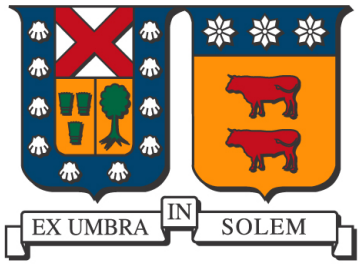
Universidad Técnica Federico Santa María

Proyecto Final

Diseño Avanzado de Sistemas Digitales



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Alumno:
Damián Quiroz

Profesor:
Gonzalo Carvajal

Viernes 15 de Junio de 2018

1. Introducción y motivación

En el área auditiva se emplean métodos de análisis de señales y modificación de características utilizando implmentación bastante cara, por lo que se trata de probar las capacidades de la FPGA Nexys4 DDR de Xilinx para aplicaciones relacionadas con el análisis y modificación de señales de audio.
Existen trabajos similares intentando enfrentar desafíos mas complejos enfocados al cálculo de transformada de Fourier o a la implementación de una herramienta con múltiples efectos de audio orientado al uso de instrumentos musicales (principalmente guitarras eléctricas), pero trabajos realizados (en su mayoría) son realizados en tarjetas de desarrollo que poseen algún Codec de Audio nativo [**7**], por lo que la implementación de los efectos y el manejo de las señales de audio es levemente diferente.
El objetivo inicial era generar un diseño capaz de modificar una señal de audio mediante la implementación de filtros digitales y algunos efectos de audio con una interfaz gráfica que permitiera visualizar los resultados de la modificación. Este objetivo se vio principalmente limitado por la falta de conocimientos sobre filtros digitales, lo que conlleva un extenso trabajo de cálculo, y finalmente de implementación. Inicialmente se utilizó una plantilla de un filtro digital en VHDL, el cual funciona de forma correcta para ordenes bajos, pero al momento de incrementar el orden del filtro, las salidas comienzan a verse afectadas por los coeficientes del filtro y la resolución de los bloques DSP (ya que la implementación en cascada requiere que con cada suma, la resolución de la salida sea de 1 bit más, agregando la multiplicación por un coeficiente), llevando la respuesta del filtro a un nivel saturado en todo momento, por lo que no se pudo realizar este tipo de implementación. En una segunda instancia se optó por utilizar el IP core de filtros FIR que posee Xilinx, en este caso se encontró el problema con los DSP Slices que posee la tarjeta, ya que no permite operaciones con punto decimal, debido a ésto todo modelo con filtros FIR que se espera, tendrá como salida una versión amplificada de dicho filtro. Es decir, si se espera un filtro FIR con banda de paso 0-1Khz y ganancia cero, en la implementación realizada en la tarjeta, éste filtro tendrá una ganancia de 40-70dB dependiendo del orden del filtro. Por lo que para utilizar las salidas de dicho filtro se deben tomar los bits más significativos pensando que será una salida (con o sin signo) amplificada de la respuesta esperada inicialmente.
La dificultad mas grande encontrada al intentar realizar este proyecto fue la implementación de filtros utilizando DSP Slices sin capacidad de cómputo decimal, ya que ésto cambia rotundamente la respuesta esperada, por otra parte calibrar éstos filtros para que funcionen correctamente es un trabajo arduo y no se llevo a cabo por completo, sino que se lograron rangos de funcionamiento.

El modelo presentado en la Figura 2, es lo esperado en una segunda instancia del proyecto. Todos los módulos fueron realizados, pero no todos terminaron funcionando correctamente, debido a ésto se explica como probar los módulos funcionales (sujetos a revisión y mejoras).

2. Desarrollo

2.1. Descripción General

A continuación se presenta el diagrama general de módulos utilizado en este diseño. Este diagrama consta de 8 partes principales, que completan el funcionamiento descrito en la sección anterior, se entiende que todos los módulos están implementados, pero no todos presentan el comportamiento deseado, por lo tanto están completamente sujetos a cambios y mejoras en su implementación.

2.2. Módulos principales

La primera parte principal de éste diseño, corresponde al módulo ADC (Analog Digital Converter), que cumple con la función de convertir las señales del mundo real (analógico) a señales digitales para su posterior análisis.
El segundo componente principal, corresponde a un banco de filtros FIR (Finite Impulse Response), que

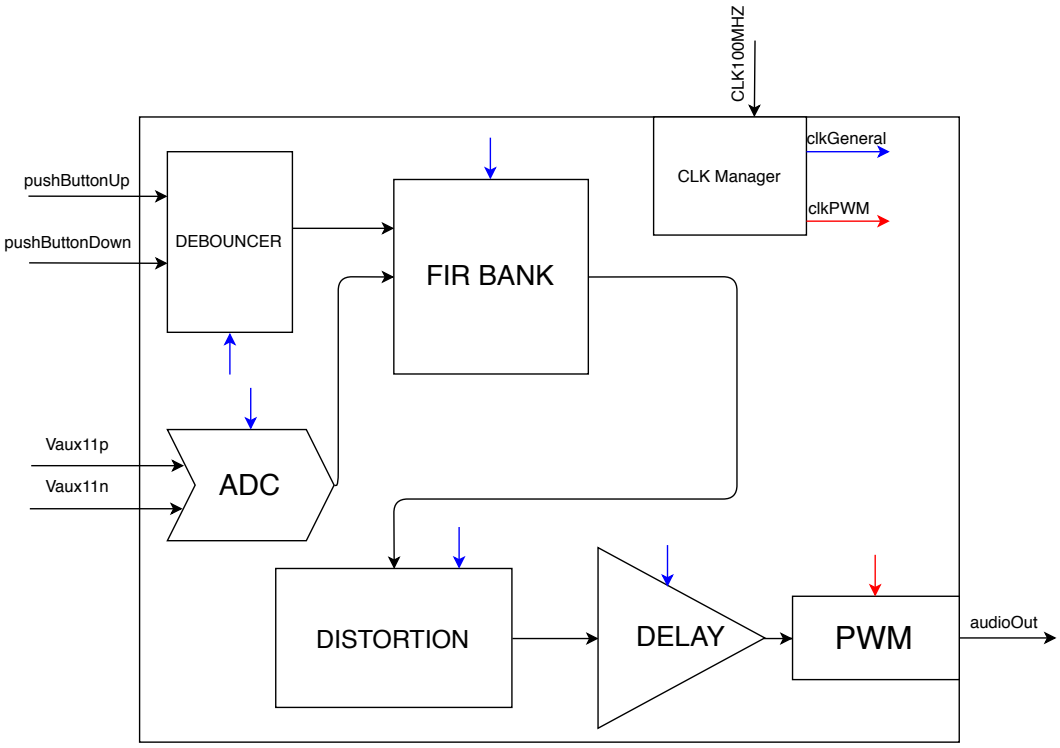


Figura 1: Diagrama General de módulos pensados en una segunda instancia.

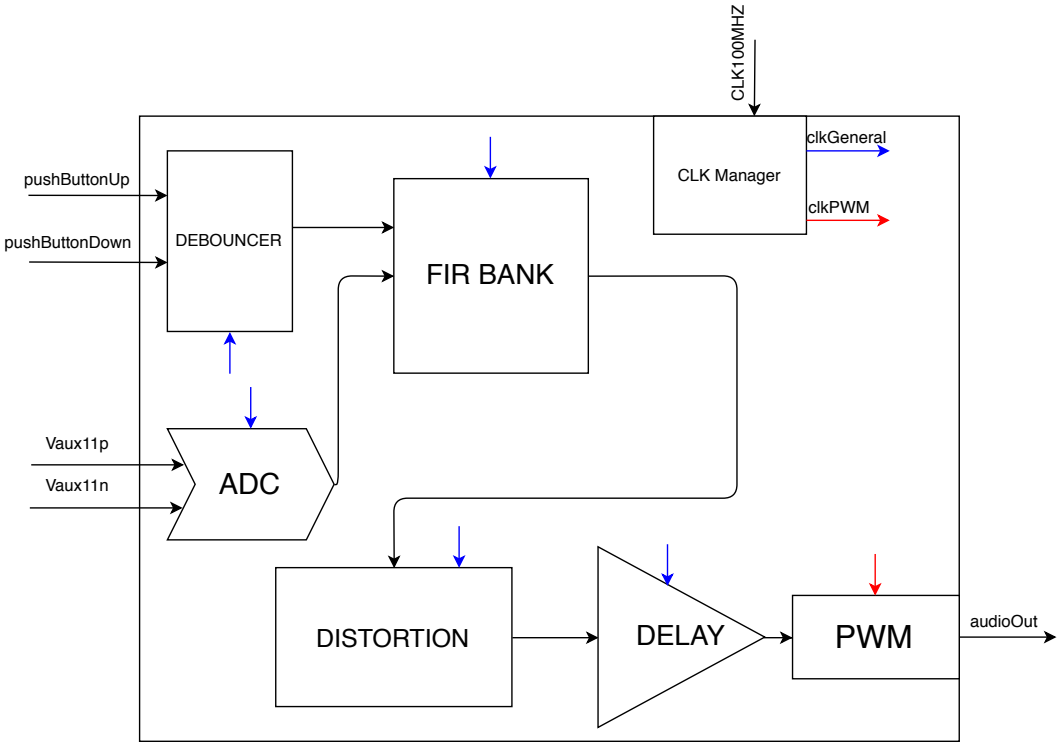


Figura 2: Diagrama General de módulos.

cumplen con la característica de ecualizador de las señales de entrada.
La tercera parte corresponde al módulo que implementa el efecto distorsión en las señales que provienen del banco de filtros, éste modulo cuenta con un nivel configurable de forma digital para modificar la distorsión

de la señal (modifica el nivel de distorsión).
La cuarta y ultima parte, corresponde al módulo que implementa el efecto Delay, que corresponde a un retraso de la señal en unos cuantos milisegundos generando un efecto parecido a un eco en el sonido.
La quinta parte corresponde a un generador de relojes (clock manager), que se utiliza para generar dos relojes que permiten el funcionamiento de los módulos internos del sistema. La sexta parte corresponde a un modulo Debouncer, que se utiliza como sistema anti rebotes en los botones utilizados por la tarjeta. Finalmente el último módulo corresponde a un generador de señales PWM (Pulse Width Modulation) para enviar la señal de audio al driver de audio interno de la tarjeta.

Módulo ADC

El conversor análogo digital (Analog Digital Converter) corresponde, como se mencionó anteriormente, a un dispositivo capaz de transformar una señal analógica, a una señal digital muestreada en tiempo y en voltaje. El funcionamiento de éste módulo está descrito en la documentación oficial de Xilinx, y en el Apéndice 3.6 de éste documento.
A continuación se presenta la estructura general del módulo, indicando la función de cada entrada y salida usada en el proyecto.

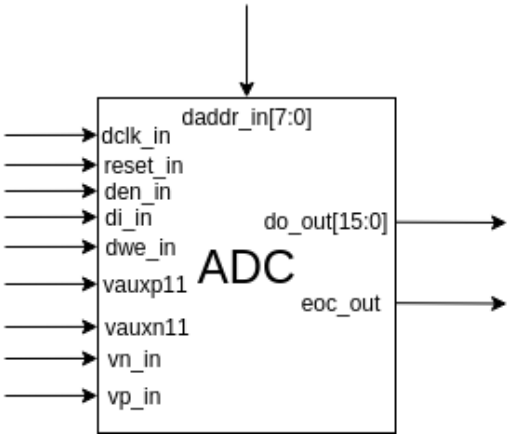


Figura 3: Descripción general de las entradas y salidas del módulo ADC

Entradas y salidas del módulo ADC

dclk_in: Corresponde a la señal de reloj utilizada por el módulo, debe ser una señal con una frecuencia entre 10Mhz y 400Mhz.

reset_in: Corresponde a la señal de reset del módulo ADC.

den_in: Señal enable, que permiete el inicio de una conversión.

vauxp11: Pin positivo del canal auxiliar 11 que se usará en el módulo ADC.

vauxn11: Pin negativo del canal auxiliar 11 que se usará del módulo ADC.

vp_in: Voltaje positivo de referencia, interno del módulo ADC.

vn_in: Voltaje negativo de referencia, interno del módulo ADC.

do_out: Bus de datos convertidos, posee un largo de 16 bits de resolución donde sólo 12 son usables de forma segura.

eoc_out: Señal que indica el fin de una conversión de dato.

drdy_out: Señal que indica si un dato se terminó de convertir y está listo para usarse.

daddr_in: Dirección que permite seleccionar a las entradas Vaux11, como las entradas al ADC.

