UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

Implementación de reconocimiento de género por voz en FPGA

Trabajo de titulación previo a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Autor: Aguilar Coronel, Manuel Alejandro

Velepucha Rodríguez, Marco René

Director: Barragán Guerrero, Diego Orlando.

LOJA

2022

2022

Aprobación del director del Trabajo de Titulación

Loja, 15 de diciembre de 2022

Doctor,
Francisco Sandoval
Director de la carrera de Ingeniería en Electrónica y Telecomunicaciones
Ciudad.-

De mi consideración:

Me permito comunicar que, en calidad de director del presente Trabajo de Titulación denominado: “Implementación de reconocimiento de género por voz en FPGA”, realizado por Manuel Alejandro Aguilar Coronel y Marco René Velepucha Rodríguez, ha sido orientado y revisado durante su ejecución, así mismo ha sido verificado a través de la herramienta de similitud académica institucional, y cuenta con un porcentaje de coincidencia aceptable. En virtud de ello, y por considerar que el mismo cumple con todos los parámetros establecidos por la Universidad, damos nuestra aprobación a fin de continuar con el proceso académico correspondiente.

Particular que comunico para los fines pertinentes.

Atentamente,

Diego Orlando Barragán Guerrero
C.I.: 1104252208
Correo electrónico: dobarragan@utpl.edu.ec

Declaración de autoría y cesión de derechos

"Nosotros, Manuel Alejandro Aguilar Coronel y Marco René Velepucha Rodríguez, declaramos y aceptamos en forma expresa lo siguiente:

Ser autores del Trabajo de Titulación denominado: "Implementación de reconocimiento de género por voz en FPGA", de la carrera de electrónica y telecomunicaciones, específicamente de los contenidos comprendidos en: Introducción, Capítulo 1. Marco Teórico: estado del arte y de la técnica, Capítulo 2. Simulación del Algoritmo, Capítulo 3. Implementación en FPGA, Capítulo 4. Análisis de resultados, Conclusiones y Recomendaciones, siendo Diego Orlando Barragán Guerrero, director del presente trabajo; también declaramos que la presente investigación no vulnera derechos de terceros ni utiliza fraudulentamente obras preexistentes. Además, ratificamos que las ideas, criterios, opiniones, procedimientos y resultados vertidos en el presente trabajo investigativo, son de nuestra exclusiva responsabilidad. Eximimos expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones judiciales o administrativas, con relación a la propiedad intelectual de este trabajo.

Que la presente obra, producto de nuestras actividades académicas y de investigación, forma parte del patrimonio de la Universidad Técnica Particular de Loja, de conformidad con el artículo 20, literal j), de la Ley Orgánica de Educación Superior; y, artículo 91 del Estatuto Orgánico de la UTPL, que establece: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad", en tal virtud, cedo a favor de la Universidad Técnica Particular de Loja la titularidad de los derechos patrimoniales que me corresponden en calidad de autor/a, de forma incondicional, completa, exclusiva y por todo el tiempo de su vigencia.

La Universidad Técnica Particular de Loja queda facultada para ingresar el presente trabajo al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, en cumplimiento del artículo 144 de la Ley Orgánica de Educación Superior.

.....

Autor: Manuel Alejandro Aguilar Coronel

C.I.: 1105099285

Correo electrónico: maaguilar2@utpl.edu.ec

.....

Autor: Marco René Velepucha Rodríguez

C.I.: 1104645708

Correo electrónico: mrvelepucha@utpl.edu.ec

Dedicatoria

Yo Manuel, dedico el presente trabajo de titulación a Lucía mi madre que siempre ha sido y será una fuente interminable de apoyo, amor y cuidado. A Manuel mi padre y a Fanny mi tía que partieron hace poco de este mundo, pero durante toda mi vida me llenaron de muchos conocimientos y enseñanzas que aún sigo utilizando en mi diario vivir, siempre estuvieron pendientes de mi vida, salud y desarrollo para salir adelante, este logro es siempre por y para ellos y mi madre.

Yo Marco, dedico este trabajo de titulación, en primer lugar, a Dios por permitirme terminar una de mis metas de vida, siendo un pilar fundamental y por brindarme fortaleza en cada proceso de mi vida. A mis padres, René y Gloria, quienes que, con su amor y apoyo, confiaron plenamente en mí dándome aliento a lo largo de mis estudios, gracias a su motivación, educación y sacrificio he logrado alcanzar mis metas y dar pelea hasta el final, sin duda haciendo posible este sueño realidad. A mis hermanos Christian e Israel, por brindarme su apoyo durante este proceso. Sin duda a mi amigo Roberto quien con sus palabras de aliento y consejos tampoco me dejó rendirme, estando siempre para mí. Finalmente, a María, quien me dio todo su apoyo y amor durante toda mi trayectoria dentro de la carrera sin dejarme desfallecer en mis estudios. Finalmente, mis amigos y compañeros, con quienes con por su apoyo durante toda mi carrera, no me dejaron rendirme aportando en mi formación personal y profesional.

Agradecimiento

Yo Manuel, agradezco principalmente a Manuel mi papá, que me enseñó desde pequeño el mundo de la electrónica y la electricidad, y a Lucía mi madre que me ayuda todos los días de mi vida a ser mejor y seguir en la lucha de convertirme primeramente en una buena persona y luego en un buen profesional. A mi director del trabajo de fin de titulación Ing. Diego Barragán quien confió en mis habilidades y me apoyó con las directrices adecuadas para lograr solventar cualquier duda durante la elaboración del presente trabajo. A mis amigos de siempre Esteban, Augusto, Tyron y Bryan. Mi más sincero agradecimiento a mis compañeros Ariana, Hermes y Karen por ser personas valiosas que con su amistad y apoyo me permitieron seguir adelante.

Yo Marco, agradezco a Dios por permitirme culminar una meta más en mi vida. A mis padres, por acompañarme estando siempre a mi lado en los días y noches más difíciles durante mis horas de estudió motivandome siempre a no rendirme, dándome la absoluta confianza de lograr lo que me proponga, siempre orientándome con valores y enseñanzas inculcados en lo largo de mi vida, estoy infinitamente agradecido por todo. Un agradecimiento especial al Ing. Diego Orlando Barragán Guerrero, por ser guía y demostrarnos que con un poco de perseverancia y paciencia se puede lograr lo que nos proponemos, por sus enseñanzas en el desarrollo del presente trabajo de titulación, brindándonos todo lo necesario y no dudar en ayudarnos cuando lo necesitamos. A mi familia quienes me apoyaron a cumplir uno de mis metas personales y me extendieron su mano.

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

El reconocimiento de voz por género es uno de los primeros pasos para muchos aplicativos ya sea de bioseguridad o domótica. El trabajo de titulación abarca el estado del arte necesario para la aplicación de técnicas y filtros que se pueden utilizar para el reconocimiento de voz por género, enfocándose en las personas cisgénero. En la literatura se encuentran múltiples soluciones mediante software. Es por esta razón que se propone una implementación en hardware reconfigurable, como puede ser en este caso, una FPGA. Bajo este contexto, el presente trabajo de titulación se lo realiza con el objetivo de implementar un sistema capaz de reconocer el género de las personas por medio de voz. El sistema implementado se realiza tanto en el software de programación Python y C, para posteriormente ser implementarlo en una FPGA.

Palabras claves: bioseguridad, domótica, filtros, FPGA.

.

Abstract

Gender-based speech recognition is one of the first steps for many applications, be it biosecurity or home automation. The degree work covers the state of the art needed for the application of techniques and filters that can be used for gender-based speech recognition, focusing on cisgender people. Multiple software solutions can be found in the literature. It is for this reason that an implementation in reconfigurable hardware is proposed, as it can be in this case, an FPGA. In this context, the present work is carried out with the objective of implementing a system capable of recognizing the gender of people by voice. The implemented system is made both in Python and C programming software, to be later implemented in an FPGA.

Keywords: biosecurity, home automation, filters, FPGA.

Índice de abreviaturas

UTPL	Universidad Técnica Particular de Loja
DCCE	Departamento de Ciencias de la Computación y Electrónica
FPGA	Field Programmable Gate Array
SRAM	Static Random Access Memory
MCU	Microcontroller Unit
RAPT	Robust Algorithm for Pitch Tracking
MIMO	Multiple-input multiple-output
GUI	Graphic User Interface

Objetivos

Objetivo General

Implementar el reconocimiento de género por voz.

Objetivos Específicos

- Investigar las técnicas de procesamiento de voz orientadas al reconocimiento de género.
- Diseñar una simulación en Python que procese un audio de voz humana y reconozca el género.
- Implementar la técnica simulada en FPGA a través de VDHL.
- Validar la implementación computando la tasa de aciertos.

Introducción

El reconocimiento de voz por género tiene variedad de propuestas implementadas por medio de un computador a través de software. Sin embargo, para aplicaciones puntuales como técnicas de reconocimiento biométrico surge la necesidad de implementar en dispositivos de hardware reconfigurable. En el caso del presente trabajo de titulación se plantea la implementación del reconocimiento antes mencionado en una FPGA. Para este fin, primero se realizó una simulación dentro de Python que procese un audio de voz humana y reconozca el género. Posteriormente se realiza el mapeo del código a un lenguaje capaz de funcionar en el FPGA y la validación del sistema implementado se la realiza por un computado a la tasa de aciertos.

En la metodología que se siguió, primero se realiza una revisión del estado del arte, realizando un énfasis en las técnicas de procesamiento de voz orientadas al reconocimiento de género y un reconocimiento de la tarjeta FPGA con la que se va a trabajar con los respectivos módulos a utilizar. Se realiza la implementación y verificación del algoritmo en la tarjeta, usamos un algoritmo de tasa de aciertos como herramienta en primera instancia. Como punto final se realiza una depuración y corrección de errores en base a la etapa de validación y una evaluación final del sistema.

El trabajo de titulación se divide en cuatro capítulos. El primer capítulo se realiza todo lo correspondiente al estado del arte, introducción a FPGA, frecuencia fundamental de la voz humana y procesamiento de señales. En el capítulo dos se realiza la simulación del algoritmo en el software de Python, una explicación de las librerías externas utilizadas en el código, su funcionamiento, la detección y corrección de errores. Dentro del capítulo tres tenemos la implementación del FPGA, el método utilizado para realizar el grabado de voz, como se realiza el procesamiento de la señal grabada, los filtros utilizados, el método de la autocorrelación, proceso de transcripción de Python a C, desventajas y ventajas, simulación tanto de hardware como de software, alcance de implementación y diferenciación entre software de Xilinx. Y por último en el capítulo cuatro contamos con el análisis de resultados donde se realiza un análisis técnico con las pruebas obtenidas y realizadas.

