Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Cuautitlán

Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA

T E S I S

QUE PARA OBTENER EL GRADO DE:

Ingeniero en Telecomunicaciones, Sistemas y Electrónica

PRESENTA:

Gutiérrez Zacarías Roberto Carlos

DIRECTOR DE TESIS:

Ing. José Luis Barbosa Pacheco

Cuautitlán Izcalli, Estado de México, 2020

VNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA escrito

Que presenta el pasante: **Roberto Carlos Gutiérrez Zacarías**
Con número de cuenta: **306345531** para obtener el título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de Octubre de 2020.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Juan González Vega	
SECRETARIO	Ing. José Luis Barbosa Pacheco	
1er. SUPLENTE	Ing. Jorge Ramírez Rodríguez	
2do. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/cga*

VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA escrito

Que presenta el pasante: **Roberto Carlos Gutiérrez Zacarías**
Con número de cuenta: **306345531** para obtener el título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de Octubre de 2020.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Juan González Vega	
SECRETARIO	Ing. José Luis Barbosa Pacheco	
1er. SUPLENTE	Ing. Jorge Ramírez Rodríguez	
2do. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/cga*

VNIVERSIDAD NACIONAL AVTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

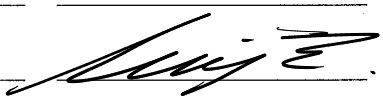
Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA escrito

Que presenta el pasante: **Roberto Carlos Gutiérrez Zacarías**
Con número de cuenta: **306345531** para obtener el título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de Octubre de 2020.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Juan González Vega	
SECRETARIO	Ing. José Luis Barbosa Pacheco	
1er. SUPLENTE	Ing. Jorge Ramírez Rodríguez	
2do. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/cga*

VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA escrito

Que presenta el pasante: **Roberto Carlos Gutiérrez Zacarías**
Con número de cuenta: **306345531** para obtener el título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de Octubre de 2020.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Juan González Vega	
SECRETARIO	Ing. José Luis Barbosa Pacheco	
1er. SUPLENTE	Ing. Jorge Ramírez Rodríguez	
2do. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/cga*

VNIVERSIDAD NACIONAL AVTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Tratamiento de audio digital con interfaz AXI-Stream e implementación en FPGA escrito

Que presenta el pasante: **Roberto Carlos Gutiérrez Zacarías**
Con número de cuenta: **306345531** para obtener el título de: **Ingeniero en Telecomunicaciones, Sistemas y Electrónica**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 07 de Octubre de 2020.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	Mtro. Jorge Buendía Gómez	
VOCAL	Ing. Juan González Vega	
SECRETARIO	Ing. José Luis Barbosa Pacheco	
1er. SUPLENTE	Ing. Jorge Ramírez Rodríguez	
2do. SUPLENTE	Mtro. Leopoldo Martín del Campo Ramírez	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/cga*

"Siempre he sentido que un nivel saludable de inseguridad te mantiene en curso. Si comienzas a sentirte demasiado bien contigo mismo, ten cuidado, estás en problemas"
Frank Gehry

Agradecimientos

El llegar a este punto en mi vida se lo debo en gran parte a mi Padre que siempre confió en mi, me dio su apoyo hasta donde pudo y jamas dejó de creer en lo que yo hacia, por supuesto también a mi Madre que me llevo en mis primeros pasos he hizo posible el estar aquí ahora, a mis hermanos Omar, Atenas, Pablo y Cintia que hacen más llevadera la vida al estar a su lado, a mis abuelos que siempre llevo en mi corazón, al resto de mi familia que me impulsan en cierta manera , a mis amigos que les debo tanto, sin ellos no podría ser quien soy, tengo una lista muy grande que no podría recordar y agradezco mucho esa situación, también a las personas que me han marcado de una manera en la que solo me hacen recordar ser mejor persona y corregir mis errores, a mis profesores que formaron parte de mi proceso educativo, sobre todo ahora al Ing. José Luis Barbosa Pacheco quien me ha alentado a seguir profundizando en el conocimiento y ver que hay que esmerarse para descubrir el sentimiento de un logro conseguido, al Dr. Carlos González Calvo quien me instruyo de forma breve pero muy concisa en mi estadía en la madre patria e hizo posible gran parte de este proyecto, entre las instituciones no hay ninguna a la que le deba más que a mi hermosa UNAM, me ha dado todo lo que tengo ahora, y me siento muy orgulloso de poder pertenecer a esta casa de estudios.

Índice general

Índice de tablas

Índice de figuras

Resumen

En la actualidad el nivel de integración alcanzado con el desarrollo de la microelectrónica ha hecho posible desarrollar sistemas completos dentro de un solo circuito integrado SOC (System On Chip), con lo cual se han mejorado de manera notoria características como velocidad, confiabilidad, consumo de potencia y sobre todo el área de diseño. Esta última característica ha permitido observar día a día cómo los sistemas de uso industrial, militar y de consumo han minimizado el tamaño de sus desarrollos. Una tendencia en el diseño de los ASIC (Application Specific Integrated Circuit) proviene de una innovadora propuesta, que sugiere la utilización de celdas programables preestablecidas e insertadas dentro de un circuito integrado. Con base en esta idea surgió la familia de dispositivos lógicos programables los PLD (programmable logic device), cuyo nivel de densidad de integración ha evolucionado a través del tiempo. Iniciaron los PAL (Programmable array logic) hasta llegar al uso de los CPLD (Complex programmable logic device) y los FPGA (Field-programmable gate array), los cuales, dada su conectividad interna sobre cada una de sus celdas, han hecho posible el desarrollo de circuitos integrados de aplicación especifica de una forma mucho más fácil y económica, para beneficio de los ingenieros encargados de integrar sistemas. [20] Al tener una idea de cómo poder crear teóricamente un sistema, el trabajo practico de elaborar el dispositivo se vuelve más técnico y con obvias limitaciones dependiendo del tipo de circuitos utilizados. Así mismo el costo es la primera limitante para el desarrollador, siempre se trata de reducir o limitar los componentes costosos, que en su mayoría se utilizan en las altas necesidades de procesamiento. Los FPGA tienen una gran ventaja, ya que diseñar con estos supone precisamente enfocarse a la idea del sistema, ya que el programa CAD (computer-aided design) ayudará a sintetizar el

código de descripción HDL (Hardware Description Language) obteniendo un circuito electrónico equivalente, además de hacer procesamiento multihilo, disminuyendo el tiempo en que se realizan las tareas como sucede en otros dispositivos. Se requiere diseñar un proceso electrónico que maneje audio digital con algunos de los efectos básicos, los cuales trabajan en el dominio dinámico, dominio de la frecuencia y dominio del tiempo. Estos procesos, al estar manejando señales analógicas que se convierten en señales digitales y que se modifican en tiempo real, poseen una alta taza de transferencia por lo que necesitan una alta capacidad de manejo de datos en volumen y velocidad, para estas características mencionadas se utilizan procesadores digitales de señal DSP (Digital signal processor). Los DSP´s que incluye el FPGA, aunque son ASIC (Application-specific integrated circuit), forman otra estructura más compleja que se fundamenta en un componente SoC (System on a chip), pues en él se incluyen todos los requisitos necesarios para eliminar la necesidad de agregar componentes adicionales, desde hace tiempo se trabaja en la idea de hacer que estos circuitos de "funciones generales" se puedan producir en masa[21], pues facilitaría la selección de hardware al comienzo de un diseño electrónico ya que contendría todas las funciones básicas en un solo circuito, modificando así el costo de la producción de los mismos y la facilidad de especificar el funcionamiento de estos SoC solamente por Software. Según Rajeev Jayaraman, director de desarrollo de herramientas físicas en XILINX®(2000-2002), es allí donde los FPGA son buena opción para sustituir a los ASIC´s, Pues no se tendría el trabajo extra al unir el FPGA y el DSP, estos SoC´s NO contienen un sistema NRE(Non-recurring engineering) el cual eleva demasiado su costo como se ha visto en los ASIC [7]. La principal motivación de este trabajo es aportar una vista sobre nuevas herramientas para las personas que desarrollan o hacen diseños electrónicos, que facilitan la generación de soluciones a problemas informáticos de diferente complejidad , donde el alto uso de datos es un requisito indispensable, el audio es un tema selecto por gusto propio, pero también es una demostración de la amplia variedad de problemas a abordar, escogiendo en cierta medida, los recursos tanto físicos como intelectuales que integran este trabajo para demostrar el proceso de implementación y bases para marcar la diferencia entre ASIC y FPGA.

Objetivos

Objetivo general

- Implementar módulos en un FPGA que manejen audio digital conectados con una interfaz de comunicación AXI-Stream, verificar su funcionamiento y observar las mejoras en velocidad y procesamiento de un sistema digital tradicional al manipular el audio, de esta forma.

Objetivos particulares

- Utilizar la metodología top-down para diseñar el sistema, verificando el flujo de datos y el de control.
- Manejar recursos de Xilinx® (FIR compiler) en la tarjeta de desarrollo BASYS 3™ con FPGA Artix-7 verificando su funcionamiento de acuerdo con el tipo, la respuesta, orden y número de coeficientes en los filtros aplicados.
- Implementar los bloques digitales que constituyen los efectos de audio que trabajan en diferentes dominios, como son, tiempo, frecuencia y rango dinámico. Estos efectos repiten, atrasar, eliminan, cortan, amplían, etc., señales de audio y crean resultados (modificaciones de audio) con cualidades propias de procesadores de audio comerciales.
- Verificar la diferencia en costos, calidad de procesamiento, cantidad de funciones y otras características básicas de este procesador contra los procesadores o efectos individuales comerciales.

Estado del arte

Se realizo una búsqueda en literatura con los temas desarrollados en está tesis a través de bases de datos, para que con antecedentes de estudios, investigaciones, artículos o proyectos relacionados , permita forjar una base solida para presentar, desarrollar, concluir y discutir pertinentemente de acuerdo con los resultados obtenidos de los trabajos anteriores, mostrando las capacidades de las herramientas para el diseño electrónico.

El primer trabajo de tesis “FILTRO DIGITAL TIPO FIR EN UN FPGA”[15] trata el filtro fir diseñado en FPGA, en la parte teórica se relacionan tres etapas importantes sobre las herramientas matemáticas que se llevan acabo, análisis de Fourier, la transformada Z, correlación y convolución. Se involucran todos los posibles usos para el tipo de filtrado que se lleva acabo, se lista el procesamiento de imágenes, instrumentación y control, audio, aplicaciones biomédicas, aplicaciones de telecomunicaciones. La fuente trata del año 2000, en esos años estos arreglos de compuertas programables en campo no eran muy conocidos en México por lo que sugieren el estudio de los ambientes de desarrollo para poder reducir tiempos en diseños de proyectos, también sugiere utilizar procesadores digitales de señal DSP para poder ampliar las etapas el mismo tipo de filtro, el FPGA utilizado es XC4000X Series de Xilinx®. Para calcular los coeficientes se ayuda con un programa de computadora "laboratorio de matemáticas", realizando un Filtro tipo Butterworth de orden 2, el tipo de dato es un entero positivo en base hexadecimal de 8 bits, se utiliza el FPGA como un dispositivo externo y es programado desde un ambiente de desarrollo por lo que no se utiliza VHDL para diseñar, pero si es sugerido.

“DISEÑO DE UN MICRO SISTEMA PROGRAMABLE PARA EFECTOS DE AUDIO DIGITAL USANDO FPGAS” [22], en este articulo se diseña un micro sistema usando un

procesador de propósito especifico e interfaz táctil con LCD, en dicho artículo se emplea el filtro FIR utilizando el compilador FIR de Quartus II, teniendo en común con el anterior trabajo que utilizan una interfaz gráfica de PC que se comunica con el FPGA y usando un programa de simulación con algunos constructores de DSP, se prueba la funcionalidad del mismo. Ambos utilizan datos de 16 bits con una tasa de muestreo de 195,62 MHz, puede verse que se dividen los efectos por campos, aquellos que se encuentran en un procesamiento por retardos, otros basados en el dominio dinámico y por último el procesamiento en el dominio de la frecuencia. El sistema esta descrito en VHDL, desarrollado en el FPGA EP2C70F896C6, donde se concluye que debido a la reconfigurabilidad del FPGA, el diseño puede ser llevado a tomarse como una IP core ya que la alta tasa de muestreo hace que el audio entregado sea de buena calidad, e incluso podría comercializarse.

Un tercer trabajo,"ESTUDIO DE EFECTOS DE AUDIO PARA GUITARRA E IMPLANTACIÓN MEDIANTE DSP" [30], trata como tema principal el procesamiento digital del audio en tiempo real, con la finalidad de ser una herramienta diseñada de un profesional para el uso de otro, como lo son los músicos, el estudio teórico se desarrolla observando los modelos de los efectos vistos desde el tratamiento digital de señales, se hace uso del software "laboratorio de matemáticas" para los diagramas de bloques y el lenguaje C para el DSP, en una plataforma llamada ADSP 21000, con una frecuencia de muestreo de 18.9 KHz, el trabajo menciona que se pueden agregar mejoras a los efectos como añadir osciladores de efectos como el phaser, agregar modificaciones de parámetros en tiempo real, así como agregar hardware. se observa que el campo del desarrollo en cuanto a los FPGA, Filtros y Procesamiento de audio es basto y tiene diferentes vertientes, se podría basar en teoría muy profunda para la realización de la etapa de filtrado o solo utilizar IP cores que estén disponibles y eliminar tiempo de desarrollo, también establecer frecuencias de muestreo muy grandes para soportar cualquier ancho de banda audible o reservar el muestreo solo a frecuencias producidas por instrumentos como una guitarra, agregar mezclas de lenguajes descriptivos o de programación para intercalar procesos en donde resulte más conveniente.

1

Estructura general del sistema

La complejidad de la tecnología actual hace que realizar una tarea no sea posible con el manejo de un solo método , si no, con la suma de una multiplicidad de estos, esto quiere decir que se realiza con un sistema. como por ejemplo en automóviles, receptores de radio, reproductores de vídeo, etc. Cuando se generan problemas en donde no se puede encontrar una solución generalizada, necesita establecer un conjunto de métodos que lleven a la solución general, es la forma en que se realizan los proyectos a medida (Ad-hoc).
Entonces existe un enfoque de **sistema**, dado un determinado objetivo, encontrar caminos o medios para alcanzar la solución general, requiere que el diseñador considere soluciones posibles y elija las que prometen optimización, con máxima eficiencia y mínimo costo en una red de interacciones compleja[6].

En la figura 1-1 se muestra el proceso general del diseño electrónico, si la solución no necesita algún componente como memorias o circuitos que necesiten de modificación o programación vía software, se podría prescindir de ello.

Dos alternativas de diseño, llamadas top-down y bottom-up, son usadas en la construcción de sistemas complejos. En la primera el diseño comienza desde la parte superior(top), con la premisa que se puede acceder a cada subcomponente necesario en el sistema de manera global. La metodología bottom-up toma como premisa que el sistema global no es fácil de obtener y parte de componentes básicos hasta llegar al global, esta forma es inversa comparada con la

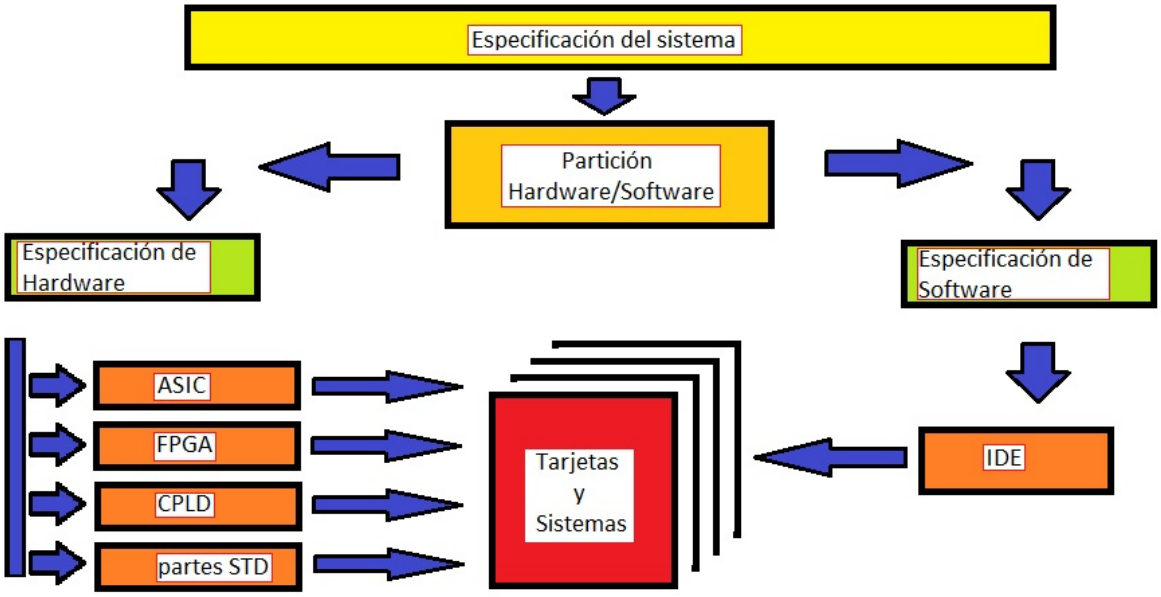


Figura 1-1: Estructura del diseño electrónico.

forma top-down[13].

Esta forma de diseño top-down toma ventaja en la rapidez de implementación al poder distribuir las tareas de creación y verificación de cada bloque propuesto, al tomar componentes individuales y conocer su función, se pueden asignar las tareas y trabajar paralelamente para que al final se integren al concepto de solución general, reduciendo tiempo de desarrollo.

La solución con estas metodologías puede ser mezclada, una vez que se identifico el problema y puede partir desde el sistema top o desde los componentes base, como se verá más adelante.

1.1. Concepto de solución por Top-Down

De manera intuitiva, la metodología es utilizada solicitando primero los requisitos del sistema en general, se elaboran niveles complementarios que deben cumplirse conforme se avanza en la unificación de los módulos, también se elaboran más fácilmente los módulos a desarrollar, el proyecto termina hasta llegar al nivel más bajo(down), también es común reutilizar algunos módulos que recrean una misma tarea.

Figura 1-2: Proceso de diseño Top-down, fuente:Aplicación y comparación de la metodología de diseño Top-down y Bottom-up, VeronicaP. R.Muñoz,2009.

De esta forma es más fácil medir el desarrollo o avance del proyecto, el sistema se hace totalmente Ad hoc, ya que se utilizan solo los sistemas o componentes simples base, la visión no se hace necesaria en el conocimiento de todos los desarrolladores, porque solo es indispensable al dividir las funciones de un nivel en módulos y por lo tanto cada diseñador hace una parte diferente en referencia a una idea general.

1.2. Concepto de solución Botton-Up

Se debe fijar detalladamente en las funciones y posibilidades que otorgan los componentes simples, pues de ellos dependerá la realización de la solución general, intuitivamente las entidades electrónicas simples se acumulan para generar sistemas pequeños que a su vez forman una estructura más grande con otros sistemas iguales, hasta avanzar al nivel(top), en ese ascenso el diseñador puede ir agregando características de cualquier componente o sistema pequeño y al final automáticamente se agregarán características extras, haciendo uso de casi todas las

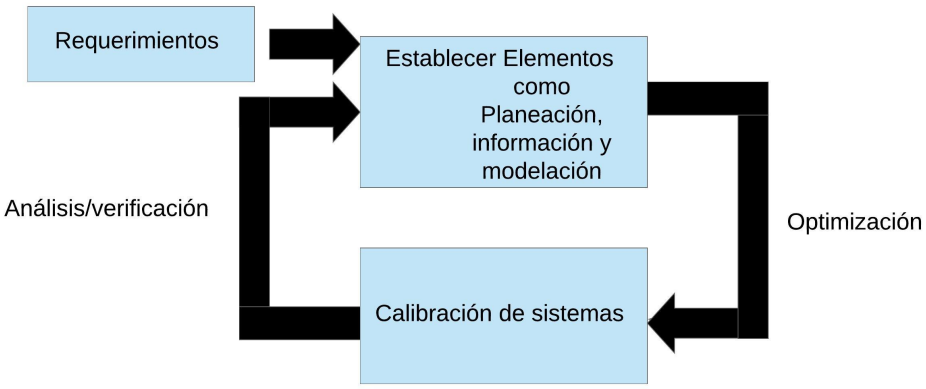


Figura 1-3: Ciclo de diseño Top-down.

capacidades de los elementos simples, este enfoque es bueno cuando se requiere expandir el sistema. Los diseñadores elaboran módulos por cada función contenida en los dispositivos base.

La metodología tiene desventajas como se menciona en[29], “Bottom-up projects, however, are hard to manage. With no overall vision, it is hard to measure progress.”, que dice “sin embargo, los proyectos Bottom-up son difíciles de administrar. Sin una visión general es complicado medir el progreso”, es entendible esta parte, ya que al no tener la idea del sistema completo, las lineas de avance en que se trabaja, pueden no ser las indicadas para el objetivo en general además de que todo el equipo debe estar en una constante retroalimentación para no desviar el fin común. Por tanto se cometen más fácilmente errores por la mala comunicación interpersonal.

1.3. Concepto de solución Híbrido

Ambas soluciones Top-down y Bottom-up tienen ventajas y desventajas, pero se debe tener en cuenta que no son incompatibles entre sí, si se comienza un proyecto con Top-down se puede entregar el desarrollo de como se llevará acabo el mismo, hasta ese punto se toma la primer solución. Después se debe optar en cada nivel del proyecto generar una idea en los diseñadores de las necesidades requeridas, haciendo que los segundos tomen la decisión de Botton-up, llevando

Figura 1-4: Proceso de diseño Bottom-up, fuente:Aplicación y comparación de la metodología de diseño Top-down y Bottom-up, VeronicaP. R.Muñoz,2009.

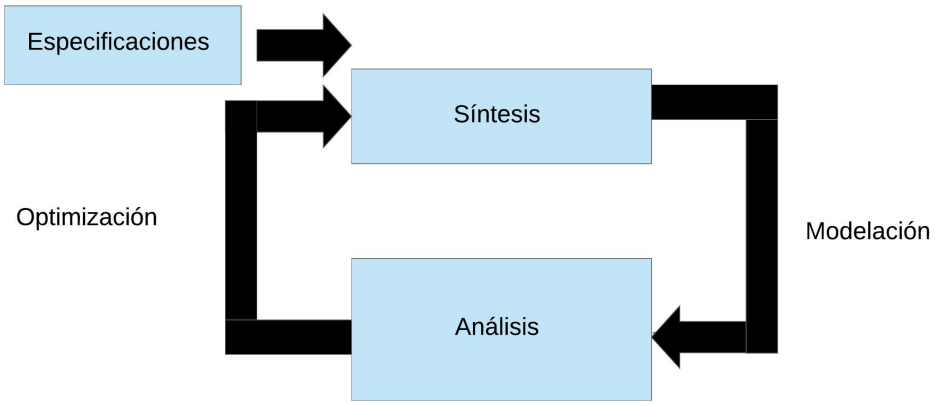


Figura 1-5: Ciclo de diseño Bottom-up.

así la mejor forma posible de construcción modular, puesto que se utilizan las habilidades de los desarrolladores en el diseño estructurado.

Esto es para evitar el código espagueti (Código con estructura de control muy complejas y casi incomprensible.). Además, este método de solución híbrido es perfectamente representado con la construcción de un muro, primero se verifican las características del mismo, para poder saber cuantos ladrillos usar y como colocarlos, pero la construcción empieza desde abajo para arriba, delimitando las características de como serán puestos los ladrillos y usando la habilidad que tiene el trabajador al colocarlos. [29]

Así pues, en esta tesis se decidió implementar el método híbrido, a continuación, en la figura 1-6, se observa de forma general el módulo PMOD I2S2 de Digilent®, se inicia desde un enfoque Bottom-Up ya que se revisan primero las herramientas físicas de que se dispone, estas herramientas son la base del proyecto, en este caso el PMOD ya mencionado y la Tarjeta de desarrollo Basys 3™ que incluye un FPGA Artix-7. Teniendo en cuenta estas dos herramientas, se conceptualiza el trabajo que sigue para el avance general del sistema.

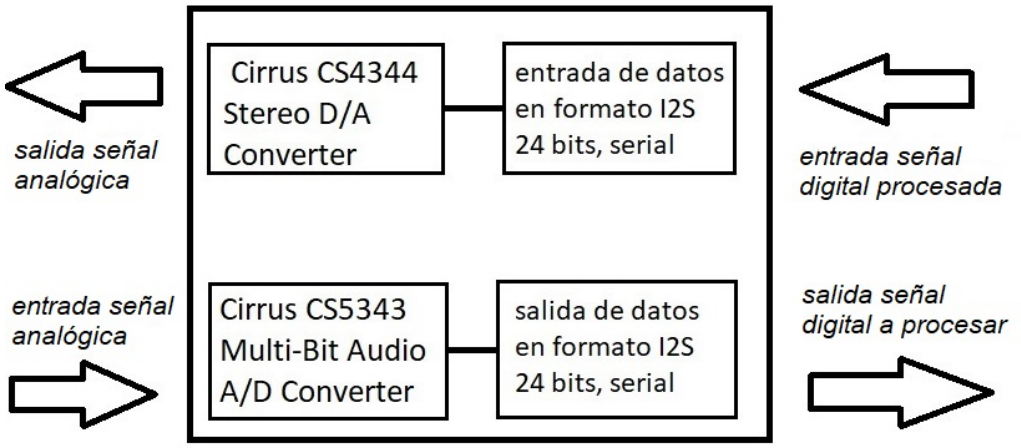


Figura 1-6: Bloques PMOD I2S2.

La figura 1-7 muestra en color amarillo al FPGA, dentro se encuentran los módulos que acondicionan los datos y realizan la comunicación, entre ellos se especifica con flechas los buses, son todos de forma unidireccional, por lo que se puede deducir que cada flecha muestra un bus de conexión, el módulo de conversión entre I2S (Inter-integrated Sound) y AXI-Stream genera

la condición para que los módulos de FIR-compiler, Control de volumen y efecto se comuniquen entre ellos en forma de anillo y efectúen el cambio en los datos antes de ser enviados al DAC del PMOD.

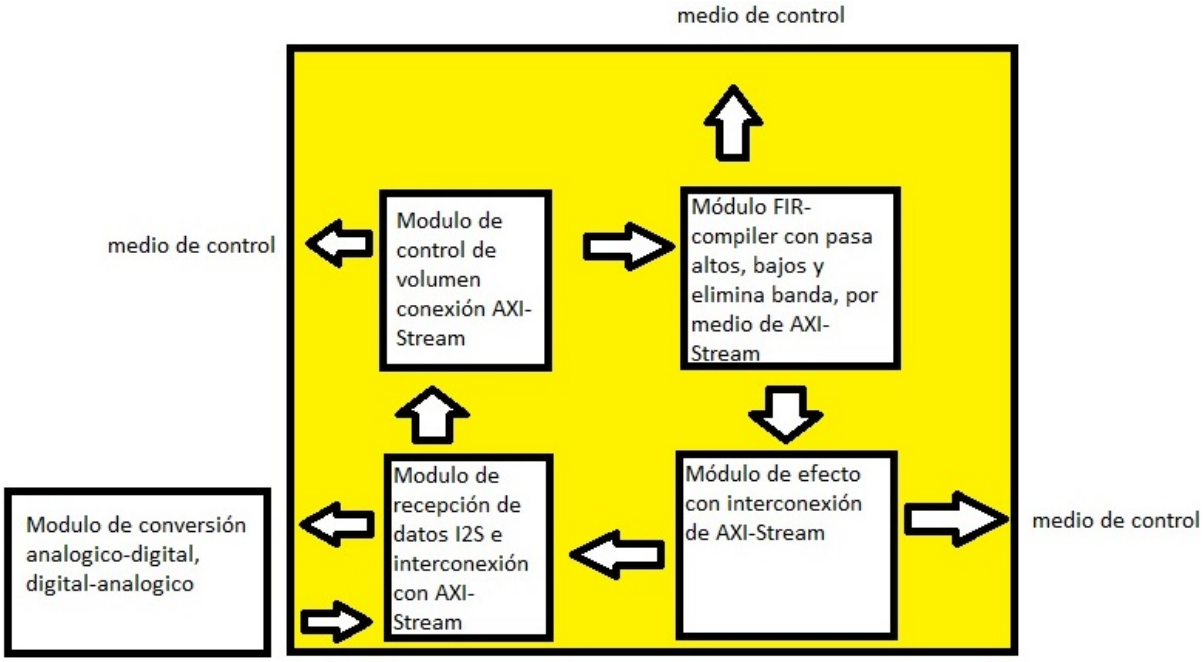


Figura 1-7: Top-level del proyecto.

También se observan las flechas que representan los buses de control y datos, se indican de manera general ya que no todas las señales en estos buses fluyen en una misma dirección, aquí solamente se observa la conclusión del diseño por el método híbrido, el cual fue cambiando paulatinamente con respecto del número de iteraciones en los ciclos de diseño Top-down y Bottom-up, se debe remarcar que existe un número ilimitado de ciclos en el proceso de desarrollo híbrido por lo que es aconsejable establecer un nivel en el cual pueda finalizar el proyecto, es decir, designar una calidad a cumplir para eliminar tiempo extra en el diseño.

2

Estructura del lenguaje VHDL

Las siglas VHDL cuyo significado es Very High Speed Intergrated Circuits Hardware Description Language, denota que se pueden describir circuitos electrónicos haciendo uso del aceleramiento del proceso de diseño. El uso principal de este lenguaje de descripción es probar circuitos de gran complejidad verificando su funcionamiento real con solo utilizar el comportamiento lógico de los circuitos como base en el diseño, sin necesidad de establecer restricciones adicionales.

Debe tenerse en cuenta que VHDL no es un lenguaje de programación, se hace uso de algunas sentencias familiares a los lenguajes , pero el saber estas sentencias o la sintaxis que usa el lenguaje descriptivo VHDL no implica saber programar con él, en vez de utilizar funciones o variables, se piensa en circuitos biestables o compuertas, se debe evitar el uso de bucles combinacionales o relojes condicionados, hay que identificar circuitos que funcionan de manera secuencial y combinacional[37], así se tendrá un mejor control de la descripción hecha.

VHDL tiene la particularidad de poder describir los circuitos de forma estructural y también de forma funcional, un programa es totalmente estructural cuando en su arquitectura no existe ningún "process", proporciona suficiente soporte para especificar su comportamiento o estructura de hardware, incluyendo las jerarquías. Por lo tanto, es útil para metodologías de diseño ascendentes (bottom-up) y descendentes (top-down)[17].

Para la estructura de los programas se tiene como referencia la siguiente figura 2-1 donde

se aprecia la entidad, arquitectura y sus definiciones.

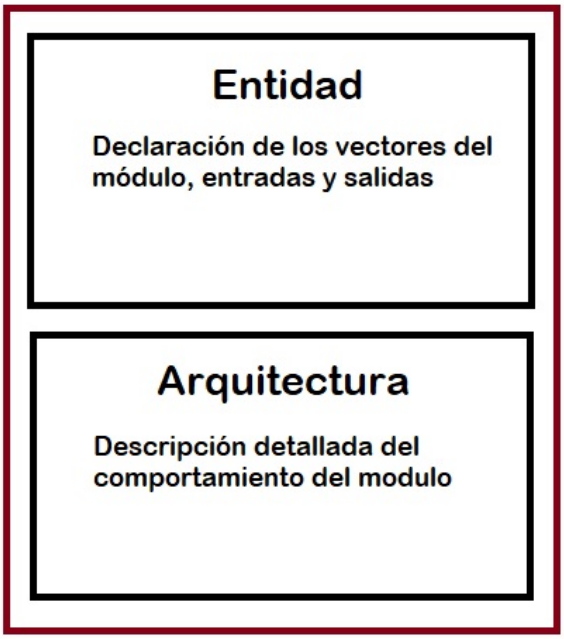


Figura 2-1: Estructura de programas en VHDL.

2.1. Entidad

La entidad es una abstracción de un circuito electrónico digital, define cuales serán las entradas y salidas del circuito, esta entidad siempre se relaciona con una arquitectura, ya que la ***entity*** únicamente describe la forma externa del circuito que representa.

Para poder identificar los módulos que constituyen a un proyecto que incluye varios submódulos a su vez, se debe establecer cual será el general de todos ellos, es decir, cual contendrá a las demás subentidades, de esta forma se pueden hacer las pruebas modularmente y corregir fallos aislando el error hasta poder identificarlo fácilmente, esto es un beneficio de trabajar bajo el esquema ***Modular***.

```
1  -- (esto es un comentario VHDL)
2  -- esta es la entidad
3     entity circuito_AND is
4        port (
5              I1 : in std_logic;
6              I2 : in std_logic;
7              O  : out std_logic);
8     end entity circuito_AND;
```

En el código anterior de la linea 3 a la 8 se identifica la entidad, dentro se observan las entradas con la palabra reservada ***port*** y se especifica si son de tipo bit o vectores de bits, pueden contener matrices conocidas como arreglos y también la posibilidad de establecer el tipo ***std_logic***.

2.2. Arquitectura

Como se observa en la estructura del lenguaje VHDL, un diseño completo esta compuesto por un par de elementos, entidad y arquitectura, esta última tiene la descripción exacta del funcionamiento del circuito al cual la entidad hace referencia, entonces el comportamiento del circuito se aloja en la ***Architecture*** de la entidad a la que se asocia, utilizando el lenguaje descriptivo VDHL.

```
1      -- esta es la arquitectura
2      architecture RTL of ANDGATE is
3      begin
4        O <= I1 and I2;
5      end architecture RTL;
```

Como se observa en el código anterior la Arquitectura comprende desde la linea 2 a la 5, la inicia una sentencia ***begin*** en la cual comienza un proceso concurrente[1], dentro del cual se

[1]sentencias que no importa en que orden o cuantas se ejecuten, no afecta el resultado final.

pueden anidar otros procesos (process) del mismo tipo. Dentro de la arquitectura se debe hacer uso de las lineas de entrada y salida definidas en la entidad, se puede leer o escribir en ellas o hacer ambas operaciones, eso lo delimitará la definición de ***in***, ***out***, ***inout*** y ***buffer***.

2.3. Jerarquías de diseño

Existen descripciones funcionales y descripciones estructurales, una se encarga de indicar como es el comportamiento de un módulo de forma operacional y la otra de especificar como esta relacionado con entidades superiores e inferiores, se debe describir algunas conexiones realizadas de un módulo a otro o la interconexión dentro del mismo con sus submódulos o componentes. Para la conexión de estos módulos se debe hacer un mapa de sus puertos, para tener una referencia establecida al conectarlos con los módulos internos o externos a él, se tienen algunas preposiciones para definir a los componentes de una descripción estructural.

- Comunicación a través de señales.
- Un componente puede ser otro módulo individual o insertarse dentro de una arquitectura con la palabra reservada ***component***.
- Los módulos pertenecientes a una estructura de VHDL son concurrentes unos con otros.
- La instancia consiste en seleccionar un módulo compilado y ligarlo a la arquitectura donde éste será usado.

2.4. Esquema de interconexión entre módulos (Entidad, Arquitectura)

Por lo visto anteriormente, se tienen definidas todas las conexiones entre los módulos del proyecto de forma estructural, gracias al desarrollo híbrido en el capitulo 1, así es posible desarrollar el módulo Top que describirá al proyecto completo, de la siguiente forma:

Cada arquitectura contiene los elementos que se necesitan para poder cumplir con la descripción de la entidad de jerarquía superior, por lo que, hay arquitecturas donde se almacenan varias entidades adicionales, como es el caso de ***filtered_fir*** y ***effect***, dentro de la arquitectura

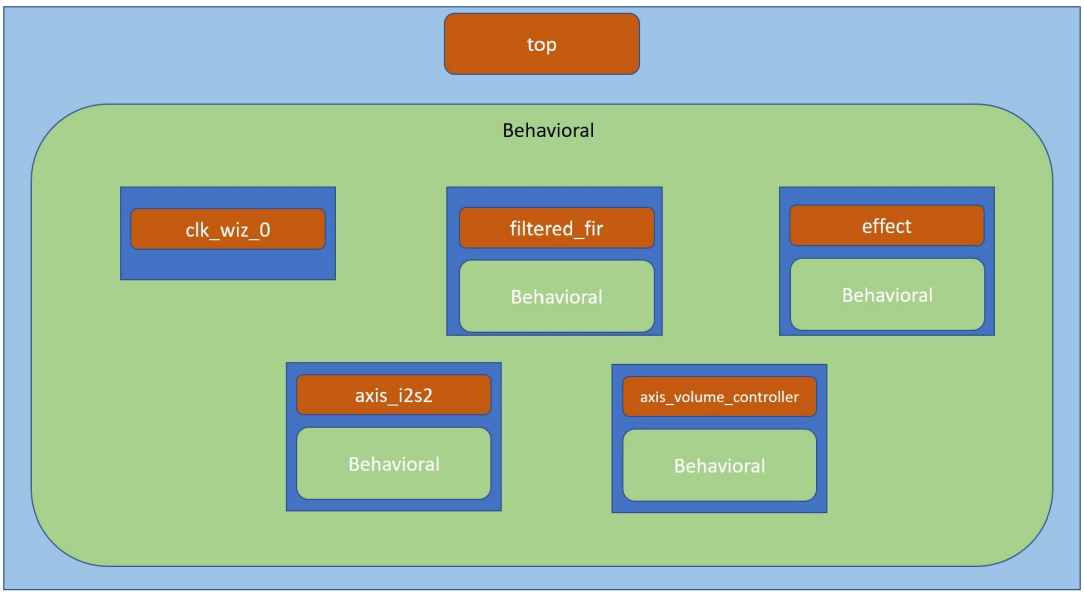


Figura 2-2: Entidad y arquitectura Top.

top que se observa en la figura 2-2, también se designa una entidad para ***clk_wiz_0***, aunque no tenga descripción de comportamiento, debido a que es un módulo de "caja negra"[2] disponible como herramienta adicional en Vivado Design Suite®.

[2]Entidad en la cual se saben las señales de entrada y salida sin conocer su funcionamiento.

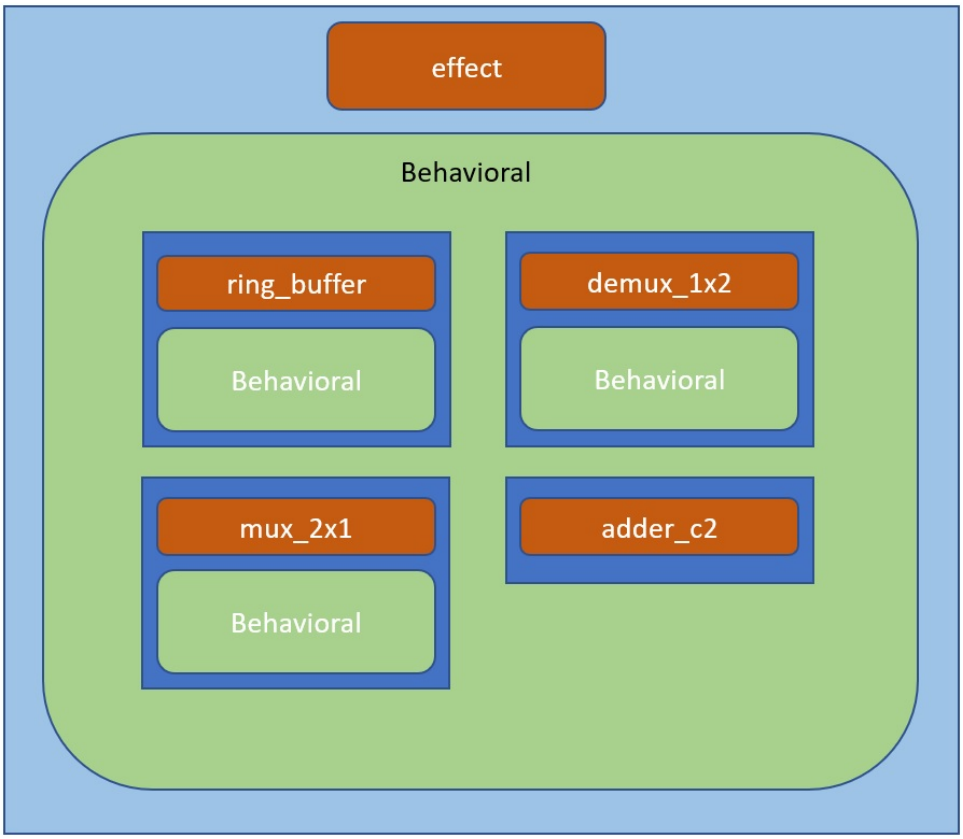


Figura 2-3: Entidad y arquitectura bloque efecto.