Módulo FIR BAKN

El banco de filtros FIR, corresponde al módulo que contiene los filtros digitales que se usarán para "ecualizar" la señal de entrada.

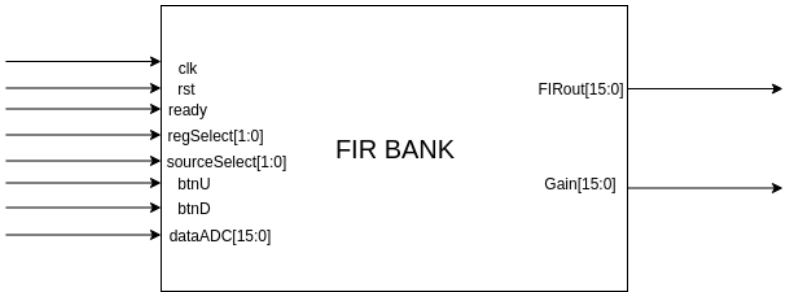


Figura 4: Descripción general de las entradas y salidas del módulo FIR BANK

Entradas y salidas del módulo FIR BANK

clk: Señal de reloj general del módulo. Se usa la misma señal que en el módulo ADC, para evitar problemas de timing entre ambos módulos.

rst: Señal de reset general del módulo. Se usa la señal de reset general de la tarjeta.

ready: Señal que indica si una conversión está lista para usarse. Viene directamente desde el pin *drdy_out* del módulo ADC.

regSelect Señal que especifica a qué filtro se le modificará la ganancia.

sourceSelect Señal que especifica desde qué filtro proviene la señal de salida.

allPass Señal que indica que no se realizarán cambios en la señal (filtrado) antes de pasarla al siguiente módulo.

btnU Señal que proviene del botón BTNU de la tarjeta. Se usa para modificar la ganancia del módulo.

btnD Señal que proviene del botón BTND de la tarjeta. Se usa para modificar la ganancia del módulo.

dataADC: Bus de datos que contiene el dato convertido por el módulo ADC.

FIROut: Bus de datos que contiene los datos ya procesados por el módulo FIR BANK.

gain: Señal que indica, la ganancia del filtro seleccionado, o en su defecto el valor numérico de la señal procesada.

Submódulos

El módulo FIR BANK posee una serie de submódulos que implementan el comportamiento requerido por el proyecto. A continuación se presentan los submódulos con su descripción general de entradas y salidas, los detalles de su desarrollo e implementación se muestran en los Apéndices 3.1 y 3.3 del documento.

Módulos FIR

Para implementar las características de un "ecualizador", el módulo FIR BANK posee 3 módulos que implementan filtros FIR, pasa bajos, pasa medios y pasa altos, cada uno con bandas de paso diferentes. La estructura de los 3 módulos es la misma por lo tanto se describe solo uno de ellos por simplicidad.

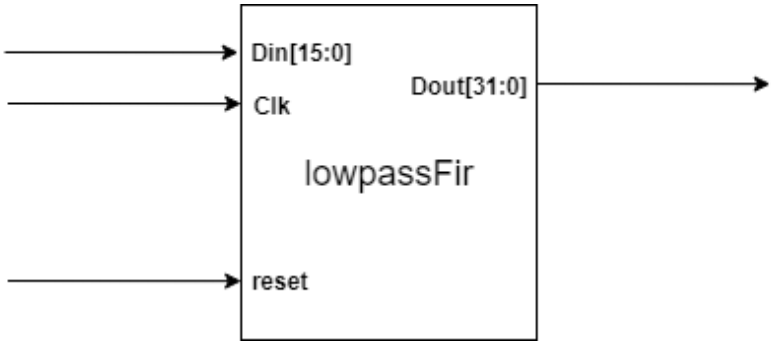


Figura 5: Estructura básica de los módulos FIR.

Din: Entrada del módulo, de largo 16 bits.

clk: Reloj de entrada, proviene de la señal clkGeneral.

reset: Señal de reset del módulo.

Dout: Salida del módulo, posee un largo de 32 bits.

Estos módulos están implementados, pero su funcionamiento para filtros de orden superior no es bueno, por lo que finalmente **se utiliza el IP core de filtros FIR entregado por Xilinx** para realizar ésta tarea. La forma de utilizarlos se explica en el Apéndice 3.4.

Módulo regControl

El módulo regControl permite controlar la ganancia de cada uno de los filtros FIR implementados.

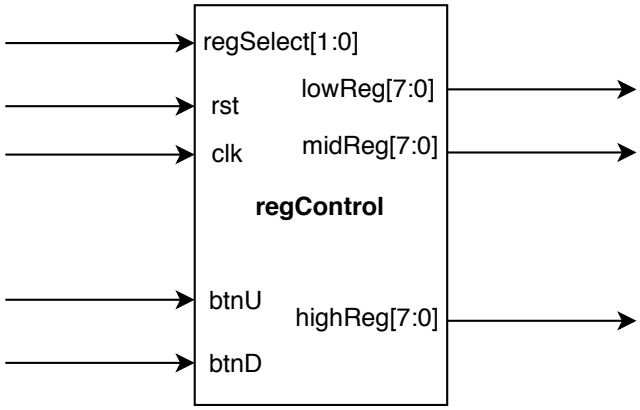


Figura 6: Estructura general del módulo regControl.

Módulo DISTORTION

El módulo DISTORTION corresponde a un módulo que implementa el efecto distorsión en la señal proveniente del módulo FIR BANK. El funcionamiento del efecto distorsión se explica en el Apéndice 2 de éste documento.

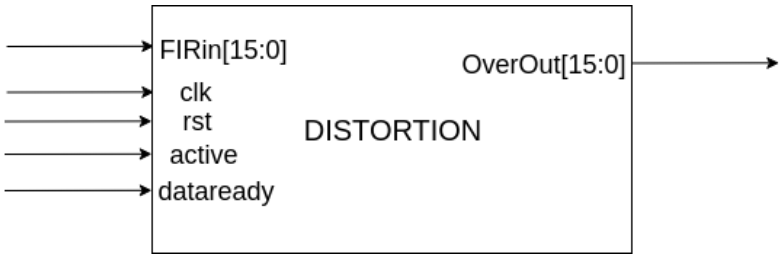


Figura 7: Descripción general de las entradas y salidas del módulo DISTORTION.

Entradas y salidas del módulo DISTORTION

clk: Señal de reloj general del módulo. Usa la misma señal de reloj que el módulo ADC, para evitar problemas de timing.

rst: Señal de reset general del módulo. Usa el botón de reset de la tarjeta.

active: Señal que indica si se desea activar el módulo o no. Es controlada por un switch.

dataready: Señal que indica si hay un dato listo para ser usado.

FIRin: Bus de datos proveniente del banco de filtros, si la señal active no está en alto, se pasa este bus de datos datos directamente a la salida.

OverOut Bus de datos de salida del módulo. Dependen de la señal "active".

Módulo DELAY

El módulo DELAY corresponde al efecto de audio "delay" el cual se aplica a la señal proveniente del módulo DISTORTION. El funcionamiento del efecto se explica en el Apéndice 3.

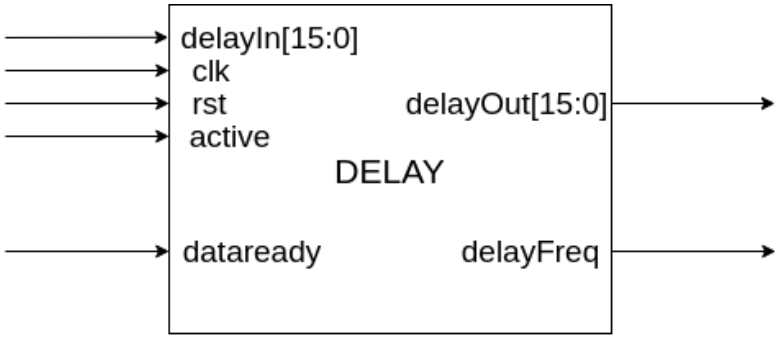


Figura 8: Descripción general de entradas y salidas del módulo DELAY.

Entradas y salidas del módulo DELAY

clk: Señal de reloj del módulo, corresponde a la misma señal de reloj del módulo ADC para evitar problemas de timing.

rst: Señal de reset del sistema, corresponde al botón reset de la tarjeta.

active: Señal que indica si se usará el efecto del módulo o no. Depende de un switch.

dataready: Señal que indica si hay un dato listo para usarse.

delayIn: Bus de datos proveniente del módulo DISTORTION.

delayFreq: Señal usada para testear la frecuencia del efecto.

delayOut: Bus de datos de salida del módulo, corresponde a una señal retardada.

Módulo Deboucer

El módulo Debouncer corresponde a un módulo anti-rebotes para que el funcionamiento de los botones de la placa no se vea afectado por los rebotes mecánicos generados.

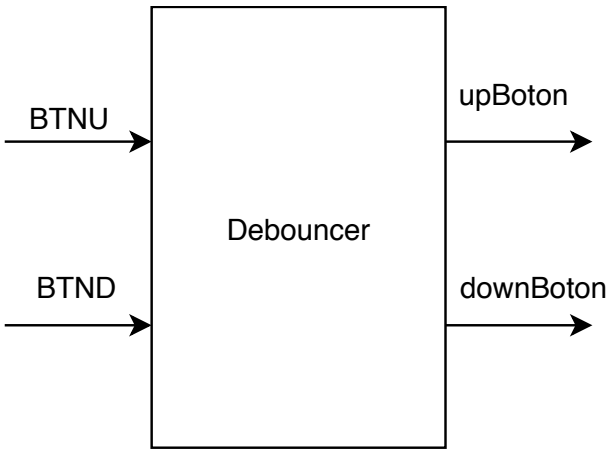


Figura 9: Descripción general de entradas y salidas del módulo Debouncer.

Entradas y salidas del módulo Debouncer

clk: Reloj general del sistema implementado.

btnU: Entrada proveniente del botón BTNU de la tarjeta.

btnD: Entrada proveniente del botón BTND de la tarjeta.

upButton: Salida sin rebotes de un pulso que indica el canto de subida de la presión del botón BNTU.

downButton: Salida sin rebotes de un pulso que indica el canto de subida de la presión del botón BTND.

Módulo Clock Manager

El módulo Clock Manager corresponde a un generador de relojes diferentes al reloj nativo de la tarjeta (100Mhz), los que se utilizan tanto para generar la salida de audio PWM y el funcionamiento de los demás módulos.

Figura 10: Descripción general de entradas y salidas del módulo Clock Manager.

Entradas y salidas del módulo Clock Manager

clkIn: Señal de reloj nativa de la tarjeta, por defecto es el reloj de 100MHZ, puede ser un reloj generado por otro ClockManager.

clkGeneral: Señal de reloj usada por los módulos generales del sistema, puede ser un reloj entre 10Mhz y 150Mhz (en este proyecto).

clkPWM: Señal de reloj usada por el módulo PWM. El reloj utilizado debe ser un múltiplo entero del reloj clkGeneral.

Módulo PWM

El módulo PWM corresponde a la transformación de un número digital en una señal PWM con un ancho proporcional al número que se ingresa.

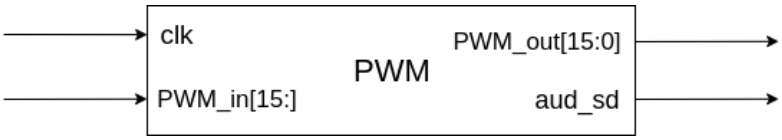


Figura 11: Descripción general de entradas y salidas del módulo PWM.

Entradas y salidas del módulo PWM

clk: Corresponde al reloj general del módulo. Es un reloj más rápido que el usado para el módulo ADC, ya que se necesita un contador rapido para generar salidas PWM. Debe ser un múltiplo entero del reloj usado para el módulo ADC.

PWM_in: Valor numérico del dato que se desea convertir a PWM.

aud_sd: Señal de activación del módulo PWM de la tarjeta, se mantiene en 1 para uso continuo.

PWM_out: Señal PWM generada por el módulo.

2.3. Resultados

Primero se verifica el comportamiento de los filtros implementados. Utilizando una señal triangular de frecuencia 500Hz se observa la respuesta a del dispositivo tanto para el filtro pasa bajos, pasa medios y pasa altos.