La importancia de la investigación radica en que es el inicio del reconocimiento biométrico de la voz, donde puede ser utilizada para fines de la domótica y sistemas de seguridad, entre otros usos.

Capítulo uno

Marco Teórico: estado del arte

1.1 Introducción a FPGA

El desarrollo de los proyectos electrónicos muchas de las veces se encuentran limitado debido a que se genera un único hardware, lo cual implica que los proyectos cumplan una sola funcionalidad. Desde hace más de treinta años fueron creadas las FPGA (Field Programmable Gate Array), las cuales se diferencian de las demás tecnologías porque permiten realizar la implementación reconfigurable de sistemas electrónicos. Estas opciones pueden ser: MCU (Microcontroller Unit), ASIC(Application Specific Integrated Circuit), DSP (Digital Signal Processor), etc. El FPGA actualmente se componen de varios módulos o bloques de tipos lógicos y matemáticos, periféricos de entrada y salida, etc. Dichos bloques pueden ser reconfigurados según sea la aplicación. Los bloques pueden ser interconectados y los conmutadores de enrutamiento existentes dentro del FPGA son controlados por millones de celdas de memoria estática de acceso aleatorio SRAM (Static Random Access Memory) (Boutros & Betz, 2021).

1.1.1 Ventajas de uso de una FPGA

Las ventajas de implementar proyectos con FPGA radica en lo versátil y flexible que se puede volver este dispositivo a la hora de realizar un circuito electrónico. Esto se debe a que es posible programar la lógica que se ejecutará entre los distintos bloques que contiene cada FPGA. También la interconexión que se puede realizar entre los mismos y el resultado que se desea obtener. A continuación, se detallas características adicionales.

Múltiples aplicaciones

La característica clave del FPGA es la reprogramación de hardware. Permite que se pueda tener un sinnúmero de aplicaciones en las siguientes ramas: procesamiento de imágenes y video, data center, procesos industriales, comunicaciones inalámbricas, etc. (XILINX, 2022).

Rendimiento

El FPGA es una de las tecnologías de hardware reconfigurable más aclamadas ya que lo que se desea lograr dentro de un diseño electrónico es aumentar la carga de datos o de información que se procese, esto se debe a que en una FPGA la lógica programable de cada bloque funciona de forma paralela, es decir, todas las funciones se desarrollan simultáneamente lo cual permite acelerar notablemente los procesos a diferencia de las demás tecnologías. Dicho esto, se puede decir que es una de las más utilizadas para procesos en donde sea necesario obtener una aceleración computacional debido a la cantidad de procesos o datos que se deben tratar (Xu et al., 2022).

1.1.2 Procesamiento de señales

Uno de los beneficios del procesamiento de señales es que se puede convertir una señal analógica a digital para poder trabajar con la información obtenida de manera más eficiente en cuanto a memoria y velocidad de procesamiento. Por ejemplo, en el campo de la medicina el procesamiento de imágenes ha avanzado notablemente permitiendo emitir nuevos criterios o diagnósticos. Asimismo, en el ámbito de las comunicaciones modernas muchos de los sistemas que se implementan son sistemas MIMO (Multiple Input-Multiple Output), sistemas basados en procesamiento digital de la señal. Otras de las aplicaciones de procesamiento de señales son los asistentes virtuales controlados por voz, los cuales a partir de un cierto comando de voz que ejecutan determinadas rutinas de control. Existen varias señales las cuales se pueden convertir de analógicas en digitales, en este caso la voz, para realizar un procesamiento exclusivamente digital a través de un circuito dedicado que se conoce como DSP (digital signal processor). Para tal fin, es necesario tomar la información de la señal de entrada, muestrearla con la tasa de Nyquist y, según sea la necesidad, convertirla a texto plano o simplemente obtener la cadena de bits que representa la muestra grabada y así poder procesarla (Vashistha et al., 2019).

1.1.3 Procesamiento de señales en FPGA

Las técnicas de procesamiento de voz se encuentran en constante implementación. Sin embargo, existen limitaciones en cuanto a la tecnología en la que se despliega dicho procesamiento. En el caso de los MCU (microcontroller) sus recursos no alcanzan a

interpretar de manera rápida y eficiente las instrucciones a ejecutarse. Es por esto que el FPGA permite mejorar el rendimiento al momento de procesar la señal ya que al tener co-procesadores, estos permiten disminuir la sobrecarga sobre el microprocesador al momento de realizar el procesamiento.

El FPGA, aparte de mejorar el rendimiento, también es reconfigurable ofreciendo distintas posibilidades para aplicaciones que se requiera al tratar señales digitales. En una plataforma de hardware con un FPGA de la familia Virtex-5 de Xillinx la implementación de filtros eliminadores de ruido han demostrado tener resultados aceptables y una simulación ideal del sistema de filtrado (Vega et al., 2019).

Existen ciertas herramientas que permiten realizar un procesamiento parcial de señales en lo que a audio se refiere. En el caso de Xilinx para la FPGA Genesys 2 existen dos herramientas que la comunidad refiere mucho en sus publicaciones, estas son: Sigma Studio(*SigmaStudio*® | *Dispositivos analógicos*, 2022) y GStreamer (*GStreamer: framework multimedia de código abierto*, 2022). El primero permite seleccionar el códec de audio que tiene el FPGA y crear una grabadora o reproductor de audio, analizar la muestra grabada y obtener una gráfica en función del tiempo de la muestra de voz. Sin embargo, no permite añadir más filtros y bloques matemáticos para obtener un valor en específico. GStreamer es una librería abierta en la cual los usuarios de Xilinx contribuyen a su desarrollo y mejoras. De dicha comunidad de usuarios también se obtienen varios proyectos y aplicaciones en forma de librerías para ser implementadas. Estas librerías se caracterizan por tener una variedad de grabadoras tanto reproductoras de audio como de vídeo. Sin embargo, tampoco es factible usar esta librería para la implementación de filtros o procesos matemáticos sobre la muestra de audio. Esto debido a que dichas librerías generan una interfaz gráfica la cual permite mezclar, grabar y filtrar la señal de entrada sin embargo la salida que se obtiene no es numérica es la muestra de audio ya procesada y lista para reproducirse, se pueden visualizar gráficas del audio en función del tiempo, pero no es posible extraer los datos completos para ser utilizada en el desarrollo del proceso de filtrado del presente proyecto.

1.2 Frecuencia fundamental de la voz humana

El primer paso para el procesamiento de voz es la detección de la frecuencia fundamental del habla. Si bien no existe actualmente un algoritmo definido que permita encontrar la frecuencia fundamental, los avances logrados hasta la fecha radican en un algoritmo que se desarrolla en torno a características principales del habla como el idioma. Según el idioma se tienen más o menos acentuaciones sobre ciertas letras o vocales lo cual genera que la frecuencia cambie. En la actualidad se han desarrollado varios métodos que permiten procesar la voz, y detectar la frecuencia fundamental los cuales son: análisis espectral, transformada de Hilbert Huang, el algoritmo RAPT (Robust Algorithm for Pitch Tracking),y el procesamiento de la señal usando la función de autocorrelación de la voz (Verde et al., 2018).

La voz es una onda que abarca una cantidad de parámetros que nos permite extraerla y caracterizarla. La característica más usada es la frecuencia fundamental (F_0) antes mencionada. Esta frecuencia es útil para identificar vibraciones de las cuerdas vocales siendo de gran utilidad para el reconocimiento de voz. Se ha determinado que la frecuencia fundamental de un hombre adulto esta entre los 85 y 180 Hz, en cuanto a la de una mujer va de los 165 a 255 Hz (Chuchuca & Esteban, 2022).

1.2.1 Géneros existentes

La Organización de Naciones Unidas (ONU, 2022), presenta un glosario de definiciones de los géneros existentes que demuestra que los seres humanos se expresan y se describen para así ellos mismos definir su identidad de género. Varios de los géneros presentados son: LGBT/LGBTI, Transgénero/Trans, Intersexuales, Hombre o Mujer, etc. La ONU expresa que la identidad de género en la mayoría de las veces se encuentra relacionada con el sexo que se le asignó al nacer. Sin embargo, existen personas que se identifican con otros géneros mencionados previamente independientemente de su sexo, como ejemplo, una persona que nació con sexo femenino pero su identidad de género es LGBTI.

Para el presente proyecto, el reconocimiento de género por voz se realizará a personas las cuales se identifiquen como CISGÉNERO. Este término hace referencia a las

personas que se identifican con el género que se les asignó al nacer, es decir si su sexo biológico al nacer fue de un hombre y se considera hombre o si su sexo biológico al nacer fue de mujer y se considera mujer. Cabe recalcar que como se menciona en (ONU, 2022), la identidad de género no está relacionada la orientación sexual de la persona.

1.2.2 Determinación de posibles casos de diferentes valores de frecuencia con respecto al valor establecido

Una vez determinado el género de personas con el que se va a realizar el presente trabajo, se debe tener en cuenta que en algunos casos existen hombres con un tono muy agudo de voz y a su vez también mujeres con un tono de voz muy grave. Es por eso que se propone realizar la prueba en al menos 100 personas de cada género para así poder obtener un valor de media de la frecuencia en ambos géneros. De acuerdo con (Paolini et al., 2018), en Argentina se realizó un estudio para obtener la frecuencia fundamental F_0 habla según el sexo y como resultado se obtuvo en el caso de los hombres una moda de 196 Hz y en mujeres una frecuencia de 98 Hz. La prueba fue realizada con 302 personas. Como se menciona en la investigación estos valores de frecuencia son para la población de la provincia de Córdoba en Argentina, demostrando que específicamente dichos valores presentados sirven para ese público ya que toma en cuenta acentos, formas típicas de hablar, etc. Por otro lado, los valores de la frecuencia fundamental no están definidos actualmente. En la recopilación de investigaciones relacionadas como la de (Paolini et al., 2018), la de (Rodero, 2001) en su tema de "*El tono de la voz masculina y femenina en los informativos radiofónicos: un análisis comparativo*" y como la de al trabajo de titulación los valores de frecuencia central se presentan como rangos de frecuencia los cuales se generan partiendo de una recopilación de varias muestras de hablantes de la zona que comparten mismo idioma, esto permite establecer una media del valor de frecuencia central según el idioma, las diferencias se presentan debido a las siguientes características del hablante: velocidad a la cual se emite la voz, la pronunciación y acentos, y en pocos casos el estado de ánimo del hablante.