Dentro de la entidad ***effect***, que se observa en la figura 2-3, se cuenta con ***ring_buffer***, ***demux_1x2***, ***mux_2x1*** y ***adder_c2***, esta última entidad, al igual que ***clk_wiz_0***, no contiene su arquitectura propia, ya que forma parte del comportamiento de ***effect***, este sumador en complemento a 2 está definido con código VHDL por lo que es claro como opera este componente, los módulos ***demux_1x2***, ***mux_2x1*** solo contienen su descripción funcional dentro de cada arquitectura.

ring_buffer, mostrado en la figura 2-4, contiene a ***ram_memory***, el cual integra solo a su descripción funcional y otros dos bloques de propiedad de Xilinx® que se manejan como cajas negras estos son: ***counter_binary_0*** y ***c_addsub_1***, también dentro de la arquitectura de ***ring_buffer*** existen sentencias que comunican a la entidad con el exterior pero no se definen como otro bloque, así que se introduce de manera intrínseca en el código que define su comportamiento.

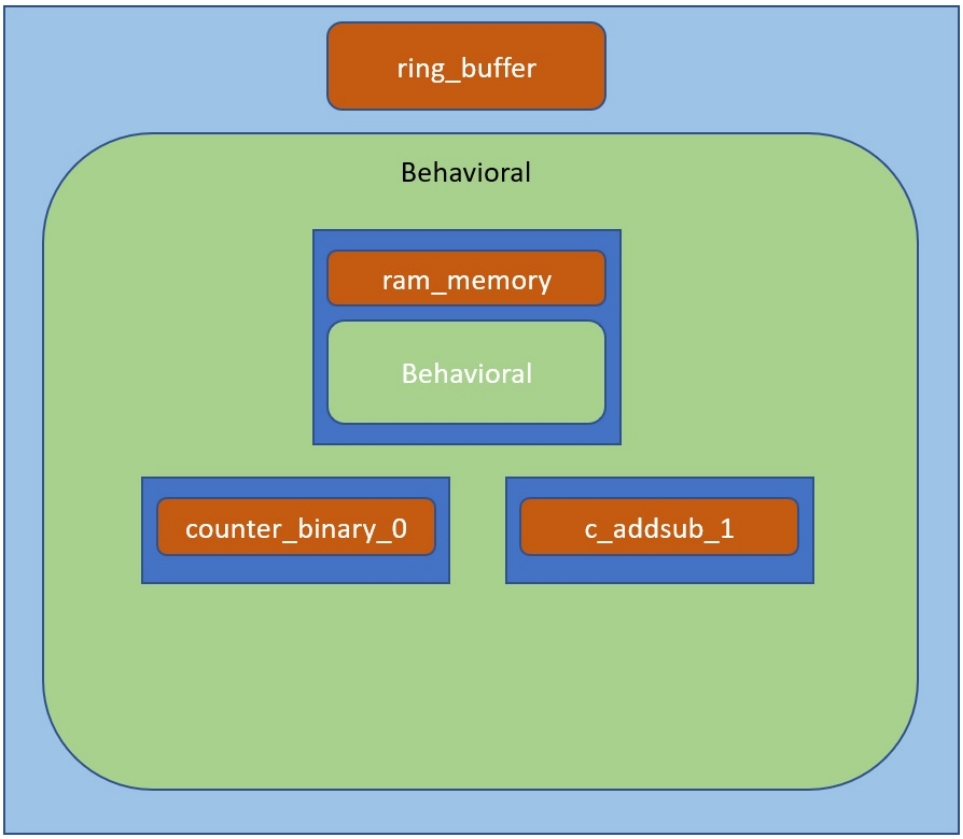


Figura 2-4: Entidad y arquitectura bufer circular.

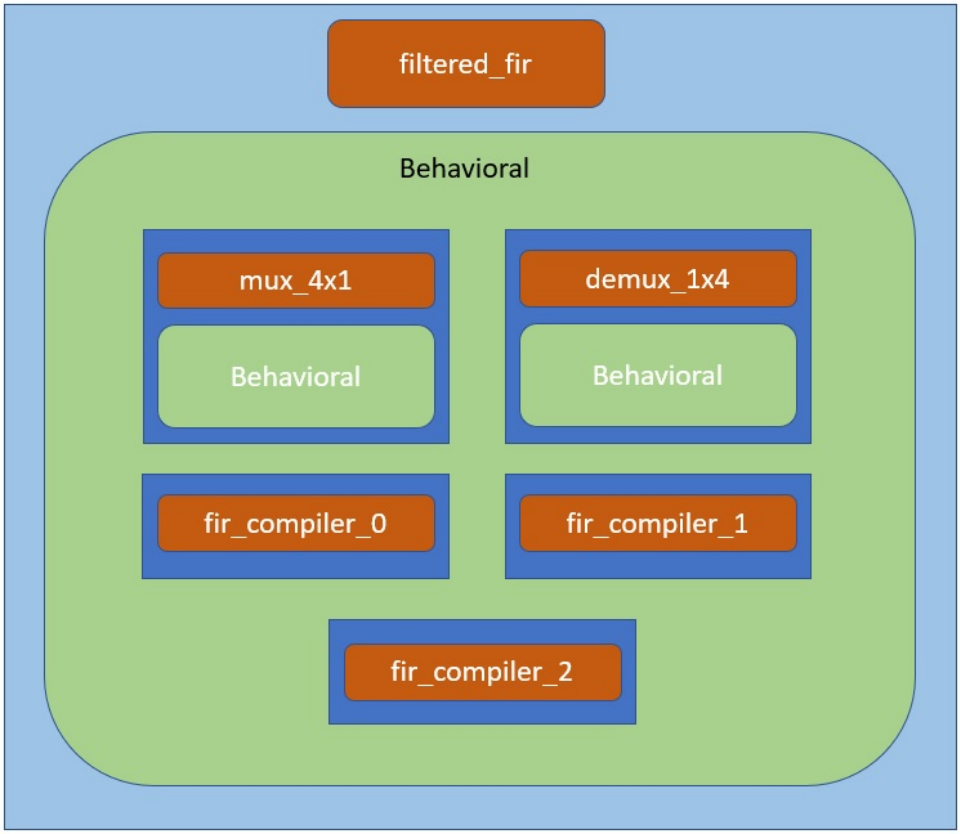


Figura 2-5: Entidad y arquitectura filtered FIR.

El módulo de filtrado, figura 2-5 ***filtered_fir***, es el último que anida otras estructuras, ***mux_4x1*** es un multiplexor 4 a 1 que contiene a su definición, es decir, su entidad y también su arquitectura, ***demux_1x4*** está en las mismas condiciones que el multiplexor y por último se agregan ***fir_compiler_0***,***fir_compiler_1*** y ***fir_compiler_2***.

Ahora se pueden ligar las estructuras de una manera jerárquica, se conoce cuales son las partes fundamentales en un bloque o módulo VHDL, por lo que, podemos proceder a definir los protocolos de comunicación para conocer las entradas y salidas de cada entidad.

3

Entrada de audio al sistema a través del ADC (Analogic-Digital converter) y salida a DAC (Digital-Analogic converter).

La conversión de señal analógica a señal digital contiene una serie de procesos básicos en un circuito regido por Nyquist, a continuación se muestran los bloques funcionales básicos de un ADC en la figura 3-1[1]:

Teóricamente los valores que contiene una señal analógica entre un punto de referencia arbitrario y otro, separados por un espacio infinitesimalmente pequeño a lo largo del tiempo, seguirá teniendo valores infinitos, a diferencia de los sistemas digitales donde solo se procesan '0' y '1' al codificar una señal, por lo que es necesario tomar solo valores representativos que ayude a reproducir de una forma lo más fiel posible a esa señal original al decodificarla. Pero surge una duda ¿ cómo se puede saber que valores caracterizan a una señal analógica?, para ello existe una teoría matemática que define una razón de muestreo que facilita esta selección,

[1]En la figura se observan tres procesos básicos para la conversión de una señal analógica a digital.

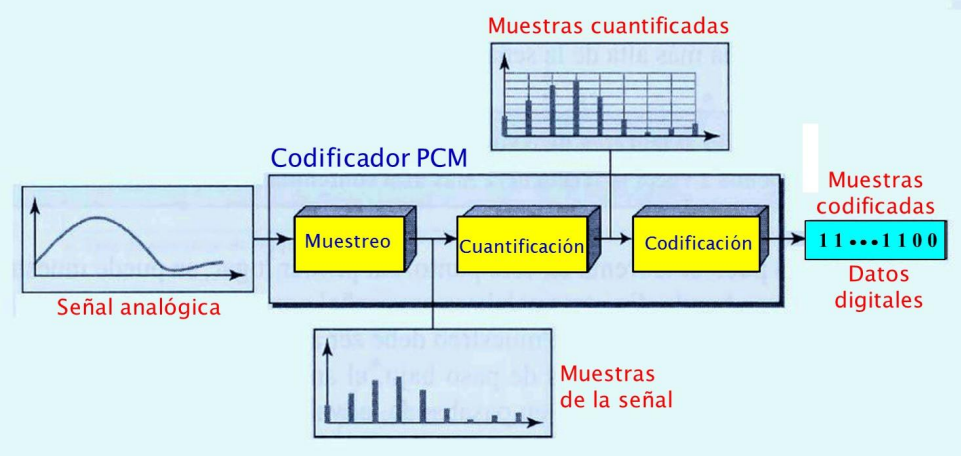


Figura 3-1: Bloques para ADC Nyquist [1]

llamada frecuencia de muestreo de Nyquist.

$$F_M \geq 2\Delta B. \tag{3-1}$$

$F_M \rightarrow$ Frecuencia de muestreo.

ΔB $\rightarrow$ Ancho de banda para la señal muestreada.

Al cumplir con la frecuencia de muestreo de Nyquist, que requiere ser como mínimo el doble del ancho de banda de la frecuencia deseada, se logra reconstruir la señal sin perdidas, esto a través de un filtro pasa-bajas, y sí se aumenta la frecuencia por encima de Nyquist, tendrá una mejor caracterización de la señal con sobre muestreo [41], además que el ADC que maneja el módulo PMOD I2S2 de Digilent® utiliza frecuencias para el muestreo por encima del criterio de Niquist definido por anchos de banda seleccionadas por el fabricante [25], una característica importante de muestreo como se observará más adelante.

Si se hace un muestreo a una frecuencia menor que el doble del ancho de banda, tendremos una superposición en los espectros de frecuencia, se puede apreciar que la frecuencia define características importantes en los procesos de conversión de la señal analógica, entonces en el adc tradicional al introducir la cuantificación después del proceso de muestreo tiene una perdida de datos ya que este proceso es no-lineal y no-reversible, esto hace que la diferencia que hay

entre la señal muestreada $x(nt_s)$ y la señal cuantificada $x_q(nt_s)$ produzca un error:

$$\varepsilon = x_q(nt_s) - x(nt_s) \tag{3-2}$$

Por lo anterior se establece otro tipo de conversor A-D, son los ADC con sobre muestreo o Sigma-Delta[8] que minimizan el problema del error de cuantificación, se debe tener en cuenta que en el muestreo no hay perdidas de información. Estos circuitos realizan el muestreo usualmente a 128, 256 o 512 veces más que la frecuencia de Nyquist, además, un sistema intermedio llamado "modulador sigma-delta" realiza las operaciones de cuantificación y conformado de ruido, estos ADC tienen aplicaciones en sistemas de audio que justamente se tocan en este trabajo, instrumentación biomédica, etc.[41], se visualizan los bloques funcionales del conversor Sigma-Delta en la siguiente figura 3-2:

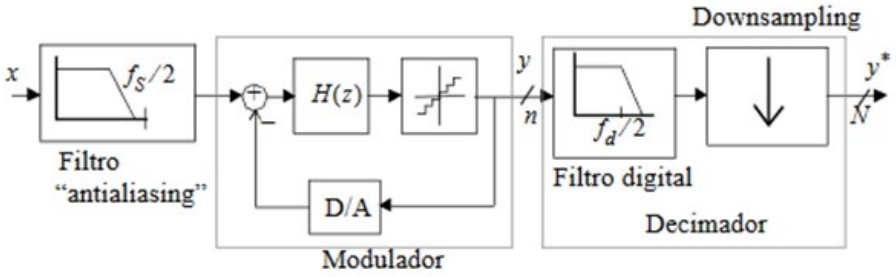


Figura 3-2: Bloques de ADC Sigma-Delta, fuente: Convertidores A/D de sobremuestreo usando técnicas de modulación Sigma-Delta: Conceptos básicos y estado del arte.

La entrada del ADC tiene filtro "antialiasing"[2], que asegura la entrada de frecuencias dentro de un rango establecido para eliminar componentes que puedan generar aliasing, la siguiente parte, llamada Modulador, contiene una función $H(z)$ que funciona como integrador, después el cuantizador, que se observa como un bloque gráfico, reduce la cantidad de ruido blanco,esto lo hace debido a que es un sistema retroalimentado por lo que tiende a irse corrigiendo es decir, esa fuente de ruido se minimiza y así pasa a ser un sistema cuasi lineal, todo este proceso es conocido como conformación de ruido o "noise-shaping", se puede consultar información adicional en[11]. La etapa siguiente, contiene un bloque de filtrado puramente digital por lo que la eliminación

[2]Filtro analógico pasabajas que antenúa idealmente a las frecuencias que superan la mitad de velocidad de muestreo.

de las componentes restantes y el error restante de la cuantización es casi perfecto ya que la digitalización hace al filtro cercano a lo ideal, después se obtiene una reducción en la frecuencia de muestreo “Decimación”, resultando en una señal de entrada codificada en un gran número de bits y a la frecuencia de Nyquist.[1]

3.1. Características del módulo I2S2

El ADC “Analogic-Digital Converter” CS5343-CZZ que esta integrado en el PMOD I2S2 stereo de Digilent® usa un modulador Delta-Sigma multi-bit de tercer orden, seguido de filtrado digital el cual elimina la necesidad de un filtro anti-aliasing externo. A continuación se listan las características más importantes del circuito:

- Conversión a valores de 24 bits con dos canales, izquierdo y derecho.
- CS5343 soporta el formato de audio I^2S que se encuentra en complemento a 2.
- Relación de relojes MCLK/LRCK de 128x, 192x, 256x, 384x, 512x, 768x .
- Rango dinámico de 94 dB a 3.3 v .
- -89 dB de THD+N 18 a 24 bits con 3.3v .

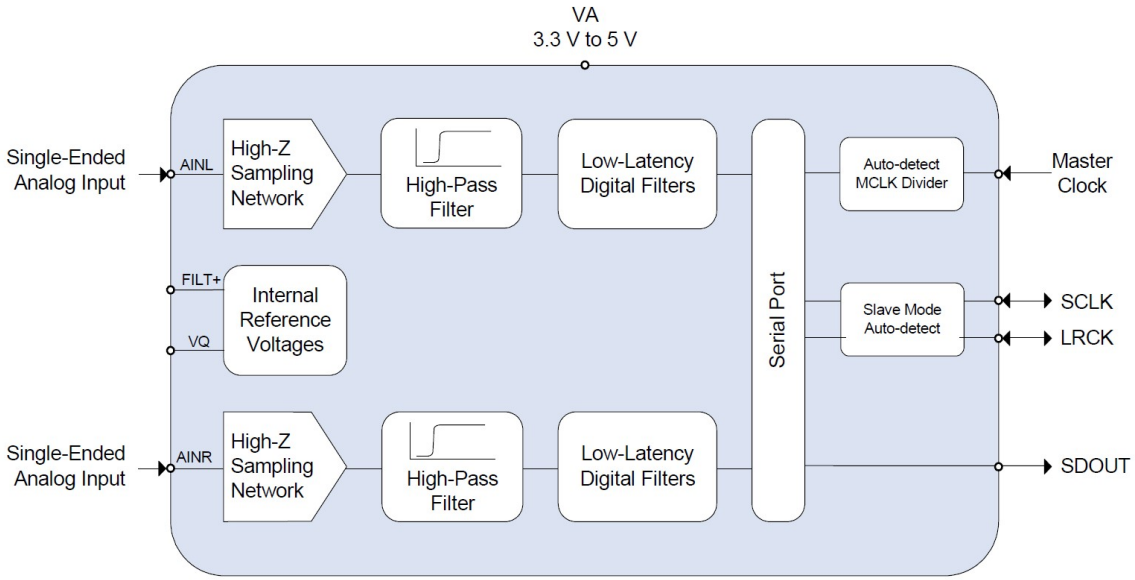


Figura 3-3: ADC datasheet CS5343, fuente: Cirrus® Logic datasheet CS5343.

En el diagrama a bloques del ADC que se muestra en la figura 3-3 contiene un módulo de alta impedancia para eliminar el error de comparación usual que se tiene al hacer este proceso usando un amplificador operacional.

El DAC "Digital-Analogic Converter" CS4344CZZ Modulador Multi-bit Delta-Sigma de cuarto orden que incluye filtros de interpolado con salida analógica filtrada, soporta la mayoría de formatos para audio, algunas de sus características son:

- Conversión a valores de 24 bits con dos canales, izquierdo y derecho.
- CS4344 soporta el formato de audio I^2S que se encuentra en complemento a 2.
- Relación de relojes MCLK/LRCK de 64x, 96x, 128x, 192x, 256x, 384x, 512x, 768x, 1024x o 1152x .
- Rango dinamico de 103 dB a 3.3 v .
- -90 dB de THD+N 18 a 24 bits con 3.3v .

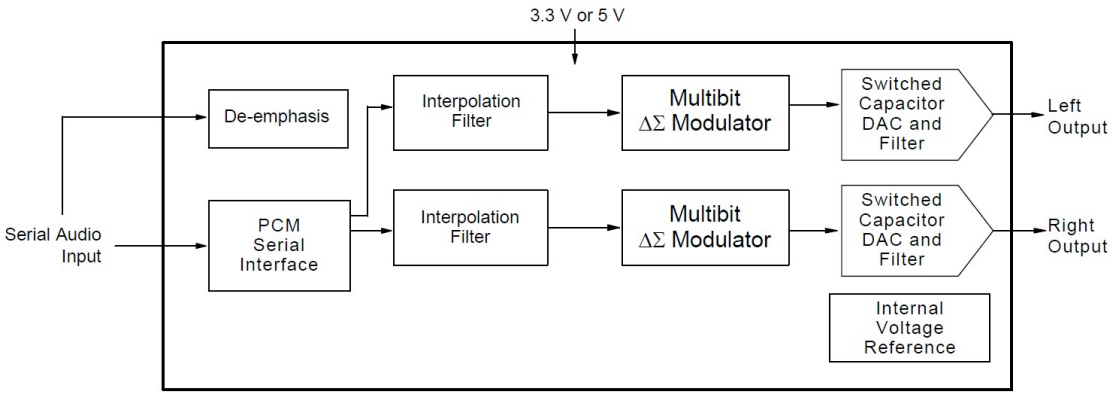


Figura 3-4: Fuente: Cirrus® Logic datasheet CS4344.

Básicamente, el PMOD I2S2 tiene estos dos circuitos ADC y DAC que conforman la entrada y salida al proyecto de audio respectivamente, tienen la configuración de conexión recomendada por el fabricante, a continuación se puede ver el esquema en la figura 3-5 de la conexión de todo el módulo.

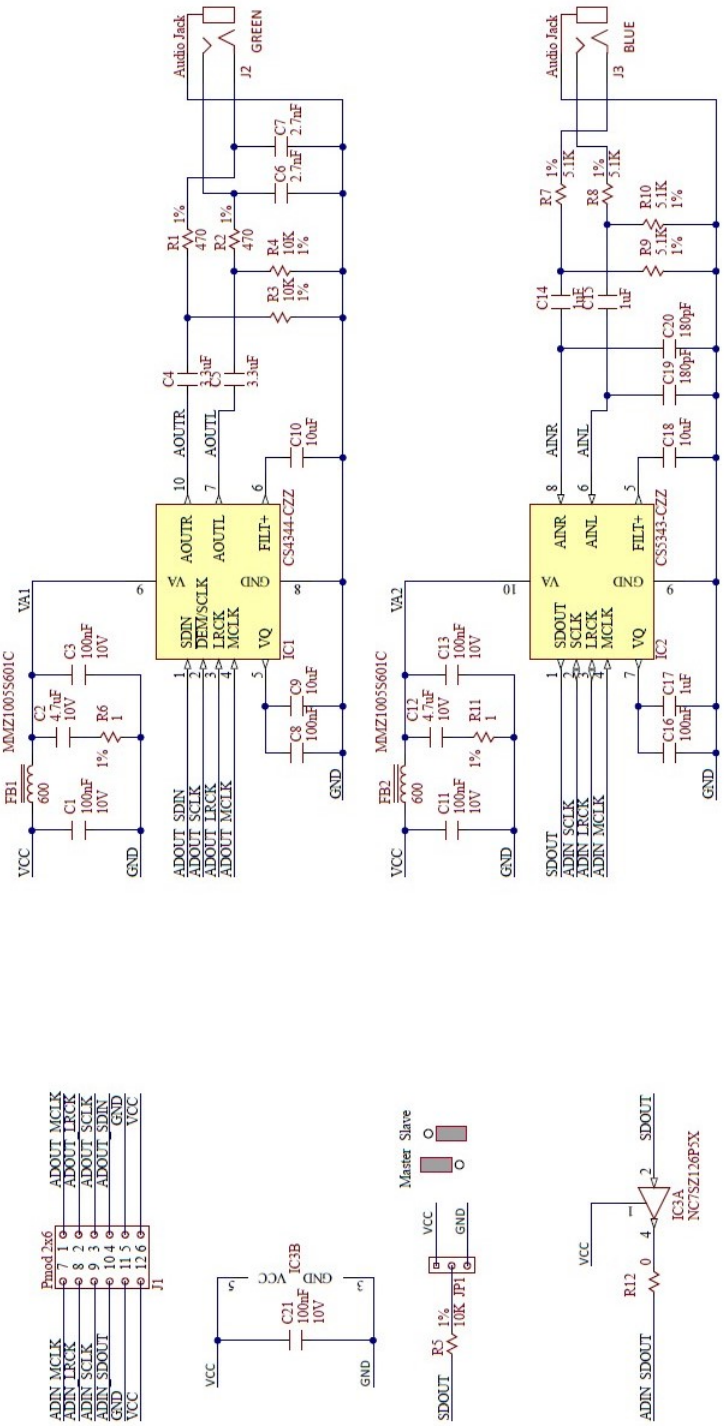


Figura 3-5: PMOD I2S2, fuente:Digilent® .

3.2. Protocolo de transferencia AXI-Stream

AXI es parte de ***ARM®*** ***AMBA®*** (Advance Microcontroller Bus Architecture) una familia de microcontroladores que se introdujo por primera vez en 1996 [3]. La primera versión de ***AXI*** se incluyo en ***AMBA® 3.0***, lanzado en 2003. ***AMBA® 4.0*** Lanzado en 2010, incluye la segunda versión de ***AXI***,***AXI4***. Es un bus de comunicaciones cuyo propósito es la interconexión de bloques funcionales en un ***SoC*** (System-On-Chip)[36], se puede encontrar el nombre ***AXI4-Stream*** donde el número ***4*** significa la especificación ***AMBA® 4.0*** que corresponde a la versión de acuerdo a la fecha de elaboración de este trabajo puesto que recientemente se introdujo una nueva especificación el 31 de marzo de 2020, que corresponde a ***AMBA® 5.0***[5], de ahora en adelante esto se sobreentiende y solo se remarcará el nombre del protocolo ***AXI*** teniendo la interfaz ***Stream*** donde se envían los datos de forma ordenada ***FIFO*** (First Input First Output) en un tipo de comunicación de maestro (envia datos) a esclavo (recibe datos).

Existen tres tipos de interfaces:

- AXI para sistemas con requisitos de mapeo de memoria con alto rendimiento (se envían ráfagas de datos en una misma transferencia).
- AXI-lite para una comunicación de mapeo de memoria, con envio simple y bajo rendimiento.
- AXI-Stream para transmisión de datos a alta velocidad de tipo maestro-esclavo.

El proyecto se centrará en el protocolo AXI-Stream el cual tiene en particular que no es mapeado en memoria, esto quiere decir que no necesita un canal con dirección de lectura o escritura, es un protocolo punto a punto a través de un canal unidireccional, como se mencionó antes la información va de un maestro a un esclavo. se cuenta con la siguiente figura 3-6 que indica la transferencia de datos.

El protocolo emplea una lista de señales especificadas en el protocolo, estas señales se indican en la tabla 3-1

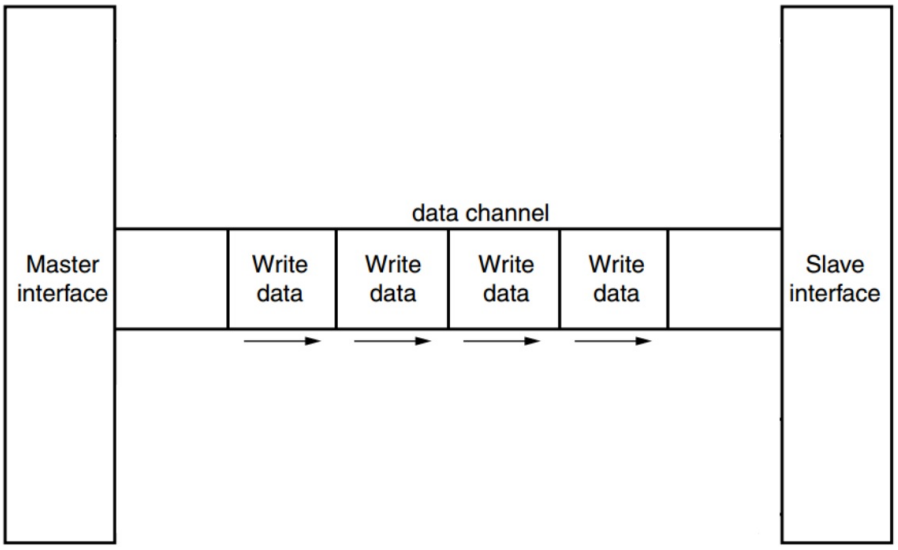


Figura 3-6: Protocolo AMBA Master Slave, fuente:ARM® AXI-Stream V 1.0.[4]

.

Señal	Fuente	Descripción
ACLK	Reloj	Señal de reloj global. Las señales se muestran en el flanco de subida de ACLK.
ARESETn	Reset	Señal de reset global activa a nivel bajo.
TVALID	Maestro	Señal que indica que los datos de entrada son válidos.
TREADY	Slave	Señal que indica que el esclavo puede aceptar un nuevo dato.
TDATA[(8n-1):0]	Maestro	Bus de datos de entrada.
TSTRB[(n-1):0]	Maestro	Indica si los bytes de TDATA son bytes de datos (datos válidos) o son bytes de posición (datos no válidos)
TKEEP[(n-1):0]	Maestro	Indica los bytes de TDATA que son válidos y deben ser transferidos.
TLAST	Maestro	Inica el final del paquete.
TID[(i-1):0]	Maestro	Es el indicador de stream de datos. Cada stream tendrá un valor diferente.
TDEST[(d-1):0]	Maestro	Aporta información de la ruta del stream de datos.
TUSER[(u-1):0]	Maestro	Para enviar información de usuario sobre la transacción.

Tabla 3-1: Señales utilizadas en el protocolo AXI-Stream

De la tabla 3-1 se tiene:

-n: Número de bytes del bus de datos.

-i: Número de bits de TID, se recomienda máximo 8 bits.

-u: Número de bits de TUSER, se recomienda que el número de bits sea un múltiplo entero del ancho de la interfaz en bits.

-d: Número de bits de TDEST, se recomienda máximo 4 bits.

Proceso de Negociación (Handshaking)

Las señales TVALID Y TREADY en el proceso de negociación determinan el control de flujo de la información a través de la interfaz. Un mecanismo de flujo bidireccional permite al maestro y al esclavo controlar la velocidad en la que se transmiten los datos y la información de control a través de la interfaz. Para que se produzca una transferencia, las señales TVALID Y TREADY deben ponerse en alto, TVALID o TREADY pueden levantarse una a la vez o ambas durante el mismo ciclo de reloj de ACLK.

Existe la restricción de que el maestro siempre debe poner en alto TVALID primero y después esperar por TREADY, jamás del modo contrario, es decir, una vez que TVALID está en alto se produce la negociación. Por lo tanto, un esclavo debe esperar a que se ponga en alto TVALID antes de que él ponga en alto TREADY. Si un esclavo pone en alto TREADY antes de que el maestro ponga en alto TVALID, es posible que el esclavo vuelva a colocar en nivel bajo la señal TREADY.

En la figura 3-7 se observa que primero el maestro pone en alto TVALID una vez que colocó la información en la entrada, esto se mantiene sin cambios hasta que la señal TREADY es puesta en alto por el esclavo lo que indica que puede aceptar los datos, la transferencia se realiza una vez que el esclavo pone en alto TREADY, la flecha indica cuando ocurre la transferencia.

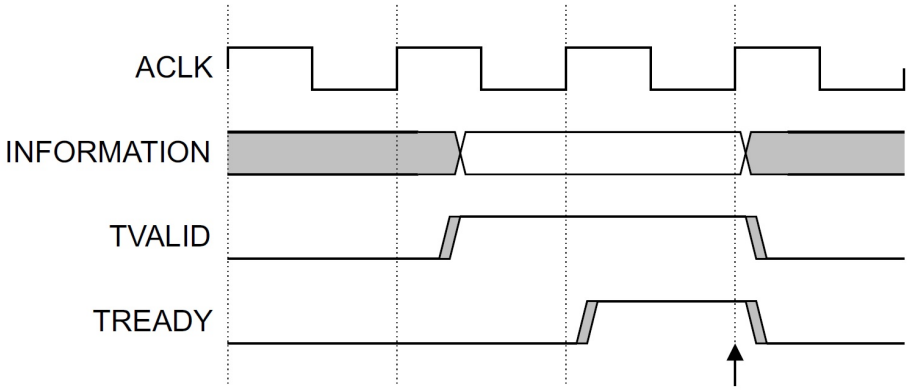
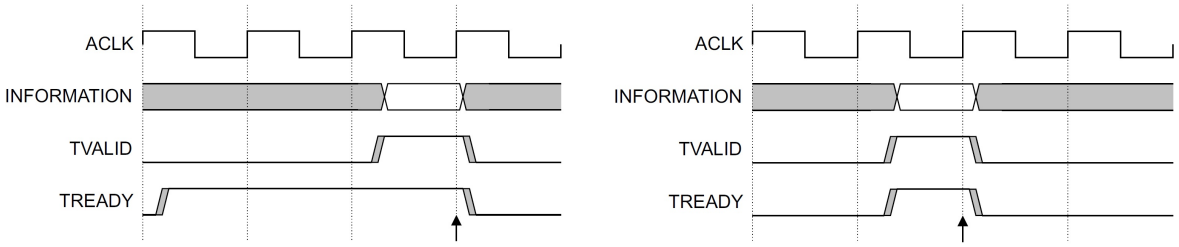


Figura 3-7: Protocolo AMBA AXI-Stream inicio de trasferencia de datos, fuente:ARM® AXI-Stream V 1.0.[4]

Existen dos opciones adicionales, que se muestran en la figura 3-8,en total son estas tres posibilidades con las que se comienza el flujo de datos, TREADY en alto antes de TVALID, TVALID en alto antes que TREADY y TVALID con TREADY en alto en al mismo tiempo.



(a) comienza TREADY en alto.

(b) TVALID y TREADY en alto a la vez.

Figura 3-8: Protocolo AMBA versiones de transferencia de datos, fuente:ARM® AXI-Stream V 1.0.[4]

Protocolo de comunicación entre maestro y esclavo

1. La entidad que enviará los datos debe cargar los mismos en el bus adecuado, terminado esto pondrá la señal TVALID a 1 (alto), con esto indica al otro extremo que los datos están listos.
2. La entidad que recibe debe poner la señal TREADY a 1 (alto) para indicar que está listo para realizar la transacción.

3. El envío de datos se produce cuando ambas señales están en alto en un mismo ciclo de reloj, como se mencionó antes, el orden en el que se levantan las señales no importa.
4. La transacción puede darse por tanto en tres situaciones:
   a) Si se levanta TVALID y mientras aún está en alto se levanta READY, figura 3-7.
   b) Si se levanta TREADY y mientras aún está en alto se levanta VALID, figura 3-8 (a).
   c) Si TVALID y TREADY se levantan simultáneamente, figura 3-8 (b).
5. Output Clocks La señal TLAST indica el final de la comunicación, permanecerá en cero en todo momento durante la transacción, levantándose únicamente cuando el último bloque de datos esté llegando hasta el receptor, quiere decir que la señal estará en alto junto con la llegada del ultimo paquete. La señal se levantará a la vez que se levanta el último TVALID, de modo que el receptor sabrá, al leer ese paquete de datos, que no van a entrar más, mientras la señal de TREADY baja temporalmente.

Se tiene por lo tanto en un bus de AXI-Stream las siguientes señales: Bus de datos que puede ir de 8 a 1024 bits de ancho proveniente del maestro, TVALID de 1 bit proveniente del maestro, TREADY de 1 bit proveniente del esclavo y TLAST de 1 bit proveniente del maestro. Solo se ocuparan estas cuatro señales, si se necesita más información sobre el tipo de paquetes, los tipos de bytes que puede manejar o información adicional puede ser consultada en [4]

3.3. Protocolo de transferencia I2S (Integrated Interchip Sound).

El protocolo *I2S* cuyas siglas significan “Inter-IC Sound” ó “Integrated Interchip Sound” es un enlace serial especial diseñado para audio digital. Es cierto que en el mercado existen muchos sistemas de audio y deben de tener una forma de comunicación entre ellos, para que exista una gran flexibilidad en estos sistemas su comunicación debe ser estandarizada , pues si esto funciona de forma estándar, tanto los desarrolladores de equipos de audio y los fabricantes de los Circuitos Integrados a escala VLSI (Very Long Scale Integration), podrán trabajar puntualmente sin necesidad de visualizar otras formas de comunicación electrónica.

Los requisitos para el buen manejo de este bus son: la linea de datos solo deberá manejar datos de audio por lo que si se necesitan otros datos de subcodificación o control, deberán enviarse de manera separada, se manejan tres lineas para este bus serie, una linea de dos canales de datos multiplexados en tiempo, una linea de selección de palabra y por último una linea de reloj para la sincronización entre las dos lineas anteriores, debe aclararse que tanto el receptor como el transmisor comparten esta linea de reloj sincronizador, para poder definir quien es el maestro o transmisor solo habrá que verificar quien genera este reloj de sincronización, la linea de selección de palabra y obviamente los datos. Existen sistemas más complejos donde pueden haber más de un maestro, en tal situación se puede complicar el proceso de identificación de éstos, para ello se debe crear un maestro que controle a estos maestros que, en función del primero servirían como esclavos. Se muestran las posibles configuraciones de esclavo, maestro y controlador del bus en la imagen 3-9.

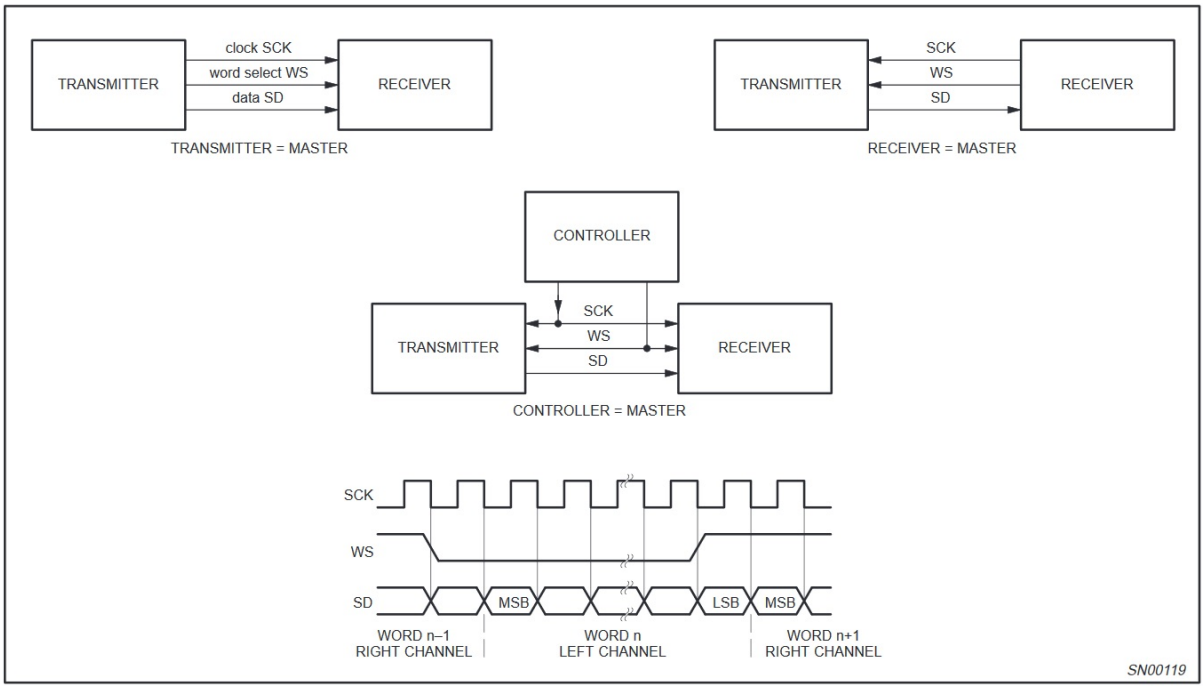


Figura 3-9: Configuración de sistemas básicos i2s, trasmisor, receptor y controlador, fuente:Philips® Semiconductors I2S bus specification.

En la imagen 3-9 se aprecian las tres lineas del bus que son como siguen:

- Un reloj serial de sincronización “Continuos Serial Clock” SCK.
- Linea de selección de palabra “Word Select” WS.
- Linea de datos “Serial Data” SD.

La transmisión de datos a través de la línea “Serial Data” es en complemento a 2 con el MSB primero, esto es porque esclavo y maestro pueden manejar diferentes longitudes de palabra ya que ambos desconocen este dato, así que si la palabra del transmisor es menor a la del sistema general, los datos menos significativos “LSB” se truncan y ponen a ’0’ y si es mayor los LSB se ignoran, esto para que los bits más significativos tengan una posición fija, siempre se envía el MSB de la siguiente palabra un pulso de reloj después del cambio de la linea de WS.

Ahora en la linea “Word Select” se designa que cuando vale ’0’ esta en transmisión el canal 1 o canal izquierdo, cuando vale ’1’ se transmite por el canal 2 o derecho, además puede cambiar en el flanco de subida o bajada, no necesita que sea simétrica la temporización de esta señal. esta señal cambia un periodo de reloj antes de que se envié el MSB.

