CAPÍTULO 5 DISEÑO E IMPLEMENTACIÓN

5.1 INTRODUCCIÓN

En los capítulos anteriores se ha descrito con profundidad la solución adoptada para la realización del sistema de reconocimiento automático de vocales y su aplicación en el control por voz de un puntero ratón PS/2. Se pretende crear ahora un prototipo cuyo diseño proporcione solución al problema especificado. El diseño del sistema digital se realizará en el lenguaje de descripción de hardware VHDL y tras un proceso de síntesis se implementará sobre una FPGA. Para facilitar el diseño, se utiliza una tarjeta de desarrollo comercial que permite una programación sencilla del sistema digital sobre la FPGA. La tarjeta de desarrollo se insertará sobre un circuito externo auxiliar que permite a través de un conversor analógico-digital obtener muestras digitales de una señal de audio captada por un micrófono.

5.2 UNIDAD FÍSICA DEL SISTEMA: FPGA

Dadas las características del sistema digital se decide implementarlo sobre una FPGA debido a las enormes posibilidades que ofrecen estos dispositivos para crear diseños electrónicos de elevada complejidad. Se utiliza una **FPGA Spartan-II de XILINX de 50-Kpuertas en un**

paquete QFP de 144 pines. En la FPGA estará la base de la lógica programable del prototipo. Las FPGA's son dispositivos digitales cuya lógica interna es totalmente configurable desde el exterior a través de métodos de programación que el diseñador realiza de forma sencilla.

La FPGA será la encargada de realizar la compleja labor del reconocimiento de las vocales pronunciadas, para lo cual deben extraerse unos parámetros adecuados que eliminen toda la redundancia en la señal de audio captada y a la vez permitan caracterizar adecuadamente a cada una de las señales vocálicas. Se realizará la Transformada Rápida de Fourier a las muestras digitales de la señal de audio para obtener las características en frecuencia de la vocal pronunciada. En base a estos datos se obtendrán los coeficientes cepstrales que resumen de forma natural la información vocálica.

Dentro de la FPGA se pondrá en funcionamiento una red neuronal de 32 neuronas en la topología de un mapa autoorganizado que detectará la vocal pronunciada. Esta red neuronal es el núcleo del sistema de reconocimiento y los parámetros de especifican su funcionalidad se encuentran almacenados en una memoria FLASH externa, por lo que la FPGA también deberá realizar su lectura. Las señales vocálicas detectadas se traducirán en los eventos de movimiento realizables por un ratón de ordenador de acuerdo con un protocolo establecido ("movimiento del puntero hacia arriba", "movimiento del puntero hacia abajo", "movimiento del puntero hacia la derecha", "movimiento del puntero hacia la izquierda", "pulsación del botón derecho del ratón" y "pulsación del botón izquierdo del ratón").

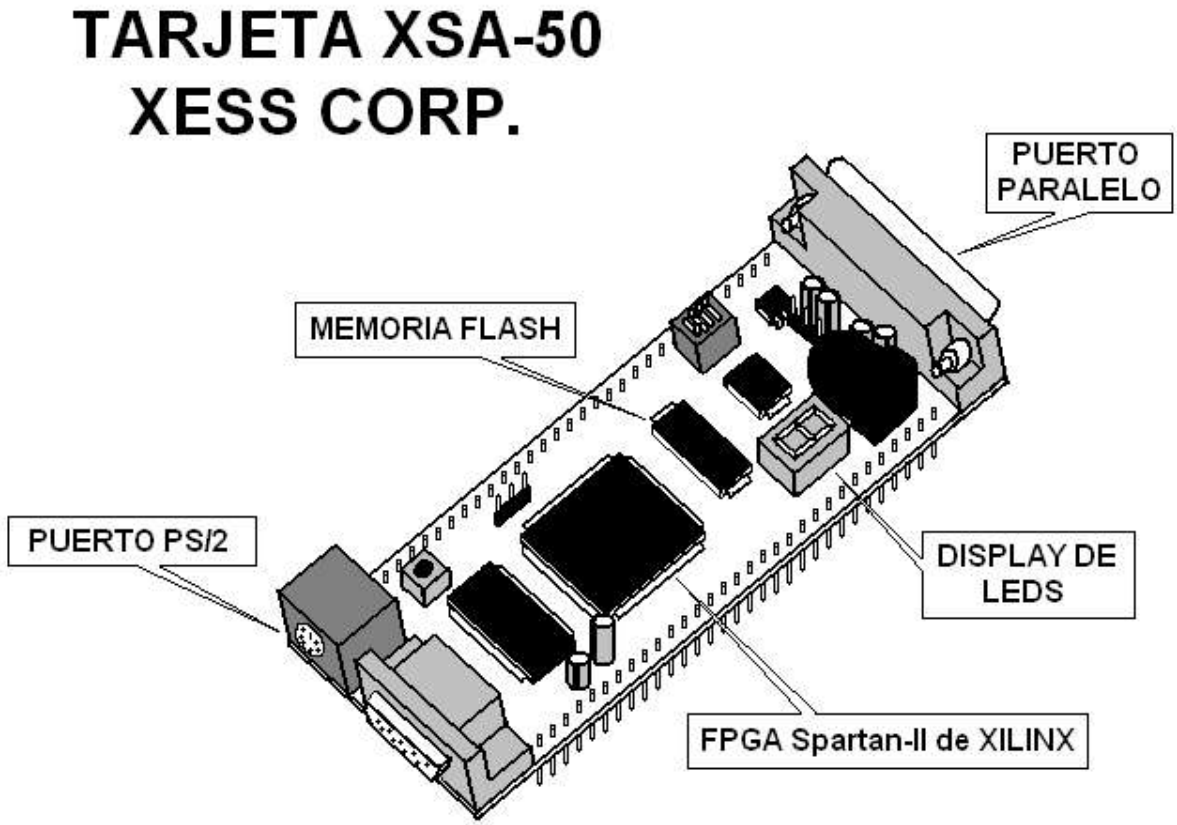
Una vez decidido el evento de movimiento solicitado por el usuario el sistema digital implementado en la FPGA debe proporcionar la respuesta adecuada y actuar en consecuencia de acuerdo con la sintaxis y semántica asociada al protocolo serie PS/2, y de esta manera emular el funcionamiento de un ratón de ordenador. Deberá gestionar toda la comunicación con el ordenador a través del protocolo serie PS/2, procesar y responder a todos los comandos recibidos, y, por supuesto, generar los paquetes de datos que informarán al ordenador de los eventos de movimiento producidos.

5.3 LA TARJETA DE DESARROLLO XSA

Con el objeto de comprobar el buen funcionamiento del sistema digital implementado en la FPGA y depurar el diseño se hace uso de la tarjeta de desarrollo XSA-50 de XESS Corporation. Permite una fácil programación desde un PC a través del puerto paralelo. La FPGA Spartan-II de XILINX de 50-Kpuertas queda incluida como núcleo de la tarjeta XSA. La aplicación "GXSLOAD" permite cargar a través del puerto paralelo los ficheros de configuración necesarios para que en la FPGA se implemente el sistema digital diseñado.

La tarjeta XSA incluye la memoria **FLASH RAM de Atmel AT49F002** de 256 Kbytes (256K x 8) que ha sido utilizada para almacenar variables internas del diseño durante el tedioso trabajo de

simulación física, comprobación de errores y depuración del código del diseño del sistema digital. Posteriormente, estas variables pueden ser extraídas de la memoria y tras su representación en la plataforma MATLAB® puede realizarse una interpretación del funcionamiento interno del sistema. La memoria FLASH se utiliza también para almacenar los parámetros funcionales de la red neuronal. Durante el proceso interno del sistema se leerán los valores de los parámetros almacenados para poner en ejecución la red neuronal.



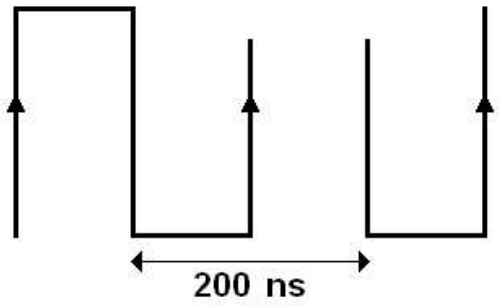
También resultó útil en el proceso de depuración del sistema el display de ocho leds presente en la tarjeta XSA. Este display permite mostrar en tiempo real ocho valores diferentes de señales lógicas internas del sistema. Modificando la lógica interna del sistema se pueden preparar los leds para que se activen en determinadas circunstancias para mostrar eventos en el funcionamiento interno del sistema digital. En la fase de ejecución del diseño el display de leds muestra de forma gráfica la vocal detectada en el proceso de reconocimiento.

Será también la tarjeta XSA la que permita el acceso bidireccional a las señales PS/2 que se utilizarán para poner en comunicación el ordenador con el prototipo diseñado para el proyecto. La tarjeta XSA proporciona un interfaz PS/2 mini-DIM (conector J4) para ratón o teclado. La FPGA recibe dos señales desde el interfaz PS/2: una señal de reloj y una de datos serie sincronizados con los flancos de la señal de reloj. En la tarjeta XSA se adaptan las señales de

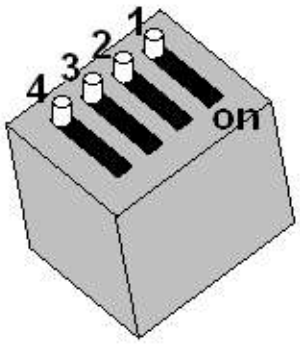
salida de la FPGA, que emulan el comportamiento del interfaz PS/2, con la potencia y el voltaje necesarios para que se pueda acceder al puerto PS/2 del ratón.

El diseño digital necesita de un reloj para funcionar, que también es proporcionado por la tarjeta XSA. El reloj de la tarjeta XSA se ha programado con una frecuencia de 5 MHz (200 ns de periodo), lo suficientemente rápido como para realizar el reconocimiento en tiempo real y lo suficientemente lento como para evitar que los registros internos de almacenamiento guarden valores digitales antes de haberse estabilizado.

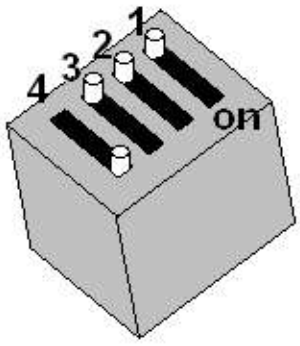
RELOJ DEL SISTEMA	
FRECUENCIA:	5 MHz
PERIODO:	200 ns



La tarjeta XSA también proporciona los dispositivos de regulación de tensión que mantienen las tensiones de alimentación necesarias para el funcionamiento de la FPGA y la activación de los circuitos mencionados anteriormente. Se utiliza uno de los interruptores de la tarjeta XSA para poner en funcionamiento el dispositivo a modo de RESET negado. Cuando el interruptor está en la posición OFF (la señal asociada toma un valor '1' lógico) se resetea el sistema diseñado. Cuando el interruptor está en la posición ON (se fija un '0' lógico en la señal asociada) se permite el funcionamiento normal del dispositivo.



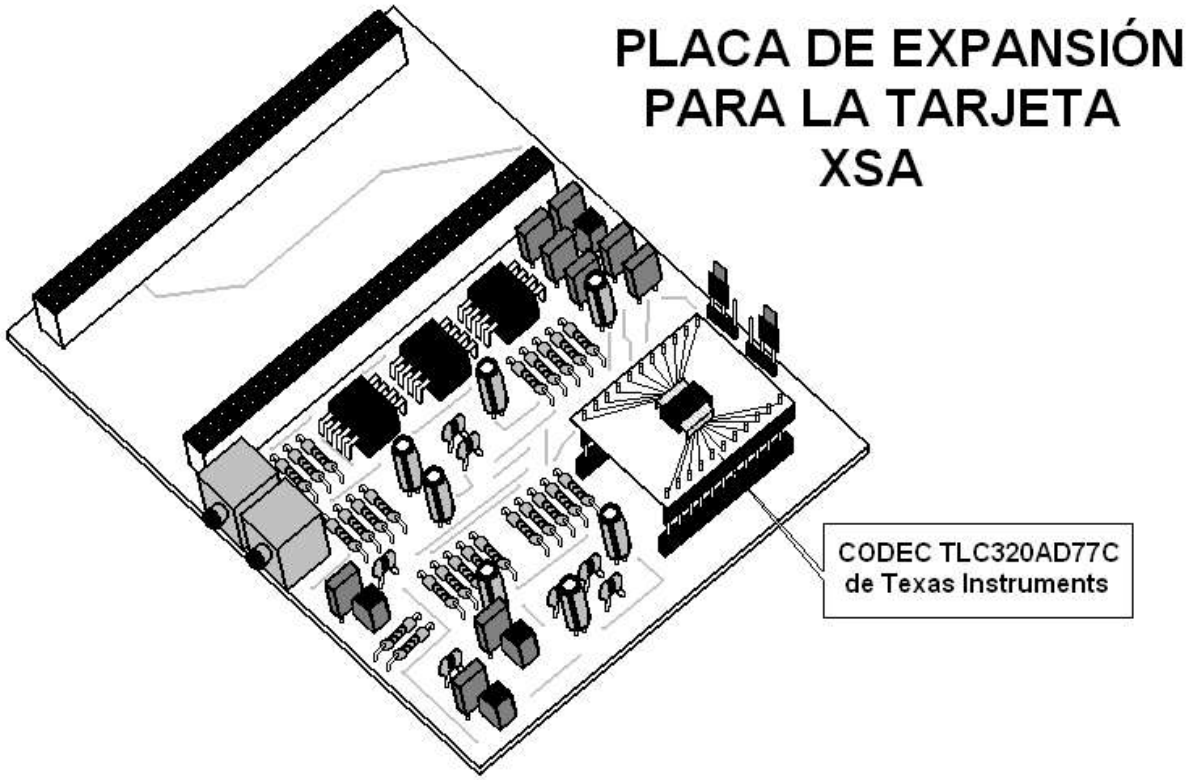
RESETEO DEL SISTEMA



FUNCIONAMIENTO NORMAL DEL DISPOSITIVO

5.4 PLACA DE EXPANSIÓN PARA LA TARJETA XSA-50

Como la tarjeta XSA no dispone de un convertidor analógico-digital se hace necesaria una electrónica auxiliar externa que proporcione la posibilidad de muestrear digitalmente una señal de audio. Aprovecho una placa de expansión de la tarjeta XSA diseñada por Luis Alba para el Departamento de Electrónica de la Escuela Superior de Ingenieros de Sevilla. La tarjeta XSA-50 y la placa de expansión pueden acoplarse a través de un conector con el que la FPGA puede acceder a las líneas de comunicación con el conversor analógico-digital. Este circuito será la interfaz que permita a la FPGA ponerse en contacto con el mundo exterior. El circuito es capaz de adaptar la señal acústica, con la que se pretende controlar el emulador de ratón, a una señal digital apta para el procesamiento en la FPGA. En la conversión se utiliza el **CODEC TLC320AD77C de Texas Instruments**, un convertidor analógico-digital y digital-analógico muy competitivo de 24-bits basado en la conversión sigma-delta para aplicaciones de consumidor que demandan una excelente calidad de audio. La señal de audio se capta a través de un micrófono que se conectará a la placa de expansión y es adaptada, filtrada y amplificada por los circuitos de la placa de expansión. La alimentación de la placa de expansión es proporcionada por la tarjeta XSA. También alimentará al micrófono utilizado para captar la señal de audio. Una descripción más detallada del proceso realizado en la placa de expansión así como la configuración adoptada para el CODEC se expone en el capítulo 2 de la memoria del proyecto.



5.5 VHDL

Para el modelado, diseño y síntesis del circuito que vamos a instalar en la FPGA recurriremos al lenguaje de descripción de hardware VHDL. Las metodologías de diseño electrónico denominadas "top-down", basadas en el empleo de lenguajes de descripción de hardware, han transformado los procedimientos de diseño de sistemas electrónicos, muy especialmente de circuitos integrados. El lenguaje VHDL es su más claro exponente, abriendo enormes posibilidades al permitir la simulación con descripciones de partes del sistema con diferentes niveles de abstracción. Este hecho, unido a la posibilidad de realizar la síntesis automática, y a la concepción de bloques reutilizables y reconfigurables en función de las necesidades de la aplicación, ha permitido dotar al diseñador de enormes recursos que hacen posible abordar la creciente complejidad de los diseños con mayores garantías de éxito.

El VHDL nace con una sintaxis (formalizada mediante una gramática) y una semántica (no formal, matemáticamente hablando) definidas y dirigidas hacia el modelado para la simulación de hardware. Sin embargo, rápidamente se abordó su uso como soporte para todas las fases del proceso de diseño, y muy especialmente para las etapas de síntesis. Las ventajas más relevantes que pueden suponer el uso de este lenguaje son:

- El VHDL es interpretable tanto por las personas como por los ordenadores, pudiendo proporcionar soporte tanto a las tareas estrictas de diseño (modelado, simulación, síntesis, verificación) como a las de comunicación e intercambio de modelos entre los distintos equipos de trabajo y, obviamente, a las de documentación y mantenimiento de los diseños.

- Es un lenguaje de disponibilidad pública, no sometido a ninguna firma ni patente. Está definido, documentado y mantenido por el IEEE, quien garantiza su estabilidad y soporte.

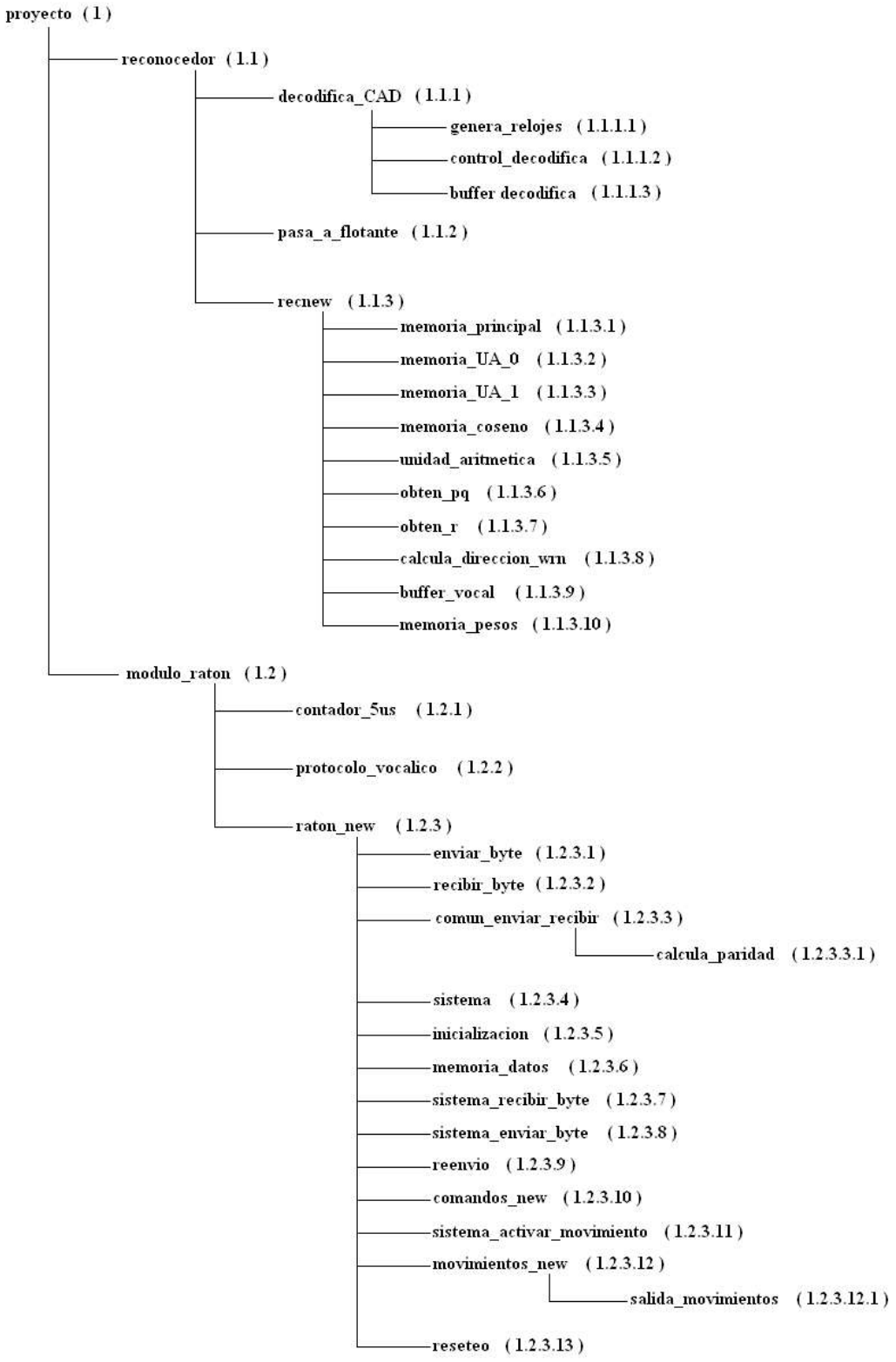
- VHDL, tanto por su definición y diseño como por los niveles de abstracción es totalmente independiente de la tecnología final de implementación de los circuitos.

- Una descripción VHDL de un diseño, inicialmente desarrollada para una determinada tecnología puede ser fácilmente reutilizado en diseños o materializaciones posteriores donde la tecnología, la alternativa de implementación y/o el CAD pueden ser distintos. Este hecho facilita la evolución de los productos, ya sea mejorando o incrementando sus características funcionales, o bien vía renovación tecnológica haciendo factible su reimplementación sobre nuevas tecnologías.

Es por todas estas características por las que decido utilizar el lenguaje VHDL para la descripción del hardware del circuito que se va a diseñar para la realización del proyecto.

5.6 UNIDADES DE DISEÑO

La descripción funcional del diseño se hace difícil de abordar si se pretende dar una visión clara de la solución adoptada en la implementación física. Por este motivo he decidido establecer un orden jerárquico en las unidades funcionales implicadas en el proyecto. Este orden se corresponde con la estructura que toman los módulos de diseño en VHDL. En cada uno de los módulos se realiza una descripción de su funcionalidad así como de los mecanismos utilizados para desarrollarla. Si un módulo se vale de otro módulo, en el primero se realizará una descripción superficial de los procesos realizados en el segundo, mientras que en el segundo la descripción será más detallada. Siguiendo el orden jerárquico de las entidades de diseño en VHDL utilizadas para implementar el prototipo de ratón de ordenador controlado por la voz puede obtenerse una visión todo lo profunda que se quiera de los aspectos tanto funcionales como estructurales del diseño, con tal de seguir el orden descendente en la instalación de los componentes del sistema. Así, la entidad de mayor orden en la jerarquía (módulo de índice 1: "proyecto") describe de forma global y aportando una visión de conjunto a todo el diseño. La comprensión del diseño empezaría con la lectura de este módulo, para posteriormente profundizar en sus componentes siguiendo el inherente orden jerárquico. Los módulos que tienen un índice que empieza por 1, se corresponden con unidades descriptivas con funciones particulares en el diseño del proyecto, mientras que los módulos que tienen un índice que empieza por 2 se corresponden con unidades genéricas de uso común en todos los módulos.


proyecto (1)
reconocedor (1.1)
decodifica_CAD (1.1.1)
genera_relojes (1.1.1.1)
control_decodifica (1.1.1.2)
buffer decodifica (1.1.1.3)
pasa_a_flotante (1.1.2)
recnew (1.1.3)
memoria_principal (1.1.3.1)
memoria_UA_0 (1.1.3.2)
memoria_UA_1 (1.1.3.3)
memoria_coseno (1.1.3.4)
unidad_aritmetica (1.1.3.5)
obten_pq (1.1.3.6)
obten_r (1.1.3.7)
calcula_direccion_win (1.1.3.8)
buffer_vocal (1.1.3.9)
memoria_pesos (1.1.3.10)
modulo_raton (1.2)
contador_5us (1.2.1)
protocolo_vocalico (1.2.2)
raton_new (1.2.3)
enviar_byte (1.2.3.1)
recibir_byte (1.2.3.2)
comun_enviar_recibir (1.2.3.3)
calcula_paridad (1.2.3.3.1)
sistema (1.2.3.4)
inicializacion (1.2.3.5)
memoria_datos (1.2.3.6)
sistema_recibir_byte (1.2.3.7)
sistema_enviar_byte (1.2.3.8)
reenvio (1.2.3.9)
comandos_new (1.2.3.10)
sistema_activar_movimiento (1.2.3.11)
movimientos_new (1.2.3.12)
salida_movimientos (1.2.3.12.1)
reseteo (1.2.3.13)

1	proyecto.vhd

```
ENTITY proyecto IS
  PORT(
    reloj_P88: IN STD_LOGIC;
    CE_NEG_FLASH_P41: OUT STD_LOGIC;
    RESET_NEG_FLASH_P59: OUT STD_LOGIC;
    OE_NEG_FLASH_P43: OUT STD_LOGIC;
    WE_NEG_FLASH_P58: OUT STD_LOGIC;
    direccion0_P40: OUT STD_LOGIC;
    direccion1_P29: OUT STD_LOGIC;
    direccion2_P28: OUT STD_LOGIC;
    direccion3_P27: OUT STD_LOGIC;
    direccion4_P74: OUT STD_LOGIC;
    direccion5_SCLK_P75: OUT STD_LOGIC;
    direccion6_P76: OUT STD_LOGIC;
    direccion7_P66: OUT STD_LOGIC;
    direccion8_P50: OUT STD_LOGIC;
    direccion9_LRCLK_P48: OUT STD_LOGIC;
    direccion10_P42: OUT STD_LOGIC;
    direccion11_P47: OUT STD_LOGIC;
    direccion12_P65: OUT STD_LOGIC;
    direccion13_P51: OUT STD_LOGIC;
    direccion14_P54: OUT STD_LOGIC;
    direccion15_P64: OUT STD_LOGIC;
    direccion16_P63: OUT STD_LOGIC;
    direccion17_reset_P56: IN STD_LOGIC;
    datos0_ledS1_P39: INOUT STD_LOGIC;
    datos1_ledDP_P44: INOUT STD_LOGIC;
    datos2_ledS4_P46: INOUT STD_LOGIC;
    datos3_ledS6_P49: INOUT STD_LOGIC;
    datos4_ledS5_P57: INOUT STD_LOGIC;
    datos5_ledS3_P60: INOUT STD_LOGIC;
    datos6_ledS2_P62: INOUT STD_LOGIC;
    datos7_ledS0_P67: INOUT STD_LOGIC;
    SDOUT_P23: IN STD_LOGIC;
    MCLK_P77: OUT STD_LOGIC;
    linea_datos_P93: INOUT STD_LOGIC;
    linea_reloj_P94: INOUT STD_LOGIC
  );
END proyecto;
```

Esta entidad y su arquitectura describen el nivel más alto en la jerarquía del diseño en VHDL del "sistema autónomo para accionamiento por voz de un ratón de ordenador". El diseño VHDL se implementa sobre una FPGA para materializar el prototipo de un ratón de ordenador PS/2 controlado mediante la pronunciación de las vocales 'A', 'E', 'I', 'O' y 'U'.

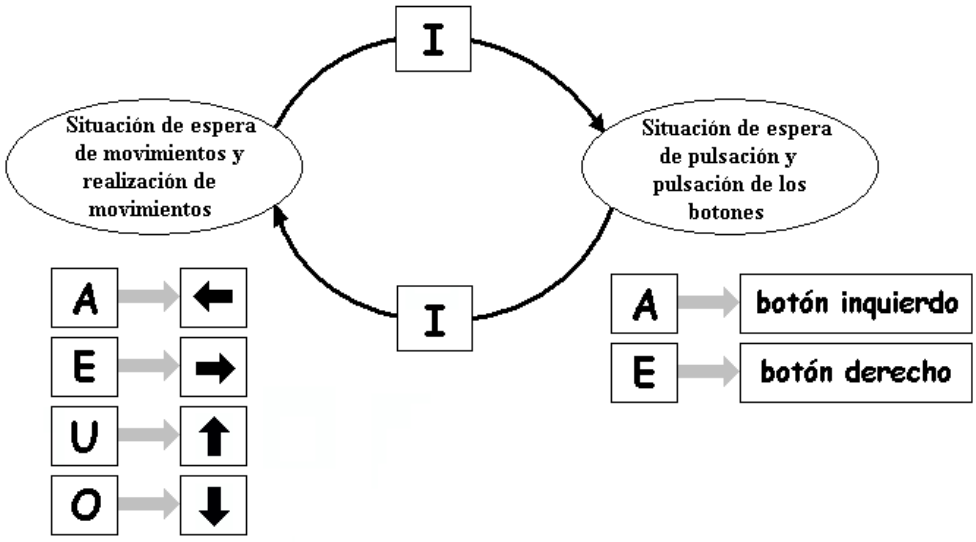
El diseño se corresponde con el prototipo de un ratón de ordenador PS/2 controlado por la voz que podrá ser utilizado por personas con incapacidad de movimiento para mover el puntero en la pantalla del ordenador y pulsar virtualmente los botones derecho e izquierdo del ratón sólo con la pronunciación de las vocales 'A', 'E', 'I', 'O' y 'U'. Será entonces necesario traducir las señales vocálicas generadas por el usuario en señales electrónicas aptas para ser procesadas en la FPGA. Un micrófono externo realiza la transducción de la señal de audio en una señal eléctrica que el conversor analógico-digital TLC320AD77C de Texas Instruments muestrea en datos digitales accesibles desde la FPGA. A partir de las muestras digitales y tras un proceso de reconocimiento, se detecta la vocal pronunciada, donde reside la información de la intención del usuario del dispositivo de acuerdo a un protocolo vocálico concertado para el uso del dispositivo. De

acuerdo con el protocolo vocálico, el dispositivo puede encontrarse realizando dos funcionalidades:

- Situación de espera de movimientos y realización de movimientos: arriba, abajo, izquierda y derecha.
- Situación de espera de pulsación y pulsación de los botones: botón derecho y botón izquierdo.

Tras el inicio del dispositivo se encontrará en la primera de las funcionalidades (Situación de espera de movimientos y realización de movimientos). En este momento, la pronunciación de la vocal "U" se traducirá en un movimiento efectivo del ratón hacia "ARRIBA". La pronunciación de la vocal "O" se traducirá en un movimiento efectivo del ratón hacia "ABAJO". La pronunciación de la vocal "E" se traducirá en un movimiento efectivo del ratón hacia la "DERECHA" y la pronunciación de la vocal "A" se traducirá en un movimiento efectivo del ratón hacia la "IZQUIERDA".

Si se pronuncia la vocal "I" se realizará un cambio de funcionalidad. Es decir, si se encuentra en situación de espera de movimientos y realización de movimientos, pasará a la situación de espera de pulsación y pulsación de los botones. Y a la inversa, si se encuentra en situación de espera de pulsación y pulsación de los botones, pasará a la situación de espera de movimientos y realización de movimientos.

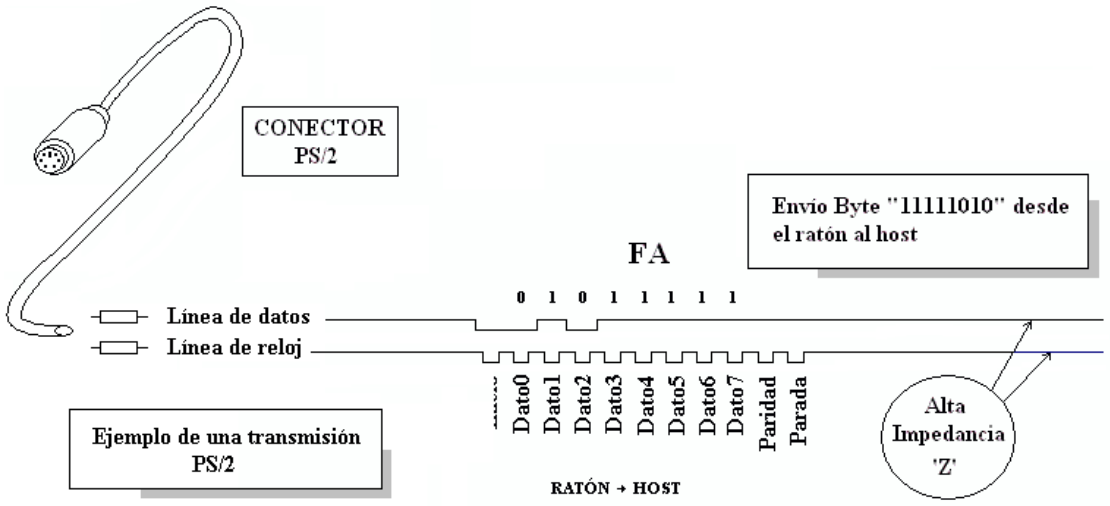


Cuando el dispositivo se encuentra en situación de espera de pulsación y pulsación de los botones, la pronunciación de la vocal "A" se traducirá en la pulsación efectiva del botón de la izquierda (CLICK). La pronunciación de la vocal "E" se traducirá en la pulsación efectiva del botón de la derecha (CLACK).

De esta forma, el usuario comunica al dispositivo sus intenciones de movimiento por medio de la pronunciación de las distintas vocales, que serán detectadas Cuando se detecten los eventos de movimiento (movimiento del puntero arriba, movimiento del puntero abajo, movimiento del

puntero a la derecha, movimiento del puntero a la izquierda, pulsación del botón derecho y pulsación del botón izquierdo) de acuerdo con el protocolo vocálico anterior se transmitirán al ordenador a través del protocolo serie PS/2. A través de la línea de datos del conector PS/2 de la tarjeta XSA se envían bytes de datos en la comunicación bidireccional con el ordenador. Sucesivamente se envían por la línea de datos y sincronizados con un reloj enviado a través de la línea de reloj del conector PS/2 de la tarjeta XSA: un bit de inicio ('0' lógico), los 8 bits del byte enviado empezando por el menos significativo, un bit de paridad impar asociado al byte enviado y un bit de parada ('1' lógico).

Mediante el protocolo serie el ordenador puede enviar comandos para provocar cambios en el funcionamiento del dispositivo o para pedir al dispositivo que le informe de determinadas variables internas. El dispositivo utiliza el protocolo serie para responder a los comandos recibidos desde el ordenador o para enviar datos que informen al ordenador de los eventos de movimiento producidos. De esta forma, y sólo con la pronunciación de las distintas vocales, el dispositivo transmite al ordenador las intenciones de movimiento o de pulsación virtual de los botones del ratón que tiene el usuario.

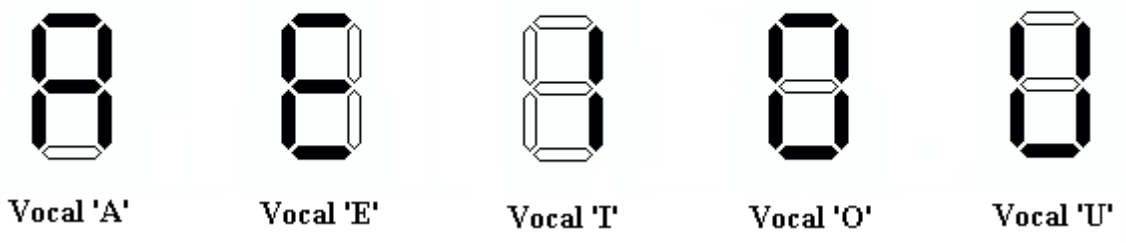


El núcleo básico del proyecto es el reconocimiento de las vocales pronunciadas a partir de las muestras digitales procedentes del conversor analógico-digital. Para detectar la vocal pronunciada se realiza la Transformada Rápida de Fourier a 512 muestras de la señal de audio.

Se realiza un escalado Mel para obtener unos coeficientes espectrales más eficaces y tras realizar el logaritmo en base 2 a los parámetros mel para desacoplar la aportación del tono generado por las cuerdas vocales de la aportación del tracto vocálico, se toman como parámetros más característicos de la vocal los ocho primeros coeficientes cepstrales obtenidos tras la realización de la Transformada Inversa de Fourier.

Una red neuronal de 32 neuronas con la topología de un mapa autoorganizado de 4 filas y 8 columnas detectará la vocal pronunciada en función de los coeficientes cepstrales captados. Los pesos sinápticos de las neuronas sintonizan con los coeficientes cepstrales de las distintas vocales de forma que al presentarse los coeficientes cepstrales se activará una única neurona de la red. Si la señal de audio supera el umbral de energía se considerará que se ha pronunciado la vocal asignada a la neurona ganadora. Cada neurona tiene asignada una vocal que será la vocal detectada en el caso de que en el proceso de reconocimiento se active esa neurona. La neurona que se activará será la que tenga los pesos sinápticos que mejor sintonicen con los coeficientes cepstrales de la señal captada. En esta fase, el display de leds mostrará la vocal reconocida para que el usuario compruebe el correcto funcionamiento del dispositivo.

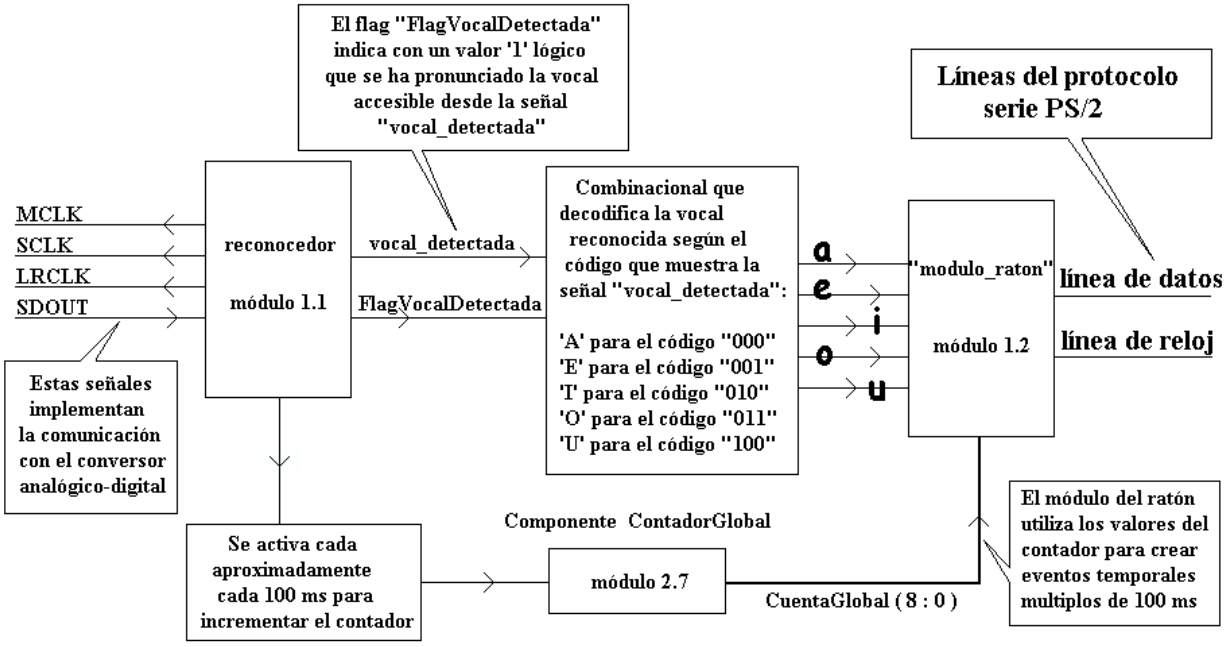
El display de leds de 7 segmentos mostrará la vocal detectada



El diseño en VHDL realizado implementa las funciones anteriores y permite realizar un prototipo operativo del dispositivo. Internamente, la funcionalidad del dispositivo se consigue mediante la interconexión de los siguientes módulos:

- **Módulo "Reconocedor" (Módulo 1.1)**: Este módulo controla el funcionamiento del conversor analógico-digital presente en el circuito al que se acopla la tarjeta XSA. Proporciona el reloj maestro con el que funcionará el conversor, el reloj de sincronismo de bit con el que el conversor transmitirá los datos en formato serie (bit a bit de forma sucesiva con los bits sincronizados con el reloj de bit) y el reloj que proporciona al conversor la frecuencia de muestreo con la que notificará al dispositivo emulador del ratón las muestras digitales de la señal analógica. En el seno del módulo se procesan las muestras digitales para generar los parámetros cepstrales. Los parámetros cepstrales resumen de forma muy acertada la información vocálica y conforman la base que se utilizará en el proceso de reconocimiento. Una vez detectada la pronunciación de una vocal, los coeficientes cepstrales excitarán a una red neuronal con la topología de un mapa autoorganizado de 32 neuronas. Cada neurona tiene asignada una vocal que será la vocal que se detecte si es la neurona que se activa. Se activará esa neurona en el caso de que los pesos sinápticos de la neurona sean los que mejor sintonicen con los coeficientes cepstrales captados. Éste es el objetivo del módulo, detectar si se ha pronunciado una vocal e informar de cuál ha sido la vocal pronunciada en ese caso.

- **Módulo "modulo_raton" (Módulo 1.2):** Extrae la información de movimiento de la vocal pronunciada según el protocolo vocálico definido anteriormente. Las pronunciaciones de las distintas vocales activarán los eventos de movimiento ('movimiento del puntero del ratón hacia arriba', 'movimiento del puntero del ratón hacia abajo', 'movimiento del puntero del ratón hacia la derecha', 'movimiento del puntero del ratón hacia la izquierda', 'pulsación del botón derecho del ratón' y 'pulsación del botón izquierdo del ratón') que serán transmitidos al ordenador a través del protocolo serie PS/2. Este módulo gestionará toda la comunicación con el ordenador, procesando los comandos recibidos desde el ordenador, generando las respuestas a estos comandos y, por supuesto, enviando los eventos de movimiento que el usuario pretende transmitir al ordenador a través de la pronunciación de las vocales.



