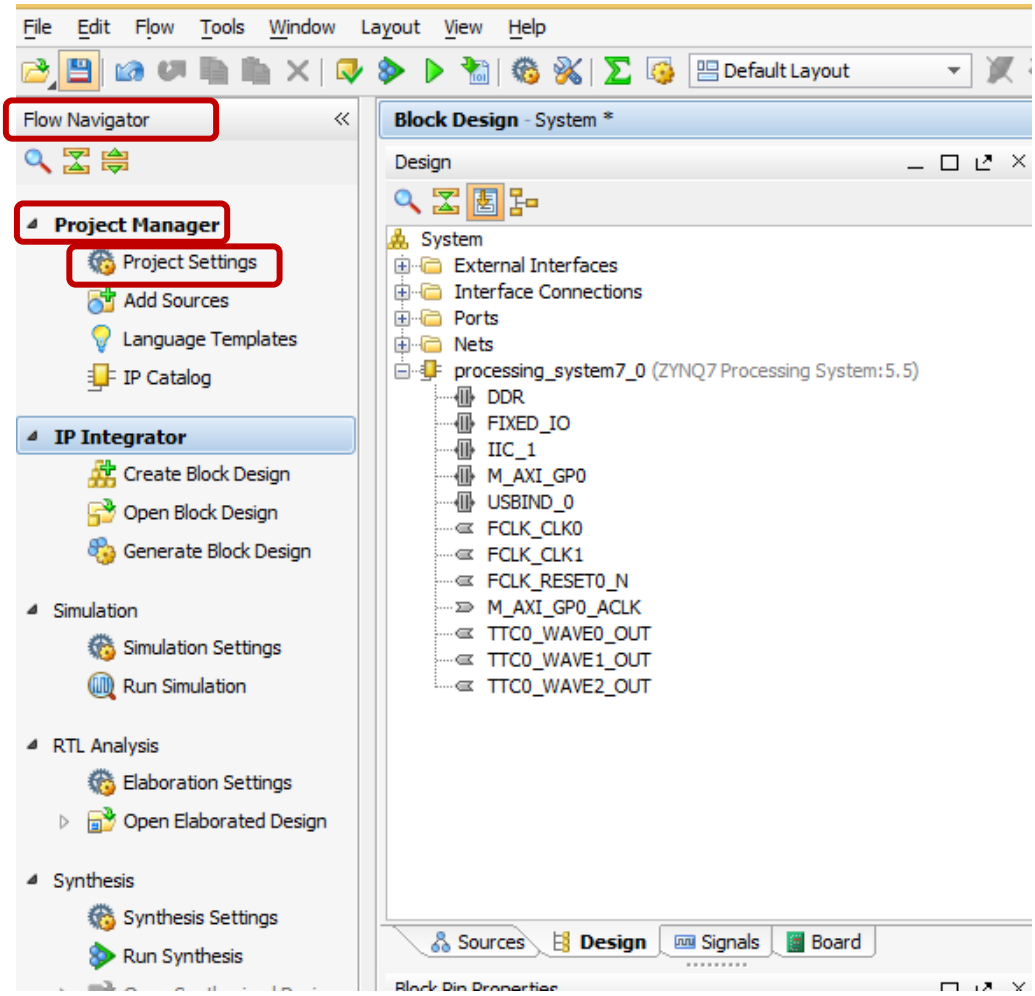

Figura 26. Instrucciones para agregar los IP Cores.

2) En la ventana que se abre hacer clic sobre *"IP"*.uego clic en la pestaña "*Repository Manager"* y hacer clic en el botón "*Add Repository"* identificado con el símbolo "+" verde, como se observa en la Figura 27.

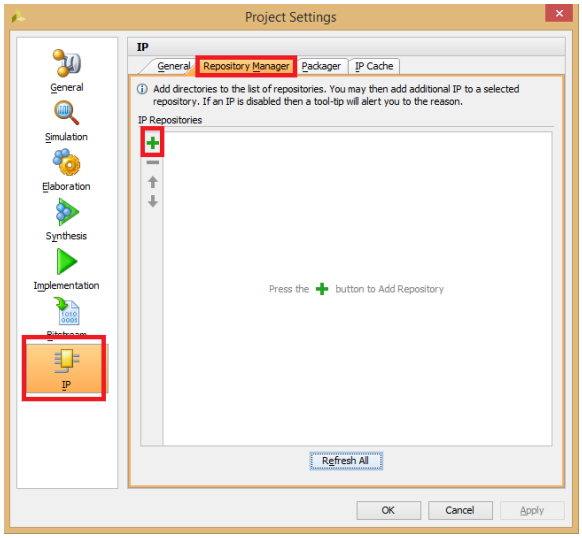

Figura 27. Ventana Emergente Project Settings.

3) En la ventana que se abre ubicar la ruta creada anteriormente (C:\Ip_cores) y hacer clic en *"Select"* como se observa en la Figura 28.

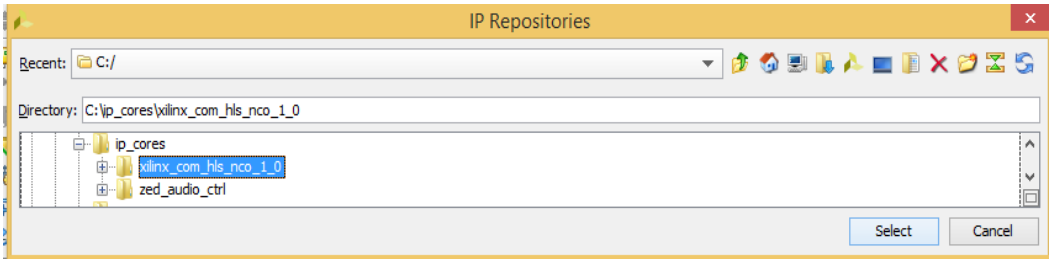

Figura 28.IP Repositories.

4) Clic en "*Apply*" y después en "*Ok*". De esta forma quedan agregados al Vivado los dos bloques que contienen las carpetas creadas.

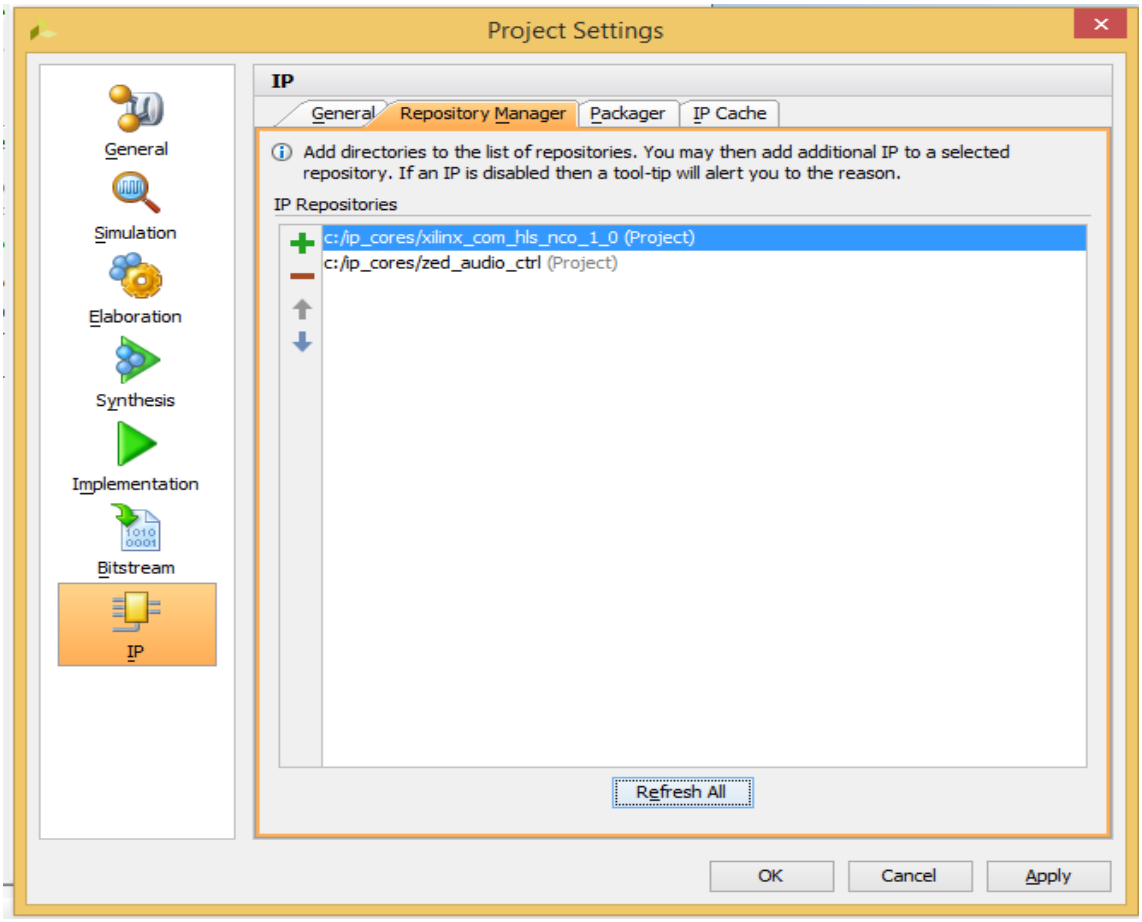

Figura 29. Ventana Emergente Project Settings-IP

Siguiendo los anteriores pasos se añade la *zed_audio_ctrl* y *xilinx_com_hls_nco_1_0* al *IP Repositories* del proyecto actual. El siguiente paso es añadirlo al diseño de bloques y conectarlo al sistema de procesamiento *Zynq.*

16. En la ventana del diagrama, clic en cualquier lugar y seleccione agregar IP. Digite *‘‘nco’’* en el campo de búsqueda. Seleccione y doble clic en *Nco* para agregarlo, como se observa en la Figura 30.

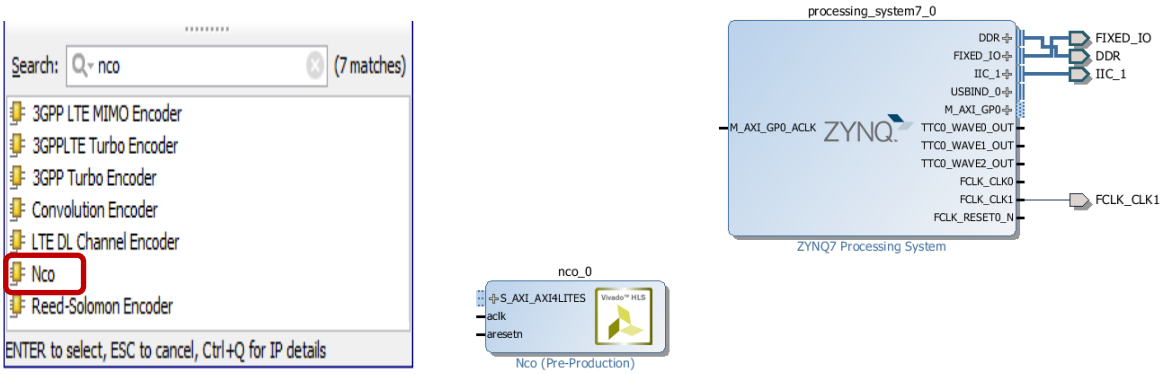

Figura 30.Agregar NCO

17. Clic en *"Run Connection Automation"* y en la ventana que se abre dejar todo por defecto y hacer clic en *"OK"*.

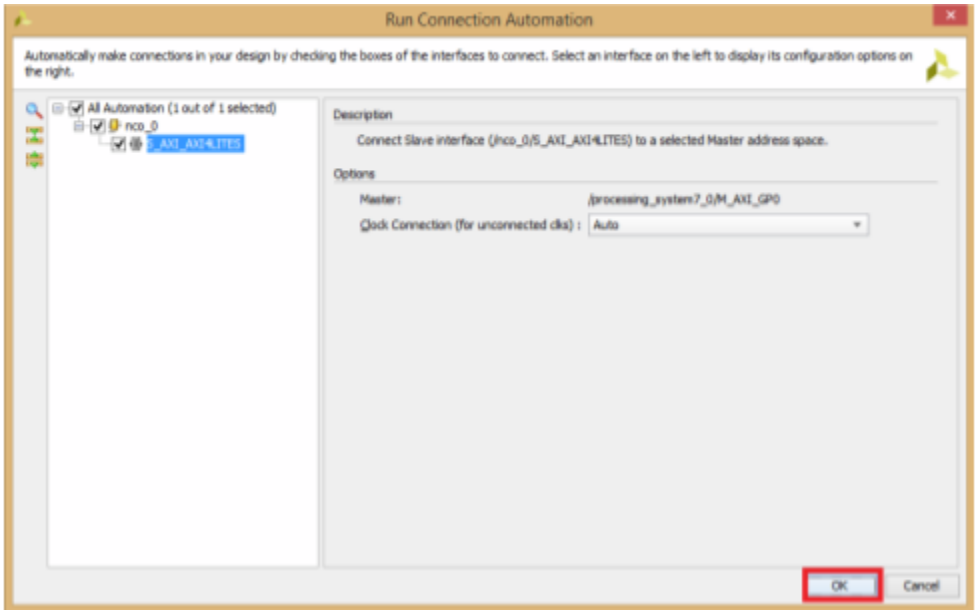

Figura 31.Run Connection Automation Nco.

18. Repetir los pasos anteriores para agregar el bloque *"zed_audio_ctrl",* como se observa en la Figura 32.

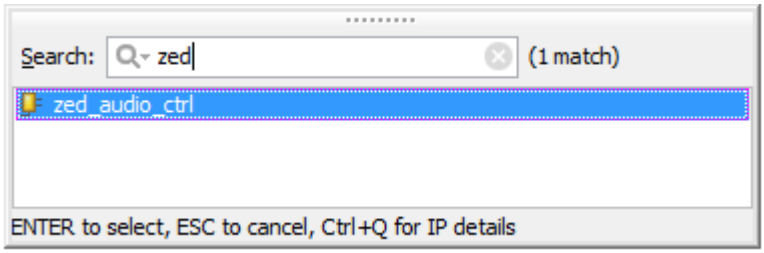

Figura 32.Búsqueda zed_audio_ctrl.

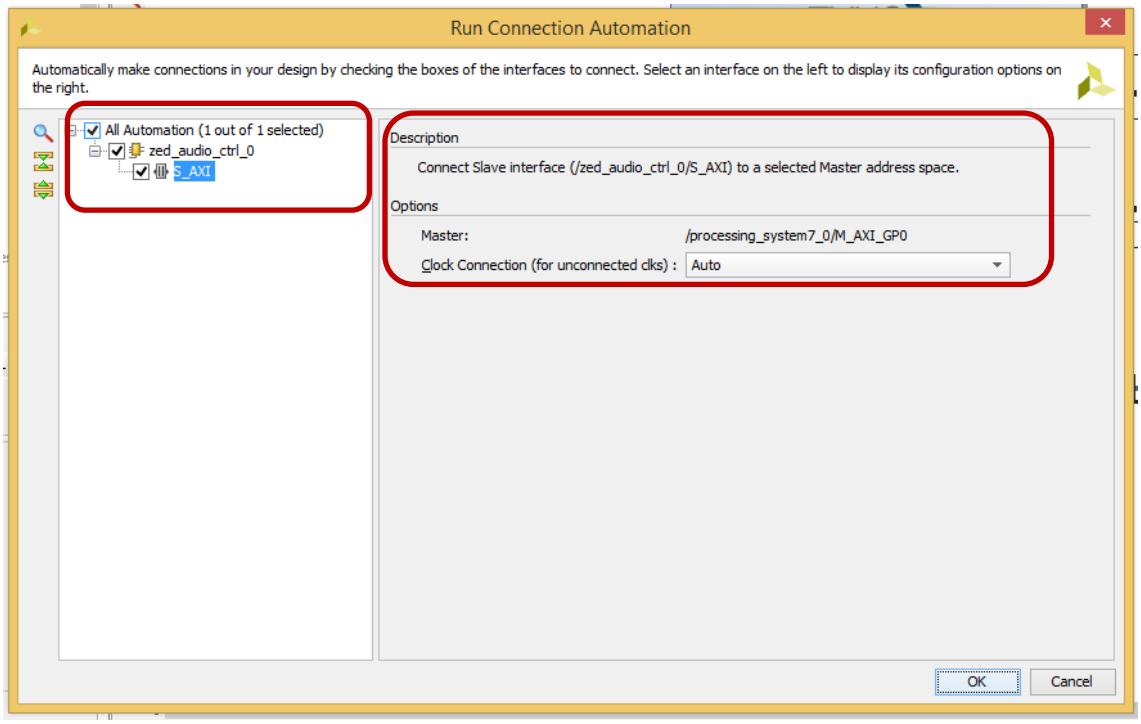

Figura 33.Run Connection Automation Zed_audio_ctrl_0.

19. Ya generados los bloques, se debe hacer clic derecho sobre el área de trabajo del diseño de bloques del diagrama y hacer clic en *"Regenerate Layout"* o en el icono.

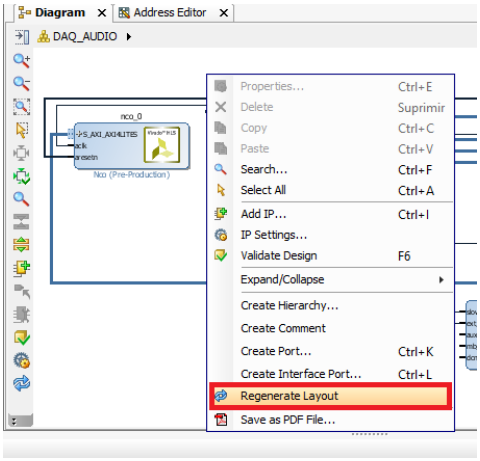

Figura 34.Regenerate Layout

20. El diagrama debe quedar como se muestra en la Figura 35, de lo contrario revisar los pasos anteriores.

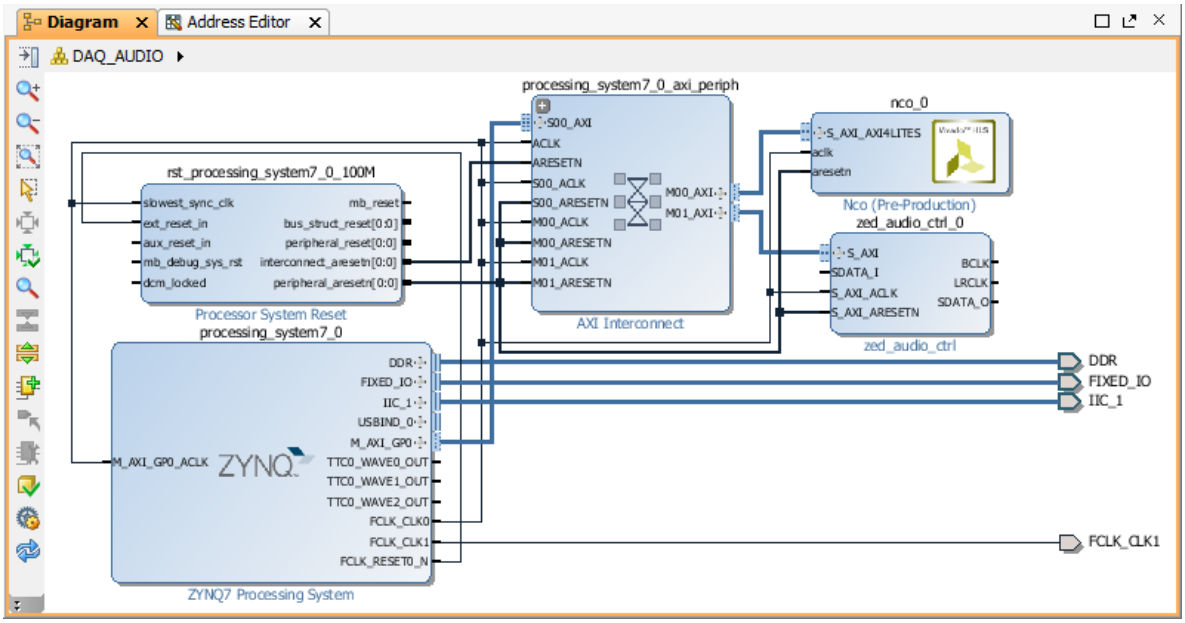

Figura 35.Diagrama de bloques del diseño

21. Clic derecho en el pin *"BCLK"* del bloque *"zed_audio_ctrl_0"* y seleccionar "*Make external"*. Repetir lo mismo para los pines *"LRCLK", "SDATA_0*" y *"SDATA_1"*. El diagrama debe quedar como se observa en la Figura 36.

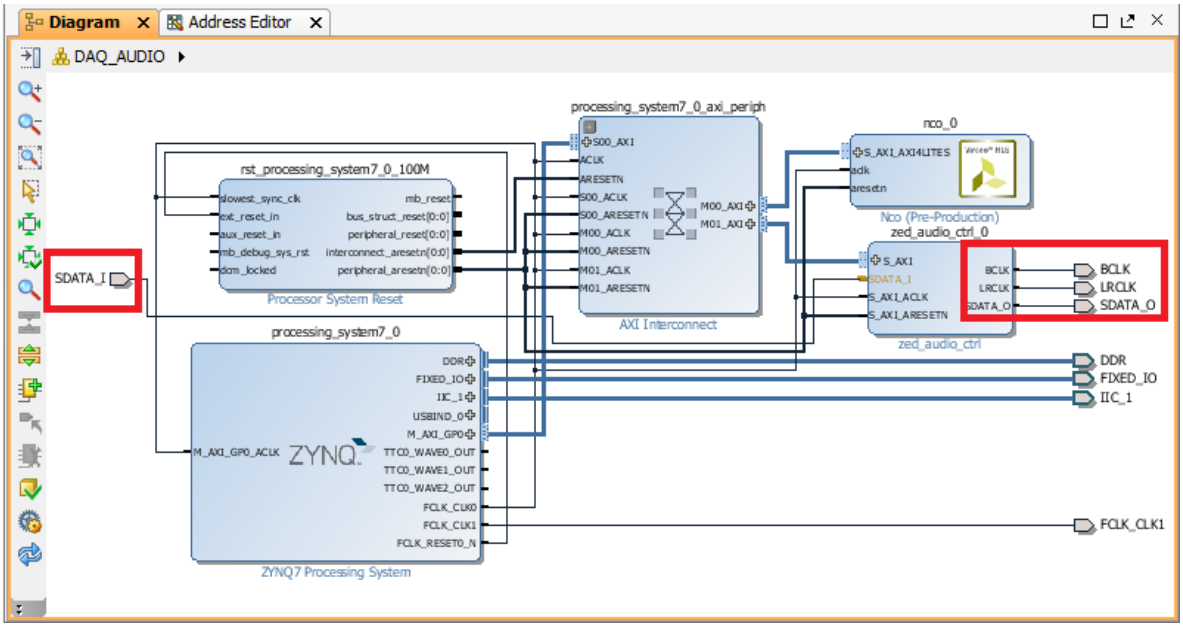

Figura 36.Conexiones externas del bloque zed_audio_ctrl_0.