El temporizado de las señales es importante para que haya una correcta transición entre los flancos de subida y bajada con los respectivos cambios en las señales que ejecutan el control de transmisión para este protocolo, así que se probará la frecuencia de trabajo con los periodos requeridos en la siguiente figura 3-10.

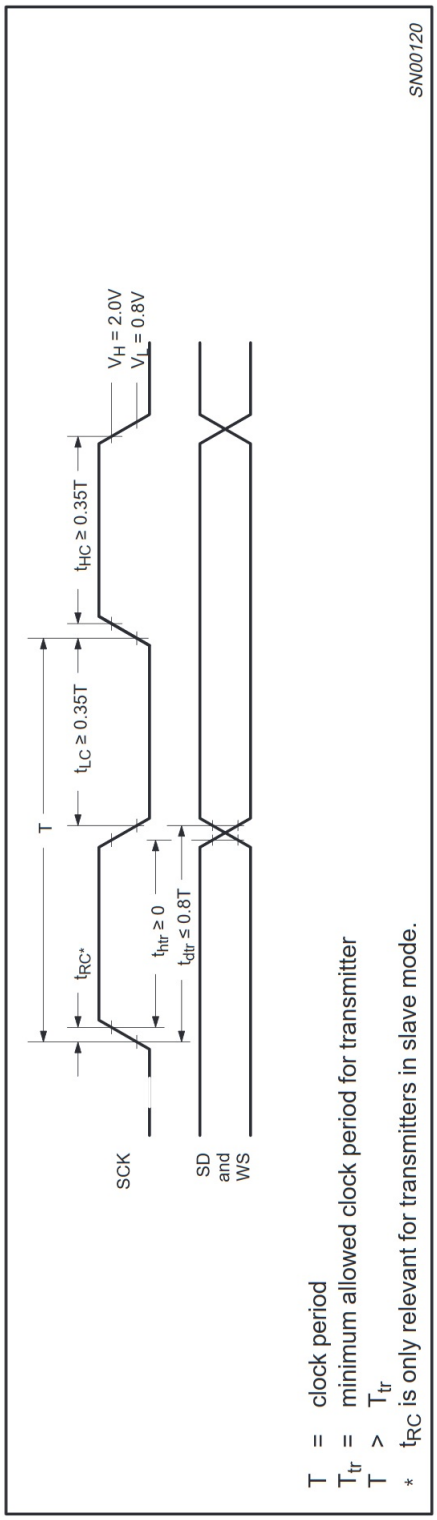


Figura 3-10: Configuración de tiempo para transmisor I2S, fuente:Philips® Semiconductors I2S bus specification trasmitter.

La frecuencia F de $MCLK$ es $22.579MHz$ por lo que $T = \frac{1}{F}$ se tiene que $T = 44.28nS$ se observa para el valor de un pulso de reloj en el transmisor $t_{HC} \geq 0.35T$ y $t_{LC} \geq 0.35T$ esto es tiempo en alto y tiempo en bajo respectivamente, por lo que si se divide el periodo entre dos se tiene que $22.14nS = \frac{T}{2}$ el requisito de temporización es $t_{HC} = 0.35(44.28nS) = 15.498nS$, se cuenta con $t_{HC} = t_{LC} = 22.14nS$ por lo que no hay problema con el tiempo manejado en el reloj principal maestro $MCLK$, el valor $t_{htr} \geq 0$ se cumple y el valor $t_{dtr} \leq 0.8T$ se cumple puesto que $t_{dtr} \leq 0.8(44.28nS)$ por lo que $t_{dtr} \leq 35.42$ y $T_{tr} = (0.35(T))(2)$ entonces $T_{tr} = 30.99$ es posible observar $T > T_{tr}$, cualquier información adicional puede encontrarse en [35] .

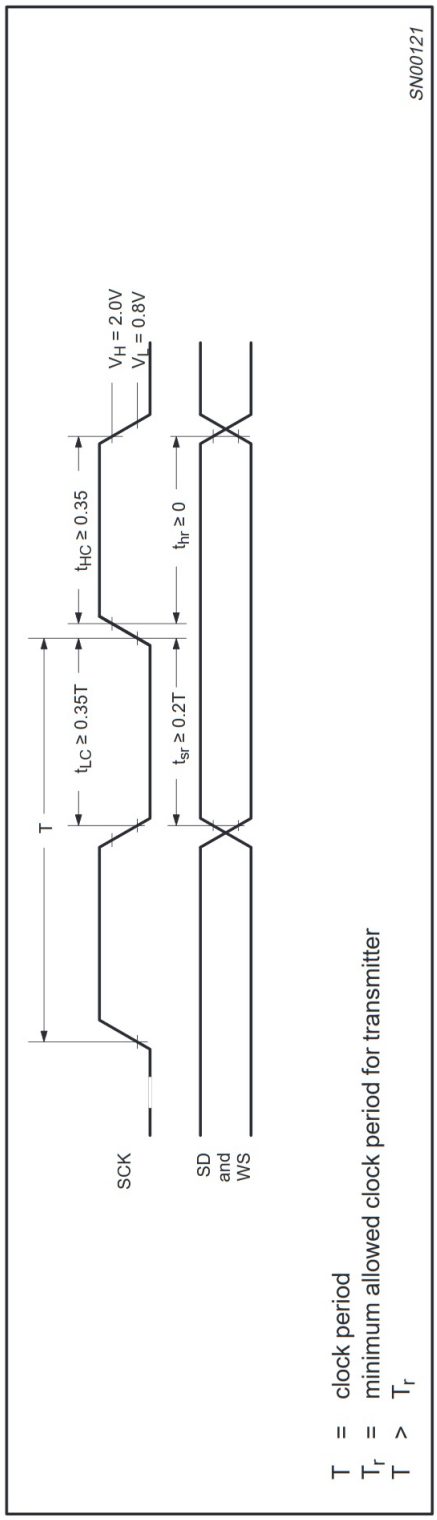


Figura 3-11: Configuración de tiempo para receptor I2S, fuente:Philips Semiconductors I2S bus specification receiver.

Es posible ver el valor de t_{sr} en la figura 3-11 que es $t_{sr} \geq 0.2T$ es el valor mínimo para el receptor en una transición de flanco de bajada a subida, por lo que $t_{sr} = 0.2(44.28nS) = 8.85nS$, la interfaz cuenta con $t_{LC} = t_{sr} = 22.14nS$ por lo que $t_{LC} > 8.85$ implica que $t_{HC} \geq 0.2T$ y $t_{hr} \geq 0$ también se cumple, así que se concluye que el tiempo mínimo para el receptor cumple con la condición $T > T_r$ que es el mismo valor T_{tr} ya que ambos, transmisor y receptor, comparten el reloj $MCLK$.

3.4. Rango de muestreo, frecuencia y resolución de bits.

Para poder definir el rango de muestreo como se indico en el subcapitulo anterior sobre $I2S$ se requiere establecer el ancho de banda utilizado, es decir que frecuencias audibles se podran escuchar en el proyecto, por lo que se decidió utilizar la calidad de CD que es “44.1 KHz”, quiere decir que es posible obtener, según Nyquist, un ancho de banda de “22 KHz” aproximadamente, el rango de frecuencia audible comprende de los “20 Hz” a “20 KHz”, así que por ello es un rango muy común y como se mencionó es utilizado en la producción de los CD´s.

La frecuencia del $MCLK$ se estableció en $22.579MHz$ pero está resulta de una característica de ambos circuitos ADC y DAC que se mostraran a continuación, una vez establecida la frecuencia de muestreo en “44.1 KHz”, se verifica que frecuencia de trabajo tienen los circuitos en sus hojas técnicas correspondientes.

Modo Maestro y Esclavo					
Frecuencia de muestreo (kHz)	Modo de velocidad	MCLK (MHz)		MCLK (MHz)	
		256x	512x	384x	768x
32 (*Solo en modo esclavo)	SSM	*8.192	*16.384	*12.288	*24.576
44.1	SSM	11.289	22.579	16.934	33.868
48	SSM	12.288	24.576	18.432	36.864
Frecuencia de muestreo (kHz)	Modo de velocidad	MCLK (MHz)		MCLK (MHz)	
		128x	256x	192x	384x
88.2	DSM	11.289	22.579	16.934	33.868
96	DSM	12.288	24.576	18.432	36.864

Tabla 3-2: Frecuencias comunes para MCLK en modo esclavo o maestro para ADC CS5343 , fuente: [25].

Como se puede verificar en la tabla 3-2 para una frecuencia de muestreo de “44.1 KHz” en modo SSM “Single-Speed Mode” existen 4 posibilidades de $MCLK$, 256x, 512x, 384x y 768x, se seleccionó la frecuencia de 512x que corresponde $(512)(44.1KHz) = 22.579MHz$, la primera razón es porque corresponde a la frecuencia de muestreo, la segunda es debido a que el DAC CS4344 tiene en común esta misma frecuencia para MCLK y es posible hacer que los dos circuitos estén sincronizados al compartir el mismo reloj a la misma frecuencia como puede verse en la tabla 3-3.

LRCK (kHz)	**MCLK (MHz)**									
	64x	**96x**	**128x**	**192x**	**256x**	**384x**	**512x**	**768x**	**1024x**	**1152x**
32	-	-	-	-	8.1920	12.2880	-	-	32.7680	36.8640
44.1	-	-	-	-	11.2896	16.9344	22.5792	33.8680	45.1580	-
48	-	-	-	-	12.2880	18.4320	24.5760	36.8640	49.1520	-
64	-	-	8.1920	12.2880	-	-	32.8680	49.1520	-	-
88.2	-	-	11.2896	16.9344	22.5792	33.8680	-	-	-	-
96	-	-	12.2880	18.4320	24.5760	36.8640	-	-	-	-
128	8.1920	12.2880	-	-	32.8680	49.1520	-	-	-	-
176.4	11.2896	16.9344	22.5792	33.8680	-	-	-	-	-	-
192	12.2880	18.4320	24.5760	36.8640	-	-	-	-	-	-
Mode	**QSM**				**DSM**		**SSM**			

Tabla 3-3: Frecuencias comunes para MCLK en modo esclavo o maestro para DAC CS4344, fuente: [24]

En las tablas se observa que también se puede seleccionar una frecuencia de 11.289 MHz en 256x para ambos CI´s o subir a 33.868 MHz en 768x, ambos validos, aunque se decidió por la media de ambos 22.5792 MHz en 512x, en este punto se cuenta con la frecuencia de muestreo y el rango de frecuencias a muestrear o ancho de banda, se tiene en cuenta que ambos circuitos tienen la capacidad de entregar valores en 24 bits, de carácter entero en complemento a 2, por lo que existen las siguientes implicaciones:

Para 24 bits se tiene que $2^{24} = 16777215$ niveles distintos en la cuantificación por lo que se establece una buena resolución para el audio, aunque existen ADC con mayor resolución de bits, un ejemplo superior es el ADS1262 de 32 bits de Texas Instruments™ [19].

La capacidad para representar valores es una característica importante para los métodos computacionales ya que de ellos dependerán los recursos a emplear en el FPGA, así pues, para 24 bits en complemento a 2 se saben los límites representables con la siguiente ecuación 3-3.

$$X_s = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2_i \tag{3-3}$$

de la ecuación 3-3 se obtiene:

$$-2_{n-1} \leq X_s \leq 2_{n-1} - 1 \tag{3-4}$$

Así substituyendo en 3-4 para $n = 24$ se tiene $-8388608 \leq X_s \leq 8388607$

De esta forma, se corrigen errores de otras representaciones como tener dos valores diferentes posibles para el cero 3-4, haciendo que todos los procesos de multiplicación y suma sean eficientes computacionalmente hablando.

Codificación	Signo y Magnitud	Complemento a 1	Complemento a 2
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

Tabla 3-4: Doble representación del cero en codificaciones de signo magnitud y complemento a 1.

3.5. Creación de bloque para comunicación.

Para poder establecer una comunicación con el protocolo AXI-Stream, en el manejo de audio con I2S se recibe de modo serial los datos y después se envian en forma paralela en el canal de datos de AXI-S, ya se definió anteriormente el protocolo I2S por lo que se conocen sus características básicas y se pueden establecer los procesos para recibir los datos a través de este de manera correcta, de igual forma se conocen las características del protocolo AXI-Stream

y su capacidad de intercambio de información en su interfaz, entonces el módulo contiene dos interfaces diferentes razón por la que se nombrará “axis_i2s2_vhdl”, en la figura 3-12 se muestra la entidad de forma jerárquica. Se tiene una estructura muy usual para la construcción

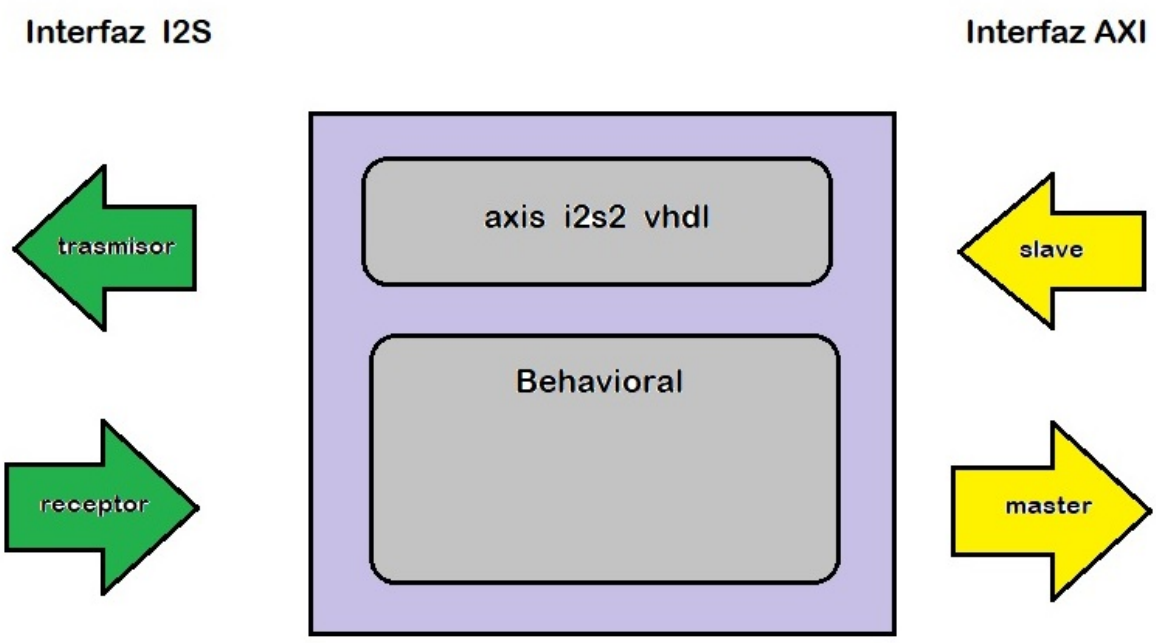


Figura 3-12: Entidad con interfaz AXI e I2S.

de sistemas de recepción de audio digital e interfaces de audio serial (SAI Serial Audio Interface), como se muestra en la figura 3-13.

Se utiliza esta configuración para la recepción y envío en la parte del protocolo I2S, se tienen estos bloques anidados en el bloque funcional. El ADC CS5343-CZZ tiene una característica que selecciona automáticamente el modo de velocidad simple o doble, funciona cuando se encuentra en modo esclavo, esto es cuando el pin de datos seriales de salida SDOUT tiene una resistencia pull-down de $10K\Omega$, también hace que el circuito se establezca en modo esclavo, esta función admite la mayoría de frecuencias de muestreo que va de los 4 KHz a los 108 KHz e incluye una tabla que define la relación de modo de velocidad con la proporción entre los pines MCLK/SCLK/LRCK, como se observa a continuación en la tabla 3-5.

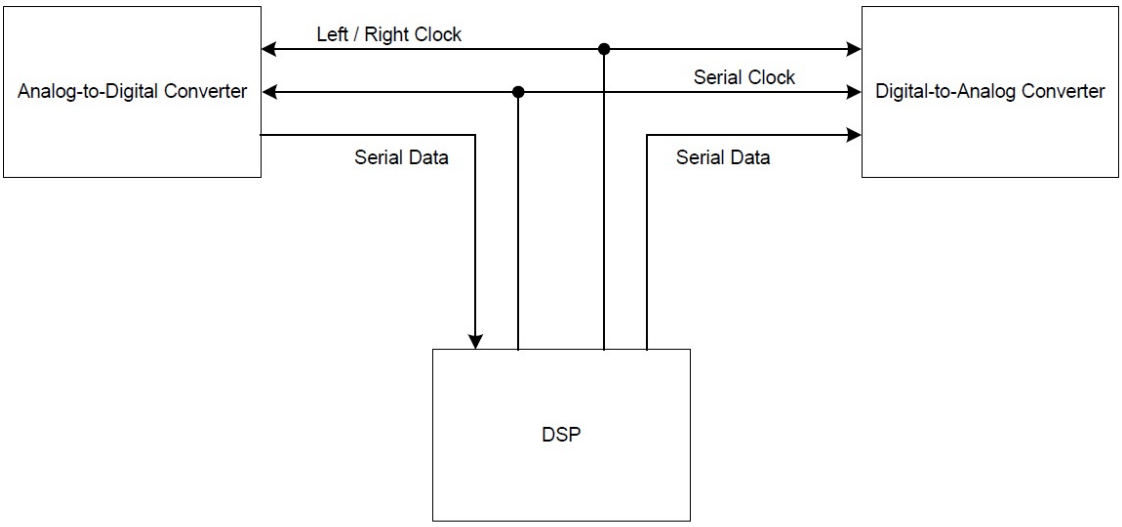


Figura 3-13: Interfaz I2S para entrada y salida del FPGA, ADC y DAC.

Modo de Velocidad	**Proporción MCLK/LRCK**	**Proporción SCLK/LRCK**	**Rango de entrada de muestreo (KHz)**
Modo de velocidad simple Single-Speed Mode	256x	64	4-54
	512x	64	4-54
	384x	48, 64	4-54
	768x	48, 64	4-54
Modo de velocidad doble Double-Speed Mode	128x	64	86-108
	256x	64	86-108
	192x	48, 64	86-108
	384x	48, 64	86-108

Tabla 3-5: Modos de velocidad con las frecuencias de muestreo (Fs) asociadas en modo esclavo.

Según se observa en la estructura SAI 3-13, se verifica que deben establecerse los relojes SCLK, LRCK y MCLK desde el proceso en el FPGA, por lo tanto, la interfaz I2S esta definida por las señales de entrada y salida siguientes:

- ***tx_mclk*** salida del transmisor de un bit para el reloj maestro.
- ***tx_lrck*** salida del transmisor de un bit para selección de palabra.

- ***tx_sclk*** salida del transmisor de un bit para el reloj de sincronización.
- ***tx_sdout*** salida del transmisor de un bit para los datos digitales.
- ***rx_mclk*** salida del receptor de un bit para el reloj maestro.
- ***rx_lrck*** salida del receptor de un bit para selección de palabra.
- ***rx_sclk*** salida del receptor de un bit para el reloj de sincronización.
- ***rx_sdout*** entrada del receptor de un bit para los datos digitales.

Desde los pines listados ***tx_mclk*** y ***rx_mclk*** es la conexión al reloj maestro el cual se conecta directamente de un pin que se nombrará ***axis_clk*** y tendrá una frecuencia de 22.579 MHz, se tiene 6 pines para control y datos en total, se agrega además un pin para reset del sistema ***axis_resetn***, el cual estará activo en bajo, se tiene una constante para los finales de conteo que se nombrara ***EOF_COUNT*** esto es con la finalidad de generalizar el bloque y si fuese necesario modificar este conteo, que resulte más sencillo al cambiar el valor de esta constante y no en todo el código.

La interfaz AXI esta definida por las señales de entrada y salida siguientes:

- ***tx_axis_s_data*** entrada del esclavo en forma vectorial para recepción de los datos.
- ***tx_axis_s_valid*** entrada del esclavo de un bit para reconocimiento de datos validos.
- ***tx_axis_s_ready*** salida del esclavo de un bit para activar recepción de un nuevo paquete.
- ***tx_axis_s_last*** entrada del esclavo de un bit para marcar último dato de un paquete.
- ***rx_axis_m_data*** salida del maestro en forma vectorial para envió de los datos.
- ***rx_axis_m_valid*** salida del maestro de un bit para reconocimiento de datos validos.
- ***rx_axis_m_ready*** entrada del maestro de un bit para activar envió de un nuevo paquete.
- ***rx_axis_m_last*** salida del maestro de un bit para marcar último dato de un paquete.

Por lo que la entidad con sus respectivas salidas y entradas se observa en el siguiente código:

```
1 entity axis_i2s2_vhdl is
2  Generic ( constant EOF_COUNT :std_logic_vector(8 downto 0) := "111000111"  );
```

```
3       Port ( axis_clk    : in std_logic;
4              axis_resetn : in std_logic;
5             --AXIS SLAVE INTERFACE
6              tx_axis_s_data  : in  std_logic_vector (31 downto 0);
7              tx_axis_s_valid : in  std_logic;
8              tx_axis_s_ready : out std_logic;
9              tx_axis_s_last  : in  std_logic;
10            --AXIS MASTER INTERFACE
11             rx_axis_m_data  : out std_logic_vector (31 downto 0);
12             rx_axis_m_valid : out std_logic;
13             rx_axis_m_ready : in  std_logic;
14             rx_axis_m_last  : out std_logic;
15            --I2S TRANSMITTER INTERFACE
16             tx_mclk  : out std_logic;
17             tx_lrck  : out std_logic;
18             tx_sclk  : out std_logic;
19             tx_sdout : out std_logic;
20            --I2S RECEIVER INTERFACE
21             rx_mclk : out std_logic;
22             rx_lrck : out std_logic;
23             rx_sclk : out std_logic;
24             rx_sdin : in  std_logic);
25  end axis_i2s2_vhdl;
```

Ahora de los 6 pines restantes para el control de datos, de la interfaz I2S se tiene a LRCK, que es al presentar un estado bajo indica que los datos corresponden al canal izquierdo de audio y en estado alto se indica que contiene a los datos del canal de audio derecho, se genera entonces un contador de 9 bits llamado “counter” que tiene como conteo máximo 512, el número es basado en la relación de la tabla 3-5 que define MCLK/LRCK a 512x, en el noveno bit se divide 256 ciclos en bajo y 256 en alto, así que se incluye esta señal sincronizada con el reloj maestro, MCLK surge de un módulo IP del IDE llamado “Clk Wizard”, éste a su vez, obtiene la señal del reloj del FPGA, por lo que del reloj principal de 100 MHz se reduce la frecuencia a 22.579 MHz y después a 44.1 KHz aproximadamente en LRCK, por lo que se deduce que esta señal debe contener la frecuencia que define al ancho de banda, análogamente la señal SCLK

esta a una proporción mayor 64 veces a LRCK, entonces,para obtener esta señal de sincronía del contador “counter” debemos ver la velocidad del tercer bit, que es una señal con frecuencia 2.822 MHz, esto se obtiene de la multiplicación 44.1KHz(64).

Por lo tanto se tienen seis pines ya definidos en la interfaz (tx_mclk, tx_lrck, tx_sclk, rx_mclk, rx_lrck, rx_sclk) con sus respectivas señales generadas en sincronía, solo restan dos señales, la entrada SDIN y la salida SDOUT, se sabe que estos puertos seriales contienen datos de 24 bits en complemento a 2 aunque se esperan 32 bits en conteo por la definición de la tabla 3-5, ya que si se cuenta con dos canales entre una proporción de 64 se tiene la relación de 32 ciclos para cada dato de 24 bits, se muestra en 3-14 el primer diagrama de flujo para el contador “counter” el cual inicia en cuanto se genera un flanco de subida en el reloj maestro MCLK y va sumando uno al conteo, así hasta llegar a 512 ciclos y al numero 511 que es el conteo final.

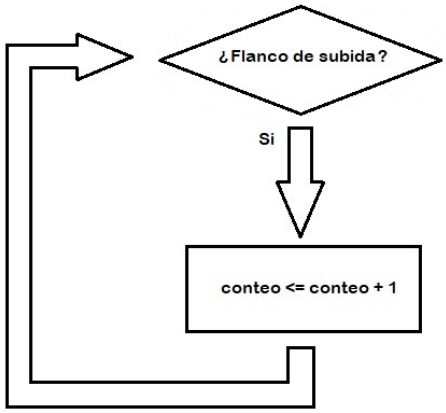


Figura 3-14: Proceso para conteo y sincronización en I2S.

Las señales generadas para los circuitos ADC y DAC con interfaz I2S se obtienen del contador, se deben recibir datos de manera serial y enviarse de manera paralela en AXI-S, para ello se cuenta por cada ciclo de SCLK 8 ciclos de MCLK que es nuestro reloj maestro, este se obtiene del modulo axis_clk. Se observa en la figura 3-15 que para cada dato en el bus SDATA con un ancho de N bits, un ciclo de SCLK equivale a un bit del dato en transmisión,por lo que hablar de ciclos de SCLK y bits del dato tienen una relación 1:1, se obtiene el dato completo del canal izquierdo durante la mitad del periodo cuando se encuentra en nivel bajo LRCK y el

dato completo del canal derecho en la siguiente mitad del periodo cuando se encuentra en nivel alto LRCK, tambien se puede observar que el comienzo de la transferencia del primer bit del dato sucede un ciclo de SCLK despues de que LRCK presenta un cambio de estado.

Tomando en cuenta el formato de señales de la figura 3-15 y asignando tamaños de datos y ciclos definidos para la transferencia en la tabla 3-5, se puede ver que los datos tienen un tamaño de 24 $bits$, para el dato del canal izquierdo se tiene 32 $ciclos$ de SCLK de los cuales 25 $ciclos$ corresponden como sigue: un ciclo que se encuentra entre la transición de LRCK y el inicio del envió de dato después se suman los 24$ciclos$ del dato correspondiente, como la duración de ciclos restantes son 7 $ciclos$, con esto suman los 32 $ciclos$ totales del dato del canal izquierdo, ahora para el canal derecho, se puede obtener el dato esperando solo los 25 $ciclos$ siguientes despues de los primeros 32 $ciclos$, por lo tanto se tiene una longitud total de 32 $ciclos + 25$ $ciclos$ o un total de 57 $ciclos$ de SCLK. Haciendo una conversión de ciclos de SCLK a MCLK tendríamos $57\,ciclos\,de\,SCLK * 8\,ciclos\,equivalentes\,de\,MCLK = 456\,ciclos\,en\,MCLK$, este valor se asigna a la constante EOF, ahora es posible visualizar que esta constante indica el fin del paquete, este numero EOF será necesario para saber en que momento transmitir los datos que se almacenaron en el registro auxiliar. Ahora los registros de desplazamiento que se sincronizan tomando como

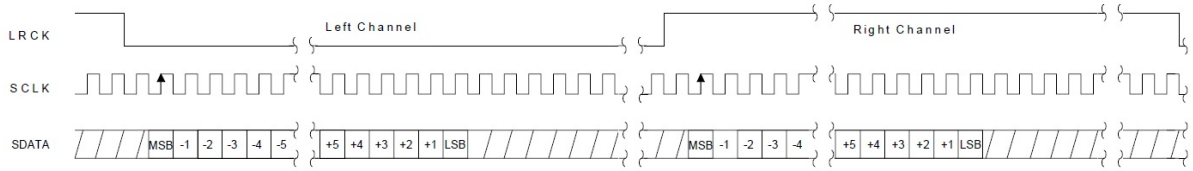


Figura 3-15: Formato de señales para sincronismo I2S, fuente:[12].

referencia a MCLK o axis_clk, ambas señales contienen los mismos valores en frecuencia, la función de este desplazamiento es servir de almacenamiento temporal a un conjunto de datos, aquí se adecua la carga y el acceso a ellos, se debe tener en cuenta que el sistema serial entrega un bit del dato cada ciclo de SCLK, por lo que se necesita identificar en tiempo o en una secuencia de conteo cuando se debe tomar el bit de la linea SDIN del ADC y almacenarlo con un valor posicional en un registro auxiliar, “counter” tiene exactamente el comienzo y final de

cada dato, solo se precisa verificar en que cambio de bit se delimita la existencia de los datos mismos, para eso se elaboró la siguiente tabla que delimita un valor exacto de conteo para cada bit del dato:

count			
Bit 8	Bits 7 a 3	Bits 3 a 0	Descripción
0	00001 00010 00011 . . . 10101 10110 10111 11000	000	canal izquierdo con Bit 8 en '0' inicio de conteo con tres bits count(3 a 0) = 000, 1 ciclo de sclk por 8 de mclk 24 bits seleccionados valor en decimal de bits seleccionados 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96 104, 112, 120, 128, 136, 144, 152, 160, 168 176, 184 y 192.
1	00001 00010 00011 . . . 10101 10110 10111 11000	000	canal derecho con Bit 8 en '1' inicio de conteo con tres bits count(3 a 0) = 000 24 bits seleccionados valor en decimal de bits seleccionados 264, 272, 280, 288, 296, 304, 312, 320, 328 336, 344, 352, 260, 368, 376, 384, 392, 400 408, 416, 424, 432, 440 y 448

Tabla 3-6: Bits de selección sincronizados por clk.

El proceso de selección de bits que se muestra en la tabla 3-6 será utilizado tanto para la recepción y transmisión de datos, salvo algunas pequeñas modificaciones en el envió del primer bit del dato en la transmisión, los registros que contienen al dato para ser desplazado en la parte del transmisor se asignan un ciclo antes a la frecuencia de MCLK del comienzo de cada bit del dato, exactamente en $count\ (3\ a\ 0)\ =$ "111", es decir que el dato de 24 bits se descompone en orden posicional, recorriendo los valores del MSB al LSB concatenando ceros para eliminar el

contenido del registro bit a bit, entonces se deberán recibir las señales de control AXI-S en la parte esclava y generarlas en la parte maestra.

Para poder estructurar los procesos de control se tiene que seguir la tabla 3-7, donde se eligen las condiciones de pines de entrada tanto en la interfaz maestra como esclava, entonces teniendo las condiciones establecidas en el protocolo AXI, se ve en que momento cambian las señales de salida regidas bajo esos cambios de las señales de entrada contenidas en el módulo.

pines salida pines de entrada	m_axis_valid '1'	m_axis_valid '0'	m_axis_last '1'	m_axis_last '0'	s_axis_ready '1'	s_axis_ready '0'
RESETN	1	0	1	0	1	0
m_axis_valid	0	1	1	0		
m_axis_ready	x	1	1	x		
m_axis_last	x	1	0	x		
count	455	0-444, 446-512	x	455	455	0
s_axis_valid					x	1
s_axis_ready					x	1
s_axis_last					x	1

Tabla 3-7: Funciones para control AXI-S.

Se debe recordar que estos procesos son llevados acabo en paralelo, por lo que se necesita establecer algunas formas secuenciales, definiendo la señal principal en la lista de sensibilidad, que en este caso es el reloj maestro, seguido a ello, la señal de “reset” es la que debe ser revisada y por último tomar decisiones referentes a la interfaz AXI-S.

Como ejemplo se tiene en la figura 3-16 un solo proceso concurrente con los demás, que se estructura secuencialmente, se observa que en un cambio del flanco de subida en el reloj se verifica después si existe el reinicio y seguido a ello las condiciones para tener el control de la salida “ready” de parte del esclavo.

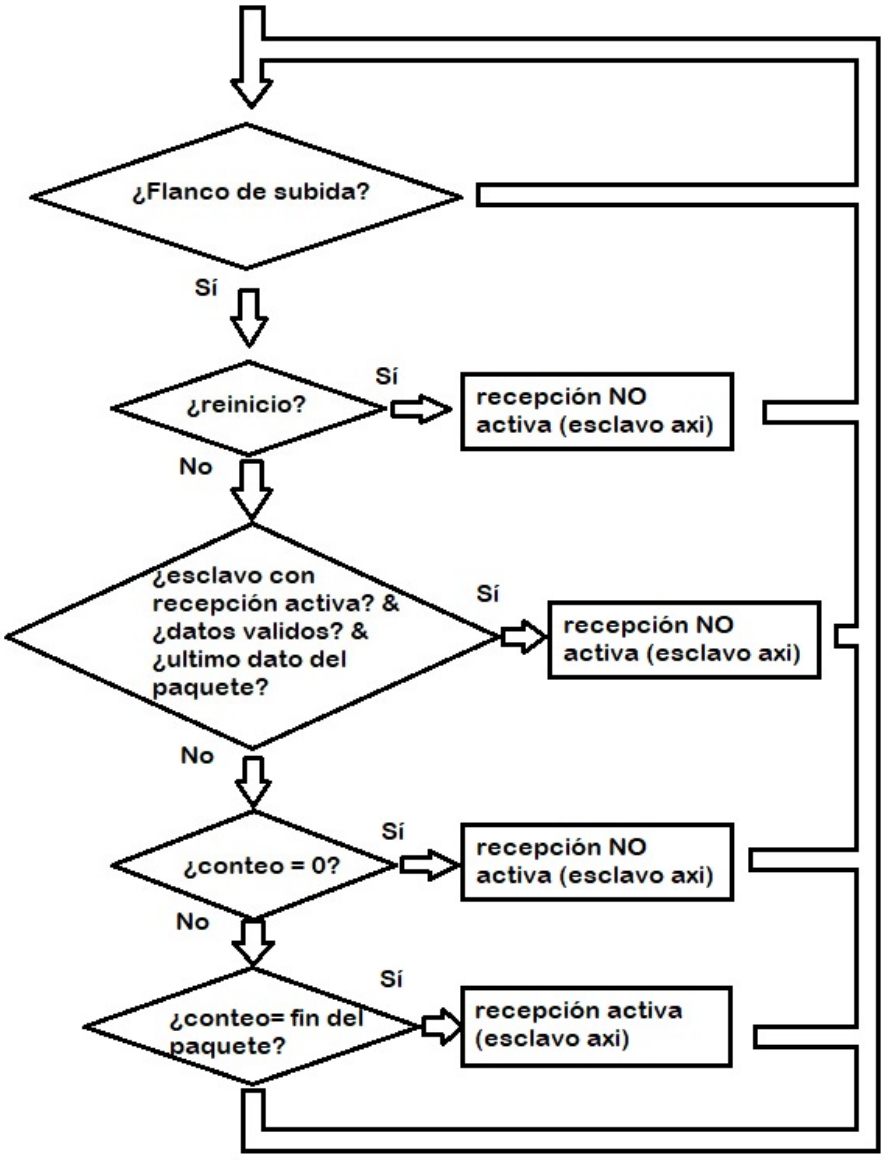


Figura 3-16: Restricciones para modificación de señal ready del esclavo.

Al final se obtienen los módulos conectados como se observa en la figura 3-17, se establecen las dos interfaces AXI-S e I2S, LRCK y SCLK se obtienen de "count" variando los bits especificados anteriormente, se tiene como referencia a [12], es posible consultar si requiere más detalles acerca de la interfaz serial y [5] para la interfaz AXI-S.

Se generará una prueba para verificar la funcionalidad del módulo, teniendo en cuenta que en el desarrollo de la prueba de este módulo es primordial darle valores iniciales a todas las

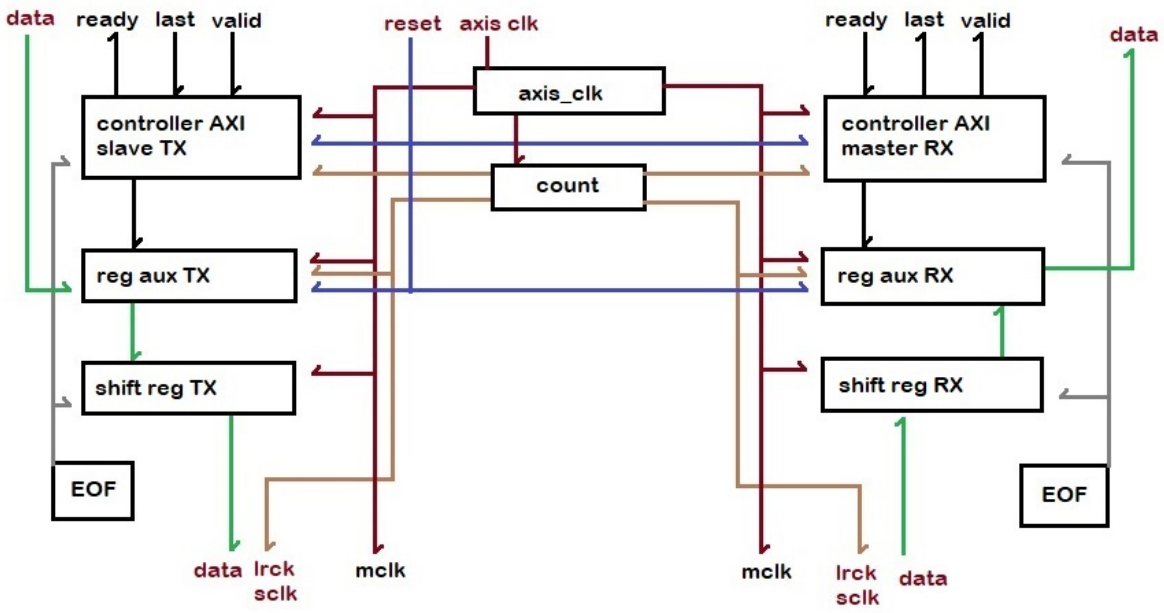


Figura 3-17: Diagrama de bloques I2S axi-s.

entradas, esto puede ser desde el mismo módulo (se recomienda esto) o puede ser dentro de la fuente de simulación, puesto que si existen valores “X”, “1 bajo”, etc. los procesos no se ejecutaran y por lo tanto no habrá respuesta del módulo. Considerando lo anterior se genera un archivo fuente de simulación en el proyecto, se obtiene el módulo a probar con las señales de entrada y salida, se hace la instancia correspondiente del módulo para su testbench y dentro del cuerpo del programa se genera un reloj, proceso cíclico donde se obtendrá una señal cuadrada de 100ns con un ciclo de trabajo de 50 %, se produce una señal serial con formato I2S de un paquete de dos datos de 24 bits, 57 bits distribuidos para la señal de entrada RX I2S formaran el paquete completo, por último se debe entregar en la entrada TX AXI-S dos datos de 32 bits en paralelo, estos dos datos conforman al paquete completo, a diferencia de I2S en este caso se incluyen las señales de control TVALID y TLAST para indicar la validez del dato y el tamaño del paquete, las dos interfaces AXI-S e I2S estarán recibiendo estos datos ciclicamente cada 456 periodos de MCLK, la diferencia es que uno se transmite TX en 2 periodos y el otro se recibe RX durante los 456 periodos.

Se genera una prueba y como resultado se obtiene este “timing diagram” 5-9 donde es

posible observar el comportamiento en el momento de la transferencia de datos tanto en el transmisor y receptor, se ve que "$count''$ ha llegado a "$1C7''$ que en decimal se traduce a "$455''$, correspondiente al final del paquete, es allí donde empieza el envió de parte del maestro AXI y termina la recepción del esclavo AXI, como se definió un periodo de 100ns se tiene en el diagrama la transferencia de datos en 45600 ns que es un periodo después de lo esperado, ya que la decisión se toma en un flanco de subida (rising edge), la transferencia empieza cuando "$valid$ y "$ready$ están en '1' en el mismo ciclo, después "$ready''$ se pone bajo o '0', mientras "$valid''$ continua hasta el segundo ciclo que contiene el último dato valido, y donde se pone en alto o '1' la señal "$last''$, se observa que en el ciclo 45700ns la señal last ha vuelto a bajo, se verifica que en los puertos de datos AXI, tanto RX como TX al final de la transmisión quedan en el puerto los datos aunque no se marcan como validos excepto en el ciclo de envió, después en 3-19, es enviado por el puerto un dato a TX, la parte esclava de AXI, el tamaño de este dato es de 28 bits en hexadecimal igual a "$0aaa0bbb''$, para observar como envía la parte del dato en forma serial.