- **Componente "ContadorGlobal" (Módulo 2.7)**: Este contador es utilizado por el módulo "módulo_ratón" para poder tener constancia de los tiempos globales transcurridos. De esta forma, el dispositivo ratón sincroniza los eventos que necesitan de un tiempo para su ejecución. El contador se incrementa aproximadamente cada 100 milisegundos, así puedo crear eventos en múltiplos temporales de 100 milisegundos.

- **Componente "RegistroFlagRecuerda" (Módulo 2.4)**: En este registro se almacena el flag "flag_recuerda", que indica con su valor la fase en la que se encuentra el dispositivo. Inicialmente, el valor del flag es '0' lógico y el dispositivo se encuentra en la fase de aprendizaje. En esta fase se almacenan de forma sucesiva los coeficientes cepstrales de las distintas vocales como patrones para detectar la vocal pronunciada en la fase de reconocimiento. Una vez terminado el aprendizaje de los patrones, el flag toma el valor '1' lógico y el sistema entra en la fase de reconocimiento de las vocales pronunciadas para activar

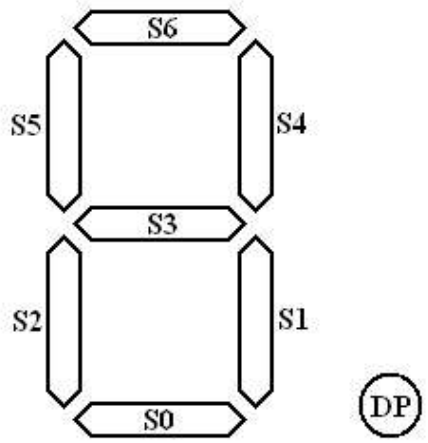
los eventos de movimiento que serán transmitidos al ordenador a través del protocolo serie PS/2.

INTERFAZ DEL MÓDULO:

RELOJ_P88: Señal de reloj de 200 ns utilizada como reloj el reloj maestro del sistema para el funcionamiento síncrono del dispositivo. A través de esta línea entra el reloj del sistema generado por el oscilador de frecuencia programable de la tarjeta XSA. Esta señal se localiza en el pin 88 de la FPGA.
CE_NEG_FLASH_P41: (CHIP ENABLE NEGADO) Para activar la memoria FLASH y posibilitar la lectura de los valores almacenados en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA esta señal debe estar activa con un valor '0' lógico. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew" y toma el valor '1' lógico cuando el componente "recnew" realiza las operaciones de reconocimiento para permitir el control del conversor analógico-digital. Esta señal se localiza en el pin 41 de la FPGA.
RESET_NEG_FLASH_59: Cuando toma el valor '1' lógico el dispositivo se encontrará en el modo estándar de operación. Cuando la señal toma el valor '0' lógico se detiene el funcionamiento normal del dispositivo y pone las salidas del dispositivo en el estado de alta impedancia. Esta señal tomará siempre el valor '1' lógico. Esta señal se localiza en el pin 59 de la FPGA.
OE_NEG_FLASH_43: (OUPUT ENABLE NEGADO) Los datos almacenados en la dirección de memoria indicada por el bus de direcciones son mostrados por el bus de datos cuando esta señal toma un valor '0' lógico estando habilitada la lectura de datos. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew" y toma el valor '1' lógico cuando el componente "recnew" realiza las operaciones de reconocimiento. Esta señal se localiza en el pin 43 de la FPGA.
WE_NEG_FLASH_58: (WRITE ENABLE NEGADO) Se habilita la escritura en la memoria FLASH cuando esta señal toma un valor '0' lógico. Se habilita la lectura de la memoria FLASH cuando esta señal toma el valor '1' lógico. Las señales "CE_NEG_FLASH" y "OE_NEG_FLASH" deben estar activas con un valor '0' lógico. Esta señal se localiza en el pin 58 de la FPGA.
DIRECCION0_P40: Bit número 0 de la dirección de la memoria FLASH desde donde se leerán los valores de los pesos sinápticos de las neuronas cuando se habilite la lectura de los datos de memoria ("WE_NEG_FLASH" con el valor '1' lógico). Las señales "CE_NEG_FLASH" y "OE_NEG_FLASH" deben estar activas con un valor '0' lógico. Los pesos se encuentran almacenados en el bloque de parámetros número uno de la memoria FLASH externa desde la dirección hexadecimal "04000" a la dirección hexadecimal "041FF". Esta señal se localiza en el pin 40 de la FPGA.
DIRECCION1_P29: Bit número 1 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 29 de la FPGA.

DIRECCION2_P28: Bit número 2 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 28 de la FPGA.
DIRECCION3_P27: Bit número 3 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 27 de la FPGA.
DIRECCION4_P74: Bit número 4 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 74 de la FPGA.
DIRECCION5_SCLK_P75: Esta señal toma el valor del bit número 5 de la dirección de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Toma el valor del reloj SCLK generado por el módulo que establece el sincronismo de bit de los datos transmitidos cuando el componente "recnew" realiza las operaciones de reconocimiento para permitir el control del conversor analógico-digital, lo cual ocurre tras la ejecución única e inicial del módulo "000" del componente "recnew". Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns. Esta señal se localiza en el pin 75 de la FPGA.
DIRECCION6_P76: Bit número 6 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 76 de la FPGA.
DIRECCION7_P66: Bit número 7 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 66 de la FPGA.
DIRECCION8_P50: Bit número 8 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 50 de la FPGA.
DIRECCION9_LRCLK_P48: Esta señal toma el valor del bit número 9 de la dirección de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". El bit número 9 de la dirección de la memoria FLASH tiene el valor '0' lógico. Toma el valor del reloj LRCLK que establece la frecuencia de muestreo de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo) cuando el componente "recnew" realiza las operaciones de reconocimiento para permitir el control del conversor analógico-digital, lo cual ocurre tras la ejecución única e inicial del módulo "000" del componente "recnew". Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit correspondiente de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio. Esta señal se localiza en el pin 48 de la FPGA.
DIRECCION10_P42: Bit número 10 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 42 de la FPGA.
DIRECCION11_P47: Bit número 11 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 47 de la FPGA.
DIRECCION12_P65: Bit número 12 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 65 de la FPGA.
DIRECCION13_P51: Bit número 13 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 51 de la FPGA.
DIRECCION14_P54: Bit número 14 de la dirección de la memoria FLASH. Toma el valor '1' lógico. Esta señal se localiza en el pin 54 de la FPGA.

DIRECCION15_P64: Bit número 15 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 64 de la FPGA.
DIRECCION16_P63: Bit número 16 de la dirección de la memoria FLASH. Toma el valor '0' lógico. Esta señal se localiza en el pin 63 de la FPGA.
DIRECCION17_RESET_P56: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo. El valor de esta señal se fija de forma externa mediante el interruptor número cuatro de la tarjeta XSA. El valor de esta señal también se corresponde con el bit número 17 de la dirección de la memoria FLASH. Esta señal se localiza en el pin 63 de la FPGA.
DATOS0_LEDS1_P39: Esta señal toma el valor del bit número 0 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". La lectura de datos debe estar habilitada con un valor '1' lógico en la señal "WE_NEG_FLASH". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S1" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada encendiéndose o no para que el display muestre la vocal 'A', 'E', 'I', 'O' o 'U' en el caso de que se detecte una vocal. Esta señal se localiza en el pin 39 de la FPGA.



DATOS1_LEDDP_P44: Esta señal toma el valor del bit número 1 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "DP" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 44 de la FPGA.
DATOS2_LEDS4_P46: Esta señal toma el valor del bit número 2 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S4" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 46 de la FPGA.

DATOS3_LEDS6_P49: Esta señal toma el valor del bit número 3 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S6" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 49 de la FPGA.
DATOS4_LEDS5_P57: Esta señal toma el valor del bit número 4 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S5" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 57 de la FPGA.
DATOS5_LEDS3_P60: Esta señal toma el valor del bit número 5 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S3" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 60 de la FPGA.
DATOS6_LEDS2_P62: Esta señal toma el valor del bit número 6 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S2" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 62 de la FPGA.
DATOS7_LEDS0_P67: Esta señal toma el valor del bit número 7 del bus de datos de la memoria FLASH cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew". Tras la ejecución única e inicial del módulo "000" del componente "recnew", toma el valor de la señal que activará el led "S0" del display de leds de la tarjeta XSA para informar visualmente de la vocal detectada. Esta señal se localiza en el pin 67 de la FPGA.
SDOUT_P23: Bit leído desde el conversor analógico-digital, correspondiente al bit actual de la muestra de 20 bits en complemento a dos que se está transmitiendo desde el conversor. Como el dato se transmite bit a bit a través de una misma línea de datos, a través de esta entrada se tendrá acceso al valor de la línea. Esta señal se localiza en el pin 23 de la FPGA.
MCLK_P77: Reloj maestro utilizado por el codificador analógico-digital generado en este módulo para el funcionamiento síncrono del sistema digital del conversor. Tiene un periodo de 200 ns. Esta señal se localiza en el pin 77 de la FPGA.
LINEA_DATOS_P93: Línea externa de comunicación con el ordenador en la que tanto el ordenador como el dispositivo transmitirán los datos de acuerdo con el protocolo serie PS/2 sincronizados con el reloj transmitido por la línea del reloj. Esta señal se localiza en el pin 93 de la FPGA.
LINEA_RELOJ_P94: Línea externa de comunicación con el ordenador en la que el dispositivo transmitirá el reloj que sincroniza los datos emitidos tanto por el ordenador como por el dispositivo por la línea de datos de acuerdo con el protocolo serie PS/2. Esta señal se localiza en el pin 94 de la FPGA.

1.1 reconocedor.vhd

```
ENTITY reconocedor IS
  PORT(
    reloj: IN STD_LOGIC;
    resetz: IN STD_LOGIC;
    CE_NEG_FLASH: OUT STD_LOGIC;
    RESET_NEG_FLASH: OUT STD_LOGIC;
    OE_NEG_FLASH: OUT STD_LOGIC;
    WE_NEG_FLASH: OUT STD_LOGIC;
    DIRECCION_FLASH: OUT STD_LOGIC_VECTOR(8 DOWNTO 0);
    DATOS_FLASH: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    SDOUT: IN STD_LOGIC;
    MCLK: OUT STD_LOGIC;
    SCLK: OUT STD_LOGIC;
    LRCLK: OUT STD_LOGIC;
    vocal_detectada: OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
    flag_vocal_detectada: OUT STD_LOGIC;
    incrementa_contador_global: OUT STD_LOGIC
  );
END reconocedor;
```

Este componente se encarga de poner en funcionamiento el conversor analógico-digital externo a la FPGA para así recibir como datos las muestras digitales de la señal de voz del usuario del prototipo definido en el proyecto. A partir de las muestras digitales recibidas, el componente inicia los procesos de reconocimiento y activa las señales que indican de forma externa si se ha pronunciado una vocal y cuál ha sido la vocal pronunciada.

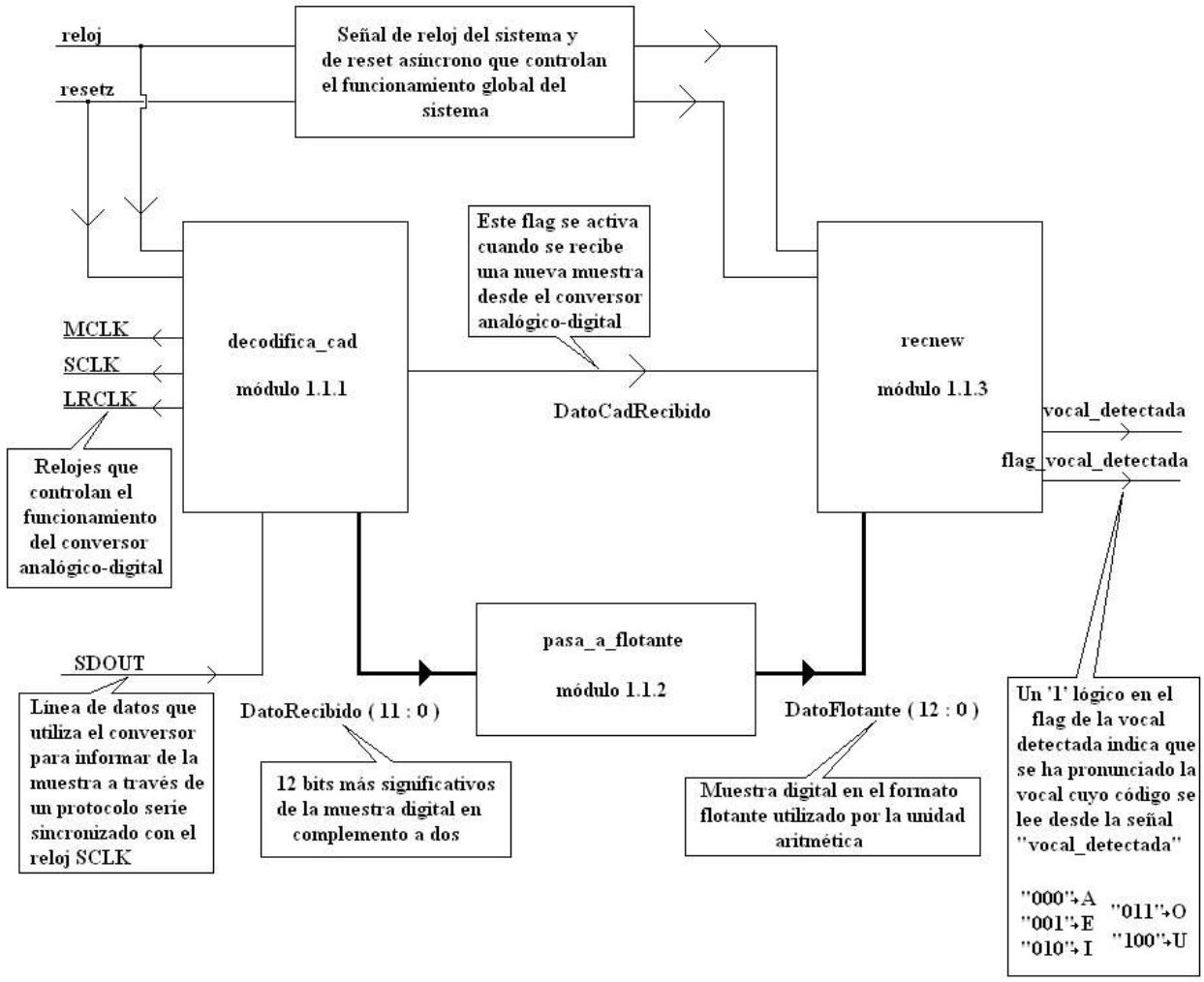
El prototipo del proyecto es un emulador de un ratón PS/2 accionado por voz. El usuario que maneje el dispositivo pronunciará las vocales 'A', 'E', 'I', 'O' y 'U' para accionar los eventos de movimiento en el ratón. El interfaz con el dispositivo es el canal de voz. Un micrófono transforma la señal vocálica en señales eléctricas que el conversor analógico-digital convierte en datos digitales. El usuario emula los movimientos del ratón y la pulsación de los botones pronunciando las distintas vocales de acuerdo con un protocolo. Un sistema de reconocimiento basado en una red neuronal detecta la vocal pronunciada y activa las señales de movimiento asociadas de acuerdo con el protocolo usado. En ese instante, el sistema emulará a un ratón e informará de los eventos de movimiento al ordenador a través del protocolo serie PS/2.

El componente “reconocedor” es el encargado de extraer la información vocálica de las muestras digitales de la señal de audio procedentes del conversor analógico-digital. Esta es el objetivo del sistema de reconocimiento, detectar la vocal pronunciada para así poder activar los movimientos de ratón asociados. Para saber cuál es la vocal pronunciada se realiza la Transformada Rápida de Fourier a 512 muestras digitales de audio. A partir de estos coeficientes espectrales se obtienen los coeficientes cepstrales, que resumen de una forma más apropiada la información vocálica de la señal.

Una red neuronal de 32 neuronas detecta la vocal pronunciada. Todas las neuronas son excitadas por los coeficientes cepstrales, pero sólo una de las neuronas se activará. Como cada neurona tiene asignada una vocal en el caso de que la señal de audio supere el umbral de energía, se decidirá que se ha pronunciado la vocal que se corresponde con la vocal asignada a la neurona. Si

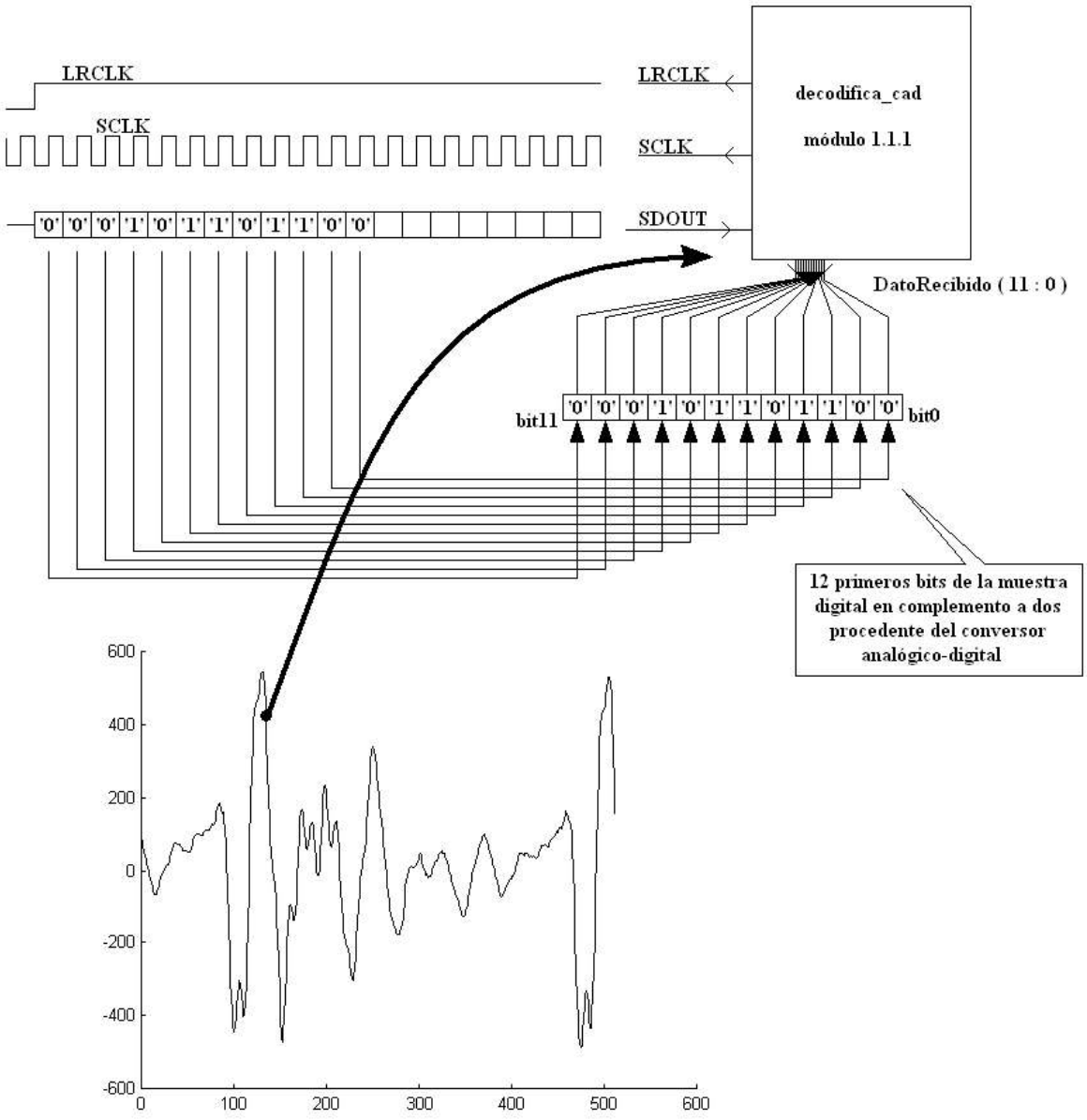
se detecta la pronunciación de una vocal se activará un flag "flag_vocal_detectada" con un valor '1' lógico para indicar al sistema encargado de traducir la vocal pronunciada en eventos de movimiento que se ha pronunciado la vocal representada por el código accesible desde la señal de salida "vocal_detectada" ("000" para la vocal 'A', "001" para la vocal 'E', "010" para la vocal 'I', "011" para la vocal 'O' y "100" para la vocal 'U').

El proceso de reconocimiento se consigue mediante la interconexión de una serie de módulos:



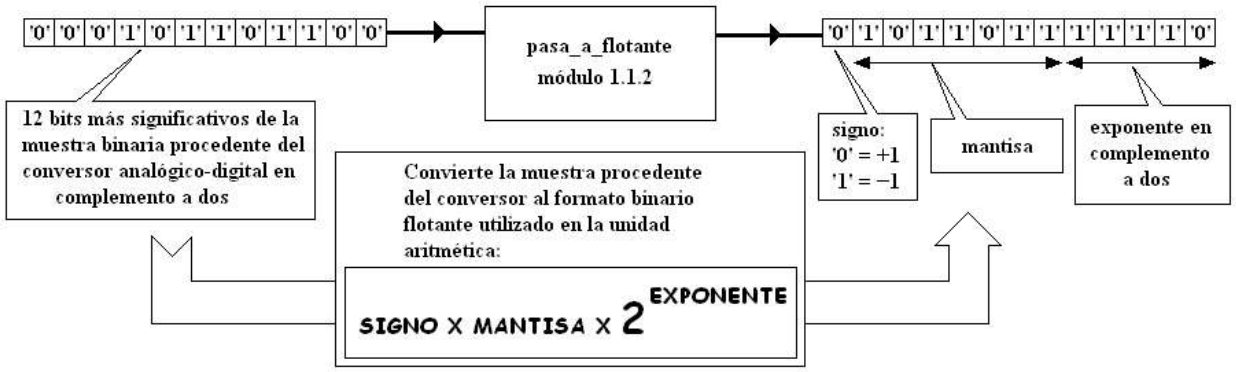
- **Módulo "decodifica_cad" (1.1.1)**: Este módulo se encarga de controlar el codificador analógico-digital para la recepción de las muestras digitales de la señal de audio. Genera los 3 relojes necesarios para el funcionamiento del conversor: El reloj MCLK es el reloj maestro que proporciona el sincronismo para el funcionamiento del conversor analógico-digital, tiene un periodo de 200 ns. El reloj SCLK es utilizado para sincronizar los bits en la transmisión de la muestra digital a través del protocolo serie utilizado por el dispositivo, tiene un periodo de

800 ns. El reloj LRCLK es utilizado para fijar la frecuencia de muestreo de las distintas muestras de la señal analógica, tiene un periodo de 51.2 us. Cada 51.2 us el conversor analógico-digital transmitirá a la FPGA una muestra digital de 20 bits en complemento a dos de la señal de audio a través de un protocolo serie de bits sincronizados con el reloj SCLK.

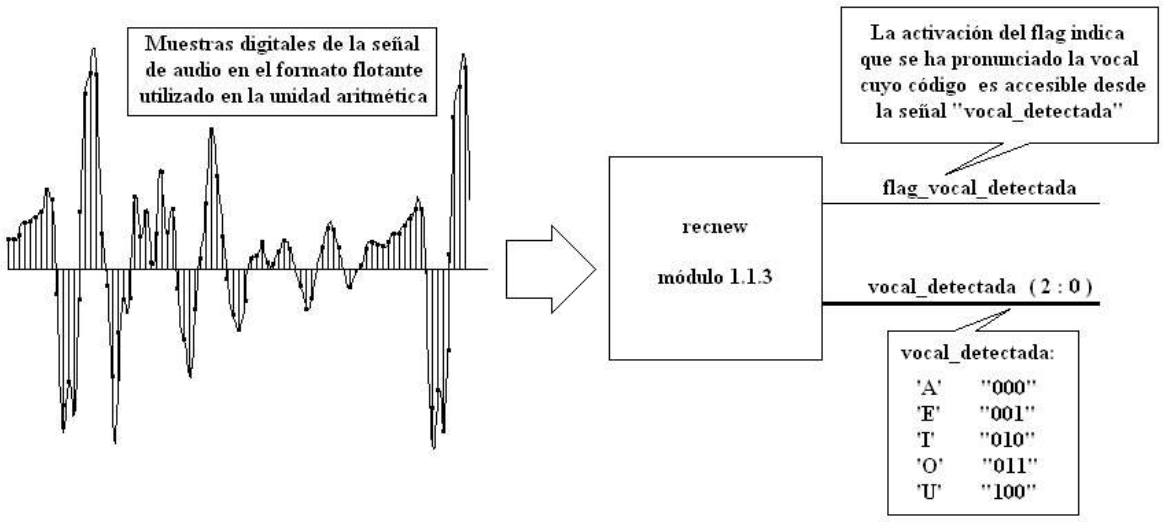


- **Módulo "pasa_a_flotante" (1.1.2)**: En el proceso de reconocimiento, para trabajar con la máxima precisión, utilizo una unidad aritmética que trabaja con datos flotantes. De esta forma evito trabajar con largas cadenas de bits y reduzco la posibilidad de desbordamiento en los resultados de las multiplicaciones. Aumenta la complejidad de la unidad aritmética lógica, pero los módulos que la utilizan salen beneficiados por el tratamiento uniforme que se da a los

datos numéricos, siempre representados por 13 bits. El formato numérico de la representación de los datos con los que se trabajará en el sistema reconocedor, y por lo tanto, en la unidad aritmética es el siguiente: El bit más significativo es el signo del número, los siguientes 7 bits conforman la mantisa del número, es decir, el módulo binario del número que se multiplicará por 2 elevado a la potencia indicada por el exponente y los últimos 5 bits conforman el valor del exponente representando a un entero binario en complemento a dos. Es el exponente en base dos por el que habrá que multiplicar la mantisa para definir el módulo del número. El módulo "pasa_a_flotante" transcodifica la muestra digital de entrada desde el conversor analógico-digital (entero de 12 bits en complemento a dos) al formato numérico flotante global utilizado en la unidad aritmética.



- **Módulo "recnew" (1.1.3)**: Módulo que realiza el proceso de reconocimiento vocálico de una forma integral. Tras recibir 512 muestras desde el conversor analógico-digital, se iniciaría el proceso de obtención de los parámetros cepstrales. En función de esos parámetros cepstrales se decidirá si se ha pronunciado una vocal y cuál ha sido esa vocal. La vocal detectada se traducirá posteriormente en un evento de movimiento que el módulo "modulo_raton" transmitirá al ordenador.



El proceso de reconocimiento de la vocal pronunciada empieza con la obtención de 512 muestras de la señal de audio procedentes del conversor analógico-digital. Seguidamente se calcula la Transformada Rápida de Fourier de esas 512 muestras para pasar al dominio frecuencial, donde las características vocálicas son más evidentes. Se obtiene el módulo de los coeficientes de Fourier ya que la información aportada por la fase no es importante. Un escalado mel en el cuadrado del módulo de los coeficientes de Fourier sintetiza la energía de las diversas frecuencias pronunciadas con las vocales de una forma más semejante a cómo lo hace el oído humano. Al realizar el logaritmo en base 2 a los coeficientes mel se desacopla la información aportada por el tono (frecuencia de vibración de las cuerdas vocales) de la aportada por el tracto vocálico (la que realmente contiene la información útil para la detección vocálica). Al realizar la Transformada Inversa de Fourier se obtienen los coeficientes cepstrales. Los coeficientes cepstrales bajos resumen la información del tracto vocálico y los coeficientes cepstrales altos sintetizan la información del tono. Los primeros coeficientes cepstrales se utilizan como parámetros de entrada a la red neuronal de mapas autoorganizados de 32 neuronas que realiza el reconocimiento vocálico. Una neurona se activará al excitar la red neuronal con los parámetros cepstrales, y propondrá como vocal pronunciada la vocal asignada a la neurona de acuerdo con la topología fijada en el mapa autoorganizado en el caso de que la señal de audio supere el umbral de energía.

INTERFAZ DEL MÓDULO:

RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.
RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CE_NEG_FLASH: (CHIP ENABLE NEGADO) Para activar la memoria FLASH y posibilitar la lectura de los valores almacenados en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA esta señal debe estar activa con un valor '0' lógico. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo “000” del componente “recnew” y toma el valor '1' lógico cuando el componente “recnew” realiza las operaciones de reconocimiento para permitir el control del conversor analógico-digital.
RESET_NEG_FLASH: Cuando toma el valor '1' lógico el dispositivo se encontrará en el modo estándar de operación. Cuando la señal toma el valor '0' lógico se detiene el funcionamiento normal del dispositivo y pone las salidas del dispositivo en el estado de alta impedancia. Esta señal tomará siempre el valor '1' lógico.
OE_NEG_FLASH: Los datos almacenados en la dirección de memoria indicada por el bus de direcciones son mostrados por el bus de datos cuando esta señal toma un valor '0' lógico estando habilitada la lectura de datos. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo “000” del componente “recnew” y toma el valor '1' lógico cuando el componente “recnew” realiza las operaciones de reconocimiento.
WE_NEG_FLASH: Se habilita la escritura en la memoria FLASH cuando esta señal toma un valor '0' lógico. Se habilita la lectura de la memoria FLASH cuando esta señal toma el valor '1' lógico. Las señales “CE_NEG_FLASH” y “OE_NEG_FLASH” deben estar activas con un valor '0' lógico.

DIRECCION_FLASH: Dirección de la memoria FLASH desde donde se leerán los valores de los pesos sinápticos de las neuronas cuando se habilite la lectura de los datos de memoria ("WE_NEG_FLASH" con el valor '1' lógico). También es la dirección de la memoria FLASH donde se almacenarán los valores de los pesos sinápticos de las neuronas cuando se habilite la escritura de los datos en memoria ("WE_NEG_FLASH" con el valor '0' lógico). Las señales "CE_NEG_FLASH" y "OE_NEG_FLASH" deben estar activas con un valor '0' lógico. Los pesos se encuentran almacenados en el bloque de parámetros número uno de la memoria FLASH externa desde la dirección hexadecimal "04000" a la dirección hexadecimal "041FF".
DATOS_FLASH: Bus de datos que mostrará el byte almacenado en la memoria principal cuando esté habilitada la lectura de datos ("WE_NEG_FLASH" con el valor '1' lógico). También es el bus de datos con el byte que se almacenará en la memoria principal cuando esté habilitada la escritura de datos ("WE_NEG_FLASH" con el valor '0' lógico). Las señales "CE_NEG_FLASH" y "OE_NEG_FLASH" deben estar activas con un valor '0' lógico.
SDOUT: Bit leído desde el conversor analógico-digital, correspondiente al bit actual de la muestra de 20 bits en complemento a dos que se está transmitiendo desde el conversor. Como el dato se transmite bit a bit a través de una misma línea de datos, a través de esta entrada se tendrá acceso al valor de la línea.
MCLK: reloj maestro del codificador analógico-digital generado en este módulo para el funcionamiento síncrono del sistema digital del conversor. Tiene un periodo de 200 ns.
SCLK: reloj generado por el módulo que establece el sincronismo de bit de los datos transmitidos. Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns.
LRCLK: reloj que establece la frecuencia de muestreo de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo). Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit correspondiente de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio.
VOCAL_DETECTADA: Código de la vocal que se ha detectado en el proceso de reconocimiento. La vocal detectada se traducirá posteriormente en un evento de movimiento que el módulo "modulo_raton" transmitirá al ordenador. El código "000" se corresponde con la vocal 'A', el código "001" se corresponde con la vocal 'E', el código "010" se corresponde con la vocal 'I', el código "011" se corresponde con la vocal 'I' y el código "100" se corresponde con la vocal 'U'.
FLAG_VOCAL_DETECTADA: Este flag indica cuando está activo con un valor uno lógico que la energía de la señal de audio ha superado el umbral y por lo tanto, los parámetros cepstrales se traducirán en una vocal cuyo código será "vocal_detectada". Este código se corresponde con la vocal reconocida y el flag indica que se ha pronunciado una vocal.
INCREMENTA_CONTADOR_GLOBAL: Se activa durante 200 ns aproximadamente cada 100 ms. Esta señal incrementa un contador externo aproximadamente cada 100 milisegundos. Este contador externo es utilizado para poder tener constancia de los tiempos globales transcurridos. De esta forma, el dispositivo ratón sincroniza los eventos que necesitan de un tiempo para su ejecución, así puedo crear eventos en múltiplos temporales de 100 milisegundos.

1.1.1 decodifica_CAD.vhd

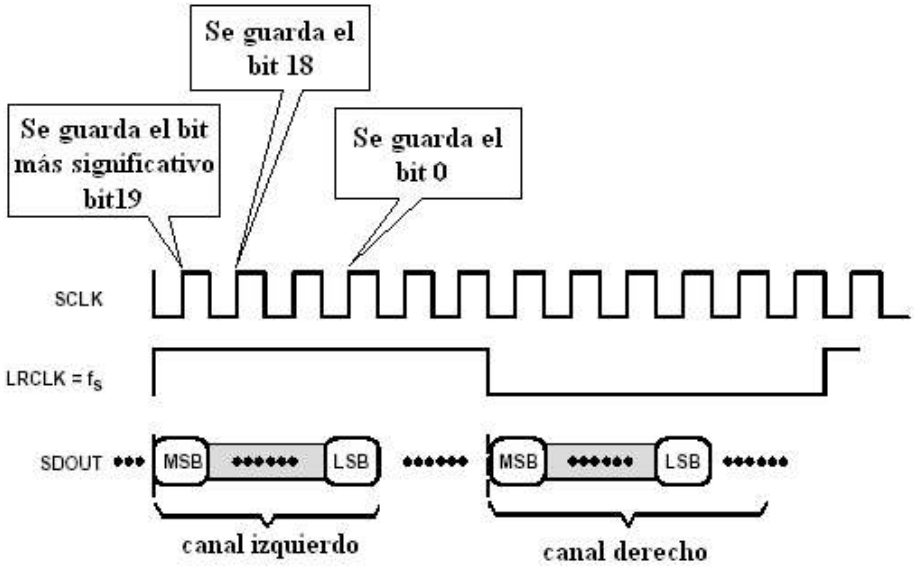
```
ENTITY decodifica_CAD IS
  PORT(
    reloj: IN STD_LOGIC;
    resetz: IN STD_LOGIC;
    SDOUT: IN STD_LOGIC;
    dato_cad_recibido: OUT STD_LOGIC;
    MCLK: OUT STD_LOGIC;
    SCLK: OUT STD_LOGIC;
    LRCLK: OUT STD_LOGIC;
    dato_recibido: OUT STD_LOGIC_VECTOR(11 DOWNTO 0)
    );
END decodifica_CAD;
```

Módulo encargado del control del codificador analógico-digital para la recepción de las muestras digitales de la señal de audio.

El codificador analógico-digital utiliza un protocolo para enviar al dispositivo la muestra de la señal analógica. Cada muestra obtenida por el conversor analógico digital es enviada al dispositivo reconocedor para que sea procesada. La muestra está codificada en 20 bits en complemento a dos. Este dato se recibe bit a bit (serie), por lo tanto, el dato debe ser también guardado bit a bit de forma secuencial, según sea emitido por el conversor. Una vez terminada la captura, la muestra será accesible a través del vector de bit "dato_recibido". El flag "dato_cad_recibido" avisará a las unidades encargadas del reconocimiento vocálico activándose con un valor uno lógico cuando la muestra haya sido recibida y guardada.

Un reloj maestro generado por el dispositivo proporciona el reloj de sistema para el funcionamiento síncrono del conversor. La frecuencia con la que se realizará el muestreo de la señal de audio es Fs=19531 Hz. El reloj maestro trabaja con una frecuencia 256*Fs. El reloj LRCLK proporciona la frecuencia de muestreo (19531 Hz). Cuando se inicia un nivel lógico alto en el reloj LRCLK, comienza el envío serie de la palabra de 20 bits en complemento a dos de la señal muestreada por el canal izquierdo de audio. El reloj LRCLK proporciona la frecuencia de muestreo (19531 Hz). Cuando se inicia un nivel lógico alto en el reloj LRCLK, comienza el envío serie de la palabra de 20 bits en complemento a dos de la señal muestreada por el canal izquierdo de audio.

Cada vez que se modifica el nivel del reloj LRCLK se inicia la transmisión de una muestra. En el primer flanco de bajada del reloj de sincronismo de bit SCLK se iniciará la transmisión del bit más significativo de la muestra (bit19) y su valor se mantendrá hasta el segundo flanco de bajada del reloj SCLK. En ese momento se inicia la transmisión del bit 18 de la muestra, que se mantiene hasta el siguiente flanco de bajada del reloj SCLK. Este proceso continúa hasta la emisión completa de la muestra. Al estar formada cada muestra por 20 bits y como la trama del reloj LRCLK que fija la frecuencia de muestreo está compuesta por 64 pulsos de reloj SCLK, sobrarán pulsos de reloj SCLK en la transmisión del dato.



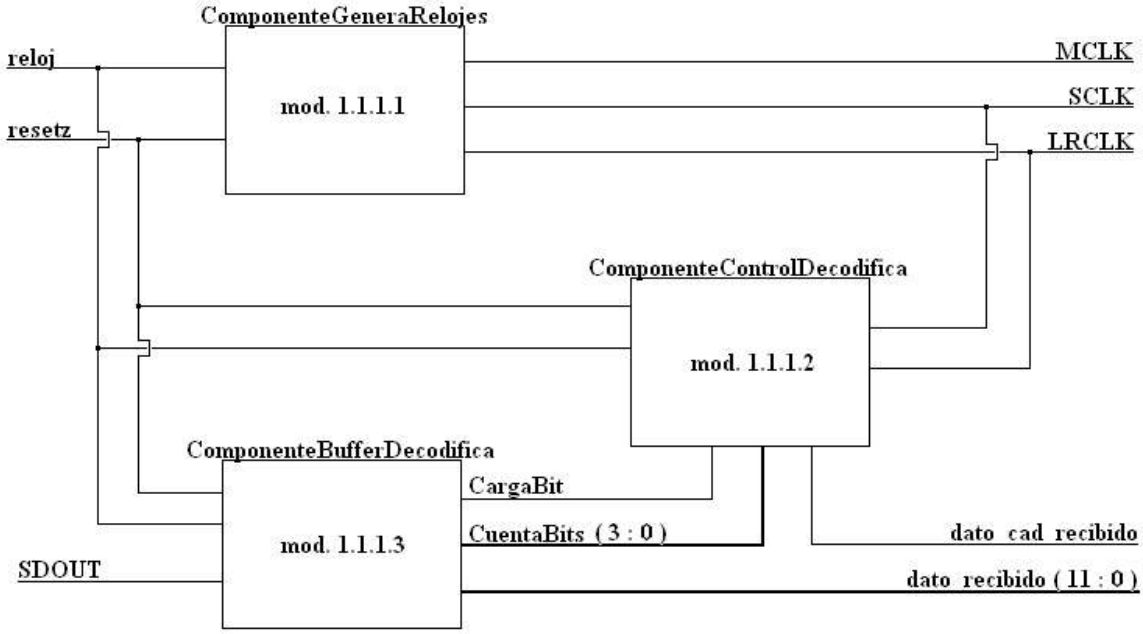
El proceso de recepción de la muestra se consigue gracias a la interconexión de los siguientes módulos:

- **MÓDULO "GENERA_RELOJES" 1.1.1.1**:
  En esta unidad se generan las señales de reloj necesarias para el funcionamiento del conversor analógico-digital.
  El reloj MCLK es el reloj maestro. Todos los dispositivos síncronos funcionan con este reloj y es el reloj del sistema digital del conversor analógico digital. Tiene un periodo de 200 ns.
  El reloj SCLK es utilizado para sincronizar los bits en la transmisión serie. Tiene un periodo de 800 ns.
  El reloj LRCLK es utilizado para fijar la frecuencia de muestreo de las distintas muestras de la señal analógica. Tiene un periodo de 51.2 us.

- **MÓDULO "CONTROL_DECODIFICA" 1.1.1.2**:
  Este módulo sincroniza la recepción de la muestra digital de 20 bits del conversor analógico-digital de acuerdo con los relojes SCLK y LRCLK. Cada vez que se modifica el nivel del reloj LRCLK se inicia la transmisión de una muestra. En el primer flanco de bajada del reloj de sincronismo de bit SCLK se iniciará la transmisión del bit más significativo de la muestra (bit19) y su valor se mantendrá hasta el segundo flanco de bajada del reloj SCLK. El módulo "control_decodifica" es el encargado de generar las señales encargadas de guardar la muestra recibida para que sea posteriormente procesada y utilizada para el reconocimiento vocálico. En el módulo "buffer_decodifica" se encuentran los registros encargados del almacenamiento. Para recorrer uno a uno los bits recibidos en cada flanco de bajada del reloj se dispone de un contador que se incrementa sucesivamente para indexar los registros que guardarán cada uno de los bits. Como cada uno de los bits de la muestra se introduce en la línea en el flanco de

bajada del reloj SCLK, en el flanco de subida del reloj SCLK el dato será estable y será en este momento cuando se active la carga del bit en el registro correspondiente indexado por el contador.

- **MÓDULO "BUFFER_DECODIFICA" 1.1.1.3**:
  En este módulo se almacenan los 12 bits más representativos de la muestra en complemento a dos, para que desde aquí sea accesible por el reconocedor. Los bits se guardarán uno a uno en sus registros correspondientes a medida que se vayan recibiendo. Una lógica combinacional selecciona el registro donde se guardará el bit actual de acuerdo con el valor del contador de bit en el momento en el que se active la carga del bit recibido.



