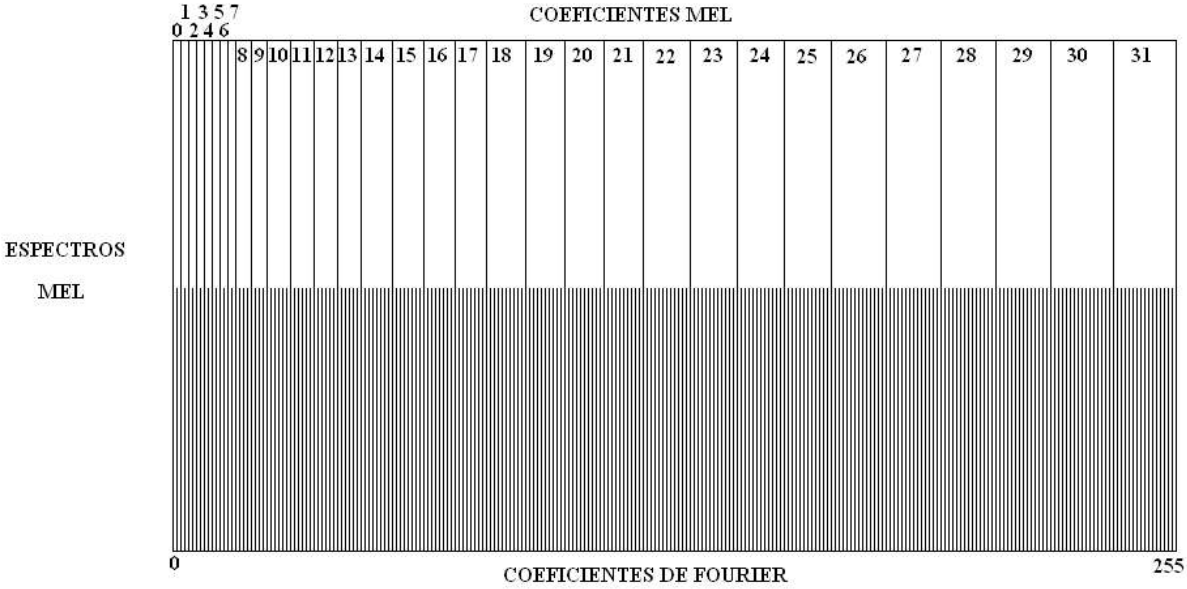
1.1.3.0.5	recnew.vhd Módulo=”100”
En este módulo se calculan los coeficientes mel de las muestras de la señal de audio procedentes del conversor analógico-digital. Los coeficientes mel se obtienen a partir del cuadrado del módulo de los coeficientes espectrales de Fourier.	

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 32 bandas, cada una de ellas utiliza la aproximación a la FFT de los siguientes puntos:

Banda 1: FFT del índice 1 al 2
Banda 2: FFT del índice 3 al 4
Banda 3: FFT del índice 5 al 6
Banda 4: FFT del índice 7 al 8
Banda 5: FFT del índice 9 al 10
Banda 6: FFT del índice 11 al 12
Banda 7: FFT del índice 13 al 14
Banda 8: FFT del índice 15 al 16
Banda 9: FFT del índice 17 al 20
Banda 10: FFT del índice 21 al 24
Banda 11: FFT del índice 25 al 30
Banda 12: FFT del índice 31 al 36
Banda 13: FFT del índice 37 al 42
Banda 14: FFT del índice 43 al 48
Banda 15: FFT del índice 49 al 56
Banda 16: FFT del índice 57 al 64
Banda 17: FFT del índice 65 al 72
Banda 18: FFT del índice 73 al 80
Banda 19: FFT del índice 81 al 90
Banda 20: FFT del índice 91 al 100
Banda 21: FFT del índice 101 al 110
Banda 22: FFT del índice 111 al 120
Banda 23: FFT del índice 121 al 132
Banda 24: FFT del índice 133 al 144
Banda 25: FFT del índice 145 al 156
Banda 26: FFT del índice 157 al 168

Banda 27: FFT del índice 169 al 182
Banda 28: FFT del índice 183 al 196
Banda 29: FFT del índice 197 al 210
Banda 30: FFT del índice 211 al 224
Banda 31: FFT del índice 225 al 240
Banda 32: FFT del índice 241 al 256



Los coeficientes mel representan mejor las características espectrales de la señal vocálica que los coeficientes espectrales de Fourier. Éstos están linealmente espaciados a lo largo de las frecuencias, pero la información frecuencial de las vocales es más importante en las bajas frecuencias. A medida que aumenta la frecuencia disminuye de forma logarítmica la información vocálica. El oído humano funciona de esta forma, dando mayor importancia a las bajas frecuencia y disminuyendo de forma logarítmica la relevancia de la información frecuencial a medida que aumenta la frecuencia.

Los coeficientes mel intentan aproximar este comportamiento, dando la mayor importancia a los coeficientes espectrales de Fourier que caracterizan las bajas frecuencias. A medida que aumenta la frecuencia, la energía vocálica correspondiente a estas frecuencias disminuye, a la vez que la información presente en estas frecuencias. Cada coeficiente mel representa la energía de un espectro de frecuencia (desde un límite inferior a un límite superior de frecuencia). Los ocho primeros espectros quedan representados por el cuadrado del módulo de los 16 primeros coeficientes de Fourier, la energía de cada espectro es la suma del cuadrado del módulo de 2 coeficientes de Fourier. A medida que aumenta la frecuencia se aumenta de forma logarítmica el tamaño de cada espectro. El coeficiente mel que representa la energía del espectro se obtiene mediante la suma de los coeficientes espectrales de Fourier que quedan incluidos dentro de las frecuencias del espectro. De esta forma, se mantiene la información frecuencial de las bajas frecuencias, donde está la mayor parte de la información, y se extrae la información de las altas frecuencias eliminando toda característica que no aporta nada de información.

Para realizar el proceso implementado durante la ejecución del módulo número "100" 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):

```
indice_mel=0;
i=0;
while i<256
  for fase=0:1:( fase_objetivo(i) )
    if fase==0
      suma=0;
    end
    suma=suma + modulo_fourier(i);
    i=i+1;
  end
  mel(indice_mel)=suma;
end

function [salida]=fase_objetivo(i)
  if i < 16
    objetivo=0;
  elseif i < 24
    objetivo=1;
  elseif i < 48
    objetivo=2;
  elseif i < 80
    objetivo=3;
  elseif i < 120
    objetivo=4;
  elseif i < 168
    objetivo=5;
  elseif i < 224
    objetivo=6;
  else
    objetivo=7;
  end
  salida= ( 2*objetivo ) + 1;
```

Al desarrollar este algoritmo me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "100" 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 de los coeficientes mel.

El índice i se corresponde con el contador de índice. El contador de índice se incrementa una vez concluido el bucle de fase. También, y de forma particular en este caso, se incrementa cada vez que se incrementa una fase. El bucle de índice está formado por 256 índices. El contador de operación lleva la cuenta del índice del coeficiente mel actual que se está calculando. No existe un bucle de operaciones propiamente dicho por cada una de las fases a recorrer, ya que la función del contador de operación es la de llevar la cuenta del contador de índice de coeficiente mel. Sólo se ejecutará una operación en la unidad aritmética (una suma) por lo que el número de operación no es necesario a la hora de establecer límites dentro del bucle de operación. Cuando se esté

ejecutando esta unidad el bucle de operaciones estará formado por una sola operación. Independientemente del valor del contador de operación, siempre se realizará la misma operación: la operación de suma del primer sumando (módulo al cuadrado del coeficiente FFT cuyo índice se corresponde con el contador de índice) con el valor 0 (valor inicial de la suma) si el número de la fase es 0, o la operación de suma para actualizar el valor de la suma global añadiendo un nuevo sumando a la suma (módulo al cuadrado del coeficiente FFT cuyo índice se corresponde con el contador de índice) si el número de la fase que se está ejecutando es distinto de 0. El contador de operación lleva la cuenta del índice del coeficiente mel y tras inicializarse con el valor 0 al inicio de la ejecución del módulo "100" (cálculo de los coeficientes mel), sólo se reseteará al final de la ejecución del módulo para iniciar las operaciones del módulo siguiente ("101" hallar el logaritmo en base 2 de los coeficientes mel). Además, el contador de operación se incrementa cada vez que se ha calculado y almacenado un coeficiente mel, hecho que ocurre cuando se incrementa el contador de índice tras concluir un bucle de fase, para así incrementar el índice del coeficiente mel a calcular.

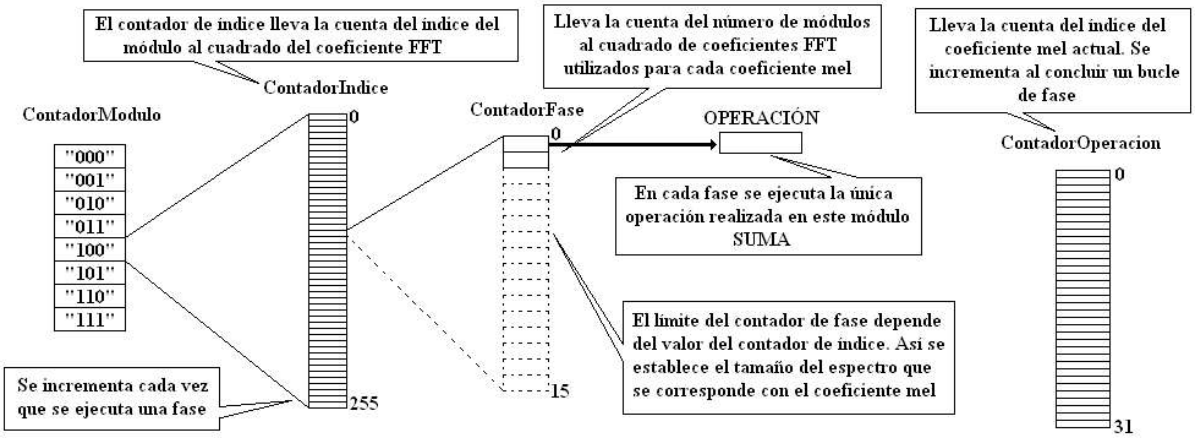
El coeficiente mel es la suma del cuadrado de los módulos de los coeficientes de Fourier que se encuentran en el espectro de frecuencias correspondientes al coeficiente mel en proceso de cálculo. El tamaño del espectro depende del valor del índice del coeficiente mel:

- Suma del módulo al cuadrado de 2 coeficientes FFT para los 8 primeros coeficientes mel.
- Suma del módulo al cuadrado de 4 coeficientes FFT para los 2 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 6 coeficientes FFT para los 4 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 8 coeficientes FFT para los 4 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 10 coeficientes FFT para los 4 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 12 coeficientes FFT para los 4 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 14 coeficientes FFT para los 4 coeficientes mel siguientes.
- Suma del módulo al cuadrado de 16 coeficientes FFT para los 2 coeficientes mel siguientes.

El bucle del contador de fase lleva la cuenta del número de módulos al cuadrado que se sumarán para obtener el coeficiente mel actual. Esta es la función del contador de fase. El límite superior del bucle de fase será el que establezca el tamaño del espectro con el número de módulos al cuadrado de coeficientes FFT que se sumarán. El contador de índice lleva la cuenta del índice del coeficiente FFT cuyo módulo al cuadrado se sumará para obtener el coeficiente mel. El contador de índice se incrementa cada vez que termina un bucle de fase, como es normal en el diseño de la unidad "recnew", pero también se incrementará cada vez que se incremente el índice de fase, para así pasar al siguiente coeficiente FFT. El límite de fase para cumplir los requerimientos anteriores dependerá del contador de índice:

- El bucle de fase estará formado por 2 elementos si el contador de índice está entre 0 y 15.
- El bucle de fase estará formado por 4 elementos si el contador de índice está entre 16 y 23.
- El bucle de fase estará formado por 6 elementos si el contador de índice está entre 24 y 47.
- El bucle de fase estará formado por 8 elementos si el contador de índice está entre 48 y 79.
- El bucle de fase estará formado por 10 elementos si el contador de índice está entre 80 y 119.
- El bucle de fase estará formado por 12 elementos si el contador de índice está entre 120 y 167.
- El bucle de fase estará formado por 14 elementos si el contador de índice está entre 168 y 223.
- El bucle de fase estará formado por 16 elementos si el contador de índice está entre 224 y 255.

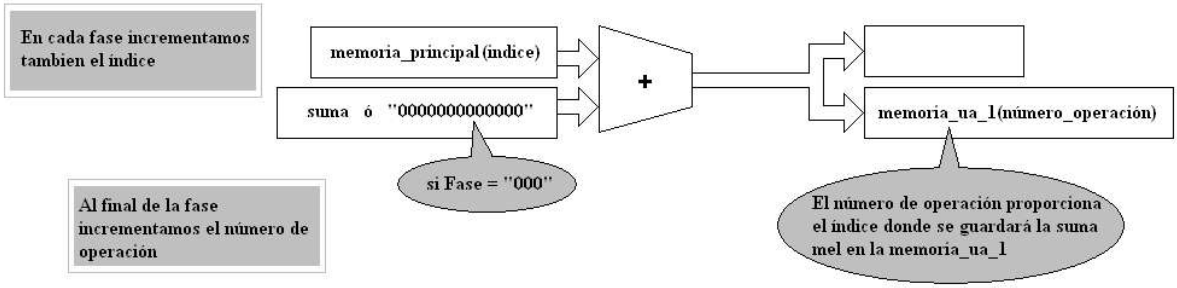
Una vez concluido el bucle de fase quedará almacenado el valor del coeficiente mel en la dirección de la memoria "memoria_ua_1" que tiene el valor binario "1000" en los 4 bits más significativos y el valor de índice del coeficiente calculado en los 4 bits menos significativos.



OPERACIÓN REALIZADA:

En cada bucle de fase se realiza la suma de los cuadrados de los módulos de los coeficientes FFT que conforman el coeficiente mel. En cada fase actualiza la suma con el cuadrado del módulo del coeficiente FFT actual.

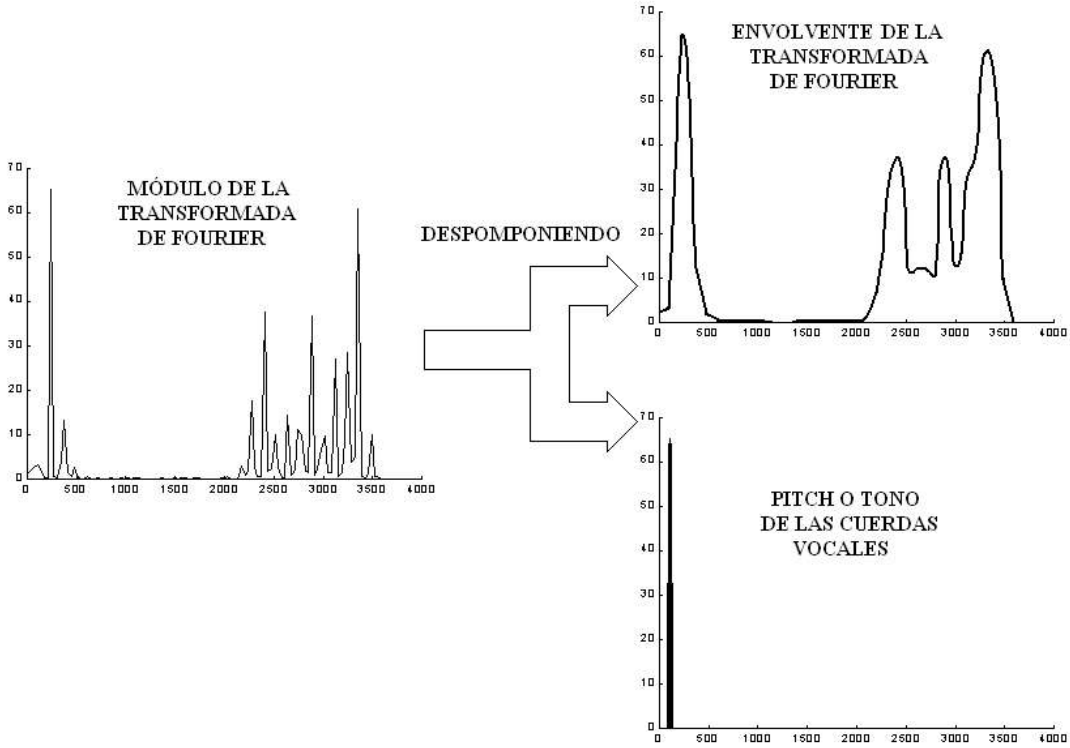
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección que se corresponde con el valor del contador de índice de la memoria principal. En este lugar se encuentra el cuadrado del módulo del coeficiente FFT 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 si el contador de fase es "00000" en binario, o desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "1000" en los 4 bits más significativos y el valor del contador de operación en los 4 bits menos significativos en caso contrario (valor de la suma anterior).
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará un valor 0 ("0000000000000" en formato flotante), como valor inicial de la suma en el caso de que la fase sea "00000" en binario.
- El resultado de la unidad aritmética, valor actual del coeficiente mel, se almacenará en la dirección de la memoria "memoria_ua_1" que tiene el valor binario "1000" en los 4 bits más significativos y el valor del contador de operación en los 4 bits menos significativos.



1.1.3.0.6	recnew.vhd Módulo=”101”
Este módulo calcula el logaritmo en base 2 de los coeficientes mel de las muestras de la señal de audio procedentes del conversor analógico-digital.	

Los coeficientes mel se obtienen a partir del cuadrado del módulo de los coeficientes espectrales de Fourier sintetizando la información frecuencial en espectros de acuerdo con la escala mel logarítmica. Los coeficientes mel dan una idea de la energía de la señal vocálica en las frecuencias de la parcela de espectro con la que se identifican.

El usuario que utilice el emulador del ratón utilizará la pronunciación de las vocales para comunicar los eventos de movimientos del ratón que quiere transmitir al ordenador. Para generar una vocal, las cuerdas vocales vibrarán a una determinada frecuencia (tono). El tracto vocálico se modificará de forma distinta para cada una de las vocales. Como el tracto vocálico es un sistema no lineal, éste aporta sus características a la señal de voz modulando la frecuencia. Si tratamos la señal vocálica desde el punto de vista frecuencial, es en la envolvente de la señal donde reside la información del tracto vocálico, y por consiguiente, la información vocálica.



La frecuencia de las cuerdas vocálicas no aporta información interesante para el proceso de reconocimiento vocálico. Sería interesante extraer sólo la información de la envolvente de la señal en el dominio frecuencial y realizar el proceso de reconocimiento en base a esa información. Cada uno de los coeficientes mel puede verse como el producto de una parte que contiene la información del tono de las cuerdas vocálicas con la parte que contiene la información del tracto vocálico.

$$\text{COEFICIENTE_MEL} = \text{PARTE DEL TONO} \times \text{PARTE DE LA ENVOLVENTE}$$

Realizando el logaritmo en base 2 a cada uno de los coeficientes mel, puedo desacoplar en un sumando la información relativa al tono de las cuerdas vocálicas y en otro sumando la información relativa al tracto vocálico. Como el sumando correspondiente al tono es idéntico en todos los coeficientes mel (se utiliza el mismo tono para las distintas vocales), en la práctica puede considerarse este sumando como una constante cuya presencia no afecta en el proceso de reconocimiento. Realizando el logaritmo en base 2 del coeficiente mel se consigue más eficiencia en la información aportada por el nuevo coeficiente. La ejecución del módulo número "101" del componente "recnew" proporciona el logaritmo en base 2 de los coeficientes mel obtenidos en el módulo anterior.

$$\text{Log}_2(\text{COEFICIENTE_MEL}) = \text{Log}_2(\text{PARTE DEL TONO}) + \text{Log}_2(\text{PARTE DE LA ENVOLVENTE})$$

CONSTANTE

En este sumando reside la información del tracto vocálico

Para realizar el proceso implementado durante la ejecución del módulo número "101" 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):

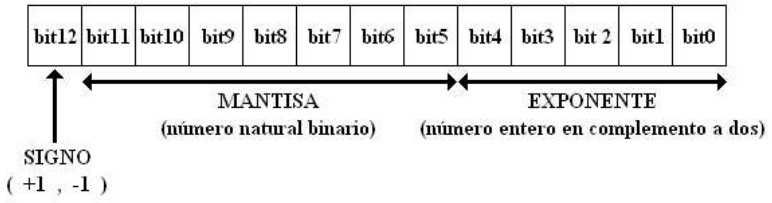
```
for indice=0:1:31
  auxiliar=memoria_mel(indice) ;
  for fase=0:1:15
    [mantisa,exponente]=descompone(auxiliar);
    if fase==0
      suma=exponente;
    end
    auxiliar=mantisa*mantisa;
    [mantisa,exponente]=descompone(auxiliar);
    sumando=exponente/( 2^(fase+1) );
    suma=suma+sumando;
  end
  memoria_log2(indice);
end
```

```
function [mantisa,exponente]=descompone(entrada)
if entrada>=0
  signo=1;
else
  signo=-1;
end
modulo=abs(entrada);
exponente=0;
while modulo>127
  modulo=floor(modulo/2);
  exponente=exponente+1;
end
while modulo<64
  modulo=floor(modulo*2);
  exponente=exponente-1;
end
mantisa=signo*modulo;
```

Para el cálculo del logaritmo en base 2 de los coeficientes mel se utiliza un logaritmo de aproximaciones sucesivas para conseguir el valor. Para ello me aprovecho del formato flotante binario utilizado para representar los datos en el reconocedor. 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.



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

Los coeficientes mel, al ser una medida de la energía de los espectros de frecuencia de la escala mel, son valores positivos. Cada coeficiente mel es el resultado de multiplicar una mantisa positiva por una potencia de 2 cuyo exponente es el valor del exponente del dato flotante.

$$\text{COEFICIENTE_MEL}_i = \text{MANTISA}_i \times 2^{\text{EXPONENTE}_i}$$

El logaritmo de una multiplicación es la suma de los logaritmos de los multiplicandos.

$$\text{Log}_2(\text{COEFICIENTE_MEL}_i) = \text{Log}_2(\text{MANTISA}_i) + \text{Log}_2(2^{\text{EXPONENTE}_i})$$

$$\text{Log}_2(\text{COEFICIENTE_MEL}_i) = \text{Log}_2(\text{MANTISA}_i) + \text{EXPONENTE}_i$$

El valor del logaritmo en base 2 del coeficiente mel lo obtengo mediante una suma donde el primer sumando es el valor del exponente del coeficiente mel en formato flotante. Habría que calcular el logaritmo de la mantisa del coeficiente mel en formato flotante. Para eso elevo al cuadrado la mantisa, y este cuadrado tendrá una representación en formato flotante, donde la mantisa la llamaré A y el exponente B. Utilizando las propiedades del logaritmo, puedo hallar el valor del logaritmo en base 2 de la mantisa del coeficiente mel en formato flotante como suma de dos sumandos. Uno de los sumandos es un logaritmo multiplicado por un factor, y el otro sumando (exponente B dividido por 2) es el segundo sumando de la suma que proporciona el valor del logaritmo en base 2 del coeficiente mel. El segundo sumando de la suma es el exponente del formato flotante del cuadrado de la mantisa del formato flotante del coeficiente mel dividido entre 2.

$$\text{MANTISA}_i^{\,2} = A \times 2^{B}$$

$$\text{Log}_2(\text{MANTISA}_i^{\,2}) = \text{Log}_2(A) + B$$

$$2 \times \text{Log}_2(\text{MANTISA}_i) = \text{Log}_2(A) + B$$

$$\text{Log}_2(\text{MANTISA}_i) = \text{Log}_2(A) \times 2^{-1} + B \times 2^{-1}$$

$$\text{Log}_2(\text{COEFICIENTE_MEL}_i) = \text{Log}_2(A) \times 2^{-1} + \underbrace{B \times 2^{-1}}_{\text{segundo sumando}} + \underbrace{\text{EXPONENTE}_i}_{\text{primer sumando}}$$

Habría que calcular el logaritmo de A. Para eso elevo al cuadrado la mantisa A, y este cuadrado tendrá una representación en formato flotante, donde la mantisa la llamaré C y el exponente D. Utilizando las propiedades del logaritmo, puedo hallar el valor del logaritmo en base 2 de A como suma de dos sumandos. Uno de los sumandos es un logaritmo multiplicado por un factor, y

el otro sumando (exponente D dividido por 2) si lo divido entre 2 es el tercer sumando de la suma que proporciona el valor del logaritmo en base 2 del coeficiente mel.

$$A^2 = C \times 2^D$$

$$\mathrm{Log}_2(A^2) = \mathrm{Log}_2(C) + D$$

$$2 \times \mathrm{Log}_2(A) = \mathrm{Log}_2(C) + D$$

$$\mathrm{Log}_2(A) = \mathrm{Log}_2(C) \times 2^{-1} + D \times 2^{-1}$$

$$\mathrm{Log}_2(\mathrm{COEFICIENTE_MEL}_i) = \mathrm{Log}_2(C) \times 2^{-2} + \boxed{D \times 2^{-2}}_{\text{tercer sumando}}$$

$$+ \boxed{B \times 2^{-1}}_{\text{segundo sumando}} + \boxed{\mathrm{EXPONENTE}_i}_{\text{primer sumando}}$$

Habría que calcular el logaritmo de C. Para eso elevo al cuadrado la mantisa C, y este cuadrado tendrá una representación en formato flotante, donde la mantisa la llamaré E y el exponente F. Utilizando las propiedades del logaritmo, puedo hallar el valor del logaritmo en base 2 de C como suma de dos sumandos. Uno de los sumandos es un logaritmo multiplicado por un factor, y el otro sumando (exponente F dividido por 2) si lo divido entre 4 es el cuarto sumando de la suma que proporciona el valor del logaritmo en base 2 del coeficiente mel.

$$C^2 = E \times 2^F$$

$$\mathrm{Log}_2(C^2) = \mathrm{Log}_2(E) + F$$

$$2 \times \mathrm{Log}_2(C) = \mathrm{Log}_2(E) + F$$

$$\mathrm{Log}_2(C) = \mathrm{Log}_2(E) \times 2^{-1} + F \times 2^{-1}$$

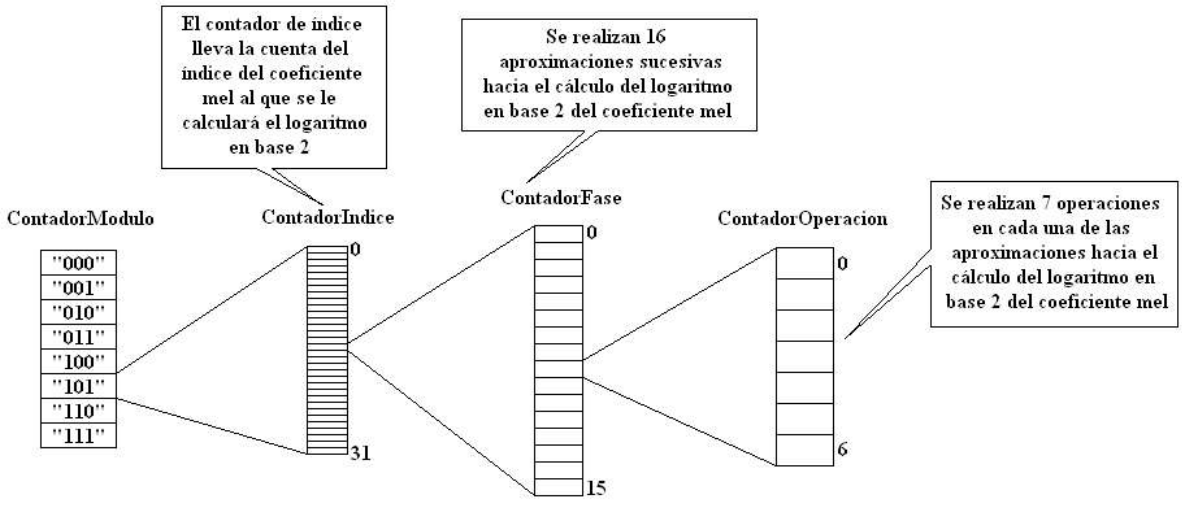
$$\mathrm{Log}_2(\mathrm{COEFICIENTE_MEL}_i) = \mathrm{Log}_2(E) \times 2^{-3} + \boxed{F \times 2^{-3}}_{\text{cuarto sumando}}$$

$$+ \boxed{D \times 2^{-2}}_{\text{tercer sumando}} + \boxed{B \times 2^{-1}}_{\text{segundo sumando}} + \boxed{\mathrm{EXPONENTE}_i}_{\text{primer sumando}}$$

El resto de los sumandos de la suma que proporciona el logaritmo en base 2 del coeficiente mel se obtendrían siguiendo la técnica anterior. Así, inicializando la suma con el valor del exponente del coeficiente mel en formato flotante. El siguiente sumando es el resultado de dividir por 2 el exponente del formato flotante que resulta de elevar al cuadrado la mantisa del formato flotante del coeficiente mel. El siguiente sumando es el resultado de dividir por 4 el exponente del formato flotante que resulta de elevar al cuadrado la mantisa del formato flotante que resulta de elevar al cuadrado la mantisa del formato del coeficiente mel. El siguiente sumando es el resultado de dividir por 4 el exponente del formato flotante que resulta de elevar al cuadrado la mantisa del formato flotante que resulta de elevar al cuadrado la mantisa del flotante que resulta de elevar al cuadrado la mantisa del formato flotante del coeficiente mel. Se continuarán añadiendo sumandos hasta que la aproximación al resultado sea lo suficientemente cercana.