En la figura 3-19 se muestra el intercambio de datos del receptor RX, en la parte de la interfaz I2S se cuenta con los datos que se deben generar por el ADC en el puerto rx_sdin_s en base binaria, en este caso el paquete completo esta formado por los bits que se muestran en el recuadro colo naranja, ambos están separados por un largo de 8 periodos, se aprecia que el dato correspondiente al canal izquierdo, encerrado en amarillo, es decir, cuando LRCK esta en "0" será "$aa0bbb''$ base hexadecimal y el canal derecho encerrado también en amarillo cuando LRCK esta en "1" será "$ddofff''$ base hexadecimal, estos datos como se ve en la figura 3-19 no cambian hasta el final del conteo $1c8$ de la señal "$count''$, en un ciclo se hace el proceso de conversión de serial a paralelo y se entrega el valor en esta figura 3-19, que no es visible por la gran diferencia en frecuencias, pero igualmente esta reflejado en la figura 5-9 en el puerto "$rx\_axis\_m\_data\_s''$, aunque se hace una notación del valor en el periodo igual a 45600ns.

Para el intercambio de datos en el transmisor TX se puede observar la figura 3-20, que contiene el dato enviado en forma paralela por la interfaz AXI encerrado en color verde con nombre de puerto $tx\_axis\_s\_data\_s$ en base hexadecimal, en este punto se hace una prueba

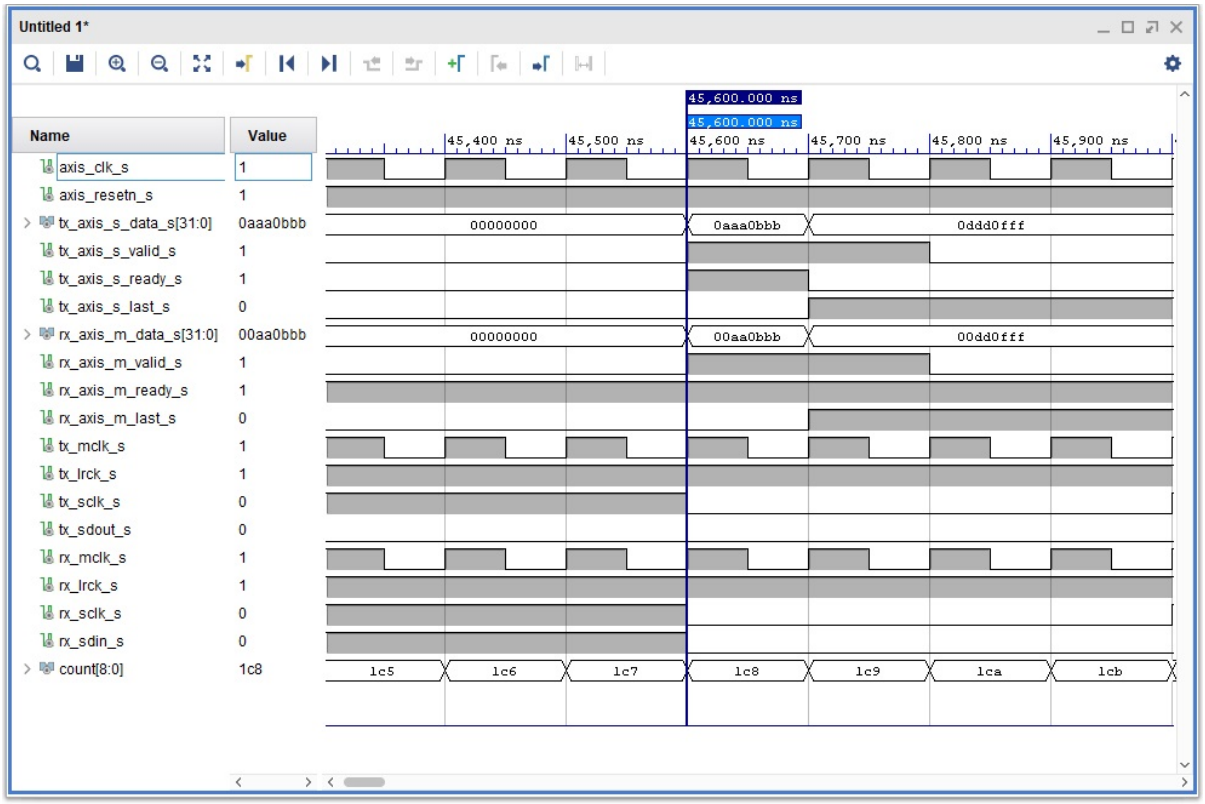


Figura 3-18: Diagrama de tiempos de bloque i2s axi-s del primer dato procesado.

para ver que sucede si se le envían 28 bits como datos, evidentemente el resultado a la salida después de la conversión de paralelo a serie son los datos en 24 bits, eliminando los bits más significativos, es decir, de los 28 bits de datos solo conserva los 24 menos significativos, aquí también se puede ver como el proceso se inicia nuevamente en el contador $count = 455$ donde el tiempo total es de 96800 ns que corresponde a 128 periodos de SCLK o dos paquetes de datos de audio estéreo encerrando el paquete serial completo en color morado y cada dato de color verde del puerto $tx\_axis\_s\_data\_s$ en base binaria.

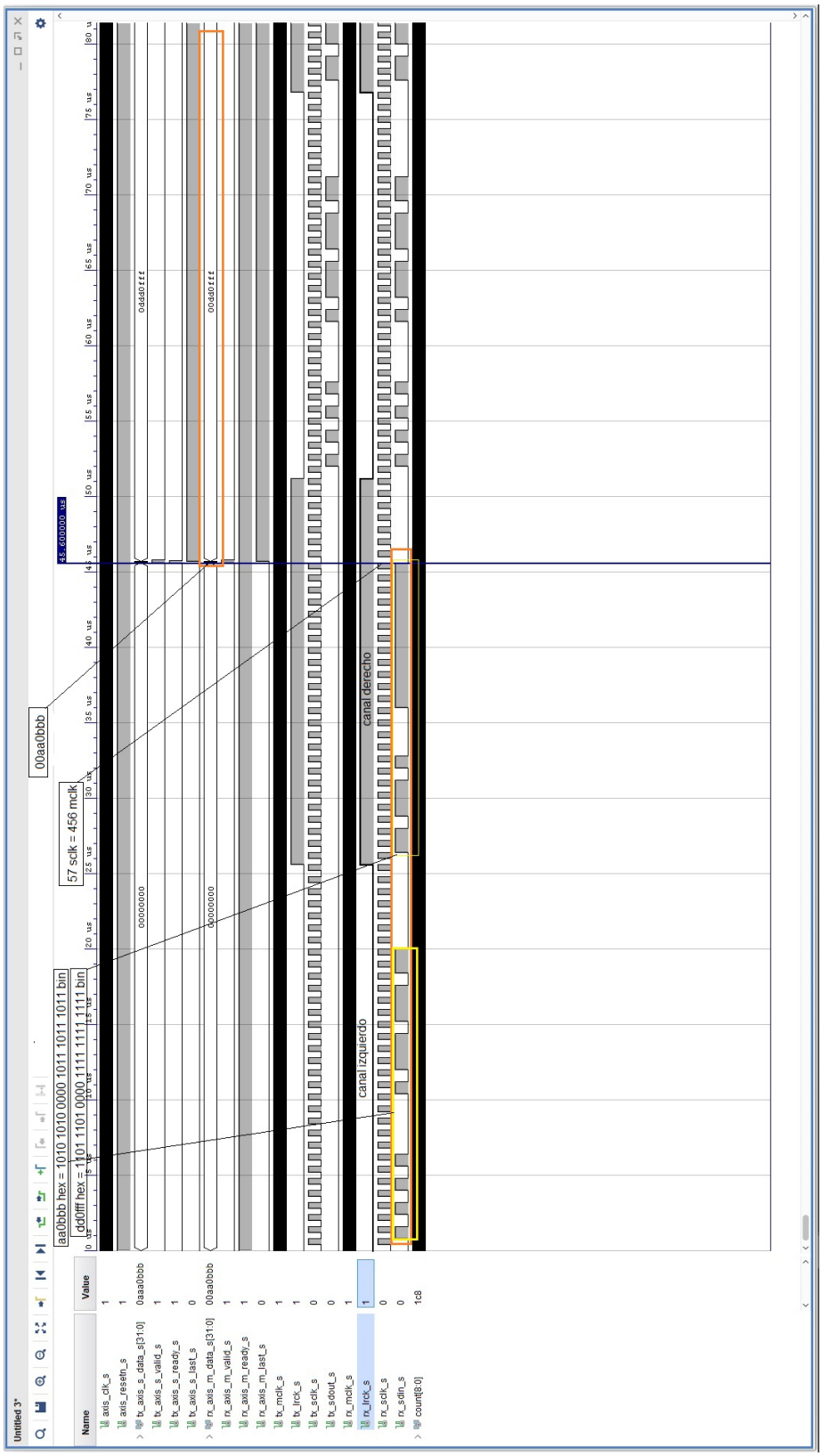


Figura 3-19: Diagrama de tiempos de bloque i2s axi-s del esclavo.

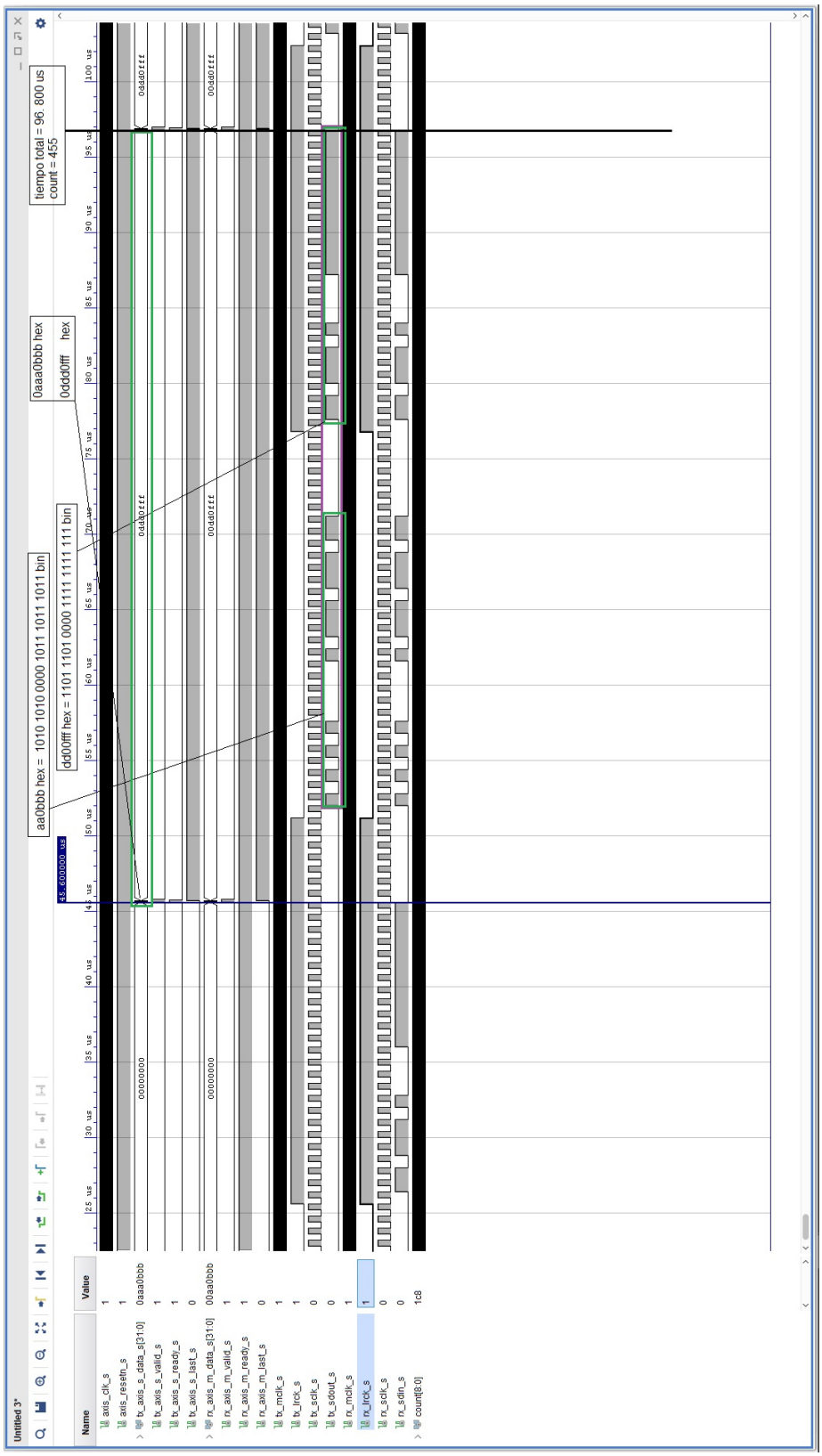


Figura 3-20: Diagrama de tiempos de bloque i2s axi-s del maestro.

4

Descripción del Entorno de Desarrollo Hardware y Software.

Los elementos que se necesitan para el desarrollo del proyecto incluyen Hardware y Software se describirán a continuación, ambos son propiedad intelectual de Xilinx® [1].

4.1. IDE Vivado Webpack 2018.1.

Vivado™ - HLx 2018.1:WebPack es un entorno de desarrollo integrado o por sus siglas en ingles IDE Integrated Development Environment, en él es posible crear proyectos en distintos niveles de complejidad, este IDE facilita el ambiente ideal para crear la mayoría de diseños electrónicos conglomerando los paquetes de software necesarios para ello, contiene además del compilador para VHDL y el editor de texto, un módulo de pruebas para el circuito, con el que se realiza las testbench(Banco de pruebas donde se generan estímulos para la verificación de un diseño), tiene RTL (Register-transfer level) análisis que muestra gráficamente el circuito comenzando modularmente hasta degradarlo en compuertas básicas para verificar la estructura electrónica creada, se mostraran algunas de las partes más importantes.

Contiene un directorio, que se muestra en la figura 4-1 del proyecto, en el cual se verifican

[1] Xilinx, Inc. Compania de tecnología estadounidense

los niveles jerárquicos “Hierarchy” de cada módulo “.vhd” dentro del proyecto, se muestra la dirección de cada módulo de la forma que se especificó en el tema de jerarquías, se pueden observar que se incluyen las “IP Sources” que se usan, los archivos de coeficientes “Coefficient Files” también se muestran junto con archivos de simulaciones y el archivo “.xdc” en el cual se definieron entradas y salidas en el mapa disponible en la Basys 3 [16].

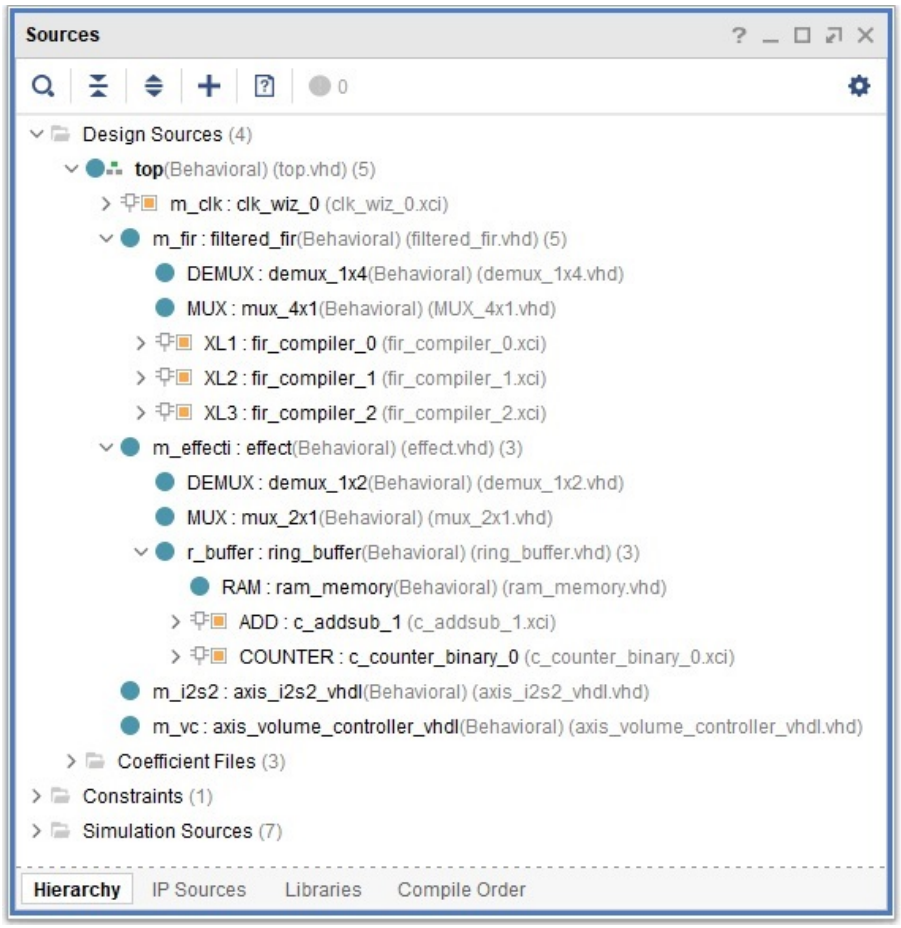


Figura 4-1: Directorio del proyecto en Vivado, obtenido del IDE Vivado.

Estos archivos fuente son los que deben ser instanciados en cada módulo con jerarquía superior hasta conectar todo con el archivo top, también hay una ventana con los mensajes de consola y rutas de diseño que básicamente son para reconocer los posibles errores y depurar el programa. En la figura 4-2 se muestran el contenido de los archivos escritos en VHDL, XDC, COE, etc, en su mayoría despliega el contenido de los archivos en el administrador del proyecto,

es posible ver que las palabras reservadas están remarcadas en colores diferentes a las entidades o también a las entradas y salidas, se observa la ruta del archivo expuesto.

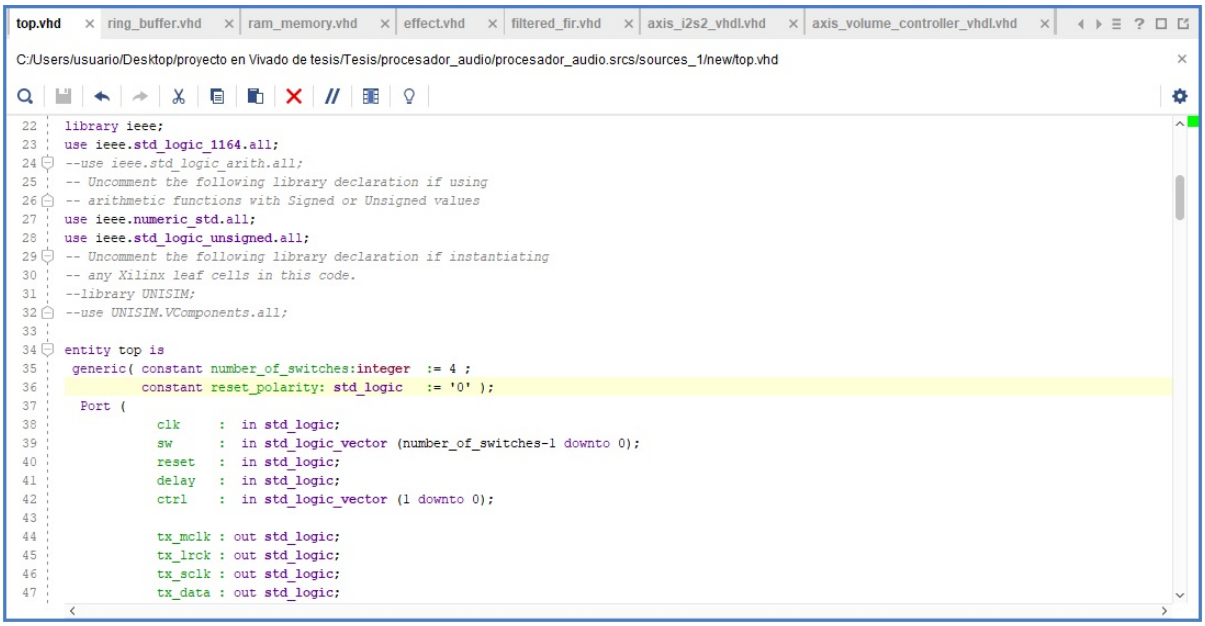


Figura 4-2: Archivos VHDL en Vivado, obtenido del IDE Vivado

Como tal el administrador del proyecto muestra cuatro ventanas, “Sources”, “Project Summary”, “Properties” y otra ventana multiproposito donde se cuenta con “Tcl Console”, “Messages”, “Log”, “Reports” y “Design Runs”, además de la barra multitarea y el navegador de flujo, este último se necesita ver a detalle para explorar las opciones que alberga. Hasta ahora se ha visto la primera opción incluida llamada administrador de proyecto o en inglés “PROJECT MANAGER”, que tiene otras sub opciones, como agregar archivos fuentes, cambiar lenguajes de compilación y un catalogo de IP para agregar al proyecto y una opción de configuraciones. Este apartado es donde se va a estar el 80 % del tiempo de diseño, lo demás se ocupara en tiempo de compilación junto con la generación del archivo bitstream “.bit” que es el requerido por el FPGA para su programación y después tiempo de simulación para verificar las variables en tiempo, etc.

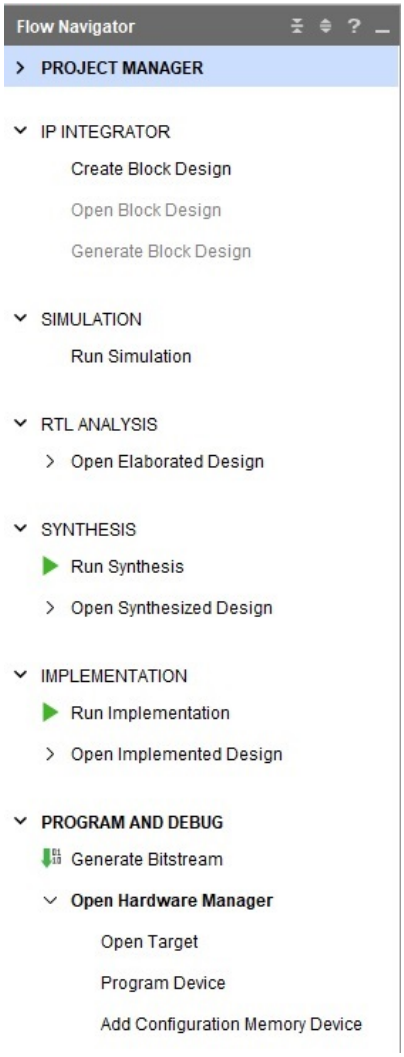


Figura 4-3: Barra de tareas en Vivado, obtenido del IDE Vivado

Toca observar la ventana “Flow Navigator” que ordena en su mayoría las capacidades de este IDE en la figura 4-3, se verifica la primer función que es administrador de proyecto, después viene el apartado de “SIMULATION”, este apartado es para la verificación del módulo diseñado, se comprueba por medio del diagrama de temporización que son creados dependiendo las entradas y salidas contenidas en la entidad, es bueno para poder verificar el estado inicial de todas las entradas y registros, así como las respectivas salidas, debe tenerse en cuenta que se debe crear un archivo en “.vhd” donde se almacenan las características que definen las condiciones de entrada al módulo en prueba, se hace con al menos un proceso y es posible verificar las señales digitales en el tiempo, los valores sucesivos de cada salida, además se pueden agregar otras variables contenidas dentro de la misma entidad, todo este proceso será mostrado en cada creación de un módulo requerido.

El “RTL ANALYSIS” mediante el código sintetiza y crea entidades electrónicas en el diagrama “Schematic”, hay otro compendio “Open Elaborated Design” que solo es un contenedor de opciones como diagrama, “Report methodology”, “Report DRC”, “Report Noise”, aunque solo se ocupa el diagrama para verificar de forma gráfica las conexiones y elementos electrónicos esperados por la descripción en el código.

De laa “SYNTHESIS” que se encarga de correr el compilador sobre el código hecho, también se obtiene otro “Schematic” con la diferencia con el anterior que éste es de forma sobre la cantidad de LUT’s utilizada en el FPGA, existe la posibilidad de revisar un Reporte del consumo de los arreglos establecidos por el diseño en “Report Power”.

Después “IMPLEMENTATION” activa la función para crear el módulo sintetizable del código, este es un proceso intermedio entre la síntesis y la generación del bitstream, dentro de la implementación se puede modificar el número de cores con los cuales trabajar para este proyecto, por ejemplo se puede establecer un equipo que cuenta con un procesador Intel core i3 que tiene 4 núcleos, los mismos que pueden ser utilizados para trabajar el mismo proceso, que es costoso hablando computacionalmente, esta opción se puede ver en la figura 4-4.

Las caracteristicas adicionales que contiene la opción de implementación tienen reportes como redes de reloj, interacciones con reloj, metodología, “DRC” “Design Rule Check” o en español “verificación de reglas de diseño”, conjunto de temporizaciones, ruido, etc. Regularmente se ocupará solo la implementación para la construcción del módulo sintetizable, si es necesario se pueden verificar todas sus opciones dentro del IDE en la opción “Help”, “Documentations and tutorials”.

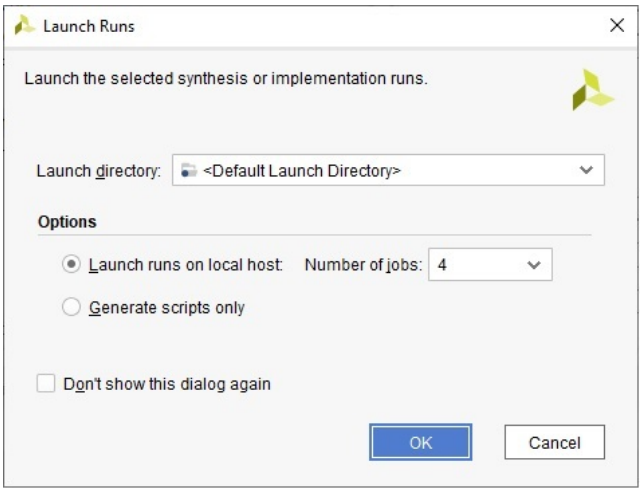


Figura 4-4: Opción de procesamiento por número de núcleos, obtenido del IDE Vivado

Finalmente se tiene la opción de “PROGRAM AND DEBUG”, aquí es posible activar la generación del “Bitstream” el cual es el archivo que pondrá en el funcionamiento esperado al FPGA, para cualquier modificación en el código debe hacerse una síntesis para compilar el código, después, si esta en orden se puede hacer la implementación y al final generar el bitstream. Se tiene que “Open Hardware Manager” despliega tres opciones “Open target” en cual se hace la conexión con la tarjeta de desarrollo en modo “localhost” dispositivo local, esta interconexión es básicamente para comunicar al IDE y FPGA , “Program Device” activa la orden de enviar el archivo “.bit” con la ruta especificada al FPGA, es decir enviar todas las conexiones que deben llevarse acabo dentro del mapa de LUT’s en el circuito y “Add Configuration Memory

Device" el cuál almacena la configuración por medio de la memoria flash contenida en la tarjeta de desarrollo. Dentro de esta opción "Add Configuration Memory Device", es posible modificar la función de la memoria en la tarjeta Basys 3™ para que mantenga almacenado el archivo ".bit" en la memoria flash via JTAG enviado por SPI y pueda cargarse automáticamente sin la necesidad de conectarse a la computadora y programarse desde el IDE , el archivo para generar esta posibilidad estará en ".mcs". Con esto se concluye la parte de Software Vivado™ 2018.1, para información adicional, se puede consultar en [38].

4.2. Tarjeta de desarollo Basys 3.

La parte de Hardware, además del PMOD I2S2 visto antes, es la tarjeta de desarrollo Basys 3 ™, que incluye un FPGA Artix-7, para las cuales se verificaran todas sus características para visualizar las utilidades físicas disponibles en esta tarjeta.

Esta tarjeta contiene un FPGA XC7A35T-1CPG236C de bajo costo, cuenta con un conjunto de puertos USB, VGA, switches, LEDs y otros dispositivos de entrada y salida, estos dispositivos pueden ser adicionados en sus entradas respectivas se trata de los PMODS de Digilent® haciendolo más versátil y ampliando la utilidad en el diseño a esta tarjeta de desarrollo, se listan sus características básicas a continuación:

- 33 280 celdas lógicas en 5200 slices (cada slice contiene cuatro LUTs de seis entradas y 8 flip-flops) .
- 1800 Kbits de fast block RAM.
- 90 DSP slice.
- cinco administradores de reloj, cada uno con lazo de enganche de fase (PLL Phase-Locked loop).
- reloj interno de hasta 450 Mhz
- un convertidor On-chip analog-to-digital converter (XADC)

En la imagen 4-5 se aprecia cada parte enumerada y dentro de la tabla 4-1 su correspondiente descripción.

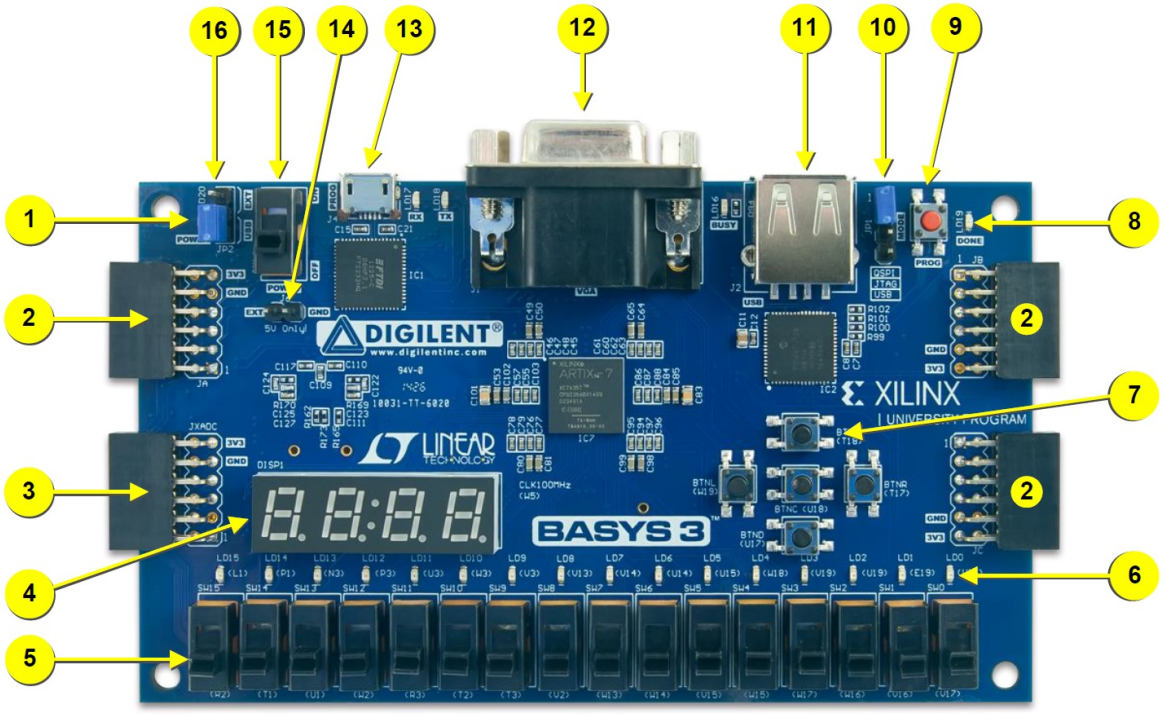


Figura 4-5: Tarjeta de desarrollo Basys 3, fuente:Digilent® Basys 3™ FPGA Board Reference Manual.

periferico	Descripción del componente	periférico	Descripción del componente
1	Led de encendido	9	boton de reset para configuración del FPGA
2	Puertos Pmod	10	Jumper para modo de programación
3	Puerto (XADC) señal análoga	11	conector USB Host
4	Display de cuatro digitos de 7 segmentos	12	conector VGA
5	16 switches	13	Puerto USB para comunicación UART o JTAG
6	16 LEDs	14	conector para fuente de alimentación externa
7	5 PushButtons	15	Switch de encendido
8	Led de estado de Programación del FPGA	16	Jumper de selección de fuente de alimentación

Tabla 4-1: Partes de Tarjeta Basys 3.

Para la ***alimentación*** de esta tarjeta hay dos opciones, una por medio del micro USB (J4) y la segunda por una fuente de alimentación externa que tiene por entrada el jumper J6, esta debe ser de 4.6 v a 5.5 v con 1 A de capacidad, para seleccionar cual de ellas se utilizará se debe mover el JP2 a EXT en caso de fuente externa o USB para la alimentación desde el micro USB, el LED LD20 indica en conjunto con un regulador de voltaje LTC3633 que los voltajes y corrientes son apropiados para su funcionamiento. Si se utiliza el conector USB host debe tener el voltaje externo a 4.6V como mínimo pero si no es usado el conector USB medinte el selector J2 entonces con 3.6V será suficiente pues el voltaje máximo ocupado dentro de la tarjeta es 3.3V, es posible obtener unos niveles de voltaje manejados en la siguiente tabla4-2 El modo de

Voltaje	Uso	Dispositivo Regulador	Corriente (Max./Tip.)
4.6 V a 5.5 V	Reguladores de voltaje y USB Host	No hay regulación	1.5 A / 1.0 A
3.3 V	I/O de FPGA, P. USB, Relojes, M.Flash, PMODs	IC10: LTC3633	2A/0.1A - 1.5A
1.8 V	Bloques auxiliares del FPGA y memoria RAM	IC11: LTC3621	300mA/0.05A - 0.15A
1.0 V	Nucleo del FPGA	IC10: LTC3633	2A/0.2A -1.3A

Tabla 4-2: Opciones de alimentación Basys 3.

programación del FPGA en la tarjeta Basys 3 puede llevarse por 3 medios diferentes, estos medios son seleccionables a través del jumper JP1, estas tres opciones son, QSPI, desde un archivo almacenado en la memoria SPI FLASH se envía por medio del puerto SPI integrado al FPGA, JTAG a traves del puerto J4 por medio de una computadora, cuando esté encendida la tarjeta se puede programar el FPGA y en el USB estando conectada una memoria con la configuración almacenada.

El archivo de configuración para el FPGA Artix-7 35T requiere 17 536 096 bits de almacenamiento, este contenido corresponde con la configuración que resulta en un archivo bitstream que al enviarse al FPGA primero se comprime para evitar tiempos largos de carga y dentro del mismo se descomprime en la memoria RAM. Incluye una Memoria Flash SPI de 32Mb (4MB) conectada al FPGA por medio de un puerto SPI QUAD marca Spansion S25FL032, la configuración del FPGA emplea aproximadamente 2MB por lo que se ocupa el 48 % de dicha memoria, dejando el resto para el uso del usuario. La tarjeta también contiene un ***oscilador*** de 100 MHz conectado al pin W5, el pin esta optimizado para señales de reloj pues forma parte de

una capa especializada para llos MMCM (Mixed-Mode Clock Manager) que pertenece al banco 34. Se puede conectar la señal de reloj al MMCM o al PLL (Lazo de enganche de fase) para generar señales de reloj de distintas frecuencias y fases que se requieran. Existen restricciones en su uso, si se quieren saber más de ellas se puede consultar[38]. También se hará mención del IP CORE Clocking Wizard IP que facilita la generación de señales de reloj, este IP crea instancias en MMCM o PLL como sea necesario dependendiendo la frecuencia y fase deseada, esta IP es altamente recomendada puesto que su utilización aparte de facilitar el diseño, hace uso óptimo de los recursos de la tarjeta de desarrollo [9].

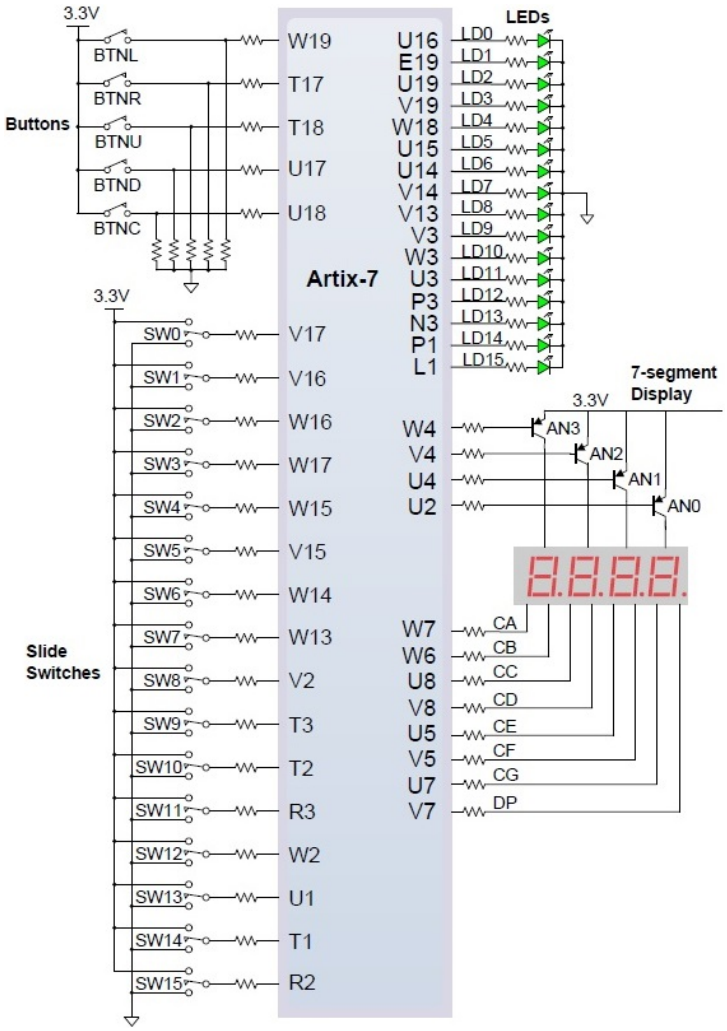


Figura 4-6: I/O Basys 3, fuente:Digilent® Basys 3™ FPGA Board Reference Manual.

Se tienen dispositivos de uso común para aplicaciones lógicas estándar: 16 interruptores, 5 pushbotton, 16 led, 4 pantallas de 7 segmentos. En la figura 4-6 se muestra la conexión, se puede verificar que los interruptores tienen resistencias además de las pull-down para evitar cortos circuitos, de la misma manera, los pushbutton contienen dichas resistencias, los display de 7 segmentos se multiplexan en tiempo con ánodo común y los leds tienen una lógica de funcionamiento sobre su encendido con “1” lógico.

La última parte del hardware base son los conectores PMOD, como se deduce su principal función es agregar hardware a la Basys 3™

, los módulos PMOD constan de 12 pines (8 de datos, 2 de alimentación y 2 de GND) dentro de los cuales contienen pines de alimentación, así no es necesario agregar alimentaciones al hardware adicionado, siempre y cuando el consumo no sobrepase de 1A, a continuación, se muestra en la imagen 4-7 la configuración de estos conectores y en la tabla 4-3 se pueden asociar los puertos con sus respectivos pines dentro de la tarjeta de desarrollo. Si se necesitan más detalles de algún dispositivo no incluido en el tema, se puede consultar el manual de la tarjeta de desarrollo Basys 3™ [16].

