UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

SISTEMA DE VERIFICACIÓN E IDENTIFICACIÓN DE INTERLOCUTOR

Alumno: José Manuel Villapún Sánchez

Tutor: Rafael Gadea Gironés

INTRODUCCIÓN

- Este trabajo se compone de un sistema de verificación e identificación de fonemas, en donde se hace uso de un dispositivo FPGA de última generación.

- Se ha elegido este diseño por los diferentes campos tratados dentro de la electrónica digital en este, como son:
  - Uso de lógica programable sobre FPGAs
  - Uso de sistemas microprocesador.
  - Uso de sistemas procesador con Linux embebido.
  - Diseño de procesamiento de señal sobre FPGAs
  - Uso de algoritmos de aprendizaje.

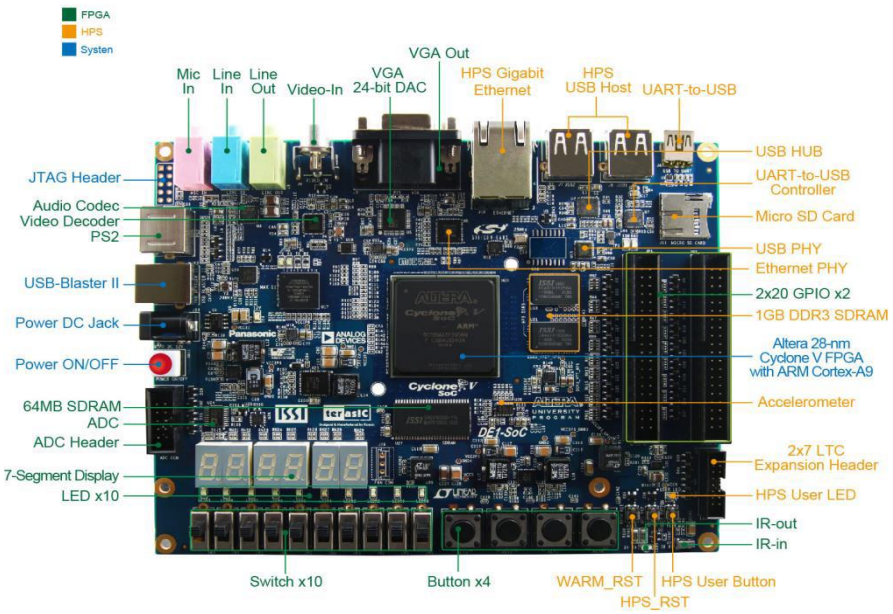
Todos estos son campos en constante evolución, de hay el interés de utilizar la última generación de dispositivos.

Objetivos del trabajo

- Preparar Linux
  - Todas la etapas
  - Sistema HPS
  - Comunicación del HPS con la FPGA
  - Control del hardware
- Realizar una interface de usuario intuitiva y agradable
- Uso de algoritmos de entrenamiento para el reconocimiento de fonemas
- Pruebas y verificación del sistema

Entorno Hardware utilizado

- Placa de desarrollo DE1-SoC
  - FPGA Cyclone V con 2 núcleos ARM embebidos
    - HPS (Hard Processor System) con un Dual-core ARM Cortex-A9 a 800MHz
    - Puentes HPS-FPGA
    - Nios II
  - Audio Codec
  - SDRAM externa de 64Mbytes
  - Módulo de tarjeta SD
  - Controlador Ethernet
  - Puerto serie.
  - Pulsadores y LEDs
- Terasic MultiTouch



Herramientas de desarrollo

- Quartus II 13.1 de Altera
  - Programación de la lógica programable
  - Megawizard Plug-in manager
  - Qsys
  - Software Build Tool for Eclipse
- QT creator
  - Compilador/depurador cruzado.
  - Compilador/depurador en nativo.
- SoC Embedded Desing Suite (EDS) de Altera
  - Bsp-Editor
  - Command Shell
- Compilador linaro-gcc