1.3 Procesamiento de señales del habla

Las señales eléctricas son generadas a partir de la tensión que produce la onda acústica de la voz en la membrana del sensor o micrófono. Dichas señales de voz se representan con funciones matemáticas, donde su principal variable independiente es el tiempo. En general, el procesamiento de señales se enfoca en la representación, transformación y manipulación de las mismas. El procesamiento analógico fue el primer procesamiento electrónico que se desarrolló, el cual se determina mediante circuitos compuestos por resistores, capacitores, inductores, etc. En cambio, el procesamiento digital de señales se basa en el procesamiento de señales discretas en el tiempo o espacio, por lo que los valores de la señal se conocen en tiempos de dominios contables. Pero a pesar de ello, su amplitud es continua. Asimismo, el procesamiento digital de Señales convierte a una amplitud discreta, requisito para que la señal pueda ser procesada en un computador digital (Barchiesi, 2008).

1.3.1 Autocorrelación de una señal

Al realizar el procesamiento de señales existen varios algoritmos que se pueden emplear para una implementación en un dispositivo digital. Sin embargo, es necesario conocer que en el procesamiento de señales el algoritmo a usarse sea robusto. Este término hace referencia a que el algoritmo sea capaz de tratar adecuadamente las variables y no generar errores durante la ejecución del mismo, es decir la robustez del algoritmo implica que no se presenten fallos antes, durante y posteriormente al procesamiento de la señal. La autocorrelación de una señal es uno de los algoritmos más robustos ya que a diferencia de otros no se ve muy afectado por el ruido. Es por eso que no es implementado solo en el procesamiento de la voz sino también en procesamiento de imágenes (Lin & Shao, 2018).

La función de autocorrelación viene dada por la siguiente ecuación:

$$r_{xx}(t,k) = \sum_{j=0}^{N-1} [x(j) * w(j)] * [x(j+k) * w(j+k)] \quad (1)$$

En donde:

$x(j) =$ Señal de la voz.

$w(j) =$ Ventana de Hanning.

$j =$ Índice de muestra.

$t\ , k =$ Índices de retardo de trama y de autocorrelación.

1.3.2 Tipos de filtro para procesar la voz

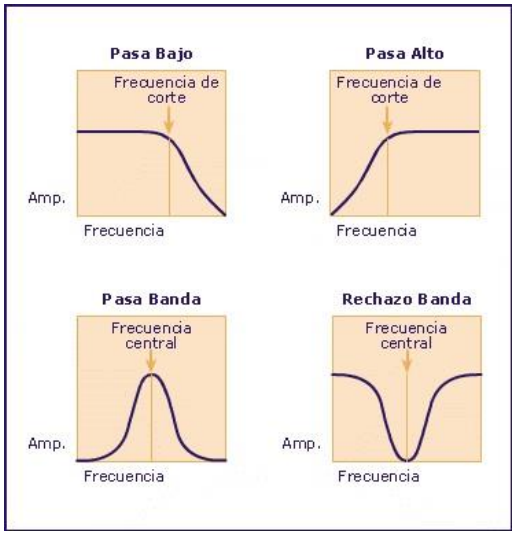
Tal como menciona (Castrosqui Florián & Castrosqui Florián, 2017), la aplicación de filtros se la realiza para obtener una versión más limpia de la voz grabada. Dado el caso de que grabamos sonidos que recogen muchas frecuencias y obtenemos un nivel de ruido que nos dificulta el tratamiento de la frecuencia de voz, al momento de utilizar un filtro se eliminan las frecuencias que no se encuentran dentro del rango que necesitamos.

Los filtros pueden ser tanto analógicos como digitales, pero en este caso nos enfocaremos en los filtros digitales dado que la señal con la que trabajamos es de naturaleza digital una vez haya pasado por el ADC del dispositivo FPGA (Ballesteros & Torres, 2018).

Un filtro digital tiene como entrada una secuencia discreta de valores y como salida produce una nueva secuencia de valores denominada señal filtrada. De acuerdo con la respuesta de frecuencia, los filtros pueden seleccionar que tipo de frecuencias se quieren alterar y cuales se van a bloquear, por lo que se determina cuatro tipos de filtros básicos como se indica en la figura 1:

Figura 1

Filtros básicos alteración de frecuencia



Nota. Adaptado de (Delgado, 2021)

El filtro paso bajo o corte alto donde pasan frecuencias que están por debajo de una determinada frecuencia eliminando frecuencias altas; filtro paso alto o corte bajo aquí pasan las frecuencias que están por encima de una determinada frecuencia eliminando frecuencias bajas; filtro pasa banda pasan las frecuencias que están en una determinada banda de frecuencia eliminando frecuencias fuera del rango y por último el filtro rechaza banda donde pasan la mayoría de las frecuencias sin alterar y atenúa las de un rango especificado bloqueando frecuencias de una banda determinada por dos frecuencias (Castrosqui Florián & Castrosqui Florián, 2017).

La respuesta al impulso que caracterizan a los filtros en el dominio del tiempo los clasifica de dos maneras: filtros de respuesta al impulso finita FIR (finite impulse response), que tiene como entrada una señal de impulsos que produce una salida de numero finito de términos y el filtro de respuesta al impulso infinita IIR (infinite impulse response), que producen una salida de números infinita (Ballesteros & Torres, 2018).

Capítulo dos

Simulación del Algoritmo

2.1 Introducción

En el procesamiento de voz se debe seleccionar el algoritmo con el que vamos a realizar dichas operaciones con las muestras del audio de la voz. En el presente trabajo se usará el algoritmo de la autocorrelación de una señal. El objetivo de seleccionar este algoritmo es con la finalidad de lograr obtener la frecuencia central de la señal procesada, para luego con dicho valor identificar dentro de que umbral de frecuencia se encuentra, para así reconocer si pertenece al género masculino o femenino.

2.2 Software y Hardware

La simulación del algoritmo de reconocimiento se realiza en Python y para el apartado de hardware se hace un diagrama de bloques como se indica en la Figura 2. Este diagrama con el fin de implementarse en la FPGA Genesys 2 Kintex-7.

2.2.1 Software

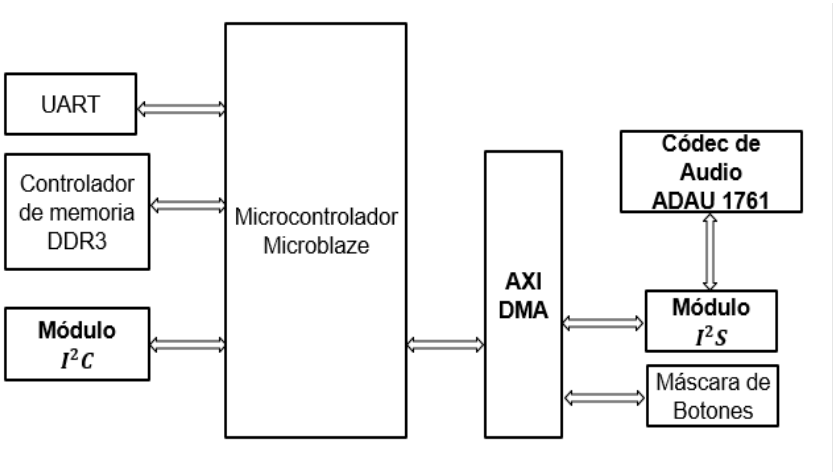
Para el desarrollo de la simulación se utiliza la versión 3.7 de Python con el IDE Spyder. Las librerías implementadas en esta simulación son las siguientes: tkinter, messagebox, sounddevice, numpy, math, scipy. La simulación consiste en la creación de una GUI (Graphic User Interface), la cual se compone de 3 botones los cuales cumplen la funcionalidad de grabar la muestra de audio, reproducir la muestra y finalmente identificar el género.

2.2.2 Hardware

El desarrollo de la simulación del hardware radica en la selección de los módulos de la FPGA que permiten implementar este proyecto. En este caso los módulos físicos de la FPGA Genesys 2 a usarse son los siguientes: Códec de audio ADAU1761, Microblaze (MCU), máscara de botones y los módulos lógicos a usarse son: módulos de comunicación UART, I2S, I2C, módulo de acceso directo de memoria o AXI

Figura 2

Diagrama de bloques de Hardware



Módulos lógicos y físicos forman parte del hardware de la FPGA.

2.3 Modelo para la adquisición de la señal de entrada

En el caso de la simulación, el modelo de captación de la señal de entrada, la librería sounddevice nos permite definir las variables que caracterizan al audio, las cuales son:

- Duración de la grabación.
- Frecuencia de muestreo.
- Tipo de dato de salida.
- Numero de muestras.
- Cantidad de canales que se usaran para la grabación.

Para la ejecución del siguiente script en Python se usará como medio de captación de entrada de la señal el micrófono de la computadora. La librería sounddevice usa como dispositivo de entrada el dispositivo que se encuentre definido como predeterminado dentro del sistema, si en caso se conecta cualquier otro periférico de entrada, lo seleccionamos en el panel de control de dispositivos de Windows y automáticamente ese periférico se convertirá en el dispositivo de entrada.

En el caso del número de muestras adquiridas viene dado por la multiplicación del tiempo de grabación por la frecuencia de muestreo. Uno de los parámetros a destacarse de la librería que usamos para capturar la muestra de audio es que permite definir el formato de

los datos de salida ya que los mismos pueden ser tipo float, int o en hexadecimal según sea necesario.

2.4 Procesamiento de los datos

Para la grabación de la muestra voz, se debe tomar en cuenta que la forma óptima de tratar los datos es mediante un arreglo de datos. Una vez almacenados los datos procedemos a llamar a la función *tono* la cual tiene como parámetros de entrada: la señal de audio y la frecuencia de muestreo de la cual se obtiene la frecuencia promedio como salida. Dentro de la ejecución de la función *tono* es en donde se realiza el cálculo de la autocorrelación de la señal de audio. Para un mejor entendimiento de la función *tono* se deben primeramente conocer los parámetros de entrada de la misma, las etapas que se desarrollan y el resultado final de la función. Como parámetros de entrada se tienen: cantidad de muestras, frecuencia de muestreo y la señal de audio muestreada.

Con los parámetros de entrada presentados es necesario conocer que esta función tiene varias etapas, las cuales son las siguientes: segmentación del audio, cálculo de la autocorrelación, determinación del valor de frecuencia central de cada uno de los segmentos a los cuales se los proceso con el algoritmo de autocorrelación, aplicar un filtro de mediana para lograr encontrar los valores menos afectados por el ruido.