Estos puertos se asocian con los puertos conectados al Codec ADAU1761 de audio, es decir, el reloj de izquierda-derecha (LRCLK), reloj de bits (BCLK) y datos de audio en serie en las líneas de entrada / salida (SDATA_I = SDATA_ADC) y (SDATA_DAC = SDATA_O) .

22. Se agrega el controlador de "*switches*" y "*leds*" *llamado* "*GPIO*". Para esto seguir los pasos:

    1) En "*Diagram*" hacer clic en "*Add IP*". En la ventana que se abre escribir "*gpio*" y dar doble clic para agregarlo como se observa en la Figura 37.

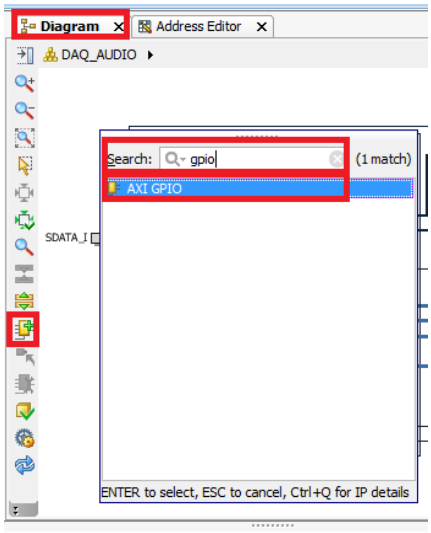

Figura 37.Ventana para agregar el bloque IP AXI GPIO.

2) El módulo GPIO se configura para tener dos canales, uno para los LED y uno de los interruptores. Doble clic en el bloque de GPIO para abrirla ventana *IP Re-Customize*. En la pestaña "*Board"*, establecer interfaz *IP GPIO* (canal 1) a *leds_8bits*, y *GPIO2* (canal 2) a *sws_8bits* como se muestra en la Figura 38. Clic en *OK* para salir de la ventana.

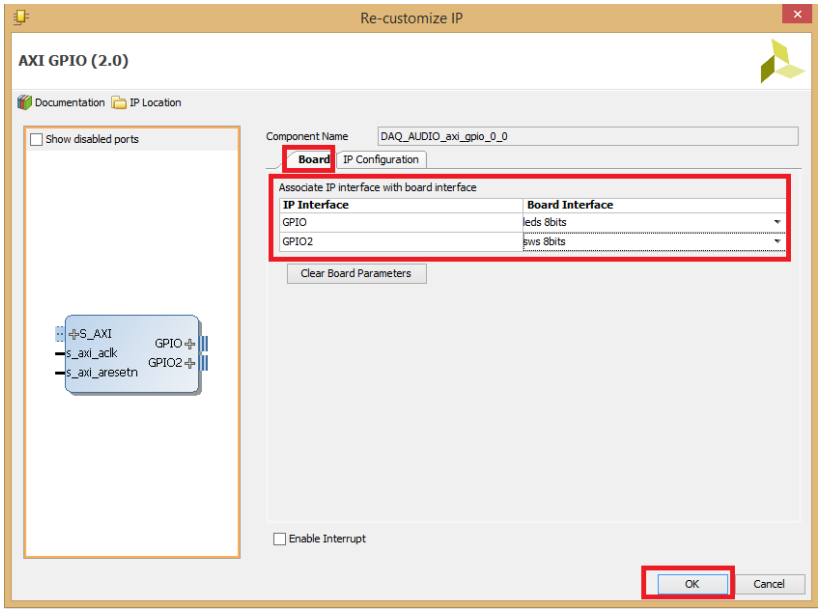

Figura 38.Re-Customize IP AXI GPIO

3) Para conectar este bloque hacer clic en *"Run Connection Automation".*

Seleccionar todas las opciones y hacer clic en *"OK"*como se muestra en la

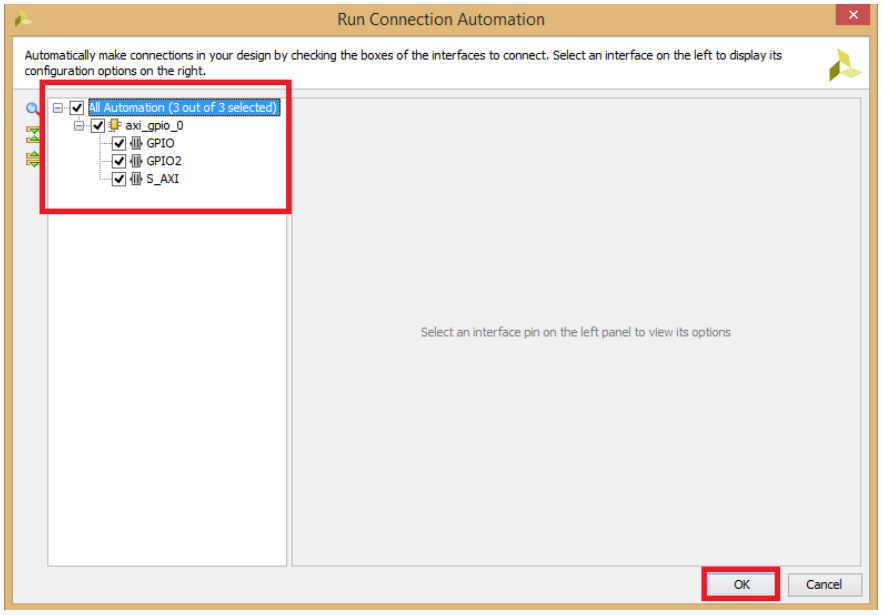

Figura 39.Run Connection Automation.

23. Se debe asignar los valores para los bits menos significativos de la dirección del I2C ADAU1761. Para esto seguir los pasos:

- Hacer clic derecho en la ventana del diagrama de bloques y seleccionar *"Create Port"*.

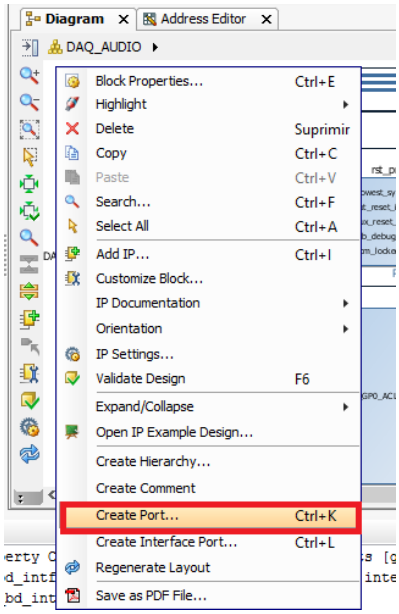

Figura 40. Menú- Create Port.

- En la ventana que se despliega, configurar como se muestra en la siguiente Figura y clic en "*OK*".

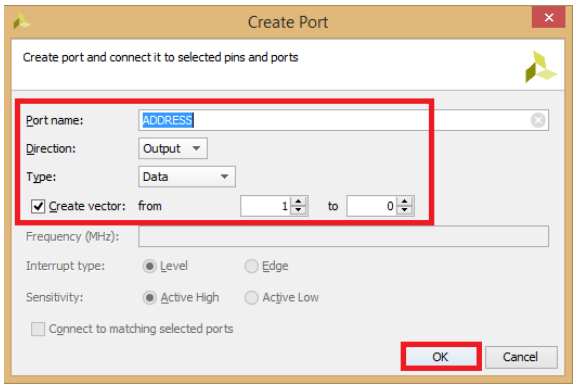

Figura 41.Create Port

- En la pestaña *"Diagram"* hacer clic en *"Add IP"*. En la ventana que se abre escribir *"cons"* y agregar *"Constant"*. El IP *"Constant"* se utiliza para unir el puerto*"ADDRESS [1:0]"* a un valor lógico fijo de cero.

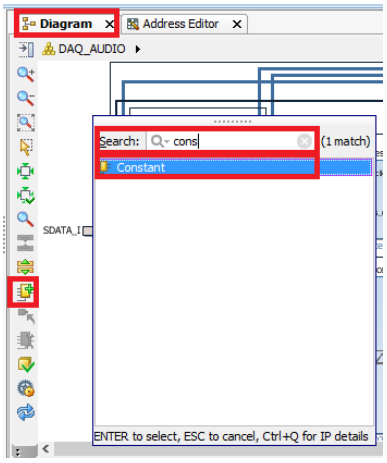

Figura 42.Constant.

- Doble clic en el bloque en el bloque creado, para acceder a su configuración y adecuar las opciones como se presentan en la Figura 43 y clic en *"OK"*.

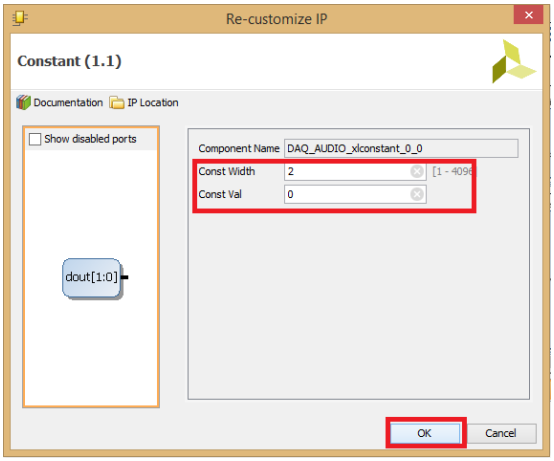

Figura 43.Re-Customize Constant.

- Conectar el pin *"dout[1:0]"* del bloque *"Constant"* al pin *"Address[1:0]"*. Hacer clic sostenido en uno de los pines y arrastrarlo hasta el otro pin.

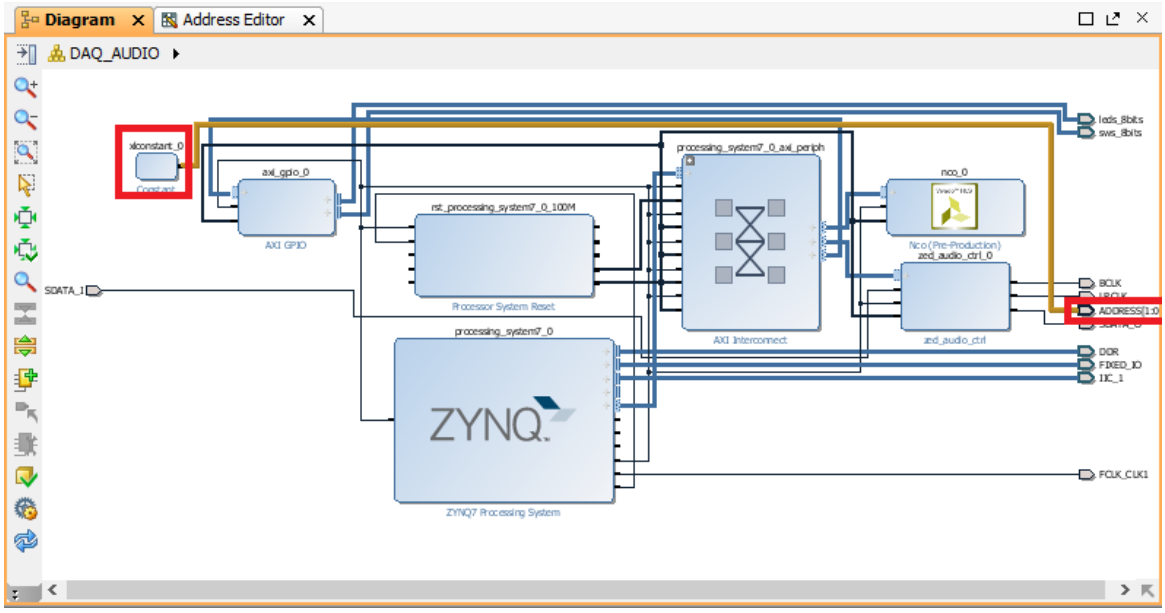

Figura 44.Conectar los pines.

- Se debe aplicar el *"Regenerate Layout"*. El diagrama debe quedar como se observa en. la Figura 45, de contrario se debe verificar los pasos anteriores hasta llegar al mismo diagrama.

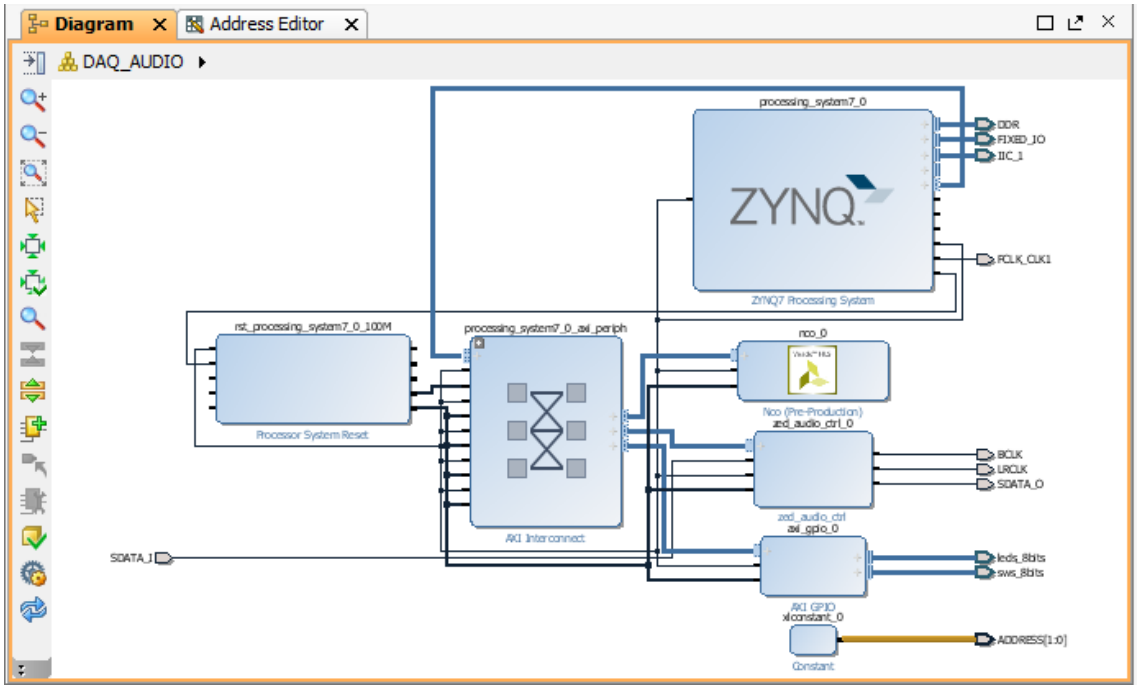

Figura 45.Diagrama con las últimas modificaciones

24. Se debe agregar *"physical constraints"*. Para esto seguir los pasos:

- Ubicar la pestaña *"Flow Navigator"* en el menú *"Project Manager"* hacer clic en *"Add Sources"*.

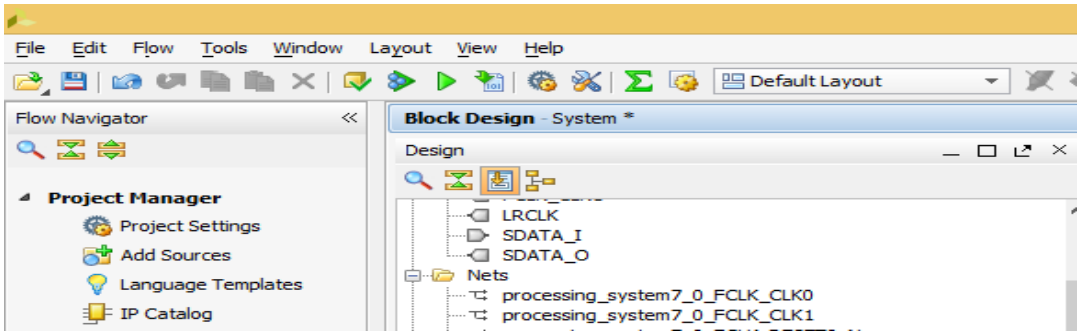

Figura 46.Add Sources.

- En la ventana que se abre seleccionar "*Add or Create Constraints*" y hacer clic en *"Next"*.

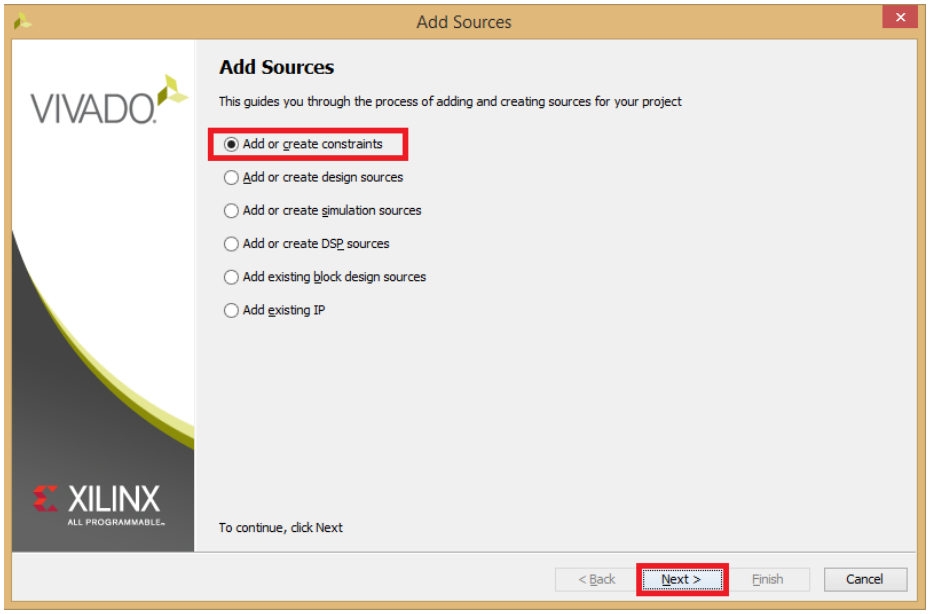

Figura 47.Add Sources.

- En la ventana que se abre hacer clic en el botón "+" de color verde y seleccionar *"Create File".*

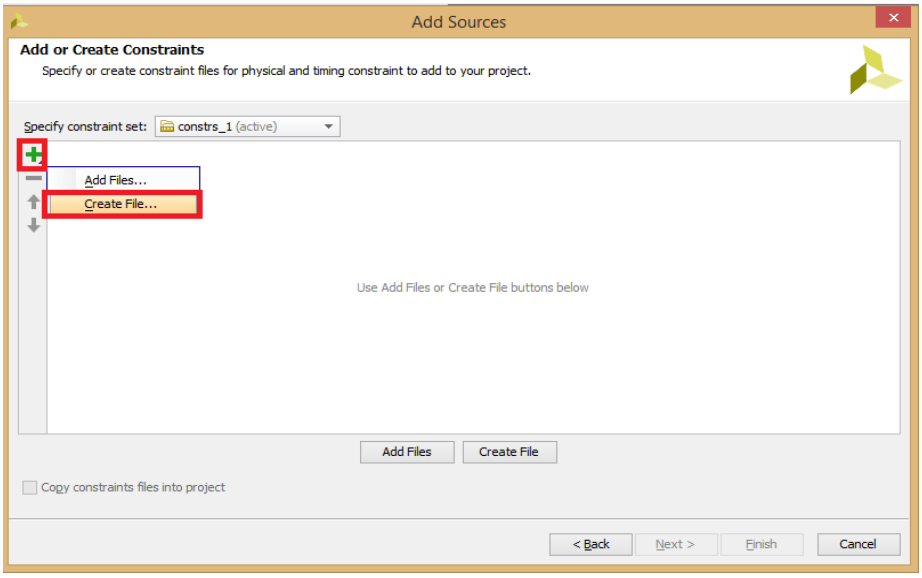

Figura 48.Add or Create Constraints.

- En la ventana que se abre escribir las opciones que se observa en la Figura 49. Clic en *"OK"* y luego en *"Finish"*.

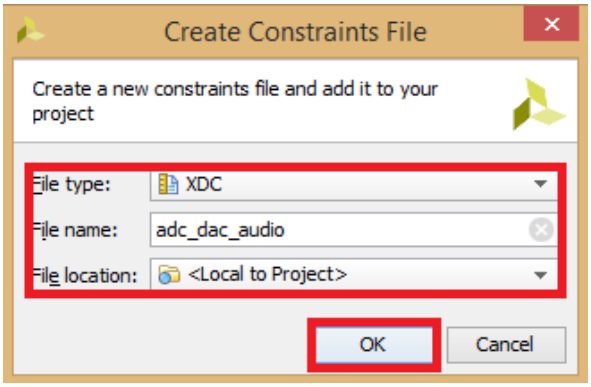

Figura 49.Create Constraints File.