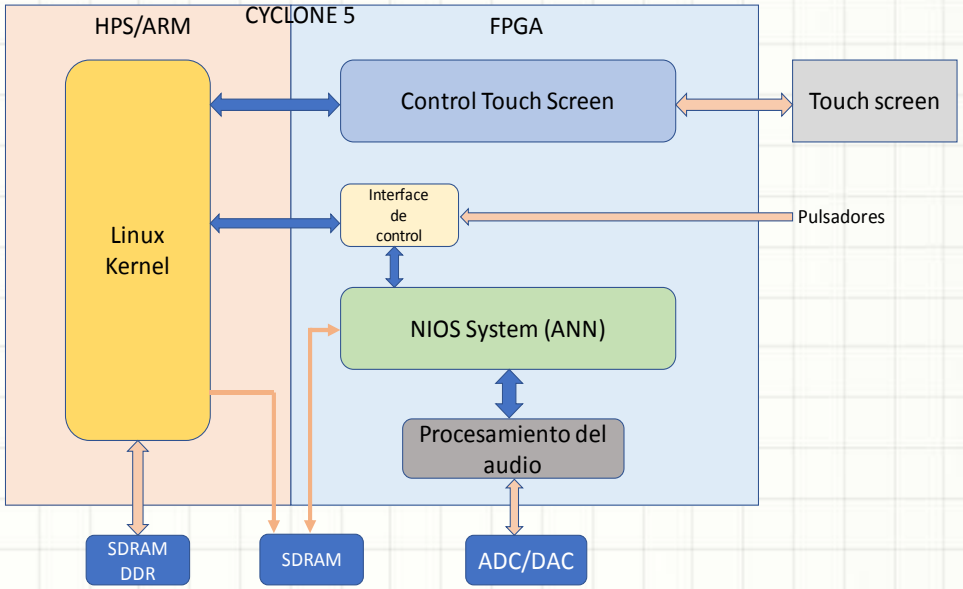
DISEÑO 1 DE LA PARTE HARDWARE

El diseño sobre la FGPA se compone de dos partes:

- Montaje y configuración del sistema HPS de la FPGA, el cual hará correr el kernel de Linux. Se usa la herramienta Qsys para instanciarlo y configurar todos los puentes con la parte de lógica programable y sus interfaces de I/O.
- Diseño de la parte de lógica programable, donde se instancian los siguientes módulos:
  - Core IP de video y control de la Touch screen
  - Sistema Nios II
  - Bloque de procesamiento del audio.
  - Interface de control.

Se usan los siguientes periféricos:

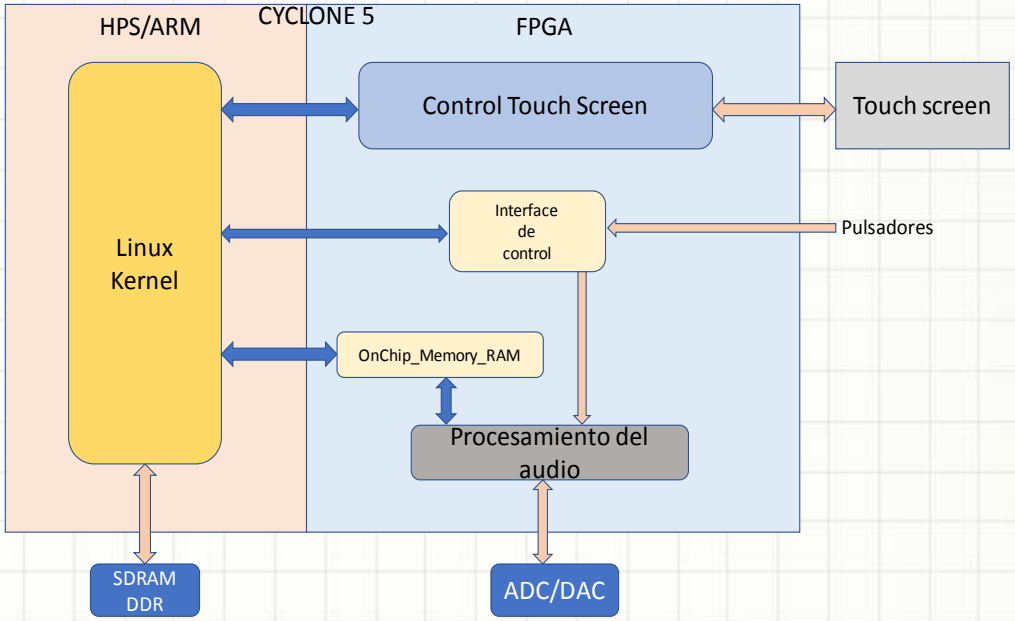
- Tarjeta SD para el montaje del sistema Linux
- SDRAM para la carga del Nios II
- Audio Codec para la adquisición de las muestras de audio
- Touchsreeen para la interface de usuario