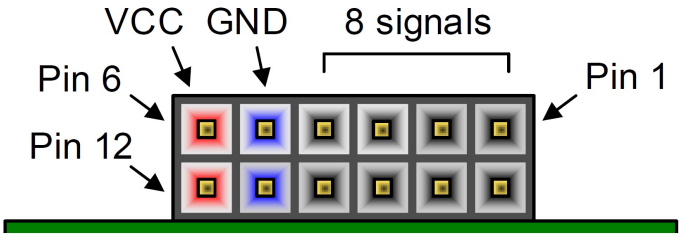


Figura 4-7: Pines pmod Basys 3, fuente:Digilent® Basys 3™ FPGA Board Reference Manual

Pmod JA	Pines JA	Pmod JB	Pines JB	Pmod JC	Pines JC	Pmod XDAC	Pines XDAC
JA1	J1	JB1	A14	JC1	K17	JXADC1	J3
JA2	L2	JB2	A16	JC2	M18	JXADC2	L3
JA3	J2	JB3	B15	JC3	N17	JXADC3	M2
JA4	G2	JB4	B16	JC4	P18	JXADC4	N2
JA5	GND	JB5	GND	JC5	GND	JXADC5	GND
JA6	VCC	JB6	VCC	JC6	VCC	JXADC6	VCC
JA7	H1	JB7	A15	JC7	L17	JXADC7	K3
JA8	K2	JB8	A17	JC8	M19	JXADC8	M3
JA9	H2	JB9	C15	JC9	P17	JXADC9	M1
JA10	G3	JB10	C16	JC10	R18	JXADC10	N1
JA11	GND	JB11	GND	JC11	GND	JXADC11	GND
JA12	VCC	JB12	VCC	JC12	VCC	JXADC12	VCC

Tabla 4-3: Asignación de pines Basys 3

5

Módulo de Xilinx® (FIR compiler).

5.1. Filtro pasa bajas.

Un filtro es un sistema que atenúa o elimina discriminando parte de una señal que procesa, dado que esto puede aplicarse a múltiples entidades electrónicas, se enfocará específicamente en los que dada cualquier señal, permiten el paso de un determinado tipo de frecuencias, atenuando y o eliminando el resto [28].

Ahora se puede definir el filtro pasa bajas, básicamente permite el paso de las frecuencias bajas y atenúa o elimina las frecuencias superiores que están separadas por un límite llamado frecuencia de corte, como se observa en 5-1, la f de la imagen no es una medida que tenga unidades en Hz, es la frecuencia normalizada por la frecuencia de muestreo.

Las frecuencias de paso y de atenuación tienen en común la frecuencia de corte fc, esto es para el filtro ideal, pero realmente esto no es aplicable, ya que se tienen margenes que aumentan el ancho de la frecuencia de corte como se muestra a continuación en la figura 5-2

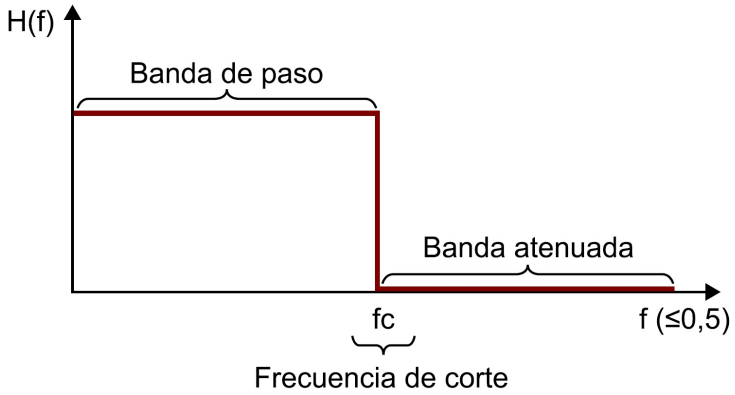


Figura 5-1: filtro pasabajas ideal.

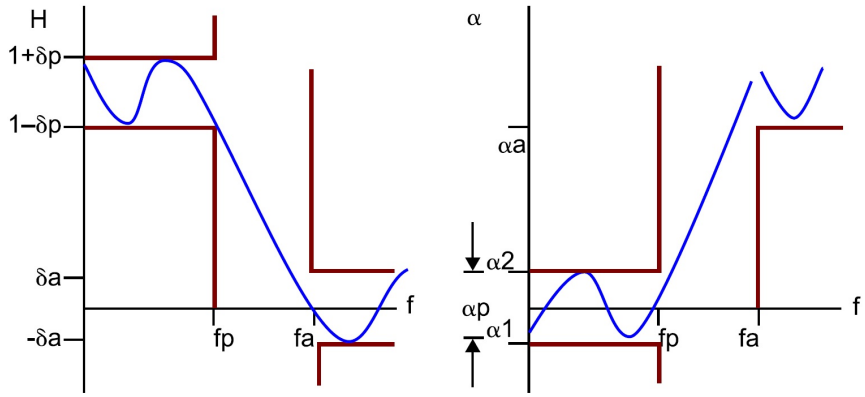
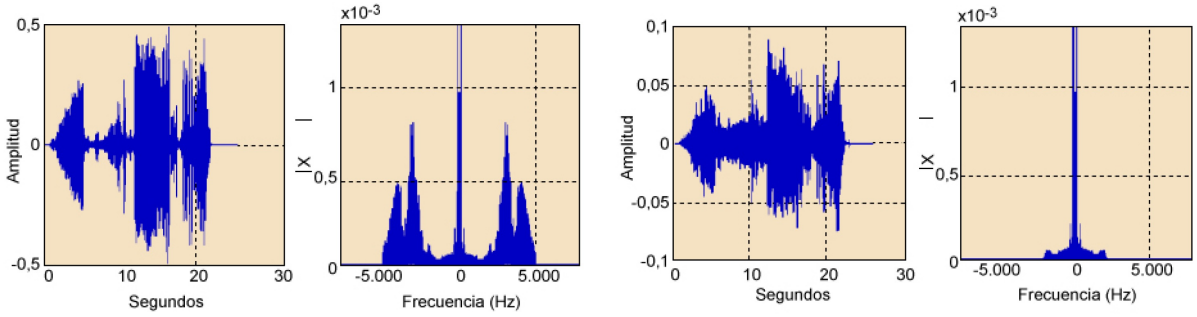


Figura 5-2: Señal obtenida de un FPB y respuesta de atenuación del mismo.

- f_p frecuencia de paso (frecuencia límite de paso).
- f_a frecuencia de atenuación (frecuencia donde empezará la atenuación).
- δ_p tolerancia de paso en módulo, que es la relajación en módulo que se permite en la banda de paso.
- δ_a tolerancia de atenuación en módulo, que es la relajación en módulo que se permite en la banda de atenuación.
- $\alpha_p = \alpha_2 - \alpha_1$ tolerancia de paso en atenuación, que es la relajación en atenuación que se permite en la banda de paso.
- $\alpha_a$tolerancia de atenuación en atenuación, que es la relajación en atenuación que se permite en la banda de atenuación.

La relajación de estos parámetros hace que el filtro tenga menos costo computacional, si se generan parámetros muy apegados a los ideales se verá reflejado en un alto número de coeficientes y por lo tanto mayor complejidad al implementar. Este costo es específicamente para filtros digitales, ya que se requiere hardware muy especifico para el manejo de cada coeficiente y dato en un filtro, se debe recordar que el proceso principal es una suma convolutiva donde la separación en componentes frecuenciales, procesado con cada una de estas junto a los coeficientes y por último la unión de la señal ya operada, son procesos que al aumentar su proporción significará un gran costo en recursos, caso contrario a los filtros analógicos que operan la señal entera al mismo tiempo.

Para entender gráficamente como trabajan los filtros a continuación se tiene la imagen 5-3, en donde se observa una señal de audio tanto en el dominio del tiempo como el dominio de la frecuencia, se observa la figura 5-3(a) en dominio del tiempo de 30 segundos y rango en amplitud de 0.5 a -0.5 , a su izquierda esta la representación de su espectro en frecuencia, este se obtiene mediante la transformación matemática FFT (Fast Fourier Transform), se observa que es simétrica dado que se representa con su parte imaginaria y real, tiene una frecuencia de hasta 5KHz, el filtro aplicado será un pasa-bajas con frecuencia de corte de 2KHz obteniendo como respuesta la 5-3(b) donde es muy difícil percatarse del filtrado en la gráfica del dominio del tiempo, pero resulta evidente el filtrado en el espectro de frecuencia, es posible ver que las componentes después de 2KHz han sido eliminadas o atenuadas.



(a) sonido a filtrar (b) sonido filtrado

Figura 5-3: Atenuación de componentes en frecuencia mediante un filtro pasabajas, fuente:[28]

5.2. Filtro pasa altas.

Es debido mencionar que el filtro base es el filtro pasa bajas, puesto que de este se pueden eleborar todos lo demás, por lo que el filtro pasa altas es el complemento del pasa bajas, la propiedad de este filtro es bloquear frecuencias contenidas debajo de la frecuencia de corte y dejar pasar aquellas que sean mayores a esta.

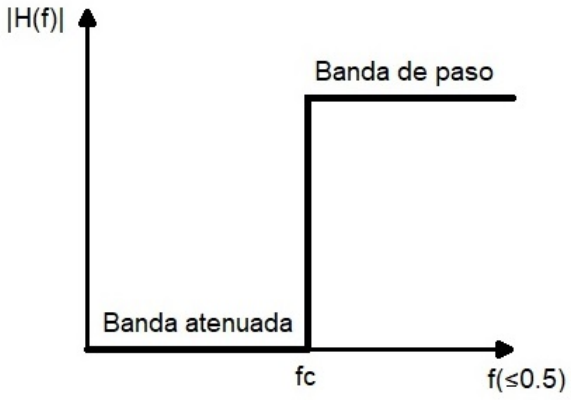


Figura 5-4: Filtro pasa altas ideal.

Este filtro tiene los mismos parámetros de diseño que el pasa bajas de la figura5-2.

5.3. Filtro pasa banda.

Este filtro puede transmitir un rango de frecuencias (banda de paso) y atenuar o eliminar dos bandas (bandas de rechazo o atenuadas), esto es, se tiene la banda de paso de forma central con respecto de las bandas atenuadas por lo que se deduce que tiene dos frecuencias de corte, $fc1$ que estará entre la banda con frecuencias menores a la banda de paso y $fc2$ que será entre la banda de paso y la banda con frecuencias mayores a esta la cual será rechazada. Existe un termino para llamar al conjunto de frecuencias que se dejaran pasar llamado Ancho de Banda o Bandwidth en ingles, comprende la diferencia entre las frecuencias que limitan la banda de paso, es posible ver esto como sigue:

$$BW = fc2 - fc1 \tag{5-1}$$

y además se tiene una frecuencia central de la banda de paso que es la media geométrica de las frecuencias de corte:

$$F_0 = \sqrt{fc1fc2} \tag{5-2}$$

En la figura 5-5 se puede apreciar gráficamente como se comporta la función de atenuación en esta configuración de filtrado, los parámetros de construcción son los mismos que el pasa bajas y el pasa altas, así que estas tres funciones de los filtros son las implementadas en el proyecto, hay otra función complementaria de este filtro que corresponde al supresor de banda, no lo implementaremos pero es junto con el filtro pasa todo las funciones más populares del filtrado de señales [31] .

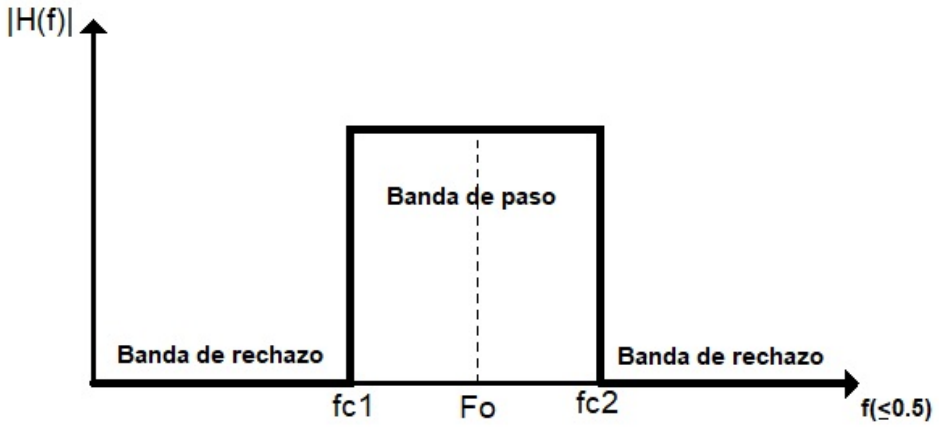


Figura 5-5: Filtro pasa banda ideal.

5.4. Base teórica de filtros FIR.

Básicamente existen solo dos tipos de filtros digitales, los FIR (Finite Impulse Response) y los IIR (Infinite Impulse Response) de los cuales se selecciona a los FIR para aplicarlos en el proyecto, se comprobará el porque de su selección para el mismo, se deben apreciar características de los filtros FIR para poder separarlos de los segundos, también se describirán las características teóricas del diseño de los filtros, no es el fin de este trabajo ahondar en esta teoría o aspectos muy profundos acerca del tema, pero es esencial revisar las bases para manejar

adecuadamente los parámetros en las soluciones existentes que proporciona el IDE de Xilinx® , ya que se trabajará con herramientas para el diseño de los filtros en los cuales se necesita una comprensión de sus parámetros y al mismo tiempo requiere un nivel de conocimiento acerca de la implementación de filtros FIR [28], se listan las ventajas de estos filtros a continuación:

- Sistema lineal e invariante en el tiempo.
- Tiene fase lineal.
- Siempre son estables es decir realizables.
- Los métodos de diseño son generalmente lineales.
- Pueden ser realizados eficientemente en hardware.

Un filtro es un sistema lineal e invariante en el tiempo, estas características juegan un papel fundamental en el análisis de señales y sistemas por dos razones principales, la primera es la facilidad de modelado, como ejemplo se cuenta a los sistemas físicos, que adicionan un conjunto de propiedades y herramientas para su análisis, una de estas propiedades es la superposición, se debe recordar que un sistema lineal en tiempo continuo o en tiempo discreto es aquel que tiene como propiedad la superposición que es: "si una entrada consiste en la suma ponderada de varias señales, entonces la salida es simplemente la superposición (es decir, la suma ponderada) de las respuestas del sistema a cada una de estas señales, matemáticamente, sea $y_1(t)$ la respuesta del sistema continuo de una entrada $x_1(t)$, y sea $y_2(t)$ la salida correspondiente a la entrada $x_2(t)$. Entonces el sistema es lineal si:"[2]

1. La respuesta a $x_1(t) + x_2(t)$ es $y_1(t) + y_2(t)$.

2. La respuesta a $ax_1(t)$ es $ay_1(t)$, donde a es una constante compleja cualquiera.

La primer propiedad es la aditividad, la segunda es el escalamiento, que denotan a un sistema LIT (lineal e invariante en el tiempo), de donde se deduce que si se puede representar la entrada de un sistema LIT en términos de una combinación lineal de un conjunto de señales básicas, entonces es posible usar la superposición para calcular la salida del sistema en términos de sus respuestas a estas señales básicas, en términos simples se puede decir que se separan las

componentes a filtrar provenientes de una señal, para operar de manera simple, después sumar las componentes individuales ya operadas, obteniendo el mismo resultado que si se operára al conjunto completo proveniente de la señal de entrada. Estas propiedades valen tanto para señales continuas como señales discretas.

Un sistema es invariante en el tiempo si las características y comportamientos de este son fijos en tiempo. Por tanto, un sistema LIT tiene la propiedad de que si se conoce la respuesta al impulso es posible obtener la salida para cualquier entrada retardada en el tiempo, se puede caracterizar con lo que se conoce como respuesta impulsional, esta respuesta relaciona la entrada y salida de un sistema lineal e invariante en el tiempo, además esta propiedad asegura que el filtrado de una suma de señales de audio se puede obtener como la suma de cada señal filtrada por separado, pero antes se probrará la propiedad de invariancia en el tiempo, considerando un sistema continuo definido como sigue:

$$y(t) = sen[x(t)]. \tag{5-3}$$

se debe determinar si cumple para cualquier entrada y desplazamiento en el tiempo t_0. Entonces, sea $x_1(t)$ una entrada arbitraria al sistema y

$$y_1(t) = sen[x_1(t)]. \tag{5-4}$$

sea la salida correspondiente. Ahora se considerará una segunda salida $x_1(t)$ con un desplazamiento en tiempo:

$$x_2(t) = x_1(t - t_0). \tag{5-5}$$

La salida correspondiente a esta entrada es

$$y_2(t) = sen[x_2(t)] = sen[x_1(t - t_0)]. \tag{5-6}$$

De manera similar, de la ecuación 5-4

$$y_1(t - t_0) = sen[x_1(t - t_0)]. \tag{5-7}$$

Comparando las ecuaciones 5-6 y 5-7, se observa que $y_2(t) = y_1(t - t_0)$ y por tanto, se prueba que este sistema es LIT.

Al hablar de la respuesta al impulso unitario discreto o continuo se entiende que las señales se pueden representar como una combinación lineal de impulsos retardados, se debe tener en cuenta que es una característica que sumado a las propiedades de superposición e invarianza en el tiempo permiten hacer la caracterización de cualquier sistema en términos de su respuesta a este impulso, es aquí donde una vez reconocidas todas estas propiedades se puede llamar a todo esto como "suma de convolución" en forma discreta o "integral de convolución" en tiempo continuo.

Después, para caracterizar todos estos sistemas en forma continua se utilizan ecuaciones *diferenciales* lineales con coeficientes constantes y los sistemas discretos por ecuaciones de *diferencias* lineales con coeficientes constantes, la teoría abarca hasta este punto, puesto que no se desarrollará más allá de estas bases, pero se puede profundizar en [2](pag 116).

Las desventajas de los FIR es que casi siempre requieren un orden muy grande en comparación con los IIR para que tengan un buen rendimiento, es decir, que se tendrán una gran cantidad de coeficientes para trabajar y es por ello que son más costosos en hardware, significa tener conjuntos de cientos de coeficientes aproximadamente, es por ello que esta IP de filtrado, propiedad de Xilinx® , ayuda a su uso sin la necesidad de preocuparse por la implementación.

Se habló sobre algunas características de los FIR, entre ellas la fase lineal, esta característica quiere decir que la respuesta al impulso será simétrica o asimétrica, estas propiedades hacen que si se tiene un filtro con fase Lineal, significará que no distorsiona la forma de onda de la señal original, algo que no es esencial en las aplicaciones para audio pero que se tomara particularmente como prioridad, ya que el oído humano genéricamente hablando no es extremadamente sensible a la distorsión de fase, en donde es más importante tener en cuenta la

fase lineal es en el procesamiento de imágenes [28] y por ultimo la principal razón de seleccionar un filtro FIR en lugar de un IIR es que estos últimos no todos los diseños son realizables debido a su naturaleza recursiva [2], los FIR siempre son estables o realizables.

Así, verificando las propiedades que se cumplen para estos filtros realizables, es decir, que tienen propiedades de linealidad, invarianza en el tiempo, la causalidad y la estabilidad, es posible decir que si en el dominio del tiempo la respuesta al impulso relaciona la entrada y la salida de un sistema LIT, por lo tanto en el dominio digital, todo esto se resume a la suma de convolución entre la entrada$x\left[n\right]$ y la respuesta al impulso$h\left[n\right]$, definida como sigue en 5-8.

$$y\left[n\right] = x\left[n\right] * h\left[n\right] = \sum_{k=-\infty}^{\infty} h\left[k\right] x\left[n-k\right] \tag{5-8}$$

Es posible obtener un bloque que generaliza a un sistema digital relacionando su entrada y salida, se debe recordar que n será para cualquier número entero, llamado muestra, mientras que t es para tiempo. Si se cambia ahora de dominio de la frecuencia que es una particularización

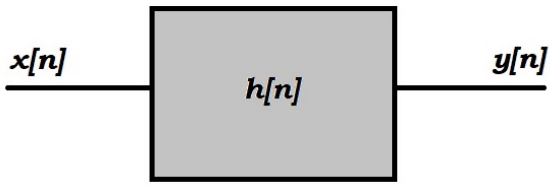


Figura 5-6: Relación generalizada de entrada y salida en un sistema digital.

de la Z se tiene que la relación de la entrada y salida del SLIT es la transformada Z de la respuesta al impulso:

$$y[n] = x[n] * h[n] \xrightarrow{TZ} Y(z) = X(Z)H(Z)$$

$$H(Z) = \frac{Y(z)}{X(z)} = \sum_{n=0}^{\infty} h[n] z^{-n} \tag{5-9}$$

5.5. Creación de bloque para filtrado de audio digital.

Este IP Core FIR Compiler 7.2 [39] provee una interfaz común que genera una alta parametrización, es decir, eficiencia en el área ocupada al implementarse en el FPGA que implica filtros FIR de alto desempeño, sus características son las siguientes:

- Interfaz compatible con AXI-Stream
- Filtros de Respuesta de impulso finito (FIR) con alto rendimiento, Decimador Polifásico, Interpolador Polifásico, media Banda, Decimador de Media Banda e interpolador de media banda, Transformada de Hilbert e implementaciones de filtros interpolados.
- Soporta hasta 256 conjuntos de coeficientes que contengan de 2 a 2048 coeficientes por conjunto
- Datos de entrada de hasta 49 bits de precisión
- Soporte de hasta 1024 canales de datos intercalados
- Soporte para secuencias de datos intercalados de forma avanzada.
- Soporte para múltiples canales de datos paralelos con lógica de control compartida.
- Factores de interpolación y decimación de 64 típicamente hasta un máximo de 1024 para filtros de canal simple
- Soporte para frecuencia de muestreo mayor que la frecuencia de reloj
- Capacidad de recarga de coeficientes en línea
- Selección de redondeo en la salida
- Eficiencia en las estructuras multi-columnas para todas las implementaciones u optimizaciones en todos los filtros.
- Soporta los dispositivos de la familia UltraScale+™ Families, UltraScale+™ Architecture, Zynq-7000 All Programmable SoC, 7 Series.

A continuación se llevará acabo la implementación en Vivado, primero se observará que el tipo más simple de filtro FIR que implementa este core es“Single Rate”, procesa la suma convolutiva

definida por 5-10, donde N es el número de coeficientes del filtro.

$$y[k] = \sum_{n=0}^{N-1} a(n)x(k-n) \qquad k = 0, 1, \ldots \tag{5-10}$$

El núcleo puede implementar dos arquitecturas para crear el flujo de calculo con los datos y los coeficientes de entrada, tiene la arquitectura “Systolic MultiplyAcumulate” y “Transpose MultiplyAcumulate”, si se observan las características de estas arquitecturas se tiene una limitación de canales entrelazados en una misma linea de datos, esta característica definirá la decisión de selección de la primera arquitectura, ya que se cuenta desde el principio con dos canales multiplexados, canal derecho e izquierdo de audio. Seguido a esto se tiene la forma de implementación de la arquitectura “Systolic MultiplyAcumulate”, primero se debe ver como es un “Multiply-Accumulate (MAC)”, cuya estructura se verifica en la figura 5-7, ahora es posible apreciar en la imagen 5-8 la representación y la forma de aplicación con los registros unidos a la estructura del MAC simple, se debe recordar que los pequeños bloques con la notación $a(N)$ corresponde a los coeficientes donde N es el último de ellos y $z^{-}1$ es la señal con un atraso en este caso un dato de 24 bits, $y(n)$ la salida y $x(n)$ la entrada. Se implementarán los filtros en una

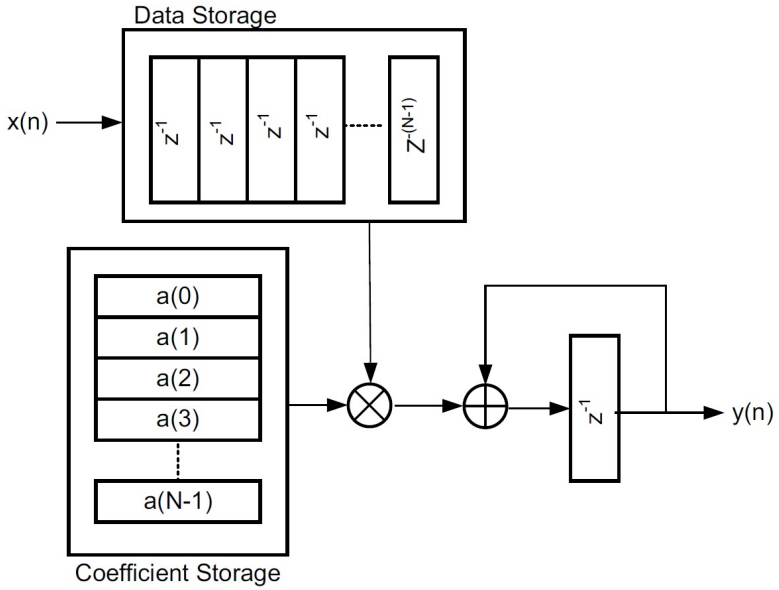
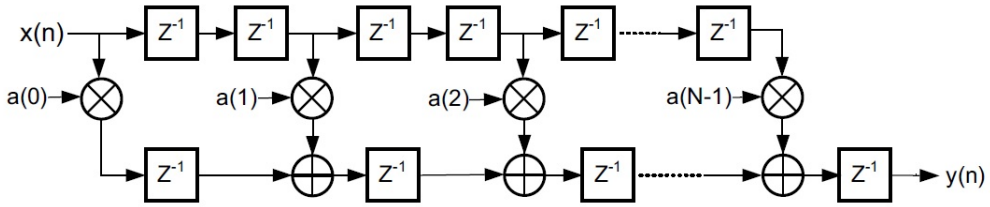
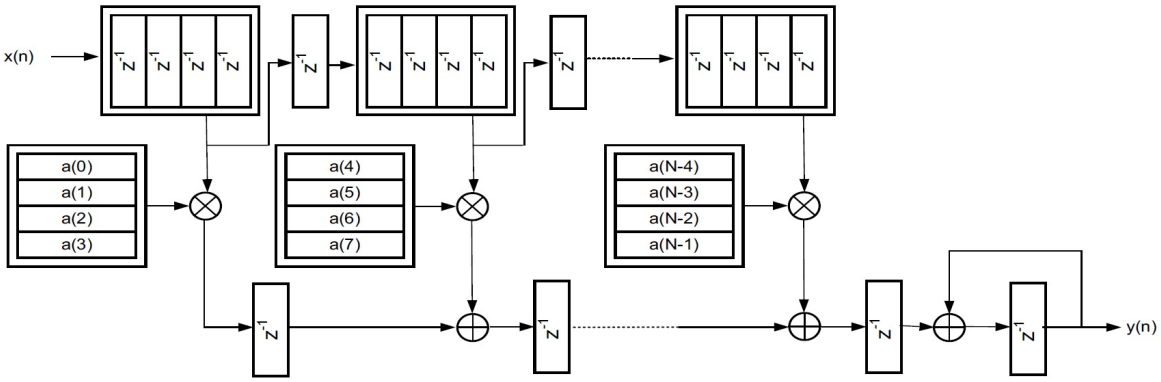


Figura 5-7: Fuente: [39] Forma simple de una unidad multiplicativa-acumulativa.



(a) Pipeline Direct-Form, fuente: [39].



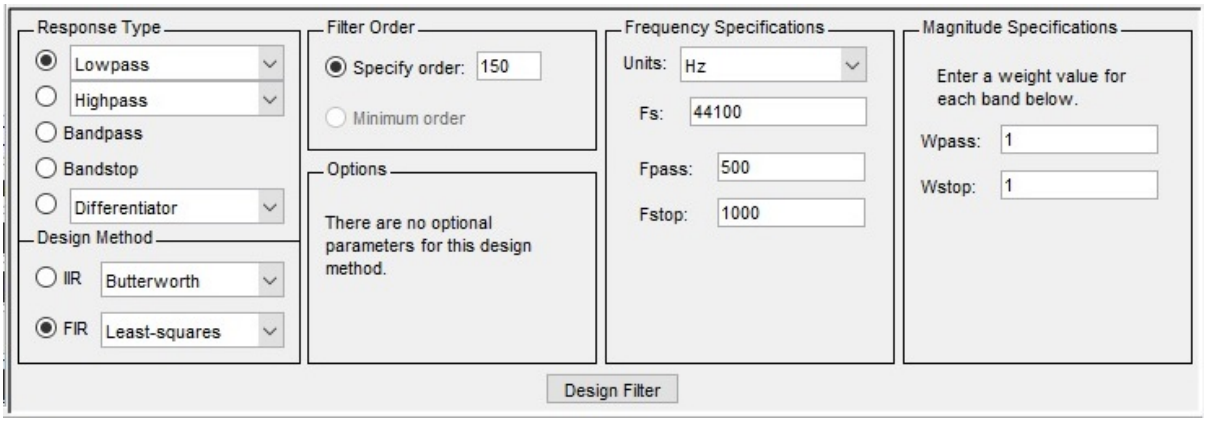
(b) Systolic Multi-MAC implementation, fuente: [39].

Figura 5-8: Representación de la arquitectura “Systolic MultiplyAccumulate” “pipelined DirectForm filter” y “Systolic Multi MAC Implementation”.

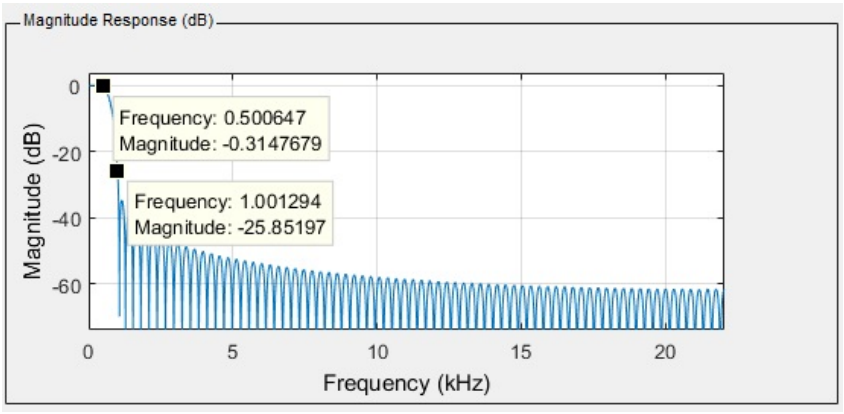
aplicación de Matlab llamada Filter Designer donde se realizarán los cálculos de los coeficientes dado un tipo de filtro pasa bajas con el método de implementación Parks-McClellan o como se llama en FDATool Least-squares, este es el filtro más utilizado en la practica ya que logra cumplir con las especificaciones en un orden más pequeño respecto a otros métodos [28]. Puede observarse en la figura 5-9(a) los parámetros de diseño seleccionados, estableciendo el orden del filtro en 150, una frecuencia de muestreo de 44100 muestras por segundo, una frecuencia de paso de $500Hz$ y una frecuencia de rechazo de $1000Hz$, adicionalmente se tienen dos valores que optimizan la atenuación en forma de pesos a las bandas de paso y rechazo, si no se utilizan se toman por defecto en 1, en el Filtro pasa bajas se dejará con 1, en la figura 5-9(b) se muestra la respuesta del filtro, también se vé la atenuación para la frecuencia de paso que es $-0.31dB$ aproximadamente y para la frecuencia de rechazo es $-25.85dB$, es posible observar que cumple con la máxima atenuación en $15KHz$ a $-60dB$.

Esta herramienta calcula los coeficientes que se necesitaran para adicionarlos en un archivo con extensión *.coe* (Xilinx BRAM initialization file) en el módulo $Fir\_compiler\ v7.2$, en este caso se tendrá el mismo número de coeficientes que grado del filtro más uno 151, se necesita agregar estos coeficientes en un archivo de texto con formato ASCII para almacenarlos con dos especificaciones, primero la base en que se tendrá los coeficientes y segundo los coeficientes mismos:

$$Radix = 10;$$
$$Coefdata =$$
$$a(1),$$
$$a(2),$$
$$a(3),$$
$$...$$
$$a(N-1);$$



(a) Parametros de Diseño, obtenido de Filter Tool Designer.



(b) Respuesta del Filtro, obtenido de Filter Tool Designer.

Figura 5-9: Obtención de los coeficientes con Filter Designer.

En este caso se observa que existe simetría en los valores, existe un coeficiente “central” y a partir de allí van repitiéndose estos hasta el inicio y fin de la lista, es decir, una forma similar a:

$$Coefdata = 2, 8, 9, 0, 9, 8, 2;$$

Esta característica se debe tener en cuenta pues el módulo necesita verificar la propiedad e implementar la estructura más adecuada, esto se puede verificar desde FDATool en el apartado de la respuesta gráfica al impulso, se observa simetría en la gráfica 5-10.

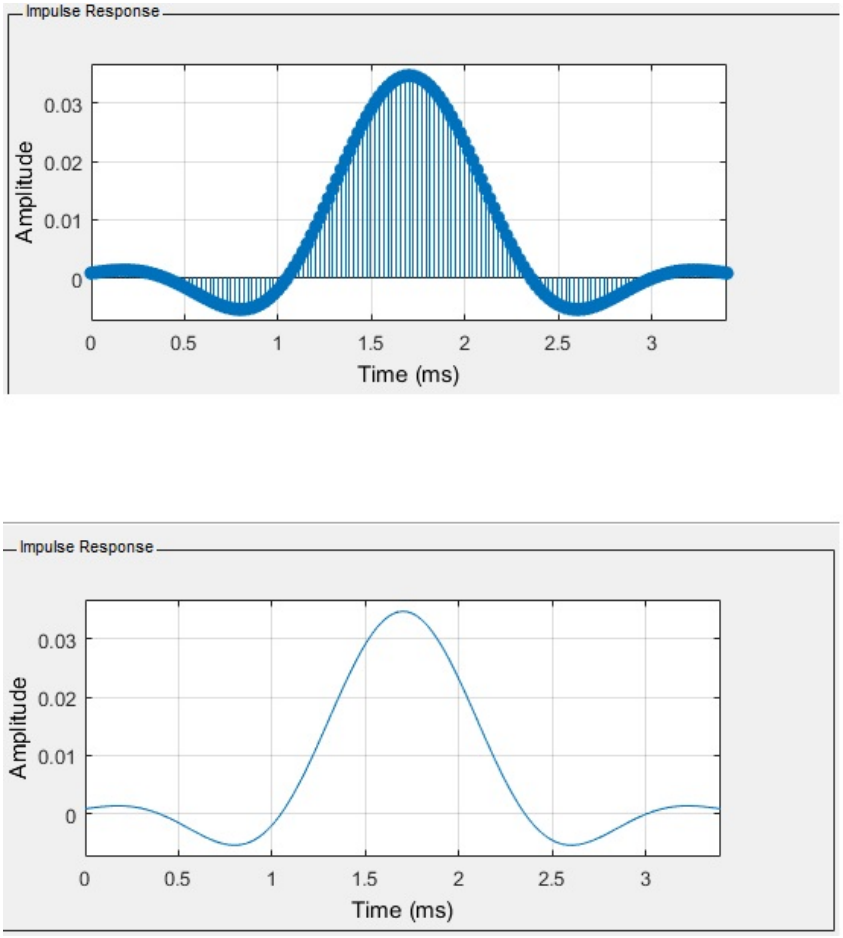


Figura 5-10: Respuesta al impulso, se observa simetría negativa y positiva.

Es posible agregar los coeficientes tanto en base decimal, hexadecimal y binaria, además el módulo también reconoce si son datos con signo, cuando se tienen coeficientes en base 10 existe un proceso de cuantización de los mismos, ya que en base 2 y base 16 se consideran ya

cuantificados, también si son números enteros pueden usarse diferentes métodos de cuantización y una última opción que es usando números decimales. Se cuenta con tres opciones de cuatización “solo cuantizar” y cuantizar para “maximizar el rango dinámico”.
Para la opción “solo cuantizar”, funciona como verificación de la respuesta al situar los parámetros y obtener mayor precisión o un buen rango de representación de la respuesta al filtro, también contiene un redondeo para la opción donde no tenga bits necesarios tanto en la parte fraccional o entera de los coeficientes, pues se puede dejar el mayor número de bits de la entrada que manejamos al valor fraccionario o entero del mismo, así se tiene la oportunidad de seleccionar la mejor respuesta a criterio propio o dejar simplemente que trabaje un algoritmo interno que hace esta cuantización, es decir, esto permite explorar la compensación entre el rendimiento del filtro y los recursos variando el parámetro de ancho del coeficiente[39].

Elegida la opción de “cuantizar para Maximizar el rango dinámico” lo que se modifica es el valor de los coeficientes por un factor que se elige dependiendo la cantidad máxima representable por los bits disponibles y el máximo valor representado en los coeficientes, este factor se muestra en el gráfico de respuesta y es medido en dB, este valor máximo en el caso de estos coeficientes con simetría suele ser el central, después de ello pasa al proceso de cuantificación, este proceso introduce una ganancia que debe tenerse en cuenta en el diseño, pero si se desea tener esta opción aplicada entonces la ganancia debe ser compensada por un circuito que regule la misma, este circuito puede ser un control automático de ganancia (AGC Automatic Gain Control).