Como parámetros de salida la función *tono* tiene la frecuencia central de la muestra de audio que fue procesada.

Es necesario que para el ingreso de los datos a la función *tono* se realice una identificación correcta de los parámetros de entrada ya que de estos depende el funcionamiento de todas las funciones y procesos que se desarrollan internamente como, por ejemplo: la segmentación del audio en partes más pequeñas, el algoritmo de autocorrelación, etc.

Del proceso de grabación de la muestra de audio, que tiene una duración 2 segundos y de la cual se obtienen aproximadamente 330750 muestras, y se realiza una segmentación del vector de datos que representa el audio muestreado. Logrando así que cada segmento se compone de 2646 datos. De cada una de las segmentaciones realizadas se obtiene las

frecuencias promedio de cada trama de datos y se realiza la indexación de estas frecuencias dentro de un vector del cual se procede a aplicar un filtro de mediana y seleccionamos las menos afectadas por el ruido.

2.5 Simulación en Python

La simulación consta de dos programas: el programa principal en el cual se crea la GUI y sus respectivos botones a los cuales se les asigna un nombre y la funcionalidad que van a cumplir. El segundo programa contiene las funciones de tono y de cálculo de la autocorrelación de la señal de audio. La función de autocorrelación tiene la siguiente lógica:

- Creación de coeficientes de un filtro pasa bajas de orden 4 a 900Hz: se usa estos coeficientes para filtrar digitalmente las muestras de audio con los coeficientes del filtro.
- Encontrar el nivel de recorte y centrar la onda recortada: Ya que se calcula la autocorrelación de la señal con varios segmentos de la señal total de audio se tiene que centrar la onda y trabajar sobre el vector de datos ya centrado teniendo en cuenta los niveles de recorte.
- Calcular la autocorrelación y encontrar el máximo valor que se encuentre entre 60Hz y 320Hz.
- Detectar la existencia de segmentos mudos: en el caso de los segmentos mudos se hace referencia a las partes de la muestra de audio en las cuales no hay audio y se procede a asignar como frecuencia central un valor de cero.

Para el caso de la función de *tono*, esta función está compuesta por las siguientes etapas:

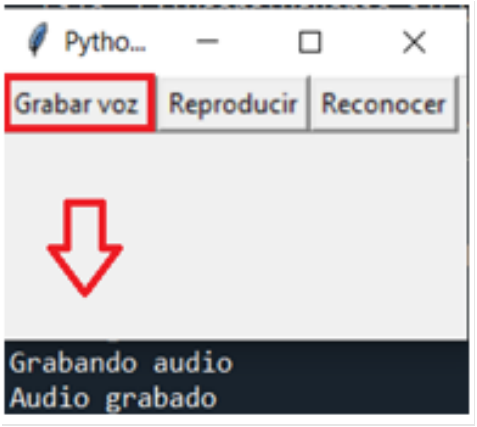
- Identificación de errores en la señal: remueve el nivel de DC existente en la muestra de audio.
- Segmentación de la muestra de audio: se procede a tomar segmentos de 30ms de la señal cada 20ms logrando así que el solapamiento existente entre cada uno de los segmentos sea de 10ms.
- Obtener el contorno o envolvente del tono.

- Procesar cada uno de los segmentos: de cada segmento de 30ms obtener la frecuencia central con la función de autocorrelación explicada anteriormente.
- Aplicar filtro de mediana: seleccionar las menos afectadas por el ruido.
- Obtener la frecuencia promedio de toda la muestra de audio.

Con el valor de la frecuencia central obtenido se procede a definir el umbral de frecuencia para las personas cisgénero masculinas y femeninas, para lo cual el umbral está definido de la siguiente forma: si la frecuencia central es mayor a 193.41 Hz la muestra de la voz pertenece a un hablante cisgénero femenino. Caso contrario la voz es de un hablante cisgénero masculino. El despliegue de la GUI para un mejor control del funcionamiento de la simulación, en el caso de la grabación se despliega un mensaje al iniciar la grabación y al finalizar la grabación, como se muestra en la Figura 3.

Figura 3

Grabación de la muestra de audio



Grabación de la muestra de audio y despliegue de mensajes por consola.

En la figura 4 se observa en el caso del accionamiento del botón RECONOCER, donde se despliega un mensaje de tipo caja de texto de aviso el cual permite visualizar el género del hablante.

Figura 4

Reconocimiento de género

Despliegue de mensaje en textbox

2.6 Detección y corrección de errores

Los errores detectados al momento de realizar la simulación radican en la incompatibilidad de las versiones de Python con la librería sounddevice. En este caso en primera instancia se seleccionó la versión 3.9 de Python, y la librería en esa versión no permitía definir el tipo de dato de salida.

Capítulo tres

Implementación en FPGA

3.1 Introducción

Para la implementación en FPGA se debe realizar un mapeo algorítmico con respecto al software de Python realizado. En el caso del dispositivo, Xilinx es necesario programar hardware y software de manera conjunta para lo cual las sentencias de código de software se realizan en lenguaje C usando el programa SDK de Xilinx y para el apartado de hardware se usa VIVADO, en donde se implementa el código demo de audio, el cual ya cuenta con los diagramas de bloques necesarios para la grabación de audio.

3.2 Adquisición de la señal de entrada

La grabación de voz la realizamos mediante la reutilización de un código demo de Xilinx que permite testear el funcionamiento del Códec de Audio ADAU1761. Este código lo que hace es activar los puertos de entrada y salida del códec de audio como son el: MIC IN, LINE IN, LINE OUT y el HP OUT. De la misma forma para la toma de la muestra de voz y para la reproducción del audio se usa, mediante una ventana de visualización a través de comunicación UART es posible informar al usuario el momento en el que se inicia y finaliza la grabación y la reproducción del audio respectivamente. El tiempo de grabación de audio es de 5 segundos, ya sea por la línea de entrada (LINE IN) o por micrófono (MIC IN). Una vez realizada la grabación lo que se procede es almacenar el audio dentro de la memoria RAM para según el botón seleccionado reproducir o grabar según sea el caso. Cabe recalcar como se mencionó previamente este código demo contiene la parte de desarrollo en Hardware y Software.

Como es conocido de cada FPGA que sale al mercado, siempre se presentan sus demos para cada uno de los módulos que contienen. Esto con el fin de tener una idea de cómo desarrollar proyectos y prácticas con dichos módulos y conocer las nuevas prestaciones de la FPGA.

Al ser un código demo del códec de audio del FPGA Genesys 2, es importante conocer que la arquitectura en hardware y software se encuentra ya desplegada y la finalidad de este

demo es demostrar la vinculación entre los módulos ya sean físicos o lógicos para lograr un funcionamiento óptimo. En este caso del códec de audio, permite realizar una grabación durante 5 segundos, almacenarla en la memoria del microcrontrolador y posteriormente reproducir la muestra de audio por las salidas que tiene el códec. El hardware permite realizar la grabación haciendo uso de un módulo físico el cual es el códec, que permite conectar un periférico de entrada y que sea posible capturar la muestra de audio.

Para el proceso de guardar la información capturada es necesario realizar una vinculación entre hardware y software ya que se debe realizar una asignación de un espacio de memoria libre en el microcontrolador Microblaze y una configuración de software que permita reescribir y leer sobre el espacio de memoria asignado ya sea para la reproducción de la muestra guardada o para una realizar una nueva grabación.

Para la elaboración del siguiente proyecto las únicas modificaciones a realizarse son en software debido a que no se va a implementar una nueva funcionalidad que implique módulos de hardware nuevos, ya que es necesario solo realizar grabación, reproducción y la identificación de género. Por lo tanto, el proceso de identificación de género consiste en la realización de un filtro que permita encontrar el valor de la frecuencia central de una muestra de audio e identificar a que rango de frecuencias pertenece ese valor. Debido a que la muestra de audio es almacenada en memoria se puede tratar esta información únicamente con software es por eso que, se realizará la programación del filtro en lenguaje C, que es el lenguaje en el cual se desarrolla el apartado de software.

3.3 Procesamiento de la señal grabada

Para el procesamiento de la señal grabada es necesario implementar las funciones utilizadas en Python y convertirlas a lenguaje C. En este caso, se debe conocer que en Python los datos que representan la señal de voz fueron indexados dentro de un vector y todas las funciones utilizadas utilizan dicha lógica. Con este antecedente, para iniciar el procesamiento de la señal, se deben conocer la frecuencia de muestreo y cantidad de muestras que se procesarán. Primeramente, la muestra de voz tomada se filtra para eliminar ruido presente en la señal, y luego se segmenta la muestra para ser procesada y encontrar mediante la

autocorrelación de cada segmento la frecuencia central promedio y finalmente encontrar la frecuencia central de toda la muestra seleccionada. Las muestras que se encuentren menos afectadas por el ruido y de las mismas encontrar estadísticamente la frecuencia central.

3.3.1 Parámetros de entrada para el procesamiento de la señal

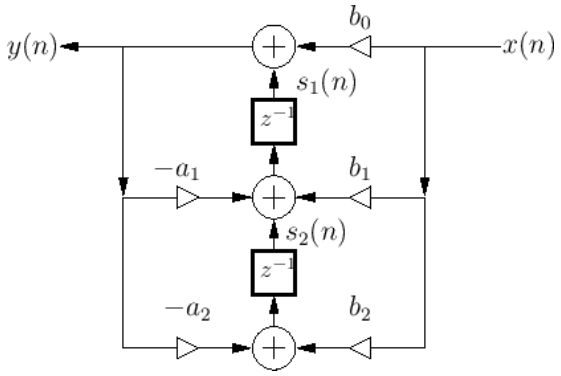
En el procesamiento digital de señales se realiza modificaciones específicas a la señal de entrada, ya sea para obtener un valor específico o modificar la frecuencia de la señal según sea la aplicación que se va a realizar. En el caso de presente proyecto, es necesario conocer dos parámetros importantes como la frecuencia de muestreo y la cantidad de muestras generadas. Estos parámetros de entrada servirán para tener un adecuado dimensionamiento de memoria para la creación de los vectores y el tratamiento que se le va a dar a la señal.