El algoritmo en lenguaje Matlab® expuesto anteriormente sintetiza la técnica utilizada para obtener el logaritmo en base 2 del coeficiente mel. La función "descompone" es añadida para integrar el formato de datos que se utiliza en el reconocedor (formato binario flotante con un bit de signo, 7 bits de módulo y el exponente en base 2 de 5 bits en complemento a dos), ya que la técnica de cálculo del logaritmo se aprovecha de esta representación constituyendo la base del algoritmo de cálculo.

Al desarrollar este algoritmo me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "101" 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 logaritmo de los coeficientes mel.

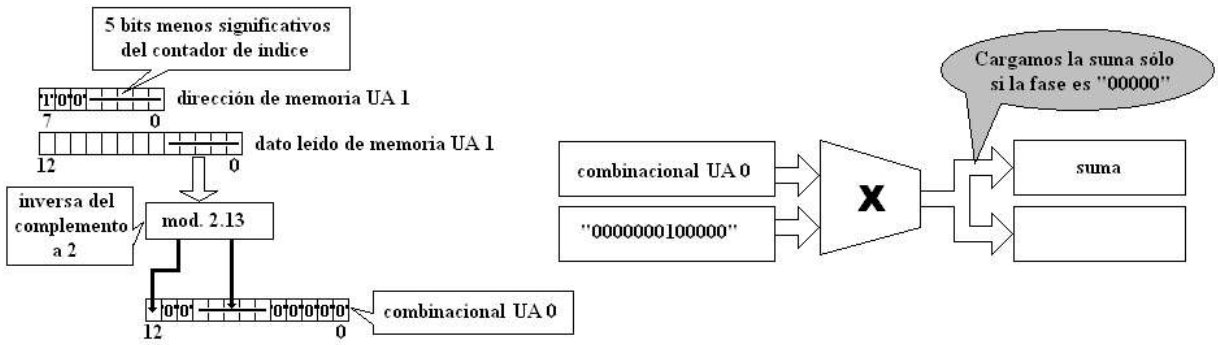


Se recorrerá el bucle de índice, y en cada uno de los índices se calculará el logaritmo en base 2 del coeficiente mel cuyo índice se corresponde con el valor del índice en la dirección de la memoria "memoria_ua_1" que tiene la cadena de bits "100" en los 3 bits más significativos y en el valor del contador de índice en los restantes 5 bits. Dentro de cada índice se realizarán 16 aproximaciones sucesivas en el cálculo del logaritmo, en cada una de ellas se calculará un nuevo sumando de la suma que conformará en la conclusión del bucle de fase el logaritmo en base 2 del

coeficiente mel. El contador de fase lleva la cuenta del número de la aproximación actual en cálculo. Finalmente, para el cálculo de cada sumando se precisan 7 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.

NÚMERO DE OPERACIÓN "0000":
Si la fase es "00000" si inicializa el valor de la variable "suma" (esta variable acumula los sumandos que conformarán el logaritmo en base 2 del coeficiente mel) con el valor del exponente del formato flotante del coeficiente mel cuyo índice coincide con el contador de índice.

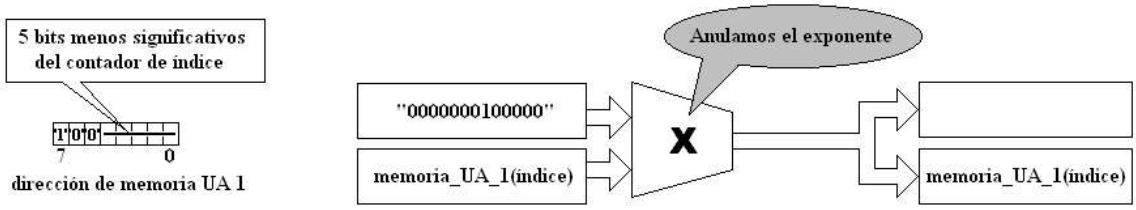
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor del exponente del formato flotante del coeficiente mel cuyo índice coincide con el contador de índice. El coeficiente mel se lee desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos. Los 5 bits menos significativos del dato leído de esta memoria conforman el valor del exponente en complemento a 2. El componente "inv_complemento_a_2" separa el módulo y el signo del exponente. El bit más significativo de la entrada combinacional toma el bit de signo del exponente como el bit más significativo, los 2 siguientes bits tienen el valor '0' lógico, los siguientes 5 bits toman el valor del módulo del exponente y los 5 bits restantes tienen el valor '0' lógico.
- 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 ("0000000100000" en formato flotante), para así no modificar el valor del otro multiplicando tras la multiplicación.
- El resultado de la unidad aritmética, valor actual de la variable "suma", se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0" si el contador de fase tiene el valor binario "00000".



NÚMERO DE OPERACIÓN "0001":

Se anula el exponente de la variable flotante utilizada para los cálculos intermedios. El valor de partida de esta variable es el coeficiente mel cuyo índice coincide con el contador de índice. Inicialmente, en la fase binaria "00000", se anulará el exponente del coeficiente mel, y posteriormente el exponente de la variable auxiliar flotante.

- La operación realizada por la unidad aritmética es una multiplicación, el primer multiplicando es el valor 1 y el segundo la variable auxiliar utilizada en los cálculos. Como se activa la señal "anula_el_exponente", al resultado, que será la misma variable auxiliar de entrada, se le anulará el valor del exponente en formato flotante.
- 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 ("0000000100000" en formato flotante), para así no modificar el valor del otro multiplicando tras la multiplicación.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos. En esta dirección, inicialmente se encuentra almacenado el coeficiente mel y posteriormente la variable auxiliar utilizada en los cálculos intermedios.
- El resultado de la unidad aritmética, valor de la variable auxiliar con el exponente del formato flotante anulado, se almacenará en la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos.

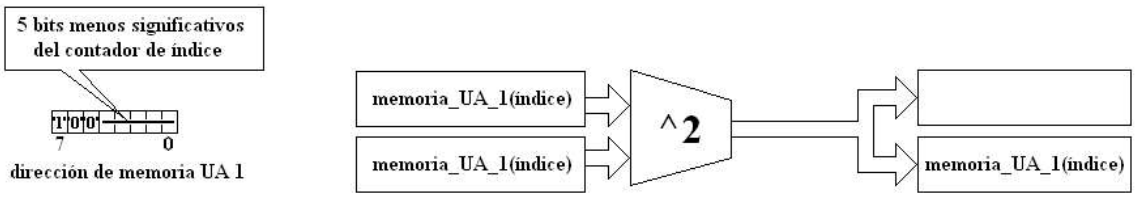


NÚMERO DE OPERACIÓN "0010":

Se eleva al cuadrado la variable flotante auxiliar utilizada para los cálculos intermedios.

- 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 el segundo operando de la unidad aritmética, ya que se realiza el cuadrado de este operando.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos. En esta dirección, inicialmente se encuentra la variable auxiliar utilizada en los cálculos intermedios.
- El resultado de la unidad aritmética, cuadrado de la variable auxiliar utilizada en los cálculos intermedios, se almacenará en la dirección de la memoria "memoria_ua_1" que tiene el valor

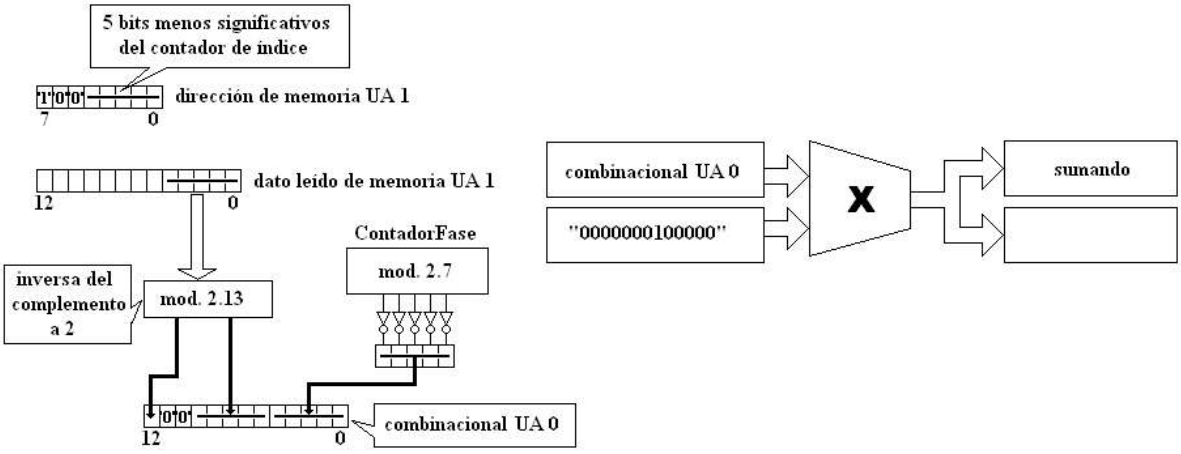
binario “100” en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos.



NÚMERO DE OPERACIÓN “0011”:

Se calcula el sumando de la fase actual de la suma que aproxima el logaritmo en base 2 del coeficiente mel. El sumando es el exponente de la variable auxiliar multiplicado por la inversa de la potencia de 2 con exponente el número de la fase (0-15) mas 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 sumando de la fase actual de la suma que aproxima el logaritmo en base 2 del coeficiente mel. Este sumando es el exponente de la variable auxiliar multiplicado por la inversa de la potencia de 2 con exponente el número de la fase (0-15) mas 1. La variable auxiliar se lee desde la dirección de la memoria “memoria_ua_1” que tiene el valor binario “100” en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos. Los 5 bits menos significativos del dato leído de esta memoria conforman el valor del exponente en complemento a 2. El componente “inv_complemento_a_2” separa el módulo y el signo del exponente. El bit más significativo de la entrada combinacional toma el bit de signo del exponente como el bit más significativo, los 2 siguientes bits tienen el valor '0' lógico, los siguientes 5 bits toman el valor del módulo del exponente y los 5 bits restantes son los 5 bits del contador de fase invertidos (de '0' lógico a '1' lógico y de '1' lógico a '0' lógico), así multiplico el exponente de la variable auxiliar por la inversa de la potencia de 2 con exponente el número de la fase mas 1.

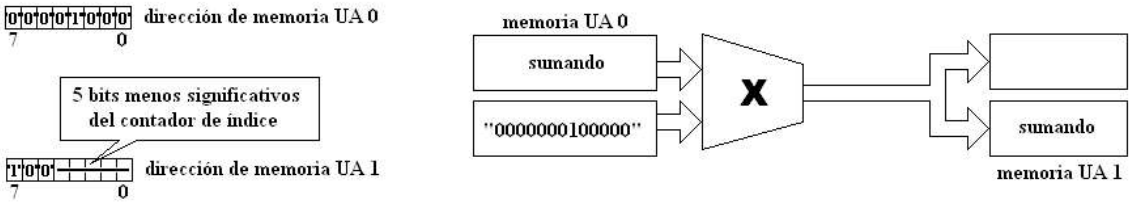


- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará el valor 1 (“0000000100000” en formato flotante), para así no modificar el valor del otro multiplicando tras la multiplicación.
- El resultado de la unidad aritmética, valor de la variable “sumando”, se almacenará en la dirección binaria “00001000” de la memoria “memoria_ua_0”.

NÚMERO DE OPERACIÓN “0100”:

Se guarda la variable “sumando” en la dirección de la memoria “memoria_ua_1” que tiene el valor binario “110” en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos.

- 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 binaria “00001000” de la memoria “memoria_ua_0”, donde se encuentra almacenada la variable “sumando”.
- 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 (“0000000100000” en formato flotante), para así no modificar el valor del otro multiplicando tras la multiplicación.
- El resultado de la unidad aritmética, valor actual de la variable “sumando”, se almacenará en la dirección de la memoria “memoria_ua_1” que tiene el valor binario “110” en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos.

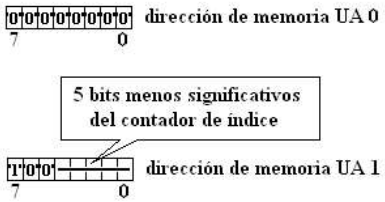


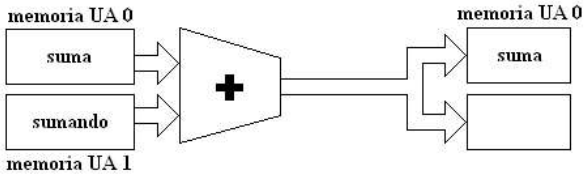
NÚMERO DE OPERACIÓN “0101”:

Se actualiza la variable “suma” con el nuevo sumando calculado en las operaciones anteriores. La variable “suma” se almacenará en la dirección binaria “00000000” de la memoria “memoria_ua_0”.

- 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 binaria “00000000” de la memoria “memoria_ua_0”, donde se encuentra almacenada la variable “suma”.

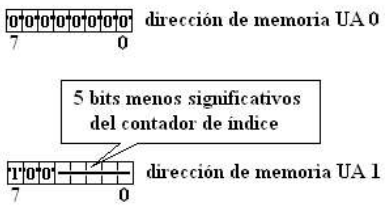
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "110" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos, donde se encuentra almacenada la variable "sumando".
- El resultado de la unidad aritmética, valor actualizado de la variable "suma", se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".

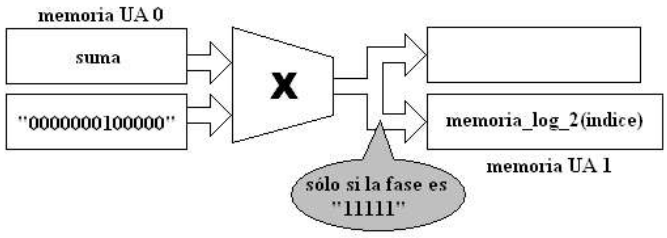




NÚMERO DE OPERACIÓN "0110":
Si la fase binaria es "11111" se almacena en la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos, el valor del logaritmo en base 2 del coeficiente mel cuyo índice coincide con 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 dirección binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenada la variable "suma".
- 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 ("0000000100000" en formato flotante), para así no modificar el valor del otro multiplicando tras la multiplicación.
- El resultado de la unidad aritmética se almacenará en la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de índice en los 5 bits menos significativos si la fase binaria es "11111". Quedará así guardado el valor del logaritmo en base 2 del coeficiente mel cuyo índice coincide con el contador de índice.





1.1.3.0.7	recnew.vhd Módulo=”110”
Durante la ejecución del módulo número “110” se calcularán 8 coeficientes cepstrales los cuales poseen características suficientes para realizar el proceso de reconocimiento vocálico.	

Al realizar la Transformada Rápida de Fourier se obtuvieron las características en frecuencia de la señal de voz codificada digitalmente por el conversor analógico-digital. Los coeficientes mel transformaron los coeficientes FFT en otros más representativos de la señal vocálica en frecuencia de acuerdo a la forma en la que el oído humano extrae también las características espectrales. En los coeficientes mel se encuentra entremezclada la información del tono generado por las cuerdas vocálicas y la que representa las características vocálicas modeladas por el tracto vocálico.

$$\text{Coeficiente Mel}(i) = \text{Aportación del tono}(i) \times \text{Aportación del tracto vocálico}(i)$$

$$i : \text{índice del espectro mel } (1 - 32)$$

Al realizar el logaritmo en base 2 de los coeficientes mel se separa en un sumando la aportación de la excitación (pitch o tono generado por las cuerdas vocálicas) y en otro sumando la aportación del filtro no lineal que puede considerarse el tracto vocálico. Es el tracto vocálico el que aporta las características a la señal de voz que diferencian a las distintas vocales.

$$\mathbf{Log}_2(\text{Coeficiente Mel}(i)) = \mathbf{Log}_2(\text{Aportación del tono}(i)) + \mathbf{Log}_2(\text{Aportación del tracto vocálico}(i))$$

$$i : \text{índice del espectro mel } (1 - 32)$$

Si se realiza la Transformada Discreta de Fourier Inversa al logaritmo en base 2 de los coeficientes mel se obtienen los coeficientes cepstrales. En el dominio cepstral, n es de nuevo el tiempo, ya que hemos aplicado una Transformada de Fourier Inversa, pero se suele llamar cuefrencia, para diferenciar claramente los coeficientes cepstrales de la señal original.

$$\mathbf{IDFT}(\mathbf{Log}_2(\text{Coeficiente Mel}(i))) = \mathbf{IDFT}(\mathbf{Log}_2(\text{Aportación del tono}(i))) + \mathbf{IDFT}(\mathbf{Log}_2(\text{Aportación del tracto vocálico}(i)))$$

$$\text{coeficiente cepstral}(n) = \mathbf{IDFT}(\mathbf{Log}_2(\text{Coeficiente Mel}(i))) = \text{cepstral del tono}(n) + \text{cepstral del tracto vocálico}(n)$$

La aportación de la excitación sólo es significativa para los valores altos de cuefrencia, siendo despreciable la aportación del tracto vocálico para las altas cuefrencias. La aportación del tracto

vocálico sólo es significativa para los valores bajos de cuefrencia, siendo despreciable la aportación de la excitación para las bajas cuefrencias. Así se consigue separar la información de la excitación y del tracto vocálico. En consecuencia, los primeros coeficientes cepstrales tienen la información del tracto vocálico que permitirá reconocer la vocal pronunciada y se ha conseguido. Al tratar sólo los primeros coeficientes cepstrales, se consigue extraer la información del tracto vocálico y eliminar la parte de información que no merece atención en el proceso de reconocimiento.

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 :

$$\text{Coeficiente Cepstral}(i) = \sum_{k=1\ldots L} \text{Log}_2(\text{Coeficiente Mel}(k)) \times \cos\left(i\left(k-\frac{1}{2}\right)\frac{\pi}{L}\right) \qquad \text{donde } i=1\ldots M$$

Los elementos de la ecuación representan:

- k es el índice del coeficiente mel.
- L es el número de bandas mel o de coeficientes mel, 32 en el caso del proyecto.
- M es el número de coeficientes cepstrales calculados, 8 en el caso del proyecto.

Para realizar el proceso implementado durante la ejecución del módulo número “110” 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):

```
%Número de coeficientes mel
L=32;

%Número de coeficientes cepstrales
M=8;

for i=1:1:M
  suma=0;
   for k=1:1:L
     suma=suma+(coeficiente_mel(k)*cos(i*(k-0.5)*pi/L));
  end
   coeficiente_cepstral(i)=suma;
end
```

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 $\pi/2$. El resto de valores del coseno entre $\pi/2$ 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.

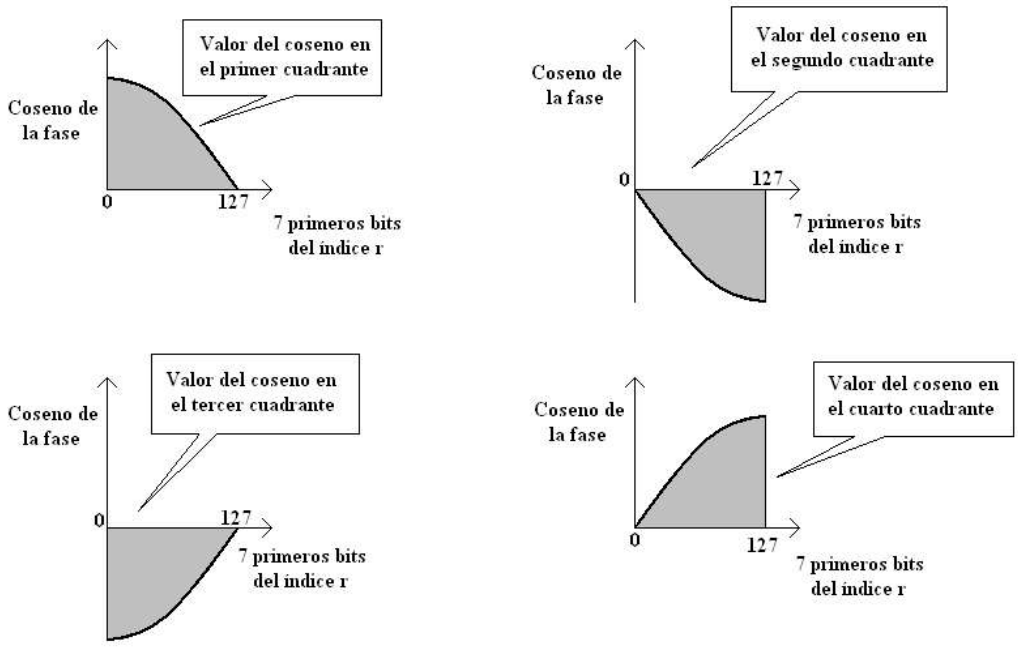
- Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el primer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el índice r.

- 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.



Para conocer los valores de los cosenos utilizados en los cálculos de los coeficientes cepstrales sólo hay que averiguar la dirección de la memoria "memoria_coseno" 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). Para ello debe aportarse como entrada el valor del índice r que se está buscando. El índice r necesario para obtener la dirección del coseno adecuado es 4 * índice cepstral * (2 * índice mel – 1). Hay que tener en cuenta que si la fase del coseno es tal que está en el segundo o en el tercer cuadrante, hay que cambiar el signo del coseno.

Valores del coseno utilizado en el calculo de los coeficientes cepstrales

$$\cos\left(i\left(k-\frac{1}{2}\right)\frac{\pi}{32}\right) \quad \text{donde} \begin{cases} i=1\ldots8 & \text{índice cepstral} \\ k=1\ldots32 & \text{índice mel} \end{cases}$$

En la memoria "memoria_coseno" se encuentran almacenados los siguientes valores del coseno

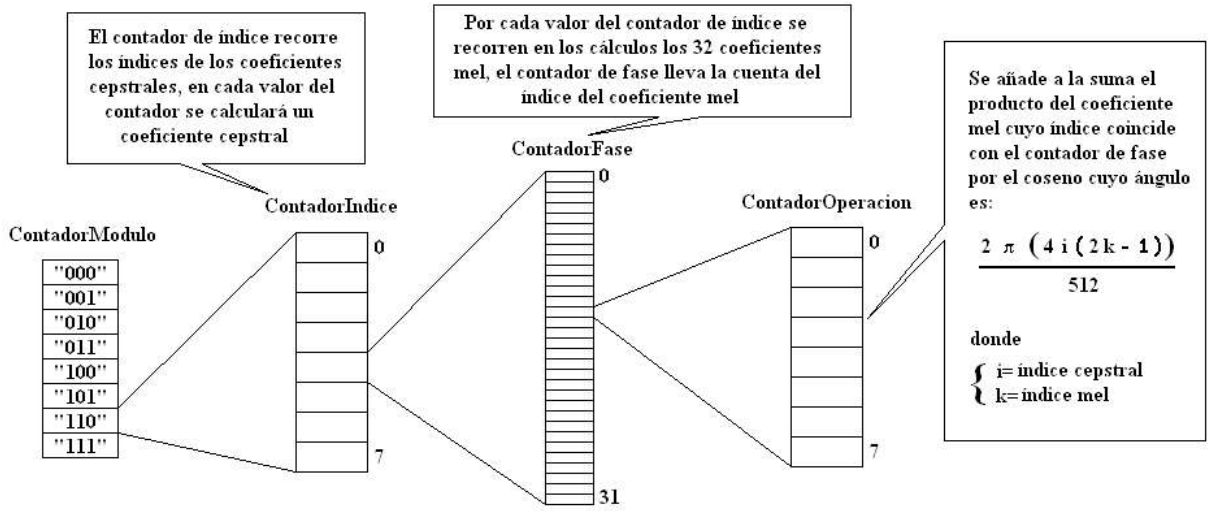
$$\cos\left(\frac{2\pi r}{512}\right) \quad \text{donde } r=0\ldots511$$

Es casi inmediato el hallar el equivalente al índice r

$$\cos\left(i\left(k-\frac{1}{2}\right)\frac{\pi}{32}\right)=\cos\left(\frac{2\pi\left(4i\left(2k-1\right)\right)}{512}\right) \quad \text{donde} \begin{cases} i=1\ldots8 & \text{índice cepstral} \\ k=1\ldots32 & \text{índice mel} \end{cases}$$

$$\text{equivalente índice } r = 4i\left(2k-1\right) \quad \text{donde} \begin{cases} i=1\ldots8 & \text{índice cepstral} \\ k=1\ldots32 & \text{índice mel} \end{cases}$$

El algoritmo en lenguaje Matlab® expuesto anteriormente sintetiza la técnica utilizada para el cálculo de los coeficientes cepstrales. Al desarrollar este algoritmo me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "111" 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 de los coeficientes cepstrales.



Se recorrerá el bucle de índice (0 – 7), y en cada uno de los índices se calculará el coeficiente cepstral cuyo índice coincide con el contador de índice. Dentro de cada índice se recorrerá el bucle de fase (0 – 31) para acceder a cada uno de los coeficientes mel. Cada coeficiente mel se multiplicará por el coseno cuya fase tiene el siguiente valor:

$$\frac{2\,\pi\,(4\,i\,(2k-1))}{512} \qquad \text{donde} \begin{cases} i=1\ldots8 & \text{índice cepstral} \\ k=1\ldots32 & \text{índice mel} \end{cases}$$

El componente "ComponenteCalculaDireccionWRN" halla la dirección de la memoria coseno desde donde se leerá el valor cos(2 * π * 4 * índice cepstral * (2 * índice mel – 1) / 512). Hay que tener en cuenta que si la fase del coseno es tal que está en el segundo o en el tercer cuadrante, hay que cambiar el signo del coseno. Se observa que el bucle de índice cepstral toma los valores que van desde 1 a 8 y que el bucle de índice mel toma los valores que van desde 1 a 32. Como el contador de índice en el módulo "110" recorre los valores que van desde 0 a 7 y el contador de fase en el módulo "110" recorre los valores que van desde 0 a 31 es necesario incrementar en una unidad los contadores de índice y de fase para que identifiquen a los índices de los coeficientes mel y los coeficientes cepstrales. Así, en realidad, cada coeficiente mel se multiplicará por el coseno cuya fase tiene el siguiente valor:

$$\frac{2\,\pi\,(4\,(i+1)(2\,(k+1)-1))}{512} \qquad \text{donde} \begin{cases} i=0\ldots7 & \text{índice cepstral} \\ k=0\ldots31 & \text{índice mel} \end{cases}$$

El producto del coeficiente mel cuyo índice coincide con el índice de fase por el valor del coseno cuya fase es (2 * π * 4 * índice cepstral * (2 * índice mel – 1) / 512) conforma el sumando actual en cálculo según el valor del índice del coeficiente mel de la suma que proporcionará una vez concluido el bucle de fase el valor del coeficiente cepstral cuyo índice coincide con el contador de índice. Hay que tener en cuenta que si la fase del coseno es tal que está en el segundo o en el tercer cuadrante, hay que cambiar el signo del coseno. Esta suma debe ser inicializada con un valor 0 antes de comenzar a recorrer el bucle de fase. El contador de operación llevará la cuenta de las operaciones cuya ejecución provocará la activación de las señales que calculará en primer lugar la dirección de memoria desde donde acceder al valor del coseno y posteriormente hallará el valor actual del sumando.

El coeficiente cepstral calculado correspondiente al índice establecido por el contador de índice se almacenará para su uso posterior en la dirección de la memoria "memoria_UA_1" que tiene el valor "110" en los 3 bits más significativos, y el índice del coeficiente cepstral (valor del contador de índice) en los 5 bits menos significativos.