Se verificará este procedimiento con respecto de lo obtenido por el módulo, si se tiene un ancho de coeficiente entero con 24 bits, el módulo calcula que 27 bits representaran la parte fraccional del coeficiente esto es porque se hace un análisis de los valores representativos en binario, entonces se verifican los valores redundantes, se evitan almacenarlos, posteriormente se hace una extensión de signo para reutilizar esos bits con información redundante y agregar precisión en el filtro con los mismos en la parte fraccional, este proceso de extensión de signo se puede verificar en [39], entonces se tiene lo siguiente:
Ancho de coeficiente de 24 bits entero que proporciona un valor positivo máximo representable de 8388607 y un valor negativo de -8388608, los coeficientes especificados oscilan entre

$+0.0346755991245149$ y -0.00531762925232388, entonces el número positivo máximo representable considerando los números fraccionarios es:

$$8388607/(2^{27}) = 0.06249999255$$

Para obtener el factor de escala se determina dividiendo el valor máximo representable con los bits del coeficiente por el valor de coeficiente máximo, así tenemos

$$0.06249999255/0.0346755991245149 = 1.802419976$$

por lo tanto

$$20\log_{10}(1.802419976) = 5.1171198\ dB.$$

Es posible ver las curvas definidas en dos colores diferentes dentro de la imagen 5-11, una ideal y la cuantizada, dentro de la última se puede ver el factor 5.117120 dB que corresponde a la ganancia introducida que se debe tener en cuenta y la cual se calculó con respecto de los parámetros de entrada. Se tiene la posibilidad de observar la respuesta de fase en la figura 5-12(a) y el diagrama de polos y ceros del filtro en la figura 5-12(b), en el primero se observa el rango de frecuencias de paso hasta aproximadamente los 1000 Hz donde la fase se comporta linealmente, una recta con pendiente constante hasta la frecuencia de rechazo, se verifica que tanto la atenuación en magnitud y la variación en fase se separan en las crestas y se atraen en los valles, hasta separarse casi por completo en las frecuencias más altas haciendo que estas, aunque casi eliminadas no correspondan en nada a la señal original por el desfase generado. Para el plano Z puede observarse que se tiene 1 polo justo en el centro del plano, de este plano es posible obtener la función de transferencia que describe al sistema, cuando son filtros de orden pequeño, pero en este caso resulta casi imposible, esta ecuación tendría el polo en 0, después, si se aprecia con mucho detenimiento, se podría ver que existen 146 ceros en el límite de la circunferencia con radio 1 que indica y limita la estabilidad de cualquier sistema descrito. A continuación se muestran otros 4 ceros en la parte izquierda dando un total de 150 ceros y

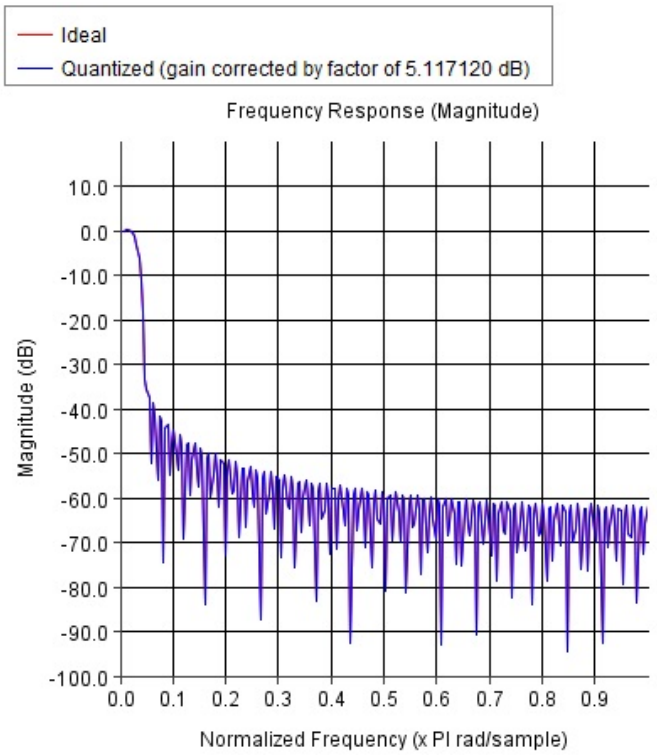
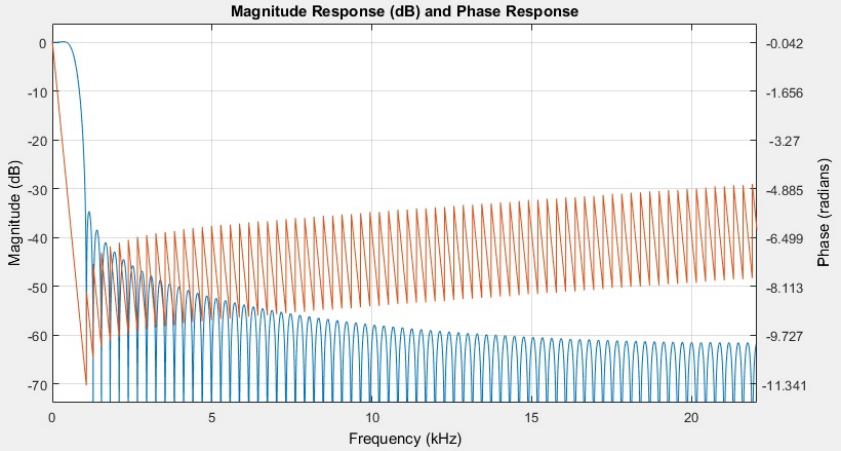


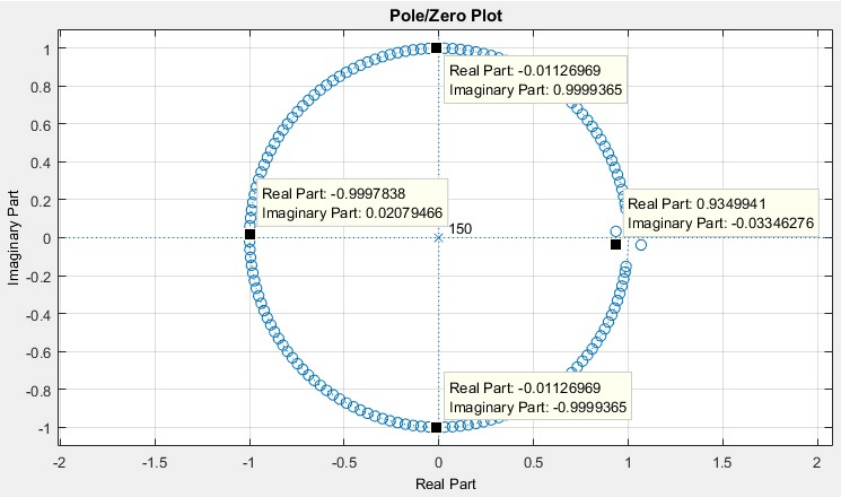
Figura 5-11: Respuesta en frecuencia ideal y obtenida con factor de cuantización.

1 polo, de los cuales sus raíces son los coeficientes mismos, en un proceso donde varia la fase podrá dar resultado al generar la respuesta impulsional el filtrado de la señal. Después de la parte de los coeficientes del filtro, sigue la configuración de los parámetros en el módulo FIR Compiler v7.2, hay una interfaz maestra y esclava AXI-Stream, esta se configura al tener las siguientes opciones habilitadas como sigue:

1. Filter Coefficients
   - Select Source : COE File
   - Coefficient Vector : Ruta de archivo .COE
   - Number of Coefficients Sets : 1
2. Filter Specification
   - Filter Type : Single Rate
3. Interleaved Channel Specification



(a) Respuesta en Fase y Magnitud, obtenido de Filter Tool Designer.



(b) Polos y Ceros en el Plano Z, obtenido de Filter Tool Designer.

Figura 5-12: Verificación de fase lineal y plano Z.

- Channel Sequence : Basic
- Number of Channels : 2

4. Hardware Oversampling Specification

    - Select Format : Frecuency Specification
    - Sample Period(Clock Cycles) : 1
    - Input Sampling Frecuency (MHz) : 0.0441
    - Clock Frecuency (MHz) : 22.5792

5. Coefficient Options

    - Coefficient Type : Signed
    - Quantization : Maximize Dynamic Range
    - Coefficient Width : 24
    - Best Precision Fraction Length : Yes
    - Coefficient Fractional Bits : 27
    - Coefficient Structure : Symmetric

6. Data Path Options

    - Input Data Type : Signed
    - Input Data Width : 24
    - Input Data Fractional Bits : 0
    - Output Rounding Mode : Truncate LSBs
    - Output Width : 32

7. Detailed Implementation

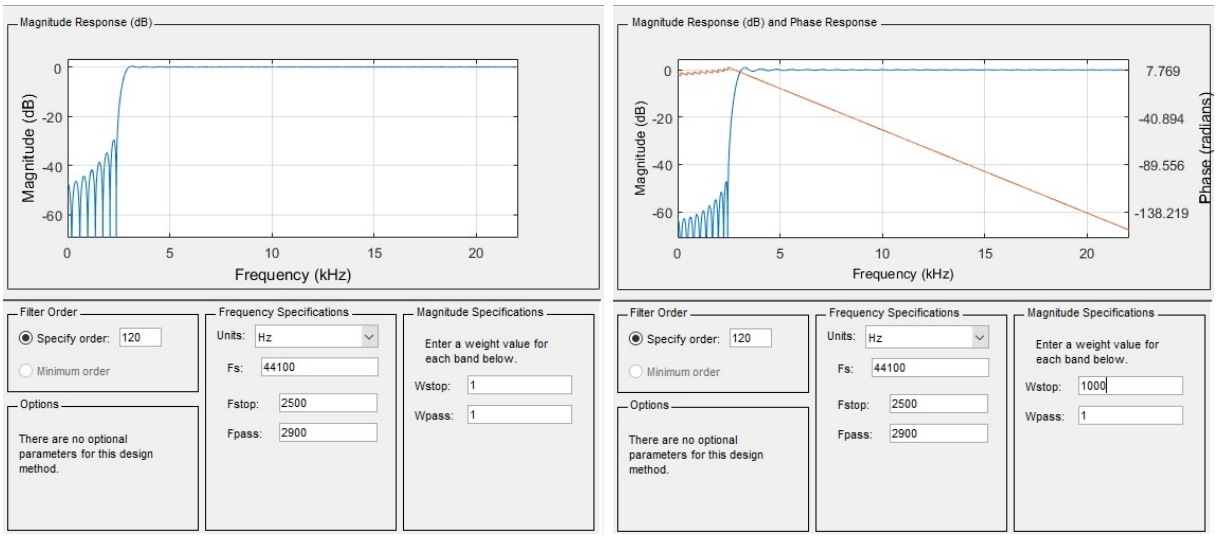
    - Filter Achitecture : Systolic Multiply Accumulate

8. Data Channel Options

    - TLAST : Packet Framing
    - Output TREADY: yes

Se pueden ver los recursos ocupados y todas las características de salida y entrada al módulo en una parte llamada “Summary”, donde están listadas las opciones elegidas anteriormente así como los detalles de implementación.

Ahora se verán las características del filtro Parks McClellan pasa altas, ya se cuenta con un ejemplo práctico, es decir, el filtro pasa bajas anterior , así que es posible mostrar los detalles más rápidamente, primero se elige el orden del filtro en 120, una frecuencia de atenuación de $2500Hz$ y frecuencia de paso de $2900Hz$, se eligieron estas frecuencias ya que tienen características importantes que representan al audio, porque si se supera el límite de los $4KHz$ se obtienen pocas componentes espectrales para escuchar, con respecto de los valores $Wpass$ y $Wstop$ que como ya se mencionaron representan los pesos para optimizar la magnitud de las bandas, se usarán en la banda de rechazo un valor de 1000 y en la banda de paso 1, haciendo esto se tiene mejor atenuación en la banda mencionada, se obtiene una diferencia de aproximadamente 20 dB como se observa en la figura 5-13.



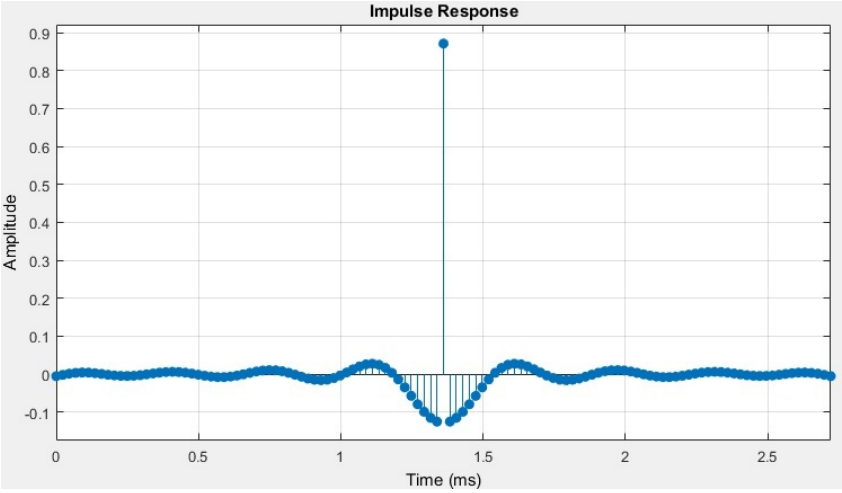
(a) atenuación de 30dB aproximadamente, obtenido de Filter Tool Designer.

(b) atenuación de 50dB aproximadamente, obtenido de Filter Tool Designer.

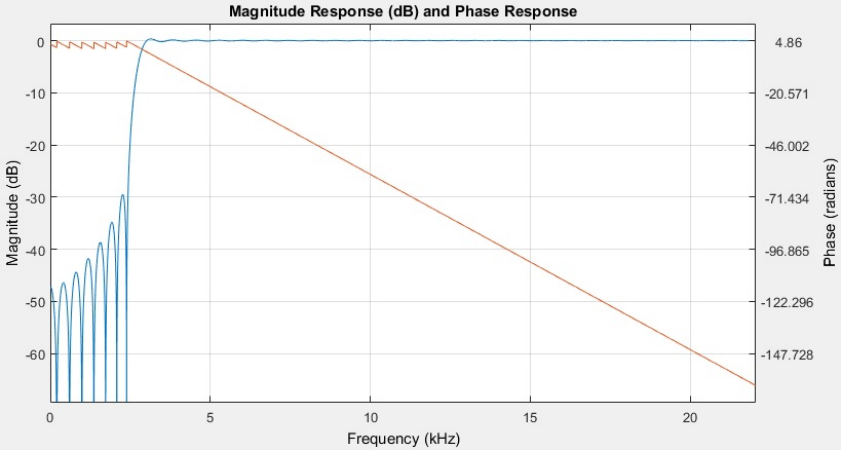
Figura 5-13: Optmización con peso en banda atenuada(Wstop) y de paso(Wpass).

Se verificará la simetría en los coeficientes, en la respuesta al impulso de la figura 5-14(a) es posible observar que cuenta con un coeficiente central definido por la simetría existente, esta

simetría es conformada por coeficientes con valores negativos y positivos, también se puede verificar la fase del filtro5-14(b) donde le sigue la respuesta en magnitud antes vista en la figura 5-13(b).



(a) Coeficientes Simétricos, obtenido de Filter Tool Designer.



(b) Fase lineal, obtenido de Filter Tool Designer.

Figura 5-14: Verificación de fase lineal y simetría en los coeficientes del filtro pasa altas.

La configuración de parámetros en el módulo Fir_compiler es exactamente igual a la del filtro pasa bajas, por lo que pueden tomarse esos valores como referencia.

Por último se tiene la función de Pasa Banda en el filtrado de señal, este filtro tiene la peculiaridad, como se vio anteriormente, que contiene dos bandas de atenuación y una banda

de paso central, se definirán unas frecuencias tales que se acerquen al ancho de banda de la voz humana que comprende estando en un formato de Banda estrecha o NB por sus siglas en ingles NarrowBand aproximadamente desde los 200 Hz a los 3.5 KHz [23], por lo que los parámetros serán un filtro pasa banda Parks McClallen de orden 150 con una frecuencia de rechazo 1 debajo de los 200Hz y de paso 1 en 600Hz, una frecuencia de paso 2 en 3000 Hz y una frecuencia de rechazo 2 superior a 3400Hz, como se observó en el filtro pasa altas se puede usar la optimización con los Wpass = 1 y en este caso Wstop1=1000 y Wstop2=1000, donde igualmente se obtuvo una atenuación adicional al agregarlos a los parámetros, se tiene entonces la gráfica en la imagen 5-15 donde se observa la respuesta en magnitud y fase.

Se verificará la simetría en los coeficientes, en la respuesta al impulso de la figura 5-16(a), es

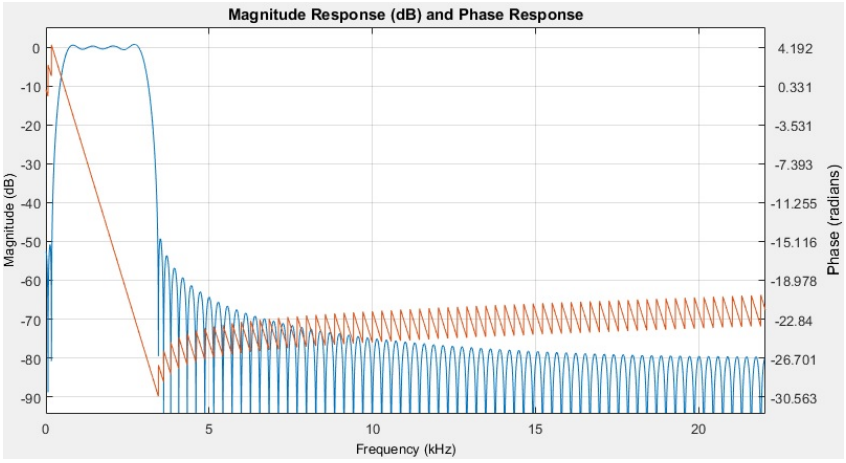


Figura 5-15: Respuesta en magnitud y fase del filtro pasa banda, obtenido de Filter Tool Designer.

posible observar que se cuenta con un coeficiente central definido por la simetría existente, esta la conforman coeficientes con valores negativos y positivos al igual que en el filtro pasa altas por lo que se dice que contiene simetría positiva y negativa.

La configuración de parámetros para los tres filtros en su módulo Fir_compiler respectivo, tiene exactamente los mismos rangos y opciones seleccionados, puesto que todos tienen coeficientes simétricos, tienen dos canales multiplexados en una sola linea de datos, etc., entonces se tienen tres opciones funcionales de filtrado, los tres cuentan con interfaz AXI-S por lo que tienen las mismas terminales de datos y control, esclavo (s_data, s_valid, s_ready y s_last)

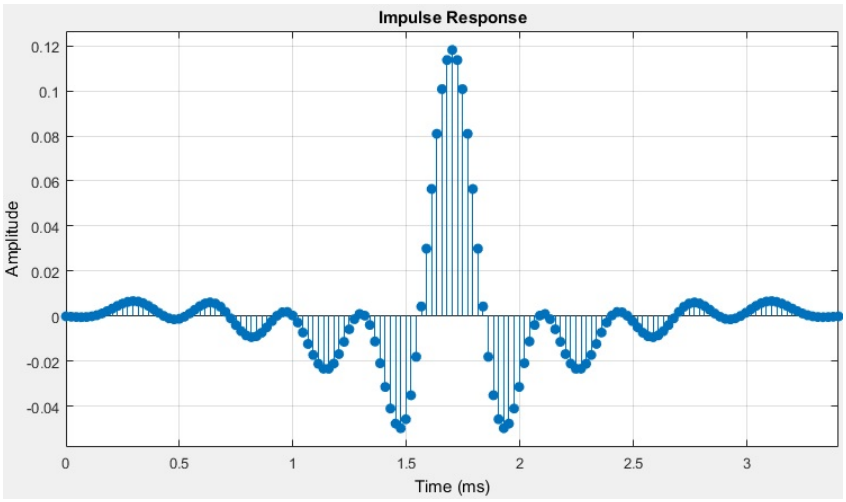


Figura 5-16: Respuesta al impulso, simetría en coeficientes, obtenido de Filter Tool Designer.

y maestro (m_data, m_valid, m_ready y m_last), entonces para seleccionar la función de filtrado correspondiente se cuenta con dos entidades electrónicas básicas, un demultiplexor 1x4 y un multiplexor 4x1 que ciclo a ciclo podrá modificar la conexión de cada filtro o hacer un bypass y dejar sin filtrado los datos que viajan en la interfaz, el bloque establecido es como se ve en la figura 5-17.

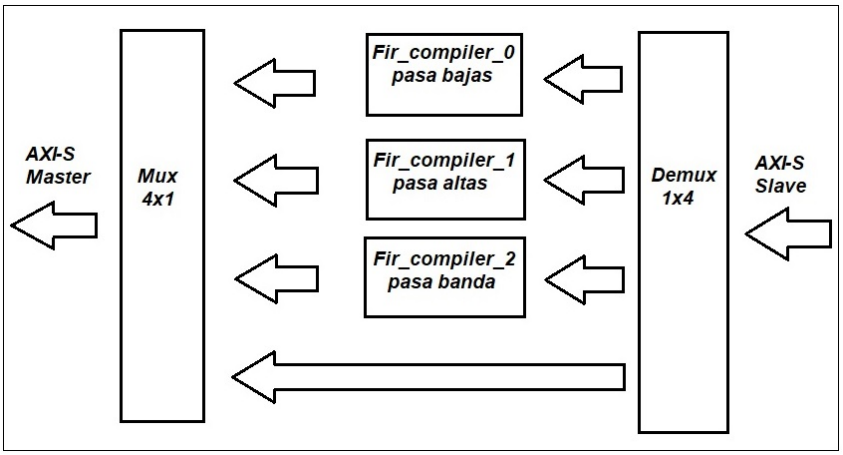


Figura 5-17: Diagrama de bloques del módulo de filtrado.

6

Procesador Digital de Señal.

Los procesadores DSP (Digital Signal Processor) tienen instrucciones y una arquitectura desarrollada específicamente para ejecutar aplicaciones de procesado de señales digitales como protocolos para voz para Internet (VoIP), comunicaciones inalámbricas (3G wireless), sistemas de radar y satélite, procesamiento de imágenes, sistemas multimedia, etc., este tipo de aplicaciones suelen trabajar con señales en tiempo real por lo que el DSP debe procesar las señales de entrada a la misma velocidad con que las obtiene, en el proyecto funciona tal como se describe, ya que la señal de audio de entrada se modifica en tiempo real, los DSP en los módulos estarán en la etapa de implementación del bloque para filtrado, en el control de volumen para la multiplicación y en algunas sumas de señales, cabe resaltar que el módulo Fir_compiler®utiliza MAC (multiply acumulator) o multiplicador acumulador, esta funcionalidad es una de las más comunes en los DSP. Ahora, se debe tener en cuenta que los DSP que no se encuentran embebidos tienen una limitante como tener memoria fija, número fijo de bloques aceleradores de hardware y ancho de datos fijo[10], por lo que los circuitos FPGA o CPLD suministran la solución de reconfigurabilidad y eficiencia necesaria para las aplicaciones DSP, los FPGA ofrecen una oportunidad para acelerar una aplicación DSP hasta 1000 veces más que un microprocesador DSP tradicional [27] aunque en este caso solo se ocuparan de forma estándar. Los bloques DSP embebidos en los circuitos FPGA suministran las funciones tales como acumulación, adición, sustracción y sumatoria, funciones comunes en DSP, en la familia Artix-7®exclusivamente el

FPGA XC7A35T, contiene 90 DSP slices específicamente el DSP48E1 [33], el cual tiene una velocidad de procesamiento límite de 550MHz. El diagrama funcional de este procesador se muestra en la figura 6-3.

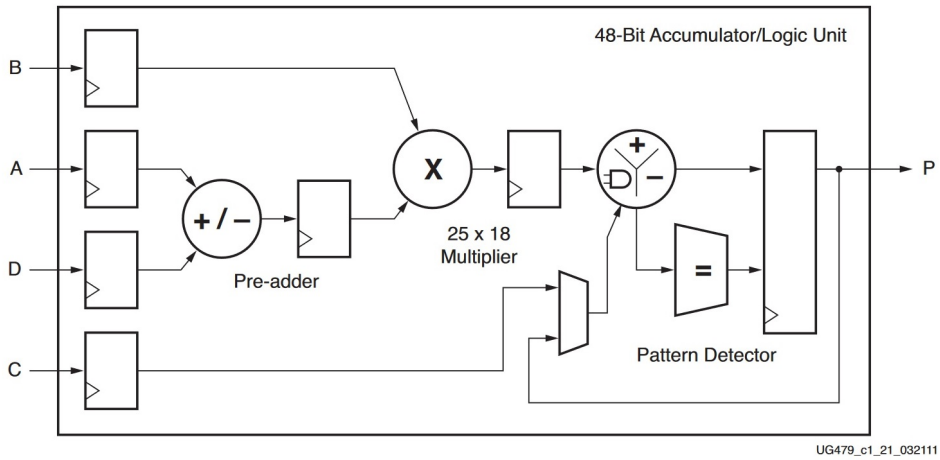


Figura 6-1: Bloques funcionales en DSP48E1 de FPGA Artix 7, fuente: [33]

Se sabe que los FPGA manejan el principio de elaborar circuitos complejos a partir de componentes básicos, a pesar que con el tiempo se han complementado con bloques de circuitos que realizan tareas muy especificas (Bram, Multiplicadores, IOBs, CLBs, DSPs) esto es porque un ASIC es más pequeño y rápido que un diseño con CLBs y es por ello que se tienen estos DSP embebidos con el FPGA.

6.1. Aplicaciónes aritméticas para los DSP´s

En concreto, un procesador DSP típico contiene las siguientes unidades funcionales y características:

- Multiplicador acumulador (MAC)
- Unidad Aritmetico Lógica: Suma, resta, and, or, not.
- Desplazadores: Para escalar los datos antes o después de una operación sobre ellos.
- Etapas: Suele aparecer segmentado para disminuir el tiempo de ciclo y aprovechar mejor los recursos del sistema y recibir una nueva instrucción ciclo a ciclo (IPC cercano a 1,

Instrucciones Por Ciclo)

- SIMD: Suelen tener un modo Single Instruction Multiple Data
- Arquitectura Harvard: Al contrario de la arquitectura Von Neumann que almacena datos e instrucciones en la misma memoria la arquitectura Harvard tiene una memoria de instrucciones y una (o dos) de datos 6-2. De esta manera se pueden tener los datos y las instrucciones a ejecutar en el mismo ciclo.

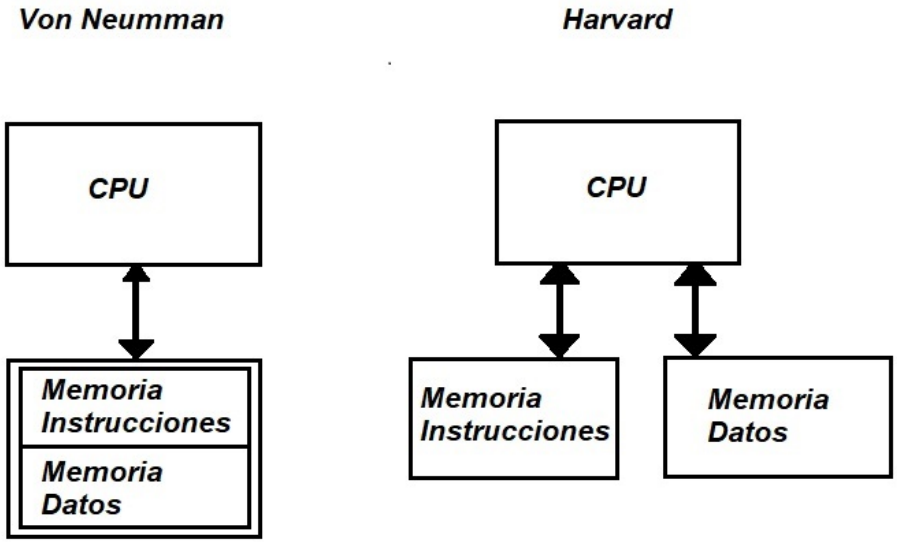


Figura 6-2: Arquitecturas Von Neumann y Harvard.

El DSP48E1 no contiene una memoria especifica para instrucciones [34] y otra para datos como sería la arquitectura Harvard, pero si cumple con entregar el resultado de un proceso en solo un ciclo, tampoco es segmentado ya que no puede enviarse ciclo a ciclo una nueva operación sin que termine la anterior.

Así pues se tienen dos opciones para definir el uso o inferir un DSP, primero es usar un DSP IPs de Xilinx® , como en el caso del FIR_compiler o con el código de descripción VHDL/Verilog, en el uso de Xilinix® IP core directamente se establece la función especifica de poder cambiar con que recursos puede sintetizarse el módulo, ya sea con DSP o LUT, para facilitar el trabajo se verá un poco sobre inferir DSP al interprete de Vivado, de ello dependerá la descripción o especificaciones dadas con VHDL, aunque también existen macros que pueden establecer

etapas extras en registros, modificar el proceso interno del DSP, etc., además, con éstos macros es posible establecer exactamente componentes específicos como:

- Carry logic (MUXY, XORCY, MULT_AND).
- RAM(block o distribuida).
- Registros de desplazamiento LUTs (SRL16, SRL32).
- Clock Buffers(IBUFG, BUFG, BUFGP, BUFR).
- Multiplexores (MUXF5, MUXF6, MUXF7, MUXF8).
- Funciones artimeticas(DSP48, MULT18x18).

No se usaran estos macros pero es una opción disponible, por si se requiere un uso muy cuidadoso y limitado de los recursos en el FPGA. En el uso del interprete de síntesis en VHDL existen plantillas muy especificas para inferir los DSP desde el código, así que se verá a continuación una plantilla utilizada para generar un multiplicador 25x18 sin etapas de registros:

```
1       entity mult is
2             generic (
3                   tamOp1: integer:=25;
4                   tamOp2: integer:=18;
5                   tamResul: integer:=43
6             );
7             port (
8                   a : in std_logic_vector (tamOp1-1 downto 0);
9                   b : in std_logic_vector (tamOp2-1 downto 0);
10                  clock : in std_logic;
11                  reset : in std_logic;
12                  p : out std_logic_vector (tamResul-1 downto 0)
13            );
14      end entity;
15      architecture mult_arch of mult is
16      begin
17      process (clock) is
18      begin
19             --Sin Registros en A, B, M ni P
20             if clock'event and clock = '1' then
```

```
21                    if reset = '1' then
22                          p <= (others => '0');
23                    else
24                          p <= a*b;
25                    end if;
26              end if;
27        end process;
28        end architecture;
```

En el reporte de síntesis6-3 se muestra que se utilizaron 2 DSP para el diseño, por lo que la estructura funciona, existen muchas otras estructuras bien definidas y pueden ser consultadas en [10], este tema es muy amplio pero necesita ser consultado para conocer la función principal de usar DSP y evitar consumir muchos ciclos de reloj al realizar operaciones aritméticas además de concluir de buena manera el proceso de manejo de datos en tiempo real.

```
7. Primitives
-------------

+----------+------+----------------------+
| Ref Name | Used | Functional Category |
+----------+------+----------------------+
| IBUF     |   45 |                   IO |
| OBUF     |   43 |                   IO |
| FDRE     |   17 |        Flop & Latch |
| DSP48E1  |    2 |    Block Arithmetic |
| BUFG     |    1 |                Clock |
+----------+------+----------------------+
```

Figura 6-3: Reporte de síntesis.

7

Implementación de diagrama de bloques con DSP´s.

7.1. Modelos de Diagramas de bloques para efectos de audio.

En esta sección se debe mostrar la representación de modelos en bloques, pues es una característica básica para poder representar y entender como se construyen estos diagramas, surgen de la necesidad de representar y comprender las operaciones de control y función en un sistema físico, se debe recordar que estos bloques pueden representar sistemas mecánicos, eléctricos, electrónicos, etc., pueden ser unidades pequeñas o todo un proceso completo integrado, internamente se verifica su funcionamiento con una ecuación matemática que representa univocamente su funcionamiento, esta ecuación es la función de transferencia, que es definida como el resultado obtenido de la relación entre la salida y la entrada a dicho bloque, esta función de transferencia es de esta forma 7-1 donde $Y(z)$ es la salida y $X(z)$ la entrada.

$$H(Z) = \frac{Y(z)}{X(z)} \tag{7-1}$$

Para estos sistemas se tienen las siguientes representaciones esquemáticas 7-1, con ellas se pueden construir diagramas extensos sin problema alguno, cabe mencionar que el proceso del

sumador bien puede ser cambiado por un multiplicador, existe la llamada álgebra de bloques que tiene equivalencias si se necesita reducir un diagrama muy complejo, pero dado que esta álgebra y las sustituciones están enfocadas a sistemas en su mayoría con re-alimentación, no será necesario para el proyecto puesto que las salidas solo dependen de la entrada [42](pag. 45), así que no es necesaria esta teoría para este desarrollo en particular, pero si se quiere profundizar puede ver el recurso aquí [31].

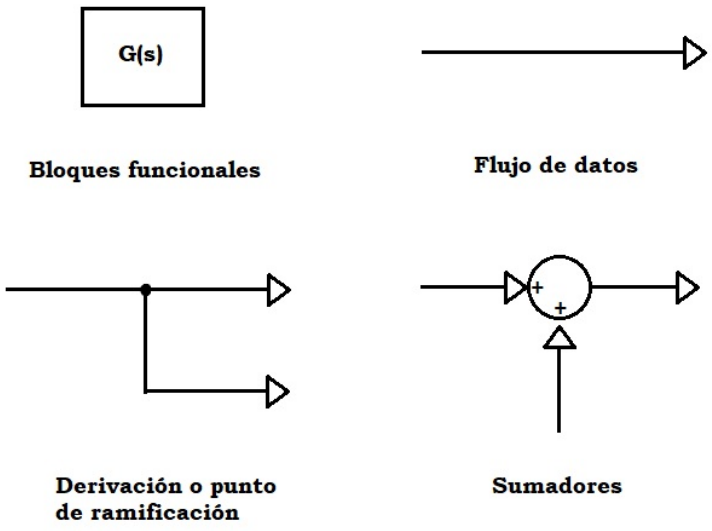


Figura 7-1: Elementos de diagramas de bloques.

Así pues, con la función de transferencia y el diagrama de bloques de cada efecto de audio, se puede saber analíticamente como funcionará en el sistema digital, se necesitará entonces crear los bloques funcionales de cada efecto, ya sean desplazamientos de bits, multiplicaciones, sumas, comparaciones, etc.

7.2. Efectos “retardo” y “volumen”.

Los efectos de audio digital, en ingles Digital Audio Effect (DAFX), son modificaciones que se hacen al audio en sus características esenciales, como amplitud, frecuencia y fase, dentro de sistemas digitales en este caso, aunque también existen los efectos de audio analógicos, estas

modificaciones básicamente se hacen en tres dominios de procesamiento, dominio dinámico, dominio en frecuencia y usando retardos, a continuación se muestra un DAFX en la figura 7-2 en una versión muy simple, el cual reduce la amplitud de audio a la mitad dado por el factor 0.5.

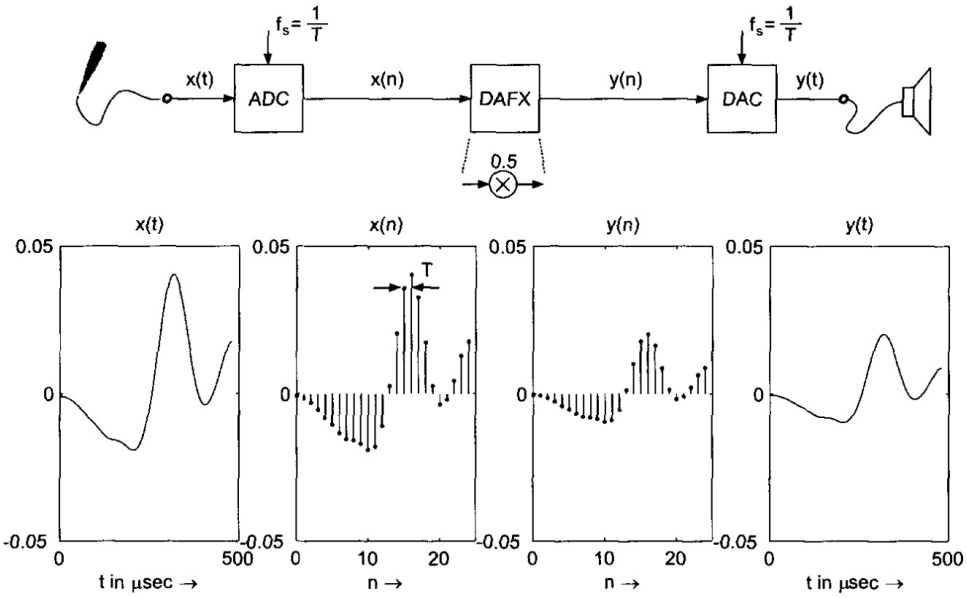


Figura 7-2: Muestreo y cuantización por ADC, DAFX y reconstrucción por DAC, fuente: [42].

Se iniciará con los efectos en el dominio dinámico, se implementará el controlador de volumen, que precisamente trabaja en el dominio dinámico, pues depende de la entrada en un instante t de tiempo que forma parte de la dinámica de la señal ya que en ese momento puede estar presente el mínimo o el máximo valor representable en el sistema digital, así con esta función se puede englobar este dominio, adicional a estos efectos, se debe recordar que deberán acoplarse a la interfaz AXI-S, así que éste será el valor agregado para el trabajo presente. Se comienza con la descripción de la entrada y salida como una función de transferencia que se observa en 7-2, se tiene para la entrada un volumen que se representará con V esta señal será atenuada por una constante k que irá de 0 a 1, este efecto en bloques es el mismo que se tiene

en la figura 7-2.

$$\begin{aligned} x(n) &= V \\ y(n) &= k * V \\ G(n) &= \frac{V * k}{V} \\ G(n) &= k \end{aligned} \tag{7-2}$$

Para la parte del proceso del dominio en frecuencia se utilizaron los Filtros FIR del capitulo 5, así que solo se mencionan para ubicar el dominio ya cubierto, al estar eliminando o atenuando espectros de frecuencias se genera una modificación en el audio, comúnmente existen estos filtros en ecualizadores donde se acoplan una multiplicidad de estos para encontrar una respuesta en el audio de tal forma que solo resalten las frecuencias deseadas.

El procesamiento usando retardos consiste en almacenar la señal y variar el tiempo donde vuelve a ser enviada a la salida, ya sea que se opere esta señal retrasada con una constante o que solo sea operada con la señal original, estos cambios son fácilmente audibles con respecto del tiempo en que dure este retraso, regularmente desde un mínimo de 0.1ms [32].

El efecto llamado “delay”, este DAFX es interesante pues dependiendo la duración del retraso tiene diferente nombre el efecto audible producido[42], se verá la ecuación que describe al “delay” 7-3, el retardo *del* comprende entre 10 ms y 374ms en el proyecto.

$$y(n) = x(n) + x(n + del) \tag{7-3}$$

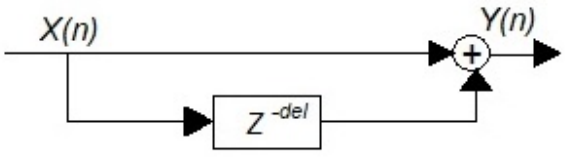


Figura 7-3: Diagrama de bloques efecto “delay”, fuente: [32].

Entonces se tienen los siguientes efectos dependiendo el valor del retraso, la mayoría son

consideraciones en base a criterios de los músicos, ya que ellos definen la utilidad de estos tiempos en base a su producción instrumental:

1. (0.1 a 0.7 ms): PANNING. El "delay" se usa para simular el desplazamiento de una fuente de sonido, cuando este se encuentra en una fuente estéreo lo que produce es una variación en el tiempo de un canal respecto del otro, este retardo es 0ms si el sonido se distingue audiblemente en el centro
2. (0.8 a 40 ms): DEPTH. Al sumar la señal original con la señal retardada se genera el efecto de profundidad en el audio
3. (40 a 50 ms): DOUBLING ECHO. Este efecto simula un coro, ya que consigue lograr un efecto estereo con una pista grabada en monoaural
4. (50 a 130 ms): SLAPBACK. Un efecto típico que normalmente era generada en música en los años 50, provenientes de guitarras principalmente.
5. (130 a 600 ms): EDGE. Se crea un sonido continuo ya que se sincronizan en tiempo las notas musicales, esta medida generalmente se establece en una corchea con puntillo.
6. (3000 ms o más): LOOP. Se generan capas de sonidos totalmente independientes.

7.3. Creación de bloque con los efectos para procesador de audio

Se iniciará con el bloque del controlador de volumen, este efecto reduce la amplitud de la señal digital, la reducción esta hecha por la multiplicación del valor de canal derecho e izquierdo por un factor que va de 0.0 a 1.0, entonces se debe saber como obtener los valores para una muestra de la entrada, almacenarla en un registro auxiliar que será operado con este factor y agregar el resultado en otro registro auxiliar para finalmente entregarlo a la salida del bloque. Para modificar el valor de este factor se tiene en la tarjeta de desarrollo varios switch que funcionaran como 0 o 1 en la entrada y está definirá en tiempo real el valor del factor, la representación del valor de este factor tendrá como base un número hexadecimal, para ello solo se necesitan 4 bits , estos bits tienen valor posicional por lo que colocar uno de ellos en 1 o 0

agregará mayor o menor volumen al audio.

Como el largo de palabra de audio digital es 24 bits, se dividira a esta en 6 valores hexadecimales, después para poder colocar el valor de los switch en un ancho del multiplicador, debe hacerse mediante una asignación del valor de los switch a un vector de 28 bits y dividir este con un entero que en hexadecimal será 15, solo se necesitaran usar funciones de conversión también llamadas casting [26] para poder elaborar la división como un número entero y después devolver un vector de bits para ser almacenado nuevamente en la salida de datos, se debe tener cuidado con el signo de esa magnitud, entonces los valores para el canal derecho e izquierdo, después de que sean operados por el factor, necesitarán tener una extensión de signo para poder eliminar problemas de desbordamiento en la multiplicación [14]. Lo más importante de este procedimiento es tener en cuenta que al multiplicar dos números binarios codificados en complemento a 2 existen varias características a tener en cuenta:

- Las multiplicaciones en base binaria natural se realizan de forma similar a las multiplicaciones en base decimal
- En Ca2 si ambos factores son positivos se realiza de forma directa la operación y se añade un bit de signo 0 al resultado pues el mismo será un número positivo.
- En Ca2 si ambos factores son negativos se complementan a 2, se realiza la operación y al resultado se añade un bit de signo 0 ya que el resultado será positivo.
- Si alguno de ambos factores es negativo se hace el Ca2 a ese número, se realiza la operación y al resultado se realiza de nuevo el Ca2 añadiéndole un bit de signo 1 pues el resultado será negativo.