3.3.2 Filtros para procesamiento de señales

Una de las aplicaciones del procesamiento de señales radica en la implementación de filtros para obtener datos más puros de la señal a procesarse y eliminar ciertos elementos externos como el ruido que se presentan al momento de la captura de la muestra a procesarse. Para el procesamiento del audio, en un principio lo que se realiza es obtener los coeficientes de un filtro pasa bajas de orden 4 Butterworth con frecuencia de corte a 900 Hz. Estos coeficientes se obtienen desde el software Matlab en donde se hace uso de la función *butter.* Para el desarrollo del presente proyecto, la frecuencia de muestreo para la simulación e implementación está definida en 22.5 KHz. El filtrado de la muestra de voz con los coeficientes del filtro pasa bajas es la siguiente etapa de filtrado. Este filtro digital es de tipo *Direct Form II Transposed*, es decir, en este proceso se tienen como entrada los coeficientes tanto para el numerador como para el denominador del filtro pasa bajas y la muestra de la voz, obteniendo como salida la señal filtrada tal como se muestra en la figura 5, es decir a_1, a_2, b_1, b_2 representan los coeficientes del filtro con respecto a cuál se va a realizar el proceso de filtrado de los datos y z^{-1} representa el delay o retraso generado en cada proceso de filtrado con cada uno de los coeficientes.

Figura 5

Ejemplo Filtro Transposed Direct Form II con dos coeficientes



Nota. Adoptado de (III, 2007)

Para facilidad de interpretación $y(n)$ es la salida del filtro donde la señal ingresa por $x(n)$ y de los coeficientes: $b_0, b_1\ y\ b_2$ sale la señal validando la información que ingresa tomando un determinado rango de la señal para sacar las muestras necesarias y comprobarlas realizando una sumatoria de los valores correctos.

Este filtro fue seleccionado por dos características notables que lo destacan frente a otros filtros, las cuales son: menor uso de los recursos de memoria, y es capaz de no generar retraso en la generación de la salida del filtro.

La última etapa de filtrado es la implementación de un filtro de medianas. Para entender dicha etapa, este filtro es usado después de encontrar la frecuencia central de cada uno de los segmentos de audio que se van generando y con los cuales se calcula la frecuencia central. El filtro de medianas tiene como entrada el vector donde se encuentran las frecuencias centrales de cada uno de los segmentos. Primeramente, lo que realiza es seleccionar las frecuencias que se encuentren menos afectadas por el ruido, para luego de esas frecuencias calcular la mediana y encontrar finalmente la frecuencia central de toda la muestra de audio.

La implementación del filtrado dentro del procesamiento de la señal radica en que es necesario obtener valores lo más exactos para que el valor que se obtiene después de realizar el procesamiento sea eficiente, en otras palabras, tratar de obtener un cierto nivel alto de

fidelidad con los datos que vamos a tratar para obtener una respuesta correcta y minimizar el rango de errores.

3.3.3 Autocorrelación

Para un mejor entendimiento de la autocorrelación es necesario fundamentarse en la raíz de esta función y el motivo por el que es comúnmente implementada en varios sistemas de procesamiento de señales. La autocorrelación parte de la función de la correlación, la cual usa dos señales de entrada para producir una tercera conocida como correlación cruzada de las dos señales de entrada.

La correlación cruzada viene definida por la ecuación 2:

$$y(k) = \sum_{k=-\infty}^{k=+\infty} x(k) \bigotimes h(k) \quad (2)$$

En donde $y(k)$ que es la salida de la correlación cruzada entre dos señales diferentes, viene representada por la convolución entre la señal $x(k)$ y $h(k)$.

Por lo tanto, la ecuación que representa la autocorrelación es una correlación cruzada de la señal de entrada consigo mismo, pero retrasada, es decir, los parámetros de entrada de esta función son la señal, el retraso de la señal o *lag* y la misma señal retrasada. Como se expresa en la ecuación 3:

$$y(n) = \sum_{k=-\infty}^{k=+\infty} x(k) * x(k-l) \quad (3)$$

En la ecuación 3 la variable l viene representado por el retraso a aplicarse para el cálculo de autocorrelación. Asimismo, existen dos puntos importantes a la hora de calcular la autocorrelación, lo cuales son:

- Cuando el valor de retraso es igual a cero el resultado de la autocorrelación es la señal de entrada multiplicada por sí misma.
- Cuando el valor de retraso es negativo se calcula el valor absoluto de este para calcularlo con la ecuación 3.

La autocorrelación tiene una gran importancia en el procesamiento de señales, esto se debe a que es una función que no genera un gasto desmesurado de recursos tanto en implementación o desarrollo. Por lo cual los beneficios de implementar este algoritmo en el presente proyecto radican en el ahorro de procesamiento computacional que realizará el FPGA.

3.3.4 Proceso de transcripción de Python a C

En la introducción del presente capítulo se indicó que la programación del software será implementada en el lenguaje C, en donde se deben conocer las características del lenguaje como: precisión, tipos de variables, dimensionamiento de datos, creación de funciones y aprovechamiento adecuado de recursos.

Para realizar el mapeo de la simulación a la implementación es necesario realizar un análisis detallado de cada una de las funciones para entender la parte matemática de cada función y la salida de cada una de las mismas, ya sea si se devuelve un vector o un valor simple.

Como primera etapa del desarrollo se definieron los siguientes parámetros:

- Dimensionamiento de vectores: el dimensionamiento hace referencia a establecer la cantidad de elementos que tendrá el vector ya que normalmente en el lenguaje C si no se establece la cantidad de elementos la creación del vector se limita a almacenar únicamente 10000 muestras en caso de ser un vector con variables tipo double y en caso de que los datos sean de tipo float solo se almacenarán 8000 muestras.
- Establecer variables globales: estas variables son la frecuencia de muestreo, el número de muestras, vectores de coeficientes para numerador y denominador del filtro pasa bajas.

Para el dimensionamiento de los vectores es necesario conocer el número de muestras que se generarán a la hora de almacenar el audio, que en este caso serán 330750 muestras, y en el caso de la frecuencia de muestreo realizamos el proceso de asignar la frecuencia que se utilizó en el proceso de simulación que es de 22.5KHz.

Para la creación de los coeficientes del filtro pasa bajas de Butterworth a 900 Hz se obtienen los siguientes valores:

$$af_0 = [1, -2.4884, 2.0996, -0.5979]; bf_0 = [0.0017, 0.0050, 0.0050, 0.0017]$$

Con respecto al proceso para mapear cada una de las funciones de Python a C se debe de priorizar los contadores ya que estos dictaminarán los tamaños de los vectores a tratarse y en caso de no manipular bien estas variables se puede llegar a no tomar en cuenta ciertos valores que forman parte de un vector para realizar un dimensionamiento adecuado. Se definen globalmente las variables y se da un adecuado dimensionamiento de los vectores que contienen los datos, esto se lo realiza con la finalidad de no realizar un malloc sobre la memoria del computador o procesador en donde se implementará el algoritmo. Un malloc es la asignación que hacemos de memoria en C que utiliza dinámicamente un bloque de memoria con un tamaño especificado (Rishabh, 2018). Debido a que malloc es una función que nos permite liberar más espacio en memoria para crear vectores con cantidades de datos más grandes de lo habitualmente permitidas, implica un uso mayor de la memoria y que el vector se convierta en un puntero, teniendo así como resultado que los datos a procesarse se expresen en hexadecimal ya que esta función apunta a un registro de memoria que por ende se obtiene su registro de memoria y el valor que contiene.

Para el tratamiento de la información se ha definido que la información debe ser tratada como una variable de tipo double debido a las mejores prestaciones en lo que a precisión se refiere. Para la verificación de funcionamiento de cada una de las funciones implementadas en C se debe testear y realizar un DEBUG para depurar paso a paso y visualizar los valores que toma en cada paso cada una de las variables y luego al momento de unificar todas las funciones evitar errores.

Con respecto a la función de autocorrelación el proceso de implementación radica en la aplicación correcta de la fórmula, para lo cual a diferencia de Python se deben agregar el retraso en cada caso y tratar adecuadamente la señal.

3.4 Python y C

En esta sección se tratarán dos puntos claves para diferenciar la etapa de simulación y de implementación, ya que cabe recalcar que el lenguaje C lleva mucho tiempo en desarrollo a diferencia de Python, que un corto periodo de tiempo ha logrado realizar numerosas aplicaciones e introducirse en muchas implementaciones y crear una comunidad internacional que crea funciones para dar soporte entre cada uno de los proyectos a realizarse.

3.4.1 Desventajas

Al momento de la implementación y verificación de errores en el código realizado en lenguaje C se puede interpretar que una de las desventajas es la gestión de memoria, es decir, no se permite trabajar normalmente con vectores ya que primero es necesario localizar y sectorizar el espacio en memoria que ocuparán los datos.

Cada una de las variables a utilizarse en C deben ser declaradas línea a línea ya que es necesario para que el código pueda ejecutarse sin errores, sin embargo, los principales errores a la hora de crear un código en C son: la adecuada asignación de memoria a utilizarse y la incompatibilidad entre variables.

Al momento de suceder un error en C únicamente el compilador de C se encarga de presentar por medio de un mensaje la línea de texto en la cual se produjo el error, pero no demuestra el error exactamente, y también en C muchas de las veces no es posible obtener un visualizador de variables en el cual se puedan presentar todos los últimos valores de cada una y el tipo de variable, esto solo es posible si a la variable se la imprime recurrentemente por consola o realizando una depuración en la sección de código en la cual sea necesario realizar el análisis de la o las variables.

En el caso de Python, como es conocido, es un lenguaje interpretado por lo cual es más lento en procesamiento a diferencia de C, esto se debe a que Python al momento de ejecutar el código convierte el código en bytes para posteriormente ser ejecutado.

La depuración de Python es diferente ya que el intérprete depura durante la ejecución lo que implica que en este lenguaje no es posible visualizar un error antes de ejecutar el

código, es decir pueden existir errores que se generen durante la ejecución y no se visualicen si no hasta realizar una depuración completa del código.

Cabe recalcar que las desventajas que presentan cada uno de los lenguajes no implica que se tenga que seleccionar un lenguaje a preferencia que el otro en ciertas implementaciones. Las desventajas radican en cuestiones de facilidad del desarrollador, ya que en Python existen muchas librerías que vuelven más fácil y rápido la elaboración de varios tipos de implementaciones.

3.4.2 Ventajas

La principal ventaja de Python es que no es necesario asignar memoria para ciertos vectores de datos a tratarse. El intérprete de Python se encarga de ir asignando el recurso necesario según sea el tamaño de los datos, y no limitar al tamaño de cada tipo de dato, es decir la variable se crea según las características necesarias como: tamaño, cantidad de datos, precisión.