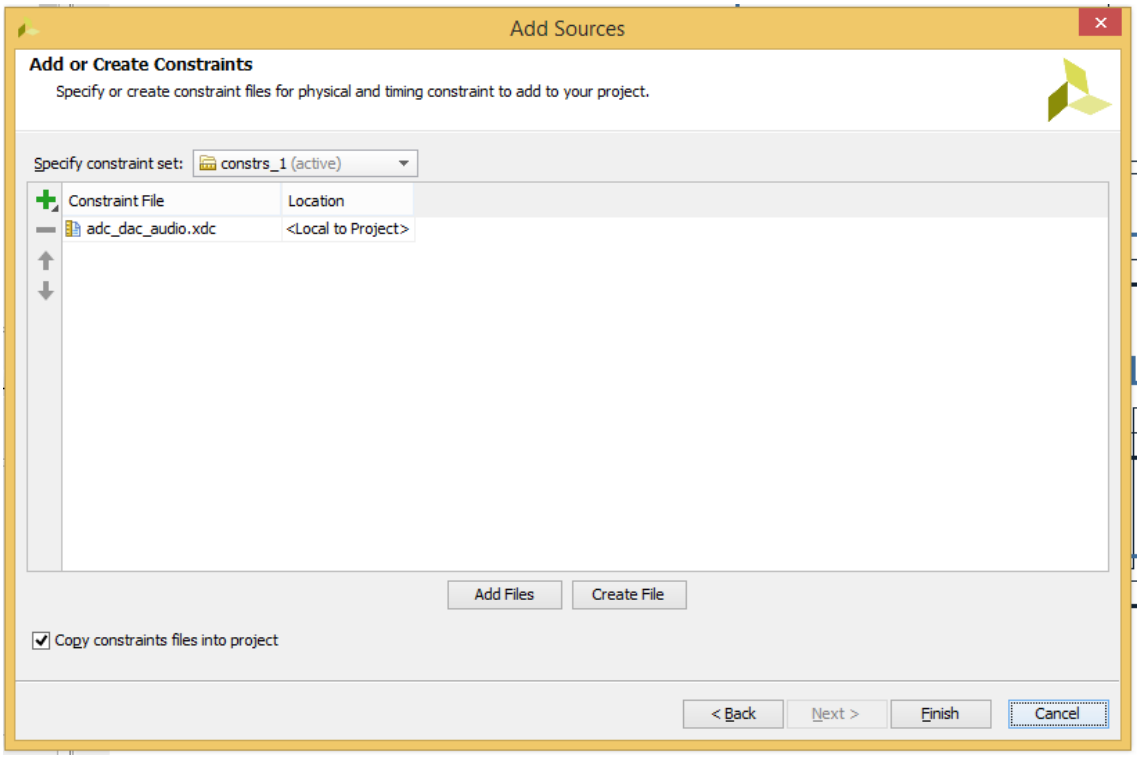

Figura 50.Ventana Emergente al Agregar adc_dac_audio.

- Digerirse a la pestaña *"Source"* desplegar la carpeta *"Constraints"* y hacer doble clic en *"adc_dac_audio.xdc"*.

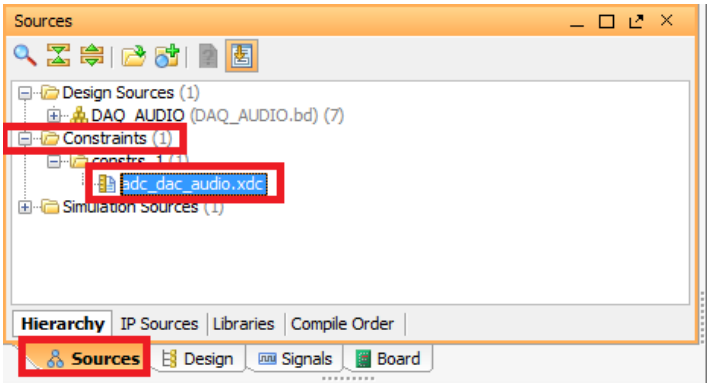

Figura 51.Pestaña "Source".

- En la ventana que se abre en blanco pegar el código que se muestra a continuación y se observa como en la Figura 52.

```
# ZedBoard Audio Codec Constraints
set_property PACKAGE_PIN AA6 [get_ports BCLK]
set_property IOSTANDARD LVCMOS33 [get_ports BCLK]
set_property PACKAGE_PIN Y6 [get_ports LRCLK]
set_property IOSTANDARD LVCMOS33 [get_ports LRCLK]
set_property PACKAGE_PIN AA7 [get_ports SDATA_I]
set_property IOSTANDARD LVCMOS33 [get_ports SDATA_I]
set_property PACKAGE_PIN Y8 [get_ports SDATA_O]
set_property IOSTANDARD LVCMOS33 [get_ports SDATA_O]
#MCLK
set_property PACKAGE_PIN AB2 [get_ports FCLK_CLK1]
set_property IOSTANDARD LVCMOS33 [get_ports FCLK_CLK1]
set_property PACKAGE_PIN AB4 [get_ports iic_1_scl_io]
set_property IOSTANDARD LVCMOS33 [get_ports iic_1_scl_io]
set_property PACKAGE_PIN AB5 [get_ports iic_1_sda_io]
set_property IOSTANDARD LVCMOS33 [get_ports iic_1_sda_io]
set_property PACKAGE_PIN AB1 [get_ports {ADDRESS[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ADDRESS[0]}]
set_property PACKAGE_PIN Y5 [get_ports {ADDRESS[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ADDRESS[1]}]
```

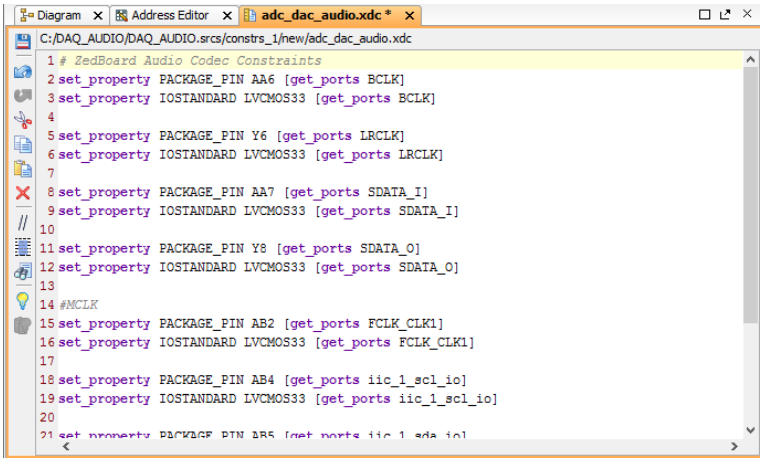

Figura 52.Adc_dac_audio.

- Finalmente hacer clic en *"save"* y los *"physical constraints"* se conectarán.

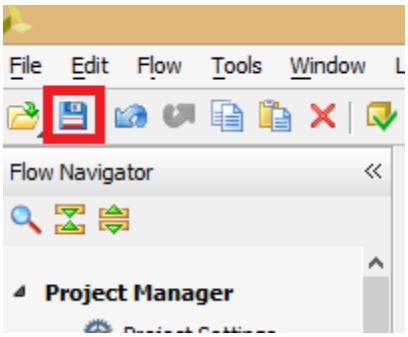

Figura 53.Save.

25. Después de haber creado el diseño del diagrama de bloques se debe generar el "*Bitstream"* para llevar el proyecto al SDK. Para esto seguir los pasos:

- Ubicar la pestaña *"Source"* y hacer clic derecho en *"DAQ_AUDIO"* y seleccionar *"Create HDL Wrapper".*

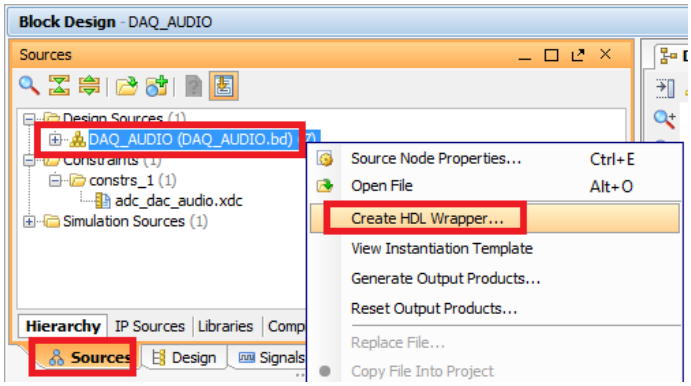

Figura 54.Create HDL Wrapper.

- En la ventana emergente seleccionar "*Let Vivado manage wrapper and auto-update"* y clic en *"OK"*.

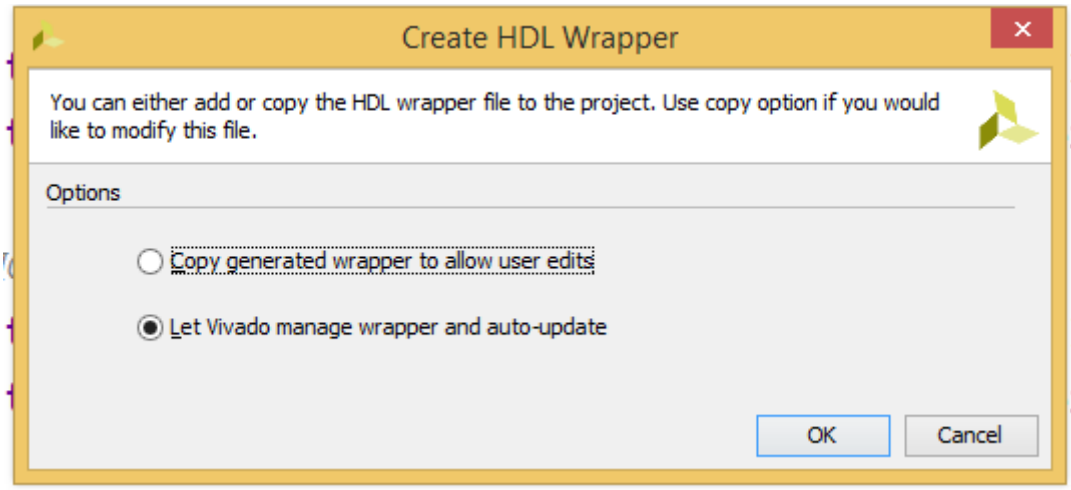

Figura 55.Ventana Emergente. Create HDL Wrapper.

- Ubicar la pestaña *"Flow Navigator"* en el menú *"Program and Debug"* hacer clic en *"Generate Bitstream" como se observa en la Figura 56.*

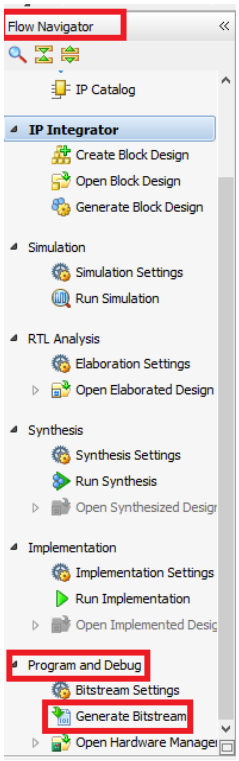

Figura 56. Generate Bitstream.

- En la ventana que se abre seleccionar “Save” como lo muestra la Figura 57.

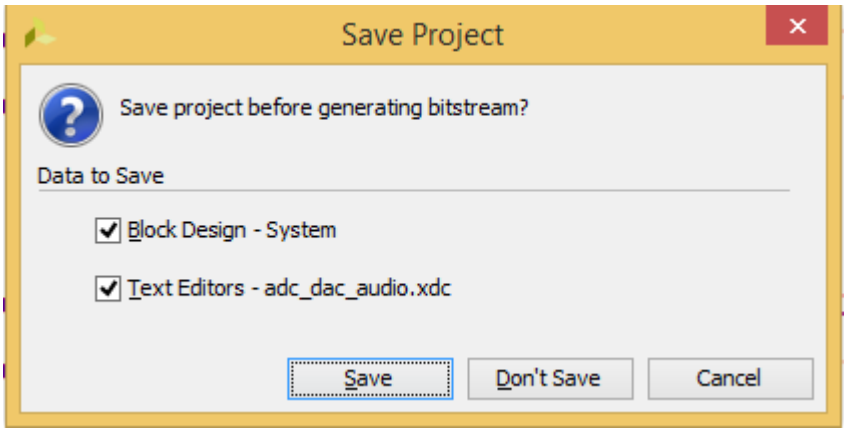

Figura 57.Save Project.

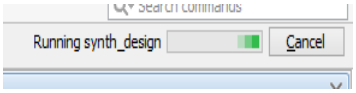

Figura 58. Barra de Proceso

- Cuando se finaliza la creación del *bitstream* sale una ventana en la cual se debe seleccionar *"View Reports"* y clic en *"OK".*

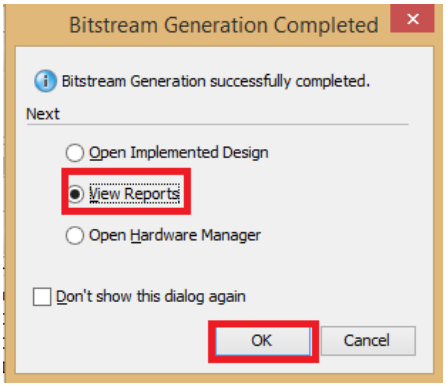

Figura 59.Bitstream Generation Completed.

- Si no se visualizan los reportes dirigirse a *Tools* en la parte superior de Vivado y en "*Windows Behavior*" seleccionar la opción para visualizarlos como lo muestra la Figura 60, luego clic en "*OK".*

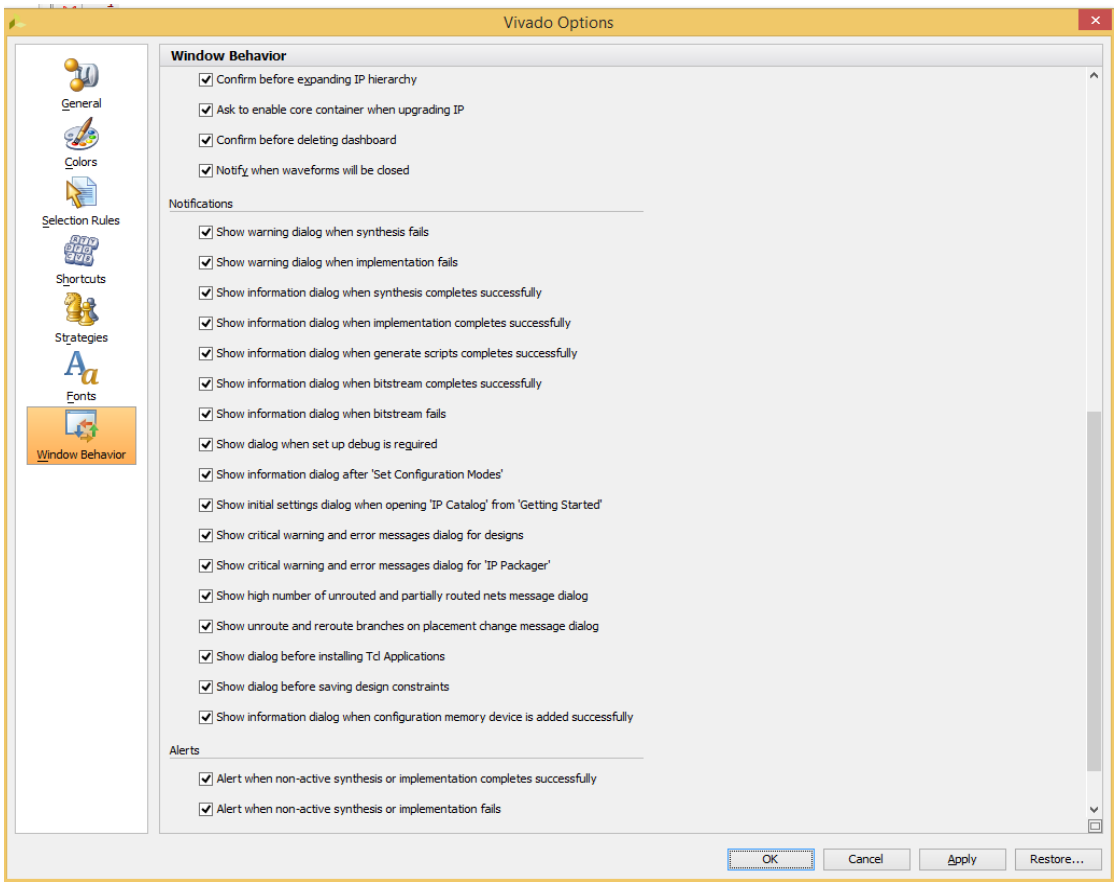

Figura 60. Vivado Options

- Verificar que en "*Project Settings*" el desplegable "*Target Language*" se encuentre en VHDL y clic en "*OK*".

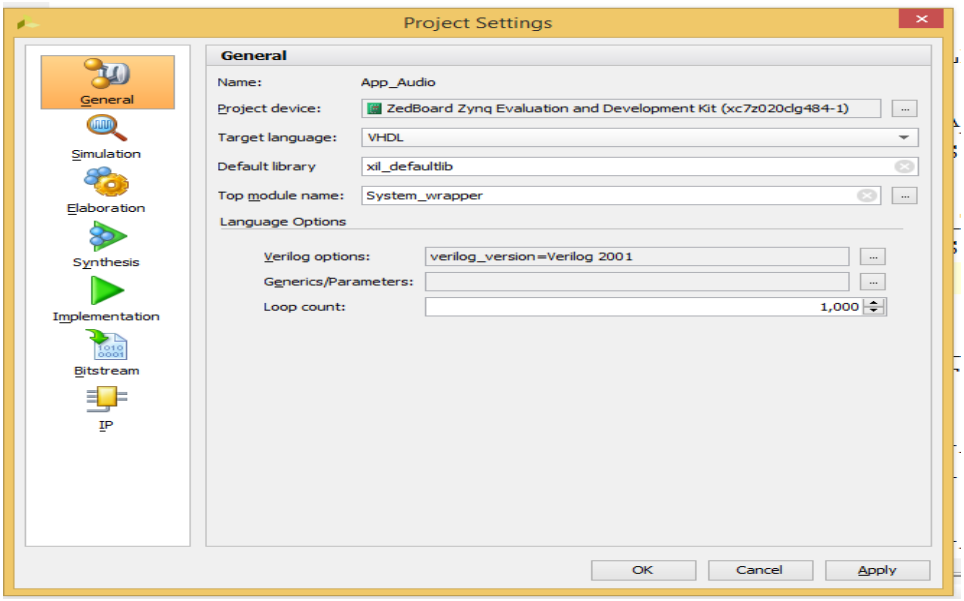

Figura 61.Proyect Settings.

26. Luego de generado el *bitstream* se debe exportar el hardware para llevarlo al SDK. Para esto seguir los pasos:

- Clic en *"File"*, luego en *"Export"* y hacer clic en *"Export Hardware"*, *como se observa en la Figura 62.*

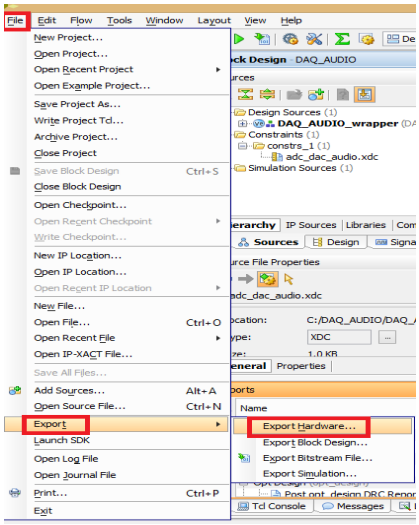

Figura 62.Ruta Export Hardware.

- En la ventana que se abre marcar la opción *"Include Bitstream"* y clic en *"OK", como se observa en la Figura 63.*

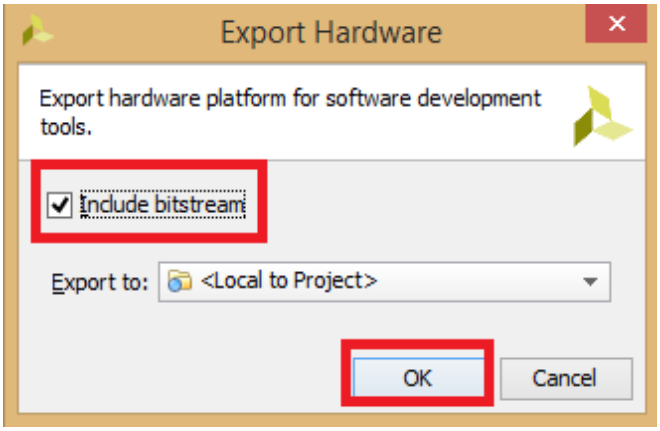

Figura 63.Export Hardware-Include Bitstream.

- Luego hacer clic en *"File"* y clic en *"Launch SDK".* En la ventana que se abre hacer clic en *"OK" como se observa en la Figura 64.*

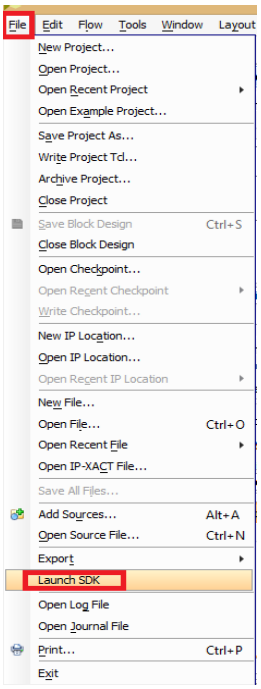

Figura 64.Ruta Launch SDK.