Entonces se tiene un proceso para poder elaborar esta multiplicación, la cual se listara, y será el flujo que seguirán todos los datos de entrada hasta la salida:

1. El valor de los 4 switch (sw) de entrada se almacenan en la parte alta de un vector auxiliar de 28 bits para la multiplicación (multiplier_aux), es decir, los valores del bit 23 a 0 deberán estar inicializados en ceros y del 24 al 27 serán los 4 bits de los switchs desde 0 a f en base hexadecimal.
2. Ahora se debe dividir el valor del vector de multiplicación auxiliar (multiplier_aux) entre

15, este vector es de 28 bits para no obtener un factor erróneo en el caso donde la división sea un valor máximo representable con 25 bits y pueda almacenarse en el vector multiplicador (multiplier) que como se vé deberá tener 25 bits .

3. Al obtener el valor real del factor dado por los switch es posible realizar la multiplicación en cada canal, esto será sincronizado con la intefaz AXI-S por medio de la señal Valid para saber que valores se deben operar, después Last para verificar que sean solo dos canales.
4. Se debe recordar que tanto la división para obtener un factor de 25 bits y la multiplicación de los valores en cada canal, deben ser hechos con funciones de conversión para que sean realizables.

La estructura que fue usada para inferir el DSP48E1 en la multiplicación es la que se muestra a continuación:

```
data_aux(0)<= std_logic_vector(signed(data(0))*to_integer(unsigned(multiplier)));
data_aux(1)<= std_logic_vector(signed(data(1))*to_integer(unsigned(multiplier)));

process(clk)
 begin
 if (rising_edge(clk)) then
   if (s_new_word = '1') then -- extensión de signo
     if (s_axis_last = '0') then
        data(0)(DATA_WIDTH-1 downto 0) <= s_axis_data;
        data(0)(MULTIPLIER_WIDTH+DATA_WIDTH-1 downto DATA_WIDTH)
        <= (others => s_axis_data(DATA_WIDTH-1));
    else
        data(1)(DATA_WIDTH-1 downto 0) <= s_axis_data;
        data(1)(MULTIPLIER_WIDTH+DATA_WIDTH-1 downto DATA_WIDTH)
        <= (others => s_axis_data(DATA_WIDTH-1));
    end if;
  elsif (s_new_packet_r = '1') then --inferencia del DSP48
     data(0) <= data_aux(0)(MULTIPLIER_WIDTH+DATA_WIDTH-1 downto 0);
     data(1) <= data_aux(1)(MULTIPLIER_WIDTH+DATA_WIDTH-1 downto 0);
    end if;
 end if;
end process;
```

En la siguiente figura 7-4, se puede observar los datos de entrada en el respectivo puerto “s_axis_data_s(23:0)” todos son en base hexadecimal “aacbee” y “bbceef” canal izquierdo y derecho respectivamente, ambos están en Ca2, después se verifica que el valor de los switch “sw_s(3:0)” es “f” el cual al ser dividido por 15 se obtiene “1000000”, que será el factor para multiplicar “multiplier(24:0)”, según los criterios de multiplicación antes vistos se realizo la operación directa, por lo que al multiplicar “1000000” con “aacbee” se obtiene “aacbee” en el puerto “m_axis_data_s(23:0)” dado que el factor se mantuvo en 1 lo cual no modifica el valor real de la amplitud de la señal de audio digital, se puede observar que se conserva el mismo signo del valor de entrada en la salida.

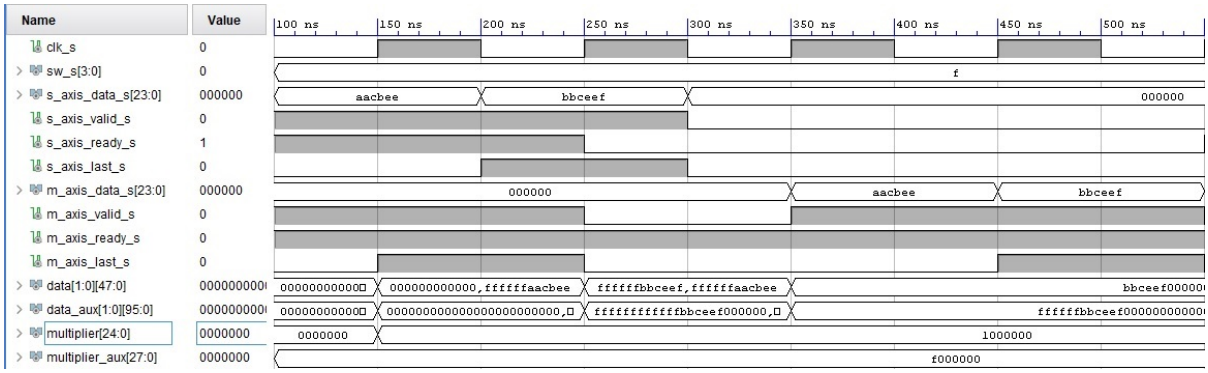


Figura 7-4: Diagrama de tiempo del controlador de volumen, multiplicación negativo con negativo.

Ahora se verá el caso particular donde la multiplicación se complica y no se obtiene un valor con la operación directa pues ambos factores son de signos diferentes, se tienen los datos de entrada en el puerto “s_axis_data_s(23:0)” todos son en base hexadecimal “aaaaaa” y “bbccdd” canal izquierdo y derecho respectivamente, después como factor se tiene “multiplier(24:0)” “0333333”, entonces se convierte el factor “aaaaaa” complementándolo a 2 “555556” el cual será multiplicado por el factor y se obtendrá en un largo del doble de bits “111111222222” al cuál nuevamente se complementa a 2 para obtener “eeeeeeddddde” del cual se hará el truncamiento de bits y solo se dejarán los 24 bits más significativos, así pasa a ser el dato de salida “eeeeee”, estos criterios de multiplicación se llevan a cabo mediante el uso

del DSP48 por lo que la única preocupación es hacer la representación adecuada del número al adicionarle bits en formato extendido del doble de tamaño inicial con la extensión de signo.

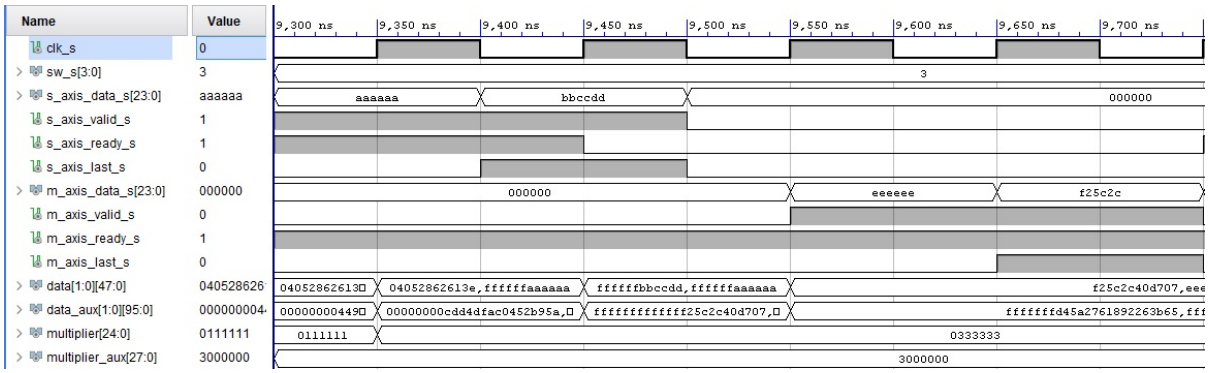


Figura 7-5: Diagrama de tiempo del controlador de volumen, multiplicación negativo con positivo.

Se tienen que sincronizar los datos de entrada, procesarlos y entregarlos en el momento adecuado a la salida, como parte de esclavo y maestro en la interfaz AXI-S, para ello se verifican 4 funciones combinacionales para englobar más simplemente el proceso de transferencia de datos, básicamente dos funciones , la primera que funciona para verificar cuando hay una nueva palabra y la segunda para verificar cuando hay un nuevo paquete, entonces se tiene 4 funciones puesto que son validas tanto para la parte del esclavo y la parte del maestro, se van a verificar éstas en la siguiente tabla 7-1:

función combinacional	m_axis_valid	m_axis_ready	m_axis_last	s_axis_valid	s_axis_ready	s_axis_last
m_new_word	1	1				
m_new_packet	1	1	1			
s_new_word				1	1	
s_new_packet				1	1	1

Tabla 7-1: Funciones combinacionales para AXI-S.

Con estas funciones es posible describir más fácilmente los procesos concurrentes que controlaran al módulo, por lo que se hará un proceso que controle la multiplicación en cada ciclo de reloj donde haya un flanco de subida, después otro proceso donde a cada nueva palabra se le aplique la extensión de signo, también el proceso de control para la señal de VALID y LAST que son parte de la interfaz maestr AXI, el proceso para la señal Ready en la interfaz esclava junto con los desplazamientos de datos entre registros auxiliares, con lo que al final se

puede obtener este diagrama de bloques 7-6 que describe el flujo del proceso.

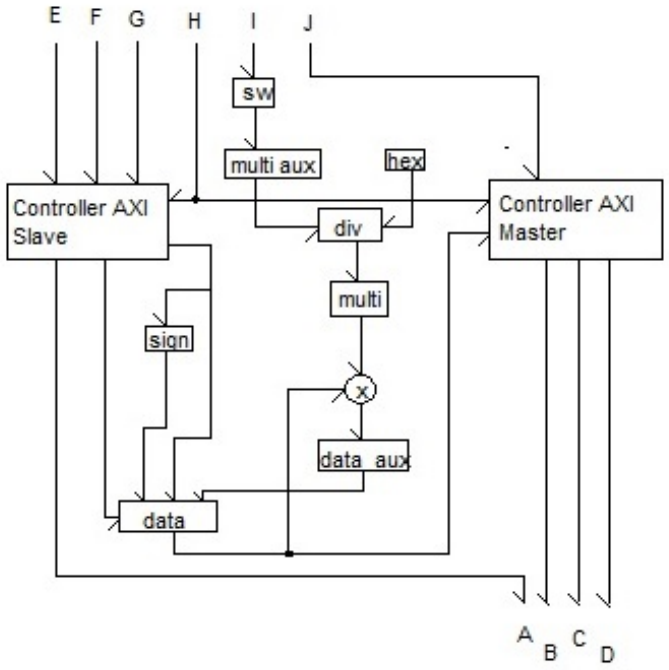


Figura 7-6: Diagrama de bloques del controlador de volumen.

A continuación se muestra la correspondencia de entradas y salidas de la imagen 7-6.

1. A.- s_axis_ready
2. B.- m_axis_valid
3. C.- m_axis_last
4. D.- m_axis_data
5. E.- s_axis_valid
6. F.- s_axis_last
7. G.- s_axis_data
8. H.- clk
9. I.- sw
10. J.- s_axis_data

Ahora se tiene que realizar el bloque del efecto “delay”, este trabaja retardando la señal de entrada y sumándola después a la señal original, así intuitivamente se necesitará entonces

una memoria en donde se almacenará esta señal durante un lapso de tiempo, también resulta evidente que no existe memoria en la cual sea posible escribir durante un tiempo indefinido sin llegar a ocupar la totalidad de esta, por lo que se necesita implementar una estructura especifica que tiene la ventaja de reocupar ciclicamente el espacio de esta memoria, llamada buffer circular, esta estructura usa una memoria ram de doble puerto donde las direcciones de lectura y escritura estarán cambiando en función de un contador, el cual deberá estar sincronizado con la interfaz AXI-S y solo avanzar en el caso donde una palabra o un nuevo paquete sea puesto en la señal de datos, que será un paquete de 2 palabras cada ciclo de muestreo, las direcciones de escritura serán tomadas directamente del contador que debe tener un largo de bits para direcciones tal que pueda recorrer desde la primera hasta la última localidad de la memoria, las direcciones de lectura serán tomadas después de sumar las de escritura más un valor que funcionará como retardo, es decir, el proceso será escribir en una posición anticipada a la de lectura por lo que entre mayor sea la diferencia entre una y otra dirección menor será el retardo, como ejemplo se debe suponer que la escritura tiene la dirección "0" y la lectura tiene la dirección "2" y que la totalidad de localidades sean "100", entonces en las primeras 98 lecturas se obtendrá lo que se almaceno con anterioridad en la memoria, si solo son ceros no habrá efecto de "delay" en el audio, pero antes de reiniciar el conteo ya será posible leer los datos de entrada, por lo que se concluye que el retraso máximo estará dado por la profundidad de la memoria, a continuación se verá la estructura del buffer circular en la figura 7-7

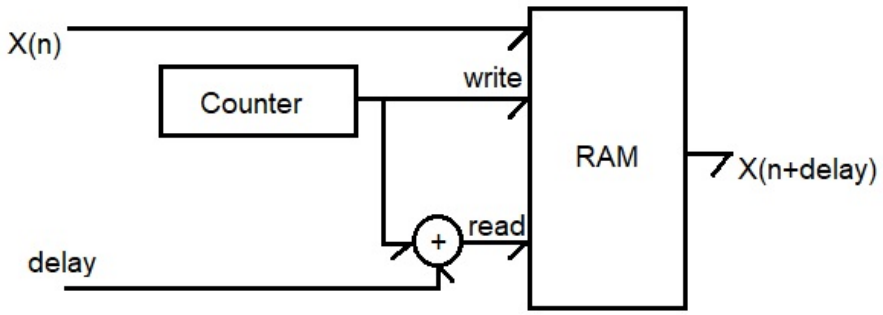


Figura 7-7: Diagrama de buffer circular, fuente: [22].

La estructura principal está basada en una memoria RAM de doble puerto, un puerto para

las direcciones de lectura y otro para las de escritura, de un tamaño de 24 bits por 32768 localidades o que es lo mismo 786 Kbits aproximadamente, un tamaño de 15 bits para las direcciones que van de 0 a 32767, este valor corresponde con el contador, será un contador descendente que reiniciará el conteo en 0 y comenzará nuevamente, el sumador tiene ambas entradas de 15 bits así como su salida, estas dos últimas entidades pueden ser implementadas con un bloque IP o realizarlas con descripción de hardware. Al final la implementación tiene esta estructura que se observa en la siguiente figura 7-8.

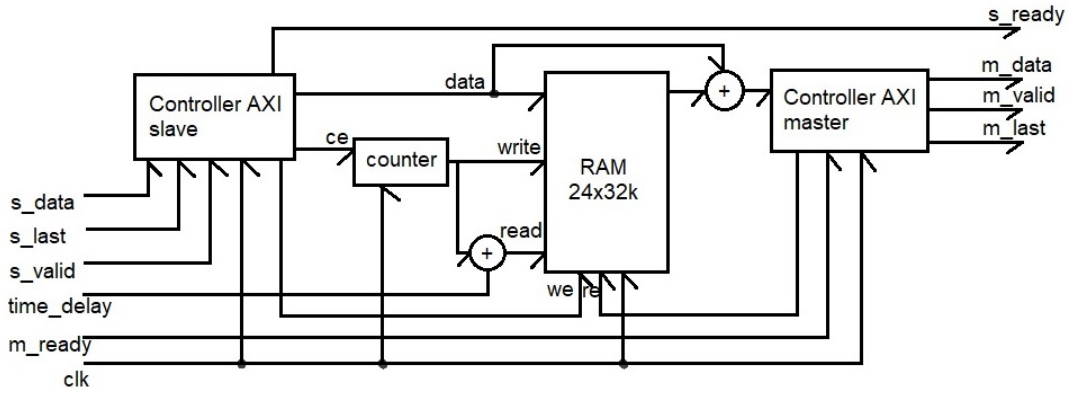


Figura 7-8: Diagrama del efecto “delay” con AXI-S.

Para probar el módulo se necesita verificar primero la función del doble puerto en la memoria RAM por lo que se inicializarán todos los elementos de esta a cero para evitar valores “U” desconocidos, entonces se probará que para un mismo ciclo de reloj se puede obtener tanto valores de lectura como de escritura, en la siguiente figura 7-9 se tiene un diagrama de tiempos donde en el primer ciclo de clk se activa una lectura y escritura de la misma dirección de memoria, se puede observar que para la misma dirección en lectura y escritura “0000” con una asignación de dato “ffffff” en la entrada “di_s(23:0)” la respuesta obtenida en la salida es el valor “000000” por lo que se establece que la memoria tiene primero la función lectura, ya que 4 ciclos después se vuelve a repetir el ciclo de la prueba y allí se obtiene el valor de dato escrito en el primer ciclo, en el segundo ciclo se pide el dato almacenado en el ciclo anterior y ese comportamiento se prueba en los siguientes tres ciclos, por lo que queda probada la memoria RAM de doble puerto.

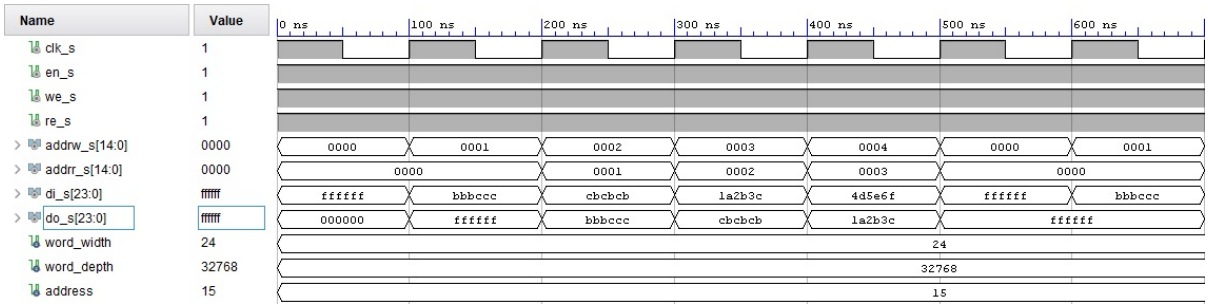


Figura 7-9: Diagrama de tiempo memoria ram.

El sistema recorrerá las localidades de memoria en los ciclos en donde comience una transferencia, por lo que se necesita una lógica combinacional, se tendrán entonces las señales de la interfaz esclava VALID y READY en alto para la transferencia, en ese momento debe activarse el contador con una dirección de escritura valida y una dirección de lectura que, sumada al retardo, siempre se atrasará respecto al proceso de escritura, después para el siguiente dato debe activarse la señal esclava Last que indica la palabra final del paquete, para el paquete completo se cuenta con dos canales, izquierdo y derecho respectivamente, por lo que en ese momento se genera la segunda dirección de escritura y la dirección de lectura sumada al mismo retardo, si esto se piensa como un proceso iterativo, se vé que la diferencia entre la dirección de lectura (siempre con un valor mayor) y la dirección de escritura (siempre con un valor menor) contendrá la varianza en el retardo pues al estar adelante por muy pocas direcciones la lectura de la escritura significará que se tendrá el menor “delay” porque para que se lea el dato dado un tiempo t_0 tomara aproximadamente el total de direcciones de diferencia, esto es debido a que el mismo reloj gobierna el avance en las direcciones de lectura/escritura, de otra forma se tiene que si el proceso de lectura esta a muy pocas direcciones por detrás de la escritura, o también puede verse como el proceso de lectura muy separado de la escritura, el tiempo de “delay” será mayor ya que lo escrito será leído de la memoria hasta casi pasar por la mayoría de sus direcciones, se debe recordar que este proceso se le llama direccionamiento indexado y tener en conocimiento que en una pila el crecimiento de la misma es en orden decreciente de direcciones de memoria, esto quiere decir que el contador que proporciona las direcciones deber

ir en conteo descendente, así pues se tiene que el "delay" esta dado por:

$$Delay_{ms} = \frac{retardo}{(Fs * Dp)} * 1000 \tag{7-4}$$

Donde $retardo$ puede ir de 0 a 32767 en una memoria de 2^{15} direcciones, Fs es la frecuencia de muestreo y Dp son datos del paquete, en este caso 2 por el canal derecho e izquierdo. Entonces para calcular 10ms, 50ms, 100ms, 200ms y 370ms se tiene lo siguiente. Despejando retardo de 7-4 se tiene:

$$retardo = \frac{Delay_{ms}}{1000} * (Fs * Dp) \tag{7-5}$$

$$retardo = \frac{10_{ms}}{1000}(44100 * 2) = 882_{10} = 372_{16}$$
$$retardo = \frac{50_{ms}}{1000}(44100 * 2) = 4410_{10} = 113A_{16}$$
$$retardo = \frac{100_{ms}}{1000}(44100 * 2) = 8820_{10} = 2274_{16}$$
$$retardo = \frac{200_{ms}}{1000}(44100 * 2) = 17640_{10} = 44E8_{16}$$
$$retardo = \frac{370_{ms}}{1000}(44100 * 2) = 32634_{10} = 7F7A_{16}$$

Con esos valores es posible asignar el retraso necesario al "delay" dependiendo cuanto tiempo medido en ms es requerido. Para el sistema de control AXI se implementará mediante el uso de máquinas de estado finito, en este caso se usará Moore ya que las salidas solo dependen del estado presente entonces se puede definir la siguiente tabla 7-2 con los estados, entradas, salidas y las señales que rigen las transiciones.

Puede observarse en la tabla 7-2 que se tiene tres estados, uno de espera que es Q1, otro de escritura y lectura del canal izquierdo Q2 y el último de lectura y escritura del canal derecho Q3, estos están regidos por las señales de control del esclavo y entrada del maestro, entonces se ven las señales que indican el estado de la máquina en sus entradas y salida y la ecuación

Estado	Entrada	Salida	Transición	Estado *	Entrada*	Salida*
Q1	s_valid = 0 s_last = 0 m_ready =0	m_valid = 0 m_last = 0 s_ready = 1	s_ready=1 & s_valid=1	Q2	s_valid = 1 s_last = 0 m_ready=1	m_valid =1 m_last=0 s_ready=1
Q2	s_valid = 1 s_last = 0 m_ready=1	m_valid =1 m_last =0 s_ready=1	s_valid=1 & s_last = 1	Q3	s_valid = 1 s_last =1 m_ready=0	m_valid =1 m_last=1 s_ready=0
Q3	s_valid = 1 s_last = 1 m_ready=0	m_valid = 1 m_last = 1 s_ready = 0	s_last= 0	Q1	s_valid = 0 s_last=0 m_ready=0	m_valid =0 m_last=0 s_ready=1

Tabla 7-2: Estados para máquina de Moore en AXI-S.

de transición. Esta máquina de estados debe seguir una sola ruta por la naturaleza de la interfaz AXI-S ya que no puede escribir solo un canal o interrumpir este proceso de escritura lectura, por lo que solo debe esperar por las condiciones necesarias para comenzar el proceso de almacenamiento en el bufer circular. Adicionalmente a la máquina de estados se pueden utilizar sus procesos combinacionales para adecuar las señales de lectura escritura dependiendo el estado presente y además se debe tener en cuenta la propagación de los datos y señales de control a través de los multiplexores y la máquina de estados por lo que es necesario almacenar en registros auxiliares los datos para ingresarlos u obtenerlos en el ciclo de reloj adecuado.

A continuación, se observa el testbench del módulo buffer circular 7-10, para verificar rápidamente el proceso de direccionamiento indexado junto con el almacenamiento, lectura de datos y sincronizado con la interfaz AXI-S se le dio un retardo igual a dos para que fuese en el siguiente ciclo de lectura/escritura la comprobación del módulo, se debe recordar que por evento de lectura/escritura se almacenan dos datos de 24 bits. Se ve en los ovalos de color rojo los datos en la entrada al módulo mediante la interfaz esclava s_axis_data(23:0), se tiene “aacbee” en base hexadecimal como canal izquierdo y “bbceef” en base hexadecimal como canal derecho, se observa que los mismos datos se encuentran en el puerto di(23:0) que corresponde a la entrada de datos de la RAM de doble puerto, después de 7 ciclos aproximadamente se tiene en el puerto do(23:0) que corresponde a los datos de salida de la memoria, los datos que anteriormente se habían almacenado en las direcciones “7fff” y “7ffe” respectivamente, y de esa misma forma los datos se observan en la terminal m_axis_data(23:0), todos los datos

mencionados se remarcaron en rojo, también se verifican las direcciones de donde se almacenó y se obtuvieron los datos estos se muestran en color morado y al final se observa en la entrada/ salida de la interfaz AXI-S las señales en alto de s_axis_last y m_axis_last para concluir el correcto funcionamiento del módulo, con esto se prueba un contador de 15 bits mas un sumador de 15 bits, la memoria RAM y la sincronización con la interfaz AXI implementada en una máquina de estados de Moore que se verifica en las variables “present_state” y “next_state”, el tiempo que tarda el buffer en almacenar y devolver un paquete de dos datos son 8 ciclos de reloj.

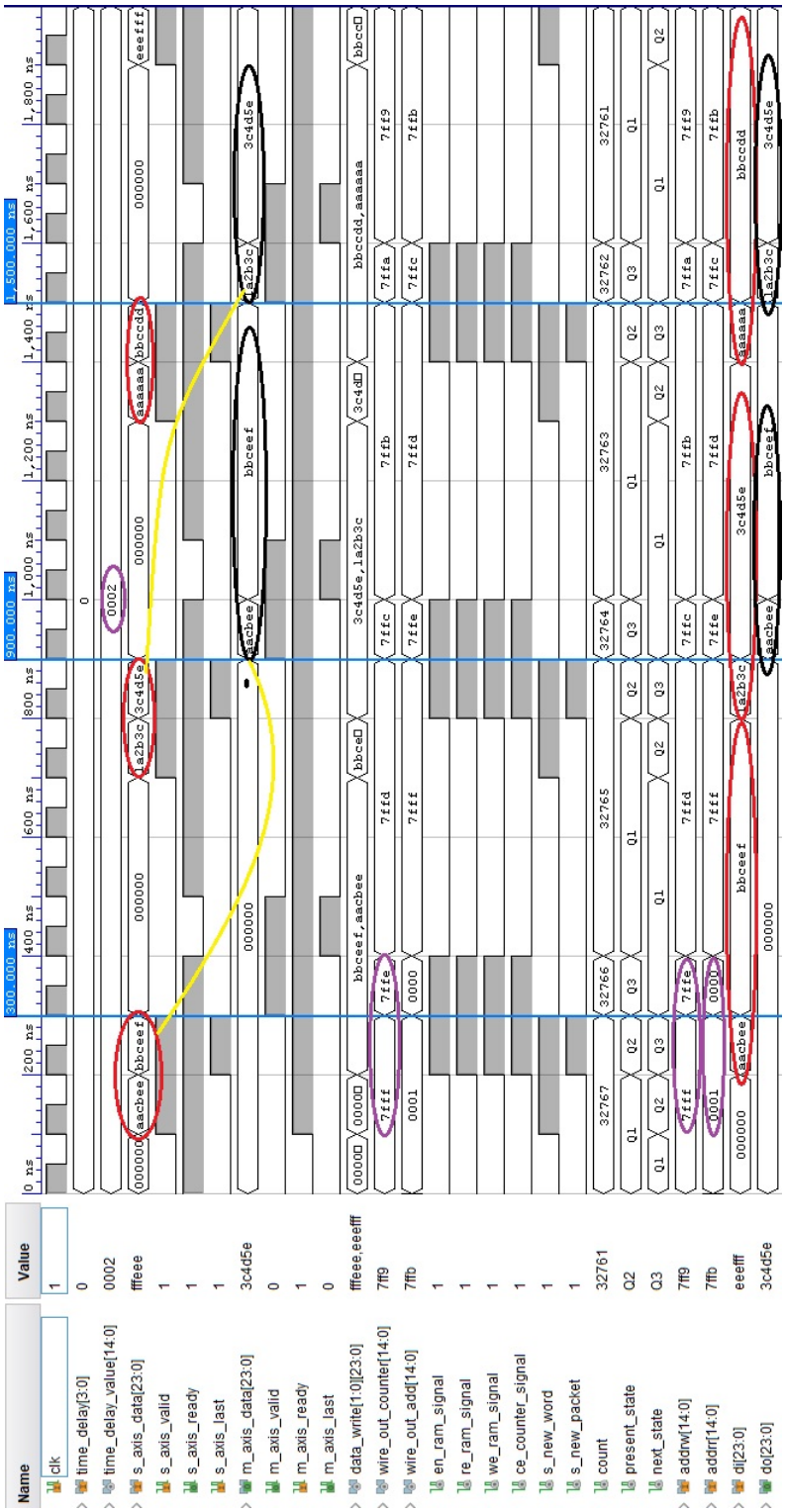


Figura 7-10: Diagrama de tiempo de buffer circular, obtenido del IDE Vivado.

Después para el último paso del efecto “delay” hay que sumar la señal de entrada al módulo con la señal de salida del buffer circular, este proceso involucra almacenar nuevamente los datos del canal izquierdo y derecho de las dos diferentes interfaces puesto que llegan con un retraso de propagación y no es posible sumarlas en el mismo ciclo de reloj, enseguida se deben generar nuevamente las señales de control para indicar la validez de los datos y el último dato del paquete, se debe recordar que la señal ready en la intefaz maestra es una entrada por lo que se desplaza a través del módulo sin ser afectada hasta los circuitos combinacionales donde indica una nueva transferencia, para esto se genera otra prueba de tiempo, este diagrama indica la entrada y salida del total del módulo, así como los procesos y señales de control, se ven en la figura 7-11

Se observa en color Rojo los datos de la entrada esclava siendo almacenados en la memoria, sumados y puestos en la salida maestra, enseguida esta otro ciclo igual en color morado donde ahora la salida maestra muestra datos diferentes a los almacenados puesto que en este nuevo ciclo existen dos datos diferentes de cero para sumar, lo que produce el efecto “delay” en el audio.

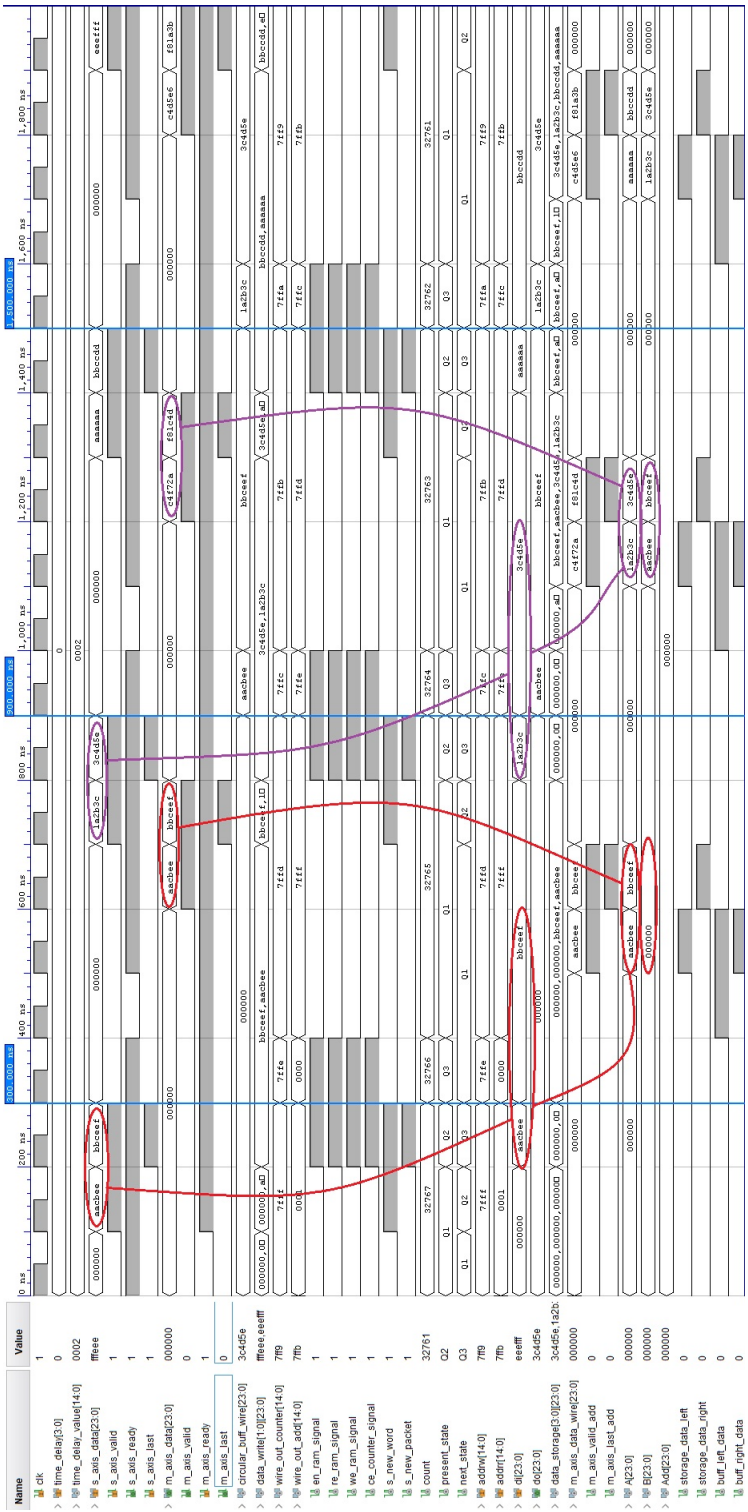


Figura 7-11: Diagrama de tiempo de efecto “delay”, obtenido del IDE Vivado.

8

Codificación conceptual en VHDL

Para poder hacer la descripción de todo el proyecto en VHDL se deben tomar en cuenta cada módulo y sub módulo expuesto, desde el capítulo de jerarquías de diseño que contiene la correspondencia de cada entidad, entonces se colocará cada módulo codificado, así como las especificaciones de módulos IP ocupados y no especificados previamente.

La lista comenzará con el reloj maestro, se utilizo la IP Clock_wizard que toma el valor del reloj a 100 MHz y lo reduce a 22.5792 MHz, se tiene un listado de sus configuraciones:

1. Board. CLK_IN1 Custom, CLK_IN2 Custom y EXT_RESET_IN Custom
2. Clocking Options
    - Disabled CLock Monitoring
    - Primitive : MMCM
    - Clocking Features on: Frecuency Syntesis, Phase Alignment
    - Jitter Optimization : Balanced
    - Input Clock Information : Clk_in1 100.000 MHz
3. Output Clocks
    - Output Clock clk_out1 on
    - Port Name: axis_clk
    - Output Freq (MHz): Request 22.5792
    - Phase(degrees): Request 0.00

- Duty Cycle(%): 50.0
- Enable Optional Inputs/Outputs for MMCM/PLL on: reset
- Reset Type on: Active High

A continuación esta el módulo de la interconexión entre las interfaces I2S del ADC y DAC con la interfaz AXI-S Slave y Master. Se mostrarán los procesos correspondientes con el módulo descrito, en el pseudocódigo siguiente se observa el proceso para el control de la señal READY del esclavo, solo podrá colocar a uno la señal de listo para poder recibir otro paquete hasta que la totalidad del paquete del lado de I2S2 se haya completado o lo que es mismo a un total de 456 ciclos de reloj.

```
Proceso Control_AXI_I2S2
Si axis_resetn = 0 Entonces
        tx_axis_s_ready <- 0
SiNo
 Si tx_axis_s_ready = 1 y tx_axis_s_valid = 1 y tx_axis_s_last = 1 Entonces
        tx_axis_s_ready <- 0
 SiNo
        Si Count = 0 Entonces
                tx_axis_s_ready <- 0
        SiNo
                Si Count= 455 Entonces
                        tx_axis_s_ready <- 1
                Fin Si
        Fin Si
 Fin Si
Fin Si
FinProceso
```

En el siguiente pseudocódigo se puede apreciar como se vacían los registros que almacenan temporalmente a los datos izquierdo y derecho de la misma interfaz esclava al momento de un reset, se debe recordar que este reset se produce en estado bajo, por el contrario si no hay reset entonces se produce el almacenamiento del dato encontrándose en primer lugar el dato del canal izquierdo y después el canal derecho.

```
Proceso Control_AXI_I2S2_Sdata
        Si axis_resetn = 0 Entonces
                tx_data_r <- 0
                tx_data_l <- 0
        SiNo
                Si tx_axis_s_ready = 1 y tx_axis_s_valid = 1  Entonces
                        Si  tx_axis_s_last = 1 Entonces
                                tx_data_r <- tx_axis_s_data
                        SiNo
                                tx_data_l <- tx_axis_s_data
                        Fin Si
                Fin Si
        Fin Si
FinProceso
```

Enseguida es posible ver la asignación de los datos del la interfaz AXI esclava a la parte I2S del DAC, el contador "Count", como se vé, contiene en combinaciones de sus bits asignados cada dato, como se mencionó en el capitulo respectivo, por lo que se vá descomponiendo bit a bit la palabra completa obtenida de la interfaz AXI, se debe recordar que el primer dato a enviar es el MSB (Most Significant Bit) entonces se observa como es el bit 23 el que se asigna ciclo a ciclo en la salida seria "tx_sdout"

```
Proceso Control_AXI_I2S2_I2Sdataout
        Si (Count (7 a 3) = 1 y count (7  a 3) = 24) Entonces
                Si Count(8) = 1  Entonces
                        tx_sdout <- tx_data_r_shift(23)
                        SiNo
                        tx_sdout <- tx_data_l_shift(23)
                        Fin Si
                SiNo
                        tx_sdout <- 0
        Fin Si
FinProceso
```

Ahora se tratarán los proceso correspondientes al receptor de I2S e interfaz maestra AXI, en

el siguiente pseudocódigo se observa como es un proceso inverso del proceso anterior, pues este se toma a partir del conteo adecuado conformado por “Count” donde la palabra se forma a partir del primer bit en la parte inferior de la misma, es decir, el bit MSB toma el espacio del LSB (Least Significant Bit), pero gracias a la asignación secuencial, este bit se va recorriendo posicionalmente por lo que al terminar la entrada de bits en “rx_sdin” es posible obtener el dato completo incluyendo el bit LSB.

```
Proceso Control_AXI_I2S2_I2Sdatain
Si (Count (2 a 0) = 0 y Count (7 a 3) = 1 y Count (7 a 3) = 24) Entonces
        Si lrck = 1  Entonces
                rx_data_r_shift <- rx_data_r_shift(22 a 0) + rx_sdin
        SiNo
                rx_data_l_shift <- rx_data_l_shift(22 a 0) + rx_sdin
        Fin Si
Fin Si
FinProceso
```

Después de conformar el dato bit a bit es preciso mandarlo cuando se establezca la condición en donde la validez de datos del maestro aun debe ser falsa y el conteo ha llegado a indicar el fin del paquete, estos registros auxiliares son muy importantes ya que hacen posible la sincronización de todas las entregas y recepciones de datos, para entregarlo a la salida de la interfaz solo se debe tener una señal a la que se puede consultar “m_last” que indicará la última palabra o canal derecho de audio.

```
Proceso Control_AXI_I2S2_Sdata
        Si axis_resetn = 0 Entonces
                rx_data_r <- 0
                rx_data_l <- 0
                Si Count= 455 y rx_axis_s_valid = 0  Entonces
                                rx_data_r <- hex'00' + rx_data_r_shift
                        SiNo
                                rx_data_l <- hex'00' + rx_data_l_shift
                        Fin Si
```

```
                Fin Si
FinProceso
```

```
Proceso Control_AXI_I2S2_Mdata
Si rx_axis_m_last = 0 Entonces
        rx__axis_m_data <- rx_data_l
SiNo
        rx__axis_m_data <- rx_data_r
Fin Si
FinProceso
```

Ahora se ve el proceso para el control de la señal de VALID en la interfaz maestra, dependiendo del valor de reset y el conteo es como se activa la señal a 1, es importante observar que estos procesos son de forma paralela por lo que las señales intervienen todas juntas formando así el bloque funcional, se puede ver que cuando el conteo llega a 455 y la señal de "s_valid" es 0 en el proceso anterior se colocaban los datos en los registros auxiliares para después de tener un valor de m_last se pudiese indicar el orden de palabra en el paquete junto con su validez.

```
Proceso Control_AXI_I2S2_Mvalid
Si axis_resetn = 0 Entonces
        rx_axis_s_valid <- 0
  Si Count= 455 y rx_axis_s_valid = 0  Entonces
        rx_axis_s_valid <- 1
   Si rx_axis_s_valid = 1 y rx_axis_m_ready = 1
      y rx_axis_m_last = 1 Entonces
                rx_axis_m_valid <- 0
        Fin Si
  Fin Si
Fin Si
FinProceso
```

Para concluir con los procesos del módulo I2S AXI se tiene el proceso que controla la señal de LAST de parte de la interfaz maestra AXI, al igual que las señales anteriores esta regida por la

señal de reset, se observa que después de enviar la primera palabra y que los datos contengan la validez y la señal de READY de parte de la interfaz esclava, se pone en contra de su estado original es decir, se pone en 1 mientras haya pasado la transferencia del primer dato del paquete, por lo que en el siguiente ciclo y manteniendo esas condiciones indicará el fin del paquete.

```
Proceso Control_AXI_I2S2_Mlast
Si axis_resetn = 0 Entonces
        rx_axis_s_last<- 0
  Si Count= 455 y rx_axis_s_valid = 0  Entonces
                rx_axis_s_last <- 0
        Si rx_axis_s_valid = 1 y rx_axis_m_ready = 1  Entonces
                        rx_axis_m_last <- notrx_axis_last
                Fin Si
        Fin Si
 Fin Si
FinProceso
```

Los módulos de filtrado FIR_compiler se nombran por default fir_compiler_0, fir_compiler_1 y fir_compiler_2 ya se sabe su configuración gracias a los listados en su capítulo correspondiente, así que esta parte solo contiene el módulo TOP que se nombra filtered_fir y dos módulos más un demultiplexor de 1x2 y un multiplexor de 2x1 que hacen función de bypass o seleccionar un filtro en especifico, también se tienen los tres conjuntos de coeficientes, estos se deben almacenar en un archivo de formato ASCII y cargarlos a cada módulo FIR Correspondiente, pasa altas, pasa bajas y pasa banda, que están almacenados en un repositorio de GitHub del hipervínculo siguiente https://github.com/RobertoCarlosGutierrezZacarias/Tratamiento-de-audio-digital-con-interfaz-AXI-Stream-e-implementaci-n-en-FPGA.git , allí mismo se encuentran todo el codigo en VHDL de cada módulo además de una descripción breve acerca de las características más relevantes.