Asimismo, Python es más permisivo con respecto a determinados errores de programación, siendo más sencillo y brindando más facilidad a la hora de programar, respecto con C que los fallos que encuentra principalmente son de texto que pueden llegar a ser los punto y coma al final de cada línea pero no es posible verificar si una función que se está usando se encuentra ejecutando de forma correcta, siempre se debe realizar la construcción de cada proceso imprimiendo cada uno de los valores sucesivamente para identificar si el resultado obtenido se encuentra correcto.

3.5 Implementación

En el proceso de implementación se ha definido dos etapas para el diseño en hardware y software.

En el apartado de hardware se debe considerar la implementación del código demo provisto por Xilinx, el cual se encuentra en el sitio web oficial del FPGA Genesys 2. *https://digilent.com/reference/programmable-logic/genesys-2/start*

Este proyecto permite realizar las siguientes acciones:

1. Grabación de voz durante 5 segundos.

2. Reproducción de la grabación.
3. Almacenamiento de la muestra de audio en memoria DDR3.

Este hardware está conformado por varios módulos de la FPGA que permiten realizar las acciones previamente descritas, los cuales se describen a continuación:

1. Módulo de acceso a memoria AXI DMA: módulo que permite el acceso para escritura o lectura de la memoria física de la FPGA en la cual se almacenarán las muestras de audio.
2. Máscara de botones para accionamiento de las funciones: pertenecen a la sección de 6 botones de la FPGA, los cuales permitirán realizar la grabación y reproducción de la muestra de audio.
3. Módulo I^2C: módulo que permite realizar la interconexión entre cada uno de los circuitos a usarse en hardware que hace referencia a la vinculación de cada uno de los periféricos a seleccionarse con todos los circuitos necesarios para el desarrollo del proyecto.
4. Códec de audio ADAU1761: códec integrado a la FPGA el cual cuenta con dos entradas que son LINE IN y MIC IN, y las salidas de este códec son HP OUT Y LINE OUT.
5. Microblaze: procesador propio de la FPGA Genesys 2 de Xilinx, encargado de dar las sentencias de control y direccionamiento adecuado de los datos tanto como la gestión de registros y delimitar la cantidad de memoria permitida para guardar cada muestra de audio.
6. Módulo UART: este módulo permite que la FPGA se conecte por medio de cable USB a una computadora y se puedan enviar información por medio de los puertos COM de una computadora y visualizar por medio de una terminal los mensajes que se despliegan al ejecutar el programa.

Cabe recalcar que en la configuración del códec de audio es necesario agregar un módulo I^2S el cual es un bus serial estándar para realizar la conexión de códec con los demás módulos mencionados y poder enviar la data desde el códec hacia la memoria.

El apartado de software del código demo incluye los direccionamientos que debe tomar el procesador tanto para la escritura y lectura de la memoria, al momento de realizar una grabación o reproducción de la muestra de audio. También se incluye las sentencias de control para identificar que acción realiza cada uno de los botones. En el caso de la grabación se realiza un acceso de memoria para escribir el audio en la memoria DDR3 de la FPGA y en el caso de la reproducción se realiza primeramente un acceso a memoria para llamar el registro en donde se encuentra la muestra de audio y luego enviar dicha data al códec de audio para la reproducción.

Para la implementación del filtro sobre el audio ya guardado en la FPGA se ha realizado un post-procesamiento. Este proceso radica en la realización de las siguientes etapas:

1. Grabación de la muestra de audio.
2. Uso de comandos del módulo AXI DMA de Xilinx para lectura del registro de memoria base en donde se encuentra almacenada la data y posterior exportación a la PC del archivo en formato .txt el cual contiene la data en formato decimal.
3. Lectura del archivo con el código C.
4. Ingreso de la data al filtro, en donde se especifican datos de entrada como el número de muestras y frecuencia de muestreo.
5. Envío de resultado mediante UART a la consola del computador.

Para un mejor entendimiento de cómo es tratada la señal, al momento de grabarse y luego almacenarse en memoria, es necesario conocer la lógica implementada en el proyecto demo, por lo cual se deben conocer como cada uno de los módulos físicos se encuentran interconectados con los módulos lógicos. Esta interconexión es realizada para lograr un eficiente enrutamiento de la información, esto se debe a que, los módulos lógicos contienen ciertas funciones que facilitan el proceso de lectura y acceso a la información o la manipulación de la información según sea necesario.

La interconexión por realizarse entre módulos viene dada por registros de memoria, los cuales se presentan en la Tabla 1

Tabla 1

Módulos físicos y lógicos y sus respectivos registros asignados en el FPGA.

Módulos	**Registro base**
Códec de audio	0x44A00000
Módulo I^2C	0x40800000
Módulo AXI DMA (Acceso a memoria)	0x41E00000
Máscara de botones	0x40000000
Módulo UART	0x40600000
Microcontrolador Microblaze	0x00000000

Tabla de los módulos utilizados y sus respectivos registros en el FPGA.

En el caso del códec de audio existen registros internos que permiten realizar la configuración del códec de audio, como por ejemplo los registros del clock del códec, entradas y salidas a seleccionarse, activación de filtros, etc. Así mismo, el microcontrolador que es Microblaze (*MCU* fabricado por Xilinx), se tienen registros internos para poder acceder a la memoria DDR3.

En el proceso de grabación se obtiene un vector que contiene un total de 330750 muestras, y para realizar el procesamiento lo que se debe localizar dentro de la lógica del código demo son los siguientes parámetros:

1. Frecuencia de muestreo establecida en 22050 Hz.
2. Duración de la grabación de 3 segundos.
3. Establecer el total de número de muestras generadas.

Para el proceso de programación del filtro que se realizó en lenguaje C, se debe lograr establecer claramente los parámetros de entrada, el formato de dato de cada parámetro, ya que debido al lenguaje utilizado no es posible usar librerías específicas. Cada una de las

funciones implementadas en la etapa de simulación deben de ser mapeadas al código elaborado en C.

3.5.1 Simulación hardware

Durante el desarrollo de cualquier proyecto de FPGA, es necesario realizar una simulación previa del funcionamiento del hardware a implementarse posteriormente, que es más conocido como test bench. En el caso de Xilinx existen dos programas que permiten realizar este proceso los cuales son Vivado e ISE.

Para el test bench realizado, en la parte de filtrado del audio y asignación del género según el valor de su frecuencia, es necesario conocer que este filtro fue programado en lenguaje C en el software SDK de Xilinx. Esto implica que el proceso de filtrado de la señal no es realizado por un módulo físico de la FPGA. En el caso de Vivado comúnmente los test bench que se realizan en dicho software únicamente permiten visualizar los estados de cada señal de entrada o de salida, pero no es posible verificar como se realiza internamente paso a paso la lógica realizada por cada módulo que se analiza.

En la figura 6 se observa la simulación de hardware del proceso de reconocimiento de género es un comparador de dos bits entre el audio y la señal a filtrarse, esto se debe a que el proceso de filtrado es desarrollado en el apartado de software.

Figura 6

Simulación tipo test bench del funcionamiento del filtro.

Simulación del filtro para reconocimiento de género en Vivado.

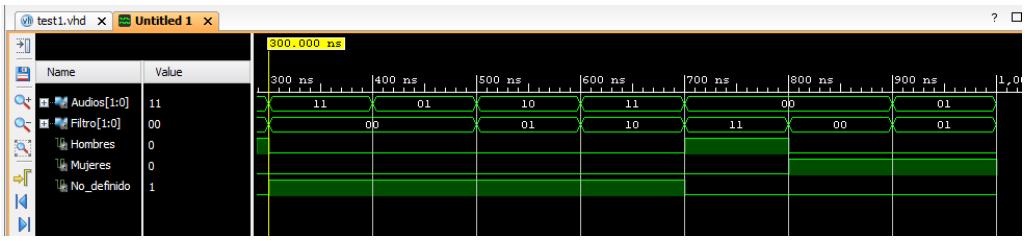
Se ha implementado un comparador de 2 bits ya que lo que es necesario es visualizar el estado de la señal que representa el género del hablante, por lo tanto, la señal de audio

que es una señal de entrada tiene varios posibles valores a lo largo de la simulación, y la señal de filtro también tendrá varios posibles valores, para lo cual se realiza la comparación entre estas dos señales. Como se explicó en el proceso de simulación existe una condición para clasificar la voz según el género, en este caso es el valor de la frecuencia central calculada. En el test bench se han definido 3 posibles clasificaciones: los hombres (personas cisgénero masculinas), las mujeres (personas cisgénero femeninas), y no definido.

Esta última condición no hace referencia a ningún género en específico sino más bien a cuando el sistema no detecta ningún audio o no le es posible realizar la operación de filtrado del audio, a continuación, se presenta en la figura 7 la comparación del audio filtrado.

Figura 7

Simulación en funcionamiento del filtro con varios tipos de audio en Vivado.



Simulación en ejecución del filtro con varios tipos de audio e identificación del género según corresponda.

En la figura 7 la señal *Audios* vendría a ser la muestra grabada y la señal *Filtro* el valor de frecuencia central que diferencia el género del hablante, este proceso de comparación entre el filtro y el audio, tiene como resultado la activación de señales internas que realizarán el proceso de enviar el resultado por consola. Es decir, en el proceso de simulación el comparador detecta que el audio pertenece a una persona cisgénero masculina el estado de la señal *Hombre* cambia a alto cuando se cumpla la condición y así sucesivamente para el caso de una mujer hablante.

3.5.2 Simulación software

La simulación de software radica en el entendimiento del funcionamiento del filtro, es decir, todas las etapas realizadas y el resultado que tiene al procesar el audio.

Para el proceso de verificación de mapeo de cada una de las funciones realizadas en Python, en primera instancia se realizó la toma de una muestra de audio haciendo uso de la librería sounddevice y se guardó los datos en un archivo *.txt* y cada muestra de audio contiene 6 decimales.

Las etapas a realizarse en el filtrado de la muestra de audio radican en ir eliminando valores de ruido que afectan el procesamiento y el resultado final que es la frecuencia central. Las etapas de filtrado se presentan con el siguiente orden:

1. Recortar la señal y eliminar niveles de ruido.
2. Segmentar la señal cada 30 milisegundos.
3. Analizar cada uno de los segmentos.
    - 3.1. Filtrar la señal con los coeficientes de un filtro pasabajas Butterworth de orden 4 a 900Hz.
    - 3.2. Encontrar el nivel de recorte de la señal.
    - 3.3. Centrar la onda
    - 3.4. Aplicar autocorrelación.
    - 3.5. Encontrar el máximo de autocorrelación entre 60 Hz y 320 Hz.
4. Aplicar filtro de medianas a los valores de frecuencia central calculados, es decir seleccionar las menos afectadas por el ruido.
5. Comparar el valor de la frecuencia central promedio con el umbral de frecuencia central que es 193 Hz e identificar el género.