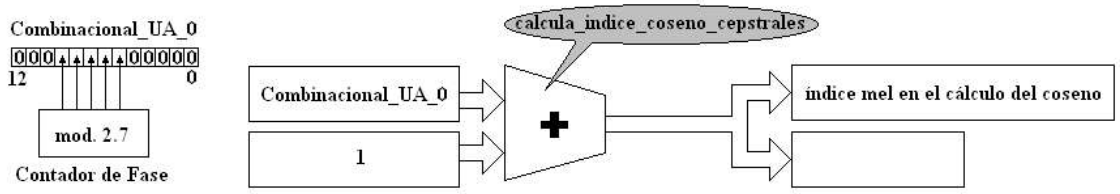
Además, si el sistema se encuentra en la fase de aprendizaje de los coeficientes cepstrales de las distintas vocales pronunciadas por el usuario, hecho indicado por una activación de la señal "carga_vocal_aprendida" con un valor '1' lógico, se almacenarán en memoria los coeficientes cepstrales calculados como los coeficientes que se utilizarán como patrón de la vocal cuyo índice coincide con el código binario de 3 bits de entrada "vocal_aprendida" en el posterior proceso de reconocimiento. En este proceso se decidirá que se ha pronunciado la vocal cuyos coeficientes

cepstrales más se acerquen al patrón guardado para esa vocal en el proceso de aprendizaje. El código binario de 3 bits de entrada "vocal_aprendida" tomará el valor "000" si se debe almacenar el patrón de reconocimiento de la vocal 'A', "001" si es el de la 'E', "010" si es el de la 'I', "011" si es el de la 'O' y "100" si es el de la 'U'. Si está activo el proceso de aprendizaje del patrón de reconocimiento, se almacenará el coeficiente cepstral calculado en la dirección de la memoria "memoria_UA_0" que tiene un '1' lógico en el bit más significativo, el código de la vocal aprendida en los siguientes 3 bits, el índice del coeficiente cepstral en los siguientes 3 bits y un '0' lógico en el bit menos significativo.

Finalmente, para el cálculo de cada sumando de la suma que proporciona el coeficiente espectral se precisan 8 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.

NÚMERO DE OPERACIÓN "0000":
Se incrementa en una unidad el valor del contador de fase para obtener el valor del índice mel utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral (recordar que el índice mel en la fase del coseno recorre los valores 1 – 32 y el contador de fase los valores 0 – 31).

- La operación realizada por la unidad aritmética es una suma. Se activa la señal "calcula_indice_coseno_cepstrales" para evitar que en el resultado se desplace la mantisa a la izquierda para conseguir el mayor número de bits significativos en ella con la consiguiente disminución del exponente. El índice utilizado en la fase del coseno debe tener un valor 0 en el exponente del formato flotante de su representación para identificar la fase. Un desplazamiento de la mantisa indicado por un exponente no nulo produciría una alteración en la fase. Esto sólo se consigue evitando el desplazamiento de la mantisa en cada una de las operaciones implicadas en el cálculo del índice de la fase del coseno.



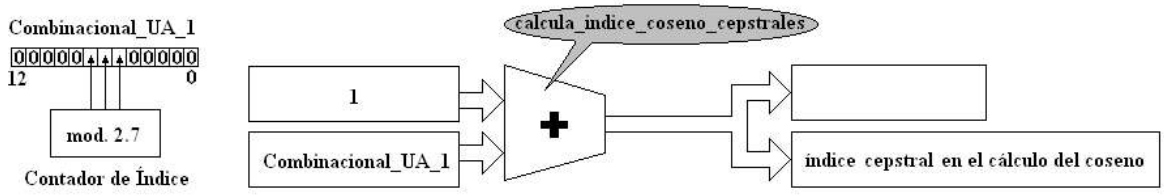
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor del contador de fase como entrada. Los 3 bits más significativos de la entrada combinacional toman el valor '0' lógico, los 5 siguientes bits tienen el valor del contador de fase y los 5 bits restantes tienen el valor '0' lógico.
- 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 ("0000000100000" en formato flotante).
- El resultado de la unidad aritmética, valor del índice mel utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral, se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".

NÚMERO DE OPERACIÓN "0001":

Se incrementa en una unidad el valor del contador de índice para obtener el valor del índice cepstral utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral (recordar que el índice cepstral en la fase del coseno recorre los valores 1 – 8 y el contador de índice los valores 0 – 7).

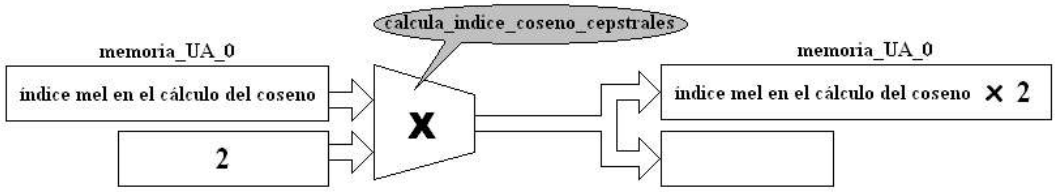
- La operación realizada por la unidad aritmética es una suma. Se activa la señal "calcula_indice_coseno_cepstrales" para evitar que en el resultado se desplace la mantisa a la izquierda para conseguir el mayor número de bits significativos en ella con la consiguiente disminución del exponente. El índice utilizado en la fase del coseno debe tener un valor 0 en el exponente del formato flotante de su representación para identificar la fase. Un desplazamiento de la mantisa indicado por un exponente no nulo produciría una alteración en la fase. Esto sólo se consigue evitando el desplazamiento de la mantisa en cada una de las operaciones implicadas en el cálculo del índice de la fase del coseno.
- 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 ("0000000100000" en formato flotante).
- 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 contador de índice como entrada. Los 5 bits más significativos de la entrada combinacional toman el valor '0' lógico, los 3 siguientes bits tienen el valor del contador de índice y los 5 bits restantes tienen el valor '0' lógico.
- El resultado de la unidad aritmética, valor del índice cepstral utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral, se almacenará en la dirección de la memoria "memoria_UA_1" que tiene el valor "101" en los 3 bits más significativos, y el índice del coeficiente cepstral (valor del contador de índice) en los 5 bits menos significativos.



NÚMERO DE OPERACIÓN "0010":

Se multiplica el índice mel por 2 en la búsqueda del valor del coseno en el algoritmo de cálculo del coeficiente cepstral. Recordar que el índice gracias al cual se accede a la dirección de memoria donde está almacenado el coseno es (4 * índice cepstral * (2 * índice mel – 1).

- La operación realizada por la unidad aritmética es una multiplicación. Se activa la señal "calcula_indice_coseno_cepstrales" para evitar que en el resultado se desplace la mantisa a la izquierda para conseguir el mayor número de bits significativos en ella con la consiguiente disminución del exponente. El índice utilizado en la fase del coseno debe tener un valor 0 en el exponente del formato flotante de su representación para identificar la fase. Un desplazamiento de la mantisa indicado por un exponente no nulo produciría una alteración en la fase. Esto sólo se consigue evitando el desplazamiento de la mantisa en cada una de las operaciones implicadas en el cálculo del índice de la fase del coseno.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el valor del índice mel utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral.
- 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 2 ("0000001000000" en formato flotante).
- El resultado de la unidad aritmética, indice mel en el cálculo de la fase del coseno multiplicado por 2, se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".

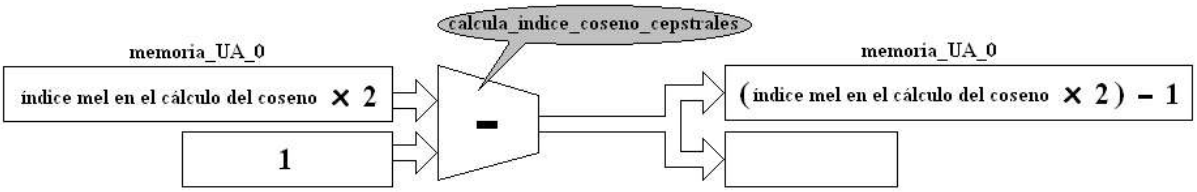


NÚMERO DE OPERACIÓN "0011":

Se le resta una unidad al doble del índice mel en la búsqueda del valor del coseno en el algoritmo de cálculo del coeficiente cepstral. Recordar que el índice gracias al cual se accede a la dirección de memoria donde está almacenado el coseno es (4 * índice cepstral * (2 * índice mel – 1).

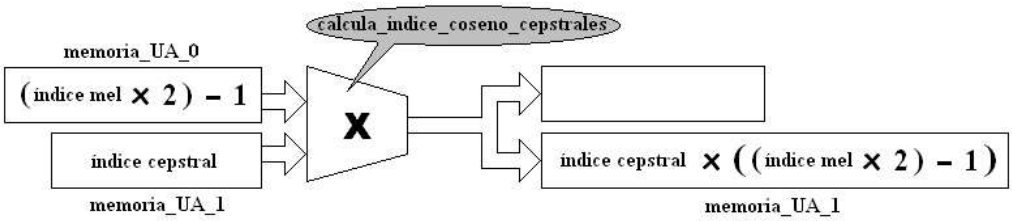
- La operación realizada por la unidad aritmética es una resta. Se activa la señal "calcula_indice_coseno_cepstrales" para evitar que en el resultado se desplace la mantisa a la izquierda para conseguir el mayor número de bits significativos en ella con la consiguiente disminución del exponente. El índice utilizado en la fase del coseno debe tener un valor 0 en el exponente del formato flotante de su representación para identificar la fase. Un desplazamiento de la mantisa indicado por un exponente no nulo produciría una alteración en la fase. Esto sólo se consigue evitando el desplazamiento de la mantisa en cada una de las operaciones implicadas en el cálculo del índice de la fase del coseno.

- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el índice mel multiplicado por 2.
- 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 ("0000000100000" en formato flotante).
- El resultado de la unidad aritmética, indice mel en el cálculo de la fase del coseno multiplicado por 2 menos 1, se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".



NÚMERO DE OPERACIÓN "0100":
Se multiplica la resta de una unidad al doble del índice mel por el índice cepstral en la búsqueda del valor del coseno en el algoritmo de cálculo del coeficiente cepstral. Recordar que el índice gracias al cual se accede a la dirección de memoria donde está almacenado el coseno es (4 * índice cepstral * (2 * índice mel – 1).

- La operación realizada por la unidad aritmética es una multiplicación. Se activa la señal "calcula_indice_coseno_cepstrales" para evitar que en el resultado se desplace la mantisa a la izquierda para conseguir el mayor número de bits significativos en ella con la consiguiente disminución del exponente. El índice utilizado en la fase del coseno debe tener un valor 0 en el exponente del formato flotante de su representación para identificar la fase. Un desplazamiento de la mantisa indicado por un exponente no nulo produciría una alteración en la fase. Esto sólo se consigue evitando el desplazamiento de la mantisa en cada una de las operaciones implicadas en el cálculo del índice de la fase del coseno.



- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el índice mel multiplicado por 2 menos una unidad.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_UA_1" que tiene el valor "101" en los 3 bits más significativos, y el índice del coeficiente cepstral (valor del contador de índice) en

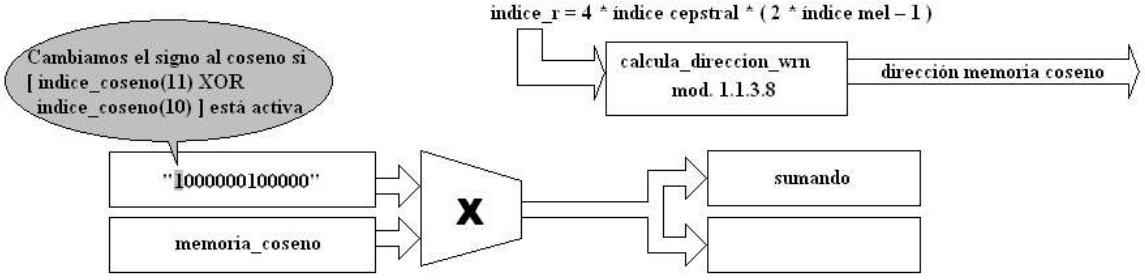
los 5 bits menos significativos, donde se encuentra almacenado el valor del índice cepstral utilizado para hallar la fase del coseno en el algoritmo de cálculo del coeficiente cepstral.

- El resultado de la unidad aritmética, índice cepstral * (2 * índice mel – 1), se almacenará en la dirección de la memoria "memoria_UA_1" que tiene el valor "101" en los 3 bits más significativos, y el índice del coeficiente cepstral (valor del contador de índice) en los 5 bits menos significativos. También se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".

NÚMERO DE OPERACIÓN "0101":

Se guarda en memoria el valor del coseno utilizado en el sumando actual del algoritmo de cálculo del coeficiente cepstral. El componente "ComponenteCalculaDireccionWRN" halla la dirección de la memoria coseno desde donde se leerá el valor cos(2 π r / 512). El índice r necesario para obtener la dirección del coseno adecuado es 4 * índice cepstral * (2 * índice mel – 1), cuyo valor se calculó en las operaciones anteriores. El signo del coseno depende también del índice r.

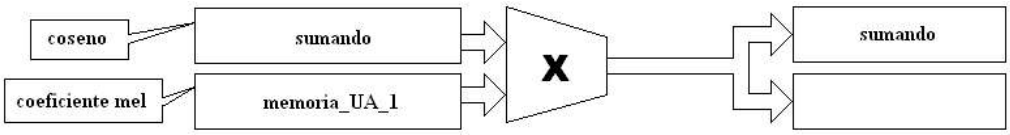
- La operación realizada por la unidad aritmética es una multiplicación.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 1 ("0000000100000" en formato flotante) si el signo del coseno debe ser positivo y proporcionará el valor -1 ("1000000100000" en formato flotante) si el signo del coseno debe ser positivo. El signo depende del valor del índice r, y será negativo sólo si los bits número 11 y 10 tienen distintos valores lógicos, lo cual indica que la fase del coseno está en el segundo o tercer cuadrante. Los 6 bits más significativos del índice r lo conforman los 6 bits menos significativos de la mantisa del dato leído de la memoria "memoria_ua_0", donde está almacenado el valor (índice cepstral * (2 * índice mel – 1)). Los 2 bits menos significativos del índice r toman el valor '0' lógico. De esta forma, desplazando 2 bits hacia la izquierda, se consigue multiplicar el valor de la mantisa por 4.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_coseno" proporcionada por el componente "ComponenteCalculaDireccionWRN".
- El resultado de la unidad aritmética, valor del coseno utilizado en el sumando actual del algoritmo de cálculo del coeficiente cepstral, se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".



NÚMERO DE OPERACIÓN "0110":

Se guarda en memoria el valor del sumando actual en el cálculo del coeficiente cepstral. Es el resultado de multiplicar el cos(2 * π * 4 * índice cepstral * (2 * índice mel – 1) / 512) por el coeficiente mel cuyo índice coincide con el contador de fase.

- 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 binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el valor del coseno utilizado en el sumando actual del algoritmo de cálculo del coeficiente cepstral.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_ua_1" que tiene el valor binario "100" en los 3 bits más significativos y el valor del contador de fase en los 5 bits menos significativos. En este lugar se encuentra almacenado el valor del coeficiente mel cuyo índice coincide con el contador de fase.
- El resultado de la unidad aritmética, valor del sumando que se añadirá a la suma que proporciona el coeficiente cepstral, se almacenará en la dirección binaria "00000000" de la memoria "memoria_ua_0".



NÚMERO DE OPERACIÓN "0111":

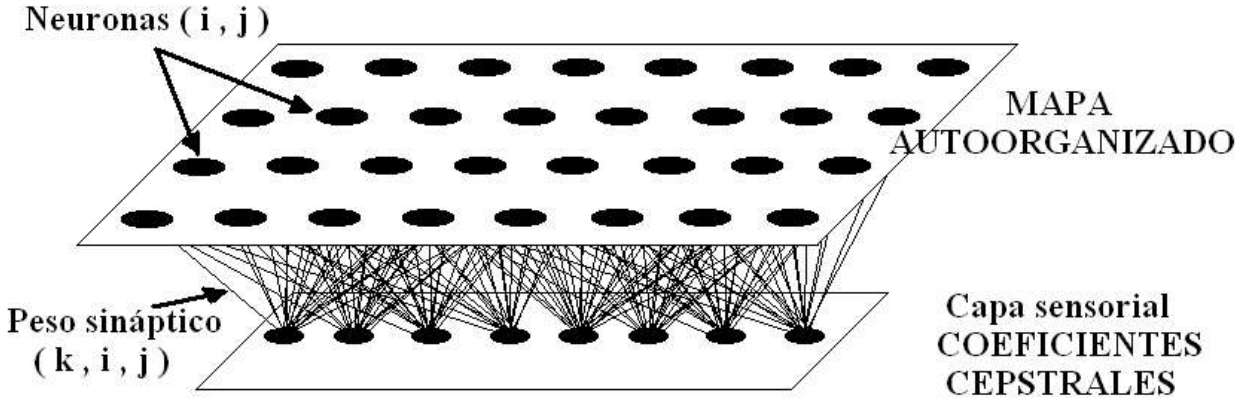
Se actualizará la suma que proporciona el coeficiente cepstral con el valor del nuevo sumando. Una vez concluido el bucle de fase, quedará almacenado en memoria el coeficiente cepstral cuyo índice coincide con el contador de índice.

- 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 binaria "00000000" de la memoria "memoria_ua_0", donde se encuentra almacenado el sumando actual.
- 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 0 ("0000000000000" en formato flotante) si la fase binaria es "00000". Este es el valor inicial de la suma. En caso contrario, leerá el valor anterior de la suma desde la memoria "memoria_UA_1".
- El resultado de la unidad aritmética, valor actual de la suma o valor del coeficiente cepstral cuyo índice coincide con el contador de índice si ha concluido el bucle de fase, se almacenará en la dirección de la memoria "memoria_UA_1" que tiene el valor "110" en los 3 bits más significativos, y el índice del coeficiente cepstral (valor del contador de índice) en los 5 bits menos significativos. Si el sistema está en la fase de aprendizaje, se almacenará también en la dirección de la memoria "memoria_UA_0" que tiene un '1' lógico en el bit más significativo, el código de la vocal aprendida en los siguientes 3 bits, el índice del coeficiente cepstral en los siguientes 3 bits y un '0' lógico en el bit menos significativo.

1.1.3.0.8	recnew.vhd Módulo=”111”

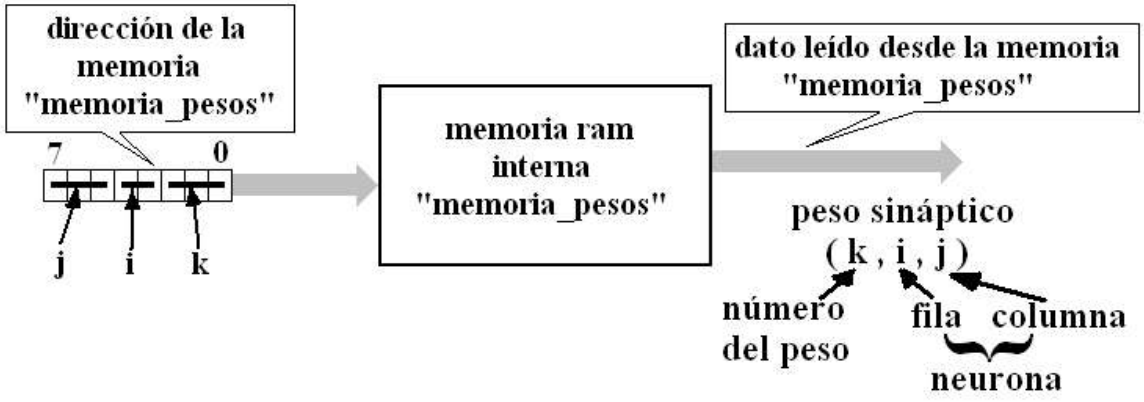
En este módulo una red neuronal de mapas autoorganizados de 32 neuronas realiza el reconocimiento de la vocal pronunciada. Los parámetros de entrada de la red neuronal son los coeficientes cepstrales de las 512 muestras de la señal de audio procedentes del conversor analógico-digital. Ante los parámetros 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”.

Se encuentran almacenados en memoria los coeficientes cepstrales que condensan la información que el tracto vocálico infirió a la señal de voz para pronunciar una vocal. Estos coeficientes cepstrales identifican a la vocal pronunciada. Los ocho coeficientes cepstrales de las 512 muestras provenientes del conversor analógico-digital conforman la capa sensorial o primera capa de la arquitectura unidireccional de dos capas de la red neuronal. Las posiciones de memoria donde se encuentran almacenados los coeficientes cepstrales pueden modelarse como los ocho buffers o las ocho neuronas de la capa de entrada de la red neuronal. En la segunda capa existen 32 neuronas distribuidas espacialmente en una arquitectura rectangular formada por 4 filas y 8 columnas. El número de la fila y de la columna determinan la localización espacial de las neuronas de la segunda capa. Cada neurona de la capa de entrada (buffer donde se encuentra almacenado uno de los coeficientes cepstrales) está conectada con todas las neuronas de la segunda capa mediante un peso sináptico. Si etiquetamos las neuronas de entrada mediante el índice k ($1 \le k \le 8$) y las neuronas de la capa de salida con los índices (i, j) (el índice i recorre la fila $1 \le i \le 4$ y el índice j recorre la columna $1 \le j \le 8$) denominaremos al peso sináptico que une la neurona de la capa de entrada número k (valor del coeficiente cepstral número k) con la neurona de la segunda capa en la fila i y en la columna j como w(k,i,j).



Los pesos sinápticos se obtienen en mediante un proceso de autoaprendizaje realizado de forma externa en un ordenador. En este proceso de aprendizaje, cada neurona del mapa autoorganizado sintoniza de forma sucesiva con los diversos rasgos de las distintas señales vocálicas. De forma sucesiva y aleatoria se presenta a la red neuronal en la fase de aprendizaje los coeficientes cepstrales de distintas pronunciaciones vocálicas. Cada neurona en paralelo calcula la similitud entre su vector de pesos sinápticos y los coeficientes cepstrales mostrados en la entrada. Se determina la neurona ganadora (g1,g2) como aquella que tenga la distancia cuadrática euclídea menor entre el vector de pesos sinápticos y los coeficientes cepstrales detectados. En el proceso de aprendizaje se actualizan los pesos sinápticos de la neurona ganadora y de sus vecinas de acuerdo con una función de vecindad para que se parezcan más aún a los coeficientes cepstrales. Debido a la actualización de la neurona ganadora y de sus vecinas, en el modelo de mapas autoorganizados se logra que neuronas próximas sintonicen con patrones similares. A medida que se presentan patrones cepstrales de distintas vocales en la red en la fase de aprendizaje, el ritmo de actualización decrece con el número de muestras presentadas a la vez que disminuye el radio de influencia en actualización de neuronas vecinas a la ganadora. La disminución en el ritmo de actualización y en el radio de influencia es muy lenta para asegurar la convergencia en el algoritmo. Finalmente, los pesos sinápticos de las neuronas hacen que éstas se distribuyan topológicamente en zonas que comparten similitud en la detección de los parámetros vocálicos. Es decir, si una vocal posee diversos parámetros diferenciadores, un grupo de neuronas cercanas con pesos sinápticos sintonizados con los parámetros diferenciadores serán las que se activen cuando se pronuncie esa vocal. En definitiva, según la neurona que se active cuando se le muestren los coeficientes cepstrales podrá detectarse la vocal que ha sido pronunciada si éste es el caso.

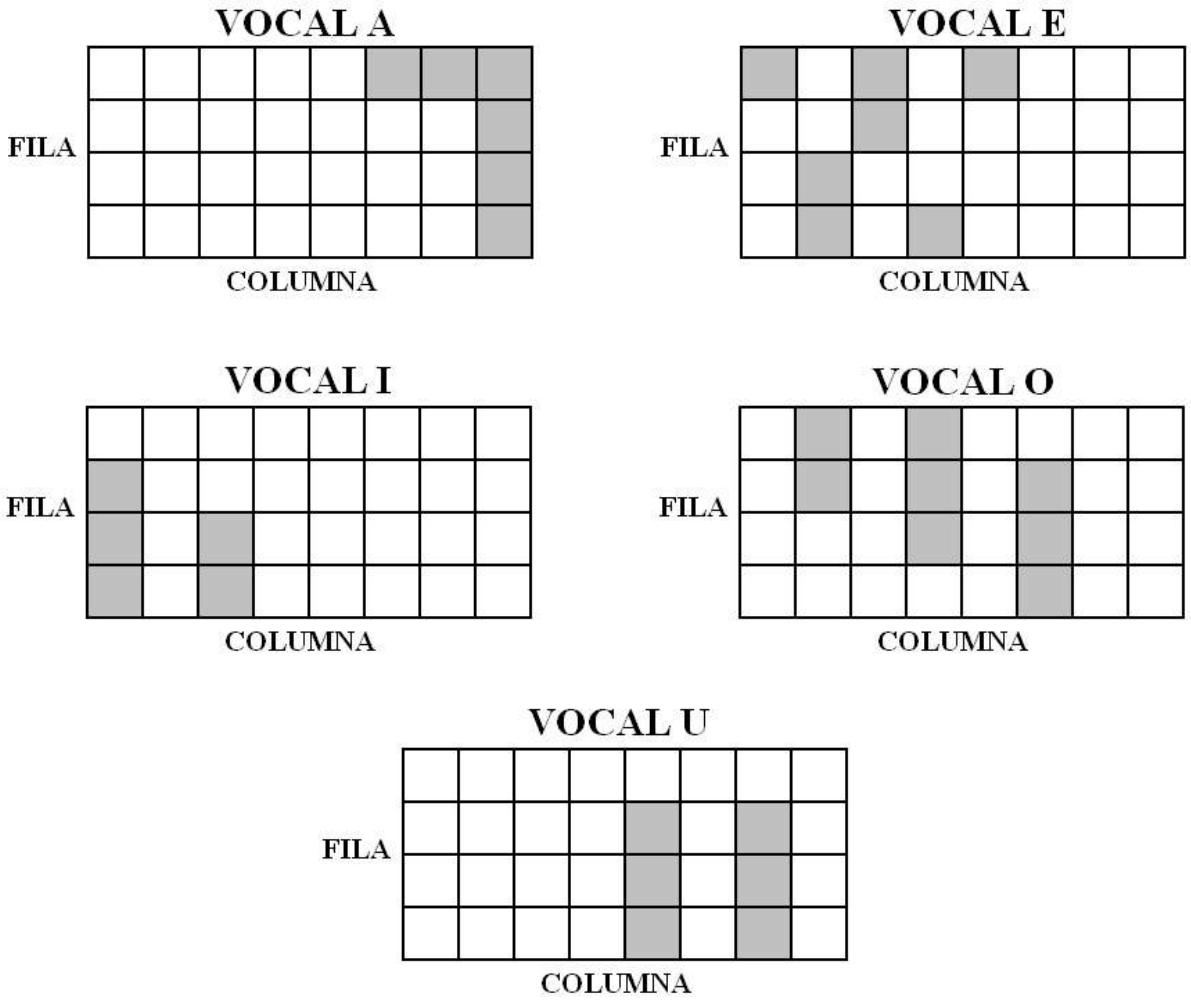
La ejecución del módulo número “111” no modela la fase de aprendizaje, ya que los pesos sinápticos ya fueron establecidos anteriormente y de forma externa para el funcionamiento de la red neuronal. Este módulo modela la fase de ejecución u operación normal y práctica de la red neuronal. Ya no se realiza ninguna actualización de los pesos sinápticos, ya que éstos están ya fijados. Cada neurona calcula la distancia euclídea cuadrática existente entre los pesos sinápticos de la neurona y los coeficientes cepstrales detectados. A continuación, se declara vencedora la neurona (g1,g2), cuyo vector de pesos es más similar a los coeficientes cepstrales. La distancia euclídea es menor cuanto más similares sean los pesos a los coeficientes. La neurona ganadora informa de los rasgos percibidos y puede decidirse la vocal pronunciada en función de la neurona ganadora.



Los pesos sinápticos de las neuronas se encuentran almacenados en la memoria "memoria_pesos" en el formato flotante utilizado por la unidad aritmética. Estos pesos se encontraban almacenados en la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA. Durante la ejecución del módulo "000" del componente recnew se leyeron los pesos y se almacenaron en las posiciones de la memoria "memoria_pesos". Los 3 bits menos significativos de la dirección de memoria indican el número del peso en la neurona (8 pesos sinápticos por neurona) y los 5 bits más significativos de la dirección de memoria indican el número de la neurona (32 neuronas distribuidas en 4 filas y 8 columnas).

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 las siguientes figuras según el número de la fila y el número de la columna de la neurona ganadora.

NEURONAS QUE SE ACTIVAN CON LA PRONUNCIACIÓN DE LAS DISTINTAS VOCALES



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.