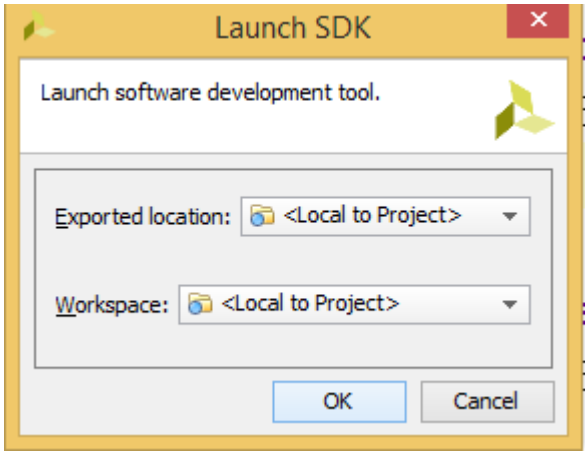

Figura 65.Launch SDK.

27. Al abrir el software "SDK" se debe crear una nueva aplicación, para esto seguir los pasos:

- Clic en *"File"*, luego en *"New"* y hacer clic en *"Application Project"* *como se muestra en la Figura 66.*

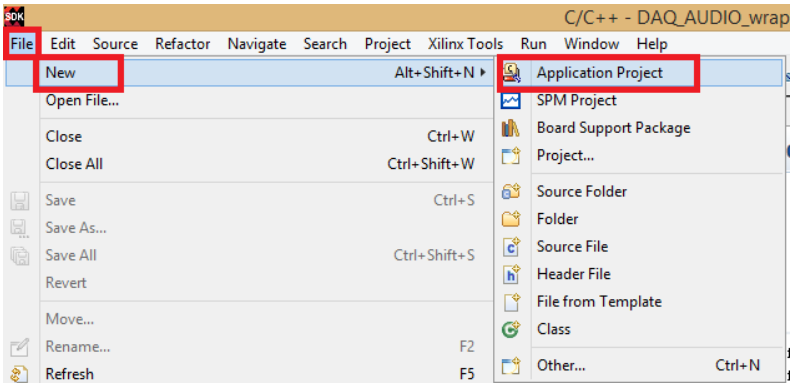

Figura 66.Ruta Application Project.

- En la ventana emergente darle nombre al proyecto, verificar que las opciones marcadas sean las correctas y hacer clic en *"Next"*.

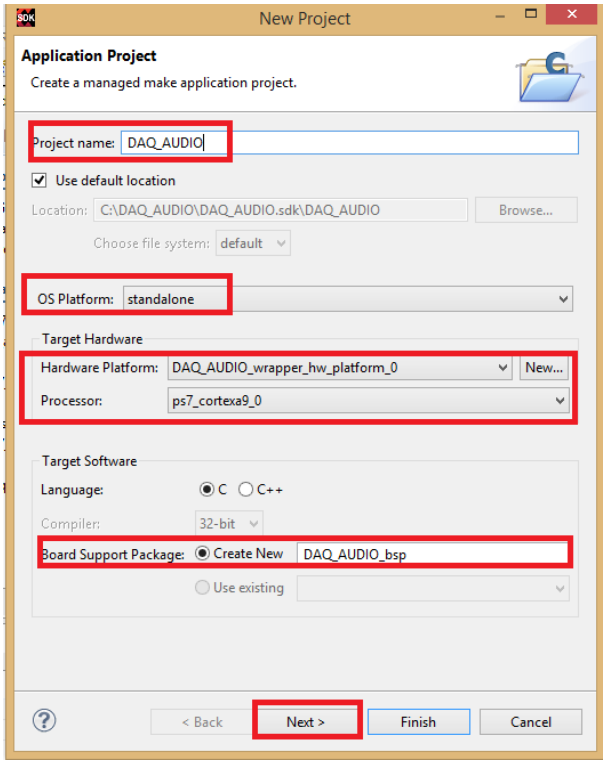

Figura 67.Opciones al Crear el Proyecto.

- En la ventana seleccionar *"Hello World"* y clic en *"Finish",* como lo muestra la Figura 68.

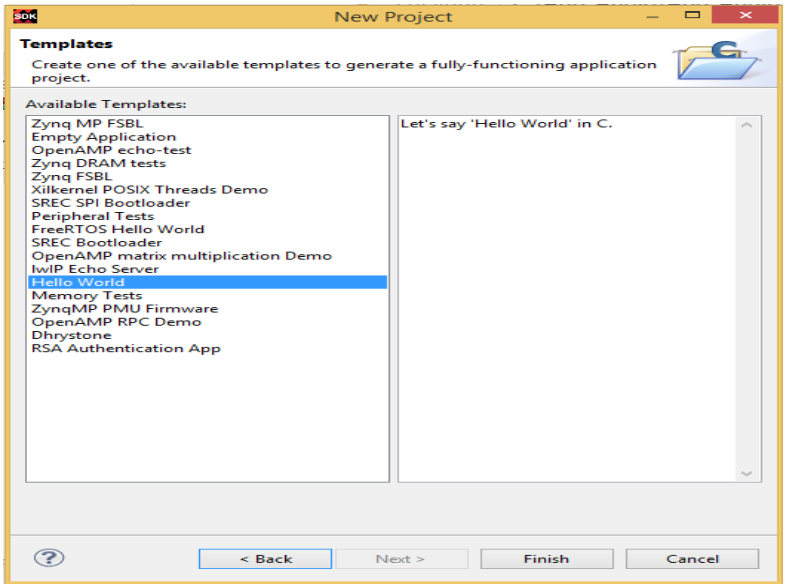

Figura 68. New Project - Templates.

- Crear el código en lenguaje C. Ubicar la pestaña *"Project Explorer"*, expandir la carpeta *"DAQ_AUDIO"*, hacer clic derecho en la carpeta *"src"*, desplegar *"New"* y hacer clic en *"Source File", como se observa en la Figura 69.*

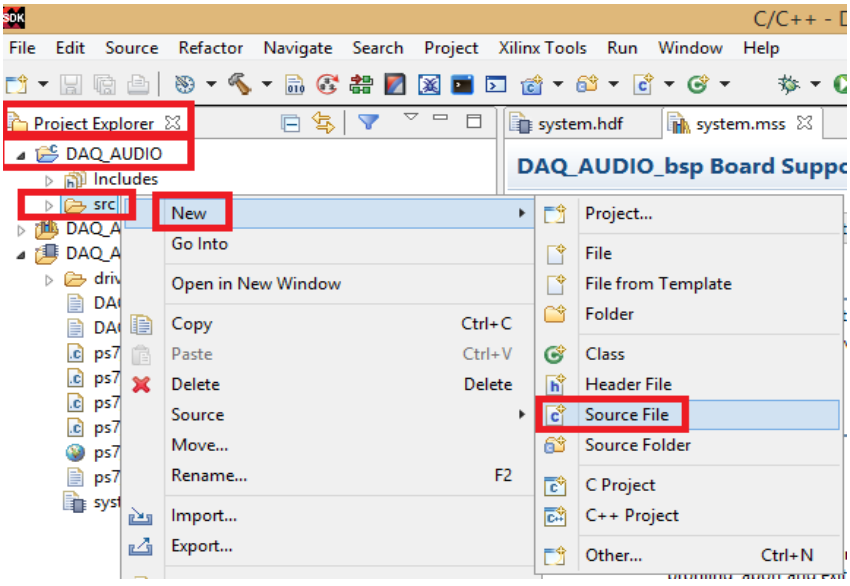

Figura 69.Ruta Source File.

- En la ventana emergente, darle nombre al archivo como se muestra en la Figura 70. Clic en *"Finish"*.

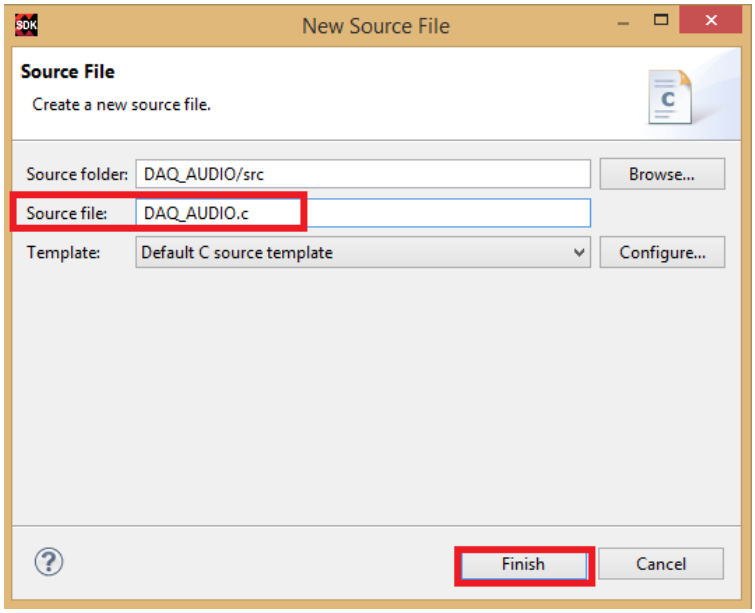

Figura 70.New Source File.

- Crear un archivo .h incluido en esta guía con las modificaciones necesarias para su funcionamiento o el código primario descargable en el link audio.h .Ubicar la pestaña *"Project Explorer"*, expandir la carpeta *"DAQ_AUDIO"*, hacer clic derecho en la carpeta *"src"*, desplegar *"New"* y hacer clic en *"Header File"*.

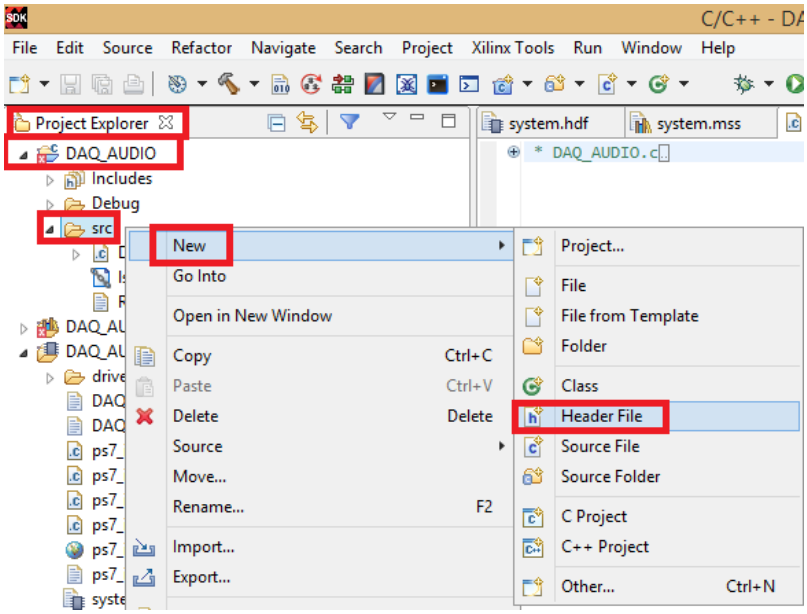

Figura 71. Ruta Header File.

- En la ventana emergente, darle nombre al archivo como se muestra en la Figura 72. Hacer clic en *"Finish"*.

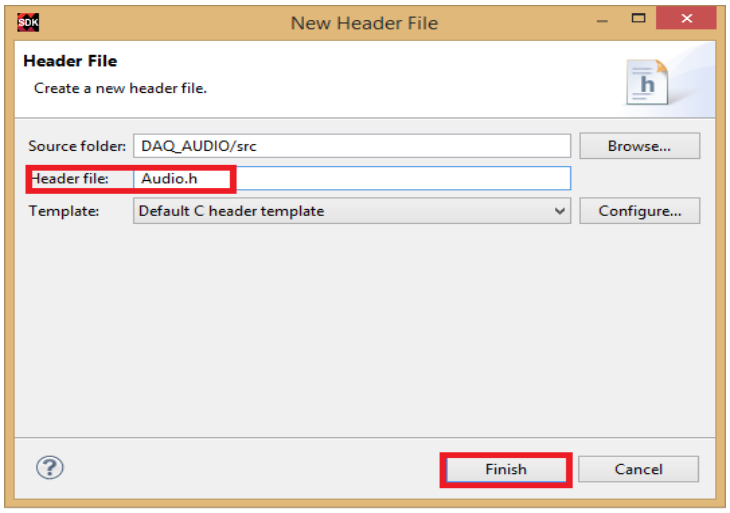

Figura 72.Header File.

- Pegar el código en C para el archivo "DAQ_AUDIO.c". Para esto ubicar la pestaña *"Project Explorer"*, expandir la carpeta

"DAQ_AUDIO", expandir la carpeta *"src"* y hacer doble clic en *"DAQ_AUDIO.c".*

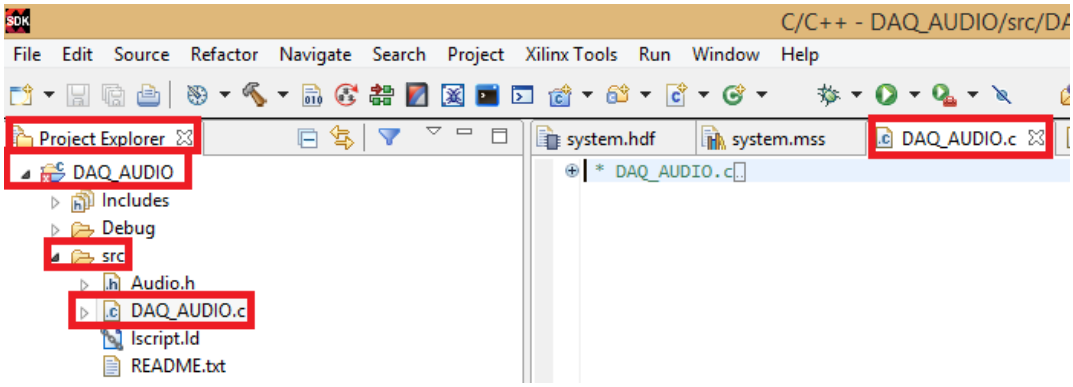

Figura 73.Ruta DAQ_AUDIO.C

- Ahora copiar el código del apéndice A de este trabajo y pegarlo en la ventana que se abre .Clic en *"Save".*

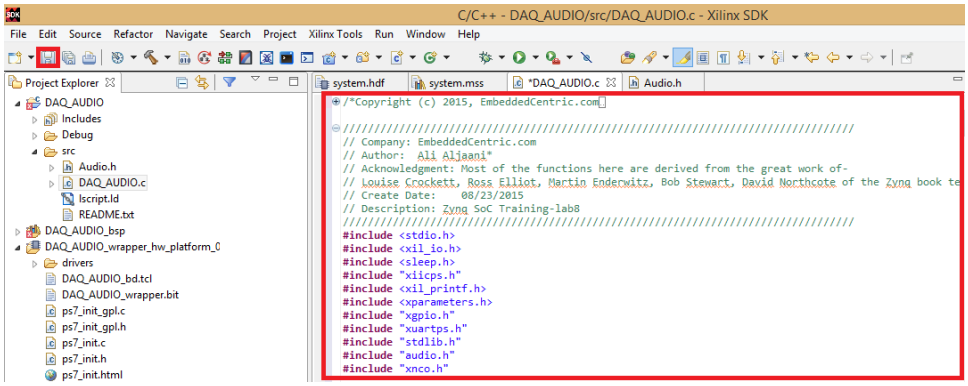

Figura 74.Código de DAQ_AUDIO.C

- De manera similar se copia el código para el archivo "Audio.h" que se encuentra en el apéndice B de este trabajo de grado.

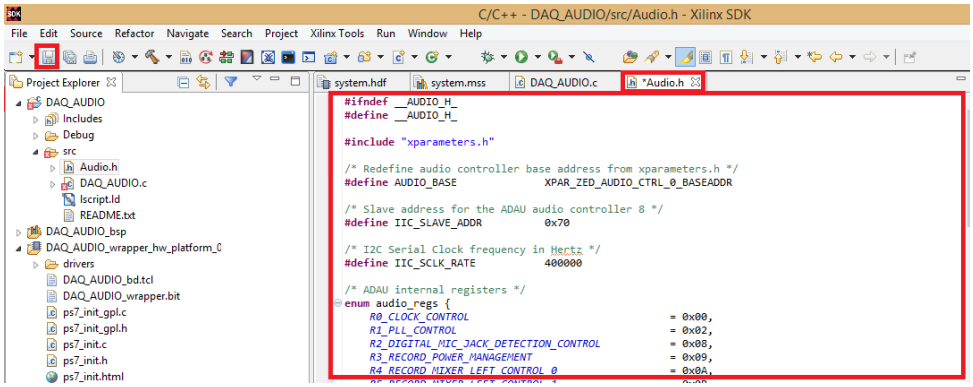

Figura 75.Código de Audio.h.

28. Después de todos estos procesos, el proyecto está listo para programarse en la tarjeta ZedBoard y compilar la aplicación. Para esto seguir los pasos:

- Se requieren dos cables USB con un conector USB A y la otro micro USB como lo muestra la Figura 76.

Figura 76. Conector MicroUsb.

Tomado de: https://www.ecoluzled.com/telefonia/909-cable-usb-con-conector-micro-usb-forever-negro.html.

- Se conectan las dos puntas mini USB a los puestos “UART” y “PROG” de la tarjeta ZedBoard y las dos puntas USB se conectan al computador. También se debe conectar la tarjeta SD al puerto correspondiente. Se conecta el cable de alimentación de la ZedBoard y se enciende por medio del interruptor “SW8” como se observa en la Figura 77.

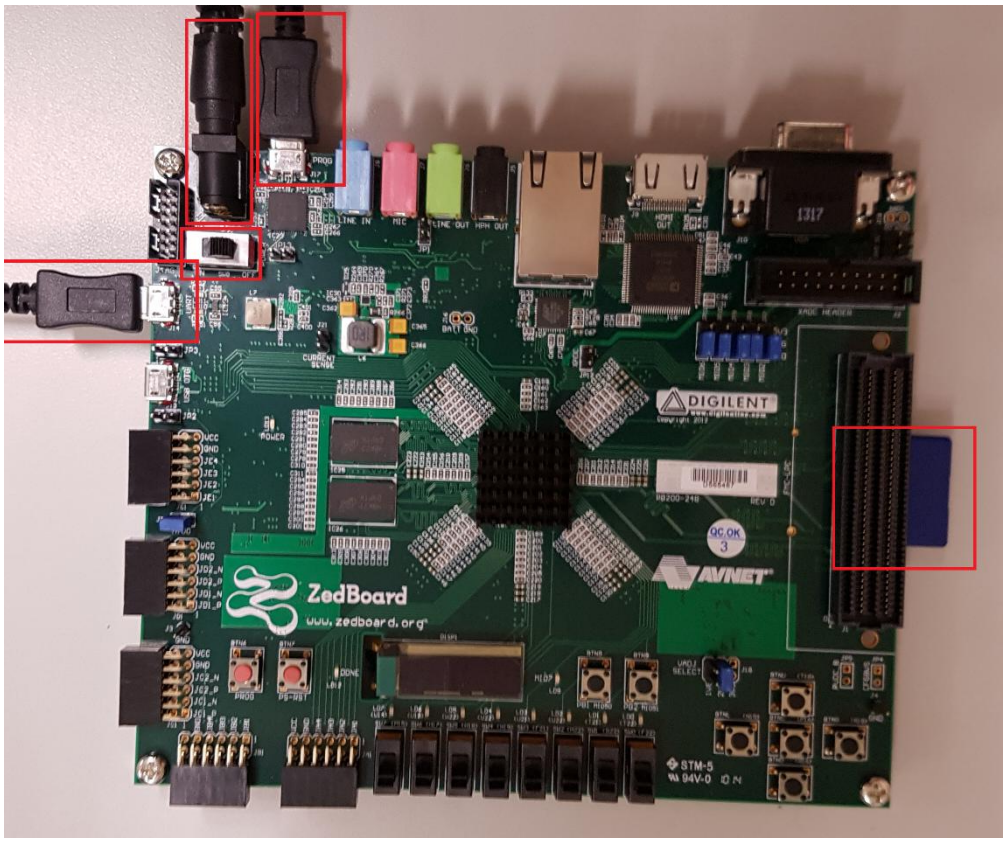

Figura 77.Conexiones Adecuadas para Programar la Tarjeta.

- Luego de tener la tarjeta encendida verificar que el puerto COM está habilitado para la conexión. Para esto ir al administrador de dispositivos del sistema operativo y desplegar el menú “Puertos (COM y LTP)”.Verificar los dos puertos como se observa en la Figura 78.

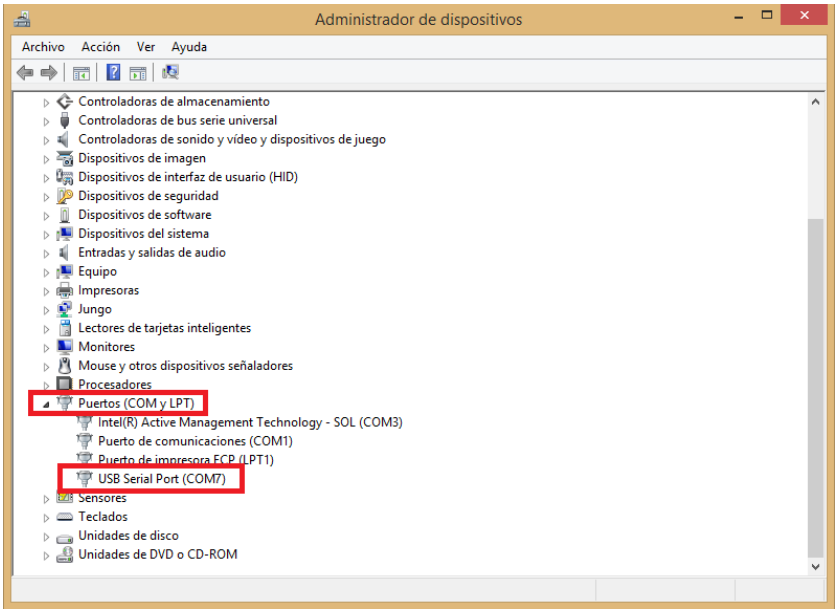

Figura 78.Administrador de Dispositivos.