DISEÑO 2 DE LA PARTE HARDWARE

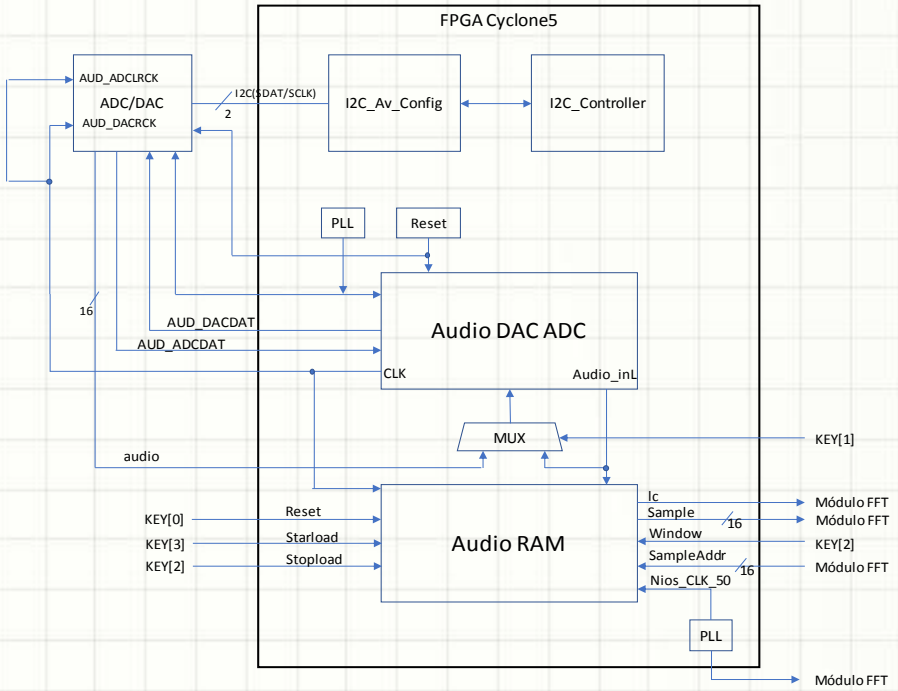
- En este diseño se elimina el módulo de Nios II utilizado en el primer diseño, donde se traslada su función al sistema Linux de la parte HPS.
- Se adapta la interface de control a este nuevo diseño, donde tendrá las siguientes funciones:
  - Interactuar entre la parte HPS y la parte de FPGA para el control e indicación.
  - Intercambiador de datos entre diferentes ejecutables que corran en el sistema Linux.

- Se interconecta la memoria de almacenamiento de coeficientes MCCF al sistema HPS. Antes esta memoria estaba conectada al módulo Nios II



ADQUISICIÓN DE MUESTRAS CON EL AUDIO CODEC

- Para la adquisición de audio se utiliza el periférico de “Audio CODEC” que dispone la placa DE1-SoC.
  - Tiene tres conectores JAG en la placa, donde conectaremos el micrófono.
  - Este se conecta a la FPGA a través de I2C y por líneas discretas de datos y reloj.
- Para el control y configuración de este dispositivo se implementan in la FPGA dos módulos.
  - I2C_Av_Config: se encarga de generar los datos a enviar por I2C para la configuración.
  - I2C_Controller: Se encarga de implementar el protocolo de comunicaciones serie I2C.
- El módulo Audio DAC ADC se encarga de controlar la recepción de los datos de muestras, los cuales llegan por una línea del Audio CODEC en serie y son separados en este módulo
- El módulo de Audio RAM se compone de una memoria onchip en donde se irán almacenando las muestras que llegan según la ventana controlada por el usuario