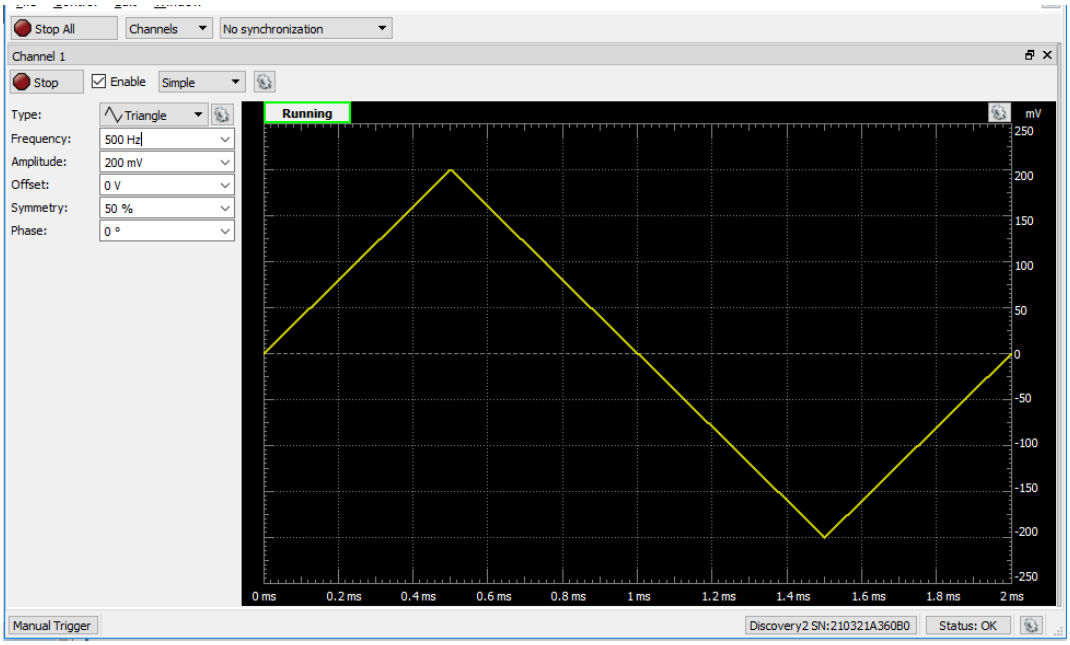


Figura 12: Señal triangular de prueba utilizada.

Como respuesta a esta señal, el dispositivo entrega la siguiente salida en los filtros. Por orden se presentan las respuesta para el filtro pasa bajos, pasa medios y pasa altos.

Como se puede observar, el filtro pasa bajos entrega una onda de frecuencia 500Hz, lo que demuestra su correcto funcionamiento, el filtro pasa medios utilizado atenúa la señal en cierta medida y muestra otras armónicas incluidas en su señal, esto se debe a que como el filtro no es de un orden tan alto, permite el paso de frecuencias no deseadas que están fuera de su banda de paso. En el filtro pasa altos se ve una clara atenuación de la señal, pero al igual que en el caso del pasa medios, se observan algunas armónicas (aunque de magnitud pequeña).

El comportamiento del módulo Delay no se puede corroborar de manera correcta ya que las señales se saturan al realizar la operación matemática necesaria en éste módulo.

2.4. Posibles cambios o mejoras

Los principales cambios y mejoras vienen en los módulos que tratan de implementar efectos de audio. Se esperaba por lo menos implementar el efecto Delay de manera correcta, pero debido a los tipos de datos y la baja resolución que tiene el driver PWM, al activar dicho módulo se llevan las señales a una versión saturada de sí misma, provocando sonidos que son prácticamente ruido en la salida. Posee un rango de funcionamiento correcto para señales pequeñas y probablemente sin signo, ya que en éste diseño se intentó implementar todo utilizando señales con signo (para no depender de un circuito externo que realice el offset de la señal de entrada), el diseño se complicó bastante al considerar las opciones de signo en cada etapa del diseño. Lo cual también limitó el reloj de funcionamiento del driver PWM. Una mejora didáctica sería implementar una interfaz gráfica (como se esperaba inicialmente) para visualizar las señales deseadas sin la necesidad de utilizar herramientas externas, ya que éstas introducen ruido y no todos las poseen a mano, o no siempre se cuenta con ellas.

Un cambio que se puede hacer para que funcione de manera correcta, es implementar todo en un entorno

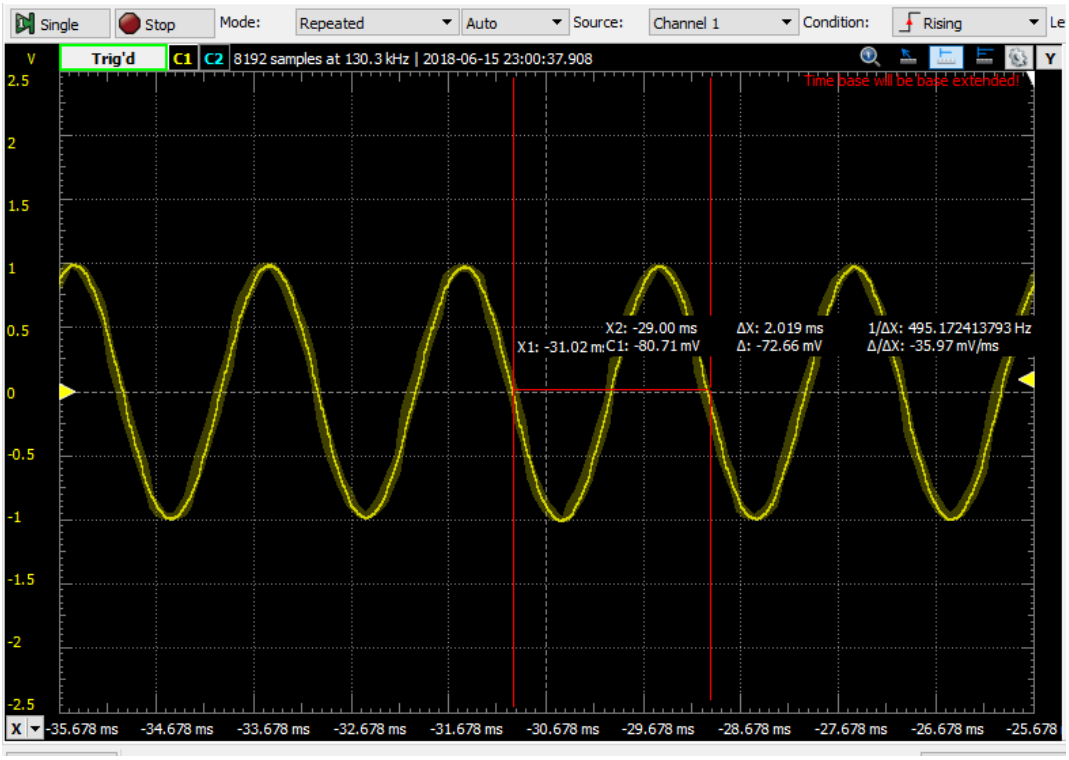


Figura 13: Respuesta del filtro pasa bajos.

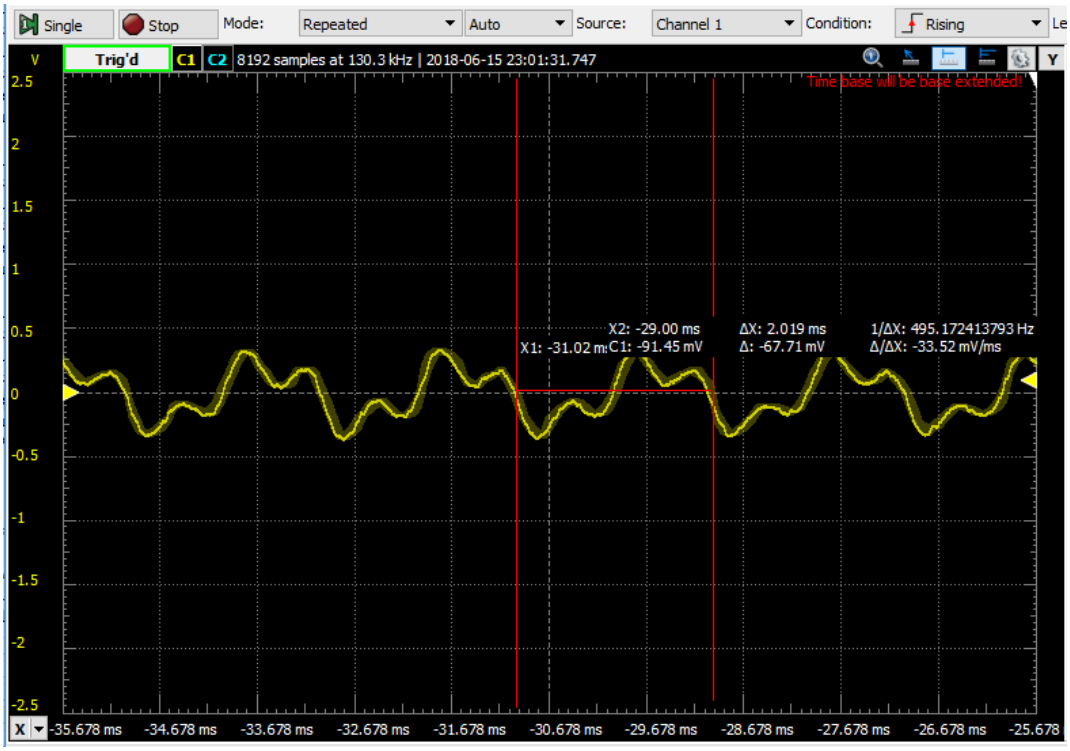


Figura 14: Respuesta del filtro pasa medios.

sin signo, para corroborar el correcto funcionamiento del dispositivo y los módulos implementados. Tomando en cuenta la necesidad de un circuito externo para insertar la señal al dispositivo y tomando en cuenta que no se deben exceder los límites de voltaje permitidos por la placa, ya que se puede dañar.

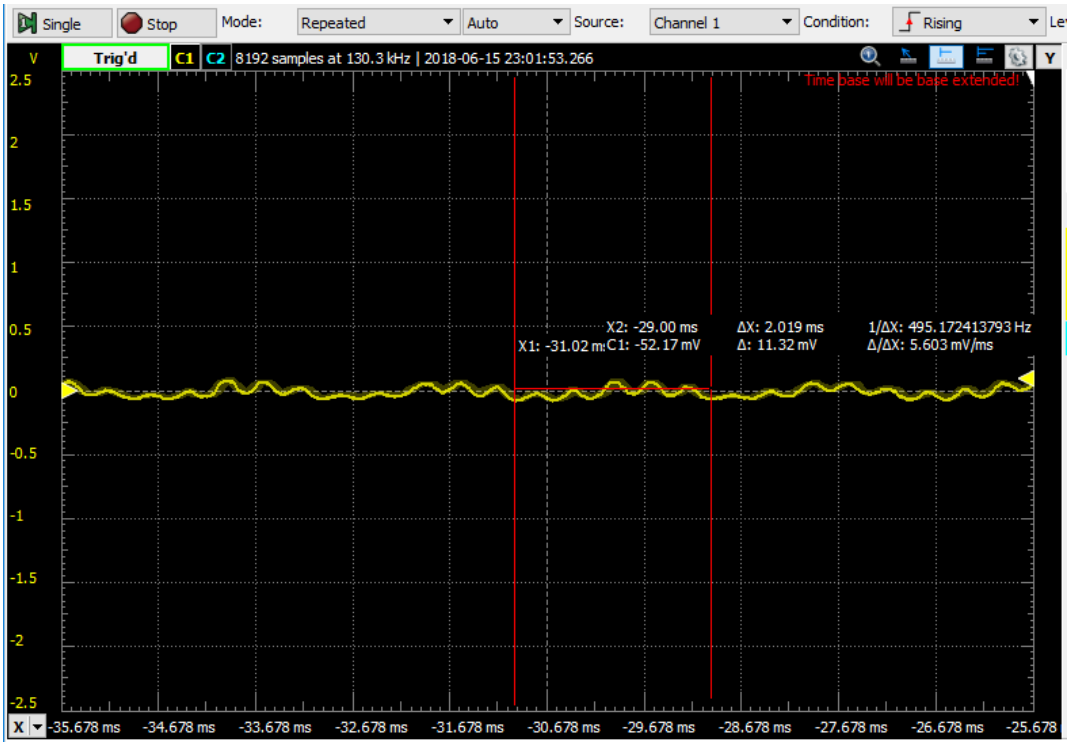


Figura 15: Respuesta del filtro pasa altos.

Referencias

[1] Digilent. Nexys 4 DDR Music Looper.
https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-ddr-looper-demo/start.