INTERFAZ DEL MÓDULO:

RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.
SDOUT: Bit leído desde el conversor analógico-digital, correspondiente al bit actual de la muestra de 20 bits en complemento a dos que se está transmitiendo desde el conversor. Como el

dato se transmite bit a bit a través de una misma línea de datos, a través de esta entrada se tendrá acceso al valor de la línea.

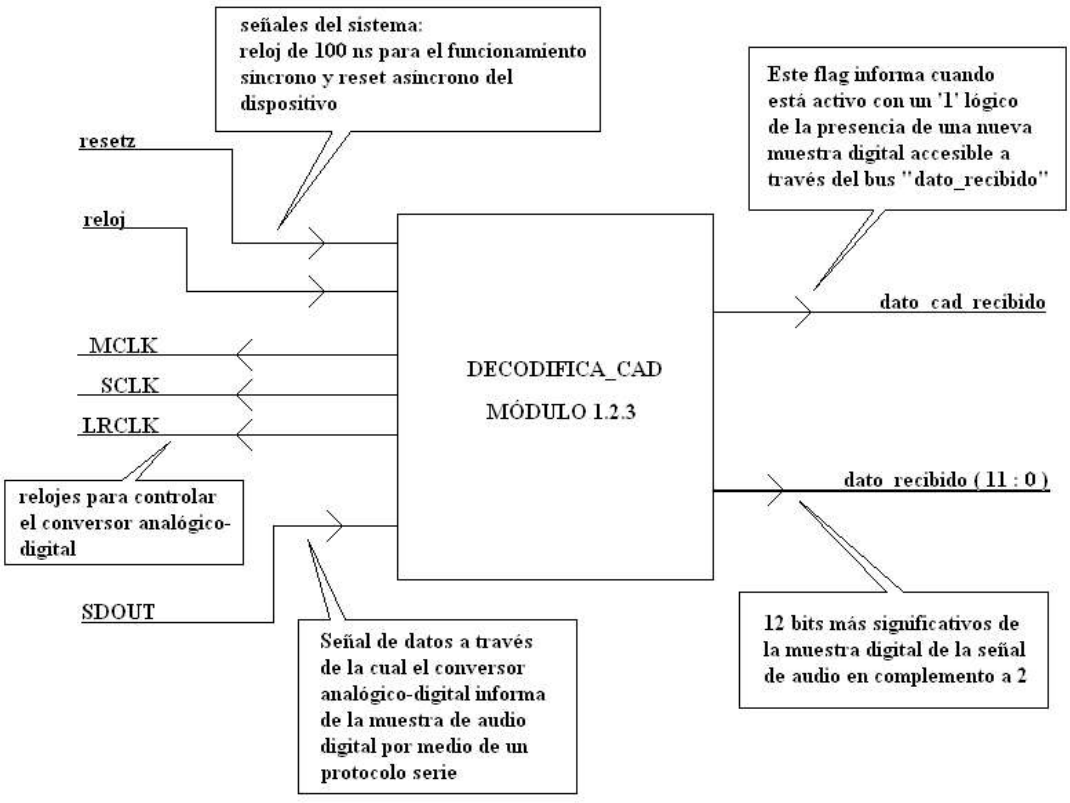
DATO_CAD_RECIBIDO: Esta señal indica a unidades externas y ajenas al control del conversor analógico-digital que acaba de recibirse una nueva muestra de la señal de audio analógica. El dato muestra está almacenado y puede ser leído por las unidades encargadas del reconocimiento vocálico.

RELOJ_MCLK: reloj maestro del codificador analógico-digital generado en este módulo para el funcionamiento síncrono del sistema digital del conversor. Tiene un periodo de 200 ns.

RELOJ_SCLK: reloj generado por el módulo que establece el sincronismo de bit de los datos transmitidos. Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns.

RELOJ_LRCLK: reloj que establece la frecuencia de muestreo de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo). Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit correspondiente de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio.

DATO_RECIBIDO: Valor de la señal de audio muestreada por el conversor analógico-digital que será accesible por la unidad de reconocimiento de voz para su procesamiento y así realizar la detección de la vocal pronunciada. La información de la muestra vocálica es accesible a través de los 12 bits en complemento a dos del vector de bits "dato_recibido".

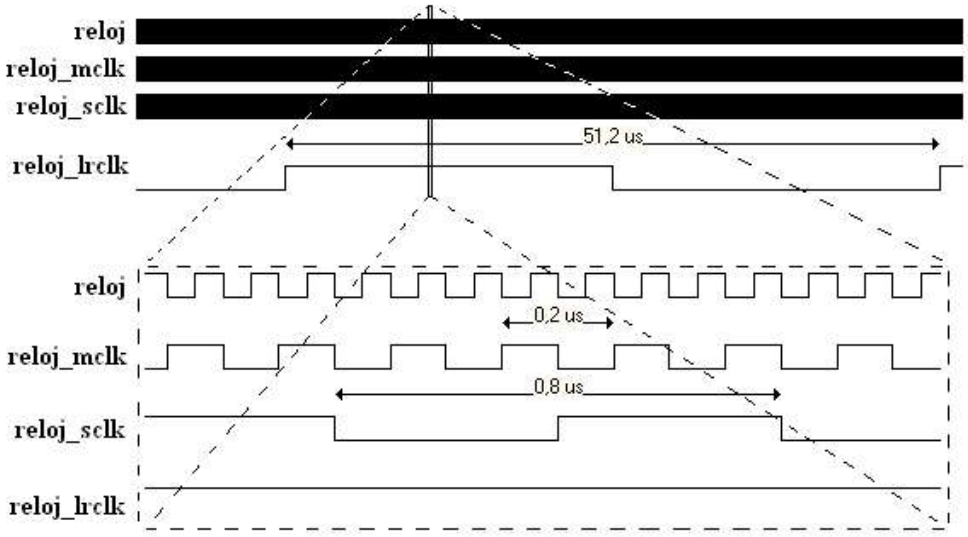


1.1.1.1 genera_relojes.vhd

```
ENTITY genera_relojes IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    reloj_mclk: OUT STD_LOGIC;
    reloj_sclk: OUT STD_LOGIC;
    reloj_lrclk: OUT STD_LOGIC
  );
END genera_relojes;
```

En esta unidad se generan las señales de reloj necesarias para el funcionamiento del conversor analógico-digital.

El conversor analógico-digital utiliza un protocolo serie para transmitir la muestra digital de la señal analógica. El dispositivo que utilice el conversor debe generar los relojes con los que el conversor analógico digital funciona.

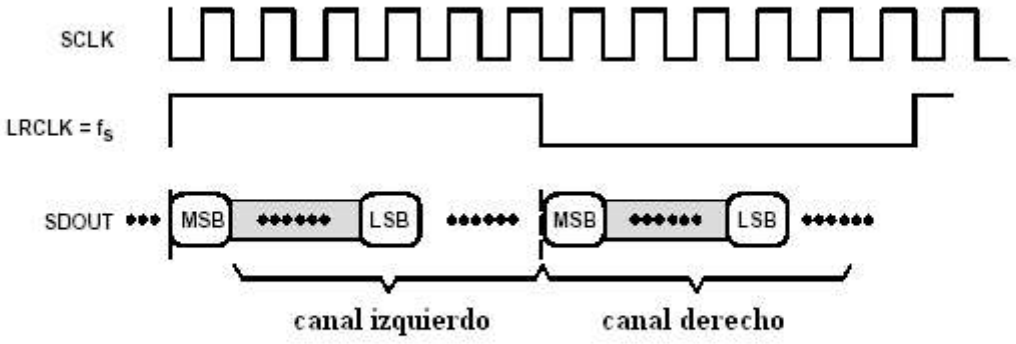


- El reloj MCLK es el reloj maestro. Todos los dispositivos síncronos funcionan con este reloj y es el reloj del sistema digital del conversor analógico digital. Tiene un periodo de 200 ns.
- El reloj SCLK es utilizado para sincronizar los bits en la transmisión serie. Tiene un periodo de 800 ns.
- El reloj LRCLK es utilizado para fijar la frecuencia de muestreo de las distintas muestras de la señal analógica. Tiene un periodo de 51.2 us.

Cada 51.2 us se genera una muestra digital de 20 bits en complemento a dos. Cuando el nivel del reloj LRCLK es alto, los 20 bits se corresponden con el canal izquierdo de una señal estéreo de audio. Cuando el nivel del reloj LRCLK es bajo, los 20 bits se corresponden con el canal derecho. En cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. El dispositivo implementado puede leer bit a bit la muestra digital transmitida de un sólo canal, ya que se utilizará sólo un micrófono. Se realiza de esta forma la transcodificación de la señal analógica de audio en una señal digital muestreada.

INTERFAZ DEL MÓDULO:

RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.
RELOJ_MCLK: reloj maestro del codificador analógico-digital generado en este módulo para el funcionamiento síncrono del sistema digital del conversor. Tiene un periodo de 200 ns.
RELOJ_SCLK: reloj generado por el módulo que establece el sincronismo de bit de los datos transmitidos. Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns.
RELOJ_LRCLK: reloj que establece la frecuencia de muestreo de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo). Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit correspondiente de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio.



1.1.1.2 control_decodifica.vhd

```
ENTITY control_decodifica IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    SCLK: IN STD_LOGIC;
    LRCLK: IN STD_LOGIC;
    carga_bit: OUT STD_LOGIC;
    cuenta_bits: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
    dato_cad_recibido: OUT STD_LOGIC
  );
END control_decodifica;
```

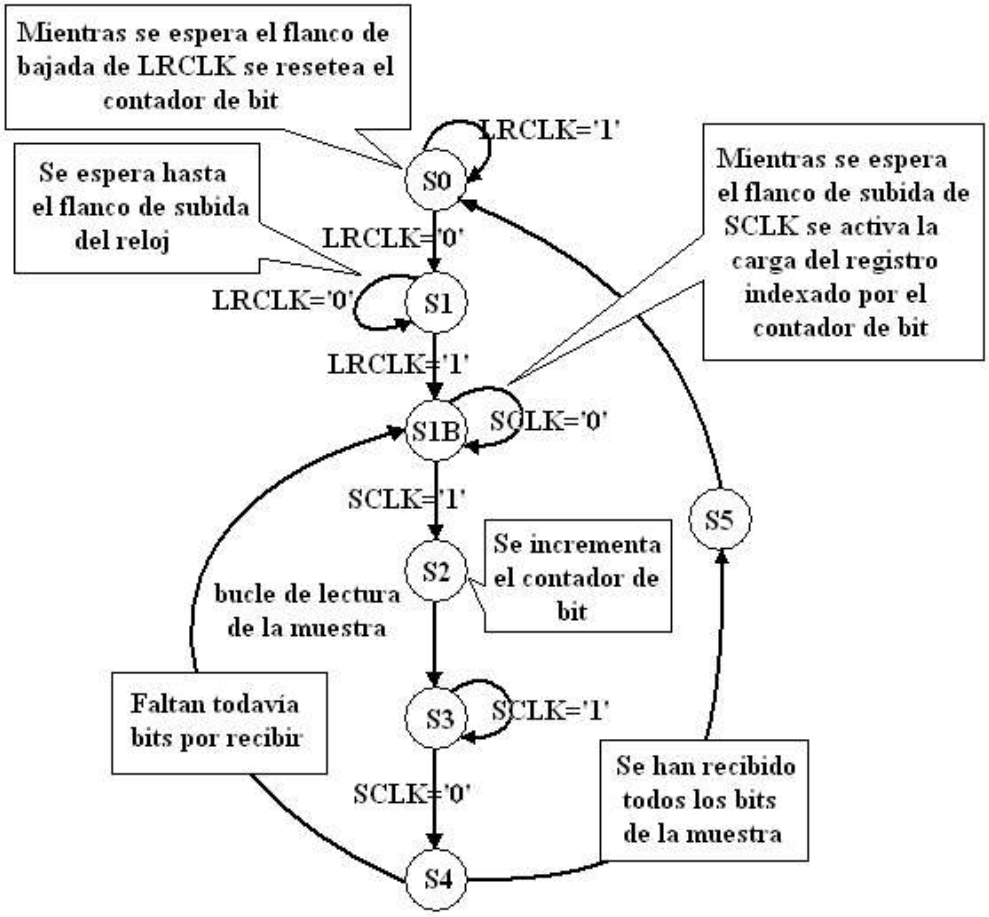
Este módulo sincroniza la recepción de la muestra digital de 20 bits del conversor analógico-digital de acuerdo con los relojes SCLK y LRCLK.

La muestra digital de la señal analógica es transmitida al dispositivo reconocedor mediante un protocolo especial. Un reloj maestro generado por el dispositivo proporciona el reloj de sistema para el funcionamiento síncrono del conversor. La frecuencia con la que se realizará el muestreo de la señal de audio es Fs=19531 Hz. El reloj maestro trabaja con una frecuencia 256*Fs. El reloj LRCLK proporciona la frecuencia de muestreo (19531 Hz). Cuando se inicia un nivel lógico alto en el reloj LRCLK, comienza el envío serie de la palabra de 20 bits en complemento a dos de la señal muestreada por el canal izquierdo de audio. El reloj LRCLK proporciona la frecuencia de muestreo (19531 Hz). Cuando se inicia un nivel lógico alto en el reloj LRCLK, comienza el envío serie de la palabra de 20 bits en complemento a dos de la señal muestreada por el canal izquierdo de audio.

Cada vez que se modifica el nivel del reloj LRCLK se inicia la transmisión de una muestra. En el primer flanco de bajada del reloj de sincronismo de bit SCLK se iniciará la transmisión del bit más significativo de la muestra (bit19) y su valor se mantendrá hasta el segundo flanco de bajada del reloj SCLK. En ese momento se inicia la transmisión del bit 18 de la muestra, que se mantiene hasta el siguiente flanco de bajada del reloj SCLK. Este proceso continúa hasta la emisión completa de la muestra. Al estar formada cada muestra por 20 bits y como la trama del reloj LRCLK que fija la frecuencia de muestreo está compuesta por 64 pulsos de reloj SCLK, sobrarán pulsos de reloj SCLK en la transmisión del dato.

El módulo “control_decodifica” es el encargado de generar las señales encargadas de guardar la muestra recibida para que sea posteriormente procesada y utilizada para el reconocimiento vocálico. En el módulo “buffer_decodifica” se encuentran los registros encargados del almacenamiento. Para recorrer uno a uno los bits recibidos en cada flanco de bajada del reloj se dispone de un contador que se incrementa sucesivamente para indexar los registros que guardarán cada uno de los bits. Como cada uno de los bits de la muestra se introduce en la línea en el flanco de bajada del reloj SCLK, en el flanco de subida del reloj SCLK el dato será estable y será en este momento cuando se active la carga del bit en el registro correspondiente indexado por el contador.

Una máquina de estados controla todo el proceso anterior:



- **ESTADO S0:** En este estado se inicia la recepción de una muestra desde el conversor analógico-digital. Por eso se resetea el contador que lleva la cuenta del bit recibido. Este contador recorre uno a uno los bits de la muestra activando el registro que guardará el bit actual que se está recibiendo. Se esperará en este estado hasta que el reloj LRCLK tome el valor lógico '0', cuando el conversor empieza a transmitir una nueva muestra, entonces pasará al estado S1.
- **ESTADO S1**: En este estado se espera hasta que el reloj LRCLK tome de nuevo el valor lógico '1', entonces pasará al estado S1B.
- **ESTADO S1B**: Ya ha empezado la recepción de un bit y se activará la carga del bit en el registro seleccionado por el valor actual del contador. En este estado permanecerá hasta que el reloj SCLK tome el valor uno lógico, después pasará al estado S2. De esta forma, se habrá guardado en el registro el bit recibido justo antes del flanco de subida del reloj SCLK, cuando el bit era estable.

- **ESTADO S2**: Se incrementa el contador de bit para leer el próximo bit en el siguiente flanco de subida del reloj SCLK. Pasará después al estado S3.
- **ESTADO S3**: En este estado permanecerá hasta el siguiente flanco de bajada del reloj SCLK. Pasará entonces al estado S4.
- **ESTADO S4**: Si se han recibido todos los bits necesarios de la muestra de la señal de audio, se pasará al estado S5. Si todavía queda algún bit por recibir, pasará al estado S1B para iniciar una nueva recepción de un bit.
- **ESTADO S5**: Se activa la señal "dato_cad_recibido" con un valor '1' lógico para indicar al reconocedor que existe una nueva muestra de la señal de audio que se encuentra guardada en el módulo "buffer_decodifica". Desde allí será accesible para todas las unidades que realicen el procesamiento del dato en el reconocimiento vocálico. Pasará luego al estado S0 a la espera de una nueva muestra.

INTERFAZ DEL MÓDULO:

RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.
RELOJ_SCLK: reloj que establece el sincronismo de bit de los datos transmitidos. Ante una modificación en el nivel del reloj LRCLK, en cada flanco de bajada del reloj SCLK se modificará el nivel de la línea de datos según el bit de la palabra de 20 bits que corresponda, empezando por el bit más significativo. Tiene un periodo de 800 ns.
RELOJ_LRCLK: reloj que establece la frecuencia de muestreo de la señal analógica (frecuencia de 19531 Hz ó 51.2 us de periodo). Cuando inicia un nivel alto, en cada flanco de bajada del reloj de sincronismo de bit SCLK se modificará la línea de datos para que se corresponda con el bit correspondiente de la palabra de 20 bits, empezando por el más significativo. Uno tras otro se emitirán los 20 bits de la muestra digital en complemento a dos del canal izquierdo de audio. Cuando se inicia un nivel bajo, se emitirán uno tras otro los 20 bits de la muestra digital en complemento a dos del canal derecho de audio.
CARGA_BIT: Señal que al activarse provocará la carga en el registro de bit indexado por el contador de bit externo "cuenta_bits", del bit actual de la palabra muestra de 20 bits se está transmitiendo por el conversor analógico-digital. Esta señal provocará la carga sucesiva en los distintos registros de bit de la palabra muestra de 20 bits.
CUENTA_BITS: Valor del contador externo gracias al cual se recorre uno a uno todos los bits de la palabra muestra de 20 bits. El valor 0 del contador se corresponde con la posición del bit más significativo y a medida que se incremente el contador se irán recorriendo todos los bits. Según el valor del contador, el dato leído desde el conversor se guardará en el registro de bit correspondiente para una vez recorridos todos los bits, conformar la muestra recibida.
DATO_CAD_RECIBIDO: Esta señal indica a unidades externas y ajenas al control del conversor analógico-digital que acaba de recibirse una nueva muestra de la señal de audio analógica. El dato muestra está almacenado y puede ser leído por las unidades encargadas del reconocimiento vocálico.

1.1.1.3 buffer_decodifica.vhd

```
ENTITY buffer_decodifica IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    dato_bit: IN STD_LOGIC;
    num_bit: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    carga_bit: IN STD_LOGIC;
    dato_recibido: OUT STD_LOGIC_VECTOR(11 DOWNTO 0)
  );
END buffer_decodifica;
```

En esta unidad se encuentran los registros que almacenan los bits de la muestra digital de la señal analógica.

Cada muestra obtenida por el conversor analógico digital es enviada al dispositivo reconocedor para que sea procesada. La muestra está codificada en 20 bits en complemento a dos. Este dato se recibe bit a bit (serie), por lo tanto, el dato debe ser también guardado bit a bit de forma secuencial, según sea emitido por el conversor. Una vez terminada la captura, la muestra será accesible a través del vector de bit "dato_recibido".

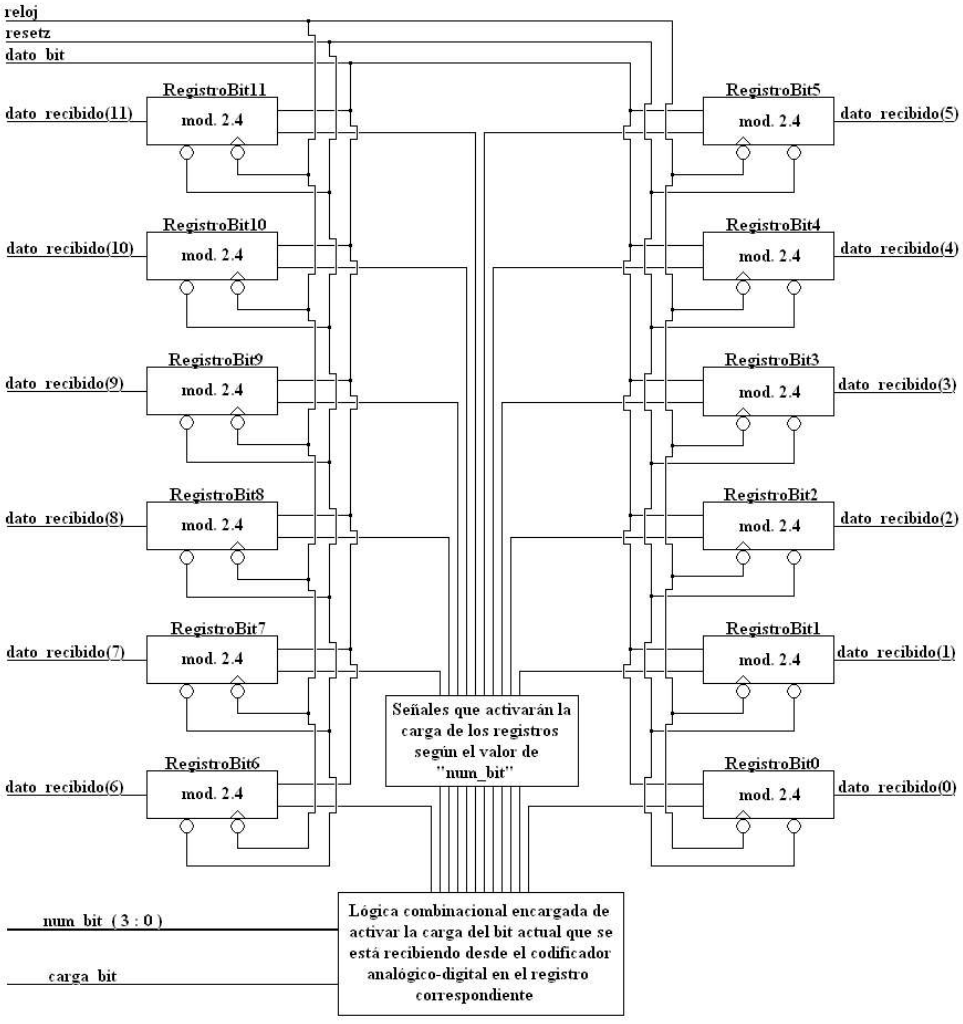
Un contador externo indicará el número del bit que se está recibiendo, correspondiéndose el bit más significativo con el cuarto valor del contador. No tomo los primeros tres bits porque estos tres bits no incrementan la información recibida, ya que la amplitud de la señal de audio no supera el umbral que haría que estos bits tuvieran información. Así y todo, el tercer bit puede tomarse como el bit más significativo sin temor a perder información, siendo éste el mismo signo que se obtendría tomándolo tres bits antes. A medida que reciban los bits, el contador se irá incrementando, hasta recorrer todos los bits del dato. Por cada uno de los bits existirá un registro para guardarlo y será una lógica combinacional la que seleccione el registro en el cual se guardará el bit actual que se está recibiendo. El bit se cargará cuando la señal "carga_bit" esté activa.

La muestra binaria se recibe a través de una palabra de 20 bits, pero sólo se utilizarán 12 bits. Los tres primeros se desechan ya que no incrementan la información recibida puesto que la amplitud de la señal de audio no supera el umbral que haría que estos bits tuvieran información. Se considera que la parte importante de información de la señal de audio se encuentra en los siguientes 12 bits, de forma que para no complicar el diseño, se tomarán sólo estos 12 bits.

INTERFAZ DEL MÓDULO:

RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.

DATO_BIT: Bit leído desde el conversor analógico-digital, correspondiente al bit actual de la muestra de 20 bits en complemento a dos que se está transmitiendo desde el conversor. Como el dato se transmite bit a bit a través de una misma línea de datos, a través de esta entrada se tendrá acceso al valor de la línea.
NUM_BIT: Valor del contador externo gracias al cual se recorre uno a uno todos los bits de la palabra muestra de 20 bits. El valor 0 del contador se corresponde con la posición del bit más significativo y a medida que se incremente el contador se irán recorriendo todos los bits. Según el valor del contador, el dato leído desde el conversor se guardará en el registro de bit correspondiente para una vez recorridos todos los bits, conformar la muestra recibida.
CARGA_BIT: Señal que al activarse provocará la carga en el registro de bit indexado por el contador de bit externo "num_bit", del bit actual de la palabra muestra de 20 bits se está transmitiendo por el conversor analógico-digital. Esta señal provocará la carga sucesiva en los distintos registros de bit de la palabra muestra de 20 bits.
DATO_RECIBIDO: Valor de la señal de audio muestreada por el conversor analógico-digital que será accesible por la unidad de reconocimiento de voz para su procesamiento y así realizar la detección de la vocal pronunciada. La información de la muestra vocálica es accesible a través de los 12 bits en complemento a dos del vector de bits "dato_recibido".



1.1.2 pasa_a_flotante.vhd

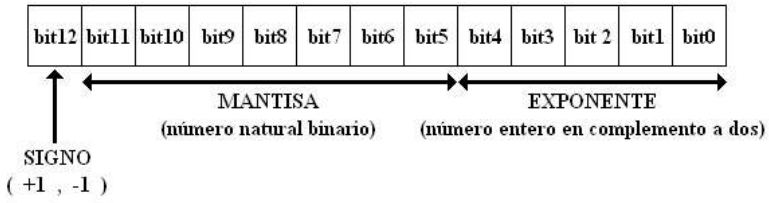
```
ENTITY pasa_a_flotante IS
  PORT(
    dato_recibido: IN STD_LOGIC_VECTOR(11 DOWNTO 0);
    dato_flotante: OUT STD_LOGIC_VECTOR(12 DOWNTO 0)
  );
END pasa_a_flotante;
```

Convierte el dato binario de entrada de 12 bits que representa a un entero en complemento a dos en un dato binario de salida de 13 bits que representa a un número flotante. El bit más significativo es el signo del número, los siguientes 7 bits representan al módulo, y los últimos 5 bits al exponente en base dos del número en complemento a dos.

En el proceso de reconocimiento, para trabajar con la máxima precisión, utilizo una unidad aritmética que trabaja con datos flotantes. De esta forma evito trabajar con largas cadenas de bits y reduzco la posibilidad de desbordamiento en los resultados de las multiplicaciones. Aumenta la complejidad de la unidad aritmética lógica, pero los módulos que la utilizan salen beneficiados por el tratamiento uniforme que se da a los datos numéricos, siempre representados por 13 bits.

El formato numérico de la representación de los datos con los que se trabajará en el sistema reconocedor, y por lo tanto, en la unidad aritmética es el siguiente:

- El bit más significativo es el signo del número.
- Los siguientes 7 bits conforman la mantisa del número, es decir, el módulo binario del número que se multiplicará por 2 elevado a la potencia indicada por el exponente.
- Los últimos 5 bits conforman el valor del exponente representando a un entero binario en complemento a dos. Es el exponente en base dos por el que habrá que multiplicar la mantisa para definir el módulo del número.



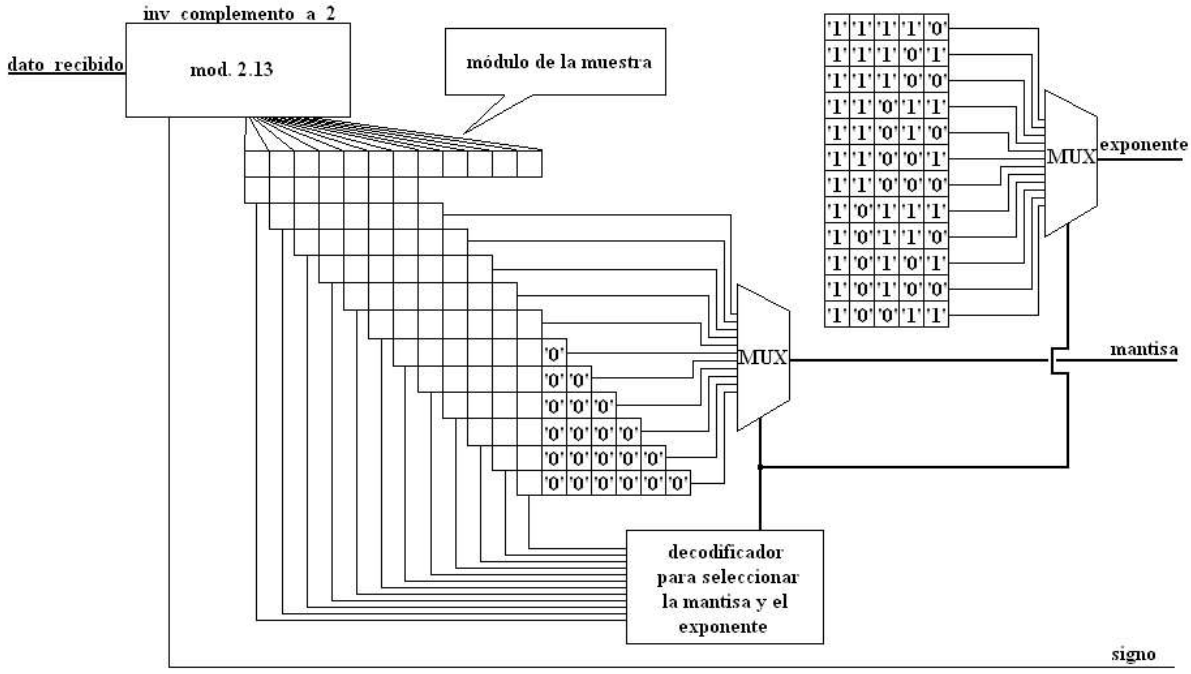
$$\text{SIGNO} \times \text{MANTISA} \times 2^{\text{EXPONENTE}}$$

Por eso es necesario transcodificar la muestra digital de entrada desde el conversor analógico-digital (entero de 12 bits en complemento a dos) al formato numérico flotante global utilizado en la unidad aritmética.

La muestra recibida desde el conversor analógico-digital representa a un entero de 12 bits en complemento a dos. Se obtiene el signo y el módulo separadamente de la muestra. Para extraer la máxima precisión del módulo, me quedo con los 7 bits que proporcionan la mayor información. Para ello divido obtengo palabras de 7 bits, cada una empezando en un bit distinto. Si hace falta completar los bits de la palabra, se añade ceros. La mantisa la conformarán los 7 bits que representen la mayor información de la muestra, esto se consigue seleccionando el bloque de 7 bits donde el bit más significativo es el primer bit distinto de cero del módulo de 12 bits de la muestra. El exponente lo forman 5 bits que representan a un número entero binario en complemento a dos. Tomará un valor según el bloque de 7 bits que se haya realizado en la mantisa, así se realiza un preescalado para aumentar la precisión en los cálculos futuros y evitar que pueda producirse un desbordamiento. Tomará el valor -1 si la mantisa la conforman los 7 bits más significativos del módulo. Tomará el valor -2 si la mantisa la conforman los 7 bits más significativos tras el primer bit más significativo del módulo. Tomará el valor -3 si la mantisa la conforman los 7 bits más significativos tras el segundo bit más significativo del módulo. Y así sucesivamente.

INTERFAZ DEL MÓDULO:

DATO_RECIBIDO: Muestra digital recibida desde el conversor analógico-digital formada por 12 bits que representan a un número entero binario en complemento a dos.
DATO_FLOTANTE: Valor de la muestra digital en el formato numérico de 13 bits utilizado por el reconocedor y su unidad aritmética. El bit más significativo es el signo del número. Los siguientes 7 bits conforman la mantisa del número, es decir, el módulo binario del número que se multiplicará por 2 elevado a la potencia indicada por el exponente. Los últimos 5 bits conforman el valor del exponente representando a un entero binario en complemento a dos. Es el exponente en base dos por el que habrá que multiplicar la mantisa para definir el módulo del número.



1.1.3	recnew.vhd

```
ENTITY recnew IS
  PORT(
    reloj: IN STD_LOGIC;
    resetz: IN STD_LOGIC;
    CE_NEG_FLASH: OUT STD_LOGIC;
    RESET_NEG_FLASH: OUT STD_LOGIC;
    OE_NEG_FLASH: OUT STD_LOGIC;
    WE_NEG_FLASH: OUT STD_LOGIC;
    DIRECCION_FLASH: OUT STD_LOGIC_VECTOR(8 DOWNTO 0);
    DATOS_FLASH: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dato_cad_recibido: IN STD_LOGIC;
    dato_flotante: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    vocal_detectada: OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
    flag_vocal_detectada: OUT STD_LOGIC;
    incrementa_contador_global: OUT STD_LOGIC
  );
END recnew;
```

Módulo que realiza el proceso de reconocimiento vocálico de una forma integral. Tras recibir 512 muestras desde el conversor analógico-digital, se iniciaría el proceso de obtención de los parámetros cepstrales. En función de esos parámetros cepstrales se decidirá si se ha pronunciado una vocal y cuál ha sido esa vocal. Una red neuronal con topología de mapas autoorganizados de 32 neuronas se encargará de reconocer la vocal pronunciada. La vocal detectada se traducirá posteriormente en un evento de movimiento que el módulo "modulo_raton" transmitirá al ordenador.

INTERFAZ DEL MÓDULO:

RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.
RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo y a todos los dispositivos que conforman el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CE_NEG_FLASH: (CHIP ENABLE NEGADO) Para activar la memoria FLASH y posibilitar la lectura de los valores almacenados en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA esta señal debe estar activa con un valor '0' lógico. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución del módulo "000" del componente "recnew" y toma el valor '1' lógico cuando el componente "recnew" realiza las operaciones de reconocimiento para permitir el control del conversor analógico-digital.
RESET_NEG_FLASH: Cuando toma el valor '1' lógico el dispositivo se encontrará en el modo estándar de operación. Cuando la señal toma el valor '0' lógico se detiene el funcionamiento normal del dispositivo y pone las salidas del dispositivo en el estado de alta impedancia. Esta señal tomará siempre el valor '1' lógico.
OE_NEG_FLASH: Los datos almacenados en la dirección de memoria indicada por el bus de direcciones son mostrados por el bus de datos cuando esta señal toma un valor '0' lógico estando habilitada la lectura de datos. Esta señal toma el valor '0' lógico cuando se están leyendo los valores almacenados en la memoria FLASH, proceso que se realiza una vez durante la ejecución

del módulo “000” del componente “recnew” y toma el valor '1' lógico cuando el componente “recnew” realiza las operaciones de reconocimiento.

WE_NEG_FLASH: Se habilita la escritura en la memoria FLASH cuando esta señal toma un valor '0' lógico. Se habilita la lectura de la memoria FLASH cuando esta señal toma el valor '1' lógico. Las señales “CE_NEG_FLASH” y “OE_NEG_FLASH” deben estar activas con un valor '0' lógico.

DIRECCION_FLASH: Dirección de la memoria FLASH desde donde se leerán los valores de los pesos sinápticos de las neuronas cuando se habilite la lectura de los datos de memoria (“WE_NEG_FLASH” con el valor '1' lógico). También es la dirección de la memoria FLASH donde se almacenarán los valores de los pesos sinápticos de las neuronas cuando se habilite la escritura de los datos en memoria (“WE_NEG_FLASH” con el valor '0' lógico). Las señales “CE_NEG_FLASH” y “OE_NEG_FLASH” deben estar activas con un valor '0' lógico. Los pesos se encuentran almacenados en el bloque de parámetros número uno de la memoria FLASH externa desde la dirección hexadecimal “04000” a la dirección hexadecimal “041FF”.

DATOS_FLASH: Bus de datos que mostrará el byte almacenado en la memoria principal cuando esté habilitada la lectura de datos (“WE_NEG_FLASH” con el valor '1' lógico). También es el bus de datos con el byte que se almacenará en la memoria principal cuando esté habilitada la escritura de datos (“WE_NEG_FLASH” con el valor '0' lógico). Las señales “CE_NEG_FLASH” y “OE_NEG_FLASH” deben estar activas con un valor '0' lógico.

DATO_CAD_RECIBIDO: Esta señal indica al dispositivo reconocedor que desde el conversor analógico-digital que acaba de recibirse una nueva muestra de la señal de audio analógica. El dato muestra está almacenado y puede ser leído por las unidades encargadas del reconocimiento vocálico.

DATO_FLOTANTE: Valor de la muestra digital recibida desde el conversor analógico-digital en el formato numérico de 13 bits utilizado por el reconocedor y su unidad aritmética. El bit más significativo es el signo del número. Los siguientes 7 bits conforman la mantisa del número, es decir, el módulo binario del número que se multiplicará por 2 elevado a la potencia indicada por el exponente. Los últimos 5 bits conforman el valor del exponente representando a un entero binario en complemento a dos. Es el exponente en base dos por el que habrá que multiplicar la mantisa para definir el módulo del número.

VOCAL_DETECTADA: Código de la vocal que se ha detectado en el proceso de reconocimiento. La vocal detectada se traducirá posteriormente en un evento de movimiento que el módulo “modulo_raton” transmitirá al ordenador. El código “000” se corresponde con la vocal 'A', el código “001” se corresponde con la vocal 'E', el código “010” se corresponde con la vocal 'I', el código “011” se corresponde con la vocal 'I' y el código “100” se corresponde con la vocal 'U'.

FLAG_VOCAL_DETECTADA: Este flag indica cuando está activo con un valor uno lógico que la energía de la señal de audio ha superado el umbral y por lo tanto, los parámetros cepstrales se traducirán en una vocal cuyo código será “vocal_detectada”. Este código se corresponde con la vocal reconocida y el flag indica que se ha pronunciado una vocal.

INCREMENTA_CONTADOR_GLOBAL: Esta señal incrementa un contador externo aproximadamente cada 100 milisegundos. Este contador externo es utilizado para poder tener constancia de los tiempos globales transcurridos. De esta forma, el dispositivo ratón sincroniza los eventos que necesitan de un tiempo para su ejecución, así puedo crear eventos en múltiplos temporales de 100 milisegundos.

BASE TEÓRICA DEL SISTEMA DE RECONOCIMIENTO:

El proceso de reconocimiento vocálico base del proyecto "Sistema Autónomo para accionamiento por voz de un ratón de ordenador" está basado en la obtención de los parámetros cepstrales de 512 muestras. Las muestras se obtienen del codificador analógico-digital y este módulo se encarga de generar los parámetros cepstrales de las 512 muestras. El proceso de obtención de los parámetros cepstrales se denomina parametrización.

Quiero realizar una parametrización de las vocales para extraer la información más importante de las señales. La parametrización tiene como objetivo fundamental la representación de la señal de habla, previamente muestreada, a través de unos parámetros que resalten las características más importantes del mensaje comprendido en la onda acústica, eliminando parte de la redundancia de la señal.

En la actualidad se prefiere el análisis frecuencial del habla al análisis temporal. Los parámetros más utilizados en el análisis temporal son la energía local, la tasa de cruces por cero y la función autocorrelación. Este tipo de análisis es muy sencillo y rápido, y tiene una interpretación física directa. Para la realización del proyecto se va a preferir un análisis frecuencial pues nos proporciona más información que la que obtendríamos si tratáramos la señal en el dominio del tiempo. No nos bastará con obtener unas características espectrales de la señal, pues de ella debemos extraer la información relevante que nos permite diferenciar las vocales. La gran mayoría de los valores espectrales no nos proporcionan información sobre las vocales, ya que sólo nos interesarán las bajas frecuencias, además mucha información es redundante o no relevante. Los parámetros que vamos a extraer son los llamados **coeficientes cepstrales**. El proceso de obtención de los parámetros cepstrales es como sigue:

- Se aplica la Transformada Discreta de Fourier y extraemos el módulo en cada uno de los puntos de la transformada, despreciando la fase debido a la poca información que proporciona en los procesos de reconocimiento del habla. Realizaremos la Transformada de Fourier mediante un algoritmo de obtención de la FFT de diezmado en el tiempo.

- Llegados a este punto se utiliza la escala Mel para dividir el espectro en un banco de filtros, mucho más estrechos y linealmente espaciados en las bajas frecuencias, y muy amplios y logarítmicamente espaciados en las altas. De este modo, se da mayor importancia a la información contenida en las bajas frecuencias en consonancia con el comportamiento del oído humano. Se calcula la energía en cada una de las bandas de frecuencias en que la escala Mel divide el espectro. Para ello se suman los módulos al cuadrado de la FFT en los puntos que se encuentren contenidos en cada una de dichas bandas. Se han utilizado 20 bandas, cada una de ellas utiliza la aproximación a la FFT de los siguientes puntos:

  Banda 1: Suma de las FFT de índice 1 al 2
  Banda 2: Suma de las FFT de índice 3 al 4
  Banda 3: Suma de las FFT de índice 5 al 6
  Banda 4: Suma de las FFT de índice 7 al 8
  Banda 5: Suma de las FFT de índice 9 al 10
  Banda 6: Suma de las FFT de índice 11 al 12

Banda 7: Suma de las FFT de índice 13 al 14
Banda 8: Suma de las FFT de índice 15 al 16
Banda 9: Suma de las FFT de índice 17 al 20
Banda 10: Suma de las FFT de índice 21 al 24
Banda 11: Suma de las FFT de índice 25 al 30
Banda 12: Suma de las FFT de índice 31 al 36
Banda 13: Suma de las FFT de índice 37 al 42
Banda 14: Suma de las FFT de índice 43 al 48
Banda 15: Suma de las FFT de índice 49 al 56
Banda 16: Suma de las FFT de índice 57 al 64
Banda 17: Suma de las FFT de índice 65 al 72
Banda 18: Suma de las FFT de índice 73 al 80
Banda 19: Suma de las FFT de índice 81 al 90
Banda 20: Suma de las FFT de índice 91 al 100
Banda 21: Suma de las FFT de índice 101 al 110
Banda 22: Suma de las FFT de índice 111 al 120
Banda 23: Suma de las FFT de índice 121 al 132
Banda 24: Suma de las FFT de índice 133 al 144
Banda 25: Suma de las FFT de índice 145 al 156
Banda 26: Suma de las FFT de índice 157 al 168
Banda 27: Suma de las FFT de índice 169 al 182
Banda 28: Suma de las FFT de índice 183 al 196
Banda 29: Suma de las FFT de índice 197 al 210
Banda 30: Suma de las FFT de índice 211 al 224
Banda 31: Suma de las FFT de índice 225 al 240
Banda 32: Suma de las FFT de índice 241 al 256

- Se calcula el logaritmo de las energías calculadas anteriormente. De esta manera obtenemos 20 coeficientes que denominaremos coeficientes Mel.