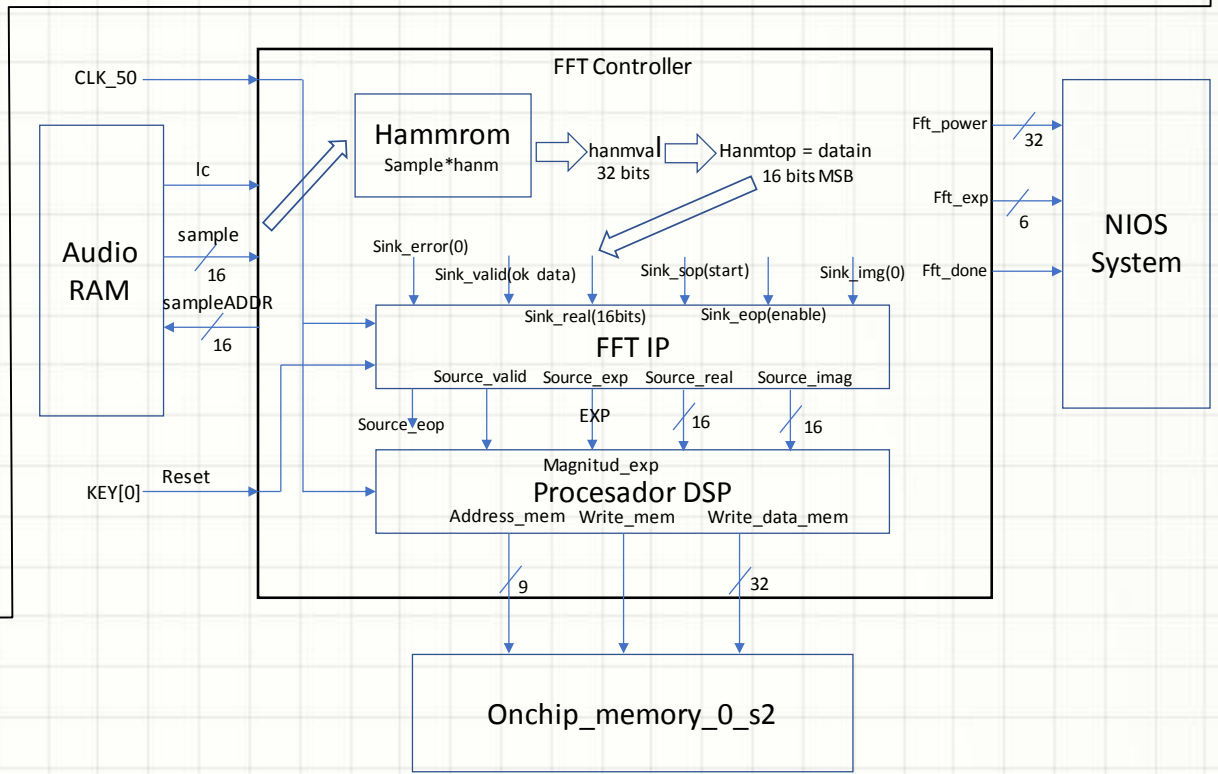
PROCESAMIENTO DE LA SEÑAL

En este módulo se realiza el procesamiento de señal para obtener los coeficientes MCCF. Para ello se implementan dentro de este los siguientes submódulos:

- Hammrom
- FFT IP
- Procesador DSP

El funcionamiento de este módulo se compone de las siguientes etapas.

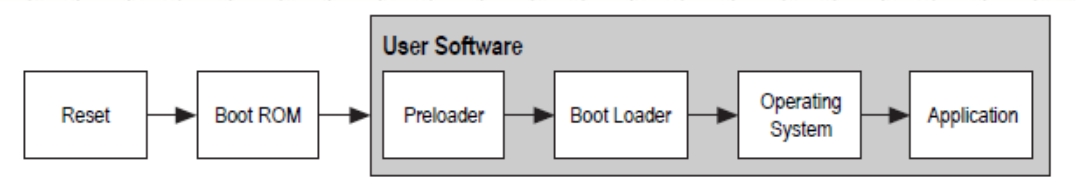
- Toma las muestras de la Audio RAM cunado esta le indica por una línea que está llena.
- Procesa las muestras y las guarda en la memoria onchip compartida con el módulo Nios II en el primer diseño y con el sistema HPS en el segundo.
- Genera el umbral de potencia alcanzado por las muestras.
- Finalmente indica al sistema Nios II o al sistema HPS el fin del procesado.



CONFIGURACIÓN DE LA DISTRIBUCIÓN DE LINUX EN EL HPS

La configuración del sistema Linux se compone de las siguientes etapas.

- Reset
- Boot ROM: Es una memoria embebida y configurada en fábrica.
- Generación del preloader
- Generación del Boot loader.
- Generación del kernel de Linux.
- Generación del Device Tree.
- Generación del root filesystem.