[2] Xilinx, XADC Artix 7 series User Guide.
https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

[3] Digilent. April 11, 2016. Nexys4 DDR™ FPGA Board Reference Manual.
https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys4ddr_rm.pdf

[4] John G. Proakis y Dimitris G. Manolakis. *Tratamiento digital de señales.*
PEARSON EDUCACIÓN S.A., Madrid, 2007.

[5] Alan V. Oppenheim y Ronald W. Schafer. *Tratamiento de señales en tiempo discreto.*
PEARSON EDUCACIÓN, S.A. 2011

[6] Emilia Gómez Gutiérrez .*Introducción al filtrado digital*
Escola Superior de Musica de Catalunya , Noviembre de 2009.

[7] Anthony Giardina. *Digital Graphic Equalizer Implemented Using an FPGA.*
San Luis Obispo ,2012.

[8] Andrew B. Shapiro y Marc P. Resnick, Field Programmable Digital Audio Effects Rack
Fall 2010.

[9] Xilinx, FIR Compiler v7.2
https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v7_2/pg149-fir-compiler.pdf
Noviembre 18, 2015.

3. Apéndices

3.1. Teoría básica de filtros FIR

Los filtros digitales son una herramienta muy útil, ya que en el último tiempo se ha optado por digitalizar la mayoría de los procesos, tanto de control como de adquisición de datos.
Para entender el desarrollo de un filtro FIR como los usados en éste documento es necesario entender la base matemática desde la cual provienen, en este caso la Transformada Z.

3.1.1. Transformada Z

La transformada Z, a grandes rasgos y para efectos de éste documento, corresponde a una discretización de la Transformada de Fourier, aunque se entiende que son herramientas totalmente diferentes.
La transformada Z y la transformada Z inversa se definen en las Ecuaciones 3.1 y 3.2, tomando en cuenta que es sólo la definición teórica de ésta, por lo que con esto no basta para replicar el funcionamiento del proyecto descrito en éste documento.

$$X(z) = \sum_{k=-\infty}^{\infty} x(k) \cdot z^{-k} \tag{3.1}$$

$$x(k) = \frac{1}{2\pi j} \oint_C X(z) \cdot z^{n-1} dz \tag{3.2}$$

En la Ecuación 3.1 se entiende a $x(k)$ como una señal muestreada digitalmente (generalmente datos discretos entregados por un ADC o datos de eventos ocurridos utilizando alguna métrica temporal). En la Ecuación 3.2 se entiende a j como la variable compleja y a la región C como la región de convergencia de la Transformada Z de $x(k)$, la cual corresponde a anillos concentricos en el plano Z, para más detalles consultar [**4**] y [**5**].

3.1.2. Interpretación de filtros digitales utilizando Transformada Z

Un filtro digital corresponde a un equivalente discretizado de un filtro analógico eléctrico, el cual tiene por finalidad limitar el paso de componentes de frecuencia no deseados en un proceso. Para el análisis de filtros digitales se debe tener en cuenta que el comportamiento de éste depende tanto de su estructura como de la frecuencia a la que se esté muestreando la señal que se desea filtrar.
Al igual que en la teoría analógica, existen varios tipos de filtros digitales, en este caso se trabaja únicamente con filtros FIR, los cuales quedan completamente representados por el equivalente a una ecuación diferencial (llamada ecuaciones de diferencias) como la Ecuación 3.3 que representa un filtro digital de orden 2.

$$y(k) = a_0 \cdot x(k) + a_1 \cdot x(k-1) \tag{3.3}$$

Los coeficientes a_0 y a_1 son los encargados de mostrar la naturaleza del filtro y el argumento $k-1$ indica que se usa una muestra anterior de la señal de entrada $x(k)$. Dependiendo de los coeficientes a_0 y a_1, se pueden obtener respuestas tipo pasa altos, pasa bajos, pasa medios, rechaza banda, pasa banda, etc. No se indagará mas en el tema, ya que es bastante amplio y requiere una matemática que se debe estudiar aparte de éste documento.

3.2. Implementación de filtros FIR en FPGA

Como se indicó en la sección anterior, para representar el comportamiento de un filtro digital basta con implementar una ecuación de diferencias, por lo tanto para replicar el comportamiento de un filtro FIR en una FPGA es necesario representar esta ecuación de diferencias en un diagrama de bloques para mejor entendimiento de lo que se desea implementar. Para la Ecuación 3.3, el diagrama de bloques básico generado será el siguiente:

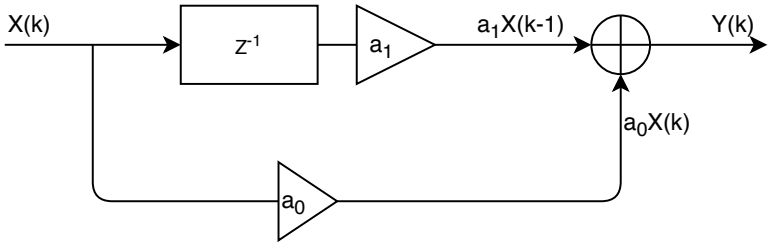


Figura 16: Diagrama de bloques básico de un filtro FIR.

En éste diagrama de bloques, el bloque Z^{-1} indica un retardo en la señal. Si se entiende el funcionamiento de una FPGA es fácil ver que éste retardo simplemente queda representado por la implementación de un Flip-Flop, por lo tanto para implementar filtros de orden superior, el uso de Flip-Flops será proporcional a la cantidad de retardos que se usen de la señal.
La representación circuital en la FPGA del diagrama mostrado en la Figura 16 se muestra a continuación. Esta representación se puede extrapolar a un modelo de orden superior, tomando en cuenta que los bloques de amplificación (buffers) se representan mediante bloques DSP o tablas LUT dependiendo de la implementación que se desea realizar, y el módulo sumador corresponde también a un bloque DSP o una tabla LUT que contenga los valores necesarios para obtener el resultado deseado.

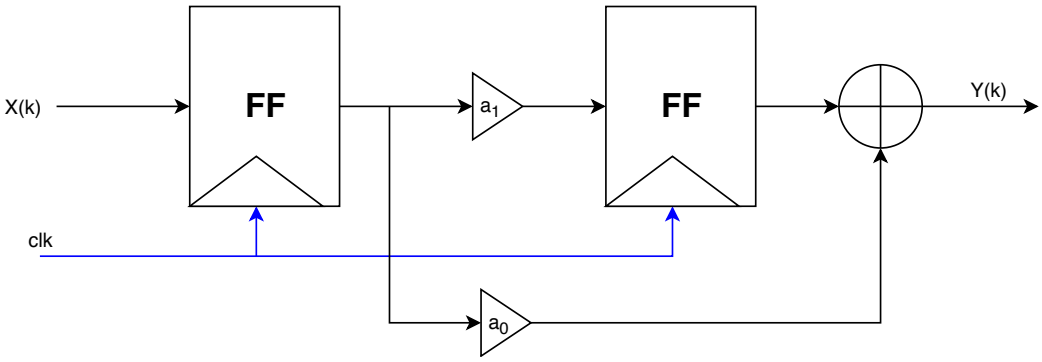


Figura 17: Representación circuital de un diagrama de bloques de filtro FIR.

En este proyecto se usaron filtros FIR de orden 40, por lo tanto se necesitan 40 Flip-Flops por cada filtro FIR implementado. Un diagrama de bloques representativo de un filtro FIR de orden superior es mostrado a continuación. Este diagrama es una representación visual de los módulos FIR utilizados para éste proyecto.

3.2.1. Frecuencia de reloj para los filtros FIR

Para el correcto funcionamiento de los filtros implementados se debe tener en cuenta que la frecuencia de funcionamiento del sistema permita que los cálculos deseados se realicen de forma correcta entre cada etapa del filtro. Debido a ésto, se probó con varias frecuencias encontrando como limitante principal a los bloques DSP que realizan las multiplicaciones de los coeficientes de cada filtro, en conjunto con el clock Manager, ya que no permite generar más de 2 clocks con frecuencias altas (superiores a 200Mhz).

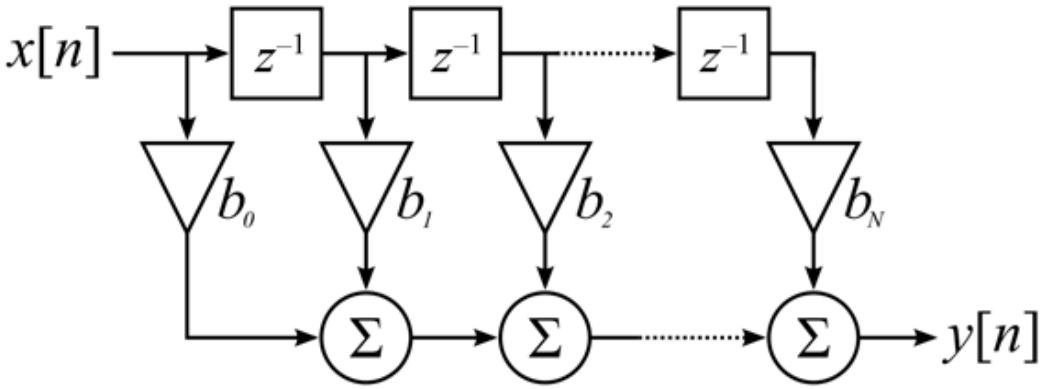


Figura 18: Representación en diagrama de bloques de un filtro FIR de orden superior.

Finalmente se decidió que para un trabajo regular y sin problemas, no es necesario utilizar una frecuencia excesivamente alta para el trabajo de los filtros, ya que solamente deben ser capaces de tener su resultado antes que de el módulo ADC tenga un nuevo dato. Debido a ésto, se optó por usar una frecuencia baja en el lado de los filtros (10-50Mhz) y una frecuencia alta solamente en la parte de la conversión PWM.

3.3. Cálculo de coeficientes FIR con ayuda de Matlab

Para facilitar el trabajo del desarrollo de filtros, ya que realizar filtros es un trabajo bastante largo, se utiliza la herramienta de creación de filtros de Matlab "Filter Builder", la cual permite diseñar filtros digitales de una manera fácil teniendo en cuenta los parámetros que se desea cumplir para su correcto funcionamiento.

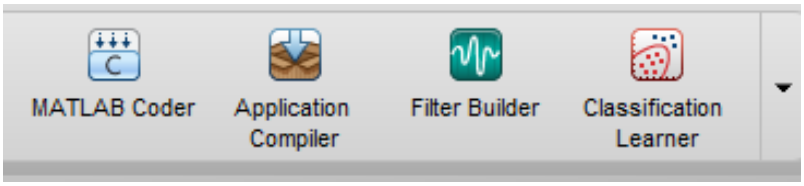


Figura 19: Filter Builder de Matlab.

Una vez seleccionado el "Filter Builder" de Matlab, aparecerá una ventana como la mostrada en la Figura 20, en la que se puede seleccionar que tipo de filtro se desea diseñar. Para efectos de ejemplo se hará un filtro pasa bajos mostrando cada etapa y configuración necesaria para su correcto funcionamiento.

Al seleccionar la opción LowPass mostrada en la Figura 20, aparecerá una ventana como la mostrada en la Figura 21, donde aparecen una serie de pestañas y opciones que se explicarán a continuación para el desarrollo de un filtro utilizable en la tarjeta Nexys4 DDR.

En la primer pestaña, denominada "Main" se especifican las características de la respuesta del filtro, el tipo de filtro que se creará, el orden, los métodos para el cálculo de coeficientes,etc. Para éste proyecto se usa un filtro tipo FIR, por lo tanto la casilla "Impulse Response" debe tener la opción FIR seleccionada. El orden que se escogió en el proyecto puede variar dependiendo de la capacidad que se desee utilizar de la tarjeta, pero para obtener un filtro de orden 40 se debe seleccionar en la casilla "Orden Mode" la opción Specify, con lo que se desplegará una nueva casilla para especificar el orden del filtro. La casilla "Filter type" se dejó por defecto en Single-rate, ya que el resto de las arquitecturas no producen un mayor cambio en la respuesta del filtro.

Cabe destacar que el orden también depende de la cantidad de filtros que se desee implementar, ya que la tarjeta solo cuenta con 240 DSP Slices, por lo que se pueden realizar operaciones de forma segura

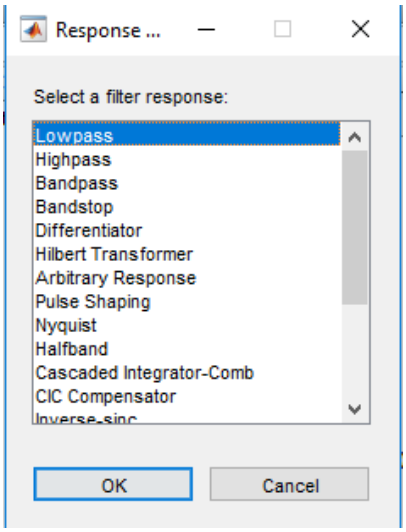


Figura 20: Selección del tipo de filtro a diseñar.