- Identificado el puerto se procede a programar la tarjeta ZedBoard. Clic en *"Xilinx Tools"* y luego en *"Program FPGA"* como se observa en la Figura 79.

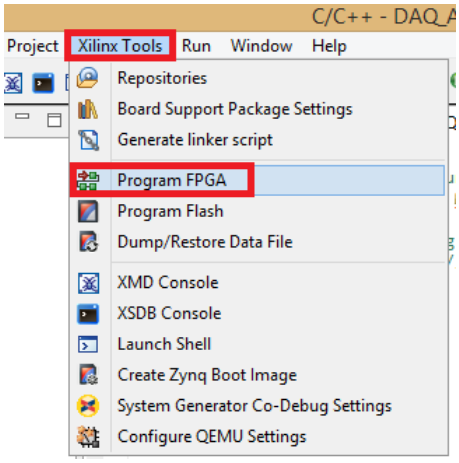

Figura 79.Ruta Para Programar la FPGA.

- En la ventana emergente asegurarse que estén seleccionado los parámetros correctos y hacer clic en *"Program"*.Con este

procedimiento y observando el *Led* azul (LD12) (continuo a la pantalla OLED) que se ilumina en la tarjeta, queda esta ya programada.

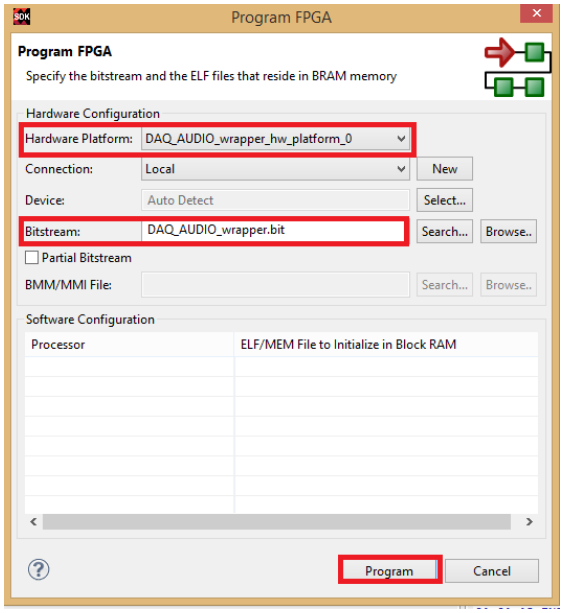

Figura 80.Ventana Emergente- Program FPGA.

- Programada la tarjeta se debe abrir un visor *"terminal"* y conectarlo a la tarjeta para observar los procesos que realiza el algoritmo de programación. Para esto hacer clic en el menú *"Window",* luego clic en *"Show View", Clic en "Other",* como se observa en la Figura 81.

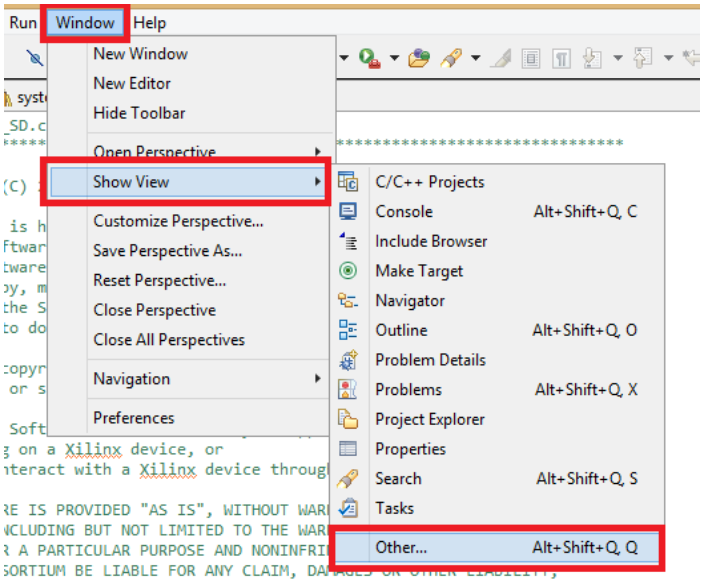

Figura 81.Crear el Terminal.

- En la ventana que se abre, desplegar el menú *"Terminal"* y seleccionar la opción *"Terminal"* y hacer clic en *"OK"*.

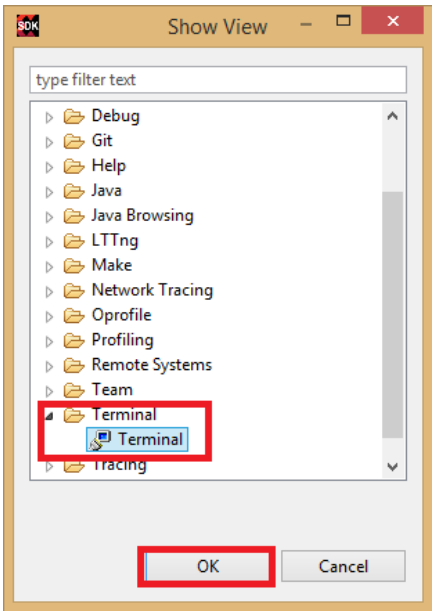

Figura 82.Desplegable Creación Terminal.

- En la ventana que se abre, hacer clic en el botón *"Settings"*.

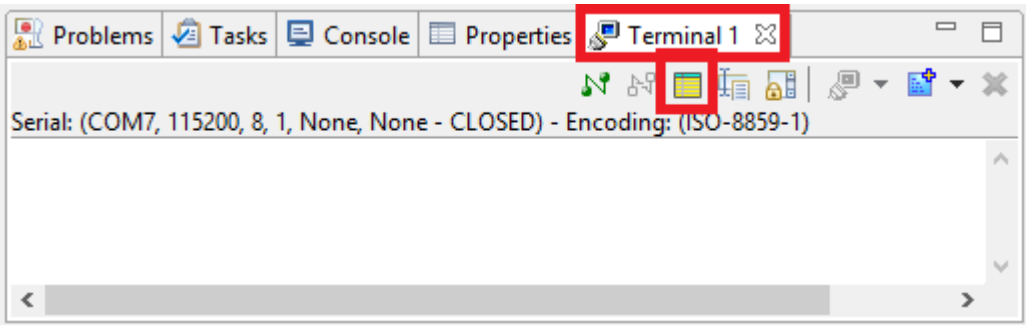

Figura 83.Recuadro Con el Terminal

- Seleccionar en *"Connection Type"* la opción *"Serial"*.En el campo *"Port"* seleccionar el puerto COM que está asignado a la tarjeta, que en este caso es *"COM7"*. En el campo *"Baut Rate"* seleccionar *"115200"* y finalmente hacer clic en *"OK"*. Con esta acción el terminal está conectado.

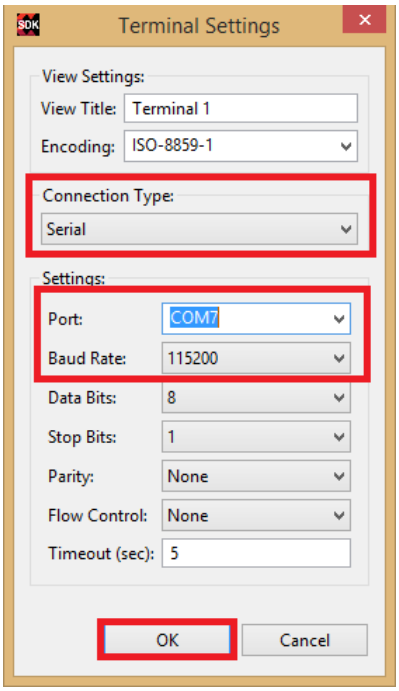

Figura 84.Terminal Settings.

- Para correr el algoritmo se dirige a la pestaña *"Project Explorer"*, clic derecho en *"DAQ_AUDIO"*, luego en *"Run As"* y finalmente clic en *"Launch on Hardware (GDB)"*. Al realizar este procedimiento se envía el código a la ZedBoard para que se ejecute.

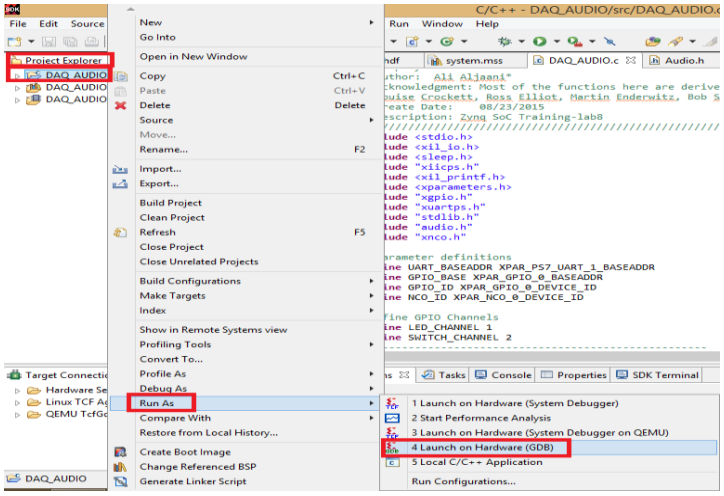

Figura 85.Ruta Launch on Hardware (GDB).

29. El código se está ejecutando y se puede observar el proceso en el Terminal. Para esto se debe dar clic en la pestaña *"Terminal 1"*.Cuando se activan los interruptores los *LED'S* de la tarjeta ZedBoard se encienden. En el *"terminal 1"* se observa que cada vez que cambia un interruptor se imprime *"Step = %d, nco_in = %d", donde "%d"* es un número como se observa en la Figura 86, esto significa que el proyecto está bien realizado. Conectando los audífonos a la línea de salida de audio (Jack Verde o Negro), se deben escuchar tonos de diferentes frecuencias al accionar los interruptores de la tarjeta ZedBoard.

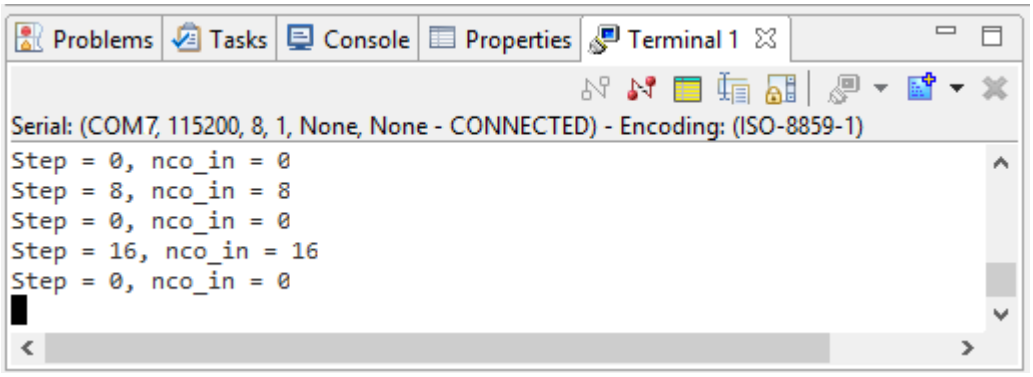

Figura 86.Visualización en el Terminal.

3.2 ALMACENAMIENTO DE AUDIO EN FORMATO DE TEXTO A TRAVÉS DE LA ZEDBOARD

Este procedimiento describe como se debe almacenar el audio adquirido por medio de la tarjeta ZedBoard. Para esto se hace uso del proyecto creado anteriormente para adquisición de audio en el Vivado. Seguir los pasos:

1. En la pantalla principal del proyecto en el software Vivado se abre el bloque diseño. Ubicar la pestaña *"Flow Navigator",* ubicar el menú *"IP Integrator"* y hacer doble clic en *"Open Block Design".*

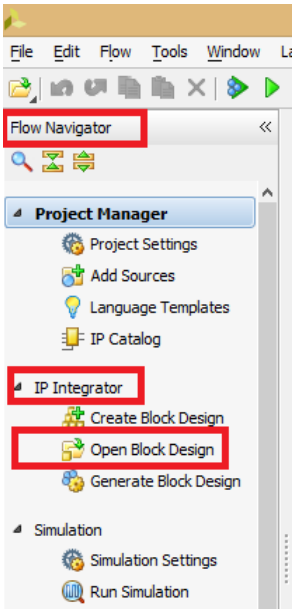

Figura 87.Ruta Open Block Design.

2. Dar doble clic al bloque *"ZYNQ7 proccesing system"* para abrir la configuración de dicho elemento.

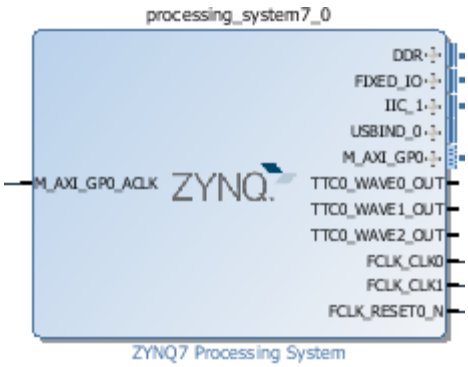

Figura 88.ZYNQ7 proccesing system.

3. Desplegar la pestaña *"MIO Configuration"* y luego abrir el menú *"I/O Peripherals"*. En esta pantalla realizar las siguientes modificaciones:

- En *"Bank 1 I/O Voltage"* seleccionar *"LVCMOS 1.8V".*
- Habilitar la casilla de *"SD 0".*
- Habilitar la casilla de *"CD"* y poner la opción *"MIO 47".*

- Habilitar la casilla de *"WP"* y poner la opción "MIO 46".
- Desde el MIO 40 al 45 en la opción *"IO type"* poner *"LVCMOS 1.8V",* en la opción *"Speed"* poner *"fast"* y en la opción *"Pullup"* poner *"disabled".*
- Habilitar la casilla *"UART 1".*
- Las demás configuraciones se dejan por defecto. Finalmente, clic en *ok*.

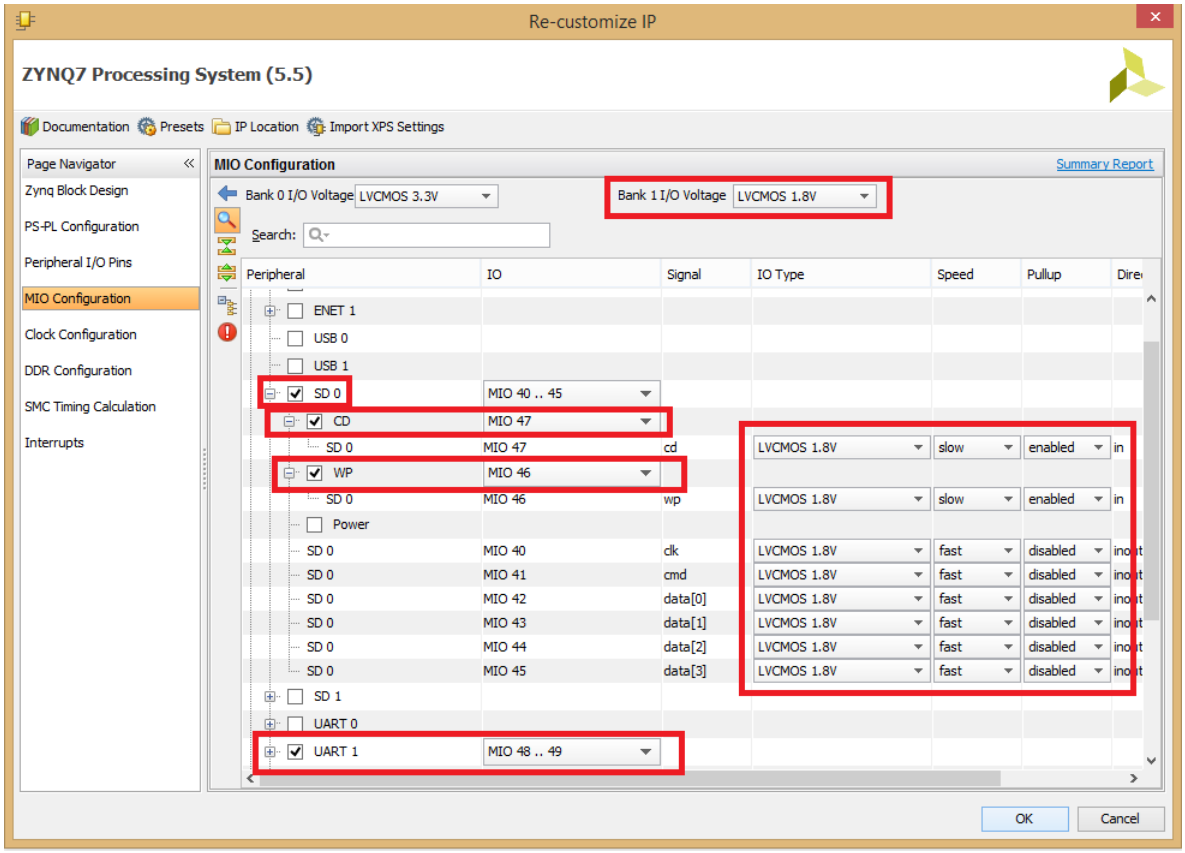

Figura 89.Re-Customize IP.

4. Se debe actualizar el "Bitstream". Para eso repetir los pasos "27" y "28" de la sección0.
5. Con el proyecto ya en el SDK, se debe configurar el SDK de manera que realice el almacenamiento de datos en la tarjeta SD. Seguir los pasos:

- Ubicarse en la pestaña *"Project Explorer",* desplegar el menú *"DAQ_AUDIO",* desplegar la carpeta *"src"* y hacer doble clic sobre

"DAQ_AUDIO.c" para abrir el programa en lenguaje C. Agregar las librerías necesarias para el almacenamiento de datos en la tarjeta SD las cuales son *"xsdps.h", "sd_card/ff.h" y "xil_cache.h".*

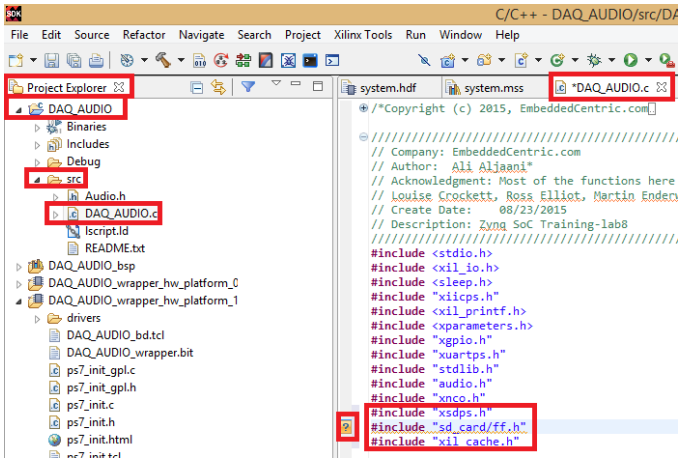

Figura 90.Ruta para comenzar a agregar las librerías.

- En la Figura 90, se muestra que hay un error en el código. Esto se debe a que la librería hay un error en el código, debido a que la librería *"sd_card/ff.h"* no está incluida en el compilador. Por este motivo se debe copiar la carpeta llamada *"sd_card"* en la carpeta *"scr",* que se incluye en este trabajo de grado. Lo primero es buscar la carpeta *"sd_card"* y darle clic derecho-copiar, luego se dirige a la pestaña *"Project Explorer"* desplegar el menú *"DAQ_AUDIO",* hacer clic derecho en *"src"* y luego hacer clic en *"Paste".* Al hacer esto debe aparecer como la Figura 91.

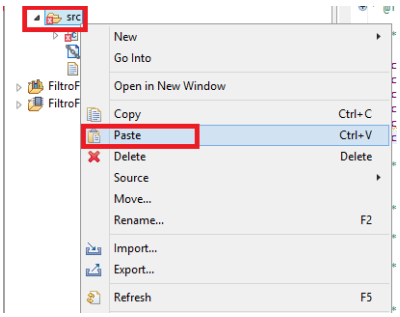

Figura 91.Ruta SCR para pegar la librería.

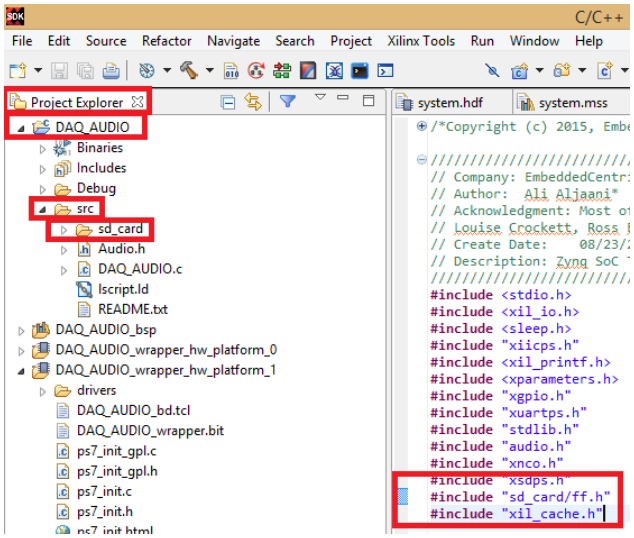

Figura 92.Librerías sin error.