- Los coeficientes cepstrales se calculan como la transformada coseno discreta (DCT), que hace las veces de transformada inversa, de las energías logarítmicas obtenidas con anterioridad. En concreto, los coeficientes cepstrales se obtienen a partir del muestreo de dicha transformada. El cálculo de los MFCC responde a la expresión :

$$MFCC_j(i) = \sum e(j,k)\cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{L} \right]$$

$$\text{donde } i = 1 \ldots M$$

Los elementos de la ecuación representan:

k es la banda de frecuencias.
j es la trama en curso.
e(j,k) es el logaritmo de la suma de los módulos al cuadrado de la FFT en la banda k de la trama j.
L es el número de bandas o filtros.
M es el número total de coeficientes MFCC.

Al finalizar la parametrización, cada vector de valores en frecuencia se convierte en un vector compuesto por M parámetros, en los cuales está contenida la información significativa del tracto vocal. A estos parámetros se les conoce como Coeficientes Cepstrales. Una red neuronal con la topología de un mapa autoorganizado detectará la vocal pronunciada en el caso de que se hubiera pronunciado una. La red neuronal tiene un total de 32 neuronas y cada una de ellas calculará la similitud entre el vector de coeficientes cepstrales detectado y el vector de pesos sinápticos de la neurona. La ponderación de la similitud se realiza a través del cálculo de la distancia euclídea entre el vector de ocho coeficientes cepstrales y el vector de ocho pesos sinápticos de la neurona.

Los pesos sinápticos de las neuronas se calcularon previamente y de forma externa mediante un algoritmo de aprendizaje en el que cada neurona del mapa sintoniza con diferentes tomas vocálicas aleatorias presentadas en forma de vectores de coeficientes cepstrales. El proceso de aprendizaje es el siguiente. Tras la presentación y procesamiento de un vector de entradas $\mathbf{x}(t)$, la neurona vencedora modifica sus pesos de manera que se parezcan un poco más a $\mathbf{x}(t)$. De este modo, ante el mismo patrón de entrada, dicha neurona responderá en el futuro todavía con más intensidad. El proceso se repite para numerosos patrones de entrada, de forma que al final los diferentes vectores de referencia sintonizan con dominios específicos de las variables de entrada, y tienden a representar la función densidad de probabilidad $p(\mathbf{x})$ (o función de distribución) del espacio sensorial. Durante el aprendizaje se actualizan tanto los pesos de la neurona ganadora actual como los de las neuronas pertenecientes a su entorno de acuerdo con una función de vecindad. De esta manera, en el modelo de mapas autoorganizados se logra que neuronas próximas sintonicen con patrones similares, quedando de esta manera reflejada sobre el mapa una cierta imagen del orden topológico presente en el espacio de entrada (ordenación de los detectores de rasgos).

En la fase de reconocimiento, se activará la neurona que tenga la menor distancia euclídea entre el vector de ocho coeficientes cepstrales de la señal de voz captada y el vector de ocho pesos sinápticos de la neurona. En el proceso de aprendizaje se creó un cierto orden topológico según los rasgos y patrones de la vocal captada que hace que cuando se pronuncie la vocal 'A' se activen determinadas neuronas, que cuando se pronuncie la vocal 'E' sean otras las neuronas que se activen, que cuando se pronuncie la vocal 'I' sean otras neuronas diferentes las que se activen, que cuando se pronuncie la vocal 'O' sea otro grupo de neuronas las que se activen y cuando se pronuncie la vocal 'U' sean las restantes neuronas las que se activen. El orden topológico se utilizará para decidir la vocal pronunciada según la neurona activada.

La correspondencia entre la neurona ganadora y la vocal pronunciada está fijada de antemano, ya que tras la finalización de la fase de aprendizaje se decidió esta correspondencia como la que consigue el número máximo de detecciones correctas con los patrones vocálicos utilizados en la

fase de aprendizaje. Esta correspondencia se muestra en la siguiente tabla según el número de la fila y el número de la columna de la neurona ganadora.

Fila / **Columna**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
4	O	A	U	A	U	A	U	E
3	I	A	O	A	U	O	U	E
2	I	A	I	O	I	O	U	E
1	I	O	I	O	U	O	U	E

Si la señal de voz ha superado el umbral de energía, se considera que se está pronunciando una vocal. En este caso se decidirá que se ha pronunciado la vocal asignada a la neurona ganadora, cuyo vector de pesos sinápticos más se acerca al vector de coeficientes cepstrales. En un registro se almacenará el código de la vocal reconocida: “000” en el caso de la vocal 'A', “001” en el caso de la vocal 'E', “010” en el caso de la vocal 'I', “011” en el caso de la vocal 'O' y “100” en el caso de la vocal 'U'. Un flag se activará con un valor '1' lógico para indicar que se ha pronunciado una vocal cuando la señal vocálica supere el umbral de energía. Cuando este flag está activo se le indica al sistema que se ha pronunciado la vocal identificada por el código anterior. El usuario se comunica con el ordenador para transmitir eventos de movimiento equivalentes a los que produciría un ratón PS/2 a través de la pronunciación de vocales. Son las señales vocálicas anteriores las que provocarán la transmisión de los eventos de movimiento al ordenador a través del protocolo serie PS/2. Fuera del módulo, la pronunciación de una vocal se traducirá en un evento de movimiento que el módulo “modulo_raton” tendrá que transmitir al ordenador utilizando el protocolo PS/2 como un ratón PS/2 genérico.

DESCRIPCIÓN GENÉRICA DEL SISTEMA DE RECONOCIMIENTO:

Este módulo implementa el proceso de reconocimiento de una forma global, dividiéndolo en la ejecución sucesiva de una serie de módulo:

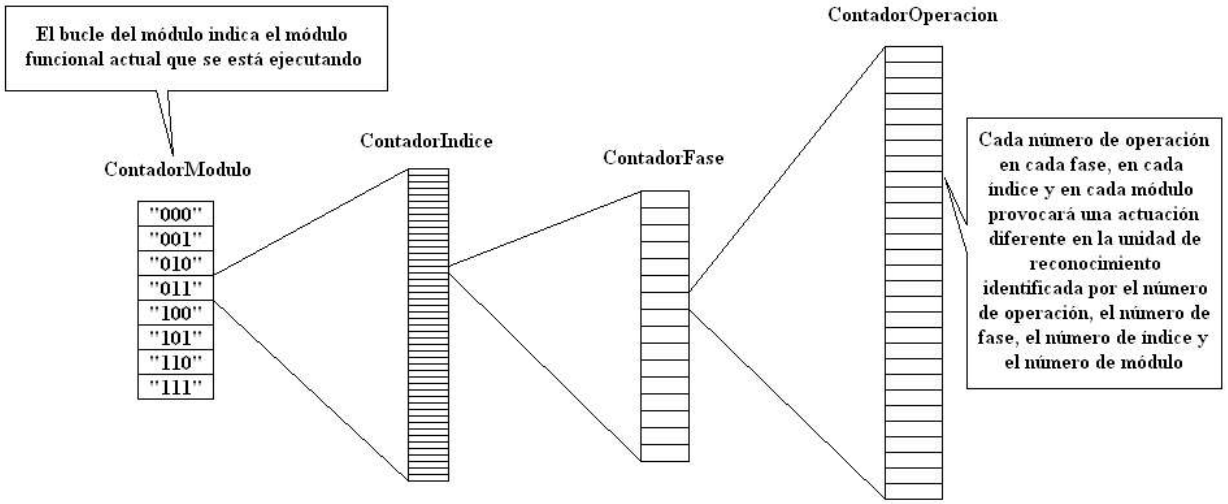
- Módulo “000”: Durante la ejecución de este módulo se guardarán en la memoria del coseno los valores del coseno que se utilizarán en la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales. También se leerán desde la memoria FLASH externa los valores de los pesos de las 32 neuronas utilizadas en el proceso de reconocimiento vocálico. Este módulo sólo se ejecutará una vez, tras el encendido del dispositivo, el resto de los módulos se ejecutarán cada vez que se reciban 512 muestras.
- Módulo “001”: En este módulo se almacenan las 512 muestras de la señal de audio en la memoria principal para su posterior procesamiento.
- Módulo “010”: Este módulo se encarga de realizar la Transformada Rápida de Fourier de las 512 muestras anteriores. Los valores de la FFT son almacenados en la memoria principal.

- Módulo "011": Realiza el módulo de la Transformada Rápida de Fourier, ya que los valores obtenidos son reales e imaginarios. Los módulos de los valores FFT son almacenados en la memoria principal.
- Módulo "100": Realiza el cálculo de los coeficientes mel de las 512 muestras, sumando los módulos de los valores FFT de acuerdo con la escala mel original del proyecto. Los coeficientes mel son almacenados en la memoria de unidad aritmética 1.
- Módulo "101": Realiza el logaritmo en base dos de los coeficientes mel de las 512 muestras. Los valores obtenidos son almacenados en la memoria de unidad aritmética 1.
- Módulo "110": Se calculan los coeficientes cepstrales de las 512 muestras de la señal de audio. Estos parámetros cepstrales son la base que se utilizará en el reconocimiento. Los coeficientes cepstrales son almacenados en la memoria de unidad aritmética 1.
- Módulo "111": En este módulo una red neuronal de mapas autoorganizados de 32 neuronas realiza el reconocimiento de la vocal pronunciada. Los parámetros de entrada de la red neuronal son los coeficientes cepstrales de las 512 muestras de la señal de audio procedentes del conversor analógico-digital. Ante los parámetros cepstrales, una de las neuronas se activará indicando la vocal pronunciada en el caso de que la señal de audio supere el umbral de energía. Tras la ejecución del módulo se iniciará de nuevo el proceso de reconocimiento de otras 512 muestras pasando de nuevo a la ejecución del módulo "001".

Un contador "ContadorMódulo" es el encargado de fijar el módulo actual que se está ejecutando, la máquina de estados que controla el reconocimiento sen encargará de incrementarlo cuando se haya terminado la ejecución del módulo y de resetearlo cuando se haya detectado la vocal pronunciada para iniciar de nuevo la adquisición de otras 512 muestras. Antes, se incrementará el contador para evitar que se ejecute el primer módulo.

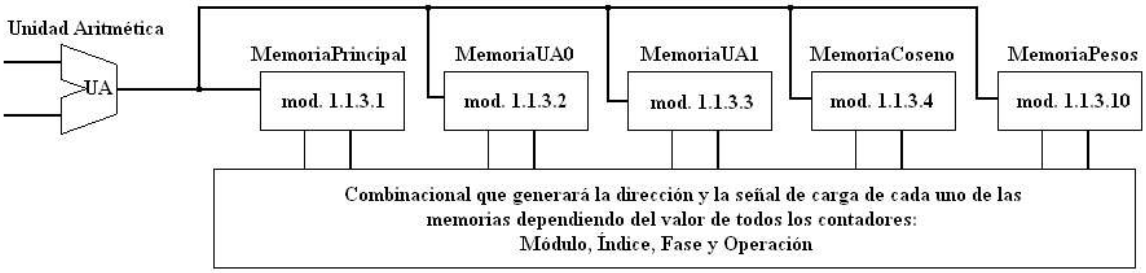
Para que todos los módulos utilicen la misma lógica y reducir así el tamaño en el diseño del dispositivo, será una única máquina de estados la que controle el funcionamiento del dispositivo, a la vez que en todos los módulos se utilizará la misma unidad aritmética, multiplexores y unidades de almacenamiento, a pesar de la diferente funcionalidad de cada módulo. Se consigue al recorrer varios bucles, uno dentro de otro. El último bucle marca la instrucción a realizar.

- El bucle externo realizado con el contador "ContadorModulo" recorre uno a uno los diversos módulos funcionales del proceso de reconocimiento.
- Dentro de cada módulo, el bucle de índice realizado con el contador "ContadorIndice" recorre uno a uno los diversos índices para los cuales debe realizarse un procesamiento individual.
- Para cada uno de los índices, el bucle de fase realizado con el contador "ContadorFase" recorre una a una las distintas fases de ejecución que para cada uno de los índices debe realizarse.
- En cada fase de cada índice de cada módulo, el bucle de operación realizado con el contador "ContadorOperación" recorre una a una las distintas operaciones que se realizarán. Dentro de este bucle se activará una única señal encargada de ejecutar todos los comandos. La funcionalidad del comando dependerá del número de operación indicado por el contador "ContadorOperacion", dependerá del número de la fase indicado por el contador "ContadorFase", dependerá del número de índice indicado por el contador "ContadorIndice" y dependerá del número del módulo indicado por el contador "ContadorModulo".

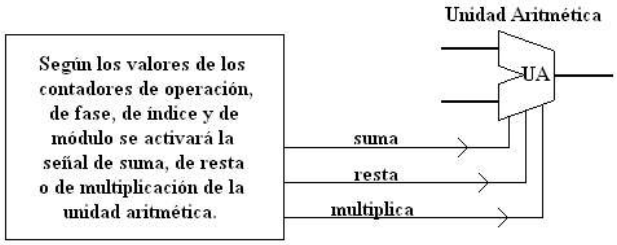


Modificando los valores finales de cada uno de los bucles y las operaciones a realizar cuando se active la señal de operación se consigue que esta entidad genérica realice las diversas funciones encomendadas en cada uno de los módulos. En cada número de operación de cada número de fase de cada número de índice de cada número de operación se realizará una operación en la unidad aritmética. Esta operación puede ser una suma, una resta o una multiplicación. Según el valor de los contadores, el resultado se almacenará en una de las memorias de datos disponibles:

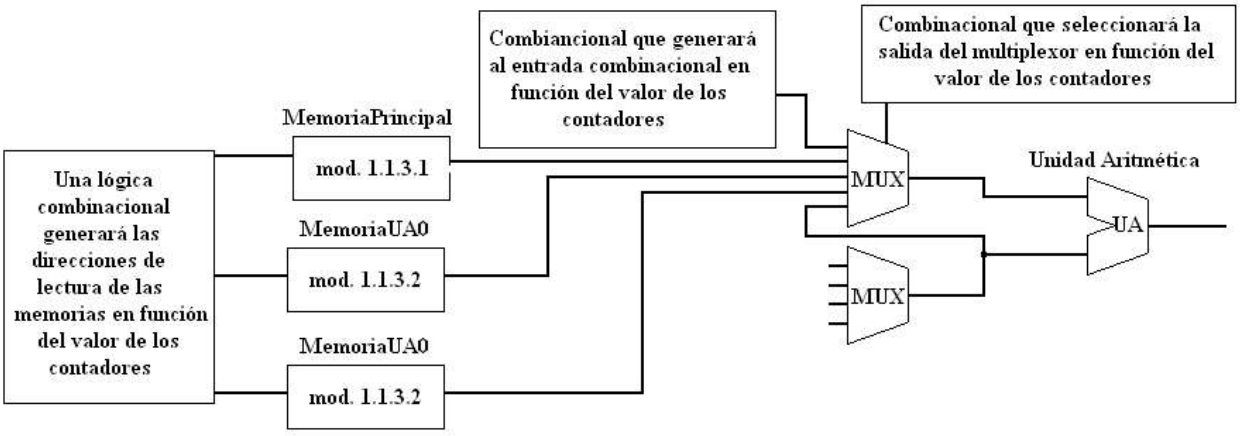
- Memoria Principal: Memoria destinada al almacenamiento de las 512 muestras recibidas desde el ordenador, así como del módulo de estas 512 muestras y otros muchos resultados intermedios utilizados en el cálculo de los coeficientes cepstrales.
- Memoria UA 0: Memoria de variables intermedias que se utilizarán para almacenar datos que entrarán posteriormente por la primera entrada de la unidad aritmética.
- Memoria UA 1: Memoria de variables intermedias que se utilizarán para almacenar datos que entrarán posteriormente por la segunda entrada de la unidad aritmética.
- Memoria coseno: Memoria donde se almacenarán los valores del coseno que se utilizarán en la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales.
- Memoria pesos: Memoria donde se almacenarán los pesos sinápticos de las 32 neuronas del mapa autoorganizado utilizado en el proceso de reconocimiento.



Las señales de carga y las direcciones de la memoria dependerán del valor actual de todos los contadores. La operación que realizará la unidad aritmética puede ser una suma, una resta o una multiplicación. Según el valor de los contadores de operación, de fase, de índice y de módulo, se activará la señal correspondiente para que la unidad aritmética realice la operación adecuada en cada momento.

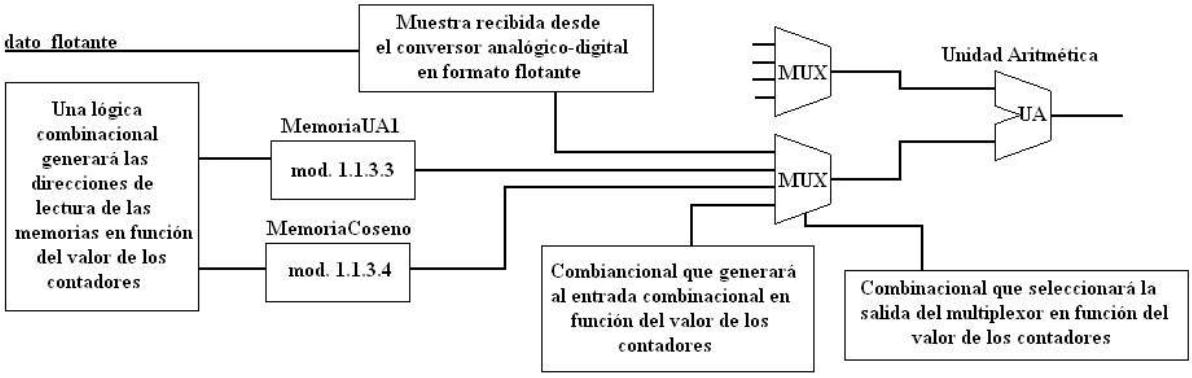


Un multiplexor controlará la primera de las entradas de la unidad aritmética. El primer dato sobre el que se operará en la unidad aritmética puede ser un valor generado por una lógica combinacional cuyo valor depende de los contadores de operación, de fase, de índice y de módulo. También puede ser el valor de lectura de la memoria principal, el valor de lectura de la memoria_UA_0 o el peso sináptico leído desde la memoria "memoria_pesos". Las direcciones de lectura de las memorias son generadas por una lógica combinacional según los valores de los contadores. Si se activa la señal "cuadrado" el valor que saldrá del multiplexor será el valor de salida del multiplexor dedicado a obtener el segundo dato de entrada de la unidad aritmética. La señal de salida del multiplexor dependerá del valor de los contadores de operación, de fase, de índice y de módulo.



Un multiplexor controlará la segunda de las entradas de la unidad aritmética. El segundo dato sobre el que se operará en la unidad aritmética puede ser un valor generado por una lógica combinacional cuyo valor depende de los contadores de operación, de fase, de índice y de módulo. También puede ser el valor de lectura de la memoria coseno o el valor de lectura de la memoria UA 0. Las direcciones de lectura de las memorias son generadas por una lógica combinacional según los valores de los contadores. Otra entrada del multiplexor es la vector de

bits de entrada "dato_flotante", que es el valor de la muestra que acaba de recibirse en formato flotante. La señal de salida del multiplexor dependerá del valor de los contadores de operación, de fase, de índice y de módulo.



Una máquina de estados controla la activación de las señales que incrementan y resetean los contadores de operación, fase, índice y módulo. También activa la señal que guarda el resultado de la operación solicitada por el valor actual de los contadores. El proceso puede encontrarse en uno de los siguientes estados:

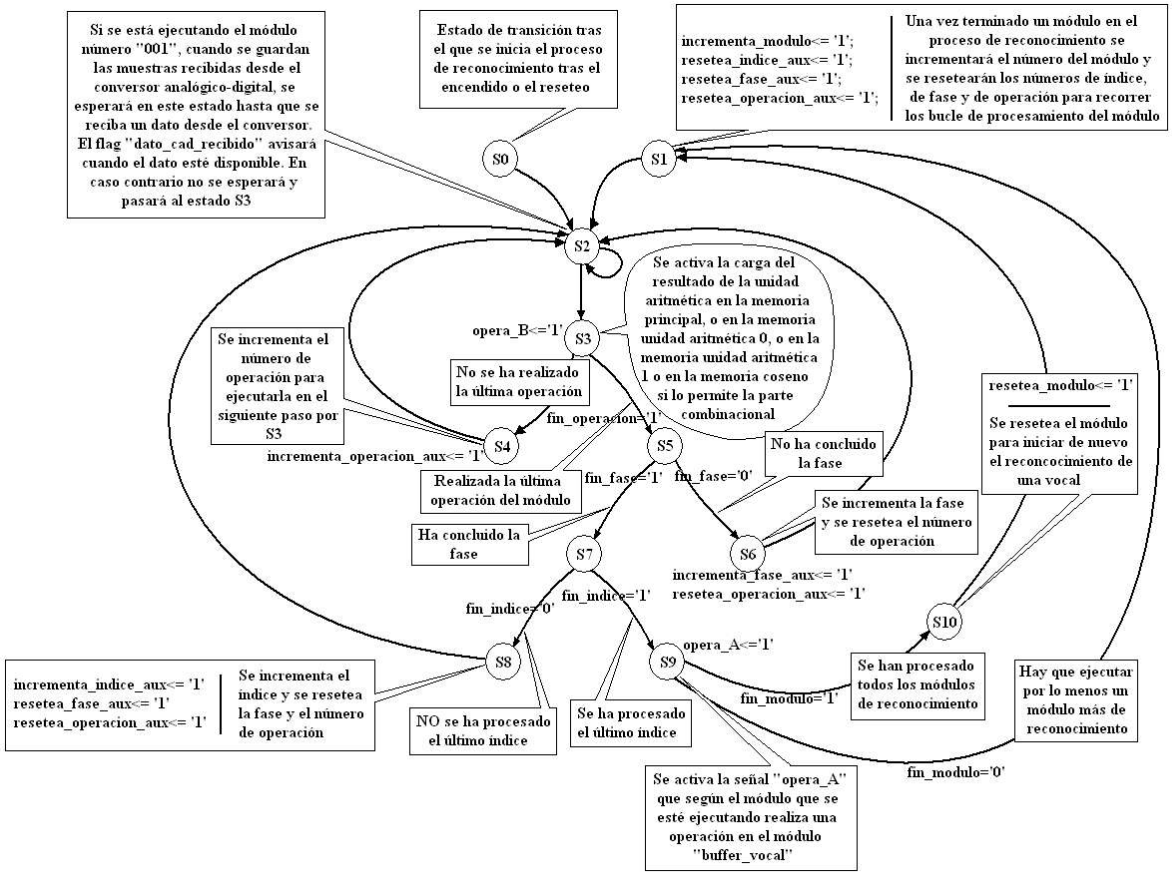
- **S0**: Estado de transición tras el que se inicia el proceso de reconocimiento después del encendido o el reseteo. Pasará después al estado S2.

- **S1**: Después de la ejecución de un módulo, la máquina de estados pasará al estado S1. Si se terminó la ejecución de todos los módulos, se reseteó el contador del número de módulo y para evitar que se ejecute de nuevo el módulo "000" (donde se almacenan los valores del coseno que se utilizarán en la Transformada Rápida de Fourier), se incrementa el contador de número de módulo y así, tras la ejecución de todos los módulos, el nuevo módulo que se ejecutará será el módulo "001". Si todavía quedaba al menos un módulo por ejecutar, se incrementará el contador de número de módulo para iniciar su ejecución. Se resetean también los contadores de índice, de fase y de número de operación para volver a recorrerlos. Después de incrementar el contador de número de módulo se pasará al estado S2.

- **S2**: Este estado sólo tiene utilidad cuando se está ejecutando el módulo "001" para guardar las 512 muestras digitales provenientes desde el conversor analógico-digital. En esta situación se esperará en este estado hasta que se reciba la señal "dato_cad_recibido" que indica que hay una nueva muestra en formato flotante guardada y lista para ser procesada. Se esperará en el estado S2 hasta que se reciba la muestra, entonces pasará al estado S3. Cuando el módulo que se está ejecutando no es el módulo "001", pasará directamente al estado S3, pues no es necesario realizar ninguna espera.

- **S3**: Se guarda el resultado de la unidad aritmética en la memoria elegida según el contador de módulo, de índice, de fase y de operación. La operación realizada (suma, resta o multiplicación) depende del valor de los contadores. También depende del valor de los

contadores el origen de datos de los operandos de la unidad aritmética. Así, en este estado puede decirse que se ejecuta la operación dependiendo del valor actual de los contadores. La lógica combinacional es la que decide qué operación realizar dependiendo del valor de los contadores. Para cada módulo se recorre un número de índices que depende del módulo, por cada índice se recorre un número de fases que depende también del módulo y por cada fase se recorre un número de operaciones que depende del módulo. Tras guardar el resultado de la unidad aritmética en la memoria principal, o en la memoria UA 0, o en la memoria UA 1, o en la memoria coseno, se pasará al estado S4 si el número de operación no era el máximo a ejecutar en este módulo y se pasará al estado S5 si el número de operación era el máximo a ejecutar en este módulo.



- **S4**: En este estado se incrementa el número de operación para recorrer el bucle de operación. Luego pasará al estado S2 para realizar la siguiente operación del bucle.

- **S5**: Si la fase actual era la última fase a recorrer en este módulo, pasará al estado S7. Si la fase actual no era la última fase por recorrer, pasará al estado S6 para incrementar la fase y recorrer de nuevo el bucle de operación de la nueva fase.

- **S6**: En este estado se incrementa el número de fase y se resetea el contador de operación para recorrer de nuevo el bucle de operación del módulo actual. Pasará al estado S2 para recorrer el bucle de operación.

- **S7**: Si el índice actual es el último índice a recorrer en este módulo, pasará al estado S9. Si el índice actual no es el último índice por recorrer en el bucle de índice de este módulo, pasará al estado S8 para recorrer el bucle de índice.

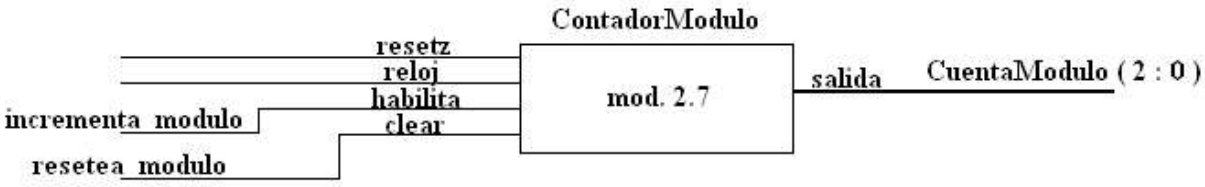
- **S8**: En este estado se incrementa el número de índice para recorrer de nuevo las fases del nuevo índice. En cada fase se recorrerán el número de operaciones estipulado en el módulo. Se resetea el contador de fase y el contador de operación para iniciar el bucle de fase y dentro de cada una de las fases recorrer el bucle de operación. Después pasa al estado S2.

- **S9**: En este momento se ha terminado la ejecución del módulo actual. Hay que proceder a la ejecución del siguiente módulo, pero el procedimiento es distinto dependiendo de si era el último módulo a ejecutar o no. Si era el último módulo, pasará al estado S10 y en caso contrario pasará al estado S1, donde se incrementará el contador de módulo y se resetearán el resto de contadores para iniciar la ejecución de todas las operaciones del nuevo módulo.

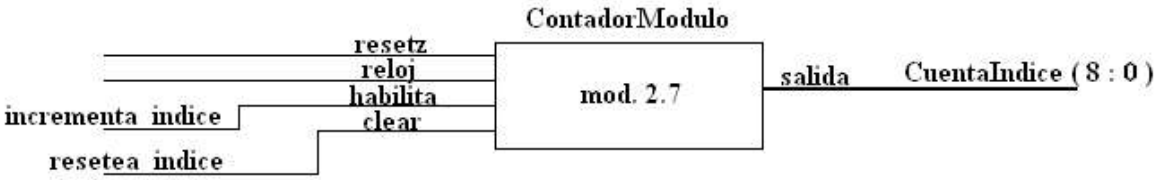
- **S10**: Como ya se han ejecutado todos los módulos se reseteará el contador de módulo. Luego pasará al estado S1 para incrementar el contador de módulo y evitar que se ejecute de nuevo el módulo "000", pues los valores del coseno ya están guardados en memoria y no es necesario volver a almacenarlos.

El funcionamiento del reconocedor se consigue mediante la interconexión de la máquina de estados con los contadores de módulo, índice, fase y operación, una serie de módulos que implementan lógica combinacional, dos multiplexores y la unidad aritmética. Cada una de estas unidades tiene una funcionalidad.

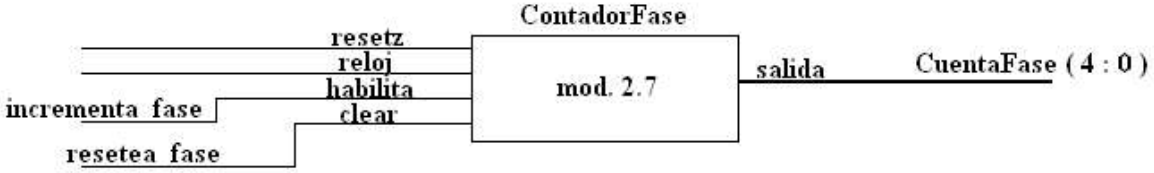
- Componente "ContadorModulo". Este contador lleva la cuenta del módulo actual que se está ejecutando en el proceso de reconocimiento. En el módulo "000" se guardan los valores del coseno y los pesos sinápticos. En el módulo "001" se guardan 512 muestras. En el módulo "010" se calcula la FFT de la muestra. En el módulo "011" se calcula el módulo de los coeficientes de Fourier. En el módulo "100" se calculan los coeficientes mel. En el módulo "101" se realiza el logaritmo en base dos de los coeficientes mel. En el módulo "110" se calculan los coeficientes cepstrales. En el módulo "111" se reconoce la vocal pronunciada utilizando la red neuronal. La máquina de estados se encarga de resetear o incrementar su valor.



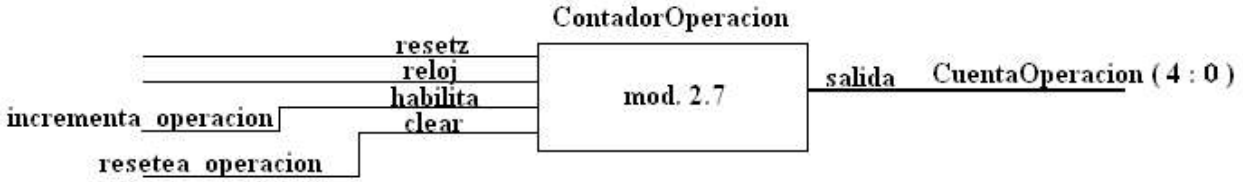
- Componente "ContadorIndice". Este módulo lleva la cuenta del número del índice actual dentro del bucle de índice cuyo valor máximo queda establecido por el número del módulo actual. Es el bucle más externo que se recorre en cada módulo. La máquina de estados se encarga de resetear o incrementar su valor.



- Componente "ContadorFase". Este módulo lleva la cuenta del número de fase actual dentro del bucle de fase cuyo valor máximo queda establecido por el número del módulo actual. Por cada uno de los índices del módulo se recorrerán todos los valores de fase. Es el bucle intermedio que se recorre en cada módulo para cada uno de los índices. La máquina de estados se encarga de resetear o incrementar su valor.

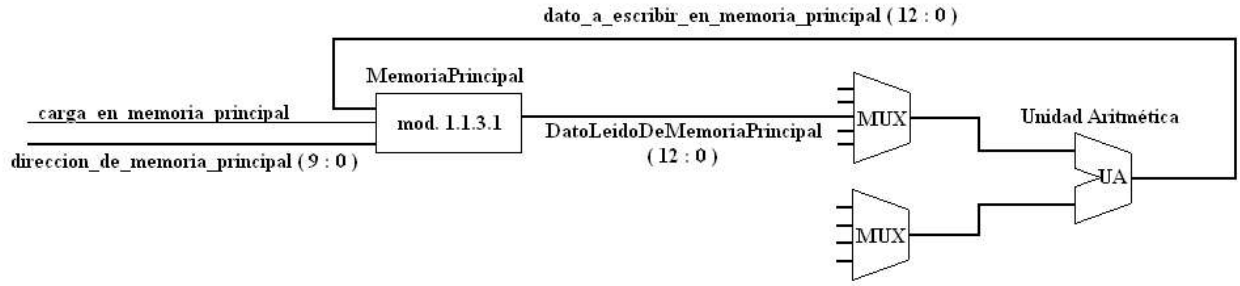


- Componente "ContadorOperacion". Este módulo lleva la cuenta del número de operación actual que se está ejecutando dentro del bucle de operación cuyo valor máximo queda establecido por el número del módulo actual. En cada cuenta del bucle de operación se realiza una acción que sólo dependerá del valor del contador de módulo, del contador de índice, del contador de fase y del contador de operación. Por cada una de las fases que se recorrerán por cada uno de los índices del módulo, se recorrerán todos los valores del contador de operación. Es el bucle final que se recorre para cada una de las fases de cada uno de los índices. La máquina de estados se encarga de resetear o incrementar su valor.

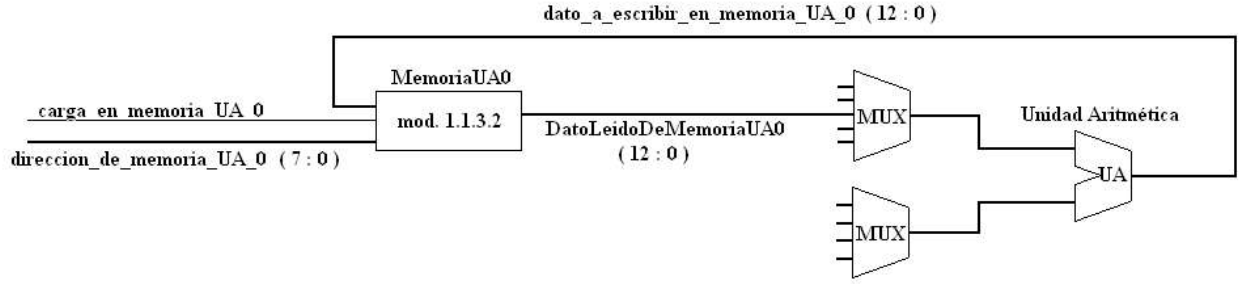


- Componente "MemoriaPrincipal": Módulo de memoria RAM de 2 Kbytes de capacidad, donde se almacenarán las 512 muestras digitales de la señal de audio. En esta memoria también se almacenarán los cálculos intermedios así como los resultados finales de la Transformada Rápida de Fourier. Se guardará el módulo de los valores reales e imaginarios de la FFT y otros datos auxiliares en los cálculos. La dirección de acceso a la memoria para

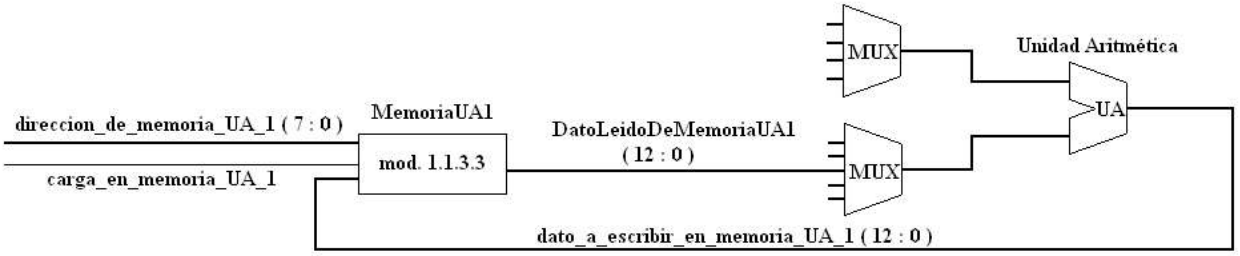
lectura o escritura sólo depende del valor actual de los contadores de módulo, índice, fase y operación. El resultado de la unidad aritmética se guardará en la dirección de escritura si al ejecutarse una operación el valor de los contadores de módulo, índice, fase y operación seleccionan a esta memoria.



- Componente "MemoriaUA0": Módulo de memoria RAM de 512 Bytes de capacidad. Esta memoria se utiliza para almacenar algunas variables usadas en las operaciones de reconocimiento. A través de un multiplexor, estos datos pueden ser el primero de los operandos de una nueva operación de suma, resta o multiplicación. La dirección de acceso a la memoria para lectura o escritura sólo depende del valor actual de los contadores de módulo, índice, fase y operación. El resultado de la unidad aritmética se guardará en la dirección de escritura si al ejecutarse una operación el valor de los contadores de módulo, índice, fase y operación seleccionan a esta memoria.



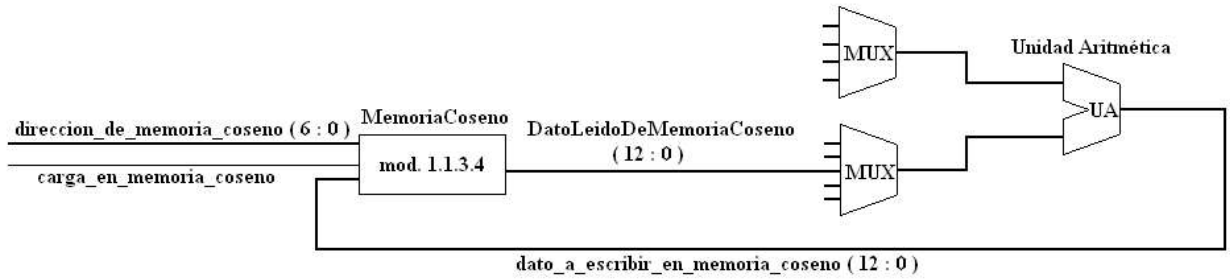
- Componente "MemoriaUA1": Módulo de memoria RAM de 512 Bytes de capacidad. Esta memoria se utiliza para almacenar algunas variables usadas en las operaciones de reconocimiento.



A través de un multiplexor, estos datos pueden ser el segundo de los operandos de una nueva operación de suma, resta o multiplicación. La dirección de acceso a la memoria para lectura o

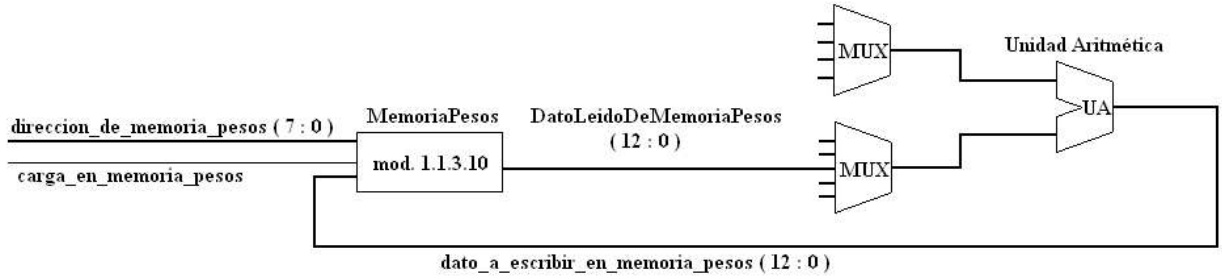
escritura sólo depende del valor actual de los contadores de módulo, índice, fase y operación. El resultado de la unidad aritmética se guardará en la dirección de escritura si al ejecutarse una operación el valor de los contadores de módulo, índice, fase y operación seleccionan a esta memoria.

- Componente "MemoriaCoseno": Módulo de memoria RAM de 256 Bytes de capacidad. Esta memoria se utiliza para almacenar los valores del coseno que se utilizarán para realizar la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales. Cuando se esté ejecutando el módulo "000" se crearán y guardarán los valores del coseno en esta memoria para posteriormente ser utilizados. Este módulo sólo se ejecutará una vez tras el encendido del dispositivo. La dirección de acceso a la memoria para lectura o escritura sólo depende del valor actual de los contadores de módulo, índice, fase y operación. El resultado de la unidad aritmética se guardará en la dirección de escritura si al ejecutarse una operación el valor de los contadores de módulo, índice, fase y operación seleccionan a esta memoria.