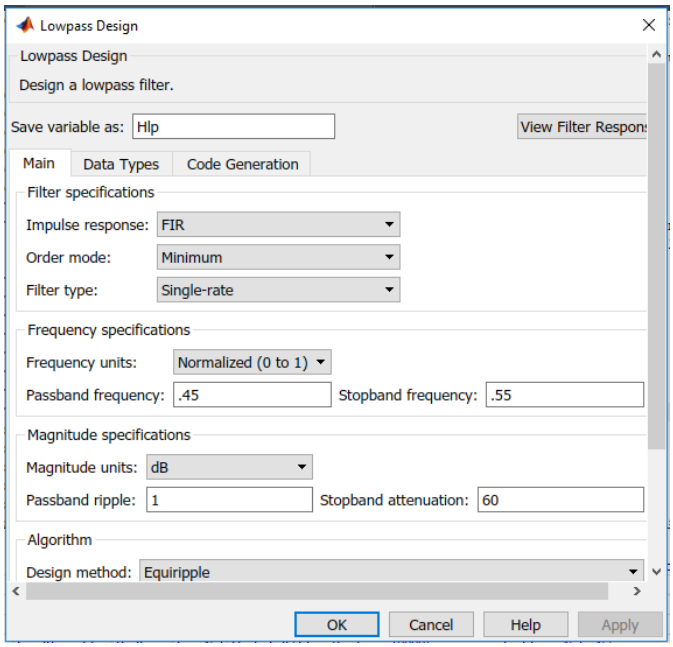


Figura 21: Menú inicial de la herramienta Filter Builder de Matlab.

con bloques DSP, de hasta 240 etapas de filtros. Es decir, en este caso, ya que se implementaron 3 filtros, pudieron ser 3 filtros de orden 80, utilizando la totalidad de los Slices DSP , pero el resto de las operaciones deberían implementarse utilizando tablas LUT lo que provocaría posibles problemas de Timing al utilizar un reloj de alta frecuencia.

En la sección Frequency specifications, se diseña la respuesta en frecuencia del filtro, seleccionando la banda de paso, la frecuencia de muestreo que se utilizará para implementar el filtro, y la magnitud en banda de paso que se desea para el filtro. Al usar la casilla Frequency Units con la opción Normalized, la banda de paso será normalizada respecto al rango 0 a π utilizado normalmente en el diseño de filtros. Pero si se desea utilizar una frecuencia específica de corte, se debe cambiar la opción a la unidad correspondiente indicando, cuando aparezca la casilla, la frecuencia de muestreo que se utilizará para el diseño del filtro. El resto de las secciones de ésta pestaña no se necesitan modificar. En una primera instancia se debería ver una ventana como lo mostrado en la Figura 22.

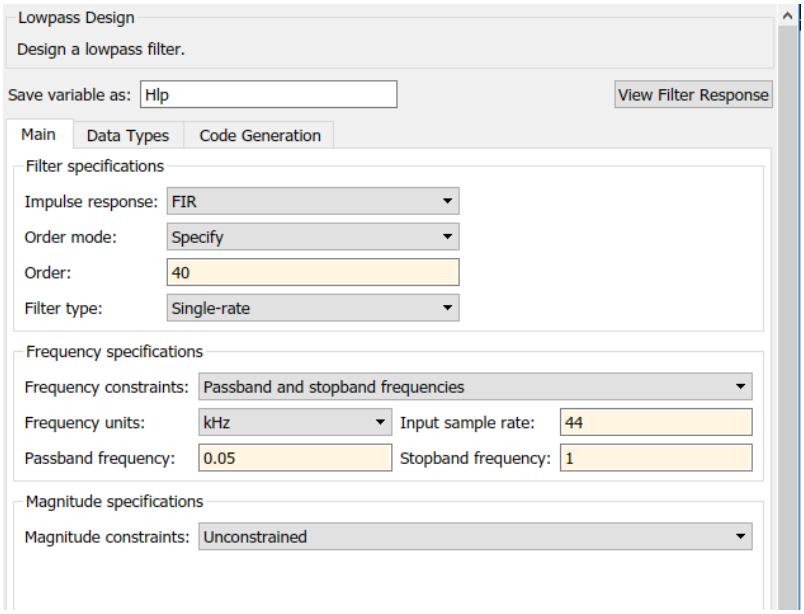


Figura 22: Especificaciones del filtro FIR pasa bajos básico.

En la pestaña Data Types, se especifican los tipos de datos con los que trabajará el filtro, en este caso la tarjeta Nexys4 DDR no posee cálculos con punto decimal en sus DSP, por lo tanto se deben eliminar todos los elementos que contengan punto decimal en el tipo de datos a utilizar. Para esto, primero se debe cambiar la opción de la casilla Arithmetic a Fixed Point, del cual se desprenderán un serie de opciones extra en las cuales se puede especificar la configuración interna del filtro, la cual por supuesto tampoco debe tener parte decimal, por lo tanto en la casilla Filter internals, se debe cambiar la opción a Specify precision , con lo que se podrá configurar todo lo relacionado al filtro físico implementado. En éste proyecto se usó la configuración mostrada en la Figura 23.

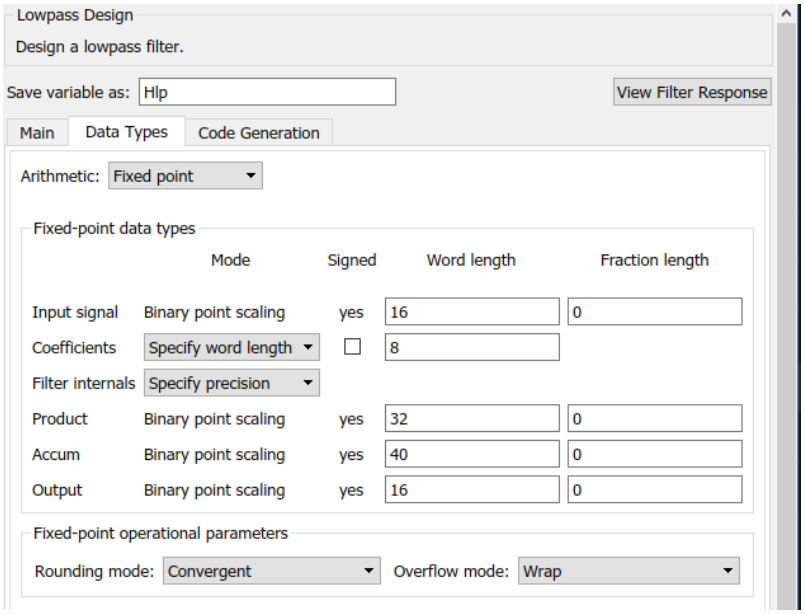


Figura 23: Configuración utilizada para la precisión de los datos del filtro.

Una vez terminado este proceso, se guardará una variable en Matlab la cual tendrá el diseño implementado del filtro, para obtener los coeficientes (necesarios para nuestro filtro en la FPGA) se debe usar

el comando *coewrite*, ingresando como parámetro del filtro que se desea obtener los coeficientes, con ésto se creará un archivo .coe (compatible con Xilinx) que contiene los coeficientes del filtro.

3.4. Configuración de módulos FIR IP CORE

Las configuraciones de éste IP core dependen de las configuraciones que se hayan realizado en los coeficientes realizados por Matlab, por lo tanto se debe tener en cuenta las consideraciones del Apéndice 3.3. Al abrir el programa en la pestaña izquierda donde aparecen todos los menús. En la sección Proyect Manager, se debe seleccionar la opción IP Catalog.

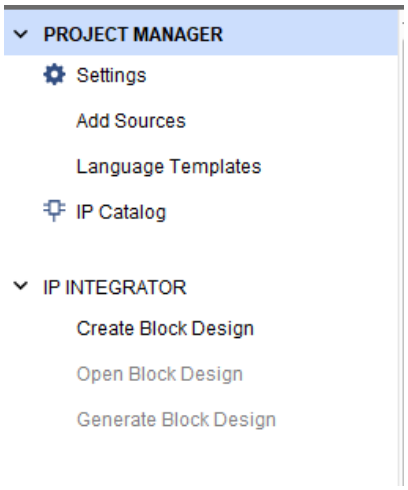


Figura 24: IP Catalog.

Al entrar en el menú del IP catalog, se debe ingresar en la barra de búsqueda la palabra clave "FIR", y seleccionar la opción mostrada en la Figura 25

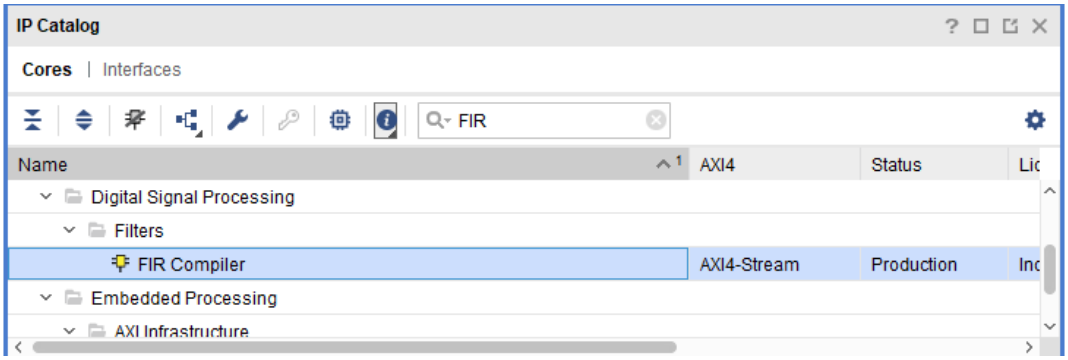


Figura 25: FIR IP core.

Una vez abierto el IP core, aparecerá una ventana como la mostrada en la Figura 26, donde se ven una serie de pestañas, se explicarán las funciones necesarias para implementar éste proyecto.

En la ventana FIR optiones se especifica el tipo de filtro que se utilizó, y desde donde provienen los coeficientes que se usarán en la implementación. En este caso el tipo de filtro que se usa en el proyecto es "Single Rate" y los coeficientes provienen de un archivo externo con extensión .coe (ver Apéndice 3.3). Al seleccionar el archivo .coe con los filtros, el IP detecta automáticamente el orden del filtro.

En la pestaña Channel Specifications hay una serie de opciones, pero en este caso nos concentraremos solo en la indicada como Hardware Oversampling specifications. En este caso se desea que la capacidad del filtro para entregar datos, sea la misma (o mayor) a la que el módulo ADC recibe datos, por lo tanto

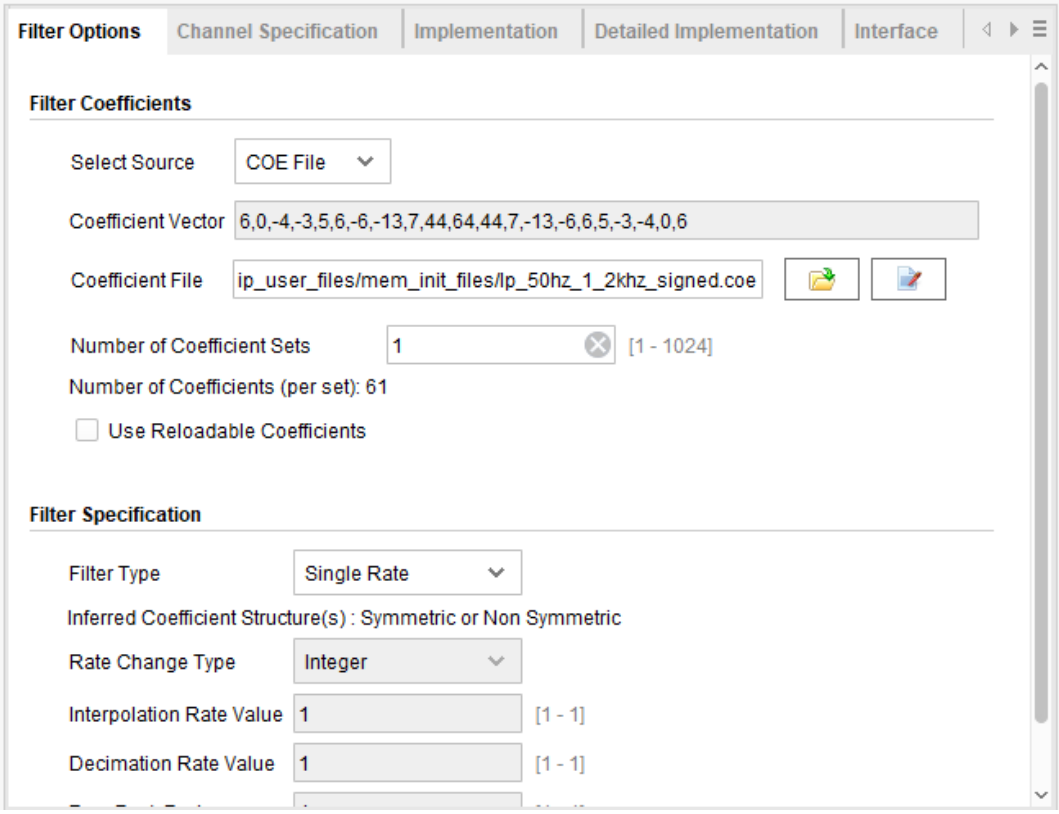


Figura 26: FIR IP core, ventana inicial.

se puede especificar la cantidad de ciclos de reloj requeridos, o en su defecto, la frecuencia de trabajo del módulo ADC.