- En la Figura 92 muestra que el error desaparece y la librería es reconocida por el compilador. Luego se abre el archivo *"DAQ_AUDIO.c"* y se agrega el código necesario para que el audio adquirido por la tarjeta ZedBoard se almacene en la tarjeta SD, como se describe a continuación (código completo en el apéndice C):
  - ✓ Se agregan las variables y definiciones necesarias para almacenar datos en la tarjeta SD.

```
#include "xsdps.h"
#include "sd_card/ff.h"
#include "xil_cache.h"

static FIL fil1n;
static FATFS fatfs;
static char FileNamein[32] = "audioin.txt";
static char *SD_File;
int n,len;
long int accum=0;
double samplein;

#ifdef __ICCARM__
#pragma data_alignment = 32
u8 DestinationAddress[13];
#pragma data_alignment = 4
#else
u8 DestinationAddress[13] __attribute__ ((aligned(32)));
#endif

// Parameter definitions
#define UART_BASEADDR XPAR_PS7_UART_1_BASEADDR
#define GPIO_BASE XPAR_GPIO_0_BASEADDR
#define GPIO_ID XPAR_GPIO_0_DEVICE_ID
#define NCO_ID XPAR_NCO_0_DEVICE_ID
```

Figura 93.Variables para almacenar en la SD.

- ✓ Estas líneas hacen que la tarjeta SD sea reconocida por la ZedBoard y crea el archivo *"audioin.txt"* donde se almacenan los datos.

```
void read_superpose_play(void)
{
    FRESULT Res;
    UINT NumBytesWritten;

    Res = f_mount(&fatfs,"",1);
    if (Res != FR_OK) {
        return XST_FAILURE;
    }

    SD_File = (char *)FileNamein;
    Res = f_open(&filin, SD_File, FA_CREATE_ALWAYS | FA_WRITE );
    if (Res) {
        return XST_FAILURE;
    }

        u32 nco_in, nco_out, in_left, in_right, out_left, out_right,step, temp;
        // step is associated with the frequency of the sin wave
        /* Read step size value from DIP switches */
        step = XGpio_DiscreteRead(&Gpio, SWITCH_CHANNEL);
```

Figura 94. Líneas de Código para que la SD sea reconocida por la ZedBoard

- ✓ Las líneas que graban las muestras de audio cuando el interruptor 1 se acciona. El comando *f_mount* registra / anula el archivo de objeto del sistema para el módulo FATFS.

```
if (step == 1){

/* Sample L+R audio from the codec */
in_left = Xil_In32(I2S_DATA_RX_L_REG);
in_right = Xil_In32(I2S_DATA_RX_R_REG);

Res = f_lseek(&filin, accum);
if (Res) {
    return XST_FAILURE;
}

n=sprintf(DestinationAddress,"%d\n",in_right);

Res = f_write(&filin, (const void*)DestinationAddress,n,&NumBytesWritten);
if (Res) {
    return XST_FAILURE;
}

len = strlen(DestinationAddress);
accum=accum+len;

/* Add scaled sin wave component to the L+R audio samples */
out_left =  temp + in_left;
out_right = temp + in_right;

/* Output corrupted audio to the codec */
Xil_Out32(I2S_DATA_TX_L_REG, out_left);
Xil_Out32(I2S_DATA_TX_R_REG, out_right);
}
/* If the DIP switch values have changed, break from while*/
if(step != XGpio_DiscreteRead(&Gpio, SWITCH_CHANNEL)) break;
}
```

Figura 95. Lineas de Codigo para control de almacenamiento de datos.

- ✓ Para que los datos queden grabados en la tarjeta SD se debe cerrar el archivo "audioin.txt". La función *f_lseek* mueve el puntero del archivo de

Lectura/escritura de un objeto fichero abierto. También se puede utilizar para expandir el tamaño del archivo (pre-asignación de grupos).

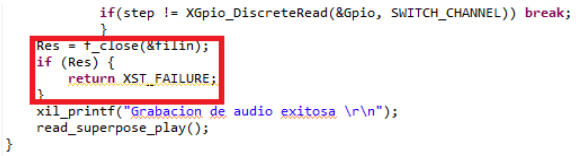

Figura 96. Líneas de Código para cerrar el archivo de almacenamiento

✓ En el terminal 1 se observa que el proceso de grabación fue exitoso cuando sale el mensaje “Grabación de audio exitosa”.

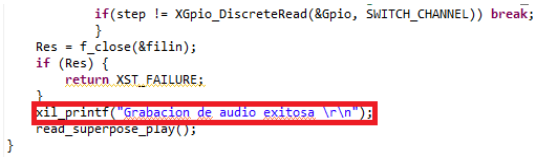

Figura 97.Grabación Exitosa

➢ Se debe programar la tarjeta ZedBoard y ejecutar el código. Para esto seguir el paso “28” de la sección 3.10.

6. Para comprobar el funcionamiento del algoritmo seguir los pasos:

➢ Luego de programar la tarjeta ZedBoard y ejecutar el código, clic en la pestaña *“terminal 1”*para observar los procesos que se están realizando.

Figura 98. Terminal

➢ Para que el audio se almacene, se debe activar el interruptor “sw0” de la ZedBoard y para detener el almacenamiento del audio se debe desactivar este interruptor. Cuando en el terminal se despliegue el mensaje “Grabación de audio exitosa” significa que el audio ha sido guardado en la tarjeta SD. El nombre del archivo generado es “audioin.txt”.

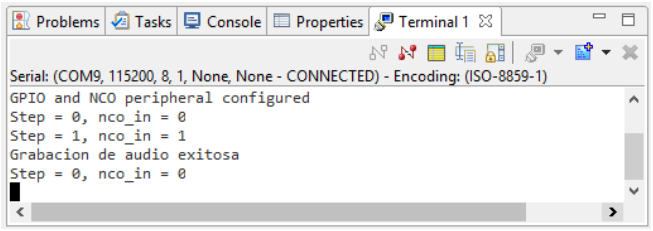

Figura 99.Grabación Exitosa del Audio.

- Para verificar si el archivo se creó y almacenó correctamente seguir los pasos:
  - ✓ Se introduce la tarjeta SD en un computador.
  - ✓ Se abre el archivo "audioin.txt" y se verifica que tenga los datos de audio que se grabaron anteriormente.

3.3CALCULO DE LA TRANSFORMADA RAPIDA DE FOURIER.

El procesamiento aplicado al audio es una transformada rápida de Fourier. Para este paso se puede continuar utilizando el proyecto de almacenamiento en SD y adecuarlo para además de almacenarlo realice las operaciones para hallar la transformada rápida de Fourier. A continuación, se describe cómo llevarlo a cabo:

- Se debe asegurar que en la tarjeta SD de la ZedBoard este almacenado el archivo de audio original "audioin.txt" en caso contrario se guarda en la tarjeta SD dicho archivo. Este procedimiento se debe hacer por medio del computador.
- Como se implementa el código en el proyecto ya creado del almacenamiento en SD a este se le deben hacer pocas modificaciones, como la adición de las librerías y si se desea el nombre del proyecto.
- Realizar todos los pasos mencionados en el proceso de almacenamiento de datos en la SD ya que solo es necesario modificar el archivo en C del SDK el cual ya está funcional la parte de almacenamiento, se adiciona la parte del código C que se realiza la FFT.
- Agregar las librerías *"stdio.h", "math.h", "complex.h".*

```
#include "math.h"
#include "xparameters.h"    /* SDK generated parameters */
#include "xsdps.h"      /* SD device driver */
#include "xil_printf.h"
#include "sd_card/ff.h"
#include "xil_cache.h"
#include <stdio.h>
#include <complex.h>
```

Figura 100.Librerías para anexar al Código.

- Al ingresar estas librerías usualmente la *"math.h"* muestra un error. Es necesario anexar esta librería ya que el código en C ejecuta líneas que realizan operaciones. Para solucionar este error hay que realizar una configuración que ayuda a que la librería *"math.h"* trabaje correctamente y es adicionarla en las propiedades del proyecto:
  - ✓ Clic en *"fft"* así se re nombra el archivo de almacenamiento. Clic en *"Project"* y después en *"Properties"*.

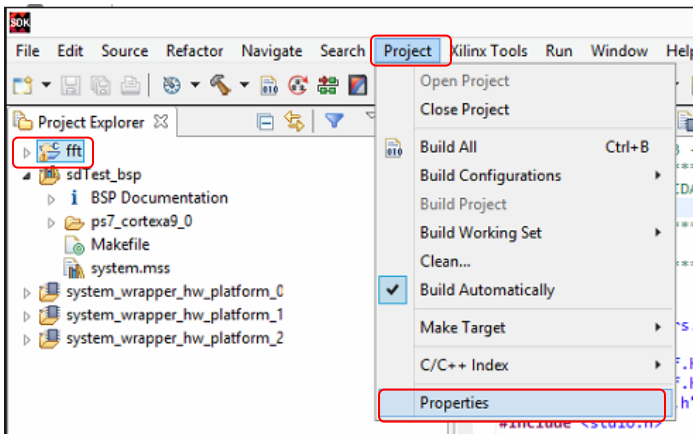

Figura 101.Ruta para Properties.

- ✓ En la ventana que se abre desplegar el menú *"C/C++ Build",* luego Clic en *"Settings"* y en los menús que se despliegan hacer clic en *"Libraries".*

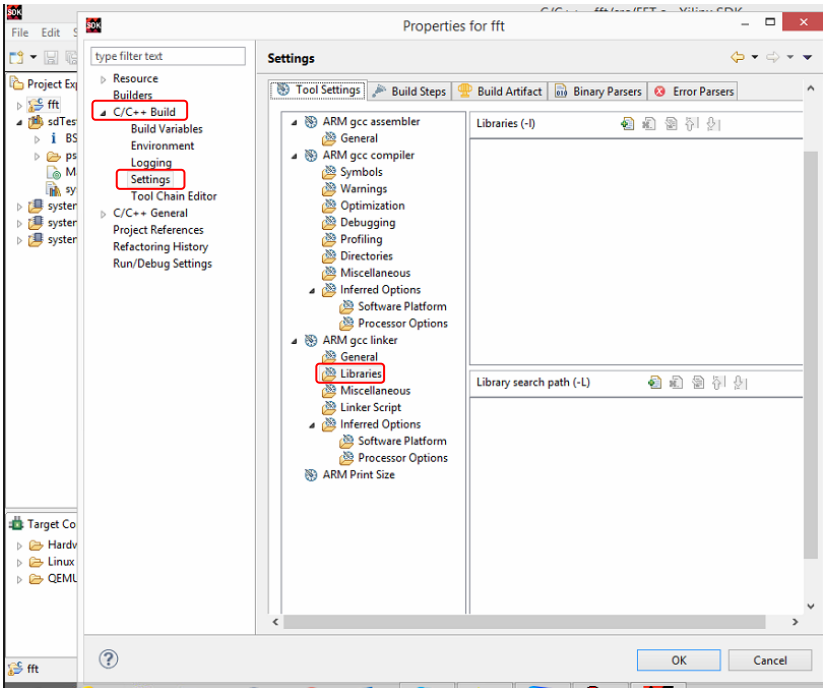

Figura 102. Opciones de Configuración de las propiedades.

- ✓ Hacer clic en el botón agregar que está identificado con el símbolo + de color verde y en la ventana que se abre escribir la letra *"m"* y clic en *"OK".* Con este proceso se corrige el error presentado en la función de la librería *"math.h"* y otra vez hacemos clic en *"OK".*

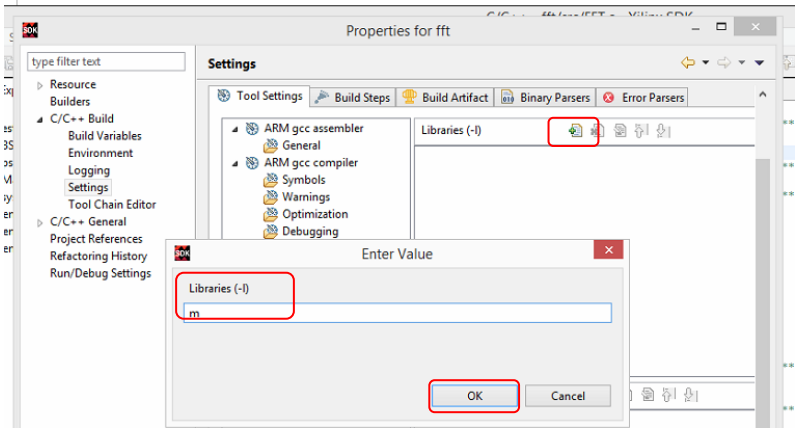

Figura 103.Agregar Librería Math.

- ✓ La longitud del vector de datos de entrada es de 2^n muestras, es decir potencias de dos (2, 4, 8, 16, 32, 64, 128, 256). En este caso se cambia este valor en la variable "A". Para realizar las debidas comparaciones tiene en cuenta que la visualización del terminal de la ZedBoard se satura y no permite desplegar tantos datos.

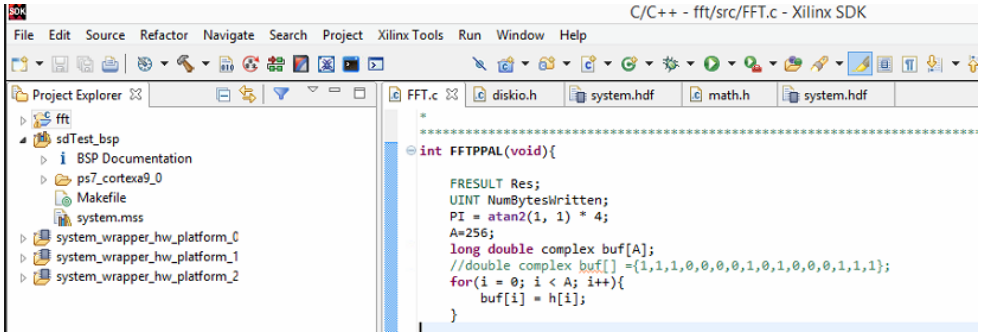

Figura 104.Valores en la Variable A.

- ✓ Se debe adicionar el algoritmo de programación que realiza la FFT. Para esto primero se copia el código que se describe en el apéndice E de este trabajo y se pega en el proyecto ya creado de almacenamiento en la SD y hacer clic en "save".

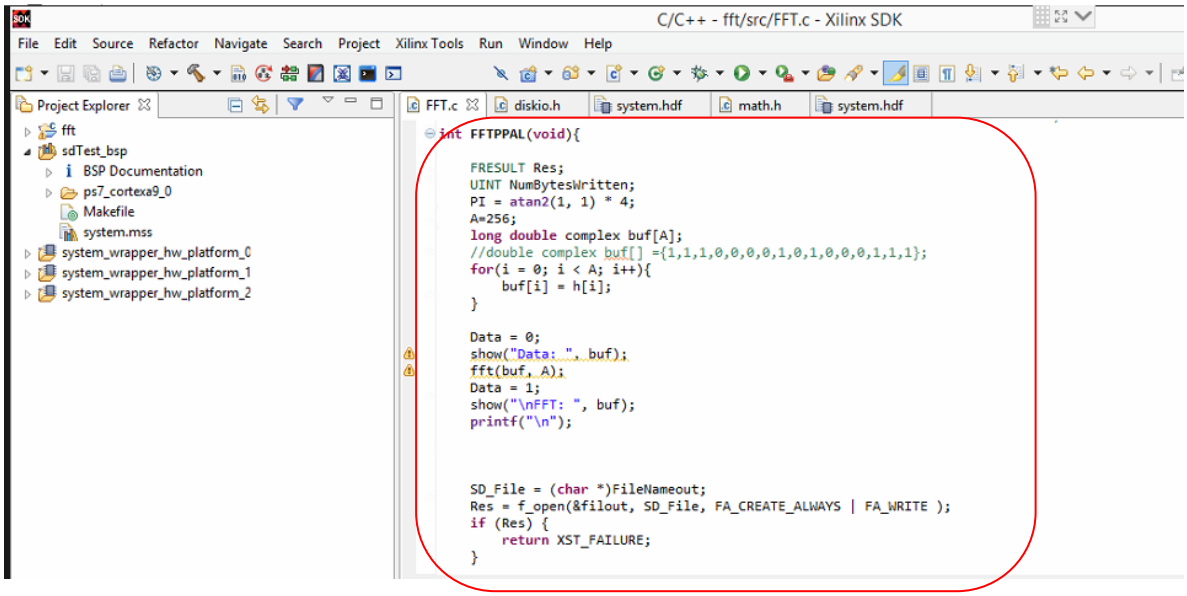

Figura 105. Código C de la FFT

- ✓ Luego de haber corregido el error se observa que ya no existen problemas con el código en C, solo existen *"Warnings"* que se pueden omitir por el momento.

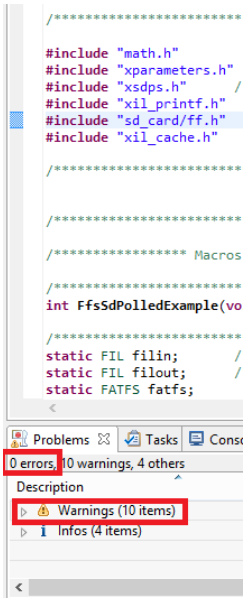

Figura 106. Warnings en el Código

- ✓ Con el código corregido y funcionando se debe programar la tarjeta ZedBoard y correr el código. Para esto repetir el paso de la sección 0 que indica como programar la ZedBoard y tener en cuenta que algunos nombres van a cambiar porque el proyecto se llama de una manera distinta.
- ✓ Para observar que proceso que está realizando el algoritmo de procesamiento de datos se debe dar clic en la pestaña “Terminal 1”.Tener en cuenta que este proceso tarda varios segundos. Cuando se despliega el mensaje “Procesamiento FFT exitoso” es porque el algoritmo se ejecutó con éxito. En la tarjeta SD se cuenta con el archivo en forma de texto con la FFT calculada y se identifica con el nombre “fftout.txt”. Para verificar que este archivo se creó se debe ingresar la tarjeta SD al computador y para su posterior análisis se recomienda guardarlo en el computador.

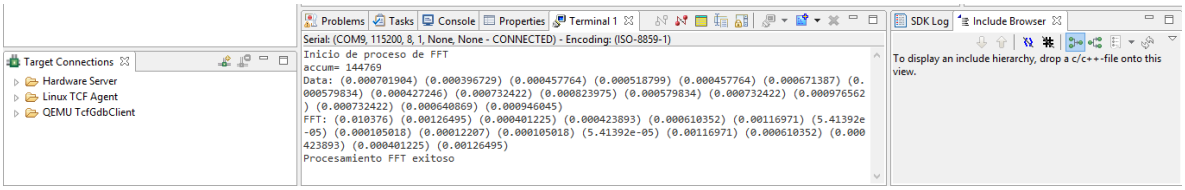

Figura 107.Visualizacion en el Terminal 1 de confirmación exitosa de procedimiento.

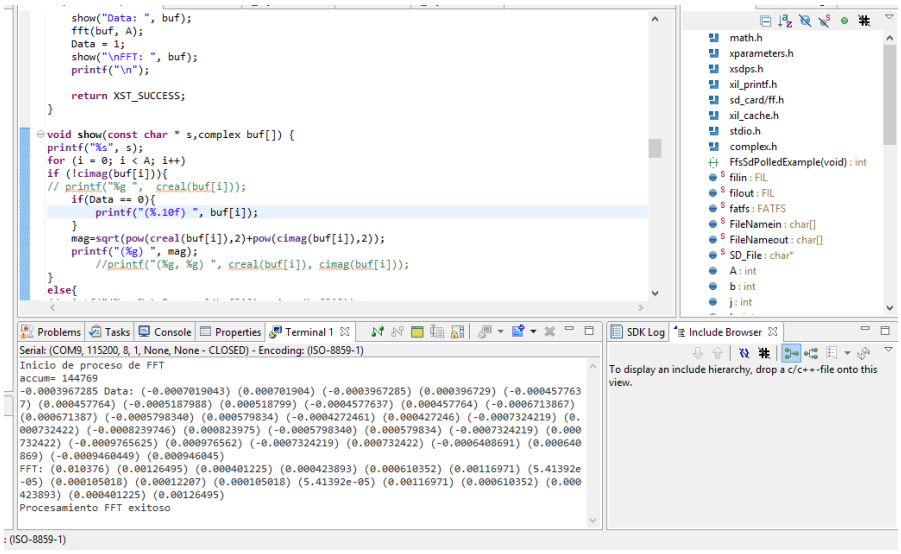

Figura 108.Compilacion con valor diferente de la variable A

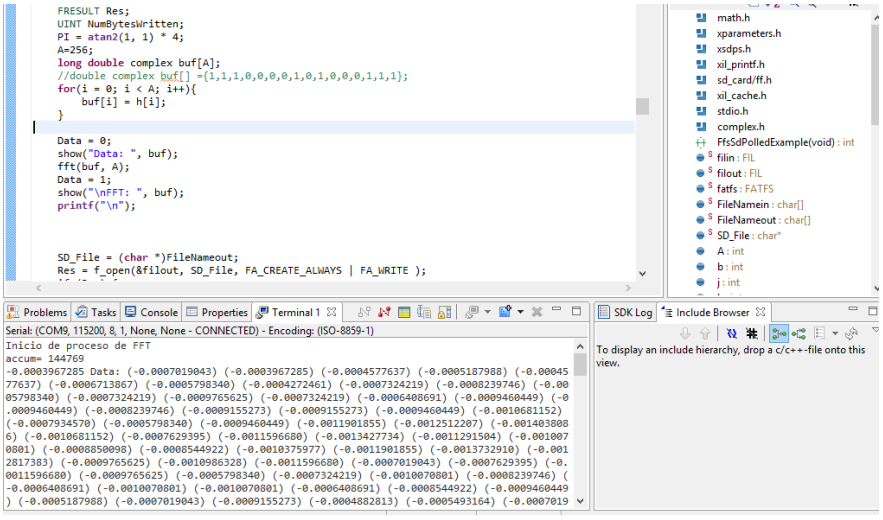

Figura 109. Compilación con la variable A=256

Figura 110. Variable A=256 (Terminal Desplegado)

3.4 PROCESAMIENTO DE AUDIO A TRAVÉS DE MATLAB

Obtenido el resultado del procesamiento en la ZedBoard se debe comparar con MATLAB. Para esto se debe implementar un algoritmo en este entorno que haga el mismo procesamiento que la ZedBoard y usar el mismo archivo de audio original almacenado en formato de texto que se usó en el procesamiento anterior. El nombre dado a este archivo es “audioin.txt”.