- Componente "MemoriaPesos": Módulo de memoria RAM de 512 Kbytes de capacidad donde se almacenarán los pesos sinápticos de las neuronas del mapa autoorganizado utilizado en el proceso de reconocimiento. Este módulo de memoria tiene posibilidad de almacenar 256 palabras de 15 bits cada una. La red neuronal tiene un total de 256 pesos (8 pesos por cada una de las neuronas, existiendo un total de 32 neuronas). En cada dirección de la memoria pesos se almacena el peso cuyo índice se corresponde con los 3 bits menos significativos del bus de direcciones de la neurona cuyo número se corresponde con los 5 bits más significativos del bus de direcciones. Cada peso sináptico de las neuronas se almacena en los 13 bits menos significativos de cada una de las palabras de memoria. El peso está en el formato binario flotante de 13 bits utilizado por la unidad aritmética en el proceso de reconocimiento. En los 3 bits más significativos de cada una de las palabras se almacenará el código de la vocal asignado a la neurona ("000" para la vocal 'A', "001" para la vocal 'E', "010" para la vocal 'I', "011" para la vocal 'O' y "100" para la vocal 'U'). En el proceso de reconocimiento se activa la neurona cuyos pesos más se acerquen a los coeficientes cepstrales detectados. Cada neurona tiene asignada una vocal que será la vocal que se reconozca en el caso de que se active la neurona. El reconocedor tiene acceso al código de la vocal asignado a cada una de las neuronas a través de la lectura de los 3 bits más significativos de las palabras almacenadas. Cuando se esté ejecutando el módulo "000" se leerán los pesos sinápticos desde la memoria FLASH externa y se almacenarán en la memoria "memoria_pesos" para posteriormente ser utilizados. Este módulo sólo se ejecutará una vez tras el encendido del dispositivo. La dirección de acceso a la memoria para lectura o escritura sólo depende del valor actual de los contadores de módulo, índice, fase y operación. El resultado de la unidad aritmética se

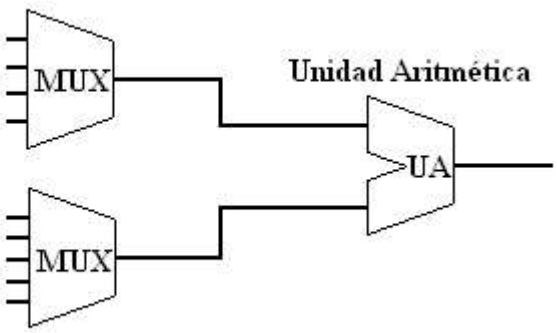
guardará en la dirección de escritura si al ejecutarse una operación el valor de los contadores de módulo, índice, fase y operación seleccionan a esta memoria.



- Multiplexor de la primera entrada a la unidad aritmética. Según el valor de los contadores de módulo, índice, fase y operación, se selecciona el origen del primer operando de la unidad aritmética, sobre el que se realizará la operación de suma, resta o multiplicación. El valor del dato puede ser el segundo operando de la unidad aritmética si quiere realizarse el cuadrado del operando. Puede ser un valor generado por una lógica combinacional cuyo valor sólo depende de los valores de los contadores de módulo, índice, fase y operación. La salida del multiplexor también puede ser el dato leído desde la memora principal o el dato leído desde la memoria de variables de la primera entrada de la unidad aritmética. Las direcciones de lectura de estas memorias sólo dependen de los contadores de módulo, índice, fase y operación.

- Multiplexor de la segunda entrada a la unidad aritmética. Según el valor de los contadores de módulo, índice, fase y operación, se selecciona el origen del segundo operando de la unidad aritmética, sobre el que se realizará la operación de suma, resta o multiplicación. El valor del dato puede ser un valor generado por una lógica combinacional cuyo valor sólo depende de los valores de los contadores de módulo, índice, fase y operación. La salida del multiplexor también puede ser el dato leído desde la memoria de variables de la segunda entrada de la unidad aritmética o el dato leído desde la memoria de valores del coseno. Las direcciones de lectura de estas memorias sólo dependen de los contadores de módulo, índice, fase y operación. También puede ser el vector de bit de entrada a través del cual se accede a la muestra digital de la señal de audio en formato flotante.

Multiplexor de la primera entrada



Multiplexor de la segunda entrada

- Componente "UnidadAritmetica". Unidad aritmética utilizada para realizar las operaciones de suma, resta o multiplicación. Tanto los operandos como el resultado están en el formato flotante utilizado en los cálculos de reconocimiento: el bit más significativo es el signo ('1' si es negativo), los siguientes 7 bits codifican el módulo de la mantisa y los 5 bits menos significativos codifican el exponente en base dos como un número entero binario en complemento a dos. La operación que realizará la unidad aritmética con los operandos (suma, resta o multiplicación) sólo dependerá del valor actual de los contadores de módulo, índice, fase y operación.

- Lógica combinacional encargada de resetear e incrementar los contadores de módulo, índice, fase y operación. Algunas veces el contador de índice hace las veces del contador de fase y viceversa. Con esta parte combinacional tengo el control sobre los incrementos y decrementos de los contadores de índice, fase y operación dependiendo del módulo que se esté ejecutando.

- Lógica combinacional dedicada a reconocer el número de módulo, índice, fase y operación actual en el que se encuentra el proceso de reconocimiento vocálico.

- Lógica combinacional que activará el flag que indica que se han recorrido todos los módulos. También activarán los flags que indican el fin del bucle de índice, de fase y de operación. El fin de estos bucles sólo depende del módulo que se esté ejecutando en cada momento.

- Lógica combinacional dedicada al control de la memoria principal. Aquí se activa la señal que seleccionará el dato leído desde la memoria principal como primer operando de la unidad aritmética en el multiplexor de entrada del primer operando de la unidad aritmética. Aquí también se generará la señal de escritura y la dirección para guardar el resultado de la unidad aritmética en la memoria.

- Lógica combinacional dedicada al control de la memoria de variables de la primera entrada de operandos de la unidad aritmética. Aquí se activa la señal que seleccionará el dato leído desde la memoria de variables de la primera entrada de operandos de la unidad aritmética como primer operando de la unidad aritmética. Aquí también se generará la señal de escritura y la dirección para guardar el resultado de la unidad aritmética en la memoria.

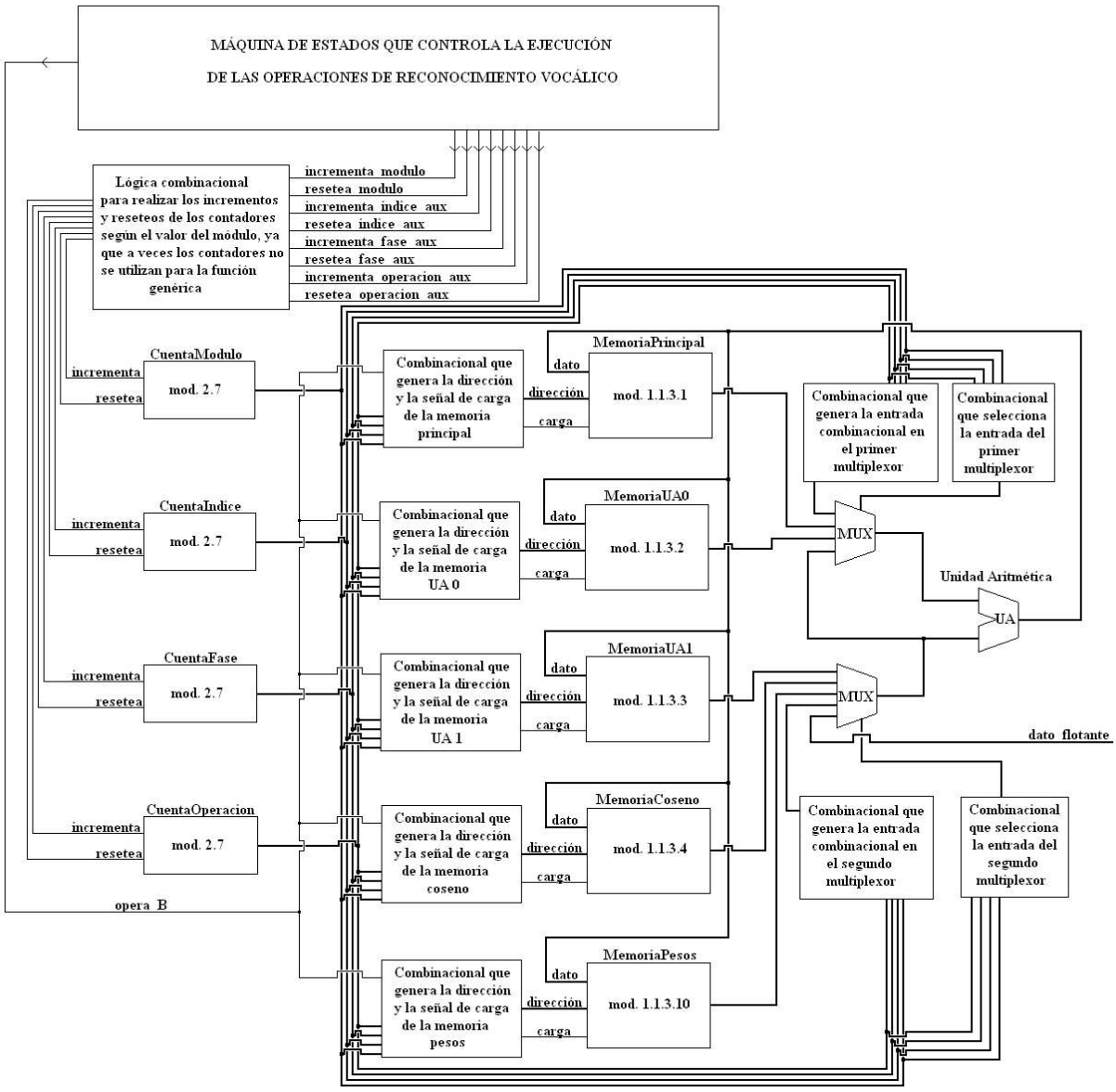
- Lógica combinacional dedicada al control de la memoria de variables de la segunda entrada de operandos de la unidad aritmética. Aquí se activa la señal que seleccionará el dato leído desde la memoria de variables de la segunda entrada de operandos de la unidad aritmética como segundo operando de la unidad aritmética. Aquí también se generará la señal de escritura y la dirección para guardar el resultado de la unidad aritmética en la memoria.

- Lógica combinacional dedicada al control de la memoria de valores del coseno. Aquí se activa la señal que seleccionará el dato leído desde la memoria de valores del coseno como segundo operando de la unidad aritmética en el multiplexor de entrada del segundo operando de la unidad aritmética. Aquí también se generará la señal de escritura y la dirección para guardar el resultado de la unidad aritmética en la memoria.

- Lógica encargada de generar la entrada combinacional que al ser seleccionada en el multiplexor entrará en la unidad aritmética como el primer operando. Representa a un dato cuyo valor sólo depende de los contadores de módulo, índice, fase y operación.

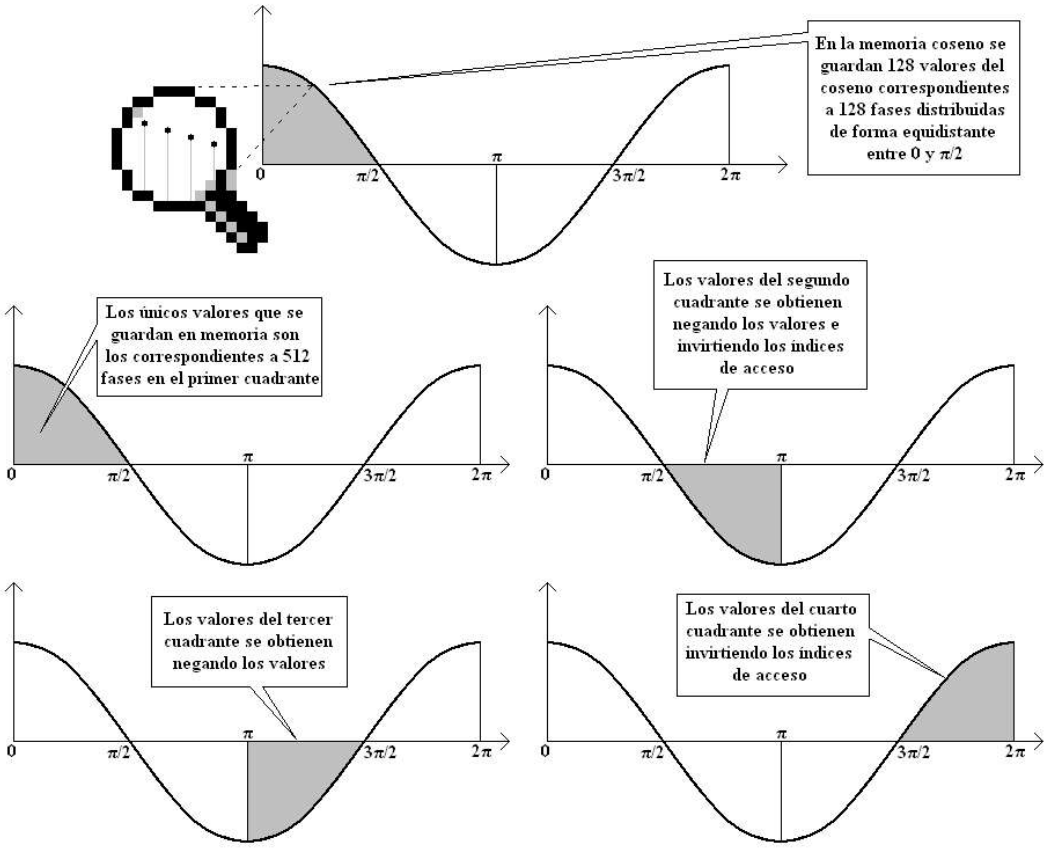
- Lógica encargada de generar la entrada combinacional que al ser seleccionada en el multiplexor entrará en la unidad aritmética como el segundo operando. Representa a un dato cuyo valor sólo depende de los contadores de módulo, índice, fase y operación.

- Lógica que genera las señales que activarán el tipo de operación que realizará la unidad aritmética: suma, resta o multiplicación. Sólo depende del valor de los contadores de módulo, índice, fase y operación.



1.1.3.0.1	recnew.vhd Módulo=”000”
Durante la ejecución de este módulo se guardarán en la memoria del coseno los valores del coseno que se utilizarán en la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales. También se leerán desde la memoria FLASH externa los valores de los pesos de las 32 neuronas utilizadas en el proceso de reconocimiento vocálico.	

Se calculan 128 valores del coseno que se utilizarán posteriormente como aproximaciones del coseno. Estos 128 valores se corresponden con el coseno de 128 fases distribuidas de forma equidistante entre 0 y π/2. El resto de valores del coseno entre π/2 y 2π no es necesario guardarlos , ya que se obtienen a partir de los anteriores. Estos valores se utilizarán para aproximar la operación coseno.



Para realizar esta operación, lo único que hay que hacer es averiguar cuál es la dirección de memoria donde está el valor correspondiente. Una lógica especializada será la encargada de

proporcionar la dirección de lectura de la memoria donde leer el valor del coseno de la fase oportuna.

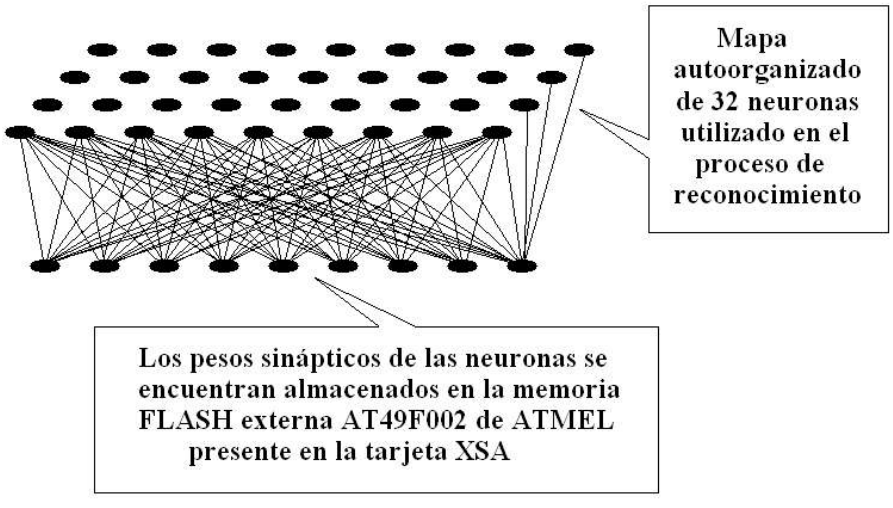
Para hallar los valores del coseno realizo una aproximación basada en series de Taylor. En la ejecución de este módulo se procederá a realizar la aproximación en series de Taylor de 128 fases separadas entre sí $\pi/256$ radianes desde 0 radianes a π radianes. En cada cálculo del coseno se realizan las siguientes operaciones:

$$\text{valor\_coseno(i)} = \frac{1}{0!}\times\left(\frac{2\times\pi\times i}{512}\right)^{0} - \frac{1}{2!}\times\left(\frac{2\times\pi\times i}{512}\right)^{2} + \frac{1}{4!}\times\left(\frac{2\times\pi\times i}{512}\right)^{4} - \frac{1}{6!}\times\left(\frac{2\times\pi\times i}{512}\right)^{6}$$

Estos cálculos realizan a través de la ejecución del siguiente algoritmo (se utiliza el lenguaje Matlab® para describir el algoritmo), que realiza la función anterior:

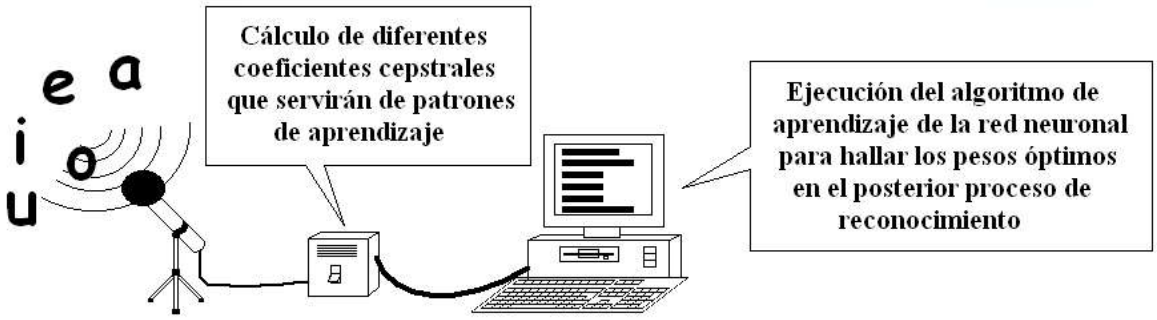
```
for i=0:1:127
  suma=0;
  for j=0:1:3
    fase=2*pi*i/512;
    auxiliar=fase^j;
    auxiliar=auxiliar^2;
    auxiliar=auxiliar/(2*j);
    signo=(-1)^j;
    auxiliar=signo*auxiliar;
    suma=suma+auxiliar;
  end
  memoria_coseno(i)=suma;
end
```

Además del cálculo de los valores del coseno, en este módulo también se leen desde la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA los parámetros utilizados en el proceso de reconocimiento. De la señal de voz pronunciada se extraen los coeficientes cepstrales, que caracterizan de forma adecuada a la señal vocálica.



Una red neuronal con la topología de un mapa autoorganizado detectará la vocal pronunciada en el caso de que se hubiera pronunciado una. La red neuronal tiene un total de 32 neuronas y cada una de ellas calculará la similitud entre el vector de coeficientes cepstrales detectado y el vector de pesos sinápticos de la neurona. La ponderación de la similitud se realiza a través del cálculo de la distancia euclídea entre el vector de ocho coeficientes cepstrales y el vector de ocho pesos sinápticos de la neurona.

Los pesos sinápticos de las neuronas se calcularon previamente y de forma externa mediante un algoritmo de aprendizaje en el que cada neurona del mapa sintoniza con diferentes tomas vocálicas aleatorias presentadas en forma de vectores de coeficientes cepstrales. El proceso de aprendizaje es el siguiente. Tras la presentación y procesamiento de un vector de entradas **x**(t), la neurona vencedora modifica sus pesos de manera que se parezcan un poco más a **x**(t). De este modo, ante el mismo patrón de entrada, dicha neurona responderá en el futuro todavía con más intensidad. El proceso se repite para numerosos patrones de entrada, de forma que al final los diferentes vectores de referencia sintonizan con dominios específicos de las variables de entrada, y tienden a representar la función densidad de probabilidad $p(\mathbf{x})$ (o función de distribución) del espacio sensorial. Durante el aprendizaje se actualizan tanto los pesos de la neurona ganadora actual como los de las neuronas pertenecientes a su entorno de acuerdo con una función de vecindad. De esta manera, en el modelo de mapas autoorganizados se logra que neuronas próximas sintonicen con patrones similares, quedando de esta manera reflejada sobre el mapa una cierta imagen del orden topológico presente en el espacio de entrada (ordenación de los detectores de rasgos).

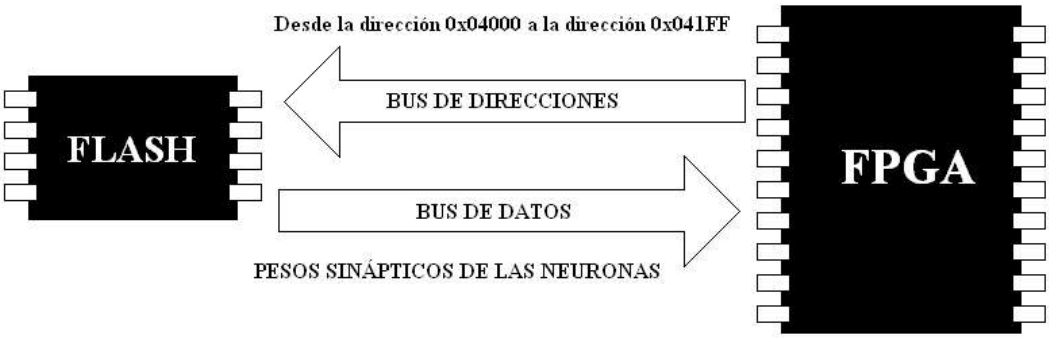


En la fase de reconocimiento, se activará la neurona que tenga la menor distancia euclídea entre el vector de ocho coeficientes cepstrales de la señal de voz captada y el vector de ocho pesos sinápticos de la neurona. En el proceso de aprendizaje se creó un cierto orden topológico según los rasgos y patrones de la vocal captada que hace que cuando se pronuncie la vocal 'A' se activen determinadas neuronas, que cuando se pronuncie la vocal 'E' sean otras las neuronas que se activen, que cuando se pronuncie la vocal 'I' sean otras neuronas diferentes las que se activen, que cuando se pronuncie la vocal 'O' sea otro grupo de neuronas las que se activen y cuando se pronuncie la vocal 'U' sean las restantes neuronas las que se activen. El orden topológico se utilizará para decidir la vocal pronunciada según la neurona activada.

Durante el reconocimiento se hace necesario disponer de los pesos de las neuronas generados en el proceso de aprendizaje, por eso, se almacenan en la memoria FLASH externa AT49F002 de

ATMEL presente en la tarjeta XSA. De esta forma, durante la inicialización del dispositivo implementado en la FPGA se puede realizar una lectura de los valores de los pesos sinápticos de las neuronas para almacenarlos en una memoria RAM interna en la FPGA "memoria_pesos". En esta memoria RAM interna se encontrarán almacenados para un acceso más rápido a los mismos, los pesos sinápticos de las neuronas en el formato binario flotante de 13 bits utilizado por la unidad aritmética (el bit más significativo es el signo del número, los siguientes 7 bits representan al módulo, y los últimos 5 bits al exponente en base dos del número en complemento a dos).

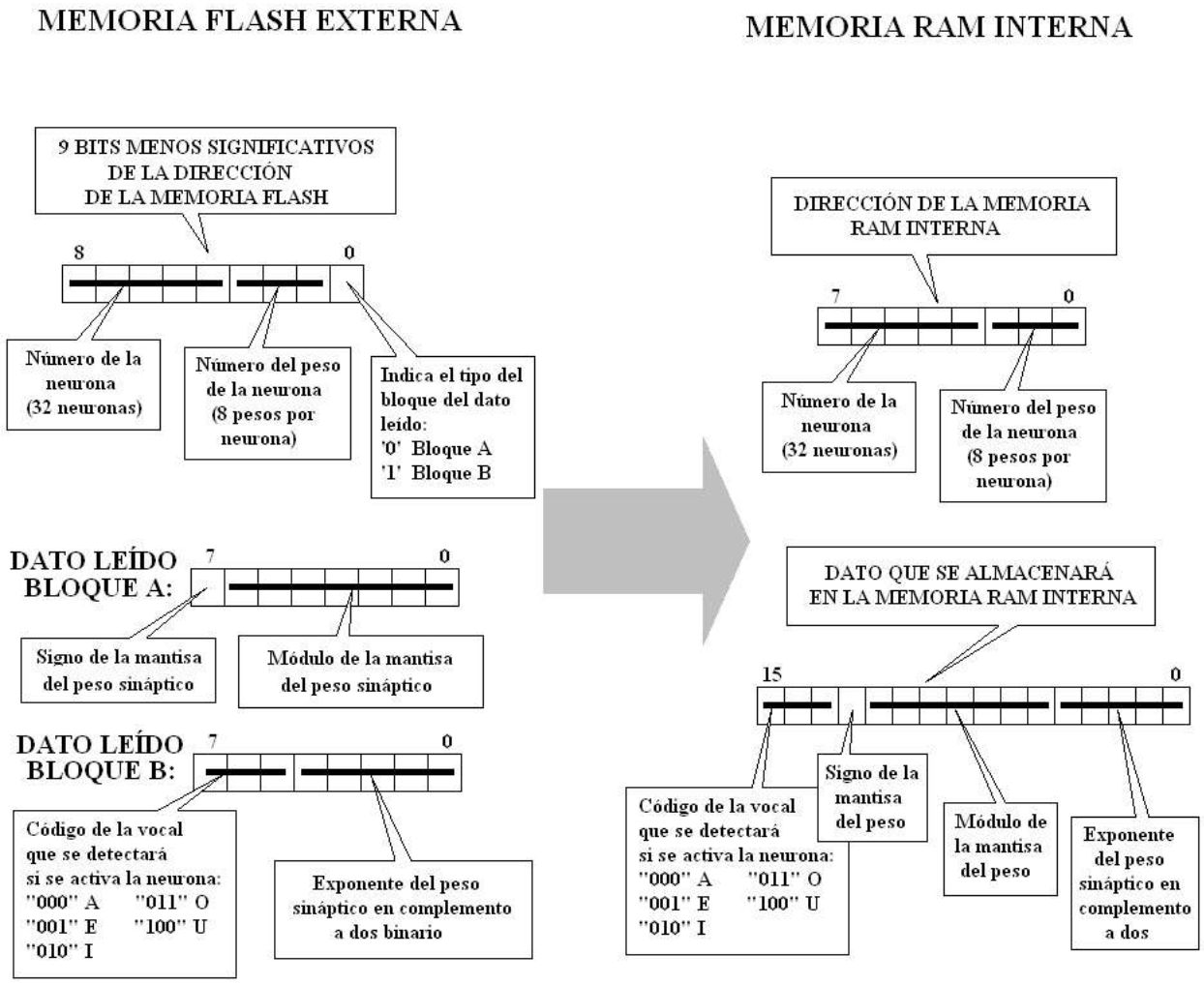
Este módulo realiza la lectura de los pesos sinápticos de las neuronas del mapa autoorganizado desde la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA. Previamente deben ser almacenados en la memoria FLASH estos pesos para posibilitar su posterior lectura. Los pesos se encuentran almacenados en el bloque de parámetros número uno de la memoria FLASH externa desde la dirección hexadecimal "04000" a la dirección hexadecimal "041FF". Cada peso se encuentra almacenado en dos direcciones de memoria sucesivas (la dirección par con un '0' lógico en el bit menos significativo y la dirección impar con un '1' lógico en el bit más significativo). El bit menos significativo del bus de direcciones de la memoria FLASH indica la parte del peso que se está leyendo. Como en cada neurona existen 8 pesos, los 3 bits siguientes del bus de direcciones indican el número del peso de la neurona actual. Y finalmente, como el mapa autoorganizado utiliza 32 neuronas en el proceso de reconocimiento, los 5 bits siguientes del bus de direcciones indican el número de la neurona cuyo peso se leerá. Así se conforman los 9 bits de direcciones donde se almacenan los pesos sinápticos de las neuronas del mapa autoorganizado en la memoria FLASH desde la dirección hexadecimal "04000" a la dirección hexadecimal "041FF".



Los pesos leídos son guardados en la memoria interna "memoria_pesos" para facilitar el posterior acceso a los mismos en el proceso de reconocimiento. En la memoria interna "memoria_pesos" los pesos de las neuronas se almacenan en el formato binario flotante de 13 bits utilizado por la unidad aritmética, pero como la memoria FLASH tiene un bus de datos de sólo 8 bits, es preciso desglosar la lectura de cada uno de los pesos en dos lecturas sucesivas. Divido el peso en dos palabras de 8 bits: una palabra se almacenará en la dirección par de la memoria FLASH (la que tiene el bit menos significativo con un valor '0' lógico) y la otra en la dirección impar de la memoria FLASH (la que tiene el bit menos significativo con un valor '1' lógico). En la primera de las palabras se almacenará el signo del peso en el bit más significativo (bit número 7) y el módulo de la mantisa en los restantes 7 bits. En la segunda palabra se

almacenará en los 3 bits más significativos el código de la vocal que se reconocería si la neurona a la que se corresponde el peso fuera la neurona que se activara en el proceso de reconocimiento y el exponente del peso en complemento a dos binario en los 5 bits menos significativos.

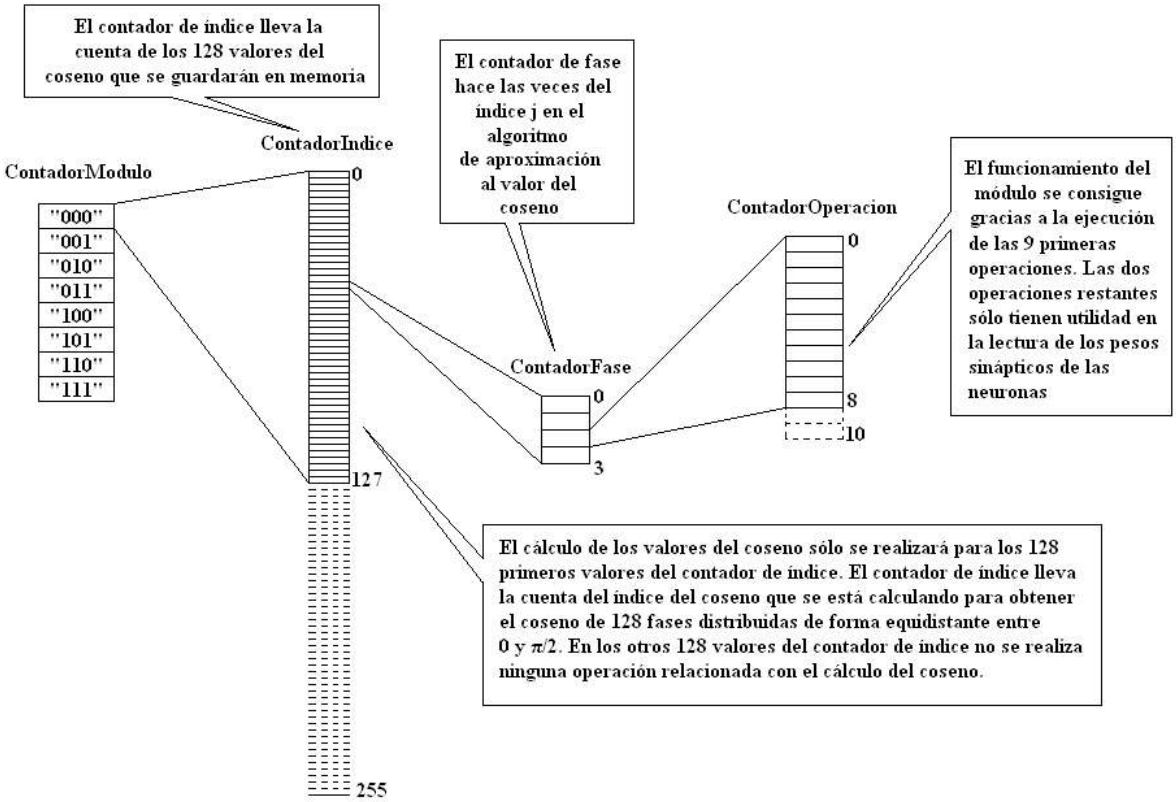
En el proceso de lectura se tomará en primer lugar el módulo y el signo del peso. En una lectura posterior se tomará el exponente por el que se multiplicará la mantisa para conformar el peso en el formato binario flotante de 13 bits utilizado por la unidad aritmética (el bit más significativo es el signo del número, los siguientes 7 bits representan al módulo, y los últimos 5 bits al exponente en base dos del número en complemento a dos). El resultado se almacena en la memoria "memoria_pesos" en la dirección donde los 5 bits más significativos indican el número de la neurona y los 3 bits menos significativos indican el número del peso de la neurona.



Uno a uno se almacenarán todos los pesos de cada una de las neuronas en los 13 bits menos significativos de cada una de las palabras en el formato binario flotante utilizado por la unidad aritmética. En los 3 bits más significativos se almacenará el código de la vocal que se reconocería

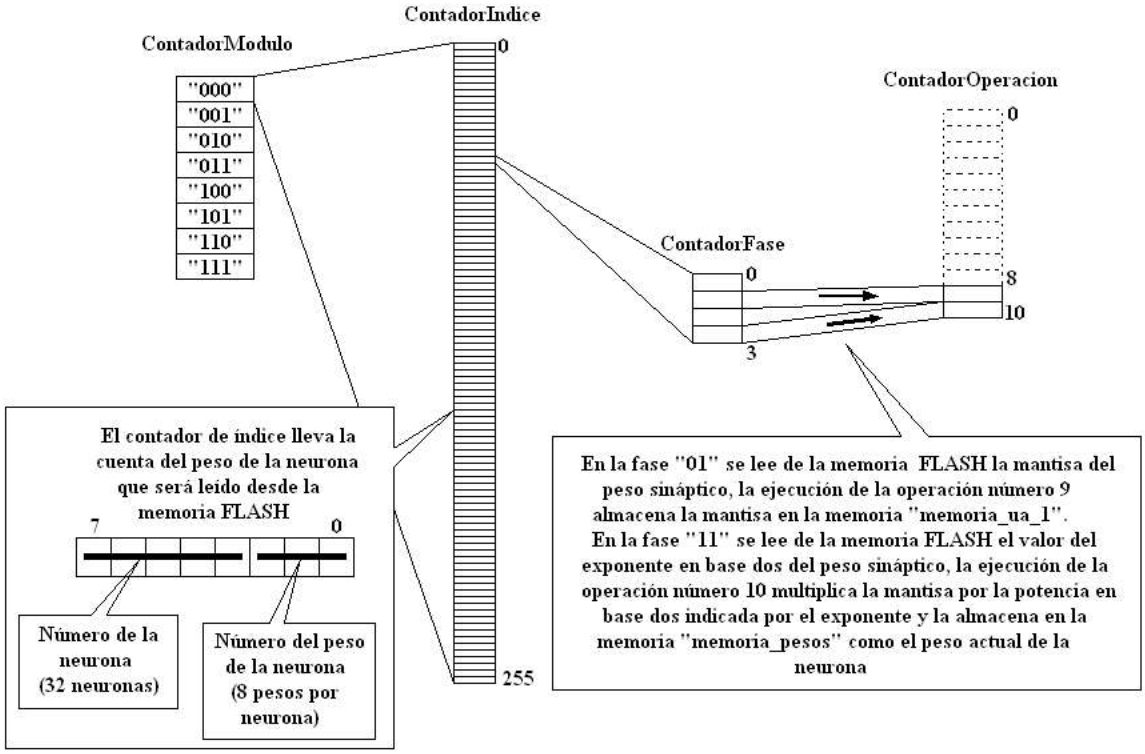
si la neurona correspondiente a ese peso fuera la neurona activada (“000” para la vocal 'A', “001” para la vocal 'E', “010” para la vocal 'I', “011” para la vocal 'O' y “100” para la vocal 'U'). En el proceso de reconocimiento se activará la neurona cuyos pesos se acerquen más a los coeficientes cepstrales de la señal de audio captada. Cada una de las neuronas se corresponde con una vocal de acuerdo con una distribución topológica, por lo que es útil tener almacenado el código de la vocal que se reconocerá si se activa la neurona. Así se elimina la lógica combinacional que se necesitaría para mostrar la vocal que se corresponde con la neurona detectada. También se simplifica el diseño y si es preciso cambiar los parámetros de las neuronas y la distribución topológica de las vocales en el mapa neuronal, sólo será necesario modificar los valores de la memoria externa FLASH y no será necesario modificar el diseño implementado en la FPGA. Cuando se detecte la neurona cuyos pesos más se aproximen a los coeficientes cepstrales, sólo habrá que almacenar el código de la vocal correspondiente a la neurona leído desde los 3 bits más significativos de la memoria RAM interna “memoria_pesos”.

Para desarrollar este algoritmo me aprovecho de la lógica existente en el componente “recnew”. Como en este módulo se realizan dos operaciones (“cálculo y almacenamiento de los valores del coseno que se utilizarán en el cálculo de la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales” y “lectura desde la memoria FLASH externa y almacenamiento en una memoria RAM interna de los pesos sinápticos de las neuronas utilizadas en el proceso de reconocimiento”) se superponen los mecanismos que las realizarán.



Cuando se esté ejecutando el módulo número "000" se activará toda la lógica encargada de implementar el algoritmo de cálculo de los valores del coseno para extraer el funcionamiento particular del módulo de la lógica común de la unidad. En primer lugar se establecen los límites en los bucles de índice, de fase y de operación para recorrer el algoritmo de almacenamiento de los valores del coseno. Se recorrerá el bucle de índice, y en cada uno de los índices se guardará en memoria el valor del coseno indexado por el índice, pero sólo se calculará el valor del coseno para los primeros 128 valores del contador de índice. El bucle de índice tiene 256 valores porque será utilizado en la lectura de los pesos sinápticos de las neuronas, pero sólo los 128 primeros valores del contador de índice serán utilizados en el cálculo de los valores del coseno. Dentro de cada índice se generarán los cuatro sumandos que componen el núcleo del algoritmo, cada sumando se corresponde con un valor del índice j. En este caso es el contador de fase el que lleva la cuenta, por eso el bucle de índice tiene 4 valores. Finalmente, para el cálculo de cada sumando se precisan 9 operaciones y el contador de operación lleva la cuenta de las mismas para que la máquina de estados ejecute cada una de ellas de acuerdo con el valor de cada uno de los contadores.

LECTURA DE LOS PESOS SINÁPTICOS DE LAS NEURONAS

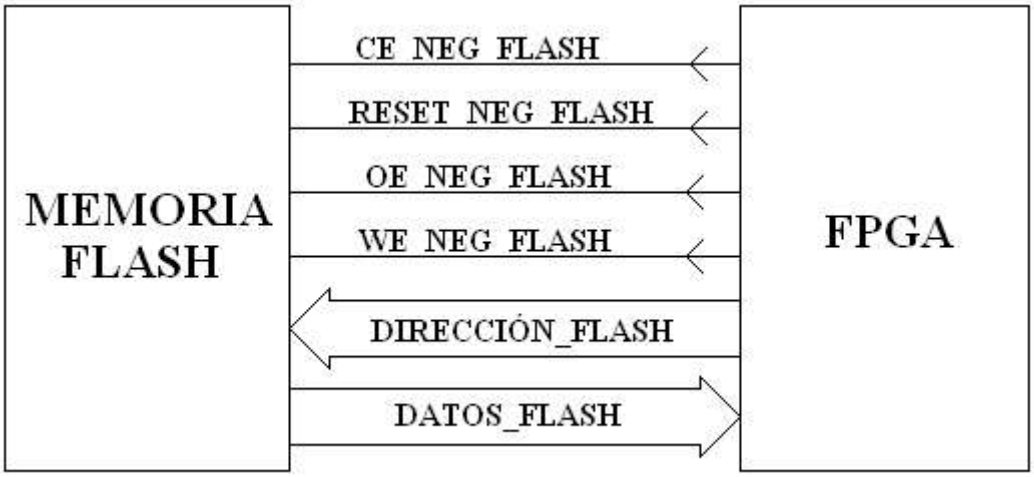


También, durante la ejecución del módulo número "000" se activará toda la lógica encargada de la lectura de los valores de los pesos de las neuronas desde la memoria FLASH externa para almacenarlos en la memoria RAM interna de acceso más rápido. El bucle de índice tiene 256 valores porque será utilizado para indexar los 256 pesos de las neuronas que serán leídos desde la

memoria FLASH (32 neuronas y 8 pesos por cada neurona). Para cada uno de los valores del índice se recorre el bucle de fase. En la fase número "01" se realiza la lectura de la mantisa del peso sináptico desde la dirección par de la memoria FLASH, la ejecución de la operación número 9 almacenará la mantisa en la memoria "memoria_UA_1". En la fase número "11" se realiza la lectura del exponente en base dos del peso sináptico desde la dirección impar de la memoria FLASH, la ejecución de la operación número 10 multiplicará la mantisa por la potencia en base dos cuyo exponente ha sido leído desde la memoria FLASH, la ejecución de la operación número 11 almacenará el resultado en la memoria "memoria_pesos" como peso actual de la neurona. El índice del peso lo marcan los 3 bits menos significativos del contador de índice (8 pesos por cada neurona) y los 5 bits más significativos del contador de índice indican el número de la neurona (32 neuronas).