En la pestaña Implementation, se deben indicar las características de los coeficientes del filtro, las características de los datos de entrada y las características que se desean a la salida del filtro. La configuración básica de los coeficientes requiere saber si son con signo y sin signo, y cuál es el largo (en bits) de los coeficientes. La configuración básica de los datos de entrada requiere saber si son con signo o sin signo y el largo de los datos de entrada. A la salida se puede especificar la resolución que se requiere para los datos, ya que se pueden utilizar todos los bits del filtro, como solo una parte de ellos.

Es de suma importancia que los coeficientes no posean parte fraccionaria al momento de generarlos en Matlab, ya que éste IP (y en general esta FPGA) no trabaja con datos que contengan parte decimal.
Las siguientes pestañas configuran la arquitectura que se desea en el filtro, si los requerimientos de diseño no son tan rigurosos no es necesario modificar los parámetros de éstas pestañas,

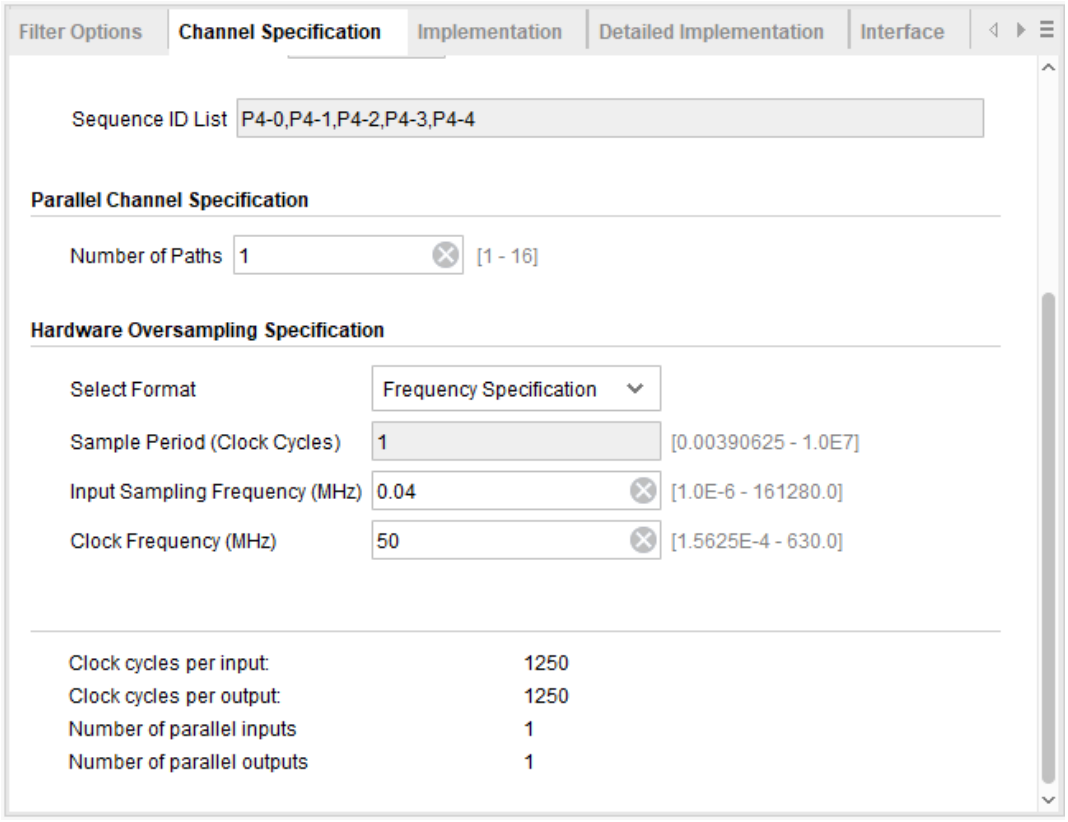


Figura 27: FIR IP core, Hardware Specifications.

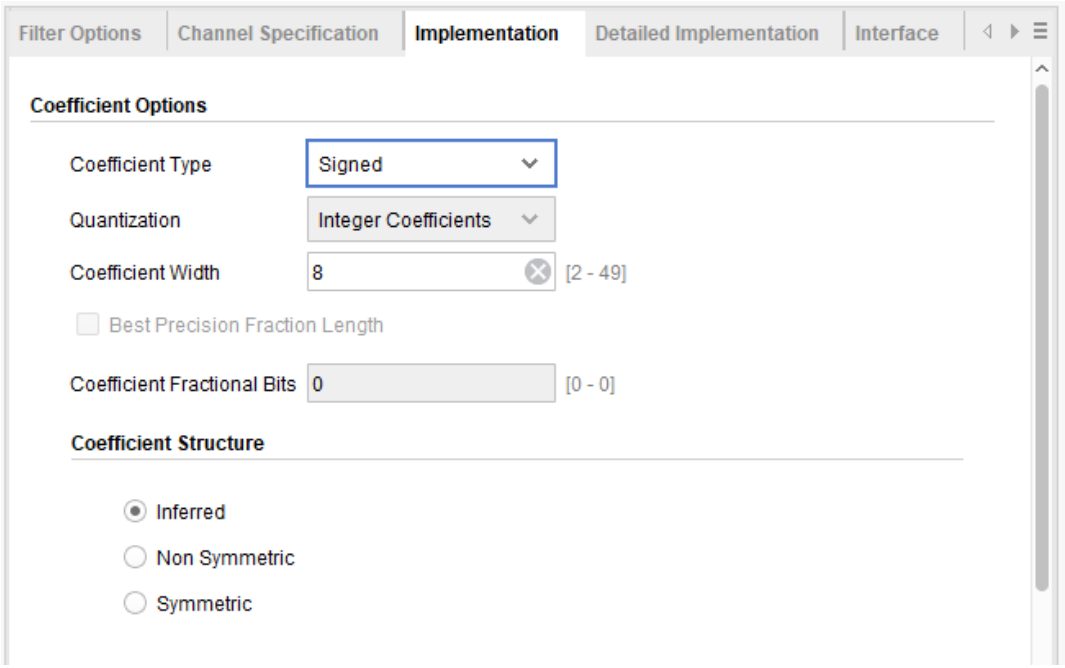


Figura 28: FIR IP core, Implementation.

3.5. Configuración del módulo ClockManager

En esta sección se explica la configuración básica para generar un reloj de alguna frecuencia específica utilizando la placa Nexys4 DDR y el programa Vivado 2017.4 de Xilinx. Al abrir el programa en la pestaña

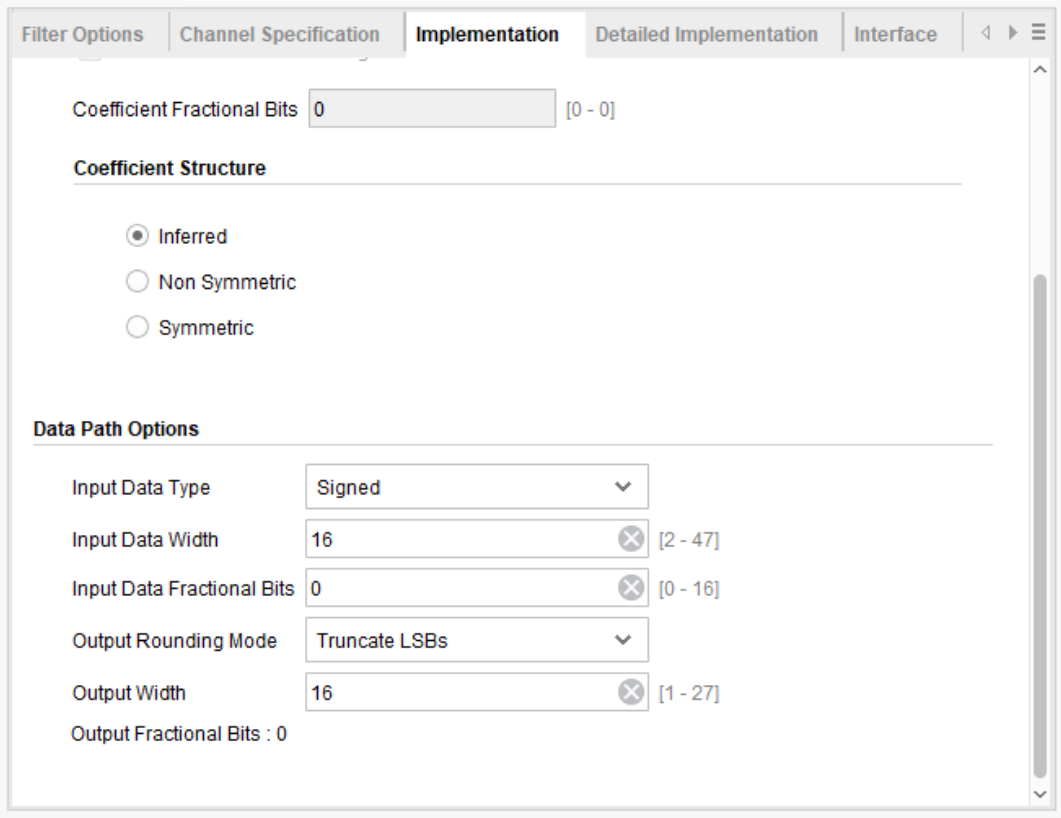


Figura 29: FIR IP core, Implementation.

izquierda donde aparecen todos los menús. En la sección Proyect Manager, se debe seleccionar la opción IP Catalog.

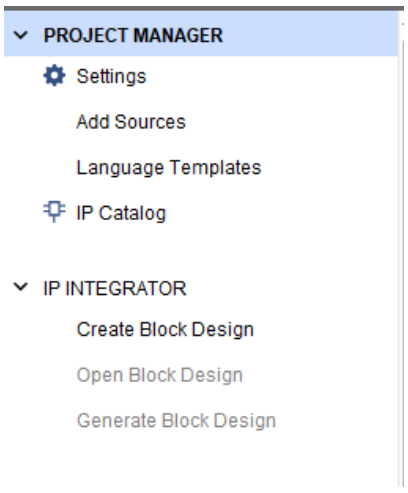


Figura 30: IP Catalog.

Al entrar en el IP catalog en la barra de búsqueda se debe ingresar la palabra clave "CLK", aparecen varias opciones por lo que se debe seleccionar la mostrada en la Figura 31.

Al seleccionar la opción mostrada en la Figura 31, aparecerá una ventana como la mostrada en la Figura 32, en este caso la única pestaña de importancia será la rotulada como "Output Clocks", ya que en ésta se selecciona la cantidad de relojes que se desea generar, el nombre y la frecuencia que tendrá cada uno de ellos.