Para crear el algoritmo en MATLAB se siguen los pasos:

- Se abre el programa MATLAB.

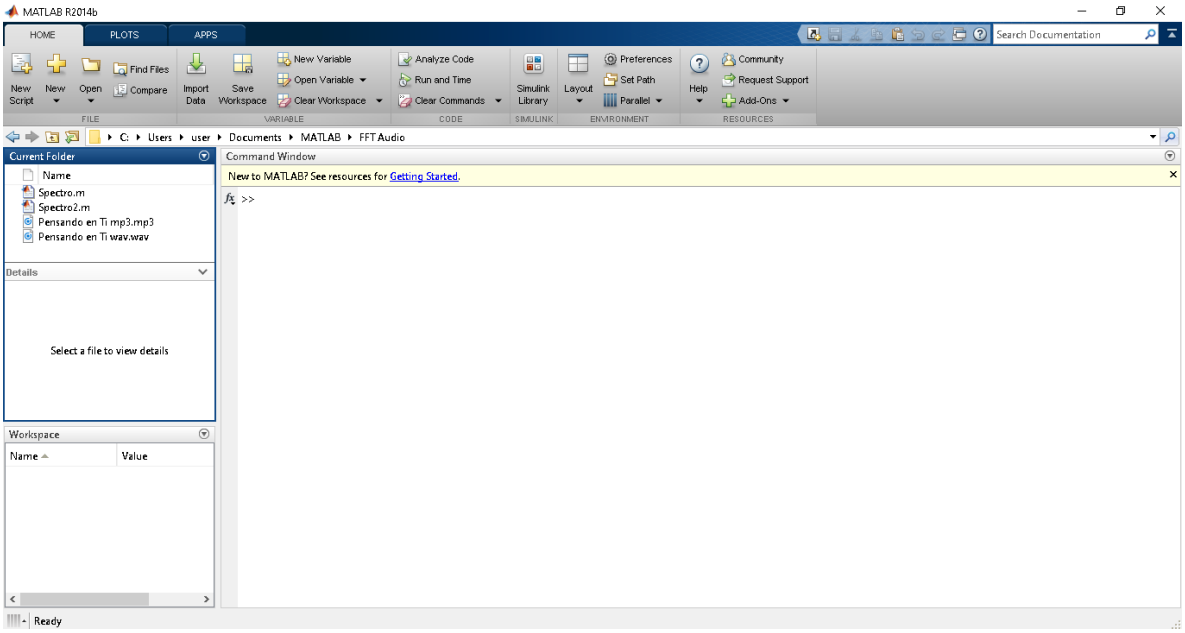

Figura 111.Ventana Principal MATLAB.

- Se recomienda crear una carpeta de trabajo y allí almacenar todos los archivos concernientes al procesamiento de audio de este trabajo. La ruta definida en este caso es "C:\Users\user\Documents\MATLAB\FFTAudio". Se debe garantizar que al ejecutar el MATLAB se esté situado en la ruta definida anteriormente.

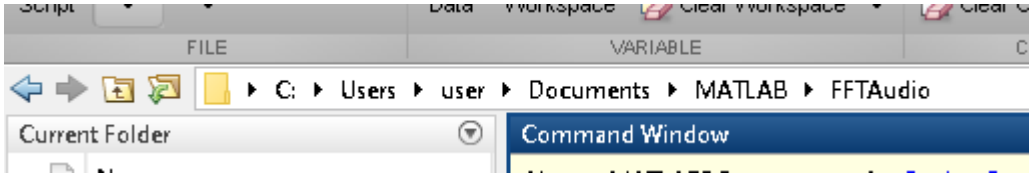

Figura 112.Ruta Definida para MATLAB.

- Para escribir el código que se encarga del procesamiento de audio se recomienda crear un script. En este caso se llamará "FFT_SPECTRO". Clic en *"New Script"* y luego en *"Save"* para guarda el script y ponerle el nombre indicado. Recordar guardarlo en la carpeta que se creó anteriormente.

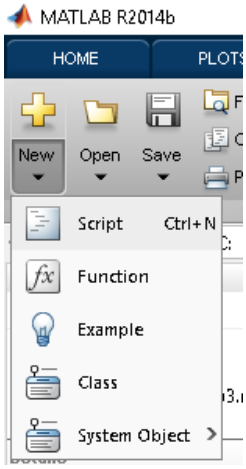

Figura 113.Ruta para Crear un Script.

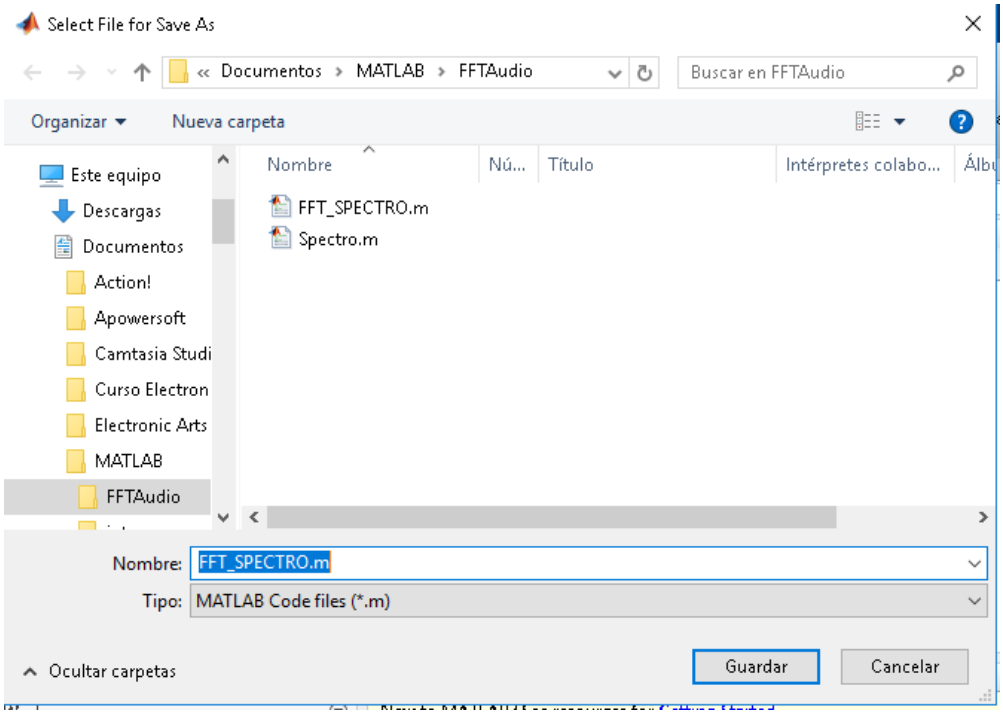

Figura 114.Guardar el Archivo.

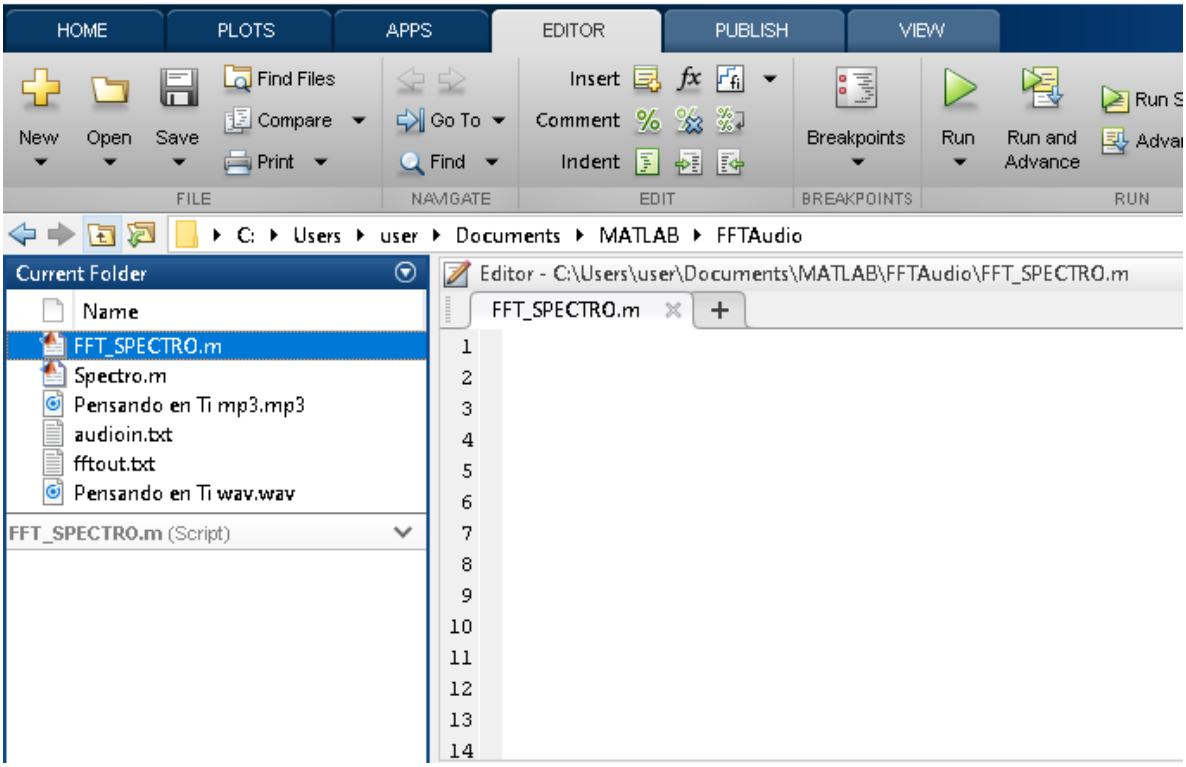

Figura 115.Script ya Creado.

- Se escribe el script para calcular la FFT y consiste en seguir la rutina:
  - ✓ Cargar el audio original en txt a un vector a MATLAB, llamado *"audioin"*. Esto se hace por medio de la instrucción *"import data"*
  - ✓ Existe una función la cual crea el vector a partir del audio *"audioread"*.
  - ✓ Definir el tiempo de muestreo, la longitud de la señal y vector de tiempo.
  - ✓ Aplicar el comando *"fft ()"* y se almacena en el vector "*Y*" que se empleara para graficarlo y también queda almacenado en MATLAB para guardarlo en texto o consultarlo.
  - ✓ Graficar el audio original y el audio con la FFT.

Editor - C:\Users\user\Documents\MATLAB\FFTAudio\FFT_SPECTRO.m

FFT_SPECTRO.m

```
%// Instruccion para cargar archivo de texto del audio original a un vector

audioin= importdata('audioin.txt');

%//Instruccion para leer un audio de entrada y guardarlo en un vector.
[y, Fs] =audioread('Pensando en Ti wav.wav');
%
%y=y.'; % Organiza los datos en vector fila.
T=1/Fs; % Tiempo de muestreo.
L=length(y); % Longitud de la señal.
t=(0:L-1)*T; % Vector de tiempo.
%
NFFT=2^nextpow2(L); % Siguiente potencia de 2 de la longitud de 'y'.
%nextpow2 función para aumentar el rendimiento de fft cuando la longitud de
%una señal no es una potencia de 2.
Y=fft(y); % Aplica Transformada de Fourier.
f=Fs/2*linspace(0,1,NFFT/2); % Vector de frecuencias.
```

Figura 116.Instrucciones necesarias para la FFT.

✓Si se desea escuchar el audio de salida.

```
%// Reproduccion de audio con FFT

sound (fftout_Matlab,Fs);
```

Figura 117.Instrucción para reproducir el audio

✓Para graficar las señales

```
%
% Visualiza la señal de audio y el espectro de dicha señal.
subplot(2,1,1),
plot(t,y), title('Señal de Audio'),...
xlabel('Tiempo (segundos)'), grid on
subplot(2,1,2),
plot(f,2*abs(Y(1:NFFT/2))),...
title('Espectro de la Señal de Audio'),...
xlabel('Frecuencia (Hz)'), ylabel('|Y(f)|'), grid on
```

Figura 118. Instrucciones para Visualización de la Señal

- Ejecutado el script con el botón *"Run"*, ver en la pestaña *"Workspace"* las matrices y variables necesarias para el cálculo. Se crea la gráfica deseada e inmediatamente se escucha el audio filtrado a través de los parlantes del computador.

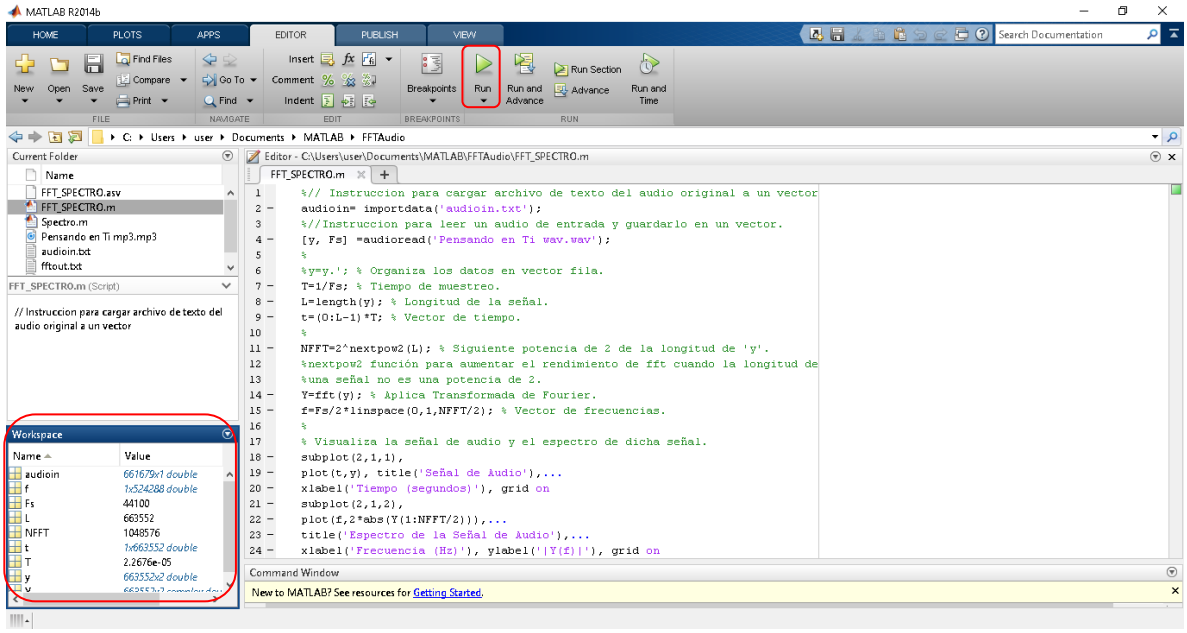

Figura 119. Código de la FFT en MATLAB.

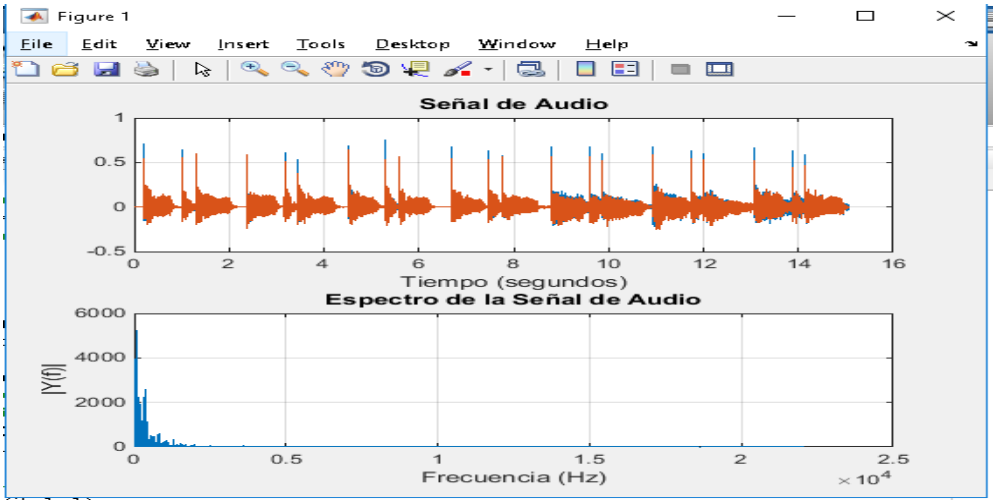

Figura 120.Grafica de la Señal Original y Señal Espectral.

- El tiempo de ejecución del algoritmo para FFT depende de la longitud de la transformada que se realiza. El tiempo es más rápido para potencias de dos y casi tan rápido para longitudes pequeñas que sólo tienen factores primos. Inclusive el procesamiento del equipo influye en este tiempo.
- Se modifica el script, para esto adicionamos la carga del audio procesado por la ZedBoard y lo guardamos en un vector llamado "fftout_Zedboard".

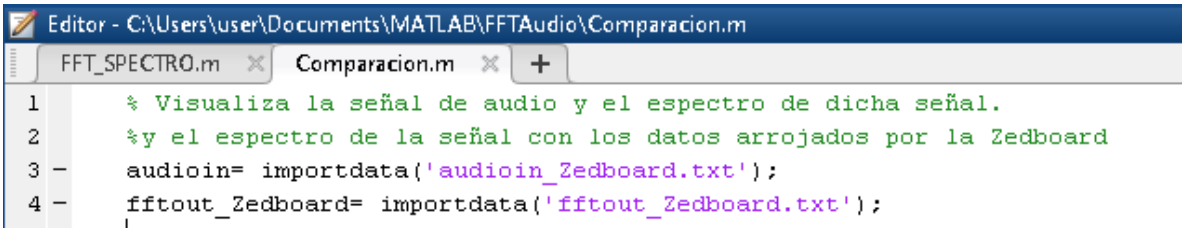

Figura 121.Datos obtenidos en Vivado.

- La parte de procesamiento del audio a través de MATLAB no se modifica. Vamos a la parte de graficas del script y anexamos las líneas necesarias para graficar los datos obtenidos en Vivado.

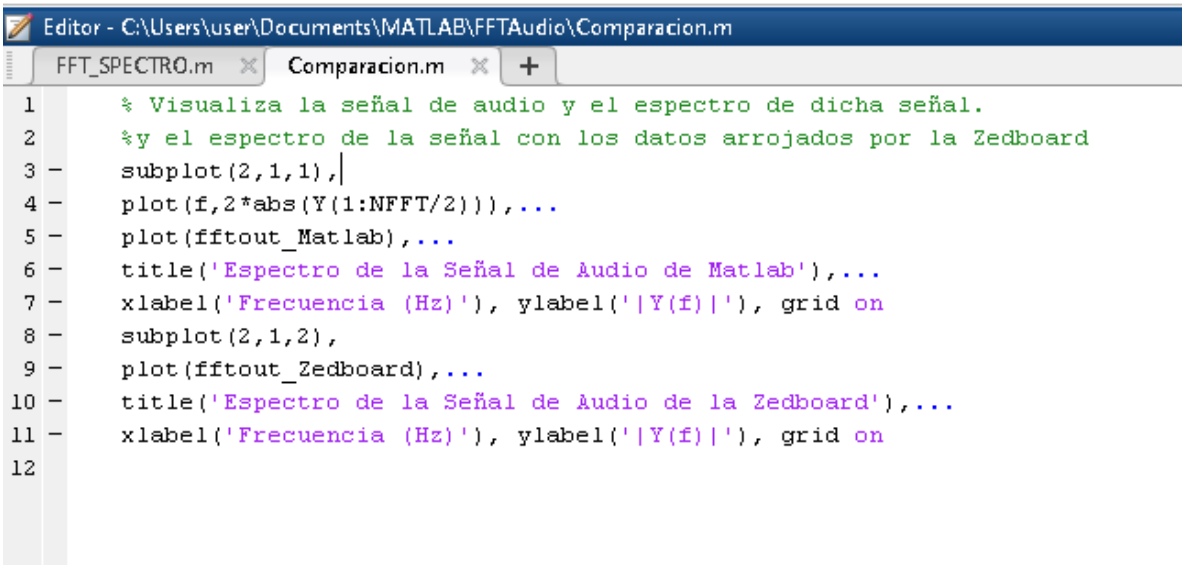

Figura 122. Comparación de los datos de MATLAB y Vivado.

- La parte de escuchar el audio se elimina y luego se compila el script desde el botón *"Run"*. Inmediatamente se observa la gráfica de comparación.