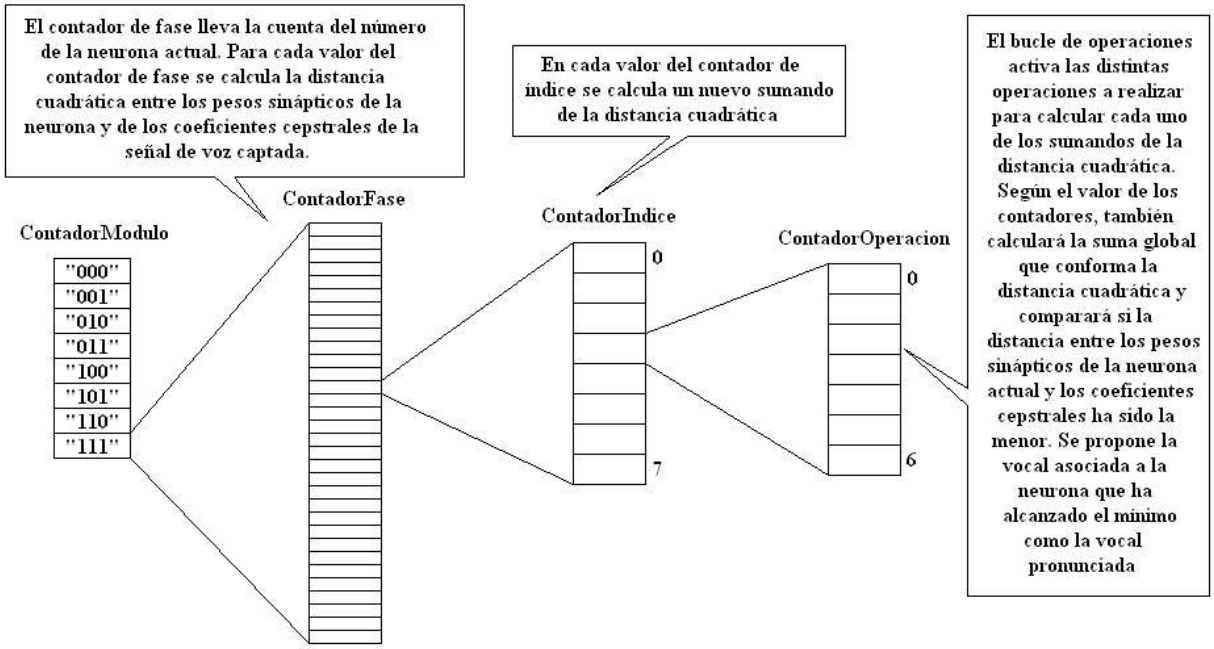
Para realizar el proceso de reconocimiento vocálico mediante redes neuronales implementado durante la ejecución del módulo número "111" del componente "recnew" se ejecutan una serie de operaciones que pueden sintetizarse en el siguiente algoritmo (se utiliza el lenguaje Matlab® para describir el algoritmo):

```
minimo = inf;
for i=0:1:31
  suma = 0;
  for j=0:1:7
    diferencia = peso(i,j) - cepstral(j);
    cuadrado =  diferencia ^ 2;
    suma = suma + cuadrado;
    if minimo > sumando
      minimo = sumando;
      codigo_vocal = j;
    end
  end
  if minimo > sumando
    minimo = sumando;
    codigo_vocal = vocal_asociada_a_la_neurona(i);
  end
end
```

Para comparar el vector de pesos sinápticos de cada una de las neuronas con el vector de coeficientes cepstrales se calcula la distancia cuadrática entre los pesos sinápticos de la neurona y los coeficientes cepstrales detectados. Se activará la neurona que tenga los pesos sinápticos cuya distancia cuadrática con los coeficientes cepstrales detectados sea menor. Se decidirá que se ha pronunciado la vocal asociada a la neurona ganadora (cada neurona tiene asociada una vocal). La distancia cuadrática entre los pesos sinápticos de cada neurona y los coeficientes cepstrales detectados se calcula realizando la suma del cuadrado de la diferencia entre cada uno de los pesos sinápticos de la neurona y del coeficiente cepstral de la vocal detectada que tienen el mismo orden. Al recorrer las distintas neuronas (32 neuronas distribuidas en 4 filas y 8 columnas) para buscar aquella que provoca la mínima distancia se compara al final del cálculo de cada distancia cuadrática el valor de la distancia con la distancia mínima alcanzada hasta el momento. Si la neurona actual en análisis genera una distancia cuadrática menor que el mínimo de distancia almacenada, debe proponerse esta distancia como la distancia mínima alcanzada hasta el

momento en el análisis de la vocal pronunciada. La vocal asociada a la neurona que haya generado la mínima distancia será la vocal propuesta en el proceso de reconocimiento.

El algoritmo en lenguaje Matlab® expuesto anteriormente sintetiza la técnica utilizada para reconocer la vocal pronunciada. Al desarrollar este algoritmo me aprovecho de la lógica existente en el componente "recnew". Cuando se esté ejecutando el módulo número "110" 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 reconocimiento de la vocal pronunciada.



Se recorrerá en primer lugar el bucle de fase (0 – 31), el contador de fase lleva la cuenta del número de la neurona actual, y en cada uno de los valores de la fase se calculará la distancia cuadrática entre los pesos sinápticos de la neurona actual y los coeficientes cepstrales de la señal de audio captada. Se comparará este valor con el mínimo global alcanzado y si es menor, se actualizará el valor del mínimo con el valor de la distancia cuadrática calculada. A la vez que se almacena el mínimo, también se almacena el código de la vocal que ha generado el mínimo. Cada neurona tiene asociada una vocal y será el código de la vocal asociada el que se almacenará como posible vocal si esa neurona ha generado el mínimo. El mínimo se inicializa con el valor de la distancia calculada en la fase "00000" binaria, y se almacena también el código de la vocal asignada a la neurona primera, es decir, la neurona de la fase "00000". Esto es así, pues en la fase "00000" no hay asignado todavía un mínimo de las distancias cuadráticas.

Una vez terminado el proceso de reconocimiento, se almacenará en un registro el código de la vocal que se corresponde con la neurona que ha generado la mínima distancia cuadrática almacenada ("vocal_detectada_aux"). Si la energía de la vocal supera el umbral establecido, se activará al mismo tiempo el flag que indica que se ha recibido una vocal

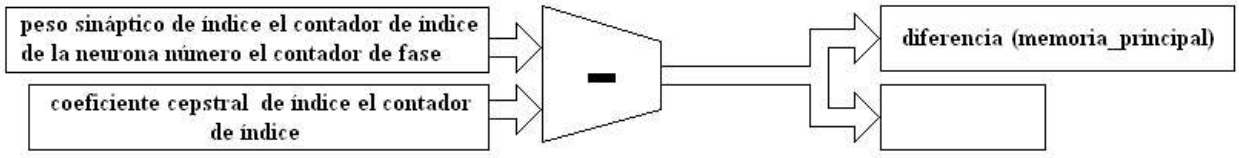
("flag_vocal_detectada_aux"). Estas señales son el resultado del proceso de reconocimiento e indican al sistema si se ha detectado una vocal y cuál ha sido la vocal detectada.

Dentro de cada fase se recorrerá el bucle de índice (0 – 7) para añadir a la suma que representa la distancia cuadrática en cada índice el sumando correspondiente al cuadrado de la diferencia existente entre el coeficiente cepstral detectado cuyo índice se corresponde con el contador de índice y el peso sináptico de la neurona actual identificada por el contador de fase y cuyo índice se corresponde con el contador de índice. La suma que representa la distancia cuadrática entre los coeficientes cepstrales de la vocal detectada y los pesos sinápticos en cada una de las neuronas analizadas se inicializa con el valor 0 cuando el contador de índice toma el valor 0. Para cada uno de los valores del índice, incluido el 0, se añadirá un nuevo valor a la suma.

Finalmente, para el cálculo de cada sumando de la suma que proporciona la distancia cuadrática entre los coeficientes cepstrales de la vocal detectada y los pesos sinápticos de la neurona actual en análisis, se precisan 7 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.

NÚMERO DE OPERACIÓN "0000":
Se calcula la diferencia entre el peso sináptico de la neurona cuyo número lo identifica el contador de fase y de índice indicado por el contador de índice y el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada.

- 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 de la memoria "memoria_pesos" que tiene el número de la neurona en los 5 bits más significativos (5 bits menos significativos del contador de fase) y el número del peso en los 3 bits menos significativos (3 bits menos significativos del contador de índice). En este lugar se encuentra almacenado el peso sináptico cuyo número se corresponde con el contador de índice de la neurona cuyo número se corresponde con el contador de fase.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección de la memoria "memoria_UA_1" que tiene el valor "11000" en los 5 bits más significativos, y el índice del coeficiente cepstral (3 bits menos significativos del valor del contador de índice) en los 3 bits menos significativos. En este lugar se encuentra almacenado el coeficiente cepstral de la señal de audio captada de índice el número del contador de índice.



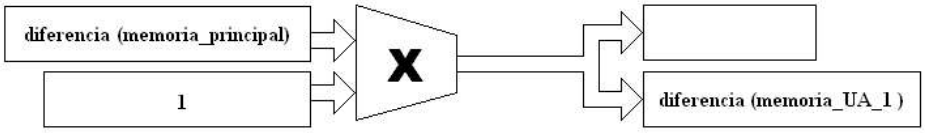
- El resultado de la unidad aritmética, diferencia entre el peso sináptico de la neurona cuyo número lo identifica el contador de fase y de índice indicado por el contador de índice y el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada, se almacenará en la dirección binaria que tiene un '0' lógico en los 7 bits más significativos y los

3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria principal.

NÚMERO DE OPERACIÓN "0001":

Se almacena en un lugar más adecuado la diferencia entre el peso sináptico de la neurona cuyo número lo identifica el contador de fase y de índice indicado por el contador de índice y el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada.

- La operación realizada por la unidad aritmética es una multiplicación por 1 para no variar el valor del multiplicando.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la dirección binaria que tiene un '0' lógico en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria principal. En este lugar se encuentra almacenada la diferencia entre el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada y el coeficiente cepstral de índice el valor del contador de índice del patrón asignado a la vocal correspondiente al valor del contador de fase.



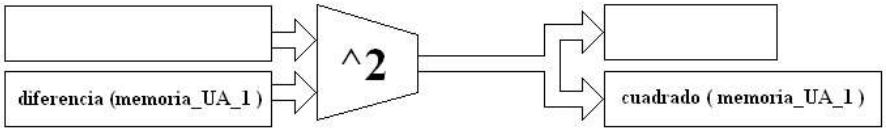
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la entrada combinacional.
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del segundo operando proporcionará el valor 1 ("0000000100000" en formato flotante) para no modificar el valor del multiplicando.
- El resultado de la unidad aritmética, diferencia entre el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada y el coeficiente cepstral de índice el valor del contador de índice del patrón asignado a la vocal correspondiente al valor del contador de fase, se almacenará en la dirección binaria que tiene en valor binario "1000000" en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria "memoria_UA_1".

NÚMERO DE OPERACIÓN "0010":

Se almacena al cuadrado la diferencia entre el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada y el coeficiente cepstral de índice el valor del contador de índice del patrón asignado a la vocal correspondiente al valor del contador de fase.

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

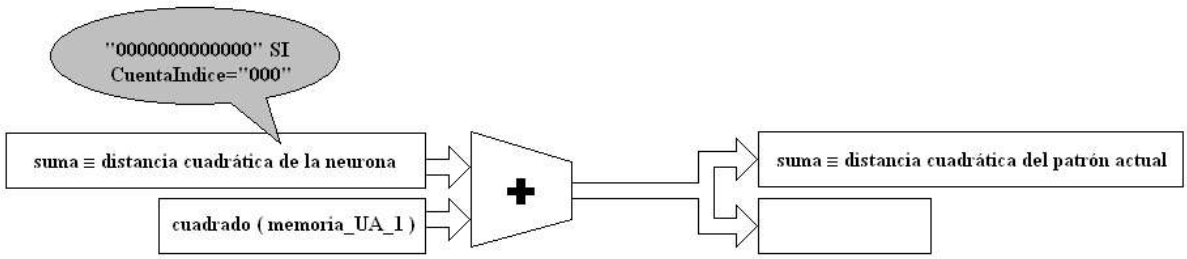
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada al segundo operando de la unidad aritmética ya que se realiza el cuadrado del segundo operando.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección binaria que tiene en valor binario "1000000" en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria "memoria_UA_1". En este lugar se encuentra almacenada la diferencia entre el peso sináptico y el coeficiente cepstral.
- El resultado de la unidad aritmética, cuadrado de la diferencia entre el peso sináptico y el coeficiente cepstral, se almacenará en la dirección binaria que tiene en valor binario "1000000" en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria "memoria_UA_1".



NÚMERO DE OPERACIÓN "0011":

Se actualiza el valor de la suma añadiendo el cuadrado anterior como sumando. Una vez concluido el bucle de índice la suma tendrá el valor de la distancia cuadrática entre el peso sináptico de la neurona cuyo número lo identifica el contador de fase y de índice indicado por el contador de índice y el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada. La suma inicial es cero, hecho que se tendrá en cuenta cuando el valor del contador de índice es 0.

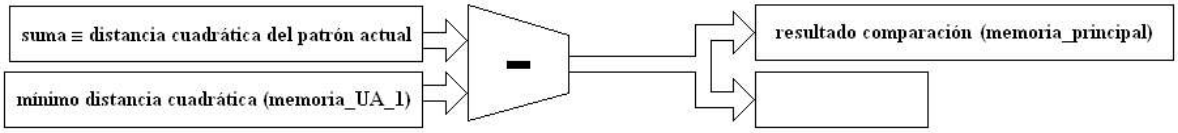
- La operación realizada por la unidad aritmética es una suma.
- El multiplexor encargado de proporcionar el primer operando a la unidad aritmética toma su valor desde la entrada combinacional si el contador de índice es 0. Si el contador de índice tiene un valor distinto de 0, el multiplexor proporcionará el primer operando a la unidad aritmética desde la dirección binaria "00000000" de la memoria "memoria_UA_0", donde se encuentra almacenada la variable "suma".
- La lógica combinacional encargada de generar la entrada combinacional del multiplexor del primer operando proporcionará el valor 0 ("0000000000000" en formato flotante) como valor inicial de la suma que aporta el valor de la distancia cuadrática.
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección binaria que tiene en valor binario "1000000" en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria "memoria_UA_1". En este lugar se encuentra almacenado el nuevo sumando de la distancia cuadrática.
- El resultado de la unidad aritmética, distancia cuadrática entre el peso sináptico de la neurona cuyo número lo identifica el contador de fase y de índice indicado por el contador de índice y el coeficiente cepstral de índice el valor del contador de índice de la señal de audio detectada, se almacenará en la dirección binaria "00000000" de la memoria "memoria_UA_0". En este lugar se encuentra almacenada la variable "suma".



NÚMERO DE OPERACIÓN "0100":

Se compara el valor de la distancia cuadrática actual calculada con el mínimo hallado de la distancia cuadrática. Si el resultado tiene un signo negativo, la distancia cuadrática calculada es menor que el mínimo almacenado, por lo que habrá que actualizar el valor del mínimo.

- 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 binaria "00000000" de la memoria "memoria_UA_0". En este lugar se encuentra almacenada la variable "suma".
- El multiplexor encargado de proporcionar el segundo operando a la unidad aritmética toma su valor desde la dirección binaria "00100000" de la memoria "memoria_UA_1". En este lugar se encuentra almacenado el mínimo.
- El resultado de la unidad aritmética, resultado de la comparación de la suma con el mínimo, se almacenará en la dirección binaria que tiene un '0' lógico en los 7 bits más significativos y los 3 bits menos significativos del contador de índice en los 3 bits menos significativos de la memoria principal.

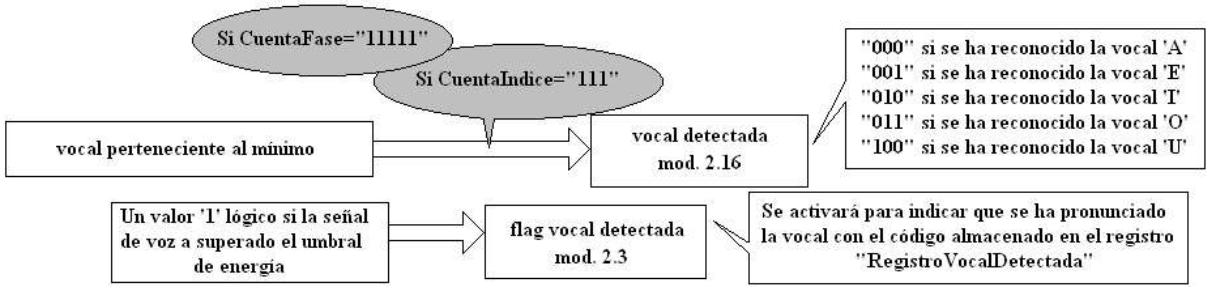


NÚMERO DE OPERACIÓN "0101":

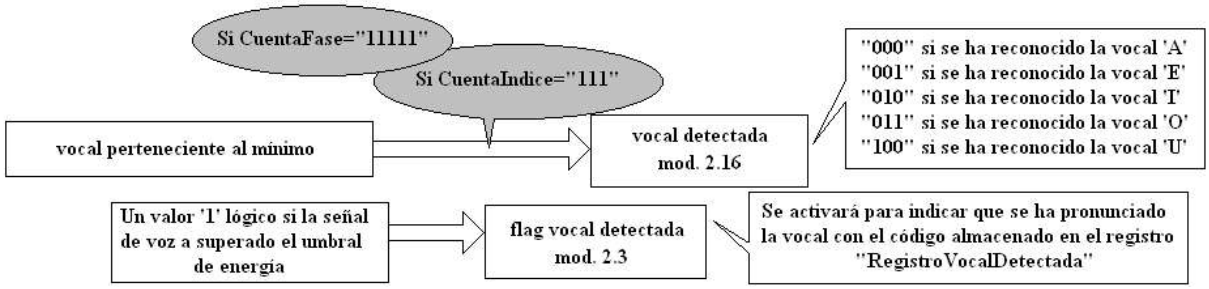
Si la distancia cuadrática calculada (para que haya concluido el cálculo el contador de índice debe tener el valor binario "111") es menor que el mínimo actual, se actualiza el mínimo y se almacena el número de la vocal correspondiente al nuevo mínimo.

- 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 binaria "00000000" de la memoria "memoria_UA_0". En este lugar se encuentra almacenada la variable "suma".
- 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 ("0000000100000" en formato flotante) para no modificar el valor del multiplicando.

- El resultado de la unidad aritmética, valor de la distancia cuadrática calculada, se almacenará en la dirección binaria "00100000" de la memoria "memoria_UA_1" como nuevo mínimo de la distancia cuadrática si el contador de índice tiene el valor binario "111" y el bit de signo del resultado de la comparación es negativo ('1' lógico). También se almacenará en el registro "RegistroVocalMinimo" el código de la vocal correspondiente a la neurona que ha generado el mínimo. Este código se encuentra almacenado en los 3 bits más significativos (del bit número 15 al bit número 13) del peso sináptico leído desde la memoria "memoria_pesos". De esta forma se tiene acceso a la vocal asignada a la neurona del mínimo. El código de la vocal almacenado en el registro es provisional, pues hasta no recorrer todas las neuronas no se sabrá si se ha alcanzado el mínimo.



NÚMERO DE OPERACIÓN "0110":
Una vez concluido el proceso de reconocimiento, es decir, se ha recorrido el último bucle de índice (índice actual es "111") de la fase "11111" (se han recorrido todas las neuronas), se almacenará en el registro "RegistroVocalDetectada" el código de la vocal reconocida: "000" si se ha reconocido la vocal 'A', "001" si se ha reconocido la vocal 'E', "010" si se ha reconocido la vocal 'I', "011" si se ha reconocido la vocal 'O' y "100" si se ha reconocido la vocal 'U'. El flag almacenado en el registro "RegistroFlagVocalDetectada" se activará con un valor '1' lógico si la energía de la señal ha superado el umbral a partir del cual se considera que se ha pronunciado una vocal. Si este flag se encuentra activo, el sistema reconocerá que se ha pronunciado la vocal asignada al código almacenado en el registro "RegistroVocalDetectada".



1.1.3.1 memoria_principal.vhd

```
ENTITY memoria_principal IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    carga_en_memoria_principal: IN STD_LOGIC;
    direccion_de_memoria_principal: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
    dato_a_escribir_en_memoria_principal: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    dato_leido_de_memoria_principal: OUT STD_LOGIC_VECTOR(12 DOWNTO 0)
    );
END memoria_principal;
```

Módulo de memoria RAM de 2 Kbytes de capacidad, donde se almacenarán las 512 muestras digitales de la señal de audio.

En este módulo de memoria RAM de 2 Kbytes de capacidad se almacenan 512 muestras digitales de la señal de audio provenientes del decodificador analógico-digital externo cada vez que se ejecuta el proceso de reconocimiento vocálico. 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.

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 la memoria. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CARGA_EN_MEMORIA_PRINCIPAL: señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones de la memoria principal de los datos entregados en el bus de datos cuando está activa con un nivel lógico '1'. Si la señal de carga no está habilitada, entonces se realiza una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones.
DIRECCION_DE_MEMORIA_PRINCIPAL: bus de direcciones de la memoria principal de 10 bits (1024 direcciones). Con este bus accedo a las 1024 palabras de 13 bits de la memoria tanto para la escritura como para la lectura
DATO_A_ESCRIBIR_EN_MEMORIA_PRINCIPAL: bus de datos de entrada que posee la información que se guardará en memoria cuando se active la señal de carga. El dato está en el formato binario flotante utilizado por la unidad aritmética y se almacenará en la dirección aportada por el bus de direcciones de la memoria principal.
DATO_LEIDO_DE_MEMORIA_PRINCIPAL: dato leído desde la dirección de memoria proporcionada por el bus de direcciones. El dato está en el formato binario flotante utilizado por la unidad aritmética.

1.1.3.2 memoria_UA_0.vhd

```
ENTITY memoria_UA_0 IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    carga_en_memoria_UA_0: IN STD_LOGIC;
    direccion_de_memoria_UA_0: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dato_a_escribir_en_memoria_UA_0: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    dato_leido_de_memoria_UA_0: OUT STD_LOGIC_VECTOR(12 DOWNTO 0)
    );
END memoria_UA_0;
```

Módulo de memoria RAM de 512 Kbytes de capacidad, donde se almacenarán las variables que han sido calculadas por la unidad aritmética y que luego serán el primer operando en una nueva operación de la unidad aritmética.

Este módulo de memoria tiene posibilidad de almacenar 256 palabras de 13 bits cada una, ya que los datos que podrán almacenarse están en el formato binario flotante de 13 bits utilizado por la unidad aritmética en el proceso de reconocimiento. 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.

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 la memoria. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CARGA_EN_MEMORIA_UA_0: señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones de la memoria de las variables primer operando "memoria_ua_0" de los datos entregados en el bus de datos cuando está activa con un nivel lógico '1'. Si la señal de carga no está habilitada, entonces se realiza una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones.
DIRECCION_DE_MEMORIA_UA_0: bus de direcciones de la memoria principal de 8 bits (256 direcciones). Con este bus accedo a las 256 palabras de 13 bits de la memoria tanto para la escritura como para la lectura
DATO_A_ESCRIBIR_EN_MEMORIA_UA_0: bus de datos de entrada que posee la información que se guardará en memoria cuando se active la señal de carga. El dato está en el formato binario flotante utilizado por la unidad aritmética y se almacenará en la dirección aportada por el bus de direcciones de la memoria de variables "memoria_UA_0".
DATO_LEIDO_DE_MEMORIA_UA_0: dato leído desde la dirección de memoria proporcionada por el bus de direcciones. El dato está en el formato binario flotante utilizado por la unidad aritmética.

1.1.3.3 memoria_UA_1.vhd

```
ENTITY memoria_UA_1 IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    carga_en_memoria_UA_1: IN STD_LOGIC;
    direccion_de_memoria_UA_1: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dato_a_escribir_en_memoria_UA_1: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    dato_leido_de_memoria_UA_1: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
    );
END memoria_UA_1;
```

Módulo de memoria RAM de 512 Kbytes de capacidad, donde se almacenarán las variables que han sido calculadas por la unidad aritmética y que luego serán el segundo operando en una nueva operación de la unidad aritmética.

Este módulo de memoria tiene posibilidad de almacenar 256 palabras de 2 bytes cada una, ya que los datos que podrán almacenarse tendrán el formato binario flotante de 13 bits utilizado por la unidad aritmética en el proceso de reconocimiento en los 13 bits menos significativos y los 3 bits más significativos se utilizarán para almacenar la vocal cuyo patrón se está guardando en el proceso de aprendizaje. 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.

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 la memoria. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CARGA_EN_MEMORIA_UA_1: señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones de la memoria de los datos entregados en el bus de datos cuando está activa con un nivel lógico '1'. Si la señal de carga no está habilitada, entonces se realiza una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones.
DIRECCION_DE_MEMORIA_UA_1: bus de direcciones de la memoria principal de 8 bits (256 direcciones). Con este bus accedo a las 256 palabras de 16 bits de la memoria tanto para la escritura como para la lectura
DATO_A_ESCRIBIR_EN_MEMORIA_UA_1: bus de datos de entrada que posee la información que se guardará en memoria cuando se active la señal de carga. El dato está en el formato binario flotante utilizado por la unidad aritmética y se almacenará en la dirección aportada por el bus de direcciones de la memoria de variables "memoria_UA_1".
DATO_LEIDO_DE_MEMORIA_UA_1: dato leído desde la dirección de memoria proporcionada por el bus de direcciones.

1.1.3.4 memoria_coseno.vhd

```
ENTITY memoria_coseno IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    carga_en_memoria_coseno: IN STD_LOGIC;
    direccion_de_memoria_coseno: IN STD_LOGIC_VECTOR(6 DOWNTO 0);
    dato_a_escribir_en_memoria_coseno: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    dato_leido_de_memoria_coseno: OUT STD_LOGIC_VECTOR(12 DOWNTO 0)
    );
END memoria_coseno;
```

Módulo de memoria RAM de 128 palabras de 13 bits cada una, donde se almacenarán 128 aproximaciones a los valores del coseno.

Los valores que se almacenarán en la memoria son el coseno de las 128 fases en que se dividiría el primer cuadrante. El resto de cosenos para fases distintas de las del primer cuadrante se calculan en función de las anteriores. Estos valores 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.

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 la memoria. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
CARGA_EN_MEMORIA_COSENO: señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones de la memoria de los valores del coseno de los datos entregados en el bus de datos cuando está activa con un nivel lógico '1'. Si la señal de carga no está habilitada, entonces se realiza una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones.
DIRECCION_DE_MEMORIA_COSENO: bus de direcciones de la memoria principal de 7 bits (128 direcciones). Con este bus accedo a las 128 palabras de 13 bits de la memoria tanto para la escritura como para la lectura
DATO_A_ESCRIBIR_EN_MEMORIA_COSENO: bus de datos de entrada que posee la información que se guardará en memoria cuando se active la señal de carga. El dato está en el formato binario flotante utilizado por la unidad aritmética y se almacenará en la dirección aportada por el bus de direcciones de la memoria de valores del coseno.
DATO_LEIDO_DE_MEMORIA_COSENO: dato leído desde la dirección de memoria proporcionada por el bus de direcciones. El dato está en el formato binario flotante utilizado por la unidad aritmética.

1.1.3.5 unidad_aritmetica.vhd

```
ENTITY unidad_aritmetica IS
  PORT(
    entrada_0: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    entrada_1: IN STD_LOGIC_VECTOR(12 DOWNTO 0);
    suma: IN STD_LOGIC;
    resta: IN STD_LOGIC;
    multiplica: IN STD_LOGIC;
    anula_el_exponente: IN STD_LOGIC;
    calcula_indice_coseno_cepstrales: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(12 DOWNTO 0);
    mayor_que: OUT STD_LOGIC
  );
END unidad_aritmetica;
```

Unidad aritmética utilizada para resolver las operaciones de suma, resta y multiplicación en el proceso de reconocimiento.

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.

La unidad aritmética dispone de dos entradas de datos en el formato numérico anterior y una salida de datos también en el mismo formato binario flotante. El resultado de la operación de los datos de entrada se muestra en la salida de datos. Las operaciones básicas de la unidad aritmética lógica son las siguientes:

- SUMA: Muestra a la salida de datos en el formato numérico de representación en coma flotante la suma de las entradas de datos también en el formato numérico de representación en coma flotante utilizado en el proyecto.

- RESTA: Muestra a la salida de datos en el formato numérico de representación en coma flotante la resta de la entrada segunda de datos a la entrada primera de datos. Las entradas de datos están también en el formato numérico de representación en coma flotante utilizado en el proyecto.

- MULTIPLICACIÓN: Muestra a la salida de datos en el formato numérico de representación en coma flotante la multiplicación de las entradas de datos también en el formato numérico de representación en coma flotante utilizado en el proyecto.