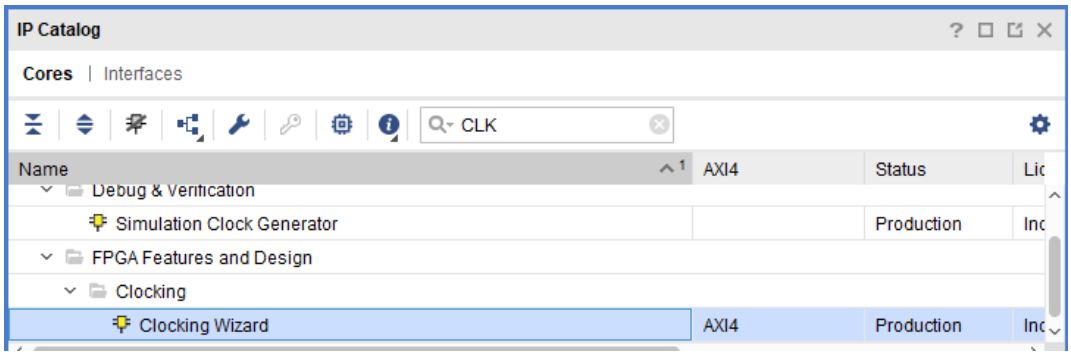


Figura 31: Búsqueda de la herramienta Clock Wizard.

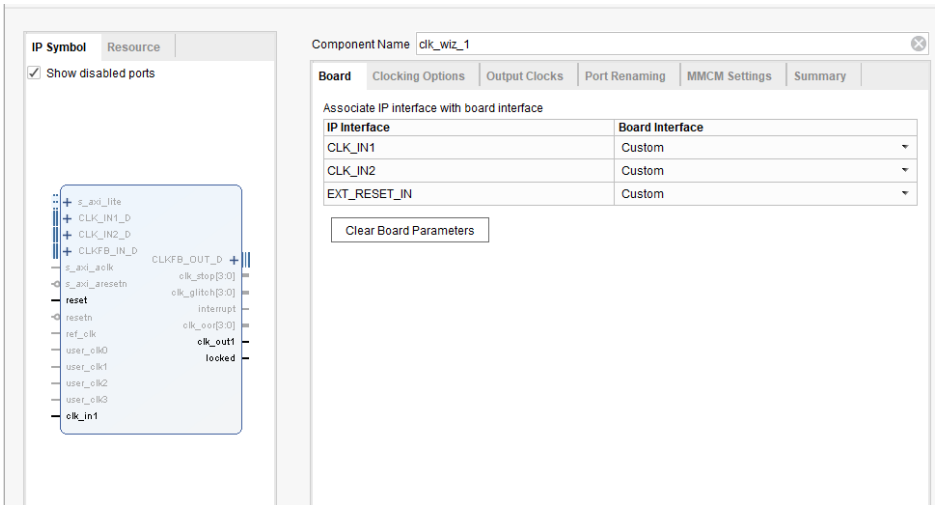


Figura 32: Primera ventana que aparece al abrir Clock Wizard.

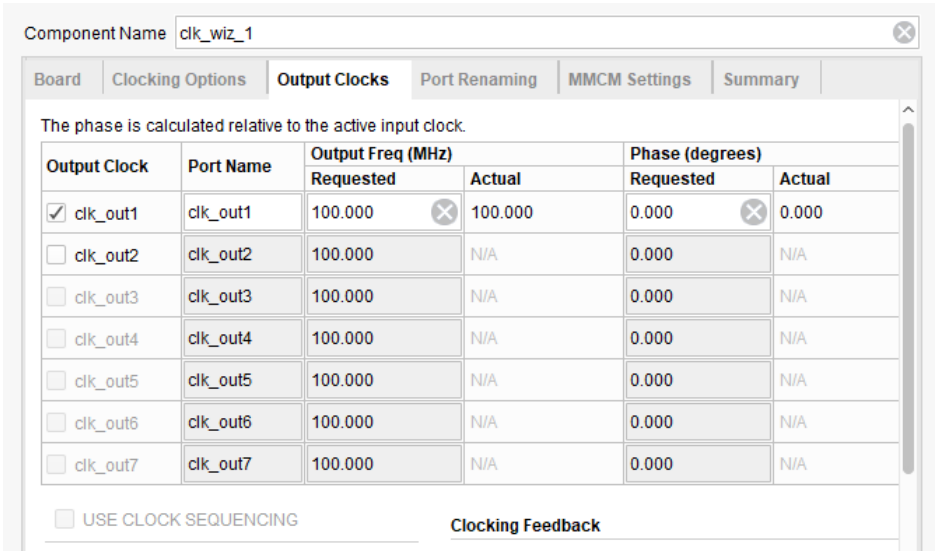


Figura 33: Ventana Output Clocks.

3.6. Configuración del módulo ADC

En esta sección se explica la configuración básica para generar un módulo ADC en cualquier proyecto utilizando la placa Nexys4 DDR y el programa Vivado 2017.4 de Xilinx.

Al abrir el programa en la pestaña izquierda donde aparecen todos los menús. En la sección Proyect Manager, se debe seleccionar la opción IP Catalog.

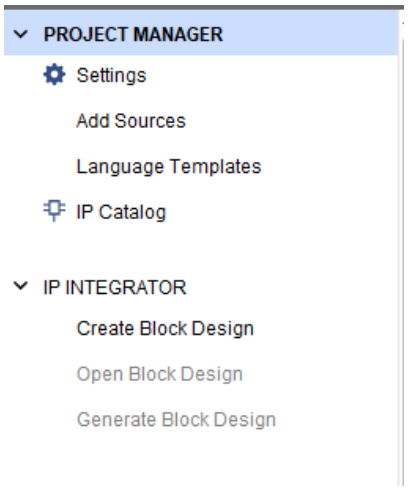


Figura 34: IP Catalog.

Al entrar en el menú del IP Catalog, en la barra de búsqueda se debe ingresar la palabra clave "ADC", de la cual se desprenderá una serie de opciones, por lo cual se debe seleccionar la opción "XADC wizard", tal como se muestra en la Figura 35.

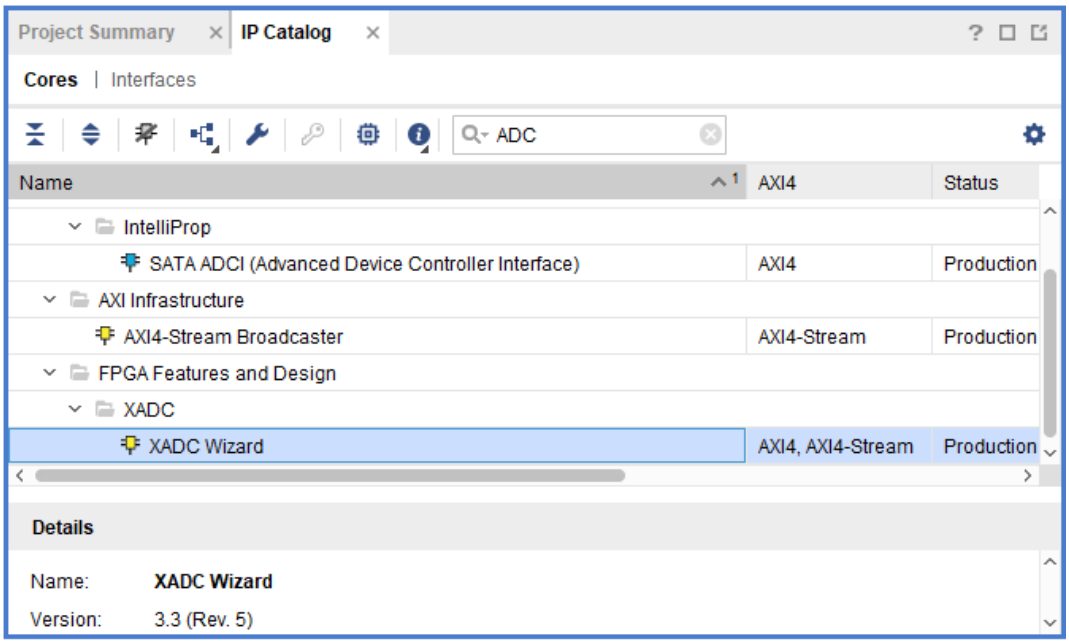


Figura 35: Selección del módulo XADC de la tarjeta Nexys4 DDR.

Una vez seleccionada la opción mencionada anteriormente se abrirá una pestaña como la mostrada en la Figura 36, donde se observan una serie de pestañas y opciones, a continuación se explica cada una de ellas.

Como se puede observar en la Figura 36, hay una gran cantidad de opciones que se deben configurar antes de utilizar el IP core del módulo ADC. A continuación se explican las opciones que fueron utilizadas en el proyecto, las demás están explicadas en el manual de usuario propocionado por Xilinx para la interfaz XADC de Artix 7 [**2**].

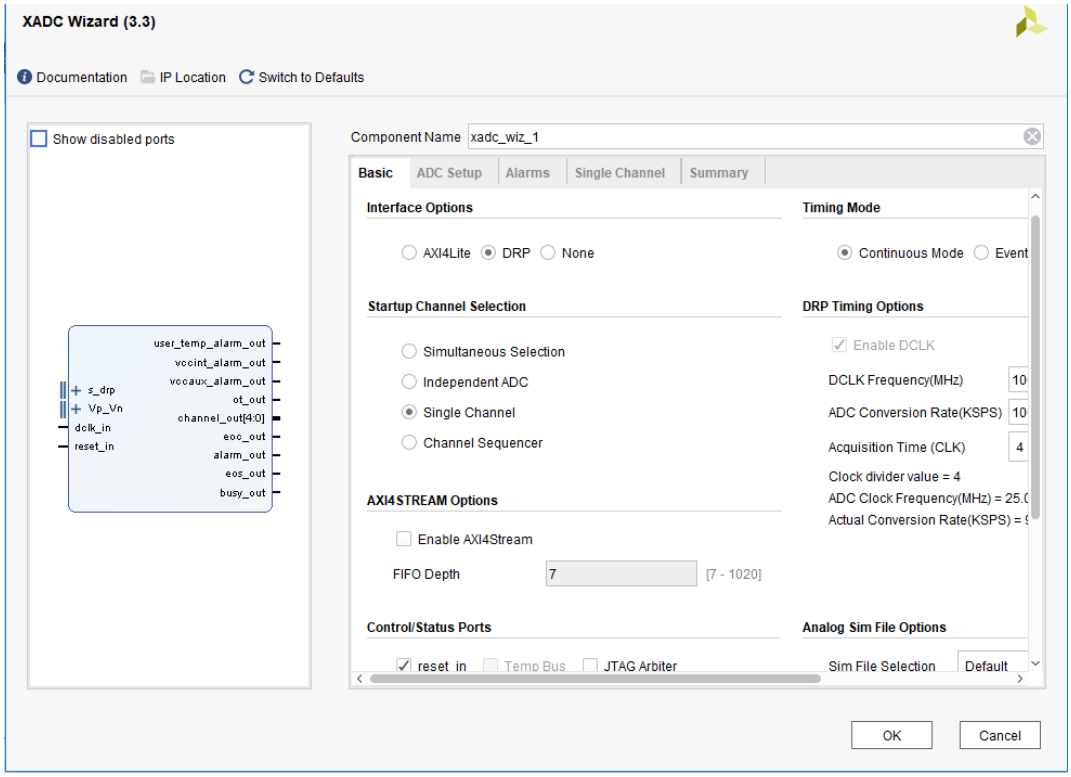


Figura 36: Ventana principal del IP core del módulo XADC.

Para empezar, en la pestaña "Basic" se configuran las características generales del ADC. Como la frecuencia de reloj, la frecuencia de muestreo deseada, el tiempo de adquisición (en ciclos de reloj), el tipo de funcionamiento que tendrá , etc.
En la sección Interface options se debe seleccionar el modo *DRP*, ya que éste permite el uso del ADC en un entorno HDL. El modo AXI4Lite se usa para conectar el ADC a un bus AXI, utilizado por una herramienta externa de Xilinx que permite implementar un microcontrolador dentro del la misma FPGA, por lo tanto no será usado. Al seleccionar None, no se puede acceder desde afuera, a los datos que obtiene el ADC.
En la sección Timing Mode, se escoge que tipo de conversión se hará (temporalmente), al seleccionar el modo continuo, el ADC estará obteniendo muestras constantemente cada vez que termine una, al seleccionar la casilla Event Mode, se tomarán datos dependiendo de un evento definido por el usuario, con lo que aparece una señal de control de eventos en la interfaz del ADC, para así informar el momento en que se desea tomar una muestra.
En la sección Startup Channel Selection,las opciones Simultaneous Selection y Channel Sequencer, permiten usar varios canales auxiliares de manera "simultánea", entendiendo que los canales son multiplexados internamente para obtener muestras por parte de todos los canales a una frecuencia específica. Las opciones Independent ADC y Single Channel, permiten utilizar un solo canal del ADC para muestrear, utilizando la totalidad de recursos de éste en el canal seleccionado. Para éste proyecto se usó la opción Single Channel, ya que no es necesaria la conversión simultánea de varias señales, pero ésto se puede extrapolar a varias señales replicando los módulos utilizados en cada uno de los canales que se desea modificar.
En la sección DRP timing options se especifica el reloj, la frecuencia de muestreo y el tiempo de adquisición de datos (en ciclos de reloj utilizado por el ADC) que se usarán internamente. Las demás opciones de ésta pestaña pueden ser dejadas por defecto, sin modificar el funcionamiento general del módulo. En la Figura 37,se muestra la configuración utilizada en éste proyecto.