Con las etapas de filtrado expuestas es necesario conocer los parámetros de entrada que se deben definir en el programa del filtro de forma global los cuales son la frecuencia de muestreo que es de 22.05 kHz y la cantidad de muestras del audio que son un total de 330750.

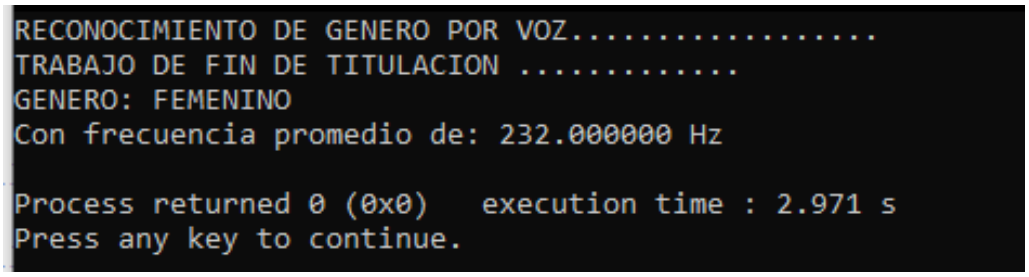
Los formatos de las variables implementadas en el filtro son de tipo double. Esto se debe a que este tipo de variable nos permite tratar con una cantidad más exacta de decimales, ya que al tener varias etapas en el filtro siempre se realizarán varias operaciones en cada

etapa, así mismo con la salida del filtro que es el valor de la frecuencia central promedio del audio también esta variable es de tipo double.

Al ejecutar el filtro haciendo la lectura sobre un archivo txt que contiene un audio de 3 segundos el cual tiene 330750 muestras, el resultado es presentado en la figura 9.

Figura 8

Despliegue por consola del resultado usando Xilinx SDK.



Simulación de software y presentación del resultado final en SDK.

Como se observa en la figura 9, se tiene el género identificado del hablante y el valor de la frecuencia promedio.

3.6 Alcances de la implementación

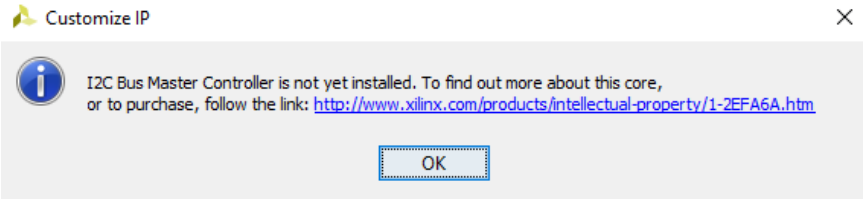
Para la realización de la implementación, se debe tener el contexto de que en la actualidad para el desarrollo de los proyectos en FPGA se tiene dos softwares que Xilinx ha proporcionado a toda su comunidad para la elaboración de proyectos. Estos son Vivado y SDK. En el primero se puede realizar todo el enrutamiento de cada uno de los módulos a implementarse y permite realizar programación en lenguajes como VHDL y Verilog. En el software SDK se realiza únicamente la programación de la lógica que va a realizar cada uno de los módulos y también la asignación de los recursos de memoria a utilizarse.

Existen varias formas de realizar un proyecto en FPGA, pero la que se destaca frente a las demás es Vivado que nos ayuda con la configuración de todo el hardware que compone al proyecto y realizar el enrutamiento de cada uno de los módulos a utilizarse para luego exportar el proyecto a SDK para así desarrollar las funciones lógicas que van a componer cada uno de los módulos en el momento de su ejecución.

En los proyectos de FPGA en el apartado de software y hardware, es necesario conocer que actualmente Xilinx ha desplegado un sistema de subscripción para adquirir licencias dependiendo del campo de desarrollo de un proyecto, en donde al realizar la compra de la licencia es permitido usar ciertos módulos de la FPGA durante un determinado tiempo. Al tener disponible la licencia para el FPGA Genesys 2 de Xilinx, dicha licencia tiene varias limitantes las cuales no permiten incorporar o editar la lógica que tienen ciertos módulos, como lo que sucede al momento de editar el módulo I2C implementado en el código demo, como se muestra en la siguiente imagen.

Figura 9

Mensaje de alerta de licencia para selección de un nuevo módulo en Vivado.



Limitantes de la licencia a la hora de editar el código demo.

Actualmente Xilinx tiene en venta cada uno de sus módulos para ser implementados en Vivado, independientemente del FPGA seleccionado. Los módulos a comprase tienen precios que dependen de si el módulo cuenta con la facilidad de ser simulable o no, y también muchos de los recursos que ofrece cada módulo son limitados de acuerdo con el tipo de licencia seleccionado. El FPGA Genesys2 fue lanzado a finales de 2015 y cada uno de sus módulos tuvieron soporte y actualización hasta mediados de 2017, es por esto que no es posible realizar la compra de una licencia que permita añadir las funcionalidades para lograr la finalidad deseada, sin embargo, para otras FPGA que se encuentran aún dentro de los dispositivos que reciben soporte y actualización, el costo de esta licencia podría llegar a ser de tres mil dólares con una duración de aproximadamente de 3 meses como lo presenta Xilinx en su sitio web: *https://www.xilinx.com/products/design-tools/vivado/vivado-ml-buy.html#whats-new*. Cabe recalcar que, los módulos que se incluyen en la licencia no permiten realizar simulación de hardware.

Los módulos que se encuentran en el FPGA Genesys 2 disponibles con la licencia que se cuenta actualmente son muy limitados, los cuales son: HMDI, CORDIC, CLK, VGA, Microblaze (Microcontrolador de la FPGA).

3.7 Diferencias entre los softwares de Xilinx

Actualmente se encuentran en vigencia Vivado y SDK, pero años atrás la mayoría de los proyectos realizados en FPGA se creaban a partir de la programación en VHDL usando el software ISE. Dicho programa permitía realizar un test bench que es una simulación de funcionamiento del proyecto y en el cual también era posible visualizar en test bench la simulación de la lógica a realizarse.

ISE es un software que contiene herramientas que facilitan la síntesis, validación por simulación y vinculación con plataformas o vinculación con herramientas auxiliares (Cuadros, 2015).

En cambio, Vivado es un software que permite crear hardware de manera más amigable para el programador, ya que su modo de operación es el de seleccionar el módulo e ir realizando el enrutamiento entre módulos, también es posible realizar síntesis del proyecto, cargar el archivo bitstream a la FPGA y permite realizar un debug paso a paso. En este software es posible realizar la programación de la parte lógica del proyecto, pero de igual forma se realiza en lenguaje C o C++, teniendo en cuenta que las asignaciones de cada uno de los pines del hardware se realizan en VHDL o en Verilog.

SDK es un software que se añadió para separar el proceso de diseño de hardware con el diseño de software, permite realizar todo el desarrollo en lenguaje C o C++ y es posible crear un proyecto en SDK a partir de un diseño previo en Vivado. Así, Vivado permite la creación de hardware y el enrutamiento de este y SDK crea la parte lógica de cada uno de los módulos de hardware a ejecutarse en el FPGA.

Una de las diferencias más puntuales y críticas entre Vivado e ISE es primeramente que Vivado es un software actualizado hasta 2022 el cual se ha adaptado al actual proceso de desarrollo de los proyectos en FPGA. En este software es posible realizar el test bench pero únicamente para visualizar los estados de cada uno de los componentes de entrada o

salida no es posible realizar una simulación que represente la ejecución de la lógica del proyecto, a diferencia de ISE en el cual es posible programar en VHDL declarando cada uno de los módulos que tendrán cada uno de los mismos, en la etapa de simulación también permite realizar un test bench que permite identificar estados del hardware, además de presentar los resultados y valores de cada una de las variables implicadas en el proyecto.

3.8 Limitantes de los lenguajes de programación

En la sección 3.7 se han mencionado los 3 softwares que ha creado Xilinx para que su comunidad realice diferentes tipos de proyectos en FPGA, los cuales son: Vivado, ISE, SDK. Cronológicamente hablando el primer software de Xilinx fue ISE brindando una infinidad de herramientas para consolidar proyectos de FPGA, haciendo uso únicamente del lenguaje de programación VHDL. Luego Xilinx decidió dar el paso y mejorar el entorno de desarrollo para hardware lanzando el software Vivado del cual se puede realizar el diseño de hardware con el uso de módulos y realizar la asignación de los pines de todo el hardware usando dos lenguajes de programación los cuales son: VHDL y Verilog. De este programa se desprende SDK ya que Vivado cuenta con una opción que permite exportar el hardware del proyecto al software SDK para realizar la lógica de programación, en esta etapa el desarrollo del proyecto se realiza en lenguaje C o C++.

Xilinx ha realizado una incorporación de varios softwares como se ha mencionado previamente, pero esto se debe a las limitantes que inicialmente tuvo el lenguaje VHDL ya que en este lenguaje se realizaban la totalidad de varios proyectos anteriormente.

VHDL es un lenguaje de programación que permite describir circuitos electrónicos digitales, es decir cuando se realiza la programación en este lenguaje el diseñador o programador debe de tener claro el funcionamiento de puertas lógicas y también identificar las partes del circuito que son combinacionales y cuales secuenciales (Chacon et al., s. f.).

Como es notable el avance que ha realizado Xilinx ha sido con beneficio para su comunidad de programadores, debido a que en un inicio solo se utilizaba VHDL sin embargo la parte de desarrollo de software no se realizaba de una forma muy eficiente, es por lo cual se vinculó C o C++ permitiendo desarrollar la parte de software de un proyecto, como bien es

conocido VHDL al ser un lenguaje que permite describir circuitos digitales, no se usan variables, se manejan estados. Sin embargo, el lenguaje de programación C ha permitido la creación de ciertos paquetes de funciones para todos los módulos existentes facilitando la realización del apartado de software de un proyecto.

Capítulo cuatro

Análisis de resultados

En este capítulo se presentan los resultados técnicos del sistema de reconocimiento de genero por voz implementado en el algoritmo de Python y con el código en lenguaje C obteniendo el rendimiento de las pruebas realizadas en el trabajo de titulación.