Si se encuentra activa la señal de entrada "anula_el_exponente" con un valor '1' lógico, se anulará el exponente del resultado. Es decir, los 5 bits menos significativos tomarán un valor '0' lógico.

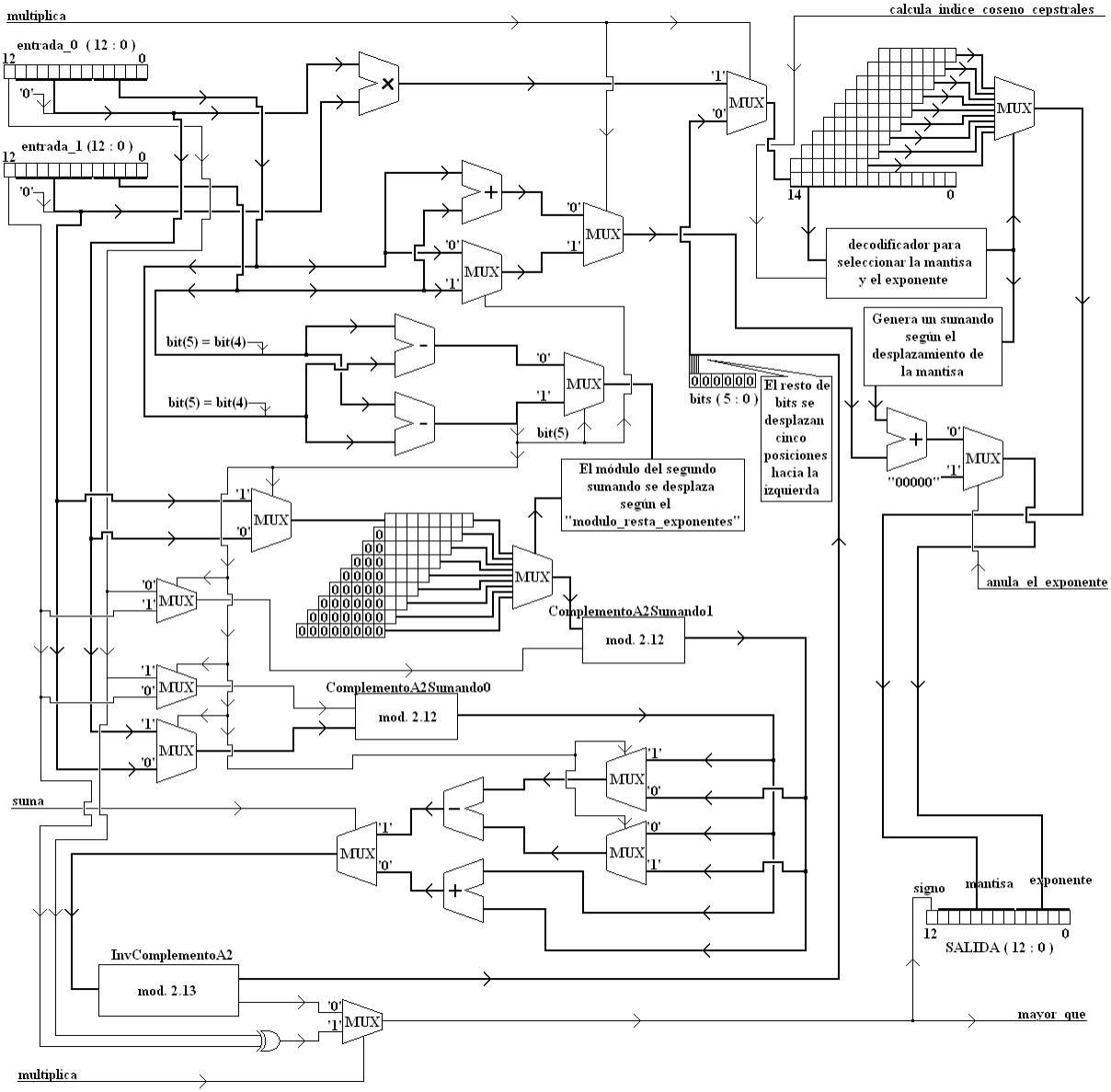
La mantisa del resultado se desplazará hacia la izquierda para conseguir que el módulo de la mantisa ofrezca como resultado la mayor información posible. Es decir, se desplazará la mantisa hacia la izquierda para que el bit más significativo de módulo sea un uno lógico. Por ejemplo, si en el producto se obtuviera un módulo como "00000101101001", la mantisa tomaría el valor "0000010". Pero con el desplazamiento anteriormente mencionado tomará el valor que proporcione la mayor información posible, y desplazando 5 unidades hacia la izquierda la mantisa global sería "1011010". El desplazamiento que se realice debe verse reflejado en el exponente y en el caso particular del ejemplo habría que restarle 5 al exponente.

Si se activa la señal "calcula_indice_coseno_cepstrales" con un valor '1' lógico, sólo se producirán hasta 3 desplazamientos hacia la izquierda en el caso de que sea preciso el desplazamiento del módulo para aumentar el número de bits activos con un valor '1' lógico. Por ejemplo, si en la suma se obtuviera un módulo como "01001100111010", la mantisa tomaría el valor "1001100" y si en el producto se obtuviera un módulo como "00001100111010", la mantisa tomaría el valor "0011001". El exponente se modificará según el desplazamiento conseguido hacia la izquierda del módulo. Esta operación es útil cuando se esté calculando el índice del coseno usado en la obtención de los coeficientes cepstrales.

INTERFAZ DEL MÓDULO:

ENTRADA_0: Dato primero en formato flotante. Dentro de una operación, será el primer sumando si se realiza una suma, será el minuendo si se realiza una resta y el multiplicando si se realiza una multiplicación.
ENTRADA_1: Dato segundo en formato flotante. Dentro de una operación, será el segundo sumando si se realiza una suma, será el sustraendo si se realiza una resta y el multiplicador si se realiza una multiplicación.
SUMA: Esta señal indica cuando está activa con un valor '1' lógico que la operación que se realizará será una suma.
RESTA: Esta señal indica cuando está activa con un valor '1' lógico que la operación que se realizará será una resta.
MULTIPLICA: Esta señal indica cuando está activa con un valor '1' lógico que la operación que se realizará será una multiplicación.
ANULA_EL_EXPONENTE: Si se encuentra activa la señal de entrada "anula_el_exponente" con un valor '1' lógico, se anulará el exponente del resultado. Es decir, los 5 bits menos significativos tomarán un valor '0' lógico.

CALCULA_INDICE_COSENO_CEPSTRALES: Si se activa la señal "calcula_indice_coseno_cepstrales" con un valor '1' lógico, sólo se producirán hasta 3 desplazamientos hacia la izquierda en el caso de que sea preciso el desplazamiento del módulo para aumentar el número de bits activos con un valor '1' lógico.
SALIDA: Resultado de la suma, resta o multiplicación realizada con los dos datos de entrada. El resultado se ofrece en el formato flotante usado en el diseño.
MAYOR_QUE: Si esta señal está activa con un valor uno lógico indica que el dato incluido a través de la entrada "entrada_0" es mayor que el dato incluido a través de la entrada "entrada_1". Es útil a la hora de realizar una comparación entre dos valores, para lo cual debe activarse la señal "resta".

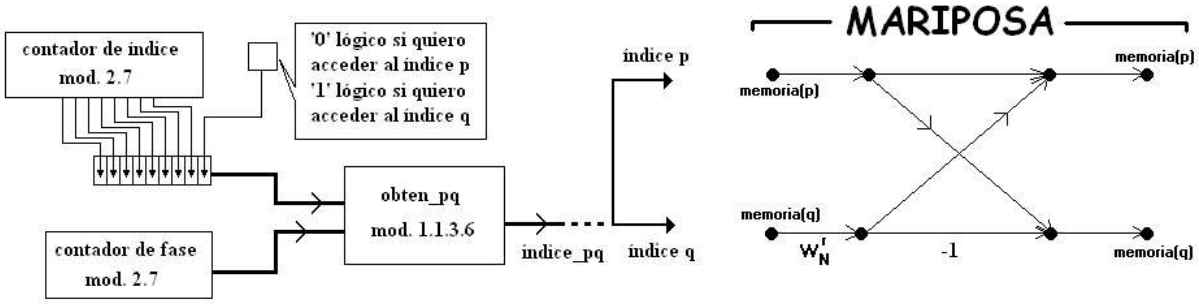


1.1.3.6 obten_pq.vhd

```
ENTITY obten_pq IS
  PORT(
    indice: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
    fase: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    indice_pq: OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
  );
END obten_pq;
```

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).

El cálculo básico que se realiza para hallar la Transformada Rápida de Fourier consiste en coger dos variables complejas, 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). 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. Este cálculo básico se denomina mariposa, dado que el diagrama de flujo recuerda a una mariposa.



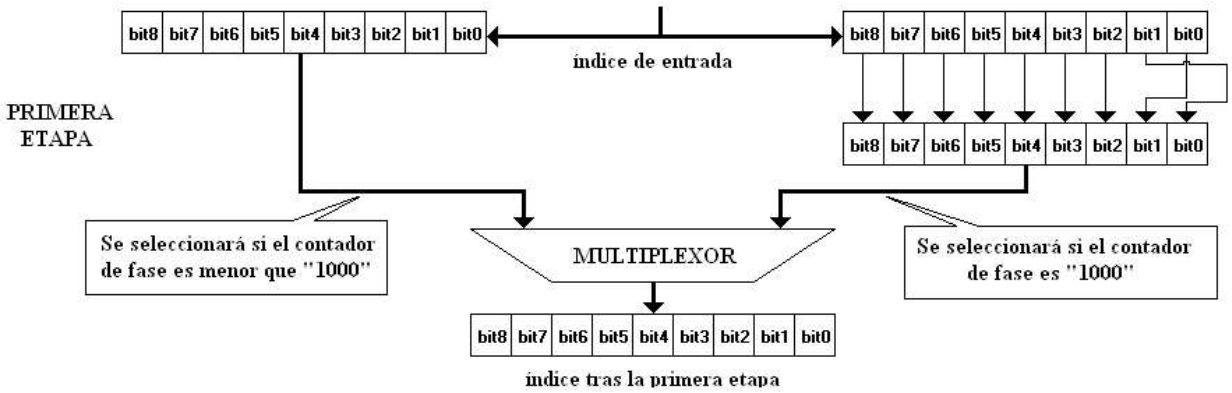
El módulo "obten_pq" proporciona tanto el índice p como el índice q. El acceso al índice p se produce cuando el bit menos significativo del índice de entrada al módulo es '0' lógico. El acceso al índice q se produce cuando el bit menos significativo del índice de entrada al módulo es '1'

lógico. Los 8 bits más significativos del índice de entrada al módulo lo conforman los 7 bits menos significativos del contador de índice utilizado en los algoritmos de reconocimiento.

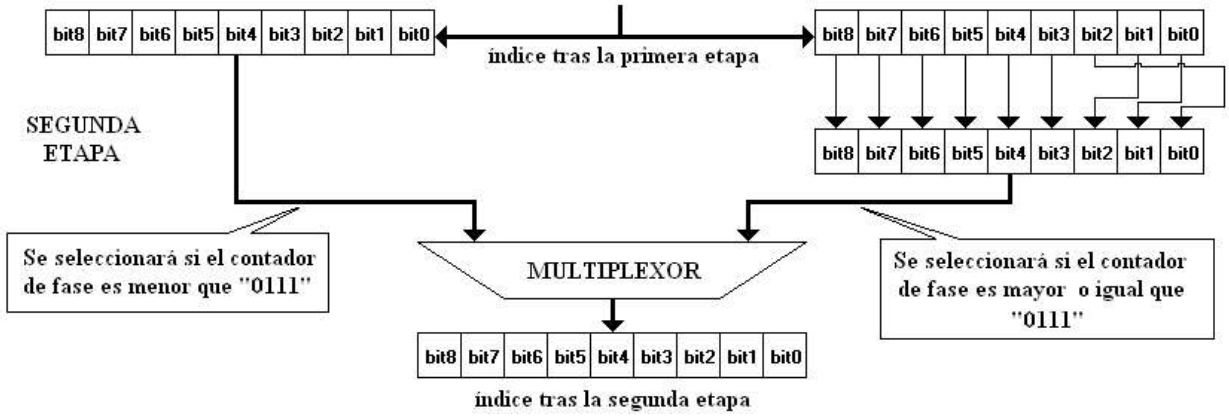
La parte real de la primera variable se encuentra almacenada en la dirección de la memoria principal que se forma de la siguiente manera: un '0' lógico en el bit más significativo (parte real), el valor del índice "indice_pq" en los 9 bits menos significativos del bus de direcciones de la memoria principal (para que el índice "indice_pq" proporcione el índice p, el bit menos significativo de entrada al módulo debe ser '0' lógico). La parte imaginaria de la primera variable se encuentra almacenada en la dirección de la memoria principal que se forma de la siguiente manera: un '1' lógico en el bit más significativo (parte imaginaria), el valor del índice "indice_pq" en los 9 bits menos significativos del bus de direcciones de la memoria principal (para que el índice "indice_pq" proporcione el índice p, el bit menos significativo de entrada al módulo debe ser '0' lógico). La parte real de la segunda variable se encuentra almacenada en la dirección de la memoria principal que se forma de la siguiente manera: un '0' lógico en el bit más significativo (parte real), el valor del índice "indice_pq" en los 9 bits menos significativos del bus de direcciones de la memoria principal (para que el índice "indice_pq" proporcione el índice q, el bit menos significativo de entrada al módulo debe ser '1' lógico). La parte imaginaria de la segunda variable se encuentra almacenada en la dirección de la memoria principal que se forma de la siguiente manera: un '1' lógico en el bit más significativo (parte imaginaria), el valor del índice "indice_pq" en los 9 bits menos significativos del bus de direcciones de la memoria principal (para que el índice "indice_pq" proporcione el índice q, el bit menos significativo de entrada al módulo debe ser '1' lógico).

El índice de salida "indice_pq" que indica el valor del índice p o del índice q (según el bit menos significativo del índice de entrada) se obtiene a partir del índice de entrada de 9 bits a partir de una serie de desplazamientos en los bits.

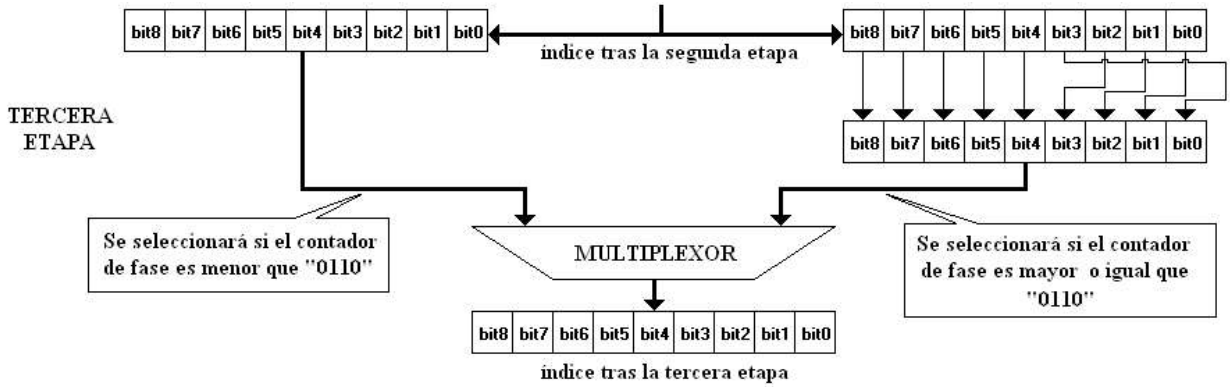
El desplazamiento del índice se produce en un proceso de 8 etapas.



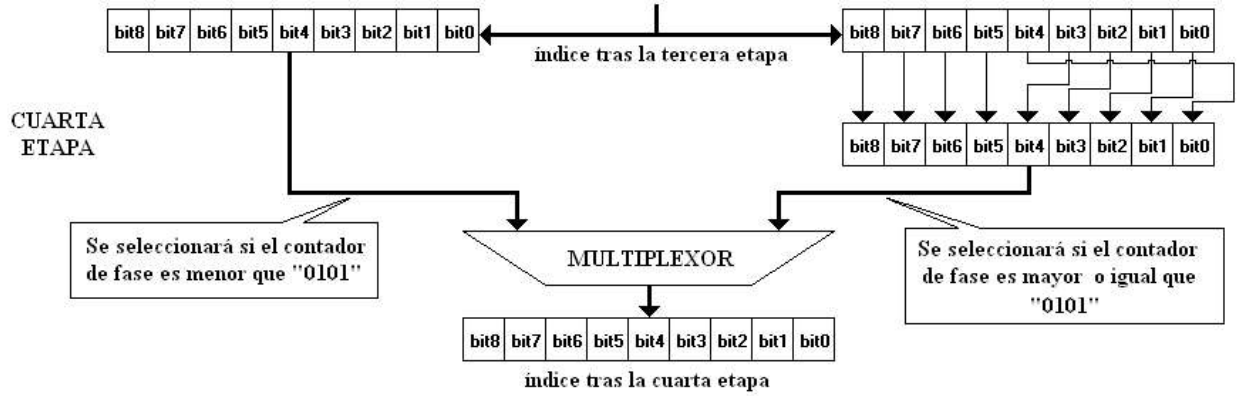
En la primera etapa se traslada el bit de la posición 1 del índice de entrada a la posición 0 del índice de la primera etapa. El bit de la posición 0 del índice de entrada se traslada a la posición 1 del índice de la primera etapa. El bit de la posición 1 se traslada a la posición 2. El resto de los bits mantienen su posición.



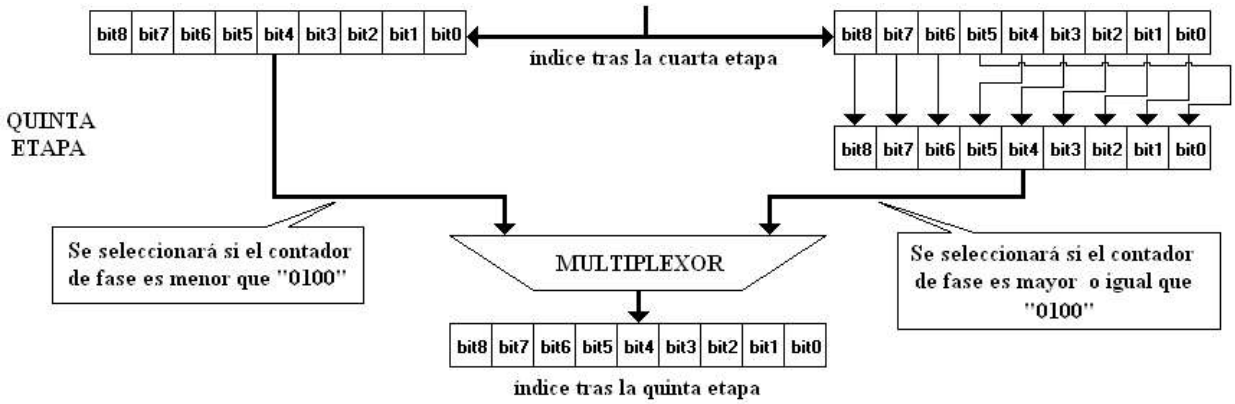
En la segunda etapa se traslada el bit de la posición 2 del índice tras la primera etapa a la posición 0 del índice de la segunda etapa. El bit de la posición 0 del índice tras la primera etapa se traslada a la posición 1 del índice de la segunda etapa. El bit de la posición 1 se traslada a la posición 2. El resto de los bits mantienen su posición.



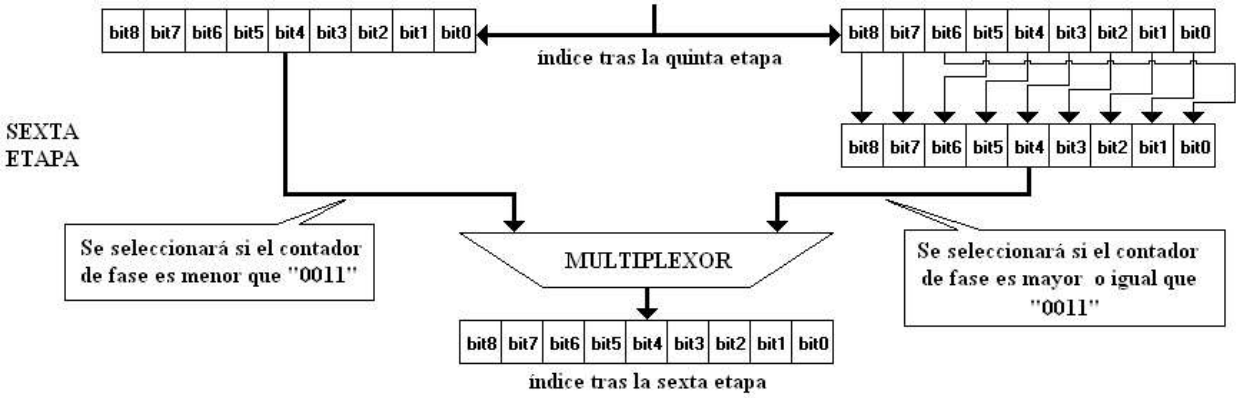
En la tercera etapa se traslada el bit de la posición 3 del índice tras la segunda etapa a la posición 0 del índice de la tercera etapa. El bit de la posición 0 del índice tras la segunda etapa se traslada a la posición 1 del índice de la tercera etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El resto de los bits mantienen su posición.



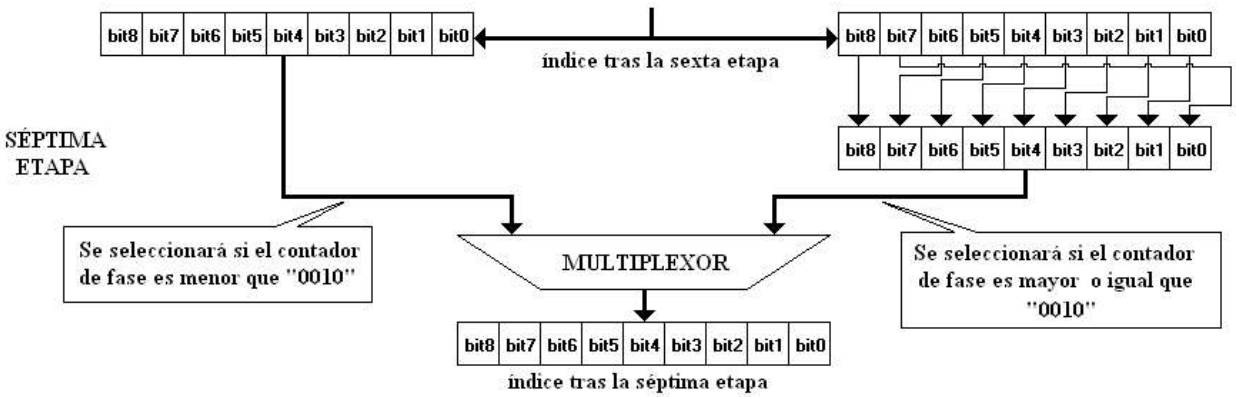
En la cuarta etapa se traslada el bit de la posición 4 del índice tras la tercera etapa a la posición 0 del índice de la cuarta etapa. El bit de la posición 0 del índice tras la tercera etapa se traslada a la posición 1 del índice de la cuarta etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El bit de la posición 3 se traslada a la posición 4. El resto de los bits mantienen su posición.



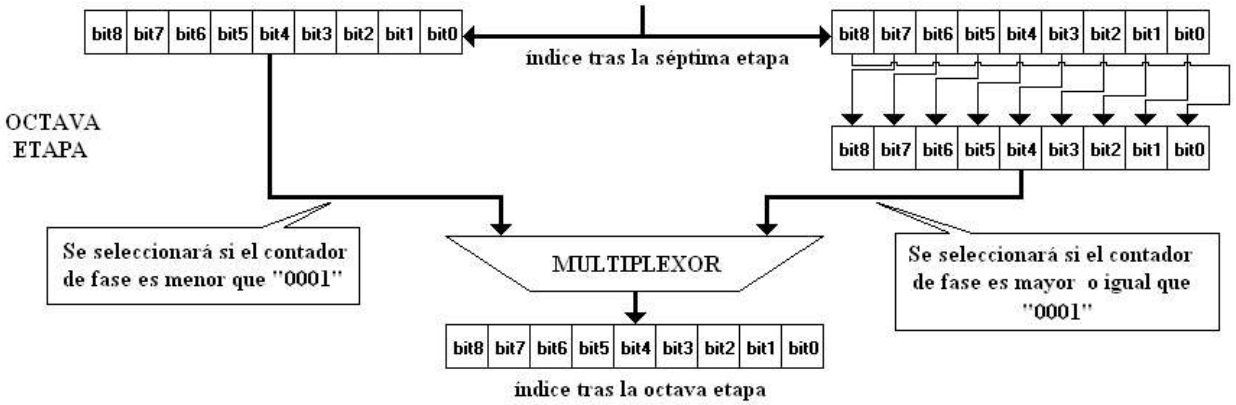
En la quinta etapa se traslada el bit de la posición 5 del índice tras la cuarta etapa a la posición 0 del índice de la quinta etapa. El bit de la posición 0 del índice tras la cuarta etapa se traslada a la posición 1 del índice de la quinta etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El bit de la posición 3 se traslada a la posición 4. El bit de la posición 4 se traslada a la posición 5. El resto de los bits mantienen su posición.



En la sexta etapa se traslada el bit de la posición 6 del índice tras la quinta etapa a la posición 0 del índice de la sexta etapa. El bit de la posición 0 del índice tras la quinta etapa se traslada a la posición 1 del índice de la sexta etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El bit de la posición 3 se traslada a la posición 4. El bit de la posición 4 se traslada a la posición 5. El bit de la posición 5 se traslada a la posición 6. El resto de los bits mantienen su posición.



En la séptima etapa se traslada el bit de la posición 7 del índice tras la sexta etapa a la posición 0 del índice de la séptima etapa. El bit de la posición 0 del índice tras la sexta etapa se traslada a la posición 1 del índice de la séptima etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El bit de la posición 3 se traslada a la posición 4. El bit de la posición 4 se traslada a la posición 5. El bit de la posición 5 se traslada a la posición 6. El bit de la posición 6 se traslada a la posición 7. El resto de los bits mantienen su posición.



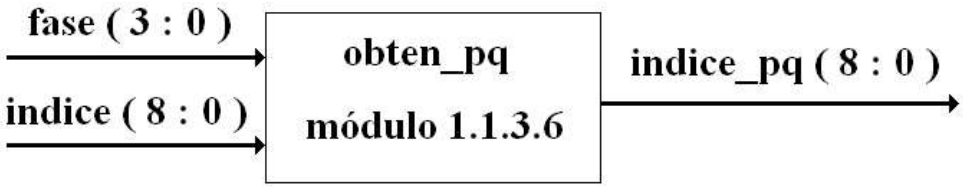
En la octava etapa se traslada el bit de la posición 7 del índice tras la séptima etapa a la posición 0 del índice de la octava etapa. El bit de la posición 0 del índice tras la séptima etapa se traslada a la posición 1 del índice de la octava etapa. El bit de la posición 1 se traslada a la posición 2. El bit de la posición 2 se traslada a la posición 3. El bit de la posición 3 se traslada a la posición 4. El bit de la posición 4 se traslada a la posición 5. El bit de la posición 5 se traslada a la posición 6. El bit de la posición 6 se traslada a la posición 7. El bit de la posición 7 se traslada a la posición 8. El índice tras la octava etapa es el índice "indice_pq", que proporciona el índice p en la memoria principal de la primera variable de la mariposa y el índice q en la memoria principal de la segunda variable de la mariposa.

INTERFAZ DEL MÓDULO:

INDICE: Vector de bits que se modificará para obtener el índice p o el índice q en el acceso a la variable de la mariposa almacenada en la memoria principal. El acceso al índice p se produce cuando el bit menos significativo del índice de entrada al módulo es '0' lógico. El acceso al índice q se produce cuando el bit menos significativo del índice de entrada al módulo es '1' lógico. Los 8 bits más significativos del índice de entrada al módulo lo conforman los 7 bits menos significativos del contador de índice utilizado en los algoritmos de reconocimiento.

FASE: Valor del contador de fase utilizado en los algoritmos de reconocimiento. Según el valor del contador de fase se producirán o no desplazamientos en la posición de los bits del índice de entrada. Los desplazamientos se producen en un proceso de 8 etapas. En cada etapa se produce un desplazamiento particular en el orden de los bits. El contador de fase activará o no los desplazamientos en cada etapa. El vector de bits de salida de cada etapa será el vector de bits de entrada en la siguiente. El vector de bits de entrada en la primera etapa es la entrada "indice" de este módulo.