El control de la memoria FLASH se consigue mediante la activación de las siguientes señales:

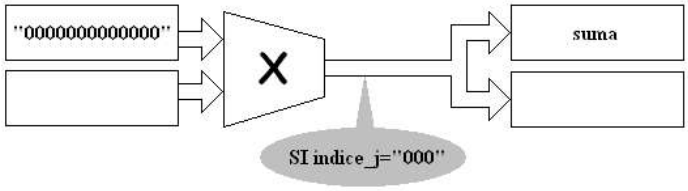
- CE_NEG_FLASH: Para activar la memoria FLASH, esta señal debe estar activa con un valor '0' lógico.
- RESET_NEG_FLASH: Esta señal toma un valor '1' lógico para no poner en marcha el reseteo de los valores almacenados en memoria.
- OE_NEG_FLASH: Los datos almacenados en la dirección de memoria indicada por el bus de direcciones son mostrados por el bus de datos cuando esta señal toma un valor '0' lógico estando habilitada la lectura de datos.
- WE_NEG_FLASH: Se habilita la escritura en la memoria FLASH cuando esta señal toma un valor '0' lógico.
- DIRECCION_FLASH: Dirección de la memoria FLASH desde donde se leerán los valores de los pesos sinápticos de las neuronas.
- DATOS_FLASH: Bus de datos que mostrará el byte almacenado en la memoria principal cuando esté habilitada la lectura y la señal "OE_NEG_FLASH" tome el valor '0' lógico.



La funcionalidad del módulo se consigue mediante la ejecución de las siguientes operaciones:

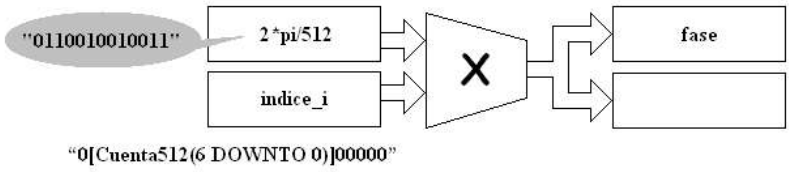
NÚMERO DE OPERACIÓN "0000":
Para cada "índice i", se inicializa la suma con el valor 0.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 a esta entrada con el objetivo de anular el resultado de la unidad aritmética.
- El resultado de la unidad aritmética es 0, y se guardará en la memoria "MemoriaUA0" si el contador de fase es 0. En este lugar se almacenará la suma parcial de la aproximación en series de Taylor, por eso se inicializa la suma cuando el contador de "índice j" es 0 (la cuenta del "índice j" la lleva el contador de fase) con un valor 0. La suma parcial se guardará en la dirección "00000001" de la memoria "MemoriaUA0".



NÚMERO DE OPERACIÓN "0001":
La fase del coseno que se va a calcular es 2*pi*("índice i")/512.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.

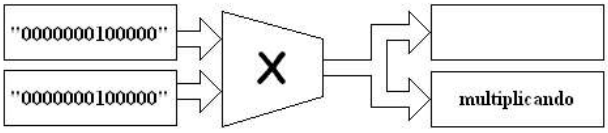


- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 2*pi/512 ("0110010010011") a esta entrada.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará el valor del "índice i" ("0[Cuenta512(6 DOWNTO 0)] 00000" en el formato binario flotante) para esta entrada.

- El resultado de la unidad aritmética es 2*pi*("índice i")/512 que se corresponde con la fase del coseno que se va a calcular. La fase se almacenará en la dirección "00000000" de la memoria "MemoriaUA0".

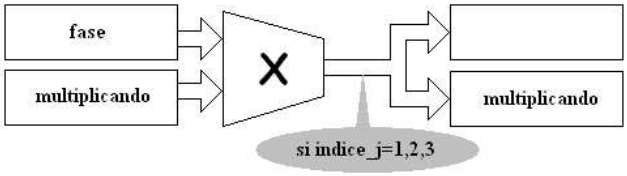
NÚMERO DE OPERACIÓN "0010":
La variable "multiplicando" toma el valor 1.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 1 a esta entrada.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará el valor 1 a esta entrada.
- El resultado de la unidad aritmética es 1, y se guardará en la memoria "MemoriaUA1". En este lugar se almacenará la variable "multiplicando" que inicialmente toma el valor 1. La variable "multiplicando" se guardará en la dirección "00000000" de la memoria "MemoriaUA1".



NÚMERO DE OPERACIÓN "0011":
La variable "multiplicando" toma el valor de la fase del coseno si el "índice j" es 1,2 ó 3.

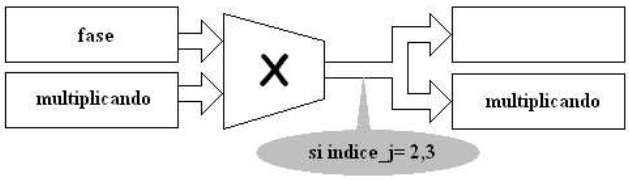
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "MemoriaUA0", donde está guardada la fase del coseno que se va a calcular.



- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “MemoriaUA1”, donde está guardada la variable “multiplicando”.
- El resultado de la unidad aritmética se guardará en la dirección de memoria “00000000” de la memoria “MemoriaUA1” en el caso de que el "índice j" tome el valor 1,2 ó 3. En este lugar se almacenará la variable “multiplicando” modificada en el caso de que la potencia a la que se elevará la fase del coseno sea mayor que 0.

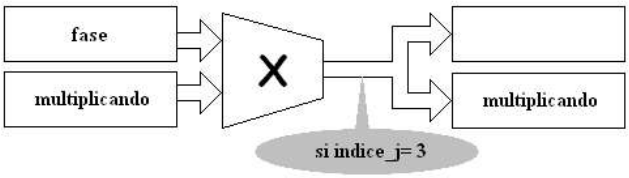
NÚMERO DE OPERACIÓN “0100”:
La variable “multiplicando” toma el valor de la fase del coseno elevada al cuadrado si el "índice j" es 2 ó 3.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “MemoriaUA0”, donde está guardada la fase del coseno que se va a calcular.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “MemoriaUA1”, donde está guardada la variable “multiplicando”.
- El resultado de la unidad aritmética se guardará en la dirección de memoria “00000000” de la memoria “MemoriaUA1” en el caso de que el "índice j" tome el valor 2 ó 3. En este lugar se almacenará la variable “multiplicando” modificada en el caso de que la potencia a la que se elevará la fase del coseno sea mayor que 1.



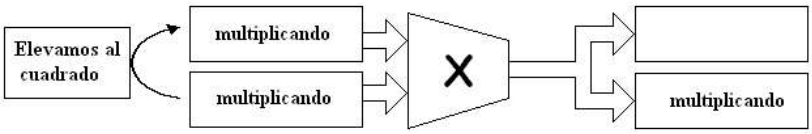
NÚMERO DE OPERACIÓN “0101”:
La variable “multiplicando” toma el valor de la fase del coseno elevada al cubo si el "índice j" es 3.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “MemoriaUA0”, donde está guardada la fase del coseno que se va a calcular.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “MemoriaUA1”, donde está guardada la variable “multiplicando”.
- El resultado de la unidad aritmética se guardará en la dirección de memoria “00000000” de la memoria “MemoriaUA1” en el caso de que el "índice j" tome el valor 3. En este lugar se almacenará la variable “multiplicando” modificada en el caso de que la potencia a la que se elevará la fase del coseno sea 3.



NÚMERO DE OPERACIÓN "0110":
La variable "multiplicando" toma el valor de la fase del coseno elevada al doble del valor del "índice j".

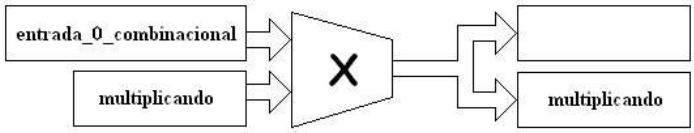
- La operación realizada por la unidad aritmética es una multiplicación, ya que la operación que se está realizando es el cuadrado del segundo operando.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la salida del segundo multiplexor, desde donde se accede a la variable "multiplicando".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "MemoriaUA1", donde está guardada la variable "multiplicando".
- El resultado de la unidad aritmética se guardará en la dirección de memoria "00000000" de la memoria "MemoriaUA1". En este lugar se almacenará la variable "multiplicando" con el valor de la fase del coseno elevada al doble del valor del "índice j".



NÚMERO DE OPERACIÓN "0111":
La variable "multiplicando" es el resultado de dividir por el factorial del doble del "índice j" el valor de la fase del coseno elevada al doble del valor del "índice j".

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará como valor de entrada el inverso del doble del factorial del "índice j". Cuando el "índice j" es 0, el valor es 1/(0!) que en el formato binario flotante es "0100000011010". Cuando el "índice j" es 1, el valor es 1/(2!) que en el formato binario flotante es "0100000011001". Cuando el "índice j" es 2, el valor es 1/(4!) que en el formato binario flotante es "0101010110101". Cuando el "índice j" es 3, el valor es 1/(6!) que en el formato binario flotante es "0010110110001".

- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "MemoriaUA1", donde está guardada la variable "multiplicando".
- El resultado de la unidad aritmética se guardará en la dirección de memoria "00000000" de la memoria "MemoriaUA1". En este lugar se almacenará la variable "multiplicando" con el valor que resulta de dividir por el factorial del doble del "índice j" el valor de la fase del coseno elevada al doble del valor del "índice j".



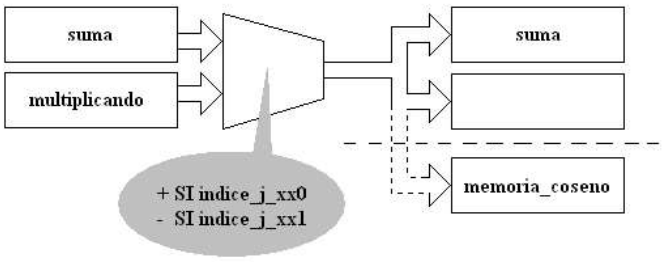

```
entrada_0_combinacional(12)<= '0'
entrada_0_combinacional(11)<=  NOT indice_j_011
entrada_0_combinacional(10)<= indice_j_011
entrada_0_combinacional(9)<= indice_j_010
entrada_0_combinacional(8)<= indice_j_011
entrada_0_combinacional(7)<= indice_j_010 OR indice_j_011
entrada_0_combinacional(6)<= '0'
entrada_0_combinacional(5)<= indice_j_010 OR indice_j_011
entrada_0_combinacional(4)<= '1'
entrada_0_combinacional(3)<= indice_j_x0x
entrada_0_combinacional(2)<= indice_j_010
entrada_0_combinacional(1)<= indice_j_000
entrada_0_combinacional(0)<= NOT indice_j_000
```

entrada_0_combinacional	
indice_j_000 "0100000011010"	$\frac{1}{0!}$
indice_j_001 "0100000011001"	$\frac{1}{2!}$
indice_j_010 "0101010110101"	$\frac{1}{4!}$
indice_j_011 "0010110110001"	$\frac{1}{6!}$

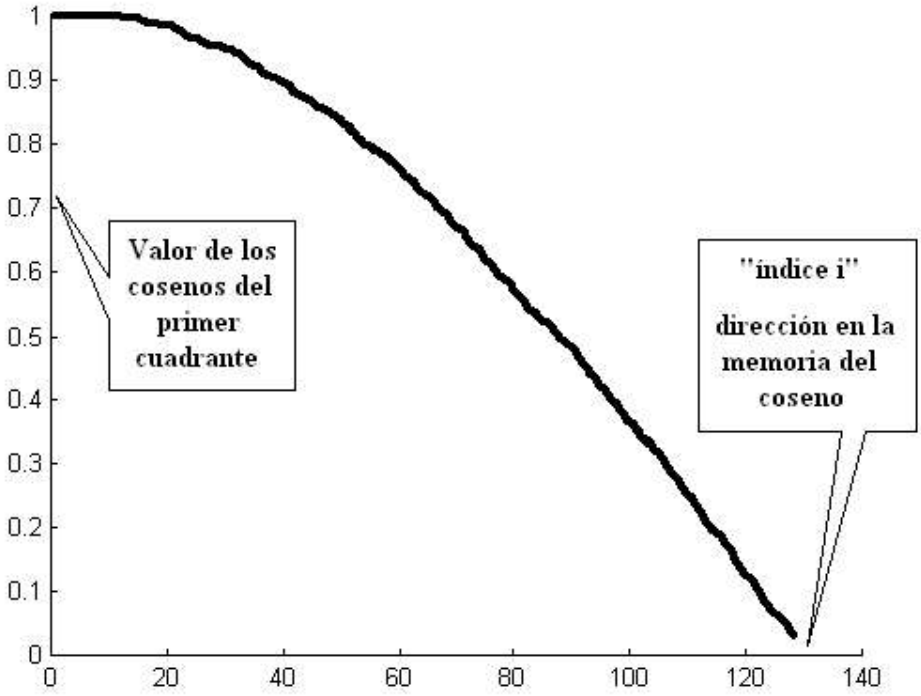
NÚMERO DE OPERACIÓN "1000":
Se actualiza la variable suma añadiendo un nuevo sumando. Si se ha completado la suma (el "índice j" actual es 3) se guardará la suma como el valor del coseno de la fase 2*π*"índice j"/512 radianes en la dirección indicada por el "índice i" de la memoria "MemoriaCoseno".

- La operación realizada por la unidad aritmética es una suma si el "índice j" es par y es una resta si el "índice j" es impar.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000001" de la memoria "MemoriaUA0", donde está guardada la variable "suma".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "MemoriaUA1", donde está guardada la variable "multiplicando".
- El resultado de la unidad aritmética se guardará en la dirección de memoria "00000001" de la memoria "MemoriaUA0". En este lugar se almacenará la variable "suma" con el valor de la suma parcial en la aproximación en series de Taylor del coseno de la fase 2*π*"índice j"/512 radianes. Correspondiente al "índice j". Si se ha completado la suma (el "índice j" actual es 3) se guardará la suma como el valor del coseno de la fase 2*π*"índice j"/512 radianes en la dirección indicada por el "índice i" de la memoria "MemoriaCoseno" sólo si el contador de

índice es menor a 128, ya que el cálculo de los valores del coseno sólo se realiza para los 128 primeros valores del contador de índice.



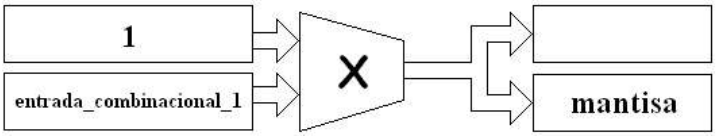
Una vez completadas todas las operaciones de cálculo del coseno quedarán almacenados en la memoria "MemoriaCoseno" el coseno de las 512 fases del primer cuadrante (de 0 a π radianes) separadas entre sí 2*π/512 radianes. Estos valores serán utilizados posteriormente para calcular la Transformada Rápida de Fourier y en el cálculo de los coeficientes cepstrales. Los valores del coseno son sólo una aproximación pero lo bastante cercana como para que los resultados en los cálculos de la Transformada Rápida de Fourier y de los coeficientes cepstrales sean totalmente correctos.

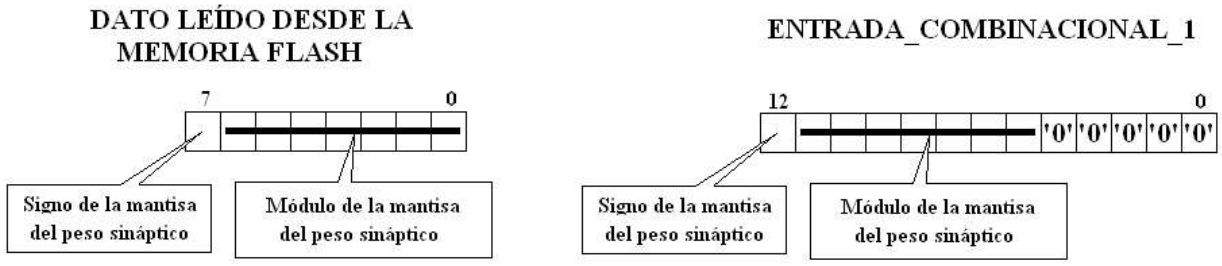


NÚMERO DE OPERACIÓN "1001":
Se lee desde la memoria FLASH la mantisa del peso correspondiente al valor del contador de índice actual. Los 3 bits menos significativos del contador de índice indican el número del peso de la neurona (8 pesos por neurona) y los siguientes 5 bits indican el número de la

neurona (32 neuronas). La mantisa es almacenada en la memoria "memoria_UA_1" si los dos bits menos significativos del contador de fase son "01". Posteriormente se multiplicará por el exponente para obtener el peso cuyo número es indicado por los 3 bits menos significativos del contador de índice de la neurona cuyo número es indicado por los 5 bits siguientes del contador de índice.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará como valor de entrada un uno "0000000100000".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará como valor de entrada la mantisa del peso de la neurona, que es el dato leído desde la memoria FLASH (el bit más significativo del dato leído desde la memoria FLASH es el signo del peso, los restantes 7 bits son el módulo de la mantisa). El bit más significativo de la entrada combinacional se corresponde con el signo del peso, los siguientes 7 bits se corresponden con el módulo de la mantisa y los restantes 5 bits tienen el valor '0' lógico para proporcionar un exponente en base dos nulo.
- El resultado de la unidad aritmética se guardará en la dirección de memoria "10000000" de la memoria "MemoriaUA1" si los dos bits menos significativos del contador de fase son "01". En este lugar se almacenará el módulo del peso actual.



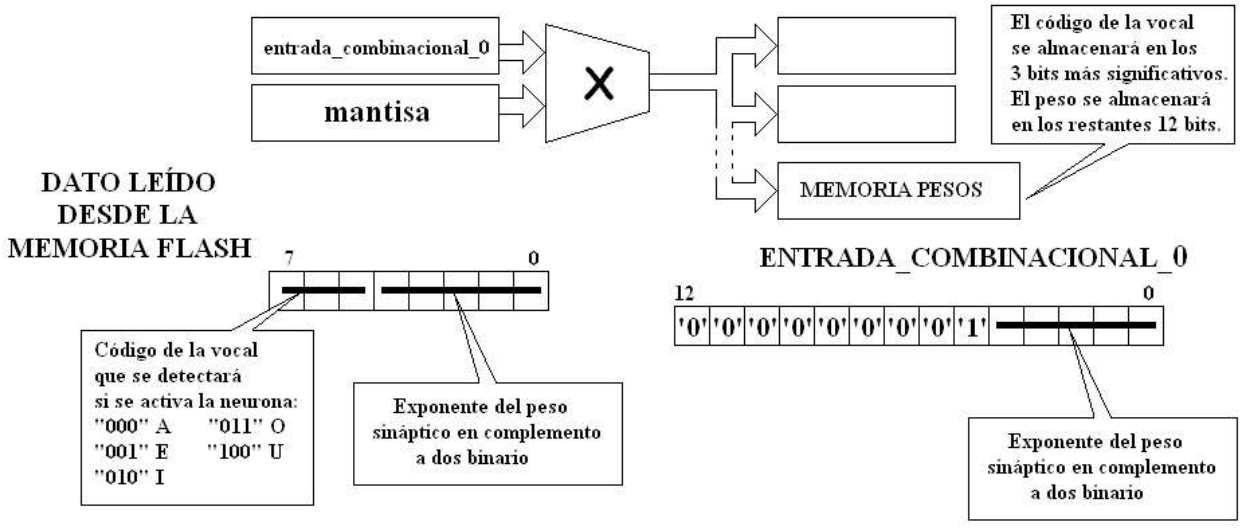


NÚMERO DE OPERACIÓN "1001":

Se lee desde la memoria FLASH el exponente en base dos del peso correspondiente al valor del contador de índice actual. Los 3 bits menos significativos del contador de índice indican el número del peso de la neurona (8 pesos por neurona) y los siguientes 5 bits indican el número de la neurona (32 neuronas). La mantisa del peso actual se encuentra almacenada en la memoria "memoria_UA_1" si los dos bits menos significativos del contador de fase

indican un número superior a "01". Si se multiplica la mantisa por la potencia de dos indicada por el valor del exponente leído desde la memoria FLASH se obtiene el peso cuyo número lo indican los 3 bits menos significativos del contador de índice de la neurona cuyo número lo indican los siguientes 5 bits del contador de índice. Cuando el contador de fase tenga en los dos bits menos significativos el valor "11" se almacenará el peso en la memoria "memoria_pesos" en la dirección indicada por el contador de índice.

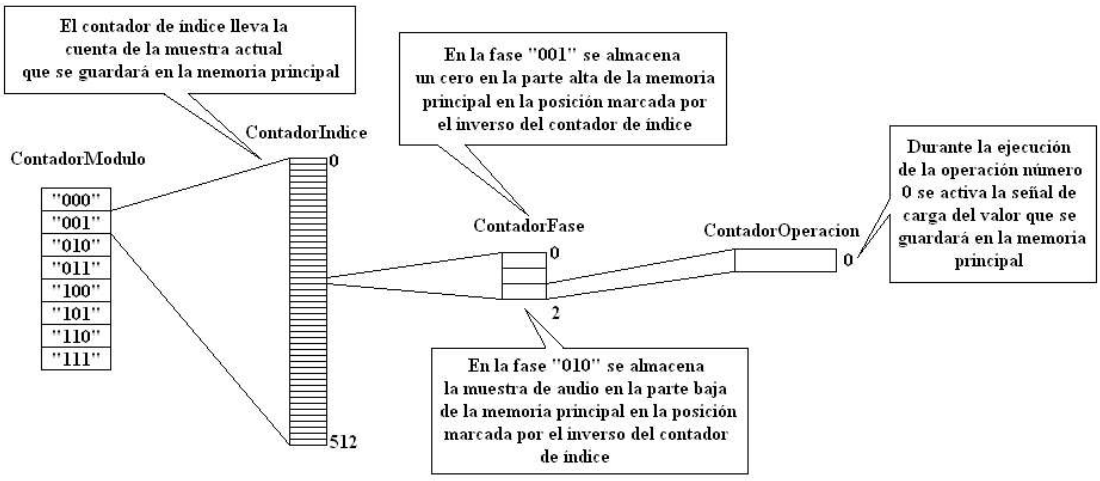
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará como valor de entrada la potencia de dos cuyo exponente tiene el valor del exponente leído desde la memoria FLASH.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de memoria "10000000" de la memoria "MemoriaUA1" donde se encuentra almacenada la mantisa del peso actual.
- El resultado de la unidad aritmética se guardará en la dirección indicada por el contador de índice de la memoria "memoria_pesos" cuando los 2 bits menos significativos del contador de fase tengan el valor "11" binario. En este lugar se almacenará el peso sináptico cuyo número lo indican los 3 bits menos significativos del contador de índice de la neurona cuyo número lo indican los siguientes 5 bits del contador de índice.
- También se almacenarán en los 3 bits más significativos de esa dirección de memoria el código de la vocal que se detectará en el caso de que la neurona actual sea la neurona que se active en el proceso de reconocimiento ("000" para la vocal 'A', "001" para la vocal 'E', "010" para la vocal 'I', "011" para la vocal 'O' y "100" para la vocal 'U'). Así no es necesaria ninguna lógica adicional a la hora de mostrar la vocal reconocida, ya que bastará con leer estos bits desde la memoria "memoria_pesos" para saber la vocal asociada a la neurona. Sólo se almacenará el código cuando los 2 bits menos significativos del contador de fase tengan el valor "11" binario.



1.1.3.0.2	recnew.vhd Módulo=”001”
Durante la ejecución de este módulo se guardarán en la memoria principal las 512 muestras digitales en formato flotante de la señal de audio recibidas desde el conversor analógico-digital.	

Estas 512 muestras se utilizarán para calcular la Transformada Rápida de Fourier y posteriormente los coeficientes cepstrales. Los coeficientes cepstrales se utilizan como base del proceso de reconocimiento vocálico. Cuando realizo la Transformada Rápida de Fourier obtengo 512 coeficientes reales y 512 coeficientes imaginarios. Los coeficientes reales se almacenan en las 512 primeras posiciones de la memoria principal y los coeficientes imaginarios se almacenan en las siguientes 512 posiciones de la memoria principal. Estas direcciones se utilizarán antes en el algoritmo para almacenar las variables intermedias. Y mucho antes, en el inicio de la ejecución del algoritmo, en las primeras 512 posiciones de la memoria principal deben estar almacenadas las 512 muestras en orden inverso (Ejemplo. La primera muestra debe encontrarse en la posición 512 de memoria, la segunda muestra en la posición 511 y así sucesivamente) y las 512 posiciones siguientes deben estar inicializadas con el valor 0. Este módulo se encarga de almacenar las 512 muestras digitales en formato flotante de la señal de audio en las primeras 512 posiciones de la memoria principal en orden inverso y de almacenar un cero en formato flotante en las siguientes 512 posiciones de la memoria principal.

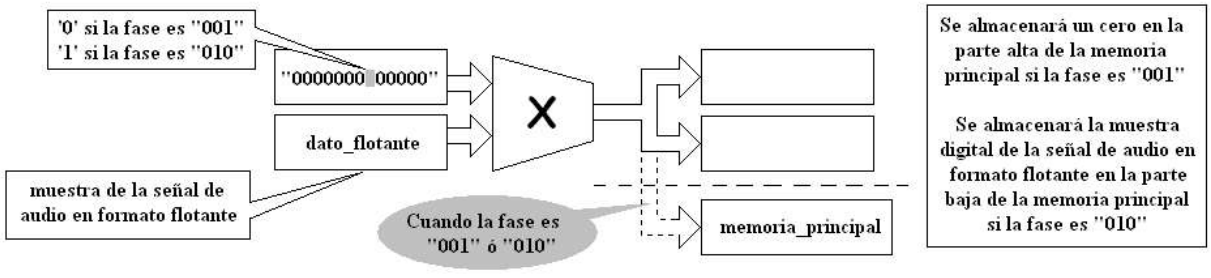
Para guardar las muestras de audio me aprovecho de la lógica existente en el componente “recnew”. Cuando se esté ejecutando el módulo número “001” se activará toda la lógica encargada de guardar las muestras en la memoria principal para extraer el funcionamiento particular del módulo de la lógica común de la unidad. En primer lugar se establecen los límites en los bucles de índice, de fase y de operación para recorrer el algoritmo de almacenamiento de los valores del coseno.



Se recorrerá el bucle de índice, y en cada uno de los índices se guardará en memoria en la parte alta de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '1' lógico) el valor 0 en formato flotante en la dirección obtenida al invertir los bits del contador de índice. También, para cada uno de los índices se almacenará en la parta baja de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '0' lógico) la muestra digital en formato flotante cuyo número se corresponde con el valor del índice en la dirección de la parte baja de la memoria principal que se forma al invertir el orden de los bits del contador de índice. Como son necesarias 512 muestras, el bucle del índice tiene 128 valores. Dentro de cada índice se recorren 3 fases ("000", "001" y "010"). En la fase "001" se almacena el valor 0 en la parte alta de la memoria principal. Estas posiciones son utilizadas como variables auxiliares imaginarias en el cálculo de la Transformada Rápida de Fourier. En la fase "010" se almacena la muestra recibida en la dirección inversa del índice de la parte baja de la memoria principal. Estas posiciones son utilizadas como variables auxiliares reales en el cálculo de la Transformada Rápida de Fourier. Finalmente, en el proceso de almacenamiento se requiere de una única operación, la que active la carga en la memoria principal del valor 0 o de la muestra en la dirección oportuna dependiendo del valor de la fase. Por eso el bucle de operación tiene un único elemento.

NÚMERO DE OPERACIÓN "0000":
Se almacena un 0 en la dirección formada por el inverso del contador de índice en la parte alta de la memoria principal si la fase es "001". Se almacena la muestra de audio cuyo número coincide con el contador de índice en la dirección formada por el inverso del contador de índice en la parte baja de la memoria principal si la fase es "010".

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 ("0000000000000" en formato flotante) a esta entrada si la fase es "001" y un 1 ("0000000100000" en formato flotante) a esta entrada si la fase es "010".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada binaria que representa a la muestra de audio en formato flotante.
- El resultado de la unidad aritmética se guardará en la parte alta de la memoria principal si la fase es "001" y en la parte baja de la memoria principal si la fase es "010" en la dirección generada al invertir los bits del contador de índice.



1.1.3.0.3	recnew.vhd Módulo=”010”
Durante la ejecución de este módulo se realiza la Transformada Rápida de Fourier de 512 muestras recibidas desde el conversor analógico-digital de la señal de audio. Los coeficientes de Fourier finales reales e imaginarios son almacenados en la memoria principal.	

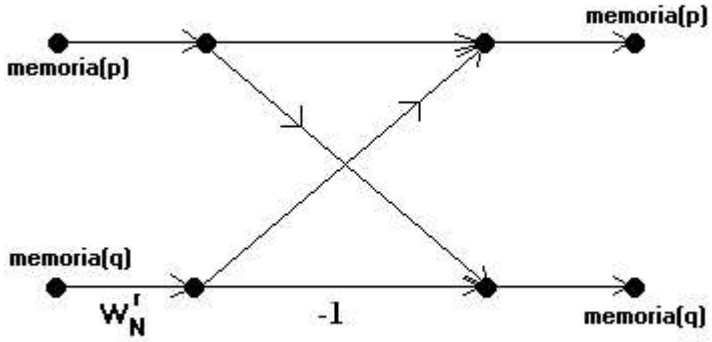
El módulo “010” toma como entradas para su procesamiento 512 muestras digitales de la señal de audio, ya almacenadas por el módulo anterior en la memoria principal. Estas muestras son los valores iniciales de las 512 variables reales utilizadas en el algoritmo de la Transformada Rápida de Fourier. Las 512 variables imaginarias tienen el valor 0 como valor inicial. El módulo anterior ya inicializó estas variables.

Una vez terminado el módulo, quedarán almacenados los 512 coeficientes reales y los 512 coeficientes imaginarios que conforman los coeficientes de Fourier. Estos coeficientes proporcionan las características espectrales de la señal de audio cuyas muestras se han tratado. Las muestras fueron almacenadas en las 512 primeras posiciones de la parte más baja de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '0' lógico) en el orden que resulta al invertir la posición de los bits del número que marcaba el orden de llegada de las muestras. Ejemplo: La primera muestra debe encontrarse en la posición 512 de memoria, la segunda muestra en la posición 511 y así sucesivamente. El dato que llegue en la posición 18, que se corresponde con el número binario ‘000010010’, será guardado en la posición 493, que se corresponde con el número binario ‘111101101’, inverso binario de la posición de entrada. Estas posiciones de memoria conforman el lugar de almacenamiento de las variables reales utilizadas en el algoritmo de cálculo de la Transformada Rápida de Fourier. En las 512 posiciones siguientes (la parte alta de la memoria principal, aquella que tiene el bit más significativo de la dirección con un valor '1' lógico) se almacenarán las 512 variables imaginarias utilizadas en el algoritmo de cálculo de la Transformada Rápida de Fourier. Estas variables fueron inicializadas con un valor nulo por el módulo anterior.

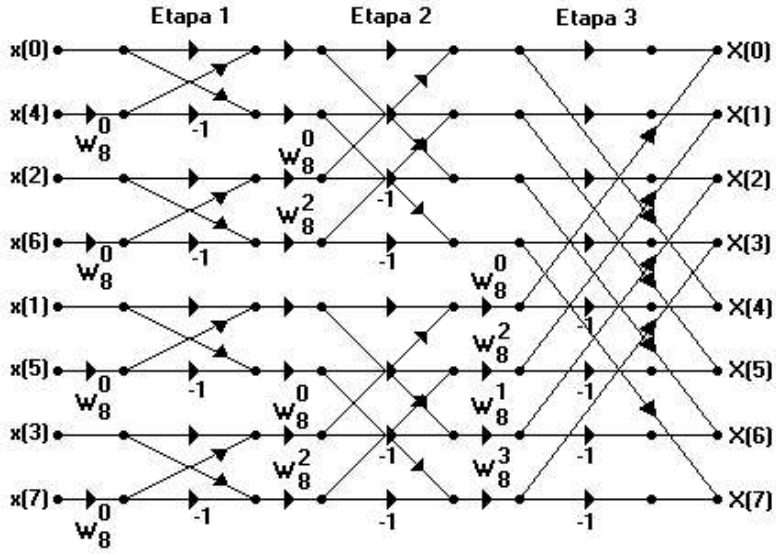
Con las variables almacenadas en la memoria principal se operará y de nuevo, los resultados serán guardados en las mismas posiciones de memoria de acuerdo con las especificaciones del algoritmo. El orden inicial de los datos tomado como el orden de las posiciones inversas binarias es necesario para que las salidas proporcionen en orden normal los diferentes coeficientes de la Transformada de Fourier.

Se recorren 9 etapas en el algoritmo, ya que el número de etapas se corresponde con el exponente de la potencia en base 2 que conforma el número de muestras (512 muestras). El cálculo básico que se realiza en cada etapa consiste en coger dos números complejos (dos variables diferentes de entrada que se almacenarán en la memoria principal) , multiplicar por un factor de fase (número complejo de módulo unidad que realiza una modificación en la fase) la segunda de las

variables, y sumar y restar el producto obtenido a la primera de las variables para obtener los dos nuevos números complejos (variables de salida que se almacenarán en la memoria principal). Este cálculo básico se denomina mariposa, dado que el diagrama de flujo recuerda a una mariposa.



En cada etapa se realizan los cálculos de 256 mariposas para actualizar los valores de los datos guardados en las distintas posiciones de memoria. Si el número de muestras fuera 8, los cálculos del algoritmo FFT podrían resumirse en el siguiente esquema. Imagínese cómo sería para 512 muestras.



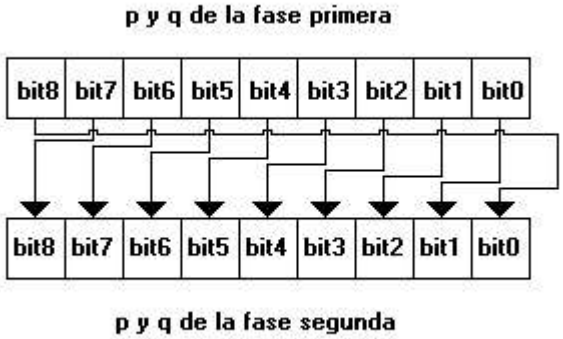
Esquema del algoritmo de cálculo de la FFT

$$W_8^r = e^{-j\,r\,2\pi/8}$$

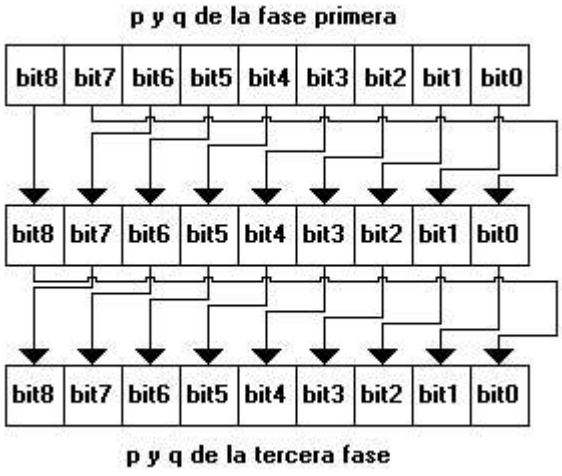
En cada mariposa se obtienen los índices p y q de las posiciones de memoria que se actualizarán con los valores guardados en esas mismas posiciones de memoria. El componente "obten_pq" proporcionará los índices p y q de la mariposa en función del número de mariposa que se esté calculando (0-255) y de la etapa actual en curso (1-9).

Para la etapa primera, se toma para el índice p el número de la mariposa que se está calculando multiplicado por dos y para q el número de la mariposa que se está calculando multiplicado por

dos mas uno. Esto es así porque las posiciones de memoria ya están ordenadas apropiadamente para producir las salidas deseadas. En las siguientes fases se debe variar el orden de acuerdo con la regla que descrita a continuación. En la etapa segunda los nuevos p y q se calculan a partir de los de la primera etapa, trasladando el bit8 a la posición 0 y desplazando hacia la izquierda los restantes bits.

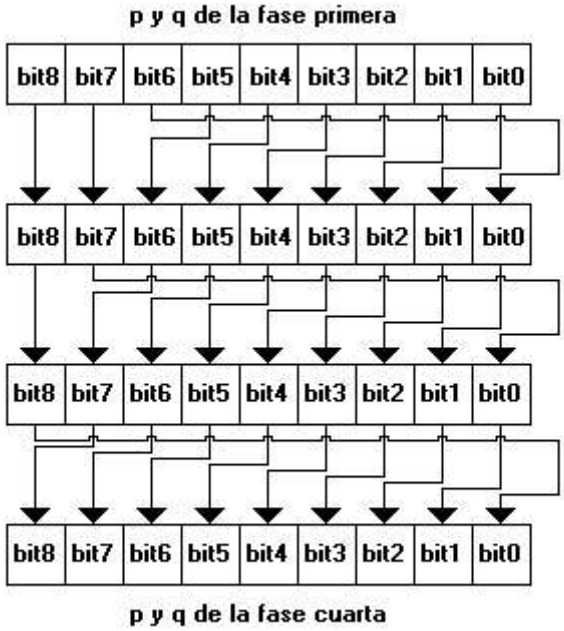


La tercera etapa, se calcula también con los bits de la primera. En primer lugar, trasladamos el bit8 a su misma posición, el bit7 lo trasladamos a la posición del bit0, y el resto de los bits los trasladamos hacia la izquierda. A estos bits, posteriormente les aplicamos el mismo proceso que necesitamos para obtener p y q en la fase anterior, trasladando el bit8 a la posición 0 y desplazando hacia la izquierda los restantes bits.

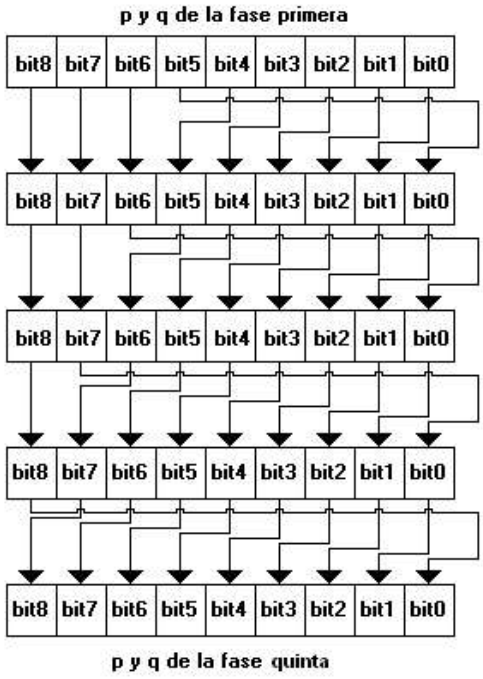


En la etapa cuarta, el bit8 y el bit7 de p y q de la primera etapa se trasladan a las mismas posiciones. El bit6 lo trasladamos a la posición del bit0 y el resto de los bits los desplazamos una posición hacia la izquierda. Posteriormente, les aplicamos el mismo proceso que necesitamos para obtener p y q en la fase anterior. Trasladamos el bit8 a su misma posición, el bit7 lo trasladamos a la posición del bit0, y el resto de los bits los trasladamos hacia la izquierda.

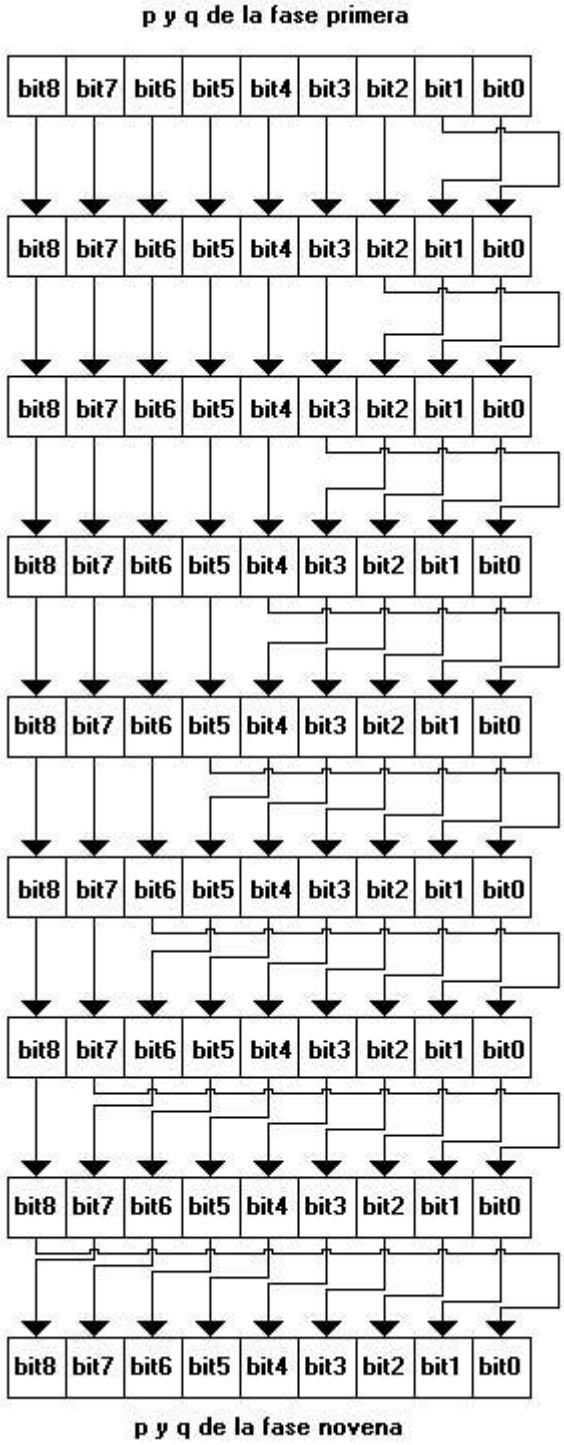
Seguidamente, desplazamos el bit8 a la posición 0 y el resto de los bits una posición hacia la izquierda.