4.1 Análisis técnico

El análisis técnico hace referencia al correcto funcionamiento del algoritmo implementado. Para ello, hemos realizado una recopilación de 61 muestras de audio, para confirmar el correcto funcionamiento, dándonos como resultado 25 muestras de audio de personas cisgénero masculinas y un resultado de 34 muestras de audio de personas cisgénero femeninas, y obteniendo dos falsos negativos debido a que los audios tenían bastante ruido que no permitían identificar de manera clara el género. Dichos resultados se presentan en la tabla 2.

Tabla 2

Resultados de audios procesados

Audios procesados	61
Personas cisgénero masculinas detectadas	25
Personas cisgénero femeninas detectadas	34
Falsos negativos	2

Tabla de audios procesados y resultados obtenidos

El algoritmo implementado no presentó ningún problema, siendo de fácil manejo para el usuario y muy intuitivo, no se requería realizar varias grabaciones para poder obtener un resultado asertivo. Esto nos quiere decir que la aplicación de las frecuencias fundamentales asignado a cada cisgénero se encuentra dentro del rango de las personas a quien se realizó las pruebas, así mismo la aplicación de filtros para la eliminación de frecuencias no deseadas fueron aplicados de manera óptima.

Para el apartado de implementación se realizó una recopilación de muestras de 100 muestras de audio. Cabe recalcar que se procesan con el algoritmo desarrollado en el lenguaje C, en este caso se obtuvieron 50 muestras de audio pertenecientes a personas cisgénero masculino y 50 muestras de personas cisgénero femenino, obteniendo como resultado la identificación de 48 muestras de audio pertenecientes al género femenino, 45 muestras pertenecientes al género masculino y se obtuvieron 7 falsos negativos, estos resultados se muestran en la tabla 3.

Tabla 3

Resultados de audios procesados con código en lenguaje C para la FPGA

Audios procesados	100
Personas cisgénero masculinas detectadas	45
Personas cisgénero femeninas detectadas	48
Falsos negativos	7

Tabla de audios procesados y resultados obtenidos

Tanto en simulación como en implementación se presentan falsos negativos, esto se debe a que algunas muestras en el momento de ser captadas por el dispositivo de entrada se generaron varios ruidos de ambiente que provocan variaciones en los valores a la hora de procesar el audio. Para lograr una mayor eficiencia a la hora de la toma de la muestra de audio en el capítulo de conclusiones y recomendaciones se presentarán ciertas características para tener en cuenta al momento de tomar la muestra de audio.

Conclusiones

La recopilación de las diferentes técnicas de procesamiento de voz realizadas para la elaboración del estado del arte permite identificar que a diferencia de otras técnicas la implementación del algoritmo de autocorrelación no demanda muchos recursos computacionales para ser implementado, ya actualmente algunas de las técnicas se encargan en primera instancia de comparar la muestra de voz que recibe el programa con una base de datos para así poder estimar el género del hablante. Sin embargo, no se realiza el cálculo de la frecuencia central como tal y no se filtra la muestra de audio para ser procesada.

Es importante el proceso de simulación ya que nos permite identificar como tratar el audio y obtener cada uno de los parámetros de entrada y salida que tiene cada una de las funciones. Python es un lenguaje de programación que con sus librerías facilita el proceso del tratado de la información. Muchas de las funciones se encuentran verificadas e incluso se presenta la documentación pertinente para el uso de esta. Además, para identificar las operaciones que se realizan paso a paso en el algoritmo Python, permite realizar una depuración línea a línea e identificar el error presentado. En el presente trabajo se usó la versión de Python 2.7.

Con los procesos realizados en el apartado de simulación se tiene una idea clara del proceso que realiza cada una de las funciones y los tipos de datos. En este caso para la implementación en FPGA se parte de un código demo de Xilinx el cual permite grabar voz durante un tiempo de 5 segundos. Ya que nos encontramos trabajando con FPGA se debe separar tanto hardware como software, en donde los lenguajes de programación utilizados son VHDL y C respectivamente.

La validación de resultados se ha realizado en la etapa de simulación con el algoritmo de Python y también con el algoritmo final implementado en el lenguaje de programación C. En el caso de la simulación se analizaron 61 muestras de audio de las cuales se obtuvieron 2 falsos negativos, y en el caso del algoritmo final se analizaron 100 muestras y se obtuvieron

7 falsos negativos. Cabe destacar que las muestras de audio de las cuales se obtuvo como resultado falso negativo tanto en la etapa de simulación e implementación tienen mucho ruido de ambiente es por lo que se eleva el valor de la frecuencia central calculada por el algoritmo, ya que al existir demasiado ruido de ambiente la etapa de filtrado no es capaz de eliminarlo por completo. También en la implementación de reconocimiento de género por voz pueden existir personas cisgénero femeninas con una tonalidad de voz muy grave y en el caso de personas cisgénero masculinas pueden existir casos en donde su voz tenga una tonalidad muy aguda. La validación de resultados fue realizada con personas cisgénero que hablan el idioma español ya que según el idioma y los acentos presentados provocarían que la media de la frecuencia central cambie totalmente.

Recomendaciones

Para futuros trabajos que impliquen el procesamiento de señales se recomienda utilizar el algoritmo de correlación ya que es uno de los más eficientes y no consume muchos recursos computacionales a la hora de ejecutarse.

Para la realización de un proyecto de procesamiento de señales en este caso la voz, para un mejor entendimiento del resultado al cual se quiere llegar es necesario dividir el procesamiento por etapas como: entrada de datos, identificación de los parámetros de los datos (tamaño, frecuencia de muestreo, tipo de datos float o double), filtrado, algoritmo para procesar los datos (puede ser autocorrelación, FCC o reconocimiento de emociones según sea el caso), comparación y finalmente la presentación del resultado.

Para la correcta ejecución del reconocimiento de género por voz, el hablante puede decir su nombre completo ya que la grabación de la muestra es de 3 segundos. Se debe evitar saturar el micrófono a la hora de la toma de muestras ya que los ruidos de ambiente modifican de cierta forma el resultado final y también alzar mucho la voz perjudica el resultado.

Referencias

Ballesteros, D. M., & Torres, D. R. (2018). *INTRODUCCIÓN A LOS FILTROS DIGITALES*. REDIPE Red Iberoamericana de Pedagogía.

Barchiesi, J. V. (2008). *Introducción al procesamiento digital de señales*. Eds. Universitarias de Valparaíso.

Boutros, A., & Betz, V. (2021). FPGA Architecture: Principles and Progression. *IEEE Circuits and Systems Magazine*, *21*(2), 4-29. https://doi.org/10.1109/MCAS.2021.3071607

Castrosqui Florián, M. A., & Castrosqui Florián, M. A. (2017). *Procesamiento de voz para mejorar la pronunciación* [Info:eu-repo/semantics/bachelorThesis]. https://bit.ly/3EXEmav

Chacon, J., Villamizar, A., & Ardila, D. (s. f.). *Verilog y VHDL diferencias ventajas y desventajas Verilog and VHDL Differences advantages and disadvantages*. 8.

Chuchuca, C., & Esteban, J. (2022). *Diseño y desarrollo de un sistema prototipo para reconocimiento automático del hablante empleando técnicas de aprendizaje profundo*. 24.

Cuadros, J. M. M. (2015). *El entorno de diseño Xilinx ISE Design Suite*. http://bit.ly/3V36cHU

Delgado, J. D. (2021, julio 3). *FIR Filter Software Part 1—DE0-NANO-SOC*. FPGALOVER. https://bit.ly/3XrwCEG

GStreamer: Framework multimedia de código abierto. (2022, noviembre 7). gstreamer. https://bit.ly/3tToJu6

III, J. O. S. (2007). *Introduction to Digital Filters: With Audio Applications*. https://bit.ly/3tWzMTC

Lin, Q., & Shao, Y. (2018). *A Novel Normalization Method for Autocorrelation Function for Pitch Detection and for Speech Activity Detection* (p. 2101). https://doi.org/10.21437/Interspeech.2018-45

ONU. (2022). Glosario [LIBRES&IGUALES]. *GLOSARIO*. https://bit.ly/3tUGCZH

Paolini, G., Hernández, A., & Pereyra, V. (2018). Frecuencia fundamental de habla de voz normal según sexo en la Provincia de Córdoba, Argentina. *Revista de la Facultad de Ciencias Médicas de Córdoba*, 99-100. https://doi.org/10.31053/1853.0605.v0.n0.21300

Rishabh, P. (2018, diciembre 13). Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc(). *GeeksforGeeks*. https://bit.ly/3EYbDSO

Rodero, E. (2001). La voz masculina y femenina en los informativos radiofónicos. *Biblioteca on-line de Ciencias da Comuniçao*.

SigmaStudio® | *Dispositivos analógicos*. (2022). https://bit.ly/3EW93MZ

Vashistha, P., Singh, J. P., Jain, P., & Kumar, J. (2019). Raspberry Pi based voice-operated personal assistant (Neobot). *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 974-978. https://doi.org/10.1109/ICECA.2019.8821892

Vega, E. G. H., Muñoz, J. L. C., & Acosta, J. A. N. (2019). IMPLEMENTACIÓN DE UN SISTEMA GENERADOR Y ELIMINADOR DE ECO CON FILTRO ADAPTATIVO USANDO EL ALGORITMO LMS EN UN FPGA ARTIX-7 (IMPLEMENTATION OF AN ECHO GENERATOR AND ELIMINATOR SYSTEM WITH ADAPTIVE FILTER USING LMS ALGORITHM IN AN ARTIX-7 FPGA). *Pistas Educativas*, *41*(134), Art. 134. http://bit.ly/3gsMTJ8

Verde, L., De Pietro, G., & Sannino, G. (2018). A methodology for voice classification based on the personalized fundamental frequency estimation. *Biomedical Signal Processing and Control*, *42*, 134-144. https://doi.org/10.1016/j.bspc.2018.01.007

Vivado ML Enterprise. (s. f.). Xilinx. Recuperado 23 de noviembre de 2022, de https://bit.ly/3U30cgM

XILINX. (2022). *What is an FPGA? Field Programmable Gate Array*. Xilinx. https://bit.ly/3hWroAz

Xu, C., Jiang, S., Luo, G., Sun, G., An, N., Huang, G., & Liu, X. (2022). The Case for FPGA-Based Edge Computing. *IEEE Transactions on Mobile Computing*, *21*(7), 2610-2619. https://doi.org/10.1109/TMC.2020.3041781