INDICE_PQ: Valor del índice p o del índice q. El índice p se corresponde con la dirección de la memoria principal donde está guardada la primera variable de la mariposa. El índice q se corresponde con la dirección de la memoria principal donde está guardada la segunda variable de la mariposa. Se accede a la parte real de la variable si el bit más significativo de la dirección de la memoria principal es '0' lógico, se accede a la parte imaginaria si el bit más significativo de la dirección es '1' lógico. Los nueve bits menos significativos de la dirección de la memoria principal lo conforman los nueve bits del vector de bits "indice_pq". Externamente se decidirá si se quiere acceder al índice p (primera variable de la mariposa) o al índice q (segunda variable de la mariposa). Si se decide por el índice q, se activará con un valor '1' lógico el bit menos significativo del vector de entrada "indice", tomando un valor '0' lógico en caso contrario.



1.1.3.7	obten_r.vhd

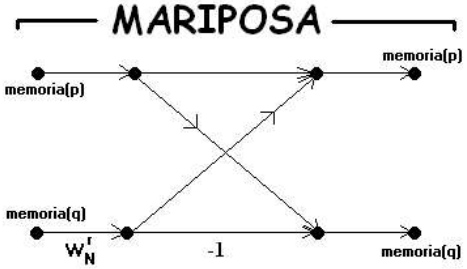
```
ENTITY obten_r IS
  PORT(
    iteracion: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    fase: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    indice_r: OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
END obten_r;
```

El componente "obten_r" proporcionará el índice r del factor de fase por el que se multiplicará la segunda variable de la mariposa.

El cálculo básico que se realiza para hallar la Transformada Rápida de Fourier consiste en coger dos variables complejas, 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). El índice p se corresponde con la dirección de la memoria principal donde está guardada la primera variable de la mariposa. El índice q se corresponde con la dirección de la memoria principal donde está guardada la segunda variable de la mariposa.

FACTOR DE FASE

$$W_N^r = e^{-j\,2\,\pi\,r\,/\,N}$$

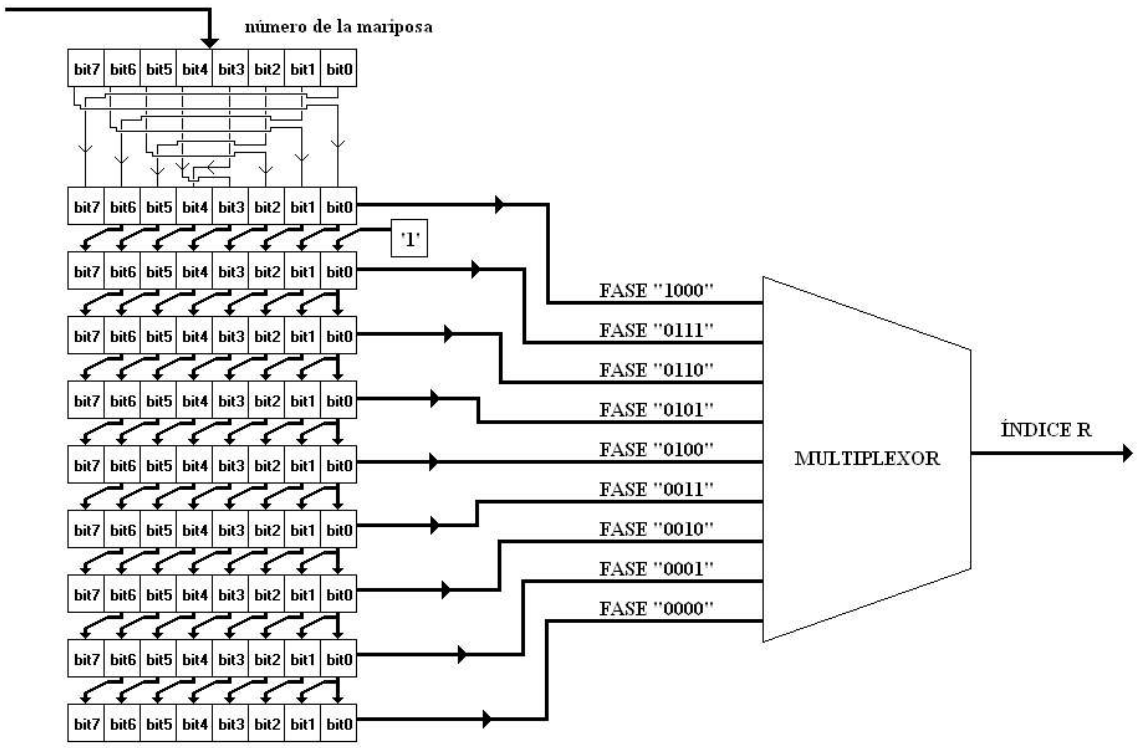


En los cálculos de cada mariposa se realiza una multiplicación por un factor de fase. El factor de fase es un número complejo de módulo unidad que realiza una modificación en la fase del dato al que se le realiza la multiplicación. Se modifica la fase de la segunda de las variables de la mariposa en un múltiplo entero negado indicado por el índice r de la fase $2\,\pi\,/\,512$. 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.

INTERFAZ DEL MÓDULO:

ITERACION: 8 bits menos significativos del contador de índice. Indica el índice de la mariposa que se está calculando (0 – 255). Este valor se procesará según la etapa en la que se encuentre el algoritmo de cálculo de la Transformada Rápida de Fourier para formar el valor de la salida del módulo "indice_r".
FASE: 4 bits menos significativos del contador de fase. Indica el número de la etapa actual en el algoritmo de cálculo de la Transformada Rápida de Fourier (0 – 8). Según el número de la etapa se realizará un procesamiento distinto sobre el número de mariposa para obtener el índice r.
INDICE_R: Valor del coeficiente r. El coeficiente r marca la variación en la fase que provocará el factor de fase sobre la segunda de las variables de los cálculos de la mariposa. La segunda de las variables de la mariposa se multiplica por el factor de fase $e^{-j 2 \pi (\text{índice r}) / 512}$.

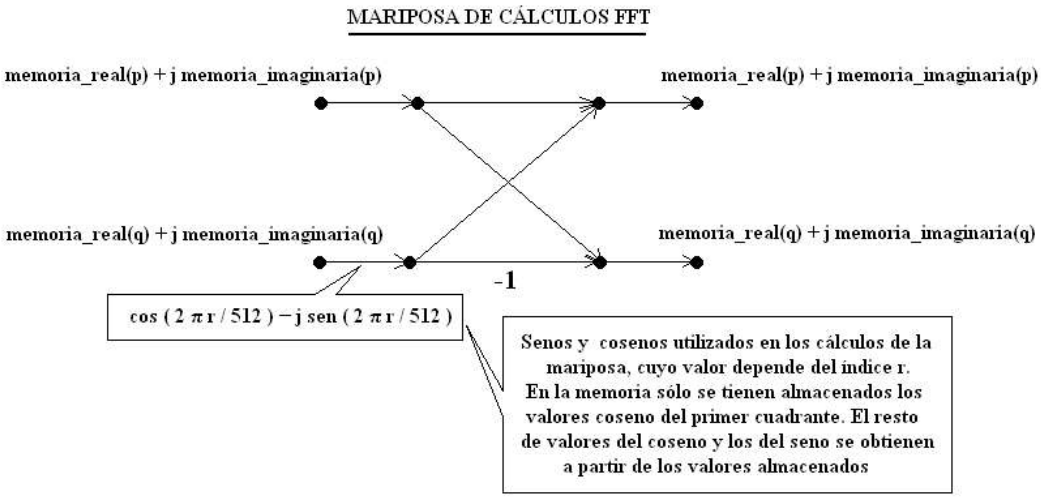


1.1.3.8 calcula_direccion_wrn.vhd

```
ENTITY calcula_direccion_wrn IS
  PORT(
    indice_r: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_imaginario: IN STD_LOGIC;
    direccion_w_r_N: OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
    anula_w_r_N: OUT STD_LOGIC
  );
END calcula_direccion_wrn;
```

El componente "calcula_direccion_wrn" proporcionará la dirección de la memoria "memoria_coseno" desde la que se accederá al valor del coseno o del seno utilizados en los cálculos de la mariposa del algoritmo de la Transformada Rápida de Fourier.

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 $\pi/2$. Fueron almacenados en la memoria "memoria_coseno" para su uso posterior. El resto de valores del coseno entre $\pi/2$ y 2π no es necesario guardarlos , ya que se obtienen a partir de los anteriores. Estos valores se utilizarán para aproximar la operación coseno. También se utilizan estos valores para aproximar los valores del seno desde la fase 0 a 2π. Para acceder al valor del seno o del coseno, lo único que hay que hacer es averiguar cuál es la dirección de memoria donde está el valor correspondiente. El módulo "calcula_direccion_wrn" es el encargado de proporcionar la dirección de lectura de la memoria donde leer el valor del seno o del coseno de la fase oportuna.



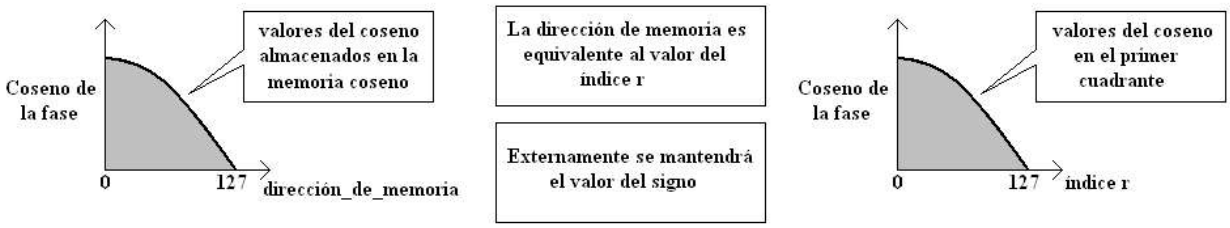
En los cálculos de la mariposa se multiplica la segunda variable compleja por un factor de escala. El factor de escala se corresponde con el siguiente valor:

$$W = \cos(2 \pi r / 512) - j \operatorname{sen}(2 \pi r / 512)$$

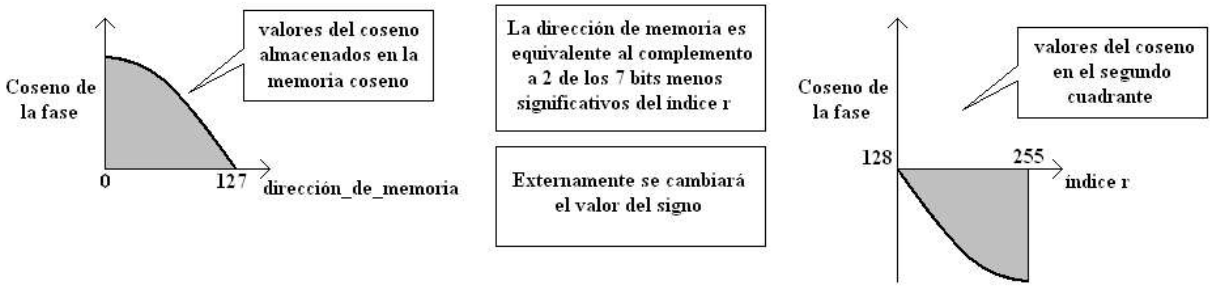
Los valores del seno y del coseno se obtienen a partir de los valores almacenados en la memoria "memoria_coseno". En esta memoria se encuentran almacenados los valores del coseno siguientes:

$$\cos(2 \pi i / 512) \quad \text{donde} \quad i = 0 , 1 , 2 , \ldots , 127$$

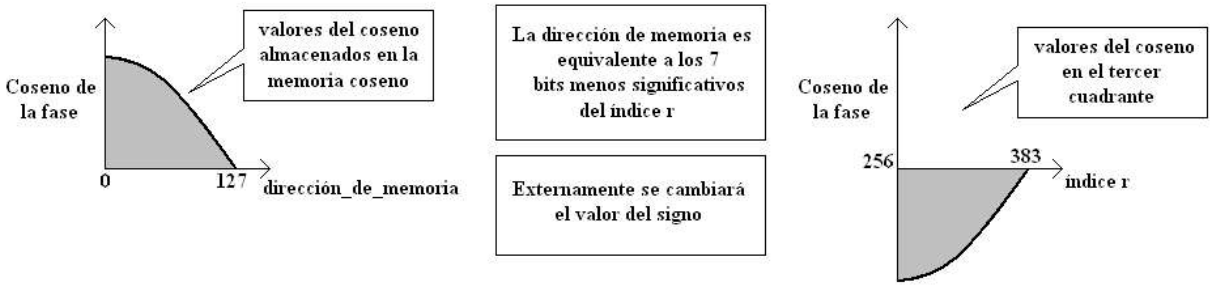
Si necesito el valor del coseno de la fase ($2 \pi r / N$) y el índice r es tal que la fase está en el primer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el índice r.



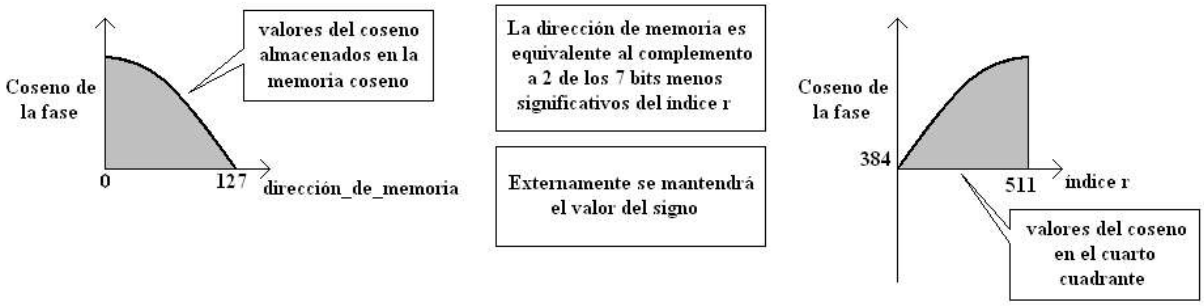
Si necesito el valor del coseno de la fase ($2 \pi r / N$) y el índice r es tal que la fase está en el segundo cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r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



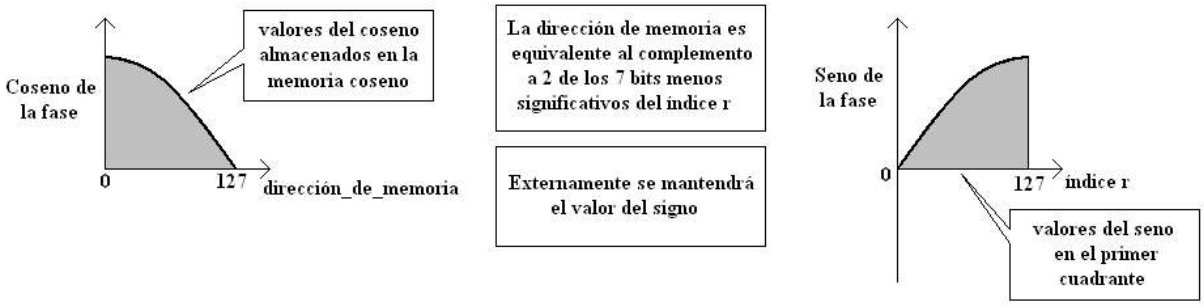
Si necesito el valor del coseno de la fase ($2 \pi r / N$) y el índice r es tal que la fase está en el tercer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el valor de los 7 bits menos significativos del índice r. Fuera del módulo se cambiará el signo del valor obtenido.



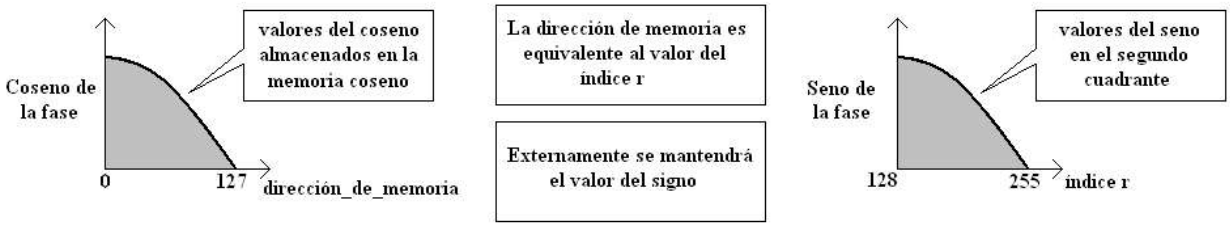
Si necesito el valor del coseno de la fase (2 π r / N) y el índice r es tal que la fase está en el cuarto cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del coseno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir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



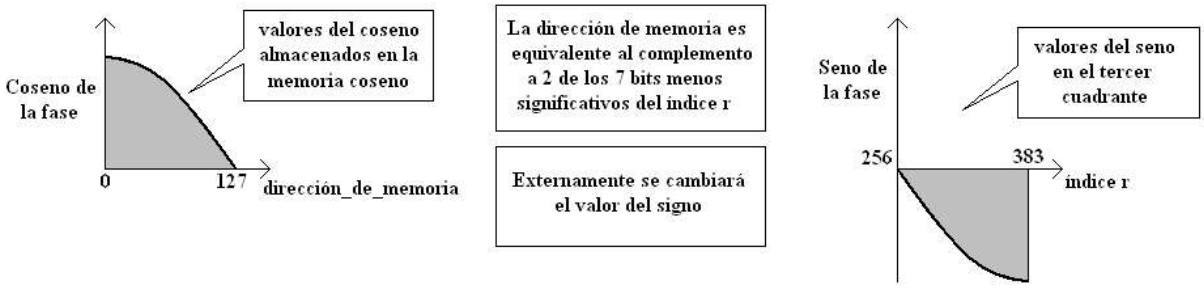
Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el primer cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el valor que resulta de realizar el complemento a dos a los 7 bits menos significativos del índice r (invertir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



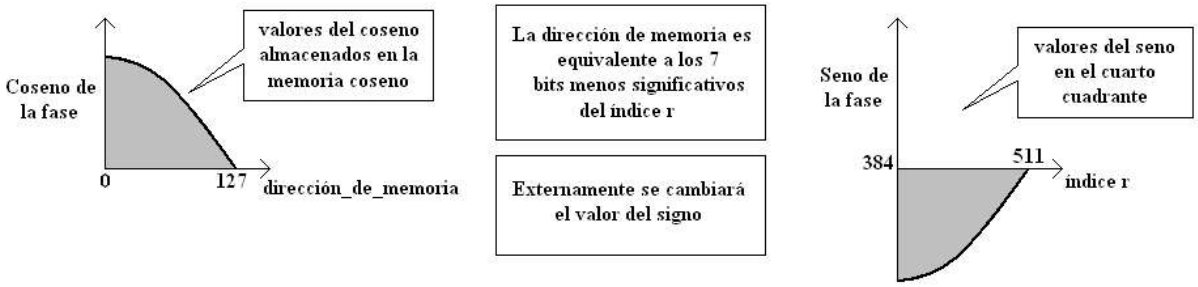
Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el segundo cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno son los 7 bits menos significativos del índice r.



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). Fuera del módulo se cambiará el signo del valor obtenido.



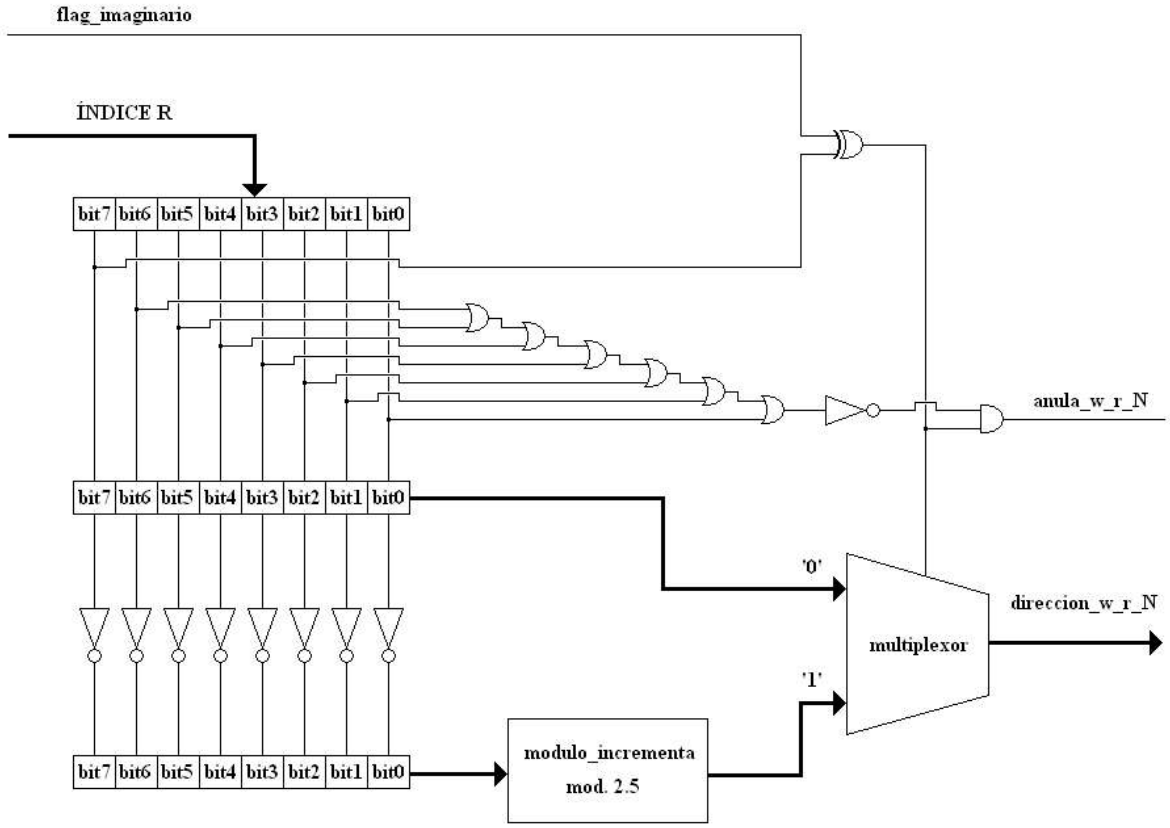
Si necesito el valor del seno de la fase (2 π r / N) y el índice r es tal que la fase está en el cuarto cuadrante, la dirección de la memoria coseno desde donde se leerá el valor del seno es el valor del índice r. Fuera del módulo se cambiará el signo del valor obtenido.



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. 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 estos casos se activa la señal de salida "anula_w_r_N" con un valor '1' lógico, lo que provocará la anulación del valor del seno y del coseno en la operación de multiplicación del factor de fase en la mariposa de cálculos FFT.

INTERFAZ DEL MÓDULO:

INDICE_R: Índice r de la fase del coseno y del seno (sen((2 π r / N) y cos((2 π r / N)). Dependiendo del valor del índice r y del flag "flag_imaginario" se calculará la dirección de acceso a la memoria coseno desde donde se leerá el valor del seno y del coseno.
FLAG_IMAGINARIO: Si el flag "flag_imaginario" se encuentra activo con un valor '1' lógico, se calculará el valor del seno (sen((2 π r / N)). Si el flag "flag_imaginario" se encuentra activo con un valor '0' lógico, se calculará el valor del coseno (cos((2 π r / N)).
DIRECCION_W_R_N: Valor del bus de direcciones de la memoria "memoria_coseno" desde donde se leerá el valor del seno o del coseno implicado en el cálculo de la mariposa FFT. Fuera del módulo se decidirá el signo del seno o del coseno.
ANULA_W_R_N: La activación de esta señal con un valor '1' lógico provocará la anulación del valor del seno y del coseno en la operación de multiplicación del factor de fase en la mariposa de cálculos FFT. 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. 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.



1.1.3.9 buffer_vocal.vhd

```
ENTITY buffer_vocal IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    vocal_nueva: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
    nueva_vocal: IN STD_LOGIC;
    flag_incrementa: IN STD_LOGIC;
    flag_carga: IN STD_LOGIC;
    flag_resetea: IN STD_LOGIC;
    vocales_global: OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
    interruptor_aprende: OUT STD_LOGIC
  );
END buffer_vocal;
```

Este módulo desecha una posible perturbación que provoque una detección incorrecta de una vocal en un instante determinado.

La ejecución completa de todos los módulos del componente “recnew” activó el flag almacenado en el registro “RegistroFlagVocalDetectada” con un '1' lógico si la energía de la señal vocálica superó el umbral. Se considera en este caso que se ha reconocido la vocal cuyo código se encuentra almacenado en el registro “RegistroVocalDetectada”: el código “000” si se ha reconocido la vocal 'A', el código “001” si se ha reconocido la vocal 'E', el código “010” si se ha reconocido la vocal 'I', el código “011” si se ha reconocido la vocal 'O' y el código “100” si se ha reconocido la vocal 'U'.

Para evitar que una perturbación en la señal de audio pueda trasladarse al proceso de reconocimiento provocando que se active la señal de pronunciación de una vocal que en realidad no ha sido pronunciada, se propone una especie de media en el proceso de reconocimiento final de las vocales. En el proceso final se decidirá que se ha pronunciado una vocal si por lo menos se ha reconocido su pronunciación en dos instantes cualquiera de tres instantes consecutivos. Ejemplo: Si en un determinado instante se reconoce la vocal 'A', en el siguiente se reconoce la vocal 'E' y en el siguiente instante se reconoce la vocal 'A' se considera que se ha pronunciado la vocal 'A'. El objetivo de este componente es proponer como error el reconocimiento puntual de una determinada vocal, si sólo se ha producido el reconocimiento en un instante puntual y en los instantes cercanos se ha reconocido otra vocal. Al proponerse como error, debe desecharse esa vocal como vocal reconocida y debe proponerse como vocal pronunciada la vocal que mayoritariamente se ha reconocido.

Para decidir que se ha pronunciado una vocal si ha sido reconocida en al menos dos instantes cualquiera de 3 instantes consecutivos hacen falta una serie de contadores:

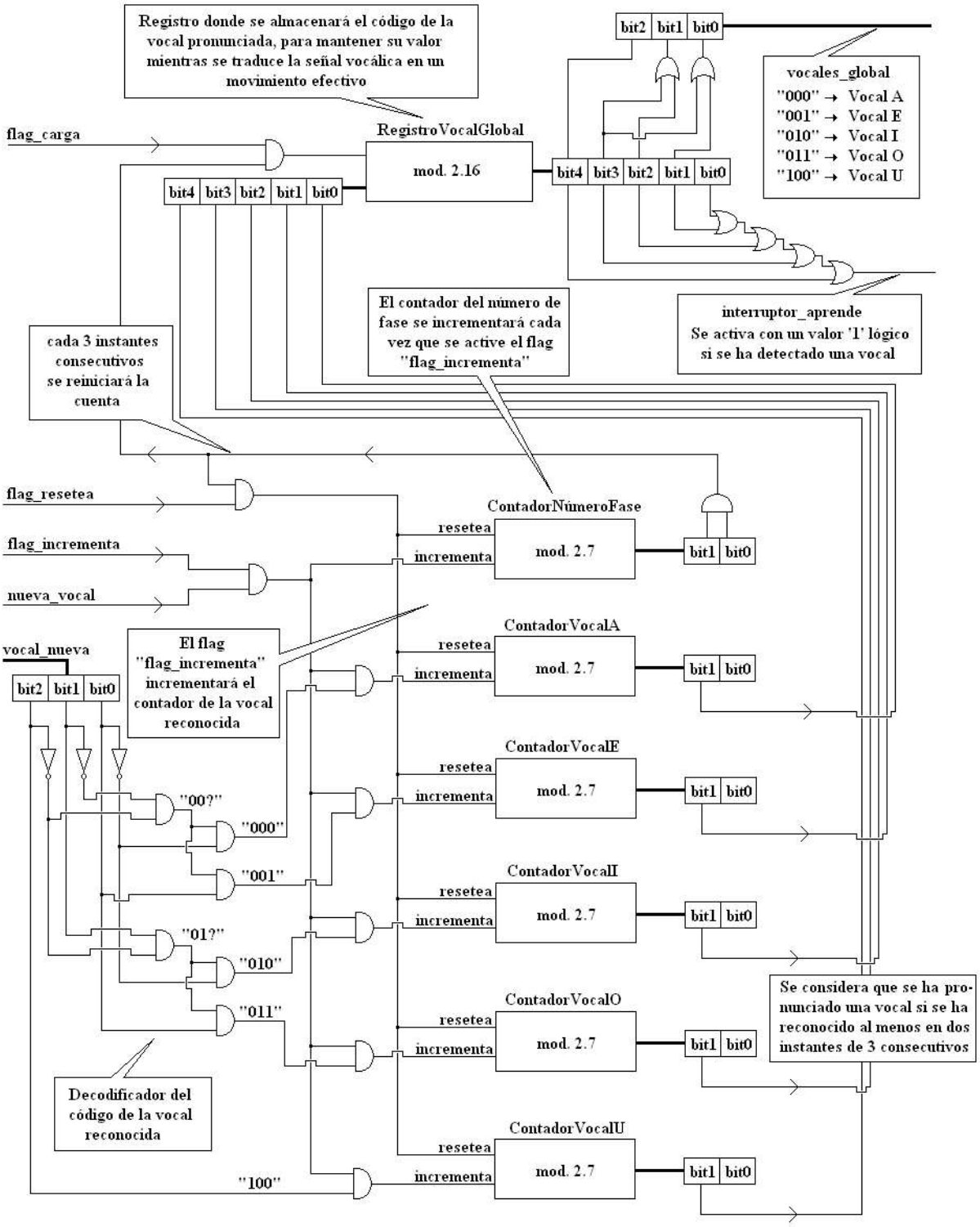
- Contador “ContadorNumeroFase”: Este contador lleva la cuenta del número de instantes que han transcurrido a la espera de los incrementos de los contadores de cada una de las vocales. Se incrementa cada vez que se activa el flag “flag_incrementa”. Como debe contar hasta 3 instantes consecutivos, reseteará todos los contadores, incluido él mismo, para iniciar un nuevo proceso de detección de pronunciación cuando tome el valor “11” binario una vez activado el flag “flag_resetea”.