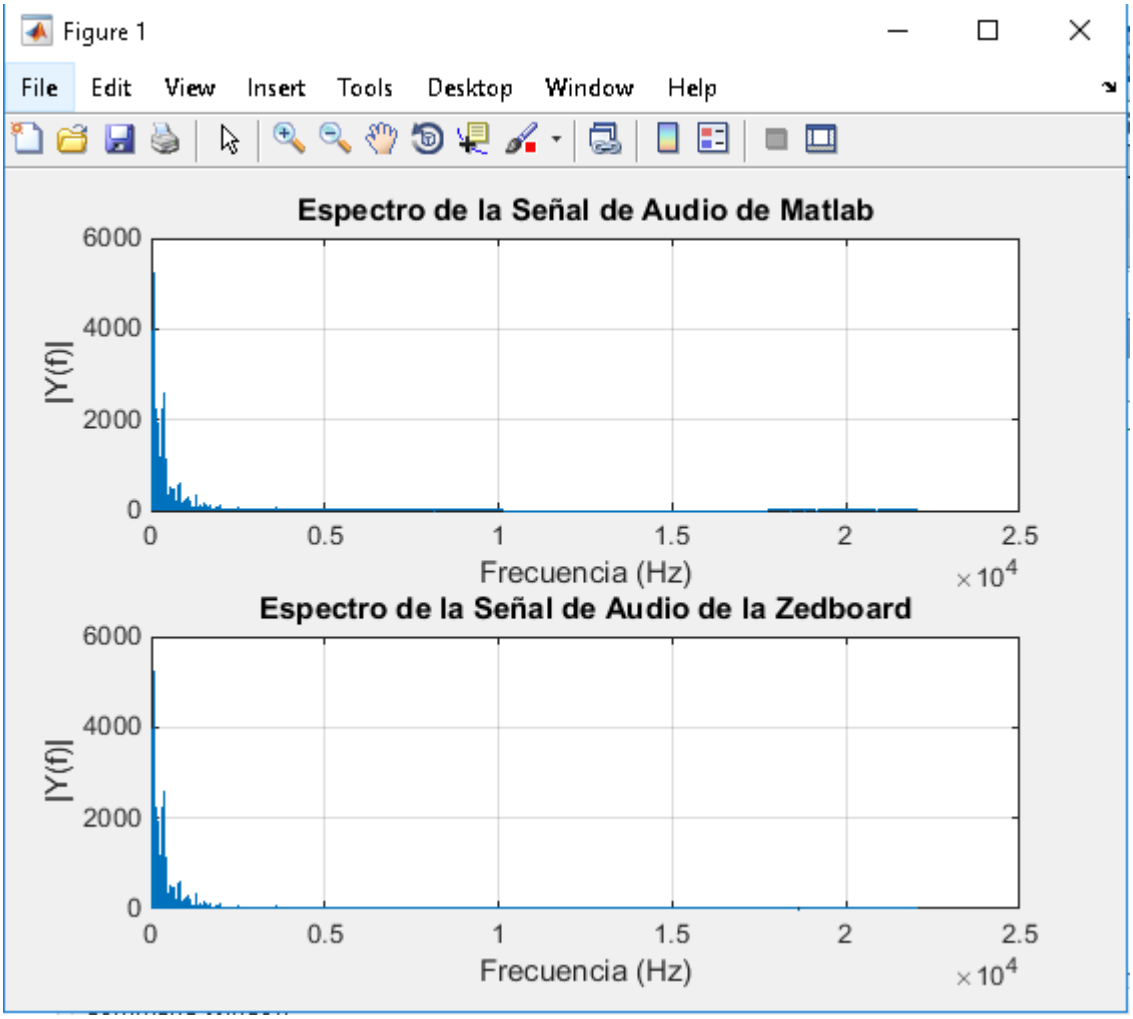

Figura 123.Graficas MATLAB vs ZedBoard.

Editor - Comparacion.m

Comparacion.m | fftout_Zedboard.txt | fftout_MATLAB.txt

```
1 -    y=fftout_MATLAB;
2 -    x=fftout_Zedboard;
3 -    error=y-x;
4 -    subplot(2,1,2);
5 -    plot(error);
6 -    title('Error de la FFT por Matlab y la ZedBoard');
7 -    xlabel('Valores de la Variable A');
8 -    ylabel('Diferencia');
9
10
```

Figura 124. Código para la Comparación del Error

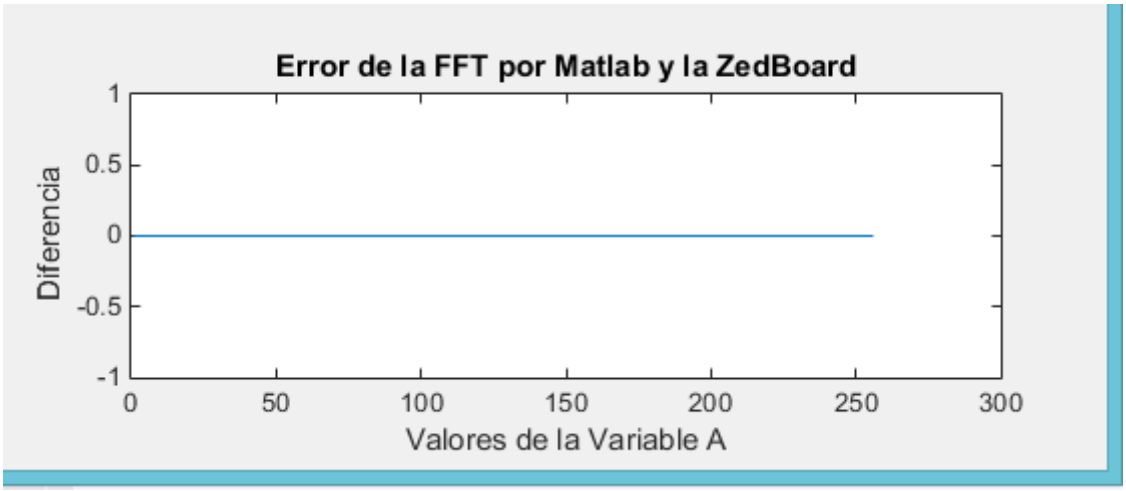

Figura 125. Comparación Error de los Datos

4. RESULTADOS Y DISCUSIÓN

En el proceso de adquisición y almacenamiento de audio el formato de los datos se da en números enteros como se muestra en la Figura 126. Pero el algoritmo de FFT trabaja con números en formato flotante. Por este motivo se debe realizar un proceso matemático para convertir los datos del formato de números enteros a flotantes teniendo en cuenta las especificaciones de rango y resolución de los conversores ADC y DAC.

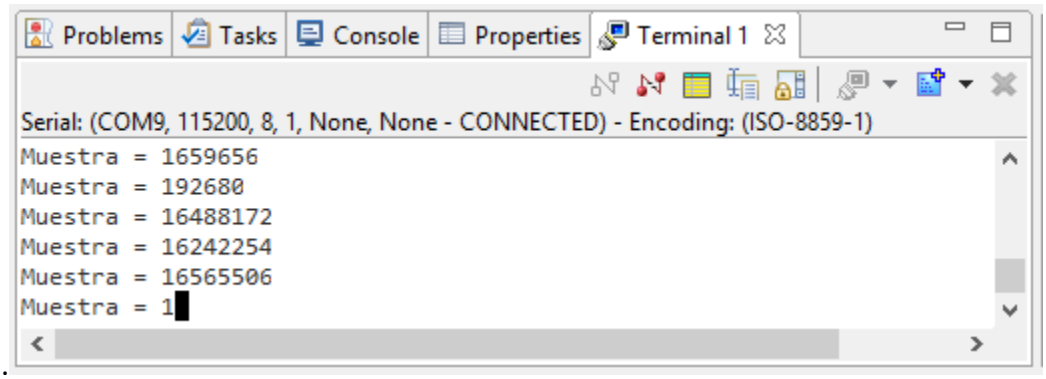

Figura 126.Resultado del Procesamiento de Audio.

Para solucionar este inconveniente se vio la necesidad de obtener un audio en formato de texto desde una fuente externa y almacenarlo en la tarjeta SD de la ZedBoard. El audio que utilizado se llama “audioin.txt” y tiene las siguientes características:

- Frecuencia de muestreo Fs = 44100Hz
- Valor decimal de mayor y menor valor 0.7595214844 y -0.2106323242 respectivamente.
- Cantidad de muestras 663552, lo que equivale a 15.04 segundos de audio.

El proceso para el almacenamiento de audio se describió en el capítulo 3.2 “**ALMACENAMIENTO DE AUDIO EN FORMA DE TEXTO A TRAVÉS DE LA ZEDBOARD**” cumple con la función de almacenar el audio.

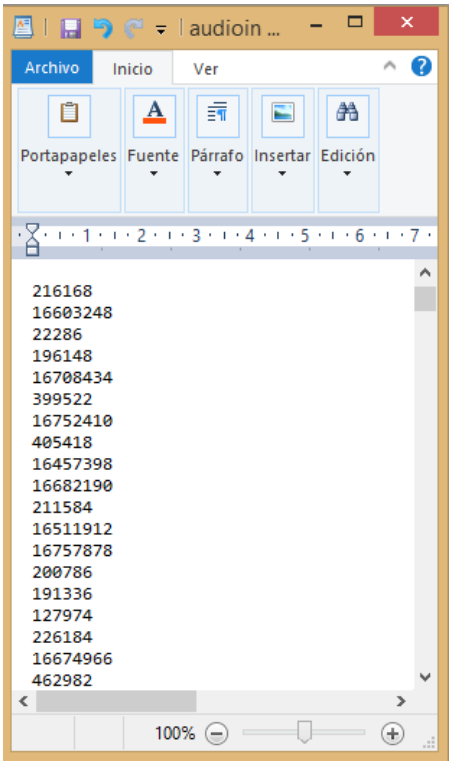

Figura 127.Audio de Entrada.

Por medio de un nuevo proyecto en Vivado, se realiza un algoritmo de programación que lee el archivo de audio original en texto llamado “audioin.txt”, lo transforma a formato flotante, realiza el procedimiento de filtrado y finalmente guarda el audio filtrado en forma de texto en la tarjeta SD con el nombre de “fftout_Zedboard.txt”. El algoritmo realiza muestra a muestra de la siguiente manera:

1. Lee la muestra de texto en la SD.
2. Convierte la muestra en formato flotante.
3. Aplica la FFT.
4. Convierte la muestra modificada nuevamente a texto.

5. Guarda la muestra modificada en la tarjeta SD.

6. Se repite la misma secuencia por cada muestra.

Al realizar la comparación del procesamiento de audio tanto de MATLAB como de la tarjeta ZedBoard se tienen resultados muy similares. La Figura 128 muestra la gráfica de error entre ambos procesamientos “Error de la FFT por MATLAB y ZedBoard”.

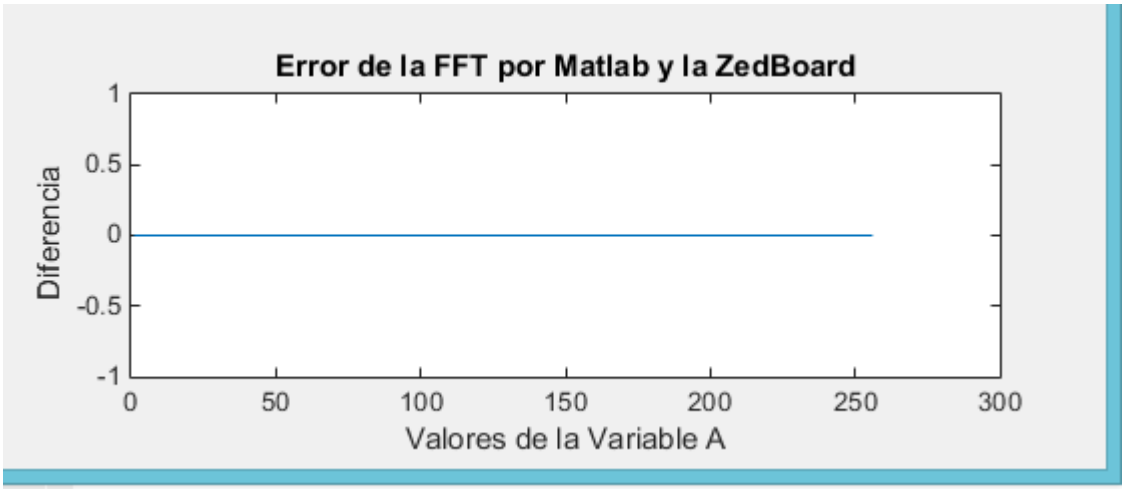

Figura 128. Comparación de los Datos de MATLAB y ZedBoard

Finalmente se tienen ventajas y desventajas de trabajar con estos dos Software, las cuales son:

Ventajas de trabajar en MATLAB:

- ✓ Se puede analizar gráficamente en el mismo medio, ya que el computador cuenta con pantalla para visualizar datos.
- ✓ Una comunidad muy extendida, por lo cual es fácil encontrar ayuda en foros sobre funciones y aplicaciones de MATLAB.

Desventajas de trabajar en MATLAB:

* A pesar de que el software corre en la mayoría de los PC, existen unos requerimientos mínimos para que le software funcione bien como se describe en Requisitos para la Ejecución de MATLAB. Si se desea obtener un mejor rendimiento se deben superar esas capacidades, lo que hace que el PC sea costoso.

Ventajas de trabajar en la ZedBoard:

- ✓ La tarjeta ZedBoard es un sistema de desarrollo, por lo cual se entiende que sirve para crear prototipos de aplicaciones con fines educativos, de investigación e incluso para la industria.
- ✓ Es un sistema embebido que puede trabajar en modo autónomo, es decir, funciona sin la necesidad de una conexión a internet o un software maestro.
- ✓ Se puede configurar la ZedBoard para que realice una aplicación específica, sin importar que la tarjeta sea des energizada, al momento de volverla a energizar se ejecuta la aplicación sin necesidad de programarla nuevamente.

Desventajas de trabajar en la ZedBoard:

- ✓ No existe una comunidad muy extendida, los foros para ayuda son limitados.
- ✓ Para programar la tarjeta y crear aplicaciones se requiere de un computador con el software Vivado.
- ✓ Si la tarjeta se daña y es reparable, los repuestos son difíciles de encontrar en Colombia, por lo general se deben importar.
- ✓ El costo de la Tarjeta ZedBoard en el presente año es de $1.657.965+IVA.

5.CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

- Se pueden aplicar otros algoritmos de la FFT con el fin de minimizar el tiempo de proceso y mejorar la calidad del procesamiento de audio.
- El estudio de la Transformada de Fourier se puede extender a la transformada de Wavelets que constituye actualmente un campo de investigación muy requerida en tratamiento de señales.
- Se logra realizar el procesamiento digital de señales propuesto en el presente trabajo de grado a través de la tarjeta de desarrollo ZedBoard por medio de sus componentes tales como Codec de audio, procesador, Jack de entrada y salida de audio y la tarjeta de almacenamiento SD.
- Se realiza la configuración para la adquisición de audio a través de la tarjeta ZedBoard y almacenamiento del mismo en la tarjeta SD.
- Se implementa un algoritmo de programación que aplica la trasformada rápida de Fourier a un audio almacenado en formato de texto. Este algoritmo cumple satisfactoriamente con el objetivo planteado.
- Se compara los resultados obtenidos a través de MATLAB y la tarjeta ZedBoard logrando resultados similares. Esta es la prueba definitiva que indica que el procesamiento digital de audio es exitoso.
- Se evidencian varias recomendaciones para obtener mejores resultados de los ya obtenidos en este trabajo de grado. Primero, se recomienda mejorar el algoritmo de procesamiento digital de tal manera que dicho procesamiento ejecute con menos líneas y sea lo más eficiente posible. Segundo, se debe estudiar o descifrar el formato con el cual la ZedBoard toma las muestras de audio para que sea

compactible con el algoritmo de programación de la FFT. Finalmente, se recomienda implementar un algoritmo para que el procesamiento digital no se haga con datos almacenados, sino en tiempo real.

REFERENCIAS

Ni.com. (2016). ¿Qué es adquisición de datos?. [Online] Available at: [Accessed 1 Jul. 2016].

Ni.com. (2016). Serie de fundamentos de mediciones con sensores. [Online] Available at: [Accessed 1 Jul. 2016].

Xilinx.com. (2016). Zynq-7000 All Programmable SoC. [Online] Available at: [Accessed 5 Jul. 2016].

Albarado Moya, J. (2011). Procesamiento Digital de señales. [Online] Available at: [Accessed 5 Jul. 2016].

Embedded Centric. (2015).ADC/DAC and Digital Audio Processing. [Online] Available at: [Accessed 1 Jul. 2016].

Zedboard.org. (2014)Standalone SD card SDIO. [Online] Available at: [Accessed 25 Jul. 2016].

Prasad, Ray, K. C., y Dhar, A. S. (2010). FPGA implementation of discrete fractional Fourier transform. International conference on signal processing and communications (spcom).

Van der Byl, Wilkinson, R. H., y Inggs, M. R. (2011). Recursive Fourier transform hardware. IEEE radar conference (radar).

Fast Fourier Transform (FFT) Online] Available at: [Accessed 20 Agosto. 2016].

APÉNDICE

APENDICE A: Programa en lenguaje C para adquisición de audio a través de la ZedBoard.

DAQ_AUDIO.docx

APENDICE B: Script del Audio.h.

AUDIO.h.docx

APENDICE C: Programa para adquisición de audio a través de la ZedBoard.

DAQ_AUDIO_SD.docx

APENDICE E: Programa para almacenamiento de audio y FFT a través de la ZedBoard.

Almacenamiento en SD con FFT.c

APENDICE F: Programa en lenguaje C del algoritmo de FFT.

CodigoenC_FFT.txt

APENDICE G: Script para procesamiento de audio por medio de MATLAB.

FFT_SPECTRO.m

APENDICE H: Script para comparación de procesamiento en MATLAB y ZedBoard.

Comparacion.m

Institución Universitaria	MODALIDAD TRABAJO DE GRADO PRODUCTO OBTENIDO EN TALLERES O LABORATORIOS DEL ITM Registro de actividades y cumplimiento de horas / Talleres o Laboratorios de DOCENCIA	Código	FDE 146
		Versión	02
		Fecha	2015-09-30

Documento de identidad:	1040741228		
Nombre completo del estudiante:	Lorena Riaño Molina		
Programa académico ITM:	Ingeniería Electrónica		
Nombre completo del Docente Asesor:	Luis Fernando Castaño Londoño		
Fecha de iniciación del producto (aaaa/mm/dd):	29/02/2016	Fecha de terminación del producto (aaaa/mm/dd):	30/06/2016
Nombre Taller o Laboratorio:	Sistemas de Control y Robótica		
Ubicación:			
Campus:	Fraternidad		

Fecha A	Fecha M	Fecha D	Actividad desempeñada por el estudiante	Hora ingreso	Hora salida	Total horas	Firma Laboratorista	Firma Estudiante
16	2	29	Socialización del producto a ejecutar	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	3	7	Socialización de cronograma y de plan de trabajo	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	3	12	Investigacion sobre el tema del producto	10:30am	12:00m	2	Luis Fernando Castaño L	Lorena Riaño
16	3	14	Ambientación con las Zedboard (componentes, funcionalidades, usos, servicios...)	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	3	19	Prueba de las tarjetas Zedboard con ejemplos tipicos para verificar el perfecto estado y funcionalidad de las mismas	10:30am	12:00m	2	Luis Fernando Castaño L	Lorena Riaño
16	3	28	Comprobación y validación del contenido del GPIO-demo (encedido y apagado de led, mediante pulsaores, switchs..)	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	4	2	Estudio de guía de laboratorio propuesta en (embeddedcentric -ADC/DAC and Digital Audio Processing)	10:00am	12:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	4	4	Prueba ADC/DAC and Digital Audio Processing embeddedcentric con vivado 2014.4 (No funciona)	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	4	9	Buscar soluciones a las fallas de la guia Lab 8 de embeddedcentric	10:00am	12:00pm	2	Luis Fernando Castaño L	Lorena Riaño
16	4	11	Buscar soluciones a las fallas de la guia Lab 8 de embeddedcentric	6:00pm	8:00pm	2	Luis Fernando Castaño L	Lorena Riaño

16	4	16	Al no encontrar solucion a la Guia Lab 8 de embeddedcentric, se opta por buscar otros ejemplos y guias para adquisicion de audio con la Zed board	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	4	18	Pruebas a guia de laboratorio encontrada en Internet basada en ISE, se hace la migracion a Vivado 2014.4	6:00pm	8:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	4	23	Pruebas a guia de laboratorio encontrada en Internet basada en ISE, se hace la migracion a Vivado 2014.4	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	4	25	No se obtiene resultados satisfactorios con la guia encontrada en internet basada en ISE, se busca posible solucion.	6:00pm	8:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	4	30	No se obtiene resultados satisfactorios con la guia encontrada en internet basada en ISE, se busca posible solucion.	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	2	Se prueba conexión entre la tarjeta zedboard y el PC por terminal, se encuentra resultado satisfactorio.	6:00pm	8:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	7	Consulta de Guias Alternas para la implementación de Audio y Adquisicion de Señales	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	14	Consulta de Guias Alternas para la implementación de Audio y Adquisicion de Señales	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	16	Pruebas de Audio, almacenamiento y Adquisicion de Señales.	6:00pm	8:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	21	Implementacion de otra guia encontrada en http://hamsterworks.co.nz/mediawiki/index.php/Zedboard_Audio	10:00am	12:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	5	23	Implementacion de otra guia encontrada en http://hamsterworks.co.nz/mediawiki/index.php/Zedboard_Audio	6:00pm	8:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	2	Consulta de otras metodologias para la implementacion del audio y procesamiento de Señales	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	3	Consulta de otras metodologias para la implementacion del audio y procesamiento de Señales	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	7	Conexión de la ZedBoard y Enlace con TeamViwer.	5:00pm	7:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	10	Hacer Pruebas con Vivado 2015.3	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	17	Implementación del algoritmo para el almacenamiento de la SD	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	6	24	Implementación del algoritmo para el almacenamiento de la SD	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño

16	7	1	Implementación del algoritmo para el almacenamiento de la SD	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	7	15	Implementación del algoritmo en c para la FFT en la Zedboard	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	7	22	Implementación del algoritmo en c para la FFT en la Zedboard	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	8	5	Implementación del algoritmo en c para la FFT en la Zedboard	7:00pm	9:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	8	13	Implementación del algoritmo en c para la FFT en Matlab	10:00pm	1:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	8	20	Implementación del algoritmo en c para la FFT en Matlab	10:00pm	1:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	9	10	Comprobar conectividad de dispositivos y comparación de resultados	10:00pm	1:00pm	2	Luis Fernando Castro L	Lorena Riaño
16	9	24	Comprobar conectividad de dispositivos y comparación de resultados	10:00pm	1:00pm	1	Luis Fernando Castro L	Lorena Riaño
TOTAL HORAS								69

Lorena Riaño
Firma Estudiante

Luis Fernando Castro Londoño
Nombre y firma Laboratorista

Nombre y firma Profesional Universitario - Centro de Laboratorios

Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

FIRMA ESTUDIANTES Lorena Riaño

FIRMA ASESOR Luis Fernando Castro Londoño

FECHA ENTREGA: 24/10/2016

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO_____ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____