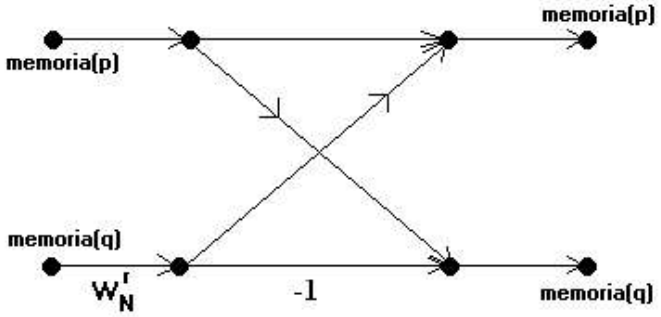
En la etapa quinta, el bit8,el bit7 y el bit6 de p y q de la primera etapa se trasladan a las mismas posiciones. El bit5 lo trasladamos a la posición del bit0 y el resto de los bits los desplazamos una posición hacia la izquierda. Posteriormente, les aplicamos el mismo proceso que necesitamos para obtener p y q en la fase anterior.



El proceso continuaría de la misma forma, y así, en la fase novena realizaríamos las siguientes traslaciones de bits:



Se observa que existen 8 etapas en las que se producen desplazamientos, desde la etapa segunda a la etapa novena. La etapa novena es la más completa, en la que los desplazamientos se generan en 8 pasos. Para simplificar el código VHDL capaz de generar los índices p y q puedo tomar estos 8 pasos como el caso más general. Las demás etapas se obtendrían por eliminación sucesiva de los primeros pasos. Para la etapa octava eliminaríamos el primer paso, para la etapa séptima eliminaríamos los dos primeros pasos, para la etapa sexta eliminaríamos los tres primeros pasos, ..., y así sucesivamente. De esta forma reducimos el número de puertas lógicas necesarias para implementar el proceso de obtención de p y q.



$$W_N^r = e^{-i2\pi r/N}$$

En los cálculos de cada mariposa se realiza una multiplicación por un factor de fase. El factor de fase es un número complejo de módulo unidad que realiza una modificación en la fase del dato al que se le realiza la multiplicación. El coeficiente r es el que marca la variación de fase que provocará el factor. El coeficiente r en la etapa 9 es este número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255). Ejemplo: El número 31 que en binario es '00011111' se corresponde con el número 248 que en binario es '11111000'. En la etapa 8, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda introduciendo un cero por la derecha. En la etapa 7, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda dos posiciones introduciendo dos ceros por la derecha. En la etapa 6, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda tres posiciones introduciendo tres ceros por la derecha. En la etapa 5, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda cuatro posiciones introduciendo cuatro ceros por la derecha. En la etapa 4, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda cinco posiciones introduciendo cinco ceros por la derecha. En la etapa 3, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda seis posiciones introduciendo seis ceros por la derecha. En la etapa 2, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la mariposa que estamos calculando (0-255) desplazado hacia la izquierda siete posiciones introduciendo siete ceros por la derecha. En la etapa 1, el coeficiente r es el número resultado de cambiar el orden de los bits del número de la

mariposa que estamos calculando (0-255) desplazado hacia la izquierda ocho posiciones introduciendo ocho ceros por la derecha. Con el coeficiente r se calcula el coeficiente W:

$$W_N^r = e^{-i2\pi r/N}$$

Este coeficiente se corresponde con el siguiente valor:

$$W = \cos(2 \pi r / N) - j \operatorname{sen}(2 \pi r / N)$$

El valor del seno y del coseno de la fase (2 π r / N) se obtiene en función de los cosenos almacenados en la memoria "memoria_coseno". El componente "calcula_direccion_wrn" obtiene la dirección de la memoria "memoria_coseno" en función del coeficiente r y dependiendo de si se quiere obtener el valor del seno o del coseno.

En la ejecución del módulo número "000" se calcularon 128 valores del coseno para que se utilizasen posteriormente como aproximaciones del coseno. Estos 128 valores se corresponden con el coseno de 128 fases distribuidas de forma equidistante entre 0 y π/2. El resto de valores del coseno entre π/2 y 2π no es necesario guardarlos , ya que se obtienen a partir de los anteriores. Estos valores se utilizarán para aproximar la operación coseno. Para realizar esta operación, lo único que hay que hacer es averiguar cuál es la dirección de memoria donde está el valor correspondiente. Una lógica especializada será la encargada de proporcionar la dirección de lectura de la memoria donde leer el valor del coseno de la fase oportuna.

- Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el primer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el índice r.

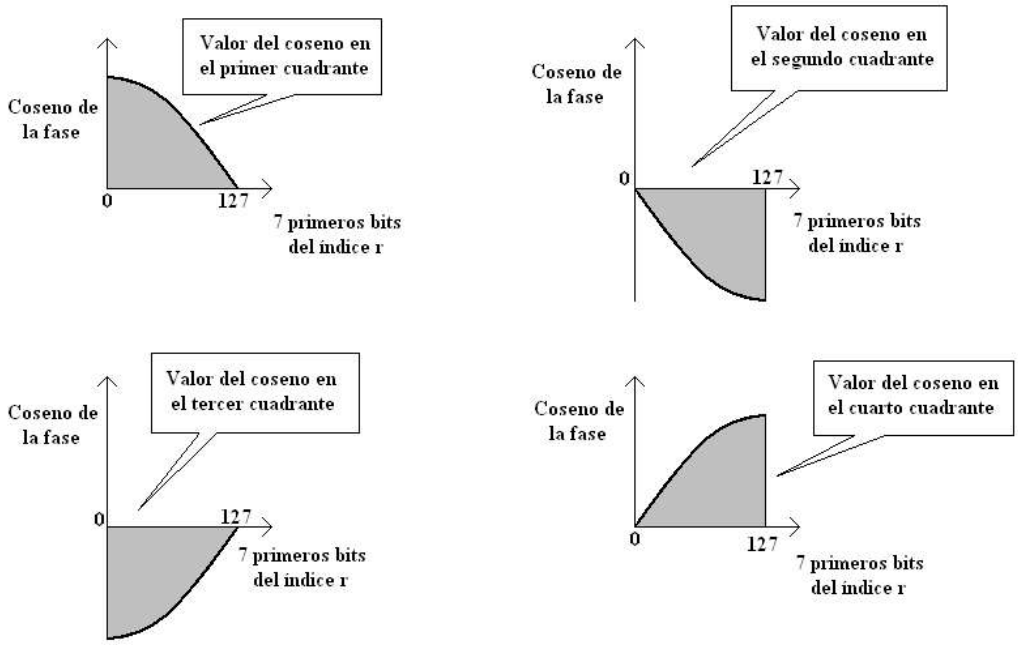
- Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el segundo cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir todos los bits de '0' a '1' y de '1' a '0' y luego sumarle una unidad). Luego habrá que cambiar el signo del valor obtenido.

- Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el tercer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el valor del índice r. Luego habrá que cambiar el signo del valor obtenido.

- Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el cuarto cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno

es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir todos los bits de '0' a '1' y de '1' a '0' y luego sumarle una unidad).

- El valor del coseno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del índice r tiene el valor '1' lógico.



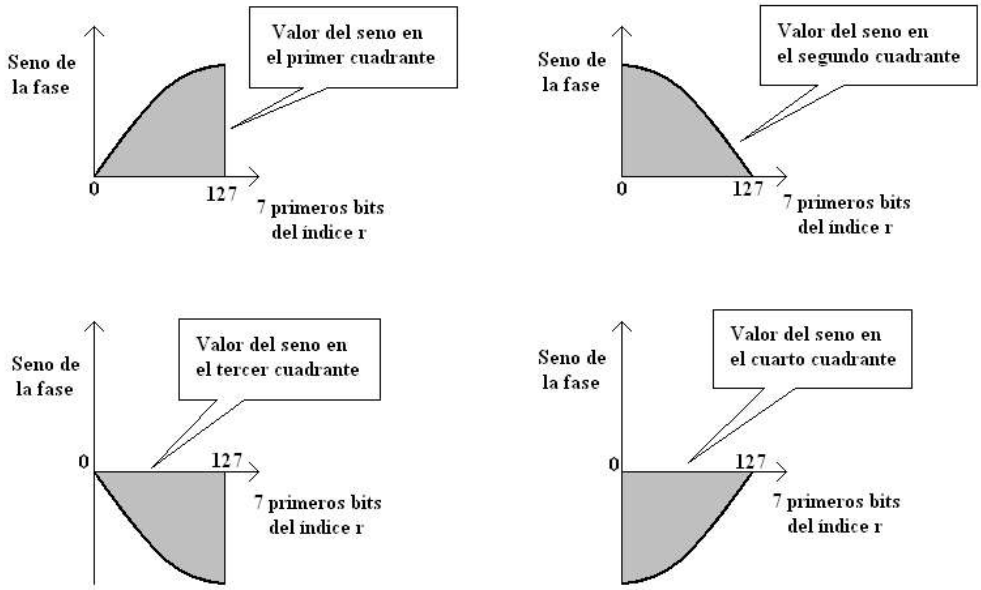
- Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el primer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir todos los bits de '0' a '1' y de '1' a '0' y luego sumarle una unidad).

- Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el segundo cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el índice r.

- Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el tercer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir todos los bits de '0' a '1' y de '1' a '0' y luego sumarle una unidad). Luego habrá que cambiar el signo del valor obtenido.

- Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el cuarto cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el valor del índice r. Luego habrá que cambiar el signo del valor obtenido.

- El valor del seno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del índice r tiene el valor '0' lógico.



Para realizar los cálculos de los coeficientes de Fourier utilizando la Transformada Rápida de Fourier se ejecuta el siguiente algoritmo (se utiliza el lenguaje Matlab® para describir el algoritmo), que realiza la función anterior:

```
for i=1:1:512
  i_invertido=func_invierte_orden_bits(i);
  memoria_real(i)=muestra_flotante(i_invertido);
  memoria_imaginaria(i)=0;
end

for etapa=0:1:8
  for mariposa=0:1:255

    [p,q]=obten_pq(etapa,mariposa);
    r=obten_r(etapa,mariposa);

    wrn_real=obten_wrn_real(r);

    producto= wrn_real * memoria_real(q);
    q_real= memoria_real(p) – producto;
    p_real= memoria_real(p) + producto;

    producto= wrn_real * memoria_imaginaria(q);
    q_imaginaria= memoria_imaginaria(p) – producto;
    p_imaginaria= memoria_imaginaria(p) + producto;

    wrn_imaginario=obten_wrn_imaginario(r);
```

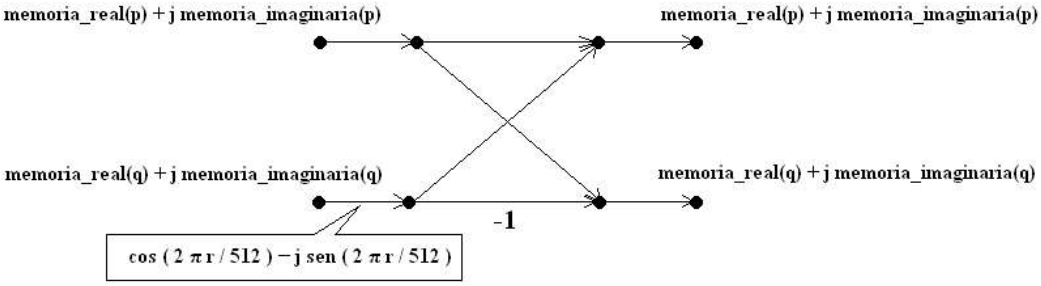
```
        producto= wrn_imaginario * memoria_real(q);
        memoria_imaginaria(q)= q_imaginario + producto;
        memoria_imaginaria(p)= p_imaginario - producto;

        producto= wrn_imaginario * memoria_imaginaria(q);
        memoria_real(q)= q_real - producto;
        memoria_real(p)= p_real + producto;

      end
    end
```

El módulo anterior ya guardó en la memoria principal las 512 muestras de la señal de audio en las 512 primeras posiciones (parte baja de la memoria) en orden inverso al orden de llegada. Ejemplo: La primera muestra debe encontrarse en la posición 512 de memoria, la segunda muestra en la posición 511 y así sucesivamente. El dato que llegue en la posición 18, que se corresponde con el número binario '000010010', será guardado en la posición 493, que se corresponde con el número binario '111101101', inverso binario de la posición de entrada. Estas posiciones de memoria conforman el lugar de almacenamiento de las variables reales utilizadas en el algoritmo de cálculo de la Transformada Rápida de Fourier. En las 512 posiciones siguientes (la parte alta de la memoria principal) se almacenó un valor 0 en cada una de ellas. Estas posiciones de memoria conforman el lugar de almacenamiento de las variables imaginarias utilizadas en el algoritmo de cálculo de la Transformada Rápida de Fourier. La parte primera del algoritmo anterior realiza estas operaciones. La función "func_invierte_orden_bits" realiza estas operaciones, después inicializa las posiciones de memoria donde se guardarán los valores reales con las muestras recibidas en orden inverso. Luego inicializa las posiciones de memoria donde se guardarán los valores imaginarios con valores nulos.

Luego se recorren las 9 etapas de procesamiento de datos en el algoritmo FFT (0-8). En cada una de las etapas se realizan los cálculos de 256 mariposas (0-255). En cada mariposa se realizan 2 multiplicaciones complejas, dos sumas complejas y dos restas complejas.



Es necesario conocer los índices p y q, donde están almacenadas las dos variables complejas utilizadas en la mariposa. El índice p se corresponde con la dirección de la memoria principal donde está guardada la primera variable de la mariposa. La parte real de la primera variable está almacenada en la parte inferior de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '0' lógico) en la subdirección p. La parte imaginaria de la primera variable está almacenada en la parte superior de la memoria principal (aquella que tiene el bit

más significativo de la dirección con un valor '1' lógico) en la subdirección p. El índice q se corresponde con la dirección de la memoria principal donde está guardada la segunda variable de la mariposa. La parte real de la segunda variable está almacenada en la parte inferior de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '0' lógico) en la subdirección q. La parte imaginaria de la segunda variable está almacenada en la parte superior de la memoria principal (aquella que tiene el bit más significativo de la dirección con un valor '1' lógico) en la subdirección q. El componente "ComponenteObtenPQ" obtiene los índices p y q (el mismo módulo obtiene los dos índices, de forma que el índice obtenido sólo depende del número de operación que se está realizando) en función del número de etapa y del número de la mariposa que se está calculando. En el algoritmo anterior es la función "obten_pq" la que obtiene estos índices.

Las variables anteriores son multiplicadas por el coeficiente W (cos(2 π r / 512) - j sen(2 π r / 512)). El componente "ComponenteObtenR" obtiene el coeficiente r de las operaciones anteriores en función del número de etapa y del número de la mariposa que se está calculando. En la memoria coseno se tienen almacenados los valores de 128 cosenos (cos(2 π i / 512), donde i va desde 0 a 127). Los cosenos y senos de los coeficientes W pueden obtenerse de los valores almacenados en la memoria coseno, sólo hay que averiguar la dirección de memoria donde está almacenado el valor que se está buscando. El componente "ComponenteCalculaDireccionWRN" halla la dirección de la memoria coseno desde donde se leerá el valor cos(2 π r / 512) o el valor sen(2 π r / 512), en función del coeficiente r y de si se quiere leer un coseno o un seno. Las funciones "obten_wrn_real" y "obten_wrn_imaginario" del algoritmo descrito anteriormente realizan estas operaciones y calculan los valores cos(2 π r / 512) y sen(2 π r / 512) en función del coeficiente r.

Como los cálculos de la mariposa son complejos, deben descomponerse las operaciones de la mariposa.

OPERACIONES COMPLEJAS REALIZADAS POR LA MARIPOSA

$$\text{memoria\_real(p)} + j\,\text{memoria\_imaginaria(p)} = \text{memoria\_real(p)} + j\,\text{memoria\_imaginaria(p)} + \left(\left(\text{memoria\_real(q)} + j\,\text{memoria\_imaginaria(q)} \right) \times \left(\cos\left(2\pi r / 512\right) - j\,\text{sen}\left(2\pi r / 512\right) \right) \right)$$

$$\text{memoria\_real(q)} + j\,\text{memoria\_imaginaria(q)} = \text{memoria\_real(p)} + j\,\text{memoria\_imaginaria(p)} - \left(\left(\text{memoria\_real(q)} + j\,\text{memoria\_imaginaria(q)} \right) \times \left(\cos\left(2\pi r / 512\right) - j\,\text{sen}\left(2\pi r / 512\right) \right) \right)$$

DESGLOSE DE OPERACIONES DE LA PRIMERA VARIABLE (ÍNDICE P)

$$\text{memoria\_real(p)} = \text{memoria\_real(p)} + \left(\text{memoria\_real(q)} \times \cos\left(2\pi r / 512\right) \right) + \left(\text{memoria\_imaginaria(q)} \times \text{sen}\left(2\pi r / 512\right) \right)$$

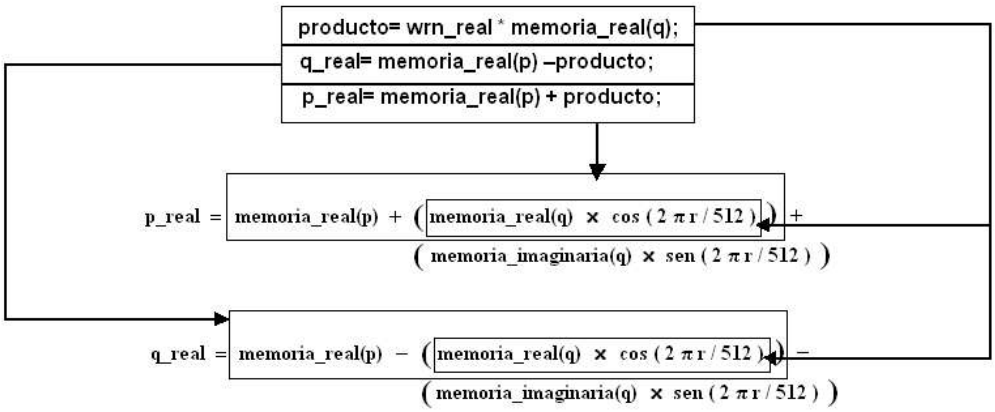
$$\text{memoria\_imaginaria(p)} = \text{memoria\_imaginaria(p)} + \left(\text{memoria\_imaginaria(q)} \times \cos\left(2\pi r / 512\right) \right) - \left(\text{memoria\_real(q)} \times \text{sen}\left(2\pi r / 512\right) \right)$$

DESGLOSE DE OPERACIONES DE LA SEGUNDA VARIABLE (ÍNDICE Q)

$$memoria\_real(q) = memoria\_real(p) - \left(memoria\_real(q) \times \cos(2\pi r / 512) \right) + \left(memoria\_imaginaria(q) \times sen(2\pi r / 512) \right)$$

$$memoria\_imaginaria(q) = memoria\_imaginaria(p) - \left(memoria\_imaginaria(q) \times \cos(2\pi r / 512) \right) + \left(memoria\_real(q) \times sen(2\pi r / 512) \right)$$

El algoritmo FFT que se ha implementado realiza las operaciones anteriormente mencionadas con el código Matlab®. El primer bloque de cálculos realiza la primera semi-operación de la actualización en las variables espectrales reales.



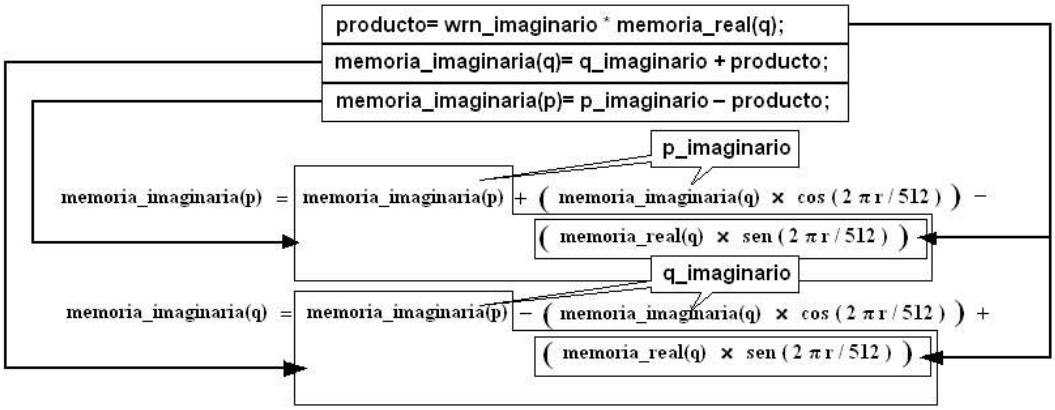
El segundo bloque de cálculos realiza la primera semi-operación en la actualización de las variables espectrales imaginarias.

DESGLOSE DE OPERACIONES DE LA SEGUNDA VARIABLE (ÍNDICE Q)

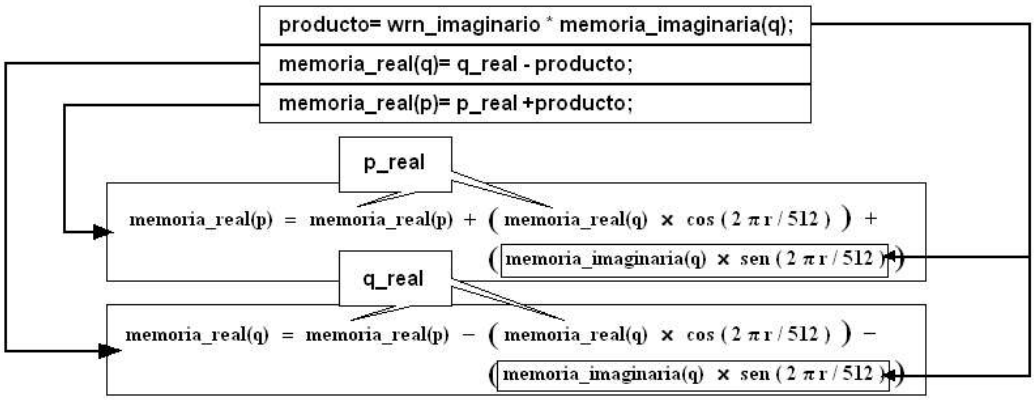
$$memoria\_real(q) = memoria\_real(p) - \left(memoria\_real(q) \times \cos(2\pi r / 512) \right) - \left(memoria\_imaginaria(q) \times sen(2\pi r / 512) \right)$$

$$memoria\_imaginaria(q) = memoria\_imaginaria(p) - \left(memoria\_imaginaria(q) \times \cos(2\pi r / 512) \right) + \left(memoria\_real(q) \times sen(2\pi r / 512) \right)$$

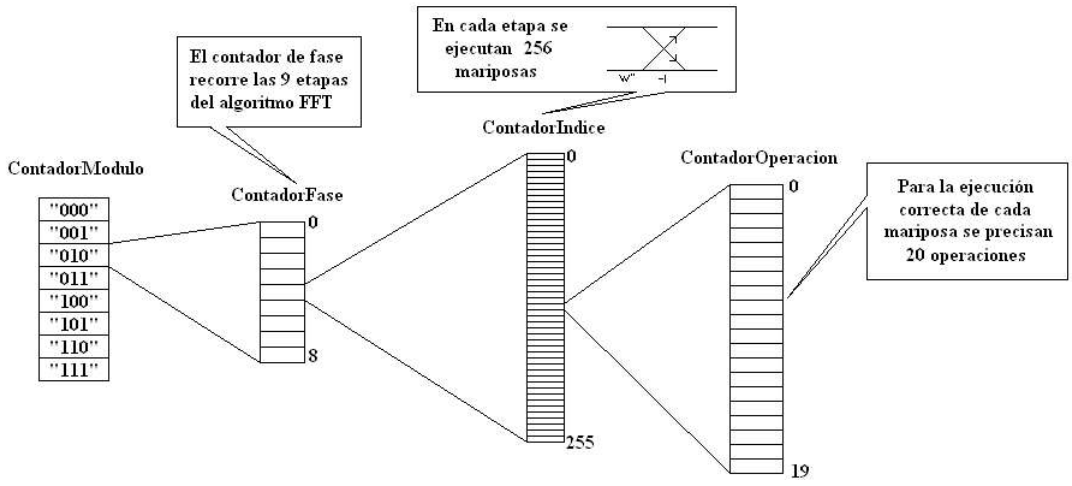
El tercer bloque de cálculos realiza la segunda semi-operación en la actualización de las variables espectrales imaginarias.



El cuarto bloque de cálculos realiza la segunda semi-operación en la actualización de las variables espectrales reales.



Una vez ejecutadas las mariposas de la última etapa, quedarán almacenadas en las 512 primeras posiciones de la memoria principal los 512 coeficientes reales de la Transformada Rápida de Fourier. En las siguientes 512 posiciones de la memoria principal quedarán guardados los 512 coeficientes imaginarios de la Transformada Rápida de Fourier.



Para realizar la Transformada Rápida de Fourier me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "010" se activará toda la lógica encargada de calcular los coeficientes espectrales de Fourier y de almacenarlos en la memoria principal para extraer el funcionamiento particular del módulo de la lógica común de la unidad. En primer lugar se establecen los límites en los bucles de índice, de fase y de operación para recorrer el algoritmo de almacenamiento de los valores del coseno.

Se recorrerá en primer lugar el bucle de fase, donde cada fase se corresponde con cada una de las etapas del algoritmo FFT (0-8). En cada una de las etapas se deben realizar los cálculos de 256 mariposas, para eso, dentro de cada etapa se recorre el contador de índice desde 0 a 255. Para cada uno de los valores de índice se recorre el bucle de operaciones desde el valor 0 al valor 20. Así se ejecutan las 20 operaciones que conforman el cálculo de cada mariposa. Resumiendo: Se recorren 9 etapas, en cada etapa 256 mariposas, y en cada mariposa se realizan 20 operaciones. Una serie de flags serán útiles en los cálculos de las operaciones. El siguiente esquema muestra el valor de los flags.

ACTIVACION_Q

NUMERO_OPERACION="0000"	
NUMERO_OPERACION="0001"	Q
NUMERO_OPERACION="0010"	P
NUMERO_OPERACION="0011"	P
NUMERO_OPERACION="0100"	
NUMERO_OPERACION="0101"	Q
NUMERO_OPERACION="0110"	P
NUMERO_OPERACION="0111"	P
NUMERO_OPERACION="1000"	
NUMERO_OPERACION="1001"	Q
NUMERO_OPERACION="1010"	Q
NUMERO_OPERACION="1011"	P
NUMERO_OPERACION="1100"	
NUMERO_OPERACION="1101"	Q
NUMERO_OPERACION="1110"	Q
NUMERO_OPERACION="1111"	P

FLAG_PQ_IMAGINARIO

NUMERO_OPERACION="0000"	
NUMERO_OPERACION="0001"	real
NUMERO_OPERACION="0010"	real
NUMERO_OPERACION="0011"	real
NUMERO_OPERACION="0100"	
NUMERO_OPERACION="0101"	imag
NUMERO_OPERACION="0110"	imag
NUMERO_OPERACION="0111"	imag
NUMERO_OPERACION="1000"	
NUMERO_OPERACION="1001"	real
NUMERO_OPERACION="1010"	imag
NUMERO_OPERACION="1011"	imag
NUMERO_OPERACION="1100"	
NUMERO_OPERACION="1101"	imag
NUMERO_OPERACION="1110"	real
NUMERO_OPERACION="1111"	real

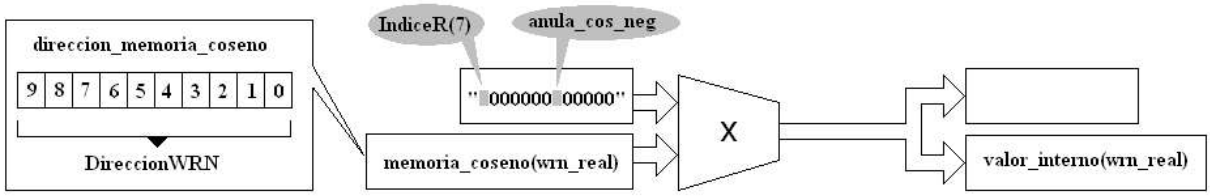
'0' lógico
'1' lógico

FLAG_WRN_IMAGINARIO

NUMERO_OPERACION="0000"	real	NUMERO_OPERACION="1000"	imag
NUMERO_OPERACION="0001"	real	NUMERO_OPERACION="1001"	imag
NUMERO_OPERACION="0010"	real	NUMERO_OPERACION="1010"	imag
NUMERO_OPERACION="0011"	real	NUMERO_OPERACION="1011"	imag
NUMERO_OPERACION="0100"	real	NUMERO_OPERACION="1100"	imag
NUMERO_OPERACION="0101"	real	NUMERO_OPERACION="1101"	imag
NUMERO_OPERACION="0110"	real	NUMERO_OPERACION="1110"	imag
NUMERO_OPERACION="0111"	real	NUMERO_OPERACION="1111"	imag

NÚMERO DE OPERACIÓN "00000":
Se almacena el valor cos(2 π r / 512) en la variable "wrn_real". El valor del índice r depende de la etapa en uso y del número de mariposa que se está calculando. El componente "obten_r" proporciona el valor del índice. El valor del coseno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del índice r tiene el valor '1' lógico; en este caso, el componente "calcula_direccion_wrn" activa la señal "AnulaWRN" con un valor '1' lógico.

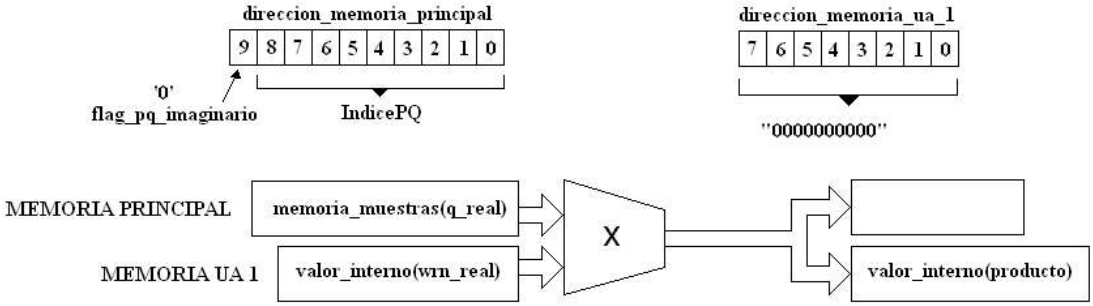
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 ("0000000000000" en formato flotante) si se activa la señal "AnulaWRN" con un valor '1' lógico. En caso contrario, proporcionará el valor 1 ("0000000100000" en formato flotante) si el bit más significativo del índice r es '0' lógico o un -1 ("1000000100000" en formato flotante) si el bit más significativo del índice r es '1' lógico.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la memoria donde se guardan los valores del coseno. La dirección de lectura de la memoria "memoria_coseno" la proporciona el componente "calcula_direccion_wrn" en función de la etapa y del número de mariposa actual.
- El resultado de la unidad aritmética se guardará en la dirección "0000000" de la memoria "memoria_ua_1", donde se almacenará la variable "wrn_real" para su posterior uso.



NÚMERO DE OPERACIÓN "00001":
Se almacena el producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q en la variable "producto". El valor del índice q es calculado por el componente "obten_pq" y representa el índice de la segunda variable espectral de la mariposa.

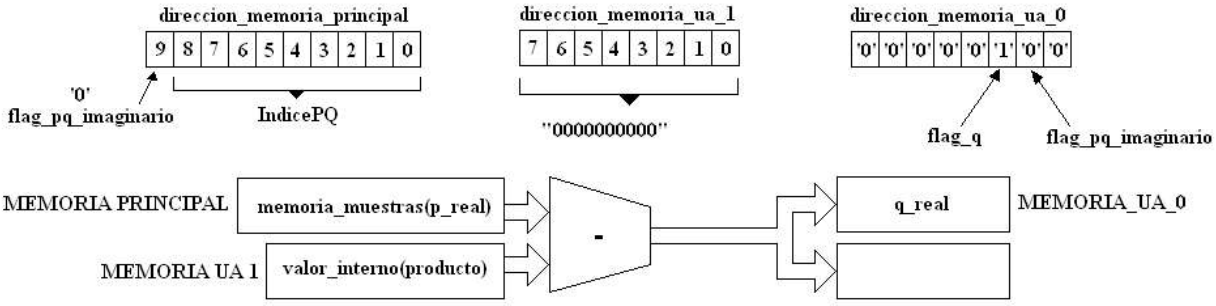
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte real de la segunda variable espectral de la mariposa, indexada por el índice q.

- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "wrn_real".
- El resultado de la unidad aritmética, producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q, se guardará en la dirección "00000000" de la memoria "memoria_ua_1", donde se almacenará la variable "producto" para su posterior uso.



NÚMERO DE OPERACIÓN "00010":
Se realiza la primera semi-operación de actualización de la variable espectral real correspondiente al índice q. Esta semioperación consiste en restar el producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral real correspondiente al índice p. Se almacena en la variable "q_real".

- La operación realizada por la unidad aritmética es una resta.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte real de la primera variable espectral de la mariposa, indexada por el índice p.

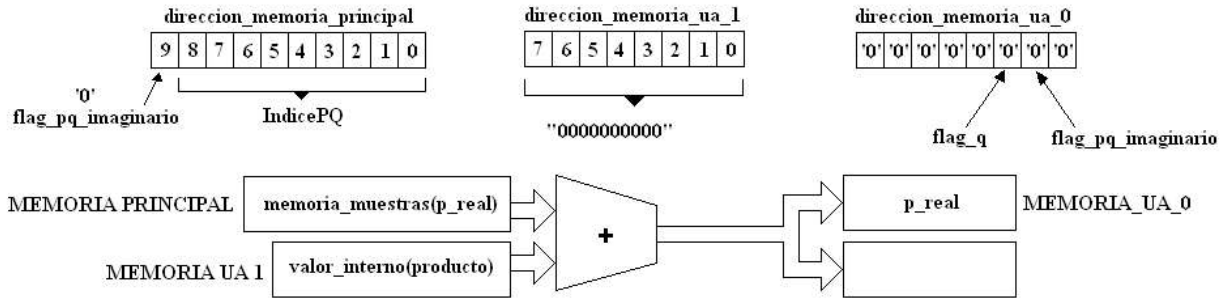


- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".

- El resultado de la unidad aritmética, primera semi-operación de actualización de la variable espectral real correspondiente al índice q, se guardará en la dirección "00000100" de la memoria "memoria_ua_0", donde se almacenará la variable "q_real" con el valor de la primer sumando del nuevo valor de la variable espectral real correspondiente al índice q.

NÚMERO DE OPERACIÓN "00011":
Se realiza la primera semi-operación de actualización de la variable espectral real correspondiente al índice p. Esta semioperación consiste en sumar el producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral real correspondiente al índice p. Se almacena en la variable "p_real".

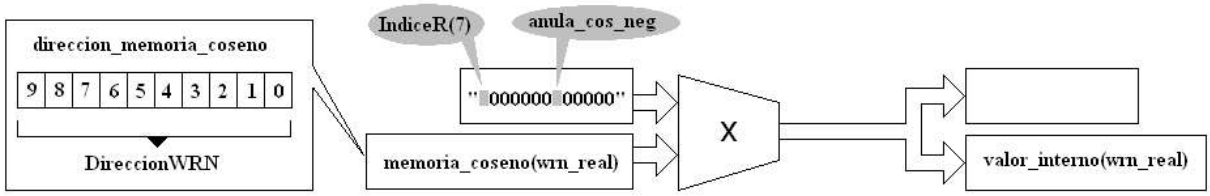
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte real de la primera variable espectral de la mariposa, indexada por el índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, primera semi-operación de actualización de la variable espectral real correspondiente al índice p, se guardará en la dirección "00000000" de la memoria "memoria_ua_0", donde se almacenará la variable "p_real" con el valor de la primer sumando del nuevo valor de la variable espectral real correspondiente al índice p.



NÚMERO DE OPERACIÓN "00100":
Se almacena el valor cos(2 π r / 512) en la variable "wrn_real". El valor del índice r depende de la etapa en uso y del número de mariposa que se está calculando. El componente "obten_r" proporciona el valor del índice. El valor del coseno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del

índice r tiene el valor '1' lógico; en este caso, el componente "calcula_direccion_wrn" activa la señal "AnulaWRN" con un valor '1' lógico.

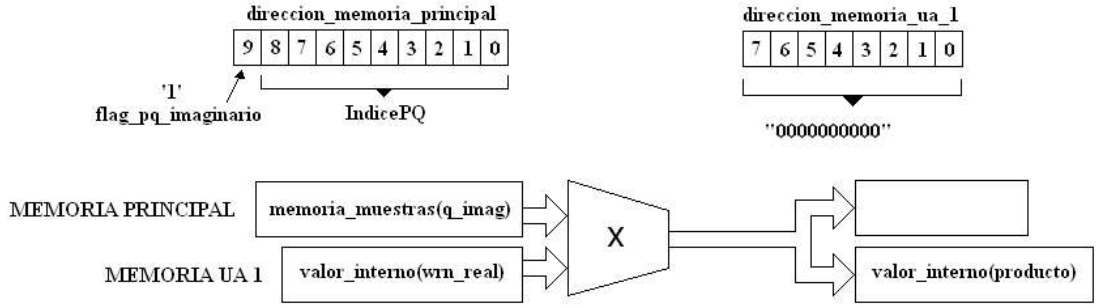
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 ("0000000000000" en formato flotante) si se activa la señal "AnulaWRN" con un valor '1' lógico. En caso contrario, proporcionará el valor 1 ("0000000100000" en formato flotante) si el bit más significativo del índice r es '0' lógico o un -1 ("1000000100000" en formato flotante) si el bit más significativo del índice r es '1' lógico.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la memoria donde se guardan los valores del coseno. La dirección de lectura de la memoria "memoria_coseno" la proporciona el componente "calcula_direccion_wrn" en función de la etapa y del número de mariposa actual.
- El resultado de la unidad aritmética se guardará en la dirección "0000000" de la memoria "memoria_ua_1", donde se almacenará la variable "wrn_real" para su posterior uso.



NÚMERO DE OPERACIÓN "00101":
Se almacena el producto del cos(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q en la variable "producto". El valor del índice q es calculado por el componente "obten_pq" y representa el índice de la segunda variable espectral de la mariposa.

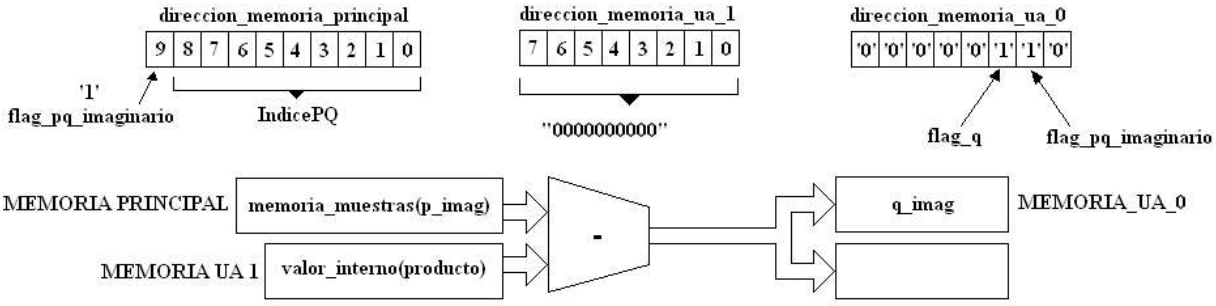
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte imaginaria de la segunda variable espectral de la mariposa, indexada por el índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "wrn_real".

- El resultado de la unidad aritmética, producto del cos(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q, se guardará en la dirección "00000000" de la memoria "memoria_ua_1", donde se almacenará la variable "producto" para su posterior uso.



NÚMERO DE OPERACIÓN "00110":
Se realiza la primera semi-operación de actualización de la variable espectral imaginaria correspondiente al índice q. Esta semioperación consiste en restar el producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral imaginaria correspondiente al índice p. Se almacena en la variable "q_imag".

- La operación realizada por la unidad aritmética es una resta.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte imaginaria de la primera variable espectral de la mariposa, indexada por el índice p.

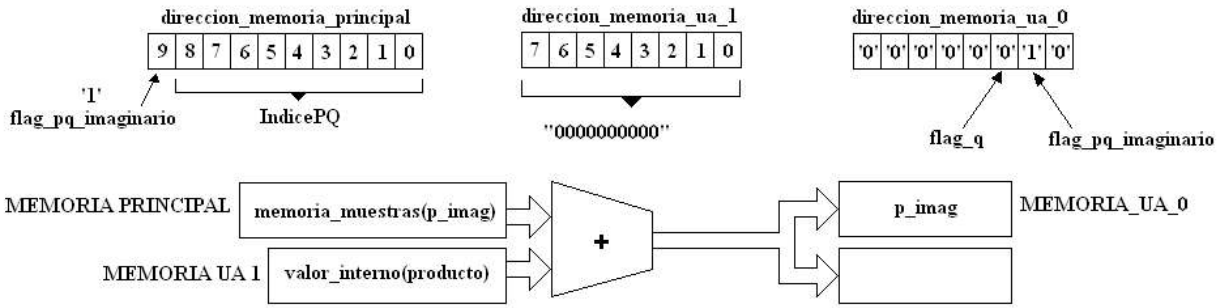


- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, primera semi-operación de actualización de la variable espectral imaginaria correspondiente al índice q, se guardará en la dirección "00000110" de la

memoria "memoria_ua_0", donde se almacenará la variable "q_imag" con el valor del primer sumando del nuevo valor de la variable espectral imaginaria correspondiente al índice q.