- Contador “VocalA”: Se incrementa si se ha reconocido la vocal 'A' cuando el flag “flag_incrementa” está activo. Se decide que se ha pronunciado la vocal 'A' si en al menos 2 instantes de los 3 se ha reconocido la vocal 'A'. En el instante 3 se activará el código de salida de la vocal 'A' para indicar que se ha pronunciado la vocal 'A' cuando se active el flag “flag_carga” si el contador de la vocal 'A' tiene al menos el valor 2.
- Contador “VocalE”: Se incrementa si se ha reconocido la vocal 'E' cuando el flag “flag_incrementa” está activo. Se decide que se ha pronunciado la vocal 'E' si en al menos 2 instantes de los 3 se ha reconocido la vocal 'E'. En el instante 3 se activará el código de salida de la vocal 'E' para indicar que se ha pronunciado la vocal 'E' cuando se active el flag “flag_carga” si el contador de la vocal 'E' tiene al menos el valor 2.
- Contador “VocalI”: Se incrementa si se ha reconocido la vocal 'I' cuando el flag “flag_incrementa” está activo. Se decide que se ha pronunciado la vocal 'I' si en al menos 2 instantes de los 3 se ha reconocido la vocal 'I'. En el instante 3 se activará el código de salida de la vocal 'I' para indicar que se ha pronunciado la vocal 'I' cuando se active el flag “flag_carga” si el contador de la vocal 'I' tiene al menos el valor 2.
- Contador “VocalO”: Se incrementa si se ha reconocido la vocal 'O' cuando el flag “flag_incrementa” está activo. Se decide que se ha pronunciado la vocal 'O' si en al menos 2 instantes de los 3 se ha reconocido la vocal 'O'. En el instante 3 se activará el código de salida de la vocal 'O' para indicar que se ha pronunciado la vocal 'O' cuando se active el flag “flag_carga” si el contador de la vocal 'O' tiene al menos el valor 2.
- Contador “VocalU”: Se incrementa si se ha reconocido la vocal 'U' cuando el flag “flag_incrementa” está activo. Se decide que se ha pronunciado la vocal 'U' si en al menos 2 instantes de los 3 se ha reconocido la vocal 'U'. En el instante 3 se activará el código de salida de la vocal 'U' para indicar que se ha pronunciado la vocal 'U' cuando se active el flag “flag_carga” si el contador de la vocal 'U' tiene al menos el valor 2.

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.
VOCAL_NUEVA: Código de la vocal reconocida por el componente “recnew” (“000” si 'A', “001” si 'E', “010” si 'I', “011” si 'O' y “100” si 'U').
NUEVA_VOCAL: Se activa esta señal si la señal de audio ha superado el umbral de energía.
FLAG_INCREMENTA: Se activa durante 200 ns cuando termina de ejecutarse el módulo “011” del componente reconocedor. Utilizada en el funcionamiento interno del dispositivo.
FLAG_CARGA: Se activa durante 200 ns cuando termina de ejecutarse el módulo “100” del componente reconocedor. Utilizada en el funcionamiento interno del dispositivo.
FLAG_RESETEA: Se activa durante 200 ns cuando termina de ejecutarse el módulo “111” del componente reconocedor. Utilizada en el funcionamiento interno del dispositivo.
VOCALES_GLOBAL: Código final de la vocal pronunciada, activo cuando la señal “interruptor_aprende” tiene un valor '1' lógico. Indica al interfaz del emulador del ratón PS/2 que se ha pronunciado la vocal asociada al código de la señal (“000” si 'A', “001” si 'E', “010” si 'I', “011” si 'O' y “100” si 'U').

INTERRUPTOR_APRENDE: Interruptor que se activará para indicar al interfaz del emulador del ratón PS/2 que se ha pronunciado la vocal cuyo código será accesible desde la señal "vocales_global". Pondrá en marcha los mecanismos de traducción de las vocales pronunciadas en movimientos inferidos.



1.1.3.10 memoria_pesos.vhd

```
ENTITY memoria_pesos  IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    carga_en_memoria_pesos: IN STD_LOGIC;
    direccion_de_memoria_pesos: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dato_a_escribir_en_memoria_pesos: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
    dato_leido_de_memoria_pesos: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
    );
END memoria_pesos;
```

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 "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.

A través de un multiplexor, el peso sináptico actual puede 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. La escritura en la memoria se realiza durante la ejecución del módulo número "000" donde se leen los pesos sinápticos desde la memoria FLASH externa AT49F002 de ATMEL presente en la tarjeta XSA y los códigos vocálicos asociados a las neuronas. Estos valores son almacenados en la memoria "memoria_pesos" durante la ejecución del módulo número "000" del componente "recnew". Las operaciones de lectura sólo se realizarán durante la ejecución del módulo número "111", cuando se activa el proceso de reconocimiento de las vocales a través de la red neuronal de mapas autoorganizados. Los pesos de las neuronas son calculados mediante un algoritmo de aprendizaje puesto en funcionamiento

de forma externa en un ordenador a través de la presentación sucesiva de los parámetros cepstrales de las distintas vocales tomadas de forma aleatoria.

INTERFAZ DEL MÓDULO:

RELOJ: Señal de reloj de 200 ns utilizada para el funcionamiento síncrono del dispositivo.

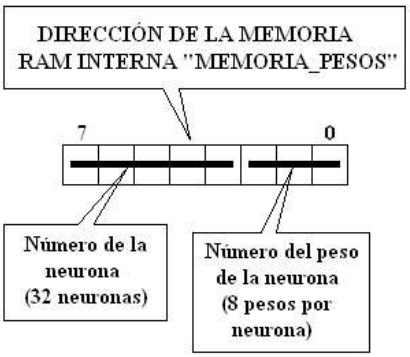
RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en la memoria. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.

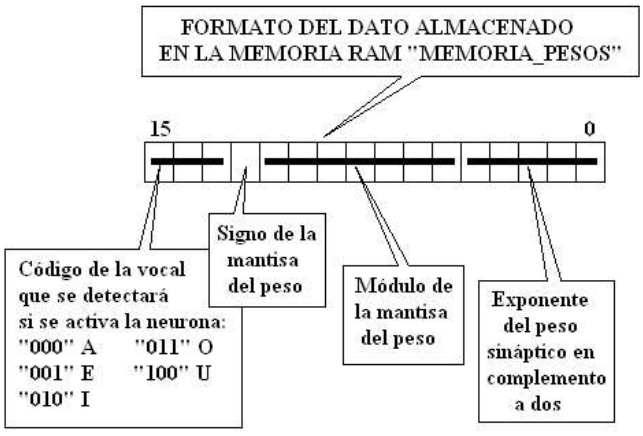
CARGA_EN_MEMORIA_PESOS: Señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones de la memoria del peso sináptico cuyo índice se corresponde con los 3 bits menos significativos del bus de direcciones de la neurona cuyo número se corresponde con los 5 bits más significativos del bus de direcciones. También se almacenará el código de la vocal asignado a la neurona cuyo peso se está almacenando. La habilitación de la escritura se produce cuando esta señal toma el valor '1' lógico. Si la señal de carga no está habilitada, entonces se realiza una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones.

DIRECCION_DE_MEMORIA_PESOS: bus de direcciones de la memoria principal de 8 bits (256 direcciones). Con este bus accedo a las 256 palabras de 15 bits de la memoria tanto para la escritura como para la lectura. Los 3 bits menos significativos del bus de direcciones se utilizan para acceder en lectura o escritura al peso cuyo índice se corresponde con esos 3 bits y los 5 bits más significativos del bus de direcciones se utilizan para acceder al número de la neurona cuyo peso queremos guardar o leer.

DATO_A_ESCRIBIR_EN_MEMORIA_PESOS: bus de datos de entrada que posee la información que se guardará en memoria cuando se active la señal de carga. Los 12 bits menos significativos del dato contienen el peso a guardar en el formato binario flotante utilizado por la unidad aritmética. Los 3 bits más significativos conforman el código de la vocal asociado 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').

DATO_LEIDO_DE_MEMORIA_PESOS: dato leído desde la dirección de memoria proporcionada por el bus de direcciones. Los 12 bits menos significativos del dato contienen el peso leído en el formato binario flotante utilizado por la unidad aritmética. Los 3 bits más significativos conforman el código de la vocal asociado a la neurona.





1.2	modulo_raton.vhd

```
ENTITY modulo_raton IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    cuenta_global: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
    a: IN STD_LOGIC;
    e: IN STD_LOGIC;
    i: IN STD_LOGIC;
    o: IN STD_LOGIC;
    u: IN STD_LOGIC;
    linea_datos: INOUT STD_LOGIC;
    linea_reloj: INOUT STD_LOGIC
  );
END modulo_raton;
```

Este módulo engloba todas las unidades necesarias para implementar un dispositivo ratón PS/2 vocales.

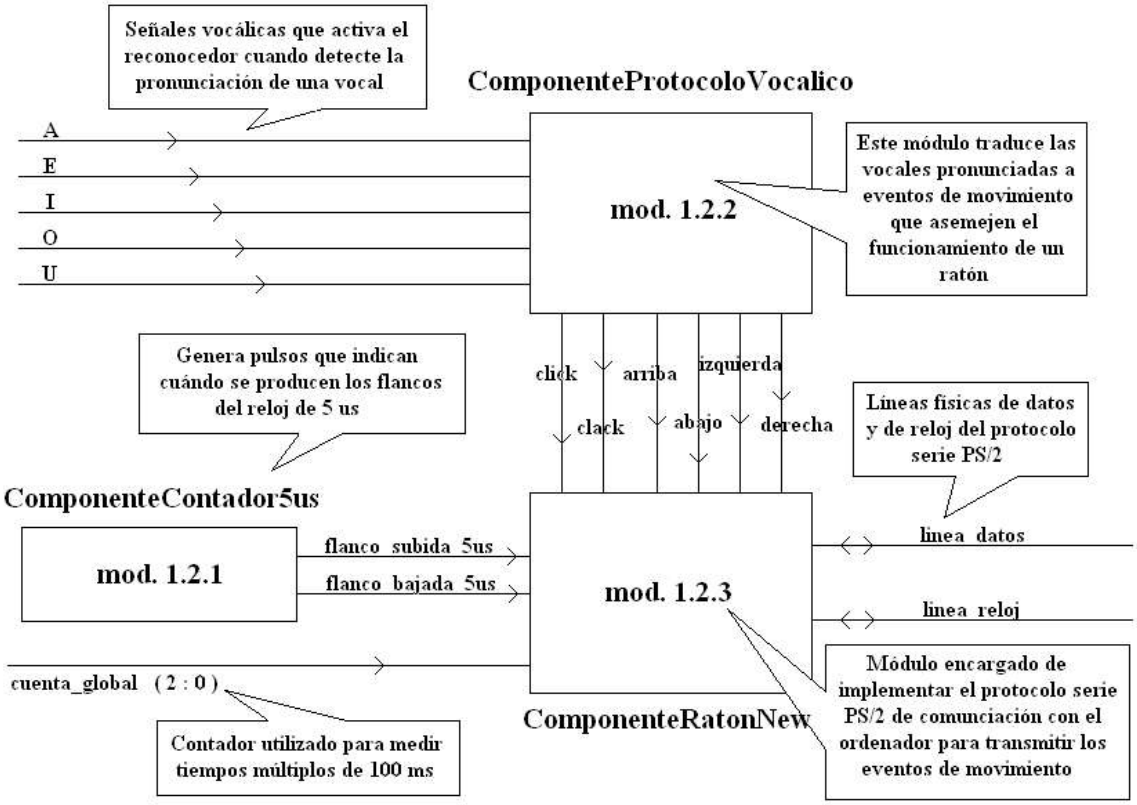
Una vez reconocida la vocal pronunciada, es preciso traducir la señal lógica a eventos de movimiento. El módulo "protocolo_vocalico" realizará esta transcodificación de acuerdo a unas reglas establecidas, obteniendo una señal digital por cada uno de los posibles sentidos de movimiento y por cada uno de los botones. Gracias a este protocolo puedo utilizar la pronunciación de las vocales en el interfaz con el ordenador, ya que el ordenador sólo entendería información relativa a movimientos del ratón y del estado de pulsación de los botones. Esta información será la que se transmita al ordenador a través del protocolo serie PS/2. El módulo "ratón_new" gestionará todo el proceso de comunicación con el host, realizando la misma funcionalidad que ocupara a un dispositivo físico real, desde la ejecución de comandos recibidos desde el ordenador hasta la emisión de paquetes de datos con la información de los movimientos producidos.

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 de periodo utilizada para el funcionamiento síncrono del dispositivo.
CUENTA_GLOBAL: Este es el valor de un contador externo 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. El contador se incrementa aproximadamente cada 100 milisegundos, así puedo crear eventos en múltiplos temporales de 100 milisegundos.
A: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "A".
E: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "E".
I: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "I".
O: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "O".

U: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "U".
LINEA_DATOS: 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.
LINEA_RELOJ: 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.

El funcionamiento de esta unidad se consigue gracias a la interconexión de los siguientes módulo:



- **MÓDULO "CONTADOR_5US" 1.2.1**:

  Este módulo genera un símil de reloj de 5 us de periodo. El flanco de subida del reloj es notificado mediante la activación de la señal "flanco_subida_5us" durante 200 ns, así, otros módulos podrán tener consciencia del evento de subida del símil de reloj. El flanco de bajada del reloj es notificado mediante la activación de la señal "flanco_bajada_5us" durante 200 ns, así, otros módulos podrán tener consciencia del evento de bajada del símil de reloj.

El reloj del sistema es de 200 ns. Este reloj es lo bastante rápido como para realizar las operaciones de reconocimiento de las vocales, pero es muy lento para utilizarlo medir tiempos en la implementación del protocolo físico serie PS/2. En la transmisión de bytes desde el ordenador y hacia el ordenador la línea de datos cambia en unidades de tiempo múltiplos de 80 us. El reloj que se transmitirá en la línea de reloj modificará su nivel cada 40 us. Para medir estos tiempos y sincronizar estas operaciones se hace más sencillo si se trabaja con un reloj de periodo superior. Este es el objetivo de este módulo: Generación de un reloj de 5 us.

No es un reloj en el sentido más estricto, ya que sólo voy a utilizar una señal que avise al sistema de cuándo se produce un flanco de subida en el reloj de 5 us (pasar de cero lógico a uno lógico en el reloj de 5 us imaginario) y una señal que avise al sistema de cuándo se produce un flanco de bajada en el reloj de 5 us (pasar de uno lógico a cero lógico en el reloj de 5 us imaginario). Estas señales serán suficientes para sincronizar eventos en unidades de tiempo múltiplos de 5 us.

- **MÓDULO "PROTOCOLO_VOCÁLICO" 1.2.2:**

Este módulo implementa el conversor vocales-movimientos. El reconocedor de vocales activará una de las señales vocálicas de entrada al módulo si se ha detectado la pronunciación de una de ellas. Un protocolo traducirá la pronunciación de las vocales a eventos de movimiento efectivo del ratón. Es preciso definir el protocolo utilizado, pues el usuario del dispositivo debe conocerlo para poder accionar el ratón.

Una vez en funcionamiento el dispositivo, este módulo 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).

- **MÓDULO "RATÓN_NEW" 1.2.3:**

Este módulo implementa a un dispositivo ratón que se comunica con el ordenador a través del protocolo serie PS/2. El dispositivo ratón se comunica con el ordenador a través del protocolo serie PS/2, usado por muchos ratones y teclados modernos. El ratón PS/2 implementa un protocolo bidireccional serie síncrono. Se utiliza una línea de datos para la transmisión de los bits de datos en forma secuencial y una línea de reloj para la transmisión del reloj que mediante el cual se sincroniza la línea de datos. En otras palabras, los datos se envían un bit tras otro en la línea de datos y se leen cada vez que se pulsa el reloj. El ratón puede enviar datos al host, y el host puede enviar datos al dispositivo, pero el host siempre tiene prioridad sobre el bus y puede inhibir la comunicación desde el ratón en cualquier instante al mantener la línea del reloj en nivel bajo.

Los datos enviados desde el ratón al anfitrión se leen en el flanco de bajada de la señal de reloj (cuando el reloj pasa de un nivel alto a un nivel bajo). Los datos enviados desde el host hasta el ratón se leen en el flanco de subida (cuando el reloj va desde el nivel bajo al nivel alto). El ratón genera siempre la señal de reloj, sin importar la dirección de la comunicación. Si el host quiere enviar datos, debe decirle primero al dispositivo que comience a generar la señal de reloj. La frecuencia de reloj máxima es de 33 KHz y la mayoría de los dispositivos funcionan entre 10 y 20 KHz. Se utiliza una frecuencia en torno a los 15 KHz con el reloj a nivel alto sobre 40μs y a nivel bajo sobre 40μs.

Todos los datos se disponen en bytes y cada byte se envía en una trama de 11-12 bits. Estos bits son:

- 1 bit de inicio. Este bit siempre está a Ø.
- 8 bits de datos, el menos significativo primero.
- 1 bit de paridad (paridad impar).
- 1 bit de parada. Este bit siempre está a 1.
- 1 bit de reconocimiento (sólo en la comunicación host a dispositivo).

El bit de paridad es puesto a uno si hay un número par de unos en los bits de datos y puesto a cero si hay un número impar de unos en los bits de datos. Esto se usa para la detección de errores. Cuando el host esté enviando datos al ratón, un bit de "handshaking" (apretón de manos) es enviado desde el dispositivo para asentir que el paquete fue recibido. Este bit no está presente cuando es el dispositivo el que envía datos al host.

De acuerdo con este protocolo serie se pueden enviar bytes de información desde el host al dispositivo y viceversa. Desde el host se enviarán normalmente comandos para que el dispositivo realice alguna acción. Desde el dispositivo se enviarán las respuestas a los comandos recibidos o la información relativa al movimiento en el caso de que se produjeran eventos en las señales de movimiento de entrada.

Gracias a las señales de movimiento de entrada el dispositivo tendrá constancia de los movimientos que están produciéndose en el ratón en un momento determinado:

- arriba: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia arriba.
- abajo: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia abajo.
- izquierda: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia la izquierda.
- derecha: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia la derecha.
- click: indica cuando está a '1' lógico que se ha pulsado el botón de la izquierda del ratón.
- clack: indica cuando está a '1' lógico que se ha pulsado el botón de la derecha del ratón.

El funcionamiento del dispositivo se consigue mediante la interconexión de una serie de módulos. La funcionalidad individual de cada uno de los módulos permite el desarrollo del protocolo PS/2 a través de la relación de unos módulos con otros.

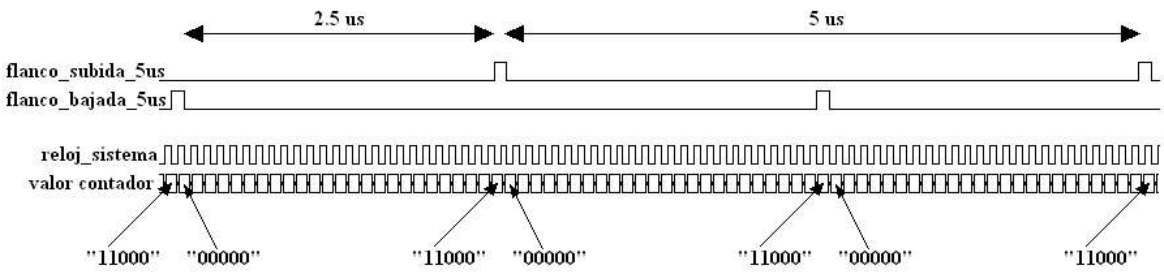
1.2.1 contador_5us.vhd

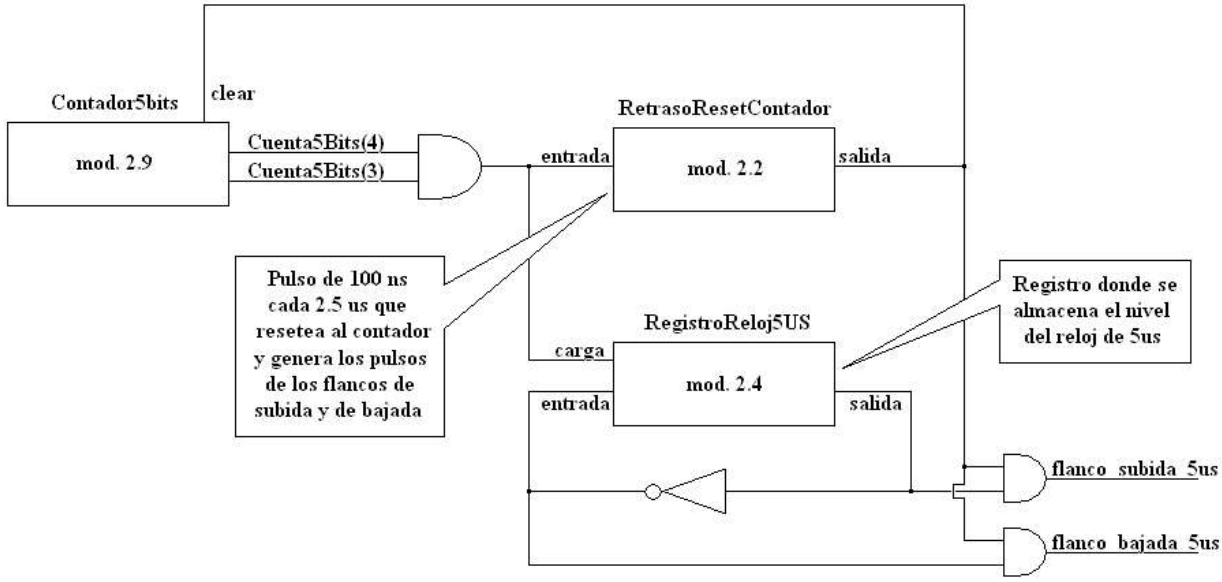
```
ENTITY contador_5us IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    flanco_subida_5us: OUT STD_LOGIC;
    flanco_bajada_5us: OUT STD_LOGIC
  );
END contador_5us;
```

Este módulo genera un símil de reloj de 5 us de periodo. El flanco de subida del reloj es notificado mediante la activación de la señal "flanco_subida_5us" durante 200 ns, así, otros módulos podrán tener consciencia del evento de subida del símil de reloj. El flanco de bajada del reloj es notificado mediante la activación de la señal "flanco_bajada_5us" durante 200 ns, así, otros módulos podrán tener consciencia del evento de bajada del símil de reloj.

El reloj del sistema es de 200 ns. Este reloj es lo bastante rápido como para realizar las operaciones de reconocimiento de las vocales, pero es muy lento para utilizarlo medir tiempos en la implementación del protocolo físico serie PS/2. En la transmisión de bytes desde el ordenador y hacia el ordenador la línea de datos cambia en unidades de tiempo múltiplos de 80 us. El reloj que se transmitirá en la línea de reloj modificará su nivel cada 40 us. Para medir estos tiempos y sincronizar estas operaciones se hace más sencillo si se trabaja con un reloj de periodo superior. Este es el objetivo de este módulo: Generación de un reloj de 5 us.

No es un reloj en el sentido más estricto, ya que sólo voy a utilizar una señal que avise al sistema de cuándo se produce un flanco de subida en el reloj de 5 us (pasar de cero lógico a uno lógico en el reloj de 5 us imaginario) y una señal que avise al sistema de cuándo se produce un flanco de bajada en el reloj de 5 us (pasar de uno lógico a cero lógico en el reloj de 5 us imaginario). Estas señales serán suficientes para sincronizar eventos en unidades de tiempo múltiplos de 5 us.





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.
FLANCO_SUBIDA_5US: Señal activa durante 200 ns para marcar el flanco de subida del reloj de 5 us utilizado en el componente ratón. Esta señal sincroniza los procesos en el ratón y permite trabajar con unos tiempos mayores que el reloj del sistema de 200 ns. Los procesos en el módulo ratón son más lentos que en el reconocedor.
FLANCO_BAJADA_5US: La señal "flanco_bajada_5us" se activa durante 200 ns cada 5 us. Es útil a la hora de sincronizar eventos para que se ejecuten según un reloj de 5 us. Debido a que el reloj del sistema es mucho más rápido que el reloj de emisión del protocolo PS/2 es necesario de un mecanismo para generar las señales con una periodicidad mayor, en este caso de 5 us.

1.2.2 protocolo_vocalico.vhd

```
ENTITY protocolo_vocalico IS
  PORT(
    a: IN STD_LOGIC;
    e: IN STD_LOGIC;
    i: IN STD_LOGIC;
    o: IN STD_LOGIC;
    u: IN STD_LOGIC;
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    arriba: OUT STD_LOGIC;
    derecha: OUT STD_LOGIC;
    abajo: OUT STD_LOGIC;
    izquierda: OUT STD_LOGIC;
    click: OUT STD_LOGIC;
    clack: OUT STD_LOGIC
  );
END protocolo_vocalico;
```

Este módulo implementa el conversor vocales-movimientos.

El reconocedor de vocales activará una de las señales vocálicas de entrada al módulo si se ha detectado la pronunciación de una de ellas. Un protocolo traducirá la pronunciación de las vocales a eventos de movimiento efectivo del ratón. Es preciso definir el protocolo utilizado, pues el usuario del dispositivo debe conocerlo para poder accionar el ratón.

Una vez en funcionamiento el dispositivo, este módulo puede encontrarse realizando dos funcionalidades:

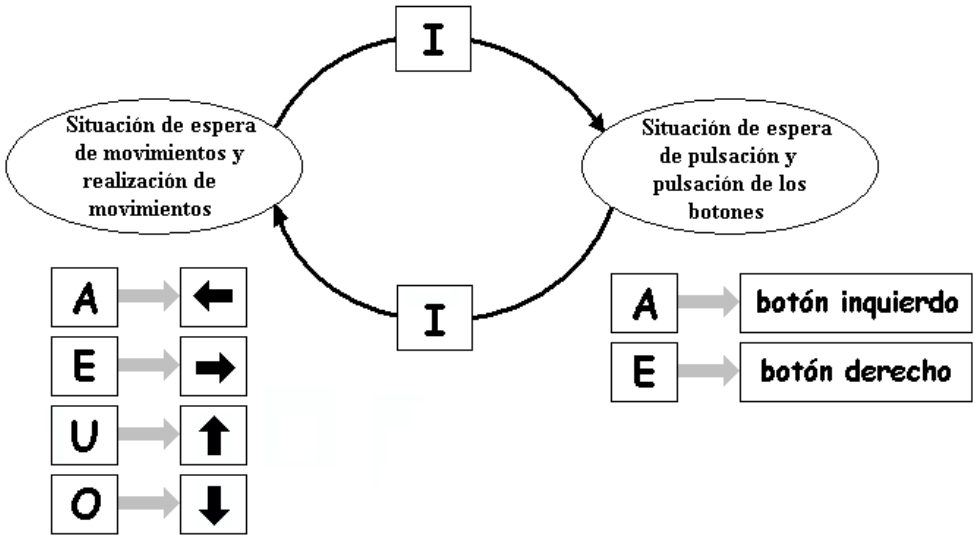
- Situación de espera de movimientos y realización de movimientos: arriba, abajo, izquierda y derecha.
- Situación de espera de pulsación y pulsación de los botones: botón derecho y botón izquierdo.

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).