El penúltimo módulo corresponde al efecto “delay”, este al igual que el de filtrado, contiene módulos más pequeños, es posible intuir que algunos pseudocódigos serán establecidos de forma similar debido al tipo de interfaz usada, específicamente en la parte de control. A continuación,

se puede ver el contador descendente que genera los valores de las direcciones para lectura y escritura. de aquí se desprende una señal de habilitación que depende de la validez del dato, pues no en todos los ciclos de reloj deben establecerse direcciones de memoria, por lo que es una parte importante controlar el recorrido en la memoria RAM.

```
Proceso RingBufferAddress
        Si ce_counter_signal = 1 Entonces
                Si count= 0  Entonces
                        count <- 32767
                SiNo
                        count <- count - 1

                Fin Si
        Fin Si
FinProceso
```

Al igual que en los procesos anteriores, se manejan los datos de la entrada esclava AXI para poder almacenarlos y tomar en cuenta la propagación de las demás señales.

```
Proceso RingBufferDataRegisters
        Si s_new_word = 1 Entonces
                Si s_new_packet= 0  Entonces
                        data_write[0](23 a 0) <- s_axis_data
                SiNo
                        data_write[1](23 a 0) <- s_axis_data

                Fin Si
        Fin Si
FinProceso
```

Aquí se inicia la máquina de estados donde se relacionan las entradas y salidas adecuadas dependiendo de un momento o estado, esta máquina desarrollada es de Moore ya que las salidas dependen únicamente de los estados, se tiene entonces al proceso secuencial:

```
Proceso RingBufferSecuencial
        Si axis_resetn = 0 Entonces
                    present_state <- Q1
                SiNo
                        present_state <- next_state
                Fin Si
FinProceso
```

El proceso combinacional 1, que se muestra a continuación, describe los estados en función de la entrada esclava AXI y las señales de asignación a la memoria RAM y al contador de direcciones.

```
Proceso RingBufferCombinacional1
Segun present_state Hacer
        Q1:  wire_s_data <- 0
                ce_counter_signal <- 0
                en_ram_signal     <- 0
                we_ram_signal     <- 0
                re_ram_signal     <- 0

                Si s_new_word = 1 Entonces
                        next_state <- Q2
                SiNo
                        next_state <- Q1
                Fin Si
        Q2:
                wire_s_data <- data_write[0](23 a 0)
                ce_counter_signal <- 1
                en_ram_signal     <- 1
                we_ram_signal     <- 1
                re_ram_signal     <- 1

                Si s_new_packet = 1 Entonces
                        next_state <- Q3
                SiNo
                        next_state <- Q2
```

```
25                  Fin Si
26          Q3:
27                  wire_s_data <- data_write[1](23 a 0)
28                  ce_counter_signal <- 1
29                  en_ram_signal     <- 1
30                  we_ram_signal     <- 1
31                  re_ram_signal     <- 1
32
33                  Si s_new_word = 0 Entonces
34                          next_state <- Q1
35                  SiNo
36                          next_state <- Q3
37                  Fin Si
38  Fin Segun
39  FinProceso
```

Se continua con un segundo proceso combinacional de la máquina de Moore el cual, dependiendo el estado en que se encuentra, asigna los valores combinacionales de control a la salida AXI maestra, esta le agrega un ciclo de reloj de propagación que junto con el ciclo de reloj que toma en almacenar y leer datos desde la RAM hace que al final los datos y las señales de control esten sincronizadas.

```
1   Proceso RingBufferCombinacional2
2   Si axis_resetn = 0 Entonces
3                   m_axis_valid <- 0
4                   m_axis_last  <- 0
5                   m_axis_ready <- 1
6           SiNo
7                   Si present_state = Q1 Entonces
8                           m_axis_valid <- 0
9                           m_axis_last  <- 0
10                          m_axis_ready <- 1
11                  Si present_state = Q2 Entonces
12                          m_axis_valid <- 1
13                          m_axis_last  <- 0
14                          m_axis_ready <- 1
```

```
                Si present_state = Q3 Entonces
                        m_axis_valid <- 1
                        m_axis_last  <- 1
                        m_axis_ready <- 0
                        Fin Si
                Fin Si
        Fin Si
Fin Si
FinProceso
```

Ahora en la parte del efecto “delay” con la jearquia mayor se tiene un proceso de almacenamiento de datos para que en conjunto con los datos de la salida del bufer circular y la interfaz esclava se sumen en el mismo ciclo de reloj.

```
Proceso EffectSDatos
Si s_axis_valid = 1 Entonces
                Si s_axis_last = 0 Entonces
                        data_storage[0](23 a 0) <- s_axis_data

                SiNo
                        data_storage[1](23 a 0) <- s_axis_data
                        Fin Si
                Fin Si
FinProceso
```

Después se almacenan los datos provenientes del bufer circular y como este elemento es el que tiene más retraso se considera el mismo para utilizar una banderas que indiquen el almacenamiento de cada palabra, se trata de las señales “buff_left_data” y “buff_right_data”.

```
Proceso EffectBCDatos
Si m_axis_valid = 1 Entonces
                Si m_axis_last = 0 Entonces
                        data_storage[2](23 a 0) <- circular_buffer
                    buff_left_data <- 1
                SiNo
```

```
                        data_storage[3](23 a 0) <- circular_buffer
                        buff_right_data <- 1
                        Fin Si
    SiNo
                         buff_left_data  <- 0
                         buff_right_data <- 0
                Fin Si
FinProceso
```

En el pseudocódigo siguiente se muestran dos proceso combinacionales para banderas de disponibilidad de datos almacenados en registros auxiliares junto con un tercer proceso que se encarga de generar las señales de control para la interfaz maestra AXI.

```
Proceso EffectBanderaLeft
                Si buff_left_data = 1 y buff_right_data = 0 Entonces
                        storage_data_left <- 1
                SiNo
                        storage_data_left <- 0
                Fin Si
FinProceso

Proceso EffectBanderaRight
                Si buff_left_data = 1 y buff_right_data = 1 Entonces
                        storage_data_right <- 1
                SiNo
                        storage_data_right <- 0
                Fin Si
FinProceso

Proceso EffectSumDEControl
Si m_axis_ready = 1 Entonces
        Si storage_data_left = 1 y storage_data_right = 0 Entonces
                m_axis_valid <- 1
                m_axis_last  <- 0

        Si storage_data_left = 0 y storage_data_right = 1 Entonces
                m_axis_valid <- 1
```

```
                m_axis_last  <- 1
        SiNo
                m_axis_valid <- 0
                m_axis_last  <- 0
        Fin Si
 Fin Si
FinSi
FinProceso
```

En penultimo, lugar se puede ver que con la combinación de las banderas anteriores se pueden asignar a la entrada del sumador los datos correspondientes en cada canal correctamente sincronizados.

```
Proceso EffectSumDEffect
Si m_axis_ready = 1 Entonces
        Si storage_data_left = 1 y storage_data_right = 0 Entonces
                A <- data_storage[0](23 a 0)
                B <- data_storage[2](23 a 0)
        Si storage_data_left = 0 y storage_data_right = 1 Entonces
                A <- data_storage[1](23 a 0)
                B <- data_storage[3](23 a 0)
        SiNo
                A <- 0
                B <- 0
                Fin Si
        Fin Si
FinProceso
```

El último módulo a describir por medio de pseudocódigo es el control de volumen, a continuación se observa el proceso para la señal de control READY del esclavo.

```
Proceso Control_AXI_VC_Sready
        Si s_new_packet = 1 Entonces
                        s_axis_ready <- 0
                Si  m_new_packet = 1 Entonces
```

```
5                                   s_axis_ready <- 1
6                           SiNo
7                                   s_axis_ready <- 0
8                   Fin Si
9           Fin Si
10  FinProceso
```

Después se encuentra el proceso de multiplicación de los datos con el factor definido por la posición de los interruptores a la entrada. Se observa en la primer asignación al dato de la entrada esclava, se asignan los primeros 24 bits a la parte baja de "data" y seguido a ello se rellenan los 24 bits siguientes con el mismo valor que contiene el MSB de la entrada, esta asignación es en realidad la extensión de signo para representar el mismo número con un mayor número de bits en complemento a 2, se hace tanto para el dato izquierdo como el derecho, también el valor total del resultado de la multiplicación es el doble del número de bits de los factores por lo que el registro "data_aux" es de 96 bits, al final solo se toman los 24 bits más representativos y se mandan a la salida "m_axis_data", como se observa en la asignación concurrente que esta fuera del proceso.

```
1   Proceso ControlVolumeMult
2           Si s_new_word = 1 Entonces
3                   Si s_axis_last = 0 Entonces
4                         data[0] (23 a 0) <-  s_axis_data
5                             data[0] (47 a 23) <-  s_axis_data(23)
6                     SiNo
7                             data[1] (23 a 0) <-  s_axis_data
8                             data[1] (47 a 23) <-  s_axis_data(23)
9                     Fin Si
10                    Si s_new_packet_r = 1 Entonces
11                            data[0] (47 a 0) <-  data_aux[0] (47 a 0)
12                            data[1] (47 a 0) <-  data_aux[1] (47 a 0)
13                  Fin Si
14          FinSi
15  FinProceso
16
```

```
data_aux[0](95 a 0)<- data[0](47 a 0)* multiplier(24 a 0)
data_aux[1](95 a 0)<- data[1](47 a 0)* multiplier(24 a 0)
```

En el siguiente proceso al vector multiplier_aux de 28 bits inicializado en ceros se le hace una asignación de los 4 bits en la parte alta del mismo es decir, de 7 partes de 4 bits en los bits (27 a 23) se almacenan los bits de los interruptores y después se dividen entre 15, que es el valor que comprende un carácter hexadecimal para que ese mismo vector tome el valor del multiplicador, se hace una asignación de 25 bits puesto que si se dejaran en 24 al tomar valor de F en los interruptores se eliminaría un valor de acarreo que es importante para tomar el valor de 1.0 que como se observa en el pseudocódigo se lleva a cabo a cada ciclo de reloj, como bandera esta la señal de nuevo paquete que indica cuando ha acabado la transmisión del mismo paquete, por lo que la multiplicación en ese momento esta hecha.

```
multiplier_aux(27 downto 23) <- sw

Proceso ControlVolumeSwitch
        multiplier(24 a 0) <- multiplier_aux(28 a 0)/ 15
        s_new_packet_r <- s_new_packet
FinProceso
```

Después se pueden ver los procesos combinacionales para las dos señales de control m_axis_valid y m_axis_last. En ellos se pueden observar combinaciones especiales que tienen lugar con señales de entradas básicas que son simplemente reducciones por criterio para englobar una combinación especifica, new_packet y new_word, tanto para el maestro como para el esclavo, están también fueron usadas en el módulo “delay”.

```
m_new_word  <= 1 Si m_axis_valid= 1 y m_axis_ready =1
                    SiNo '0';
m_new_packet<= 1 Si m_new_word = 1  y m_axis_last  =1
                    SiNo '0';
s_new_word  <= 1 Si s_axis_valid =1 y s_axis_ready =1
                    SiNo '0';
```

```
s_new_packet<= 1 Si s_new_word = 1  y s_axis_last  =1
                    SiNo '0';

Proceso Control_AXI_VC_Mvalid
        Si s_new_packet_r = 1 Entonces
                        m_axis_valid <- 1
                Si  m_new_packet = 1 Entonces
                        m_axis_valid <- 0
                Fin Si
        Fin Si
FinProceso

Proceso Control_AXI_VC_Mlast
        Si m_new_packet_r = 1 Entonces
                        m_axis_last <- 0
                Si  m_new_word = 1 Entonces
                        m_axis_last <- 1
                Fin Si
        Fin Si
FinProceso
```

Finalmente se observa el proceso de envio de datos a la salida de la interfaz AXI maestra, que se sincroniza con las señales de control y estos datos son obtenidos directamente de los registros auxiliares que funcionaron tambien para el proceso multiplicativo.

```
Proceso ControlVolumeMdata
        Si m__axis_valid = 1 y m_axis_last = 0 Entonces
                m_axis_data <- data[0] (47 a 23)
        Si m__axis_valid = 1 y m_axis_last = 1 Entonces
                m_axis_data <- data[0] (47 a 23)
        SiNo
                m_axis_data <- 0
        Fin Si
FinSi
FinProceso
```

8.1. Diagrama de flujo del proceso de diseño

Como se especifica en el primer capítulo, se tiene una solución Híbrida entre TopDown y BottonUp por lo que el proceso toma algunas características de cada método e involucra estas en nuevas etapas que se realizaron para poder diseñar este proyecto, el diagrama que muestra el flujo del proceso de diseño se muestra en la figura 8-1. Es posible ver que el primer paso es

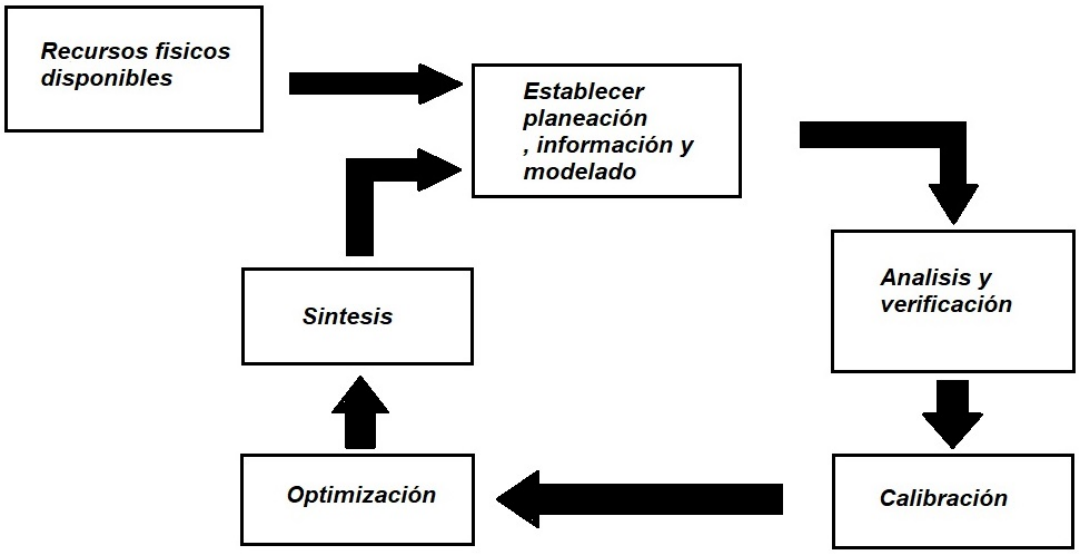


Figura 8-1: Diagrama del Flujo de diseño híbrido.

verificar los recursos físicos, en este caso la tarjeta de desarrollo disponible Basys 3, IDE vivado con disponibilidad para su uso, el módulo PMOD I2S2 que tendrá la función de adquirir la señal analógica y convertirla a datos digitales y también de manera inversa, con ello se pudo establecer un módulo básico, al estar trabajando con el ambiente Xilinx se optó por el desarrollo de su interfaz que trata de estandarizar en sus IP cores llamada AXI-Stream en este caso particular, se conoce que hay más versiones de esta interfaz, y como el PMOD utiliza I2S fue obvio que era más fácil procesar en tiempos cortos los datos digitales (paralelos) que en su forma nativa del ADC (serie), así se conceptualizo el módulo conversor I2S a AXI. Al observar que se tiene una comunicación esclavo maestro se ve que puede recurrirse a una cadena de modificaciones de los datos y dejar que sigan su curso desde que se reciben en el ADC hasta que se envían por el DAC, así que con esa información solo fue cuestión de buscar algoritmos para la modificación

y tratamiento del audio digital, así surgen el control de volumen, filtrado digital y el efecto "delay", a expensas de regirse por la sincronización de la interfaz AXI.

Este mismo flujo del proceso es valido para el desarrollo de cada bloque especifico, solo que la rapidez dependerá de la habilidad con la que se cuente para la verificación, calibración y optimización del proyecto en VHDL.

Resultados y discusión

El consumo de recursos en el proyecto es una parte importante del resultado, ya que es posible cuantificar que tanto podría crecer este mismo proyecto con el mismo hardware, así que se desglosaran los recursos utilizados que son mostrados en el reporte de utilización dentro de vivado, se muestra en la figura 8-1 la cantidad, tipo y porcentaje utilizado en el proyecto completo:

Recurso	Utilizado	Disponible	% Utilizado
LUT	954	20800	4.59 %
LUTRAM	174	9600	1.81 %
FF	1345	41600	3.23 %
BRAM	27	50	54.00 %
DSP	21	90	23.33 %
IO	21	106	19.81 %
BUFG	2	32	6.25 %
MMCM	1	5	20.00 %

Tabla 8-1: Recursos totales utilizados en Basys 3.

De estos recursos, se puede identificar el tipo de RAM utilizada en el apartado respectivo, se utilizan 6 BRAM18 y 24 BRAM36 que para el uso de TDP (True dual-port memory) la configuración puede dar como selección a 32k x 1, 16k x 2, 8k x 4, 4k x 9, 2k x 18 o 1k x 36 [40], el mayor uso de la memoria se tiene en la implementación del bloque de RAM en el bufer

circular, se tienen 24 bloques de BRAM36E1 que implementan el mapa de memoria de 32k x 24, el resto de la misma es ocupada para implementar los registros de almacenamiento en los datos de los módulos de volumen y conversión de I2C a AXI, en la parte de implementación con DSP se observa un uso de 21 de ellos, en los módulos de filtrado Fir_compiler se utilizan 3 de estos, tomando en total de 9 en este bloque, 1 DSP en el uso del módulo del contador para el indexado de direcciones, en el módulo de volumen se ocupan 10 DSPs y el último en el bloque del efecto "delay" para suma de la señal de entrada y la señal retardada. El consumo en potencia es esencial para visualizar el ahorro en potencia que intrínsecamente se obtiene al manejar un sistema SoC a diferencia de los ASIC, por lo que se puede obtener en el apartado de potencia las siguientes características para cada detalle de utilización en la tabla 8-2 :

Recursos	consumo	% consumo
Clocks	0.002 W	1 %
Signals	0.004 W	3 %
Logic	0.001 W	1 %
BRAM	0.022 W	16 %
DSP	0.004 W	3 %
MMCM	0.101 W	72 %
I/O	0.006 W	4 %
	0.140 W	100 %
Consumo dinamico	0.140 W	66 %
Consumo estatico	0.073 W	33 %
Total	0.213 W	100 %

Tabla 8-2: Consumo utilizado en cada recurso de Basys 3.

Como se observa en la tabla 8-2, es importante señalar que existe un consumo por defecto en el momento de alimentar el FPGA que se establece como consumo estático igual a 0.073 W, todos los consumos de los recursos se establecen como un consumo general dinámico puesto que funcionan en etapas que no implican el uso de todo al mismo tiempo, puede verificarse que este consumo dinámico tiene un recurso que predomina en la tabla al ocupar un 72 % que es MMCM (Mixed-Mode Clock Manager), el cual otorga la opción de cambiar la frecuencia, fase y ciclo de trabajo del reloj principal, seguido al consumo del 16 % que

es utilizado por los bloques de RAM y al final continua con los bloques de entrada/salida con un consumo de 0.006 W, los demás consumos son casi insignificantes, por lo que el recurso que es implementado y más costoso en potencia son los bloques de RAM. Queda por último establecer latencias, se necesita observar como es la rapidez de propagación de los datos a través de cada módulo, comenzamos con el módulo ADC, como se vio en el capitulo 3 tiene una frecuencia de muestreo de 44.1Khz por lo que en un segundo se tienen 44100 muestras, para el reloj maestro definido de 22.579 MHz axi_clk se tiene que cada 456 ciclos de reloj se obtienen dos datos de 24 bits aunque este esta contenido en un conteo máximo de 512 ciclos que marca el reinicio de la adquisición de un nuevo dato, entonces el máximo retardo no puede sobrepasar los 455 ciclos de reloj, ya que se comenzaría a disminuir la frecuencia de reconstrucción de la señal digital de salida analógica en el dispositivo correspondiente DAC, para los módulos Fir_compiler se tienen retardos entre 70 y 90 ciclos aproximadamente que pueden ser consultados en el apartado de suma de características en el uso del mismo, y para los módulos de volumen y "delay" comprende entre 3 y 7 ciclos de retraso en la propagación de la entrada y obtención de un dato, estos datos están listados en la tabla 8-3:

Módulo	Latencia medida en ciclos
AXI-S a I2S	456
AXI-S volume controller	3
Effect delay	7
Filter LP	91
Filter HP	76
Filter BP	89

Tabla 8-3: Latencia por módulo medida en ciclos de reloj del proyecto

Como puede verse, el retardo total máximo esta dado por la suma de los tres módulos, volume controller, effect delay y el módulo filter en condición de filtrado como pasa bajas o LP por sus siglas en ingles Low Pass, el cual suma $3 + 7 + 91 = 101$ ciclos de reloj que cumple con la condición de ser menor a los ciclos de termino de un primer dato.

Entonces, dadas las condiciones anteriores en utilización de recursos es posible ver que las características en almacenamiento son bajas en la tarjeta de desarrollo Basys 3 para la implementación de efectos que usen retardos, listados entre ellos están: Flanger, Phaser, Vibrato, Reverb, también tenemos efectos donde puede destacar el uso de FPGA con sus DSP embebidos como es el caso del Wah-Wah[32], aunque debería aumentarse la memoria y establecer además un proceso que tenga una latencia baja para evitar la reducción de la frecuencia de muestreo, este problema solo se encuentra en los sistemas digitales ya que en sistemas analógicos se trata la señal entera en un mismo instante.

La mejora que supone diseñar con sistemas digitales es que gracias a su bajo consumo, se pueden obtener los mismos resultados con dispositivos mucho más pequeños, ya que no necesitan un tamaño grande para disipar calor, también es posible separar el ruido que se genera en diferentes módulos y cuantificarlo con más precisión ya que generalmente este es producido en la conversión de analógico a digital, actualmente existen métodos para reducir este error en el proceso de conversión, en el proyecto se pudo observar el ADC Sigma-Delta que cambia de resolución en amplitud a tener resolución en frecuencia [1], haciendo que este error sea disminuido considerablemente, ya que normalmente el error es generado por la diferencia entre niveles de cuantificación y si se tiene un aumento de bits para cuantificar, mayor será el ruido introducido.

Se debe tener en consideración que al asignar un ancho de banda al proceso de muestreo pueden excluirse muchas frecuencias por debajo y por encima de las frecuencias audibles, y estas contienen a otras que son llamadas frecuencias armónicas, las cuales son múltiplos de las primeras, en un proceso donde se operan ondas complejas se pueden generar efectos particulares donde sumen o resten a otras componentes frecuenciales, entonces esta particularidad puede notarse en el audio analógico a diferencia del digital, haciendo que el primero contenga un audio más “natural” o con mayor contenido de frecuencias que sufrieron el proceso antes mencionado.

Una de las principales características al manipular datos de audio de manera digital y con una gran velocidad en modo paralelo es que pueden agregarse características importantes a estos datos aún digitales, es decir, es posible manejar múltiples canales en un paquete de datos,

en ellos pueden venir un conjunto de sonorizaciónes diferentes y por lo tanto resulta evidente la posibilidad de implementar sonidos en multicanal como lo son 2.1, 5.1, 7.1, donde el primer número indica la cantidad de parlantes en frecuencias medias y agudas y el segundo los parlantes con frecuencias bajas [18], este proceso esta contenido en la mayoría de grabaciones estéreo y en cualquier audio envolvente, pero ahora se ha desarrollado al campo de transmisión de tv digital y grabaciones de audio más complejas como en los Blue Ray donde hay codificación de audio sin perdidas [18].

Por último, se puede comparar el aspecto económico al diseñar con electrónica analógica o digital, se debe eliminar el costo del diseño conceptual, pues ambos tienen el grado de complejidad y un fin práctico muy similar, llegar a un circuito electrónico que modifique el audio en sus características básicas, eliminando esta variable se puede hacer más fácil esta comparación y con mayor objetividad, en el año en curso, 2020, la tarjeta de desarrollo Basys 3 de Digilent tiene un costo de 149 DLS, y el PMOD I2S2 un costo de 21.99 DLS lo que suma un aproximado de 171 DLS, para comparar el precio se tomará un valor de los efectos en un proyecto DIY (Do It Yourself), se considera esto pues será comparable por tener dispuesto el hardware aproximado para construir el diseño establecido, estos proyectos tienen un precio en Amazon de 40 DLS para el “delay”, 25 DLS para volumen y 30 DLS para un reductor de frecuencias bajas como podría ser la función de filtrado, suman entre los tres 95 DLS, se mencionaron algunos de los beneficios de cada sistema así como algunas desventajas, entonces se cuenta con las bases para poder hacer una conjetura y escoger la mejor solución dependiendo las necesidades requeridas.

Conclusiones

Se observó que con el método de diseño para un sistema digital que trata valores de largo de 24 bits los cuales conforman una señal de audio, se verifica la capacidad de los métodos Top-down y Bottom-up así como la mezcla de ambos para el desarrollo de un proyecto, también se estructura el código VHDL haciendo uso de entidad y arquitectura dentro del mismo para señalar además las interconexiones entre los módulos funcionales, se obtuvo un desarrollo amplio de la parte de adquisición y conversión de señal de audio puesto que ello dependerá la calidad base del audio.

Adicionalmente, la interfaz AXI es un método de solución para estandarizar los diseños modulares, con ello se vé que podriá solucionar mucho tiempo en el desarrollo, ya que cualquier método que se requiera puede comunicarse con esta interfaz sin siquiera saber como funciona internamente, como sucede en los IP cores, también es un excelente método de procesamiento de señales ya que reduce el tiempo de propagación si es usado con los DSP, así surgen bloques en los cuales se obtienen en la salida los datos procesados en la entrada a diferencia de sistemas seriales como I2S, aunque realmente uno es complemento del otro, ya que viendo estrictamente el tipo de control de las interfaces seriales y paralelas puede decirse que las interfaces paralelas son una suma de datos seriales. Por lo visto, la cantidad de bits en los métodos de muestreo de señal definen un mínimo de calidad al reproducir la señal original pero no define la calidad máxima que puede obtenerse, en el caso de los ADC Sigma-Delta se apuesta por la alta velocidad de muestreo o, como se mencionó, resolución en frecuencia para después utilizar un proceso de conformado de ruido de cuantización y un proceso de decimación para obtener el valor de frecuencia de Nyquist, adicionalmente se verificó la codificación binaria para el proyecto, que es el complemento a dos,

también signo y magnitud o binario natural, cada uno contiene características propias que los hacen ideales en usos específicos. Se observó la manera de implementar circuitos electrónicos síncronos con descripción de hardware, es posible describir por metodos combinacionales y secuenciales cualquier tipo de circuitos, se puede observar el uso de una máquina de estados finitos de Moore en el "delay" y circuitos combinacionales en el control de volumen, también debe tenerse en cuenta que, aunque los DSP tengan una respuesta casi inmediata (alrededor de 1 ciclo), se sumarán estos retrasos a la respuesta del módulo, se debe saber cuando utilizar registros para almacenar datos y cuando solo establecer códigos estructurales para delegar ese almacenamiento a módulos con jerarquías superiores o inferiores.

En cuanto al filtrado se observa que lleva un proceso matemático riguroso pero gracias a los bloques ya existentes puede hacerse uso de su funcionalidad sin necesidad de involucrarse a detalle, esto hace que se puedan utilizar mayores recursos con solo conocer la respuesta adecuada a su salida, es un gran apoyo para el conocimiento y experiencia en el diseño, ya que el tiempo del mismo se utiliza para crear módulos completamente nuevos. También se debe establecer la posibilidad de continuación de este proyecto, se tienen algunas características a corregir y otras para ampliar, en el caso de los módulos se debe establecer un rango de amplitud entre ellos por lo que un AGC (Automatic Gain Control) resulta imprescindible a futuro, también sería más fácil manipular las características del manejo de audio mediante una interfaz gráfica como puede ser un display o pantalla con algún menú donde se desplacen las posibilidades del circuito. Entonces es posible comparar a los ASIC con los FPGA en el audio para procesado del mismo que se observó en los módulos realizados, la especialidad de los FPGA es que se pueden establecer procesos concurrentes reduciendo tiempos y en los ASIC tienen un tiempo de fan in y fan out que siempre será existente entre ellos, además del tiempo de proceso interno, por lo que se establece una rapidez mayor en cantidad de procesos en el FPGA, en cuanto al valor económico de ambos, aún sigue siendo más barato obtener una solución con ASIC, aunque depende del nivel de integración de efectos implementados, ya que existen moduladores complejos donde el costo se eleva demasiado con respecto a FPGAs con características de muy alto rendimiento y costo inferior.

Bibliografía

[1] *Convertidores A/D de sobremuestreo usando técnicas de modulación Sigma-Delta: Conceptos básicos y Estado del Arte.* Instituto de Microelectrónica de Sevilla (2015) 19, 128

[2] Alan V. Oppenheim, Alan S. Willsky, S.H.N. *Señales y sistemas* (2008) 65, 67, 68

[3] ARM. AMBA Specification 2.0. Informe técnico, ARM (1999) 22

[4] ARM. AMBA 4 AXI4-Stream Protocol Version: 1.0 Specification. Informe técnico, ARM (2010) 23, 25, 26

[5] ARM. AMBA5 AXI and ACE Protocol Specification. Informe técnico, ARM (2020) 22, 43

[6] Bertalanffy, L.V. *TEORÍA GENERAL DE LOS SISTEMAS Fundamentos, desarrollo, aplicaciones.* George Braziller (1989) 1

[7] Brown, S. y Vranesic, Z. *Fundamentos de Lógica digital con diseño VHDL.* Mc GRAW-HILL/INTERAMERICANA EDITORES, S.A. DE C.V. (2006) xiii

[8] Candy, J. y Bell Laboratories Holmdel N.J. A Use of Limit Cycle Oscillations to Obtain Robust Analog-to-Digital Converters. *IEEE Transactions on Communications* (1974) 18

[9] Carpio, F.P. y Grau, J.A.B. *VHDL lenguaje para síntesis y modelado de circuitos.* Ra-Ma (2011) 58

[10] Casamayor, E.D.L. *Metodología de síntesis para uso de bloques DSP con HDL sobre FPGAs.* Proyecto Fin de Carrera, Universidad Complutense de Madrid (2011) 84, 88

[11] Castillo, J.A., Hidalgo, J.J.O., y Álvarez, I.V. Delta-Sigma Converter Processing: aplicación de herramienta de software libre para el análisis y caracterización de convertidores Delta-Sigma. *Pistas Educativas 112* (2015) 18

[12] Cirrus Logic, I. The 2-Channel Serial Audio Interface: A tutorial AN282. techreport, Cirrus Logic (2005) 40, 43

[13] Crespi, V., Galstyan, A., y Lerman, K. Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots* **24**(3):303–313 (2008) 2

[14] Cuadros, J.M.M. Especificación de sistemas digitales usando VHDL (2018). Universidad Complutense de Madrid 94

[15] Dante, I.M.C. filtro digital tipo FIR en un FPGA (2000). Tesis de grado xv

[16] Digilent. *Basys 3 FPGA Board Reference Manual* (2016) 51, 59

[17] Gomis, A.M. *Practicas de máquina sencilla con FPGA.* Escuela Técnica Superior de Ingenieria (2011) 8

[18] Guerra, O.J. Postproducción de audio con sonido envolvente 5.1 y codificación perceptual. *Universidad Autral de Chile* (2013) 129

[19] Instruments, T. *ADS126x 32-bit, Precision, 38-KSPS, Analog-to-Digital Converter (ADC) with Programmable Gain AMplifier (PGA) and Voltage Reference.* Texas Instruments (2015) 33

[20] Jara, J.A. y Maxinez, D.G. *VHDL El arte de programar sistemas digitales.* Compañia editorial continental (2004) xii

[21] Jayaraman, R. When will FPGA's kill ASIC´s? En *Xilinx Inc.*, pág. 17 (2001) xiii

[22] John Michael Espinosa Durán, Pedro P. Liévano Torres, Claudia P. Rentería Mejía, y Jaime Velasco Medina. DISEÑO DE UN MICROSISTEMA PROGRAMABLE PARA EFECTOS DE AUDIO DIGITAL USANDO FPGAS. *Revista EIA Escuela de Ingenieria de Antioquia* **11**(22):133,146 (2014) xv, 99

[23] Kumar, N.P..T.K. International Conference on Microwave, Optical and Communication Engineering (ICMOCE). En *Evaluation of bandwidth extension of telephony speech by data hiding in three languages* (2015) 82

[24] LOGIC, C. 10-Pin, 24-Bit, 192 kHz Stereo D/A Converter. Informe técnico, Cirrus Logic Inc. (2013) viii, 33

[25] LOGIC, C. 98 dB, 96 kHz, Multi-Bit Audio A/D Converter. Informe técnico, Cirrus Logic Inc. (2015) viii, 17, 32

[26] Luis Entrena, Celia López, M.G.y.E.S.M. Aritmética Binaria (2008). Universidad Carlos III de Madrid 94

[27] Mario Vera-Lizcano, Gustavo Vejarano, J.V.M. Diseño de funciones DSP usando VHDL y CPLDs-FPGAs. *Grupo de Bioelectronica y Nanoelectronica, EIEEE, Universidad del Valle* (2003) 84

[28] Marta Ruiz Costa jussá, H.D.B. *Diseño y análisis de filtros en procesamiento de audio*. Universitat Oberta de Catalunya, fuoc edición (2010) 60, 62, 65, 68, 72

[29] Masi, C.G. Hybrid approach to system design. *Control Engineering* (2009) 4, 6

[30] Ángel Pérez Rodríguez, I. ESTUDIO DE EFECTOS DE AUDIO PARA GUITARRA, E IMPLEMENTACIÓN MEDIANTE DSP. (2006). Proyecto fin de carrera xvi

[31] Ogata, K. *Sistemas de control en tiempo discreto*. 2ª edición. Pearson Education (1996) 64, 90

[32] Pedro P. Liévano Torres, John M. Espinosa-Durán, J.V.M. Implementación de algoritmos para efectos de audio digital con alta fidelidad usando hardware programable. *ing. Univ. Bogotá* (2013) 92, 128

[33] products, X. 7 series DSP48E1 Slice User Guide. techreport v1.10, Xilinx (2018) 85

[34] Products, X. Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics. techreport v1.25, Xilinx (2018) 86

[35] Semiconductors, P. I2S bus specification. Informe técnico, Philips Semiconductors (1996) 30

[36] Sharma, A. *Evaluation of AXI-Interfaces for Hardware Software Communication.* mathesis, TECHNISCHE UNIVERSITÄT CHEMNITZ (2018) 22

[37] Sánchez-Élez, M. *Introducción a la Programación en VHDL.* Facultad de Informática Universidad Complutense de Madrid (2014) 8

[38] Xilinx. *Vivado Design Suite User Guide Programming and Debugging*, all programmable edición (2014) 55, 58

[39] Xilinx. *FIR Compiler v7.2 LogiCORE IP Product Guide* (2015) 69, 70, 71, 75

[40] Xilinx. UltraScale Architecture Memory Resources. Informe técnico, Xilinx (2019) 125

[41] Young-Ha Hwang, Jun-Eun Park, y Deog-Kyoon Jeong. A compact 87.1-dB DR bandwidth-scalable delta-sigma modulator based on dynamic gain-bandwidth-boosting inverter for audio applications. *IEEE Asian Solid-State Circuits Conference* (2017) 17, 18

[42] Zolzer, U. *DAFX- Digital Audio Effects* (2002) 90, 91, 92

Apéndice

FPGA: Field Programmable Gate Array.

VHDL: Very High Speed Integrated Circuits Hardware Description Language.

AXI: Advanced eXtensible Interface.

AMBA: Advanced Microcontroller Bus Architecture.

I2S: Inter-integrated Sound.

Mux: Multiplexor, entidad electrónica básica para la interconexión de múltiples entradas a una única salida.

Demux: Demultiplexor, entidad electrónica básica para la interconexión de una entrada a múltiples salidas.

Fir: Finite Impulse Response o Respuesta Finita al Impulso.

ADC: Analog to Digital Converter o Conversor Analógico Digital.

DAC: Digital to Analog Converter o Conversor Digital Analógico.

Nyquist: Teorema de muestreo tambien llamado Nyquist-Shannon gracias a Harry Nyquist (1889-1976) Harlingen Texas USA, Físico e ingeniero sueco-estadounidense reconocido por contribuir a la teoría de la información.

FM: Frecuencia de Muestreo.

PMOD: Plug-in-modules basados en la Pmod Interface Specification.

THD+N: Total Harmonyc Distortion + Noise.

SoC: System On a Chip.

MSB: Most Significant Bit.

LSB: Least Significant Bit.