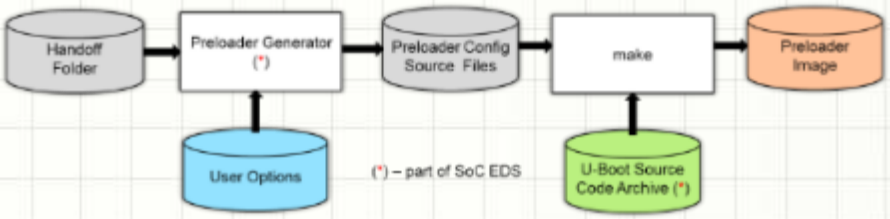
Una vez que tenemos todos los elementos generados, estos se cargarán en la tarjeta SD en sus respectivas particiones.

Para la carga de la FPGA se añadirá a la partición el binario de esta.

GENERACIÓN DEL PRELOADER Y BOOTLOADER

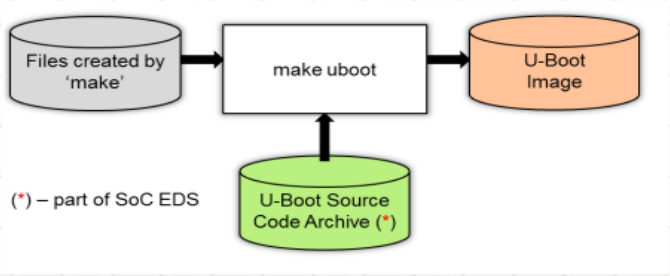
El preloader va a cumplir las siguientes funciones:

- Inicializar la interface de la SDRAM.
- Configurar el registro “remap” de la on-chip RAM a la dirección 0x0, por lo que las excepciones van a ser manejadas por este.
- Configurar las entradas y salidas del sistema HPS.
- Configurar el multiplexado de pines.
- Configurar los relojes del HPS.
- Inicializar la FLASH Controller.
- Cargar el BootLoader dentro de la SDRAM y pasarle el control.



El bootloader cumplirá las siguientes funciones:

- Configuración del entorno del sistema operativo
- Recuperar la imagen del sistema operativo desde la SD/MMC.
- Almacenar la imagen de arranque en SDRAM y pasar el control al cargador de arranque siguiente, es decir, arrancar el Kernel.
- Modificar el Device Tree Blob.
- Cargar el binario de la FPGA.

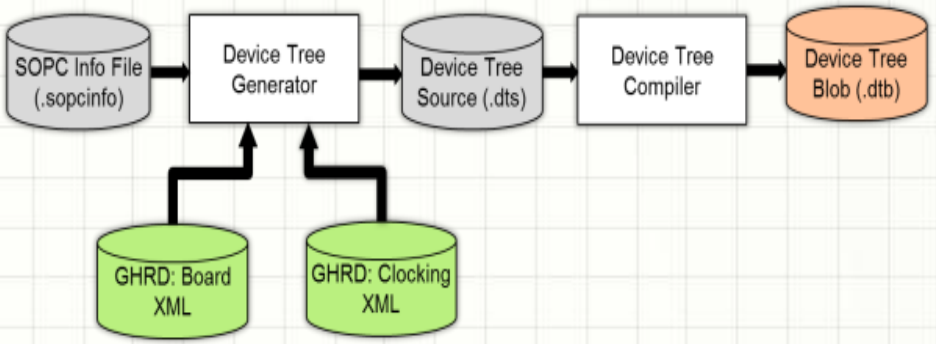


COMPILACIÓN DEL KERNEL DE LINUX Y CONFIGURACIÓN DEL DEVICE TREE

- El Device Tree es una estructura de datos que describe el hardware presente en el sistema operativo, el cual será leído en el arranque del kernel de Linux para que este sepa la configuración de Hardware.
- Esta estructura de datos va a contener la configuración de los siguientes elementos:
  - Número de CPUs.
  - Tamaño y ubicación de varias RAMs.
  - Buses y puentes.
  - Conexiones de los dispositivos periféricos
  - Controladores de interrupciones y conexiones de línea IRQ.
  - Configuración específica del controlador de dispositivo, como:
    - Dirección MAC de Ethernet
    - Reloj de entrada del periférico

Para la generación del Device Tree se necesitan los siguientes elementos:

- Fichero .sopcinfo generado con el Qsys.
- Fichero xml de relojes del sistema.
- Fichero xml de configuración de tarjeta.