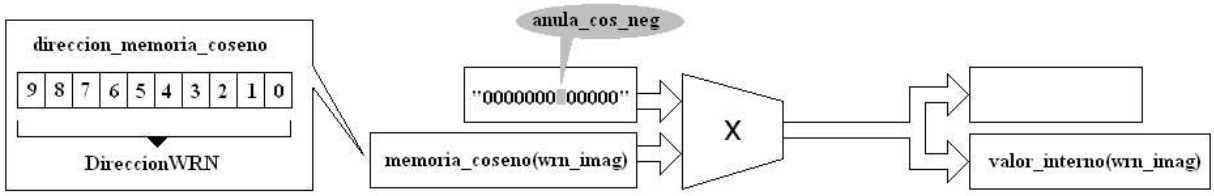
NÚMERO DE OPERACIÓN "00111":
Se realiza la primera semi-operación de actualización de la variable espectral imaginaria correspondiente al índice p. Esta semioperación consiste en sumar el producto del cos(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral imaginaria correspondiente al índice p. Se almacena en la variable "p_imag".

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte imaginaria de la primera variable espectral de la mariposa, indexada por el índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, primera semi-operación de actualización de la variable espectral imaginaria correspondiente al índice p, se guardará en la dirección "00000010" de la memoria "memoria_ua_0", donde se almacenará la variable "p_imag" con el valor de la primer sumando del nuevo valor de la variable espectral imaginaria correspondiente al índice p.



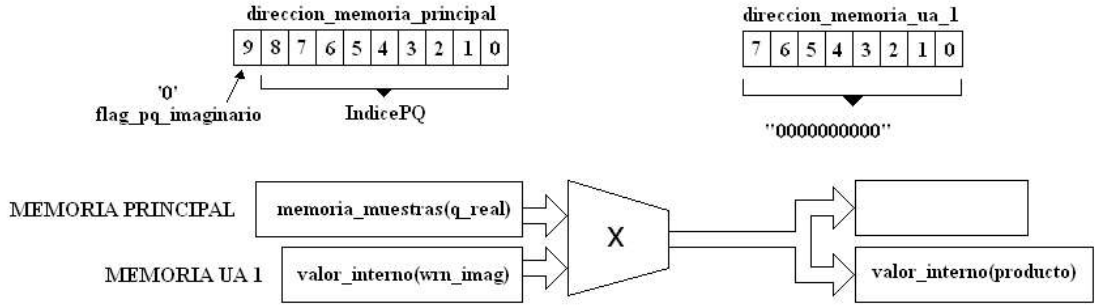
NÚMERO DE OPERACIÓN "01000":
Se almacena el valor sen(2 π r / 512) en la variable "wrn_imag". El valor del índice r depende de la etapa en uso y del número de mariposa que se está calculando. El componente "obten_r" proporciona el valor del índice. El valor del seno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del índice r tiene el valor '0' lógico; en este caso, el componente "calcula_direccion_wrn" activa la señal "AnulaWRN" con un valor '1' lógico.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 (“0000000000000” en formato flotante) si se activa la señal “AnulaWRN” con un valor '1' lógico. En caso contrario, proporcionará el valor 1 (“0000000100000” en formato flotante).
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la memoria donde se guardan los valores del coseno. La dirección de lectura de la memoria “memoria_coseno” la proporciona el componente “calcula_direccion_wrn” en función de la etapa y del número de mariposa actual.
- El resultado de la unidad aritmética se guardará en la dirección “0000000” de la memoria “memoria_ua_1”, donde se almacenará la variable “wrn_imag” para su posterior uso.



NÚMERO DE OPERACIÓN “01001”:
Se almacena el producto del sen(2 π r / 512) por la variable espectral real correspondiente al índice q en la variable “producto”. El valor del índice q es calculado por el componente “obten_pq” y representa el índice de la segunda variable espectral de la mariposa.

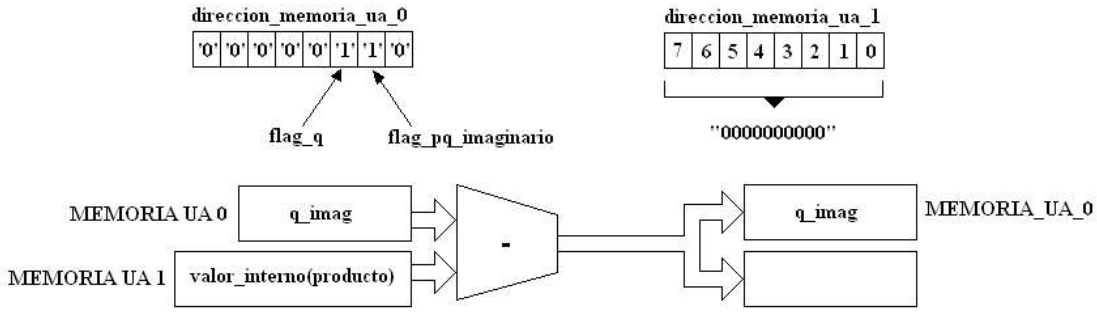
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente “obten_pq”. De esta forma se accede desde la memoria principal a la parte real de la segunda variable espectral de la mariposa, indexada por el índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “memoria_ua_1”, donde está almacenada la variable “wrn_imag”.
- El resultado de la unidad aritmética, producto del sen(2 π r / 512) por la variable espectral real correspondiente al índice q, se guardará en la dirección “00000000” de la memoria “memoria_ua_1”, donde se almacenará la variable “producto” para su posterior uso.



NÚMERO DE OPERACIÓN "01010":

Se realiza la segunda semi-operación de actualización de la variable espectral imaginaria correspondiente al índice q. Esta semioperación consiste en restar el producto del sen(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral imaginaria correspondiente al índice q. Se almacena en la variable "q_imag".

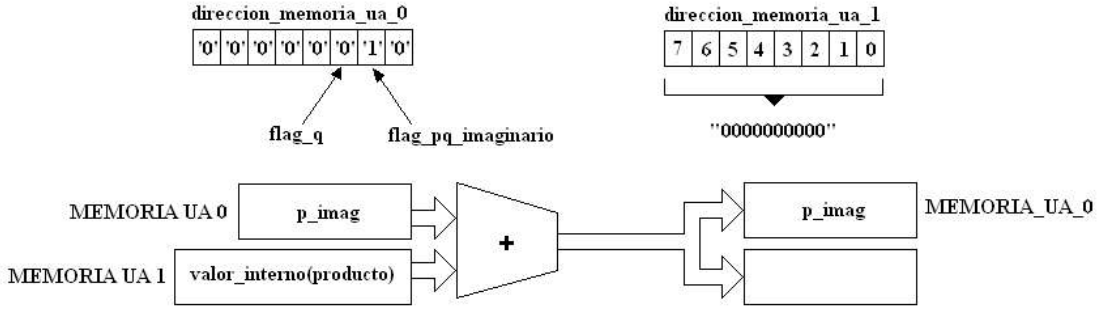
- La operación realizada por la unidad aritmética es una resta.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000110" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "q_imag" con el valor del primer sumando del nuevo valor de la variable espectral imaginaria correspondiente al índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, resultado de la segunda semi-operación de actualización de la variable espectral real correspondiente al índice q, se guardará en la dirección "00000110" de la memoria "memoria_ua_0", donde se almacenará la variable "q_imag" con el nuevo valor que tomará la variable espectral imaginaria correspondiente al índice q (segunda de las variables espectrales de la mariposa).



NÚMERO DE OPERACIÓN "01011":

Se realiza la segunda semi-operación de actualización de la variable espectral imaginaria correspondiente al índice p. Esta semioperación consiste en sumar el producto del sen(2 π r / 512) por la variable espectral real correspondiente al índice q (almacenado en la variable "producto") a la variable espectral imaginaria correspondiente al índice p. Se almacena en la variable "p_imag".

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000010" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "p_imag" con el valor de la primer sumando del nuevo valor de la variable espectral imaginaria correspondiente al índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, resultado de la segunda semi-operación de actualización de la variable espectral imaginaria correspondiente al índice p, se guardará en la dirección "00000010" de la memoria "memoria_ua_0", donde se almacenará la variable "p_imag" con el nuevo valor que tomará la variable espectral imaginaria correspondiente al índice p (primera de las variables espectrales de la mariposa).

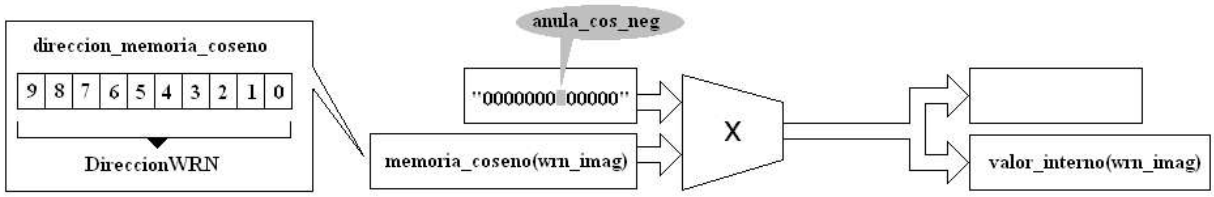


NÚMERO DE OPERACIÓN "01100":
Se almacena el valor sen(2 π r / 512) en la variable "wrn_imag". El valor del índice r depende de la etapa en uso y del número de mariposa que se está calculando. El componente "obten_r" proporciona el valor del índice. El valor del seno se anula si los 7 bits menos significativos del índice r tienen el valor '0' lógico y el bit más significativo del índice r tiene el valor '0' lógico; en este caso, el componente "calcula_direccion_wrn" activa la señal "AnulaWRN" con un valor '1' lógico.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará un valor 0 ("0000000000000" en formato flotante) si se activa

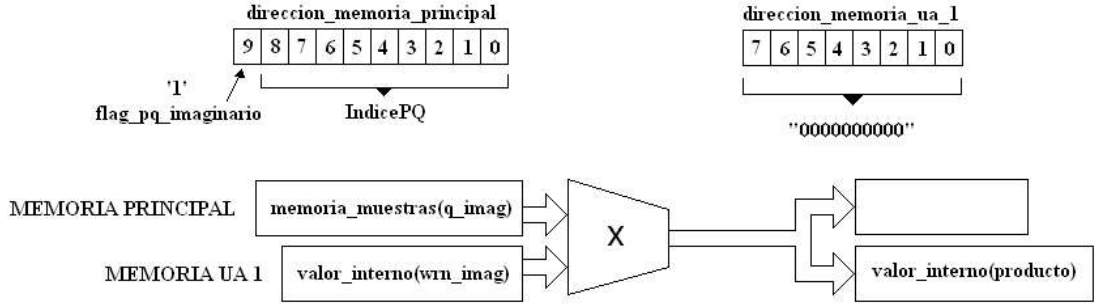
la señal "AnulaWRN" con un valor '1' lógico. En caso contrario, proporcionará el valor 1 ("0000000100000" en formato flotante).

- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la memoria donde se guardan los valores del coseno. La dirección de lectura de la memoria "memoria_coseno" la proporciona el componente "calcula_direccion_wrn" en función de la etapa y del número de mariposa actual.
- El resultado de la unidad aritmética se guardará en la dirección "0000000" de la memoria "memoria_ua_1", donde se almacenará la variable "wrn_imag" para su posterior uso.



NÚMERO DE OPERACIÓN "01101":
Se almacena el producto del sen(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q en la variable "producto". El valor del índice q es calculado por el componente "obten_pq" y representa el índice de la segunda variable espectral de la mariposa.

- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente "obten_pq". De esta forma se accede desde la memoria principal a la parte imaginaria de la segunda variable espectral de la mariposa, indexada por el índice q.

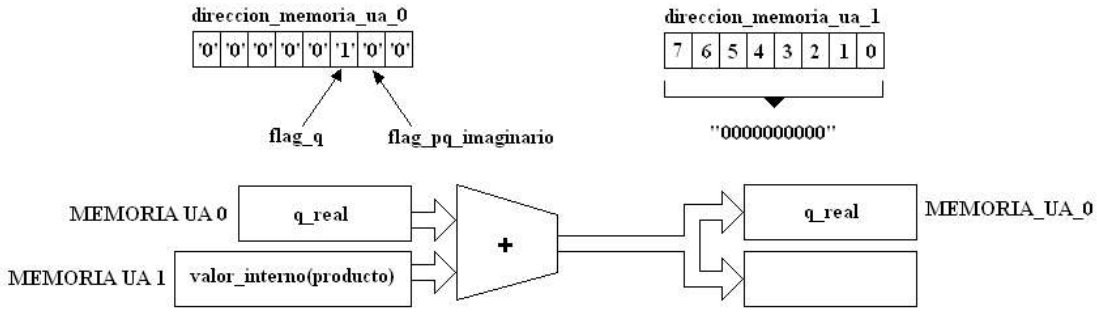


- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "wrn_imag".

- El resultado de la unidad aritmética, producto del sen(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q, se guardará en la dirección "00000000" de la memoria "memoria_ua_1", donde se almacenará la variable "producto" para su posterior uso.

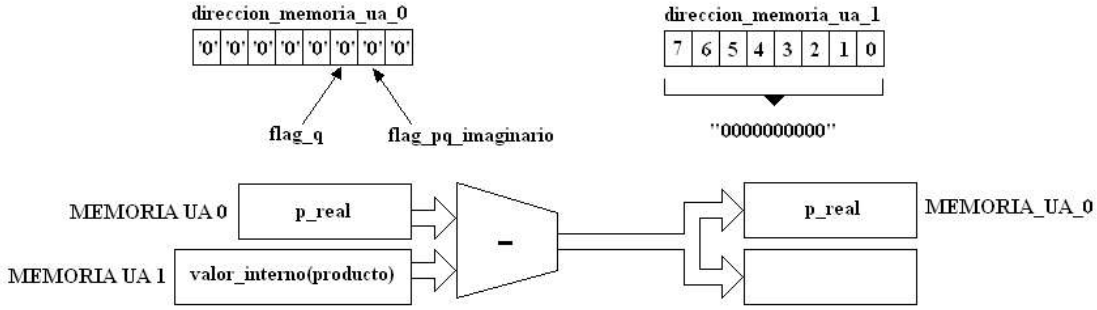
NÚMERO DE OPERACIÓN "01110":
Se realiza la segunda semi-operación de actualización de la variable espectral real correspondiente al índice q. Esta semioperación consiste en sumar el producto del sen(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q (almacenado en la variable "producto") a la variable espectral real correspondiente al índice q. Se almacena en la variable "q_real".

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000100" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "q_real" con el valor del primer sumando del nuevo valor de la variable espectral real correspondiente al índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, resultado de la segunda semi-operación de actualización de la variable espectral real correspondiente al índice q, se guardará en la dirección "00000100" de la memoria "memoria_ua_0", donde se almacenará la variable "q_real" con el nuevo valor que tomará la variable espectral real correspondiente al índice q (segunda de las variables espectrales de la mariposa).



NÚMERO DE OPERACIÓN "01111":
Se realiza la segunda semi-operación de actualización de la variable espectral real correspondiente al índice p. Esta semioperación consiste en sumar el producto del sen(2 π r / 512) por la variable espectral imaginaria correspondiente al índice q (almacenado en la variable "producto") a la variable espectral real correspondiente al índice p. Se almacena en la variable "p_real".

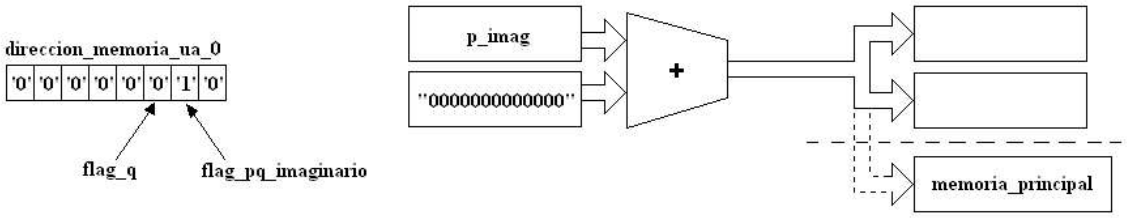
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "p_real" con el valor de la primer sumando del nuevo valor de la variable espectral real correspondiente al índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde está almacenada la variable "producto".
- El resultado de la unidad aritmética, resultado de la segunda semi-operación de actualización de la variable espectral real correspondiente al índice p, se guardará en la dirección "00000000" de la memoria "memoria_ua_0", donde se almacenará la variable "p_real" con el nuevo valor que tomará la variable espectral real correspondiente al índice p (primera de las variables espectrales de la mariposa).



NÚMERO DE OPERACIÓN "10000":

Se almacena en la memoria principal el nuevo valor de la variable espectral imaginaria cuyo índice se corresponde con el índice p (el índice p es el correspondiente a la primera variable espectral en los cálculos de la mariposa).

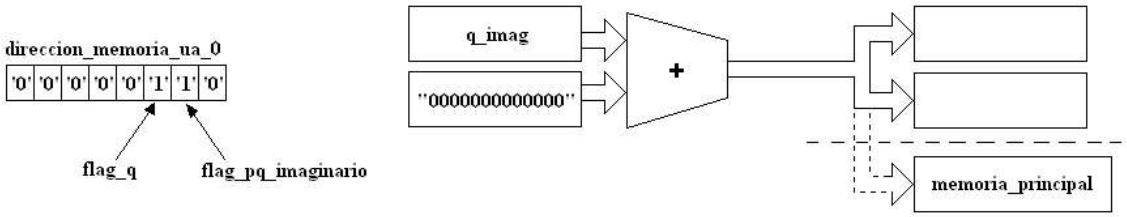
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000010" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "p_imag" con el nuevo valor de la variable espectral imaginaria correspondiente al índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 0 ("0000000000000" en formato flotante) para que no se modifique el valor de la variable espectral tras la suma.
- El resultado de la unidad aritmética, variable espectral imaginaria correspondiente al índice p, se guardará en la memoria principal donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede a la memoria principal para guardar la parte imaginaria de la primera variable espectral de la mariposa, indexada por el índice p.



NÚMERO DE OPERACIÓN "10001":

Se almacena en la memoria principal el nuevo valor de la variable espectral imaginaria cuyo índice se corresponde con el índice q (el índice q es el correspondiente a la segunda variable espectral en los cálculos de la mariposa).

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000110" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "q_imag" con el nuevo valor de la variable espectral imaginaria correspondiente al índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 0 ("0000000000000" en formato flotante) para que no se modifique el valor de la variable espectral tras la suma.
- El resultado de la unidad aritmética, variable espectral imaginaria correspondiente al índice q, se guardará en la memoria principal donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente "obten_pq". De esta forma se accede a la memoria principal para guardar la parte imaginaria de la segunda variable espectral de la mariposa, indexada por el índice q.

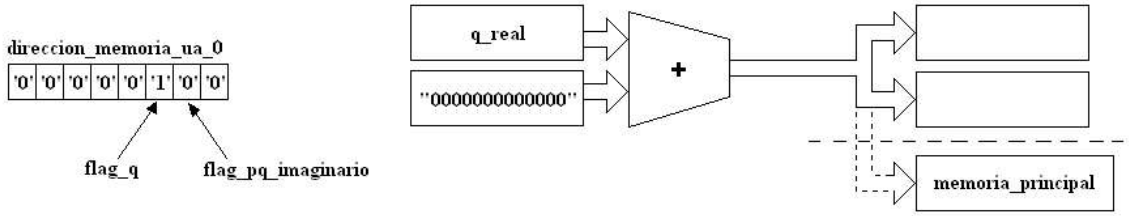


NÚMERO DE OPERACIÓN "10010":

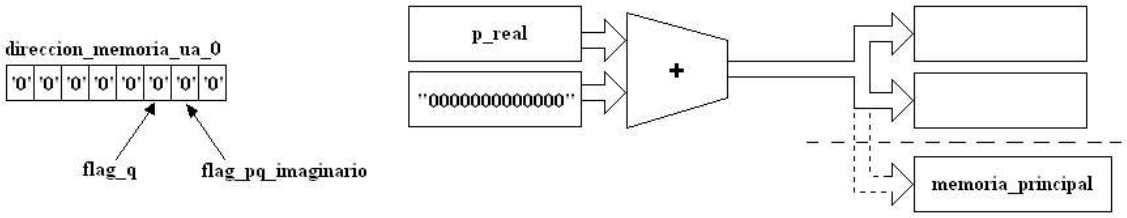
Se almacena en la memoria principal el nuevo valor de la variable espectral real cuyo índice se corresponde con el índice q (el índice q es el correspondiente a la segunda variable espectral en los cálculos de la mariposa).

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000100" de la memoria "memoria_ua_0", donde se encuentra

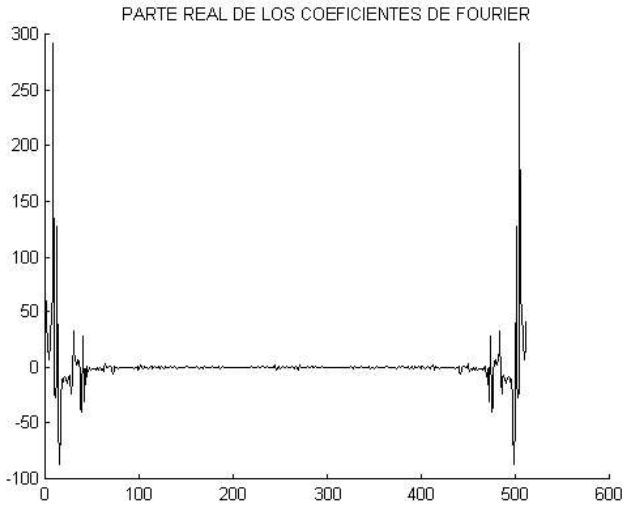
almacenada la variable "q_real" con el nuevo valor de la variable espectral real correspondiente al índice q.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 0 ("0000000000000" en formato flotante) para que no se modifique el valor de la variable espectral tras la suma.
- El resultado de la unidad aritmética, variable espectral real correspondiente al índice q, se guardará en la memoria principal donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice q, calculado por el componente "obten_pq". De esta forma se accede a la memoria principal para guardar la parte real de la segunda variable espectral de la mariposa, indexada por el índice q.

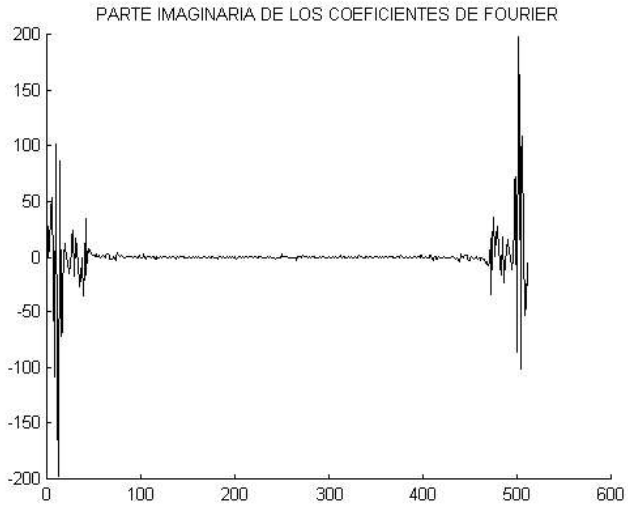


NÚMERO DE OPERACIÓN "10011":
Se almacena en la memoria principal el nuevo valor de la variable espectral real cuyo índice se corresponde con el índice p (el índice p es el correspondiente a la primera variable espectral en los cálculos de la mariposa).
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "p_real" con el nuevo valor de la variable espectral real correspondiente al índice p.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 0 ("0000000000000" en formato flotante) para que no se modifique el valor de la variable espectral tras la suma.
- El resultado de la unidad aritmética, variable espectral real correspondiente al índice p, se guardará en la memoria principal donde se almacenan las variables espectrales. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real de la variable. Los 9 bits restantes conforman el índice p, calculado por el componente "obten_pq". De esta forma se accede a la memoria principal para guardar la parte real de la primera variable espectral de la mariposa, indexada por el índice p.



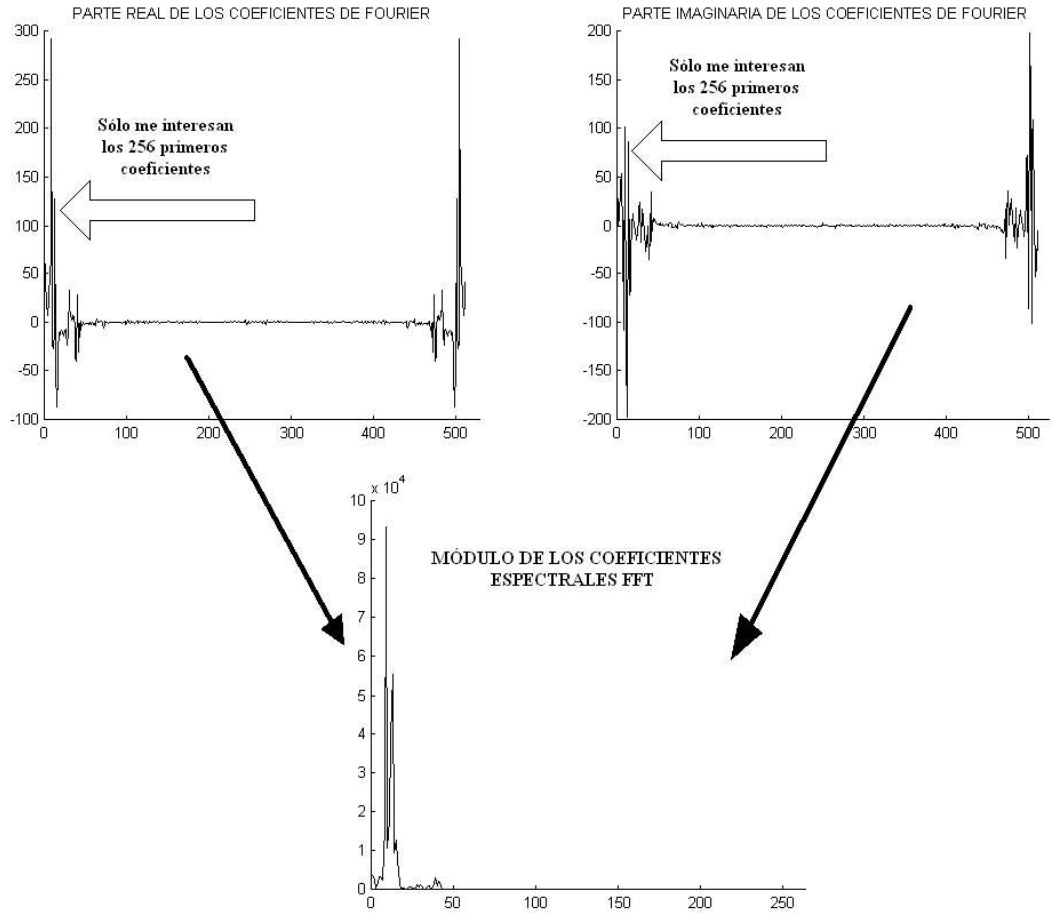
Después de la ejecución todas las operaciones de todas las mariposas en cada una de las etapas de procesamiento del algoritmo de cálculo de la Transformada Rápida de Fourier, se tendrán almacenadas en la memoria principal las 512 variables espectrales y las 512 variables imaginarias correspondientes a los 512 coeficientes de Fourier. Los siguientes esquemas muestran los coeficientes obtenidos a partir de 512 muestras de una señal de audio.





1.1.3.0.4	recnew.vhd Módulo=”011”
Se calcula el módulo al cuadrado de la Transformada Rápida de Fourier de las 512 muestras recibidas desde el conversor analógico-digital de la señal de audio. Se obtiene el módulo al cuadrado de los coeficientes cepstrales obtenidos anteriormente ya que estos coeficientes son complejos (tienen una parte real y una parte imaginaria).	

La información relativa a la fase contenida en los coeficientes espectrales de Fourier no tiene información relevante para el reconocimiento de las vocales. Sólo interesa el módulo de la Transformada Rápida de Fourier. A partir de la parte real e imaginaria de cada uno de los coeficientes de Fourier se obtiene el módulo al cuadrado de cada uno de los coeficientes. No es preciso realizar la raíz cuadrada al módulo ya que en el proceso de obtención de los coeficientes cepstrales el dato que se utiliza es el cuadrado del módulo de cada una de las bandas de frecuencia.

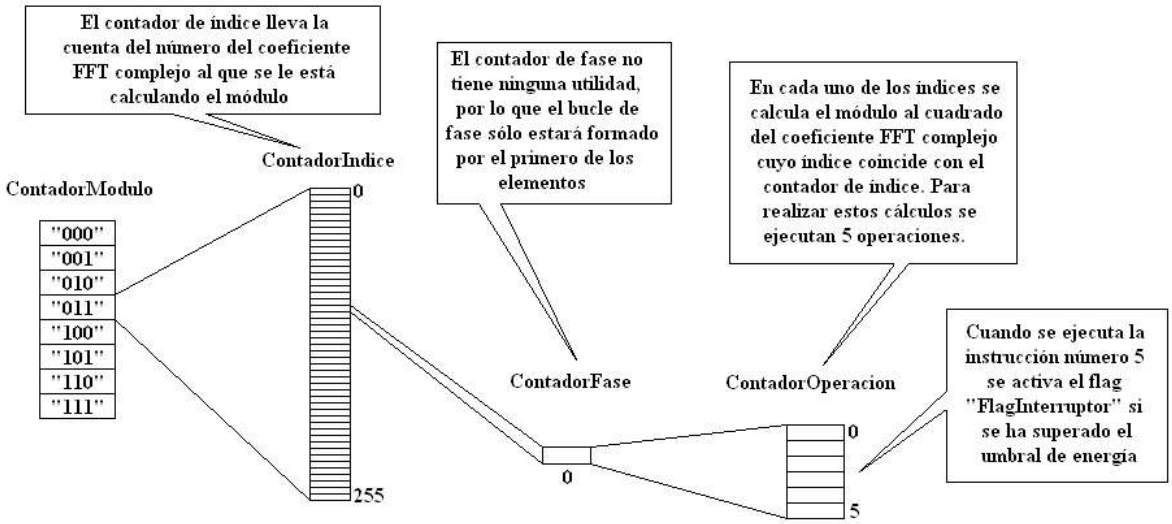


De 512 coeficientes espectrales complejos, sólo me interesan los 256 primeros, ya que los 256 siguientes son iguales a los primeros pero en orden inverso. Si el módulo de alguno de los coeficientes supera el valor 4096, se considera que se está pronunciando una vocal, puesto que la energía de la señal ha superado el umbral. En esta caso, se activa el flag "FlagInterruptor" para que tras el proceso de reconocimiento se genere una señal de activación vocálica. Este flag se reinicia cada vez que se resetea el número de módulo, por lo que si no se supera el umbral, el "FlagInterruptor" permanecerá con el valor '0' lógico, indicando al sistema encargado de reconocer las vocales que no se ha pronunciado ninguna vocal.

Para realizar el proceso implementado durante la ejecución del módulo número "011" del componente "recnew" se ejecutan una serie de operaciones que pueden sintetizarse en el siguiente algoritmo (se utiliza el lenguaje Matlab® para describir el algoritmo):

```
interruptor='0';
for i=0:1:255
  sumando_0=(memoria_FFT_real(i))^2;
  sumando_1=(memoria_FFT_imaginaria(i))^2;
  memoria_modulo(i)=sumando_0 + sumando_1;
  if memoria_modulo(i)>=4096
    interruptor='1';
  end
end
```

Al desarrollar este algoritmo me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "011" se activará toda la lógica encargada de implementar el algoritmo anterior para extraer el funcionamiento particular del módulo de la lógica común de la unidad. En primer lugar se establecen los límites en los bucles de índice, de fase y de operación para recorrer el algoritmo de cálculo del módulo al cuadrado de los coeficientes espectrales complejos.



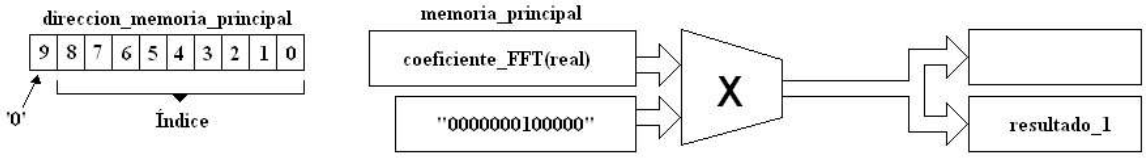
Se recorrerá el bucle de índice, y en cada uno de los índices se calculará el módulo al cuadrado del coeficiente espectral FFT complejo cuyo índice se corresponde con el contador de índice, por eso, el bucle de índice tiene 256 valores. Por cada uno de los índices se calcula el índice de fase, pero como en este caso el contador de fase no tiene ninguna utilidad, el bucle de fase sólo estará formado por una fase, que es necesario recorrerla para reutilizar la lógica del componente "recnew". Para cada uno de los índices se ejecutan 5 operaciones con el fin de hallar el módulo al cuadrado del coeficiente espectral FFT complejo. La parte real de cada uno de los coeficientes espectrales es leída desde la parte baja de la memoria principal (aquella que tiene el bit más significativo de la dirección de memoria con un valor '0' lógico) en la posición marcada por el contador de índice. La parte imaginaria de cada uno de los coeficientes espectrales es leída desde la parte alta de la memoria principal (aquella que tiene el bit más significativo de la dirección de memoria con un valor '1' lógico) en la posición marcada por el contador de índice. El módulo al cuadrado de cada uno de los coeficientes es la suma del cuadrado de la parte real del coeficiente con el cuadrado de la parte imaginaria del coeficiente. El módulo de cada uno de los coeficientes se almacena en la memoria principal en la dirección que se corresponde con el índice del contador de índice.

Cuando se ejecuta la instrucción número 5 se activa el flag "FlagInterruptor" con un valor '1' lógico si se ha superado el umbral de energía en una determinada frecuencia. Se considera que se ha superado el umbral de energía si el módulo de uno de los coeficientes es superior al número 4096. En este caso se considera que se ha pronunciado una vocal y el flag "FlagInterruptor" pondrá en marcha los mecanismos de reconocimiento que activarán la señal vocálica correspondiente con la vocal detectada. Si no se activa el flag, se considerará que no se ha pronunciado ninguna vocal, por lo que no es necesario el proceso de reconocimiento. El flag se desactiva cuando se resetea el contador de módulo, una vez terminado el proceso de reconocimiento y de haberse activado la señal vocálica correspondiente en el caso de que el flag tuviera un valor '1' lógico. Son necesarias 5 operaciones para el cálculo del módulo y una operación para activar el flag, por lo que el bucle de operaciones estará formado por 6 operaciones (de "0000" a "0110").

NÚMERO DE OPERACIÓN "0000":
Lectura de la parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. Se almacena en la variable "resultado_1" para su posterior uso.

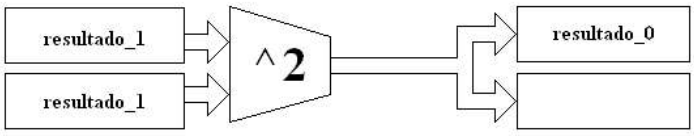
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan los coeficientes espectrales de Fourier. La dirección de memoria tiene el bit más significativo con un valor '0' lógico, correspondiéndose con el acceso a la parte real del coeficiente. Los 9 bits restantes se corresponden con el índice actual. De esta forma se accede desde la memoria principal al coeficiente real de la Transformada Rápida de Fourier cuyo índice se corresponde con el contador de índice.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.

- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará un valor 1 (“0000000100000” en formato flotante), para que al multiplicar la entrada primera de la unidad aritmética no se produzca ninguna modificación del valor.
- El resultado de la unidad aritmética, parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice, se almacenará en la dirección “00000000” de la memoria “memoria_ua_1”. En esta posición se almacena la variable “resultado_1”.



NÚMERO DE OPERACIÓN “0001”:
Se calcula el cuadrado de la parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. El resultado se almacena en la variable “resultado_0” para su posterior uso.

- La operación realizada por la unidad aritmética es una multiplicación (cuadrado).
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la segunda de las entradas de la unidad aritmética.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección “00000000” de la memoria “memoria_ua_1”, donde se encuentra almacenada la parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice en la variable “resultado_1”.

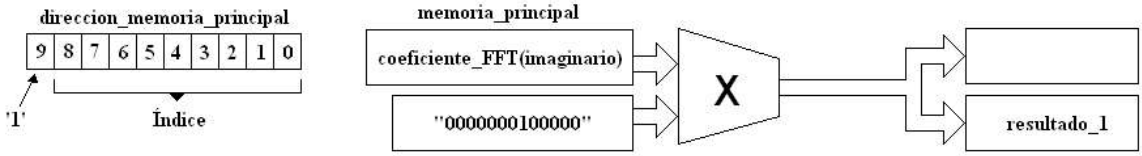


- El resultado de la unidad aritmética, cuadrado de la parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice, se almacena en la dirección “00000000” de la memoria “memoria_ua_0”. Este es el lugar de almacenamiento de la variable “resultado_0”.

NÚMERO DE OPERACIÓN “0010”:
Lectura de la parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. Se almacena en la variable “resultado_1” para su posterior uso.

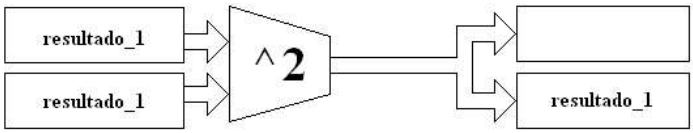
- La operación realizada por la unidad aritmética es una multiplicación.

- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la memoria principal, donde se almacenan los coeficientes espectrales de Fourier. La dirección de memoria tiene el bit más significativo con un valor '1' lógico, correspondiéndose con el acceso a la parte imaginaria del coeficiente. Los 9 bits restantes se corresponden con el índice actual. De esta forma se accede desde la memoria principal al coeficiente imaginario de la Transformada Rápida de Fourier cuyo índice se corresponde con el contador de índice.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará un valor 1 ("0000000100000" en formato flotante), para que al multiplicar la entrada primera de la unidad aritmética no se produzca ninguna modificación del valor.
- El resultado de la unidad aritmética, parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice, se almacenará en la dirección "00000000" de la memoria "memoria_ua_1". En esta posición se almacena la variable "resultado_1".



NÚMERO DE OPERACIÓN "0011":
Se calcula el cuadrado de la parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. El resultado se almacena en la variable "resultado_1" para su posterior uso.

- La operación realizada por la unidad aritmética es una multiplicación (cuadrado).
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la segunda de las entradas de la unidad aritmética.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde se encuentra almacenada la parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice en la variable "resultado_1".



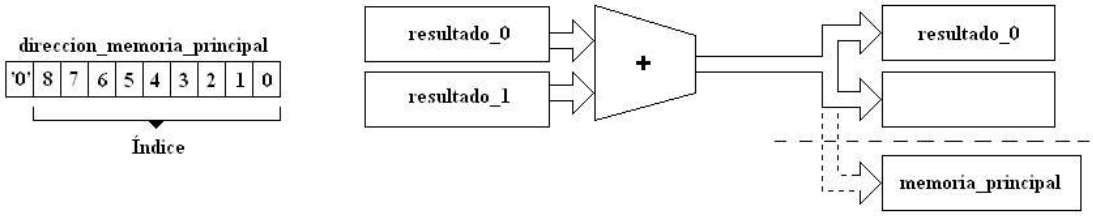
- El resultado de la unidad aritmética, cuadrado de la parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice, se almacena en la dirección

"00000000" de la memoria "memoria_ua_1". Este es el lugar de almacenamiento de la variable "resultado_1".

NÚMERO DE OPERACIÓN "0100":

Se calcula el cuadrado del módulo del coeficiente espectral de Fourier cuyo índice se corresponde con el contador de índice. Para ello se suma el cuadrado de la parte real del coeficiente con la parte imaginaria del mismo. El resultado se almacena en la dirección de la memoria principal que se corresponde con el contador de índice.

- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el cuadrado de la parte real del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. Esta posición de memoria es la variable "resultado_0".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_1", donde se encuentra almacenado el cuadrado de la parte imaginaria del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. Esta posición de memoria es la variable "resultado_1".
- El resultado de la unidad aritmética, módulo del coeficiente espectral de Fourier cuyo índice se corresponde con el contador de índice, se almacena en la dirección "00000000" de la memoria "memoria_ua_0". Este es el lugar de almacenamiento de la variable "resultado_0". También se almacena en la dirección de la memoria principal que se corresponde con el contador de índice para su posterior uso en otro módulo.



NÚMERO DE OPERACIÓN "0101":

Se activa el flag "FlagInterruptor" con un valor '1' lógico si se ha superado el umbral de energía en una determinada frecuencia.

- La operación realizada por la unidad aritmética es una resta.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el cuadrado del módulo del coeficiente de Fourier complejo cuyo índice se corresponde con el contador de índice. Esta posición de memoria es la variable "resultado_0".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará el umbral de energía a partir del cual se activa el flag "FlagInstruccion" ("0100000000110" en formato flotante 4096 en formato decimal).

- El bit más significativo del resultado de la unidad aritmética toma el valor '0' si el coeficiente espectral actual ha superado el umbral de energía. En este caso se considera que se ha pronunciado una vocal y se activa el flag "FlagInterruptor" que pondrá en marcha los mecanismos de reconocimiento que activarán la señal vocálica correspondiente con la vocal detectada.