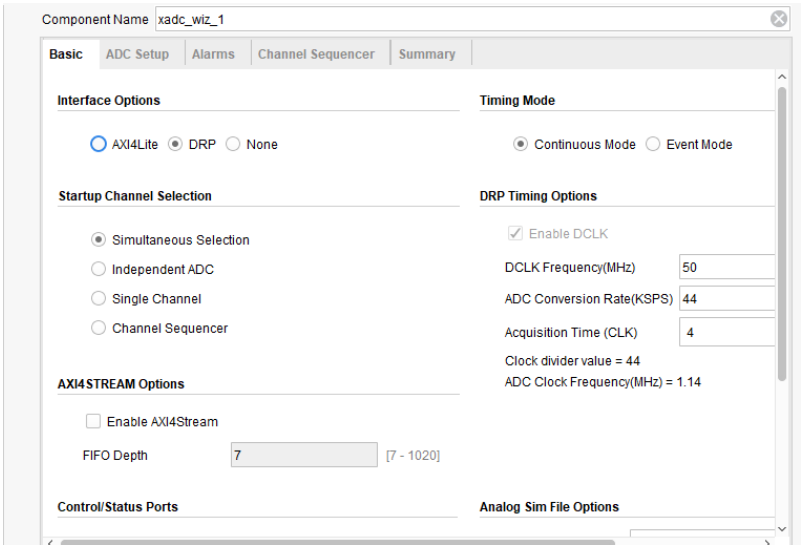


Figura 37: Configuración utilizada en éste proyecto para la pestaña Basic.

En la pestaña ADC setup, se especifican las condiciones de inicio del ADC, ya sea la calibración o el promedio de señales internas. Si no se consideran necesarias se pueden desmarcar todas las opciones de esta pestaña. En la Figura 38 se muestra la configuración utilizada en éste proyecto.

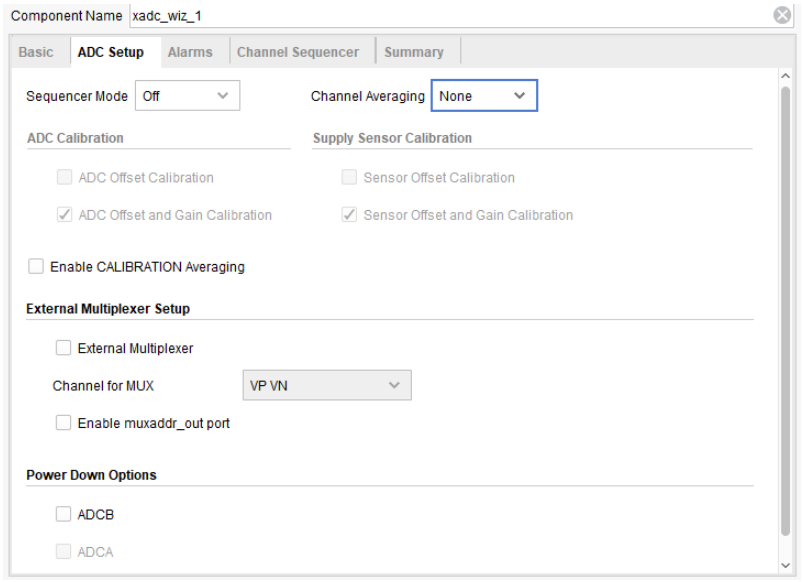


Figura 38: Configuración utilizada en éste proyecto para la pestaña Setup.

La pestaña Alarms, contiene una serie de señales de advertencia que se pueden utilizar internamente para no sobre explotar los recursos del ADC, y en caso de algún fallo interno. En este caso como el uso es tan sencillo, no es necesaria ninguna de estas señales por lo tanto se desmarcan todas. La pestaña siguiente varía dependiendo de la selección que se realice en la sección Interface options, ya que si se escoge un método que utilice más de un canal, la pestaña cambiará su nombre a Channel Sequencer, y permite la selección de varios canales (los que se desee utilizar). Como en éste caso se usó solamente uno, se explica la configuración para un solo canal, entendiendo que esto simplemente se replica para varios canales.

3.6.1. Configuración de un canal del ADC

Las configuraciones que aparecen en la pestaña de configuración de canales del IP core, se deben explicar para entender por qué se escogió la opción utilizada en éste informe. En primer lugar, los canales poseen 2 funcionamientos básicos, Unipolar y Bipolar. En configuración Unipolar la entrada positiva del módulo debe tener un Offset (recomendado de 500mV) el cual posee montado sobre sí, la señal que se desea medir. Esto es ya que la configuración unipolar lee voltajes entre 0 y 1V. La función de transferencia digital de la configuración unipolar se muestra en la Figura 39.

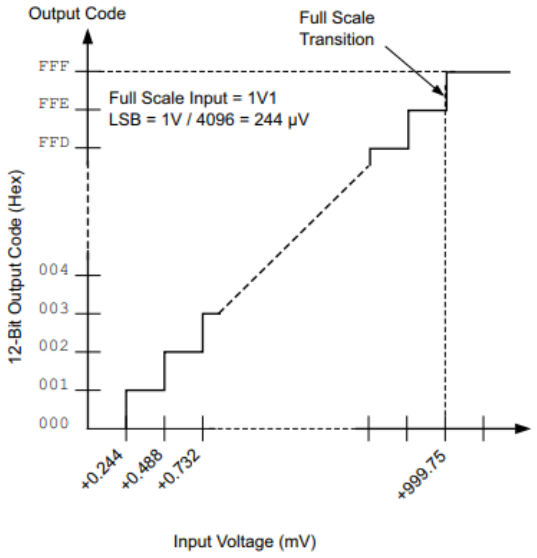


Figura 39: Función de transferencia digital para el modo Unipolar.

Por otra parte, la configuración del modo Bipolar puede leer voltajes entre -0.5V y 0.5V, por lo tanto no es necesario agregar un Offset a la señal, pero se trabaja con números negativos internamente, agregando lógica adicional al diseño. La función de transferencia digital del modo Bipolar se muestra en la Figura 40.

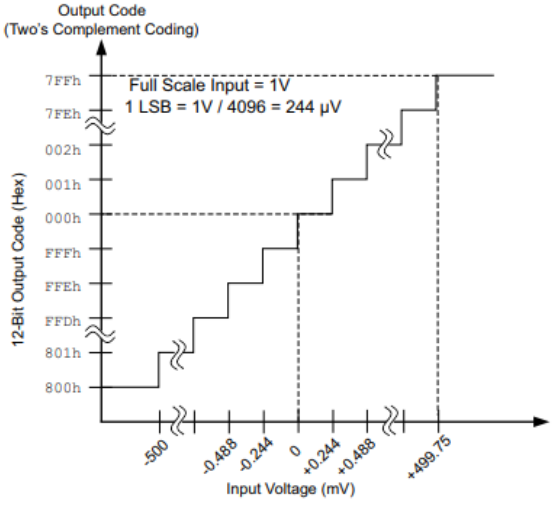


Figura 40: Función de transferencia digital para el modo Bipolar.

La FPGA Nexys4 DDR posee una serie de canales internos configurables por el IP core, pero los canales físicos utilizables (que poseen comunicación hacia el exterior) son tan solo 4: Vaux11, Vaux3, Vaux10 y

Vaux2. Por lo tanto, para leer señales desde el exterior en modo Single Channel (unipolar o bipolar) se debe utilizar uno de éstos canales.

3.7. Cálculos generales para el funcionamiento del módulo PWM

La placa Nexys4 DDR posee un controlador de audio de un solo canal, el cual corresponde a un filtro Butterworth Salley-Key de orden 4, el cual se muestra en la Figura 41. Este filtro se utiliza para filtrar una señal PWM de hasta 10kHz, por lo tanto una frecuencia de muestreo mayor o igual a 20kHz es suficiente para generar una señal PWM de salida con funcionamiento normal. Ya que el módulo ADC usa se usa con una frecuencia de muestreo de 40kHz y las señales usadas son de 12bits de resolución, se debe usar una frecuencia de reloj que cumpla con:

$$F_{clock} \geq 44000 \cdot 2^{12} = 180224000[Hz]$$

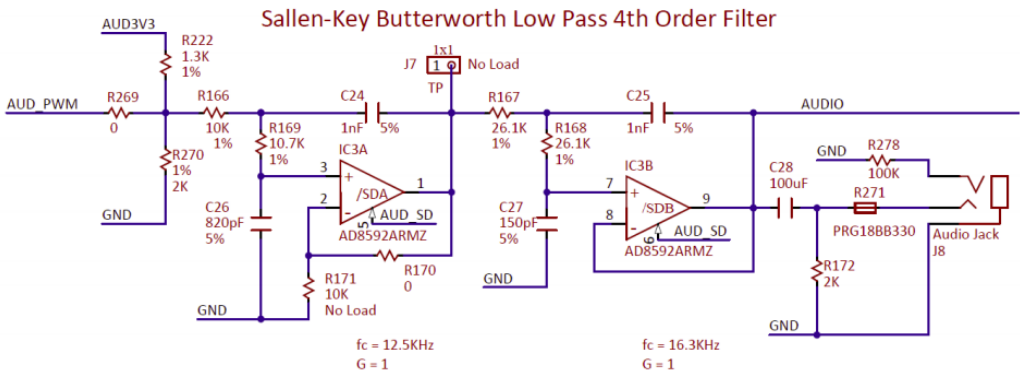


Figura 41: Diagrama circuital del filtro pasa-bajos de la tarjeta Nexys4 DDR.

En el transcurso del proyecto se llevó este reloj al límite, encontrando que los niveles realizables por la tarjeta en cuanto al reloj de funcionamiento normal para éste módulo, rondan los 300MHz. Por lo tanto en la implementación se usa este valor, demostrando la capacidad de la tarjeta.
Cabe destacar que , como se mencionó anteriormente, el reloj seleccionado para realizar la tarea de generar una señal PWM que satisfaga las necesidades de proyecto, debe ser un múltiplo entero del reloj utilizado para tomar señales desde el módulo ADC, ya que si no se cumple con ésta regla se generar problemas de **Timing** al momento de hacer el cruce de los dominios de reloj.
Finalmente para un funcionamiento correcto del módulo PWM (ya que se registra una gran cantidad de ruido al hacer la conversión) , se decide usar una frecuencia *alta* de 300Mhz para el módulo PWM, esto desplaza los armónicos de ruido de alta frecuencia a una frecuencia dada por

$$300000000/2^{12} \approx 73Khz$$

4. Conclusiones

El documento intenta presentar una base para el diseño de un procesador de Audio utilizando la tarjeta Nexys4 DDR. La cantidad de complicaciones presentadas al momento de implementar el diseño en ésta tarjeta es bastante grande, partiendo por las limitaciones de los bloques DSP que posee. Para éste tipo de aplicaciones se esperaría usar una FPGA que posea bloques DSP con capacidad de cómputo en punto fijo (o en su defecto punto flotante). Se pudo comprobar que la tarjeta posee limitaciones físicas en cuanto al las frecuencias de reloj y la cantidad de relojes que se pueden generar simultáneamente, ya que el diseño no fue tan complejo ésto no jugó un rol importante dentro de las limitantes de diseño, pero puede jugar un papel importante en diseños que requieran múltiples relojes de alta frecuencia. Los principales problemas de Timing que se presentaron, fueron al hacer cálculos matemáticos con varios DSP simultáneamente, por lo que se optó por usar un reloj de una frecuencia relativamente baja (50Mhz) para realizar cómputos y para desarrollar los módulos generales del diseño. También se comprobó la limitante en cuanto a la resolución que posee el driver PWM de la tarjeta, ya que para generar señales de más de 12 bits de resolución se requieren relojes de frecuencias que no son alcanzables por ésta FPGA. Además se observó que si la frecuencia utilizada para generar dichas señales PWM está en el límite mínimo requerido, se introduce ruido no deseado de alta frecuencia en la salida, el cual es bastante difícil de eliminar.