GENERACIÓN DE LA LIBRERÍA GRÁFICA PARA QT

Esta es una parte importante para el diseño software que se realizará sobre Linux, con la podremos crear interface gráficas de usuario de forma sencilla.

- Se compone de un framework multiplataforma orientado a objetos.
- Es un software libre y de código abierto.
- Está programada en C++.
- Nos proporciona diferentes elementos de diseño gráfico como widgets, ventanas, botones, listas desplegables, etc.
- Se programa desde el QT creator.

Para poder hacer uso de ello, se ha compilado el código fuente con las siguientes configuraciones:

- Plataforma altera.
- Uso del framebuffer de Linux.
- Compilador gcc-linaro.
- Uso del módulo touchscreen.
- Tamaño de pantalla.
- Etc.

También se genera la librería TSLIB para el control de la touchscreen.

- Esta librería se añade a la librería QT.

DISEÑO SOFTWARE SOBRE LINUX

- El diseño software sobre Linux se compone de un ejecutable de interface de usuario con las siguientes características y funciones.
  - Se programa en C++ y se compila con el compilador gcc-linaro.
  - Hace uso de las librerías de QT
  - Se encarga de hacer de interface entre el usuario y el sistema.
  - Programa el Frame Buffer de Linux con el que se escribirá en la pantalla LCD, por lo que es la capa alta de control de la pantalla.
  - También controla la multitouch.
  - Se comunica con la FGPA a través de registros mapeados en memoria de Linux, con los cuales puede controlar y obtener datos del hardware de la FPGA.
  - Se inicia automáticamente una vez el kernel de Linux esté arrancado
- En el caso del segundo diseño se implementa otro ejecutable para albergar el algoritmo de aprendizaje.
  - En este caso combiben los dos ejecutables en el sistema Linux.
  - Se intercambian información a través de la interface de control implementa en la parte de la FGPA.
  - El segundo ejecutable lee de la memoria onchip implementa en la FPGA los coeficientes MCCF.

DISEÑO SOFTWARE SOBRE NIOS II

- Este diseño será el que se ejecute sobre el sistema Nios II implementado en la FPGA, el cual tiene la función de albergar los algoritmos de aprendizaje.
- El control de este software se hará a través de la interface de control implementa en la FPGA, la cual configurará los siguientes modos de funcionamiento.
  - Modo entrenamiento o prueba.
  - Modo reconocimiento de vocales o números, o verificación de vocales o números.
  - Indicación del número o vocal reconocido.
  - Resultado de la verificación.
- Para obtener los coeficiente MCCF, el software que corre sobre el Nios II utiliza una variable apunta a la sección de memoria donde se encuentra mapeada la memoria de coeficientes.
  - Utiliza la HAL para leer los siguientes registros mapeados en memoria:
  - FFT done: el cual indicará que tenemos un conjunto de coeficientes procesados.
  - FFT power y exp: los cuales indicarán el nivel de potencia de las muestras y con los que podremos discriminar con un umbral los coeficientes que no nos interesen.

INTERFACE DE USUARIO FINAL

Se compone de las siguientes patallas o widgets:

- Pantalla de Inicio.
- Pantalla de selección.
- Pantalla de entrenamiento.
- Pantalla de prueba de reconocimiento de vocales.
- Pantalla de prueba de identificación de vocales.
- Pantalla de verificación de interlocutor.
- Pantalla de identificación de números.
- Ventana de información.

Pruebas realizadas:

- Correcto control del hardware mediante estas pantallas
- Funcionamiento final del sistema entrenando y probando el reconocimiento e identificación de los algoritmos de aprendizaje

FUTURAS MEJORAS

- Crear un sistema de pruebas de algoritmos de aprendizaje más completo, donde para ello:
  - Permitir el uso de diferentes tipos de algoritmos.
  - Hacer una interface de usuario más completa.
  - Buscar otras áreas aparte del reconocimiento de voz como pueden ser:
    - Procesamiento de imágenes.
    - Uso en sistemas de RF como filtros y sistemas RADAR.
  - Implementar en software el almacenamiento mediante un fichero de las redes neuronales entrenadas.
- Liberar la parte de lógica programable de procesamiento, para ello:
  - Implementar en software de Linux parte del procesamiento.
  - Buscar acelerar el procesado y los algoritmos con aceleradores hardware implementados en la FPGA con OpenCL.

SISTEMA FINAL