Una máquina de estados controlará la alternancia entre las dos funcionalidades. Cuando se encuentre en situación de espera de movimientos y realización de movimientos tendrá desactivada la señal "boton_pulsado" con un valor '0' lógico. Si se encuentra en situación de espera de pulsación y pulsación de los botones tendrá activada la señal "boton_pulsado" con un valor '1' lógico. A continuación quedan definidos los posibles estados:

- **ESTADO S0**: Situación de espera de movimientos y realización de movimientos. En este estado, 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" pasará al estado S1 y si no, permanecerá en este estado.

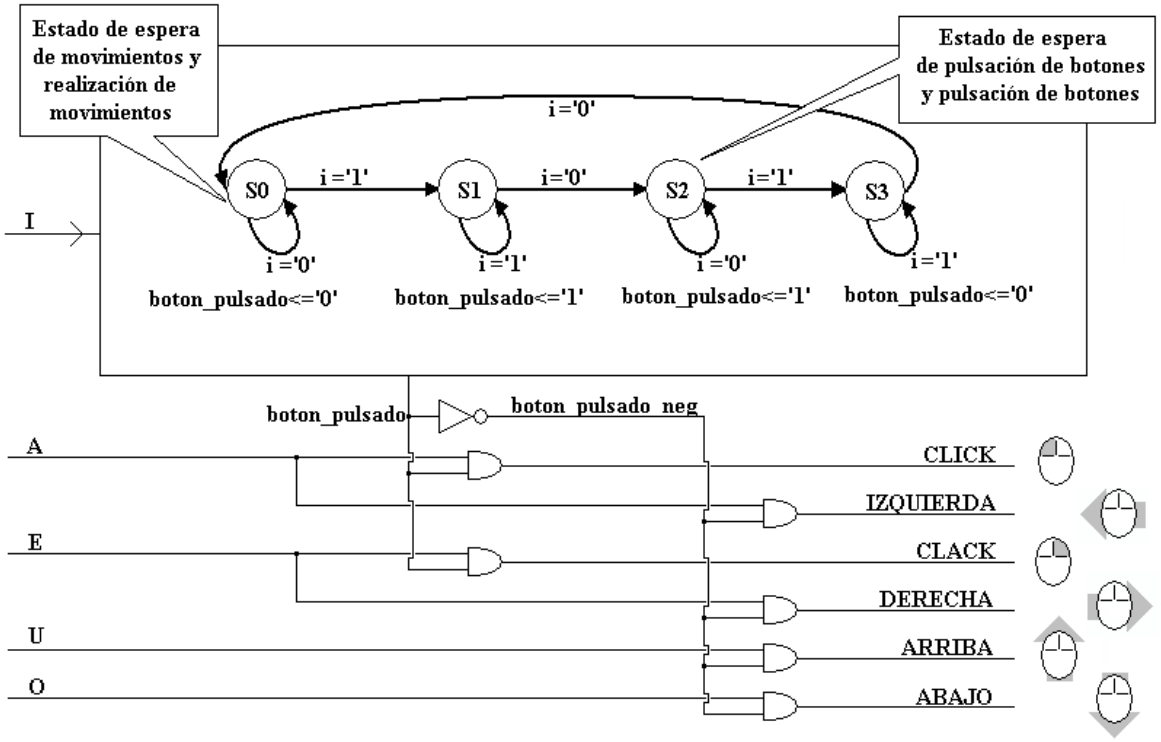
- **ESTADO S1**: Estado de transición en el que permanecerá hasta que deje de pronunciarse la vocal "I". Luego pasará al estado S2.

- **ESTADO S2**: Situación de espera de pulsación y pulsación de los botones. En este estado, 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). Si se pronuncia la vocal "I" pasará al estado S3 y si no, permanecerá en este estado.

- **ESTADO S3**: Estado de transición en el que permanecerá hasta que deje de pronunciarse la vocal "I". Luego pasará al estado S0.

INTERFAZ DEL MÓDULO:

A: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "A".
E: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "E".
I: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "I".
O: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "O".
U: Indica con un valor '1' lógico que se está detectando la pronunciación de la vocal "U".
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 utilizada para el funcionamiento síncrono del dispositivo.
ARRIBA: La pronunciación de una vocal se ha traducido en el movimiento del ratón hacia arriba.
DERECHA: La pronunciación de una vocal se ha traducido en el movimiento del ratón hacia la derecha.
ABAJO: La pronunciación de una vocal se ha traducido en el movimiento del ratón hacia abajo.
IZQUIERDA: La pronunciación de una vocal se ha traducido en el movimiento del ratón hacia la izquierda.
CLICK: La pronunciación de una vocal se ha traducido en la pulsación del botón izquierdo del ratón.
CLACK: La pronunciación de una vocal se ha traducido en la pulsación del botón derecho del ratón.



1.2.3 raton_new.vhd

```
ENTITY raton_new IS
  PORT(
    reloj: IN STD_LOGIC;
    flanco_subida_5us: IN STD_LOGIC;
    flanco_bajada_5us: IN STD_LOGIC;
    cuenta_global: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
    resetz: IN STD_LOGIC;
    arriba: IN STD_LOGIC;
    derecha: IN STD_LOGIC;
    abajo: IN STD_LOGIC;
    izquierda: IN STD_LOGIC;
    click: IN STD_LOGIC;
    clack: IN STD_LOGIC;
    linea_datos: INOUT STD_LOGIC;
    linea_reloj: INOUT STD_LOGIC
  );
END raton_new;
```

Este módulo implementa a un dispositivo ratón que se comunica con el ordenador a través del protocolo serie PS/2.

INTERFAZ DEL MÓDULO:

RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
FLANCO_SUBIDA_5US: Señal activa durante 200 ns para marcar el flanco de subida del reloj de 5 us utilizado en el componente ratón. Esta señal sincroniza los procesos en el ratón y permite trabajar con unos tiempos mayores que el reloj del sistema de 200 ns. Los procesos en el módulo ratón son más lentos que en el reconocedor.
FLANCO_BAJADA_5US: El contador temporal utiliza el flanco de bajada del reloj para sincronizar el incremento del contador. Este contador funciona con el reloj del sistema de 200 ns y la sincronización con el reloj de 5 us se consigue imponiendo que sólo se pueda incrementar en el flanco de bajada del reloj de 5 us. La señal "flanco_bajada_5us" se activa durante 200 ns cada 5 us. Es útil a la hora de sincronizar eventos para que se ejecuten según un reloj de 5 us. Debido a que el reloj del sistema es mucho más rápido que el reloj de emisión del protocolo PS/2 es necesario de un mecanismo para generar las señales con una periodicidad de 5 us.
CUENTA_GLOBAL: Este es el valor de un contador externo 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. El contador se incrementa aproximadamente cada 100 milisegundos, así puedo crear eventos en múltiplos temporales de 100 milisegundos.
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.
ARRIBA: Señal que indica un movimiento del ratón hacia arriba cuando está activada con un valor '1' lógico.
DERECHA: Señal que indica un movimiento del ratón hacia la derecha cuando está activada con un valor '1' lógico.

ABAJO: Señal que indica un movimiento del ratón hacia abajo cuando está activada con un valor '1' lógico.
IZQUIERDA: Señal que indica un movimiento del ratón hacia la izquierda cuando está activada con un valor '1' lógico.
CLICK: Señal que indica que se ha pulsado el botón izquierdo cuando está activada con un valor '1' lógico.
CLACK: Señal que indica que se ha pulsado el botón derecho cuando está activada con un valor '1' lógico.
LINEA_DATOS: 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.
LINEA_RELOJ: 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.

El dispositivo ratón se comunica con el ordenador a través del protocolo serie PS/2, usado por muchos ratones y teclados modernos. El ratón PS/2 implementa un protocolo bidireccional serie síncrono. Se utiliza una línea de datos para la transmisión de los bits de datos en forma secuencial y una línea de reloj para la transmisión del reloj que mediante el cual se sincroniza la línea de datos. En otras palabras, los datos se envían un bit tras otro en la línea de datos y se leen cada vez que se pulsa el reloj. El ratón puede enviar datos al host, y el host puede enviar datos al dispositivo, pero el host siempre tiene prioridad sobre el bus y puede inhibir la comunicación desde el ratón en cualquier instante al mantener la línea del reloj en nivel bajo.

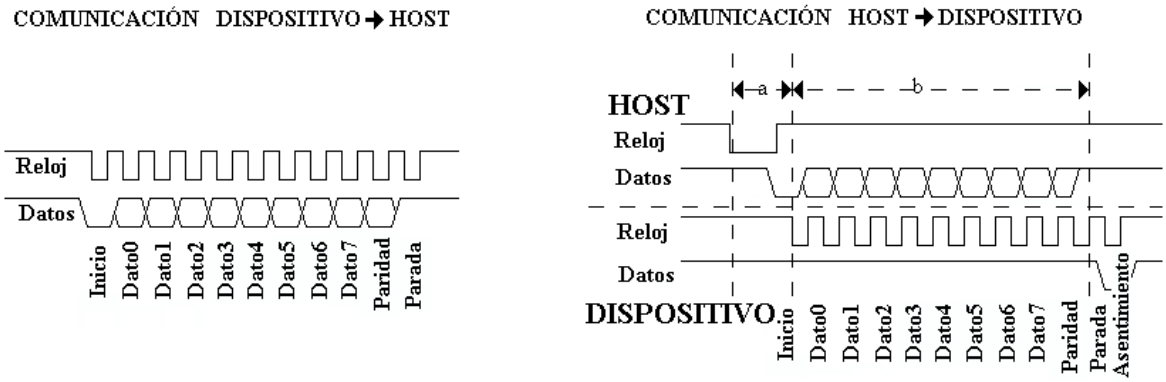
Los datos enviados desde el ratón al anfitrión se leen en el flanco de bajada de la señal de reloj (cuando el reloj pasa de un nivel alto a un nivel bajo). Los datos enviados desde el host hasta el ratón se leen en el flanco de subida (cuando el reloj va desde el nivel bajo al nivel alto). El ratón genera siempre la señal de reloj, sin importar la dirección de la comunicación. Si el host quiere enviar datos, debe decirle primero al dispositivo que comience a generar la señal de reloj. La frecuencia de reloj máxima es de 33 KHz y la mayoría de los dispositivos funcionan entre 10 y 20 KHz. Se utiliza una frecuencia en torno a los 15 KHz con el reloj a nivel alto sobre 40μs y a nivel bajo sobre 40μs.

Todos los datos se disponen en bytes y cada byte se envía en una trama de 11-12 bits. Estos bits son:

- 1 bit de inicio. Este bit siempre está a Ø.
- 8 bits de datos, el menos significativo primero.
- 1 bit de paridad (paridad impar).
- 1 bit de parada. Este bit siempre está a 1.
- 1 bit de reconocimiento (sólo en la comunicación host a dispositivo).

El bit de paridad es puesto a uno si hay un número par de unos en los bits de datos y puesto a cero si hay un número impar de unos en los bits de datos. Esto se usa para la detección de errores. Cuando el host esté enviando datos al ratón, un bit de “handshaking” (apretón de manos) es

enviado desde el dispositivo para asentir que el paquete fue recibido. Este bit no está presente cuando es el dispositivo el que envía datos al host.



De acuerdo con este protocolo serie se pueden enviar bytes de información desde el host al dispositivo y viceversa. Desde el host se enviarán normalmente comandos para que el dispositivo realice alguna acción. Desde el dispositivo se enviarán las respuestas a los comandos recibidos o la información relativa al movimiento en el caso de que se produjeran eventos en las señales de movimiento de entrada.

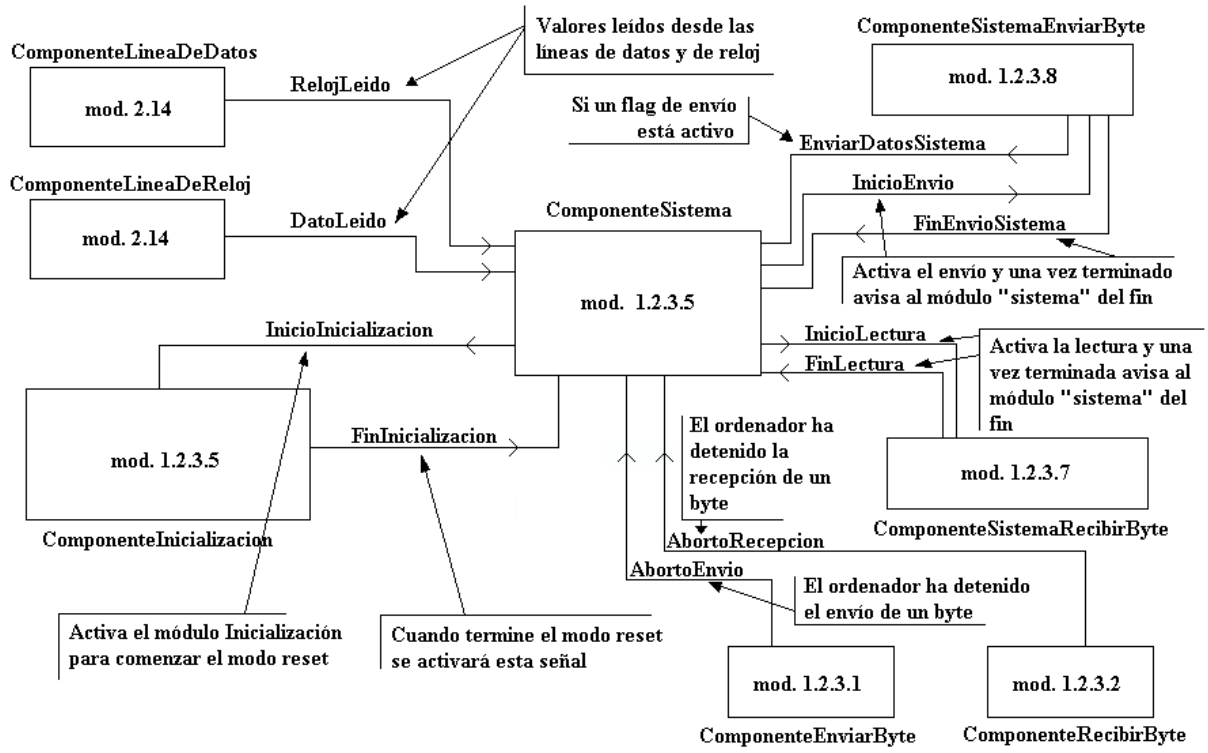
Gracias a las señales de movimiento de entrada el dispositivo tendrá constancia de los movimientos que están produciéndose en el ratón en un momento determinado:

- arriba: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia arriba.
- abajo: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia abajo.
- izquierda: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia la izquierda.
- derecha: cuando está a '1' lógico indica que el puntero del ratón debe moverse hacia la derecha.
- click: indica cuando está a '1' lógico que se ha pulsado el botón de la izquierda del ratón.
- clack: indica cuando está a '1' lógico que se ha pulsado el botón de la derecha del ratón.

El funcionamiento del dispositivo se consigue mediante la interconexión de una serie de módulos. La funcionalidad individual de cada uno de los módulos permite el desarrollo del protocolo PS/2 a través de la relación de unos módulos con otros.

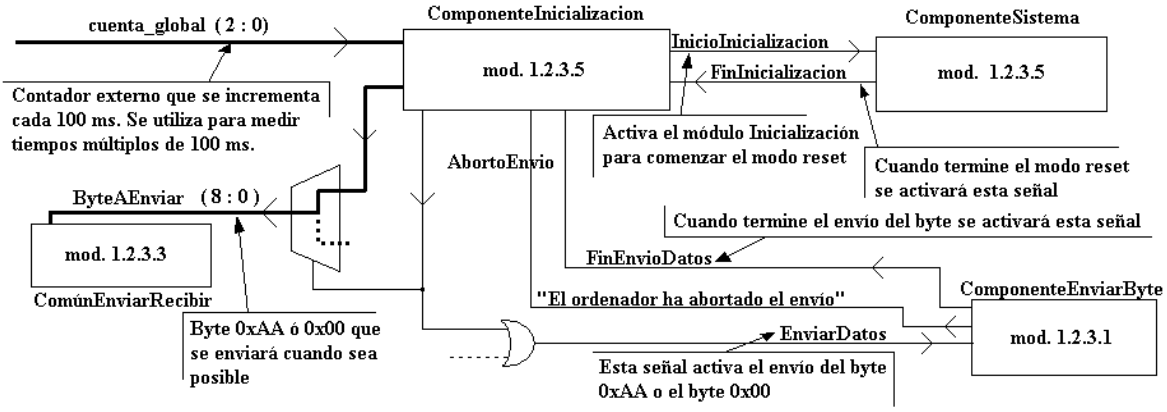
MÓDULO "SISTEMA" 1.2.3.4:
Módulo encargado de gestionar el funcionamiento del sistema-ratón de manera global. En un principio activará el proceso de inicialización mediante el cual el ratón envía una serie de señales al host (0xAA y 0x00) con el fin de que detecte y reconozca al dispositivo. Una vez terminado el proceso de inicialización (MODO RESET) alternará su ocupación entre el de envío de datos y la recepción de datos. El ordenador solicita al dispositivo que active las señales de reloj para poder enviar un byte (comando o dato) llevando la línea de datos a un valor cero lógico y dejando la línea de datos en alta impedancia. Si alguno de los flags de la memoria de datos solicita el

envío de datos (flag de asentimiento, flag del primer registro de respuesta, flag del segundo registro de respuesta o flag del tercer registro de respuesta) estando las líneas de datos y de reloj en alta impedancia, se activará la señal "inicio_envio" que pondrá en marcha el procedimiento de envío del byte correspondiente al flag de mayor prioridad.



MÓDULO "INICIALIZACION" 1.2.3.5:

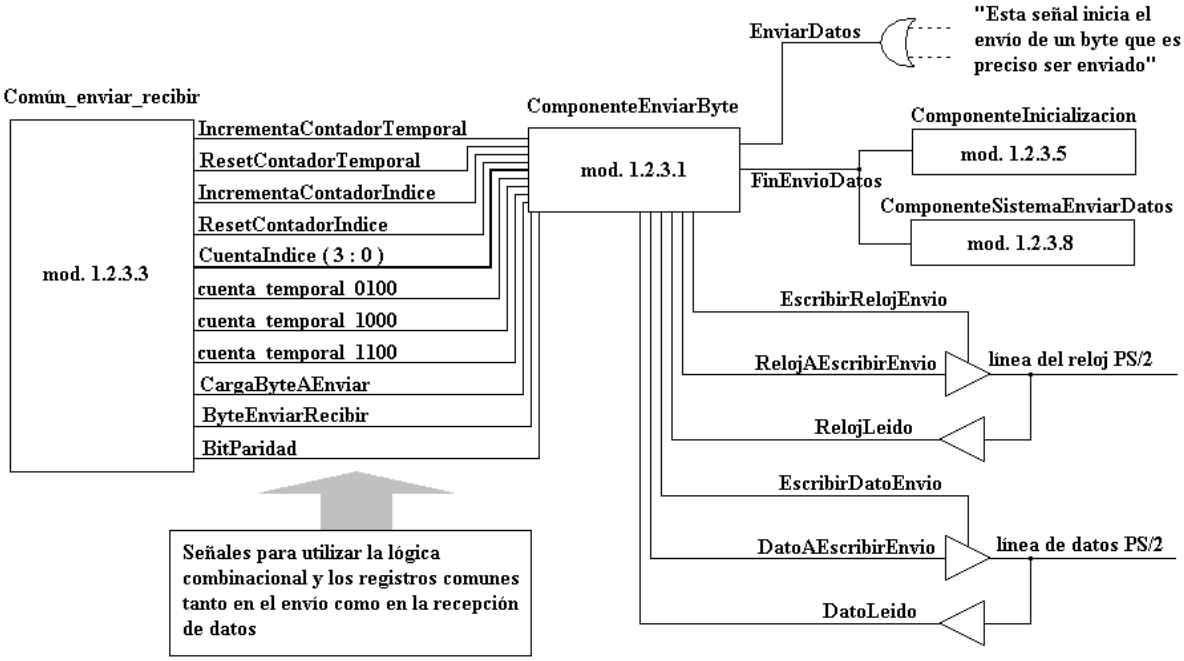
Este módulo implementa el funcionamiento del dispositivo ratón en el "modo reset". Aproximadamente 500 milisegundos después del encendido del dispositivo ratón, se deben transmitir los bytes ØxAA y ØxØØ.



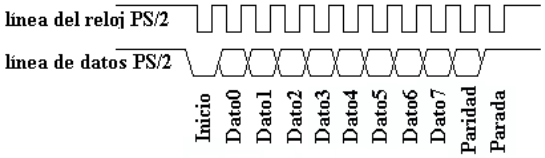
MÓDULO "ENVIAR_BYTE" 1.2.3.1:

Este módulo realiza la implementación física de la parte del dispositivo ratón encargada de la transmisión de datos con el protocolo serie PS/2. A través de la línea de datos se transmitirán los bits de información sincronizados con un reloj que se transmite a través de la línea de reloj. El ratón PS/2 implementa un protocolo bidireccional serie síncrono. En otras palabras, los datos se envían un bit tras otro en la línea de datos y se leen cada vez que se pulsa el reloj. El ratón puede enviar datos al host, y el host puede enviar datos al dispositivo, pero el host siempre tiene prioridad sobre el bus y puede inhibir la comunicación desde el ratón en cualquier instante al mantener la línea del reloj en nivel bajo. El ratón usa un protocolo serie consistente en una trama de 11-bits. Estos bits son:

- 1 bit de inicio. Este bit siempre está a Ø.
- 8 bits de datos, el menos significativo primero.
- 1 bit de paridad (paridad impar).
- 1 bit de parada. Este bit siempre está a 1.



Cuando se active la señal "EnvirDatos" se enviará al ordenador el byte almacenado en el componente "ComunEnviarRecibir" accesible a través del bus "ByteEnviarRecibir



MODULO "RECIBIR_BYTE" 1.2.3.2:

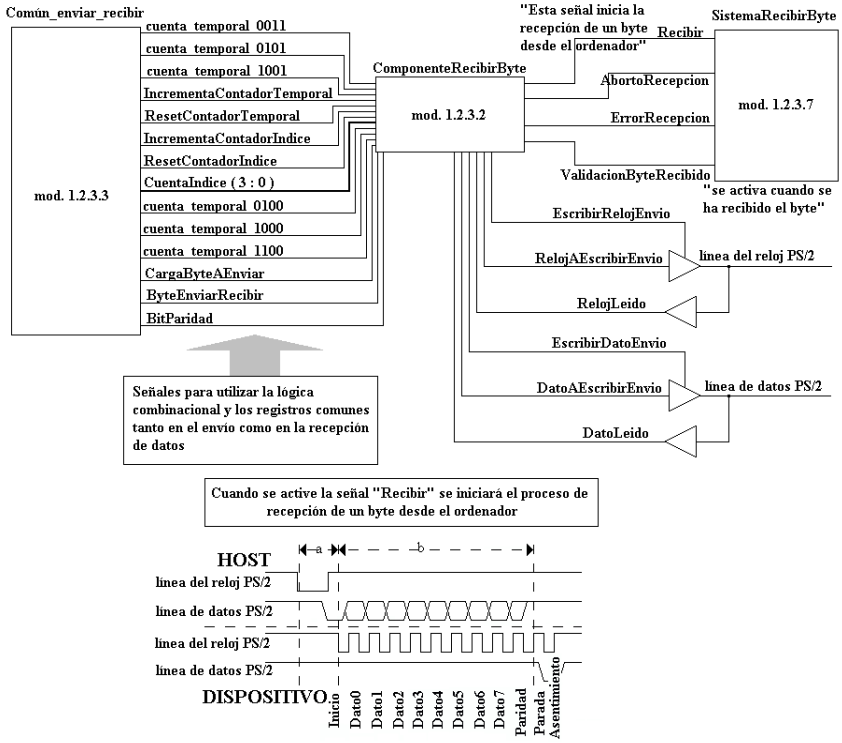
Esta unidad controla la recepción de un byte desde el host de acuerdo con el protocolo PS/2. Este módulo generará las señales adecuadas en las líneas de datos y de reloj para permitir la transmisión de un byte desde el host al dispositivo según el protocolo PS/2. Cuando el host realiza una petición de envío lleva primero la línea de reloj a cero para inhibir cualquier transmisión desde el dispositivo, luego lleva la línea de datos a nivel lógico cero y después libera la línea de reloj para que tome el valor uno lógico. El dispositivo a continuación modifica las líneas de datos y de reloj para leer el dato desde el ordenador:

- Chequea la línea de datos para ver si está a nivel bajo ("Petición de envío"). Si la línea de datos ha sido llevada a nivel bajo por el host l, leer un byte desde el host.
- Esperar hasta que la línea de Reloj esté en nivel alto.
- Leer 8 bits de datos.
- Leer el bit de paridad.
- Leer el bit de parada.
- Salida del bit de asentimiento.

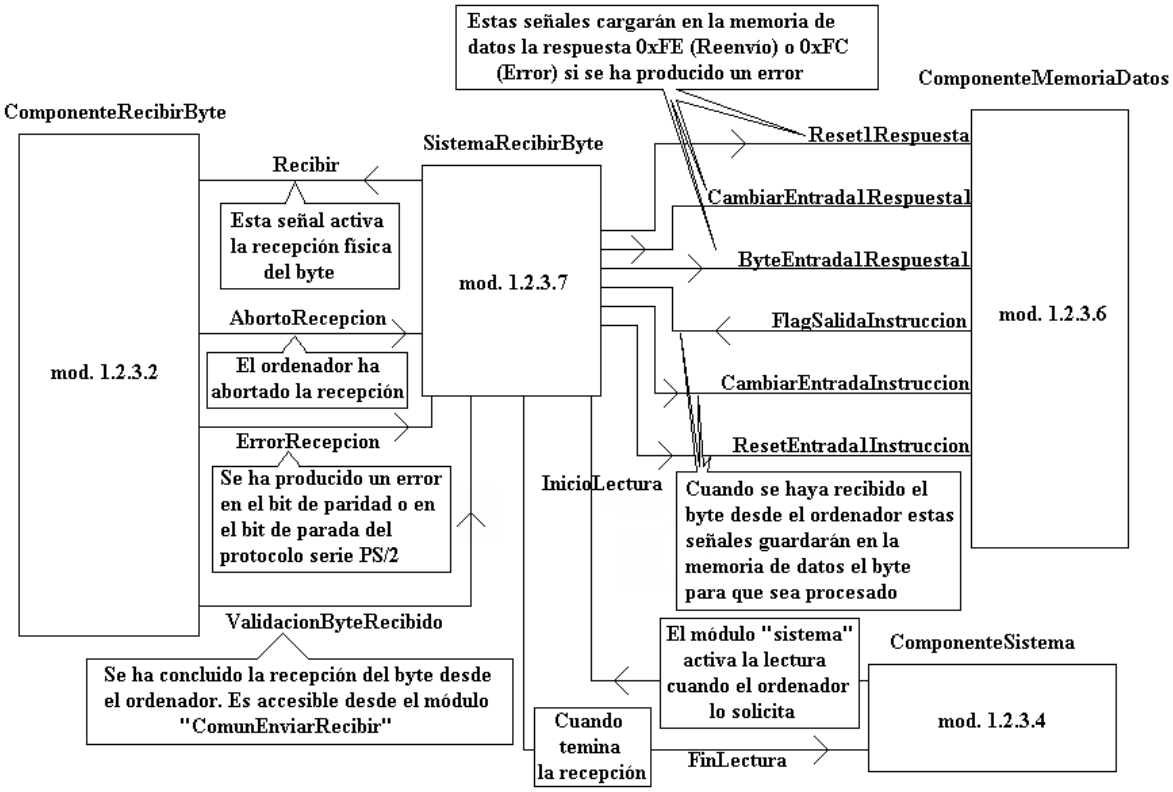
La lectura de cada bit (8 bits de datos, bit de paridad, y bit de parada) es como sigue:

1) Retraso de 20μs
2) Llevar el Reloj a nivel bajo.
3) Retraso de 40μs
4) Liberar el Reloj.
5) Retraso de 20μs.
6) Leer la línea de datos.

El byte recibido será almacenado en el módulo "común_enviar_recibir" hasta que el módulo "sistema_recibir_byte" lo guarde en la memoria como un byte de instrucción que debe ser atendido.



MÓDULO "SISTEMA_RECIBIR_BYTE" 1.2.3.7:
Este módulo activa la recepción de un byte cuando se detecte la intención del ordenador de enviar un byte. Si el ordenador quiere enviar algún byte (ya sea comando o datos) llevará la línea de datos a nivel bajo manteniendo en alta impedancia la línea de reloj del protocolo serie PS/2. Si el ratón está enviado datos al ordenador se interrumpirá el envío. El módulo "sistema" detectará cuándo la línea de datos está a nivel bajo y activará la señal "inicio_lectura" con un valor uno lógico para iniciar el proceso de recepción de un byte desde el ordenador.

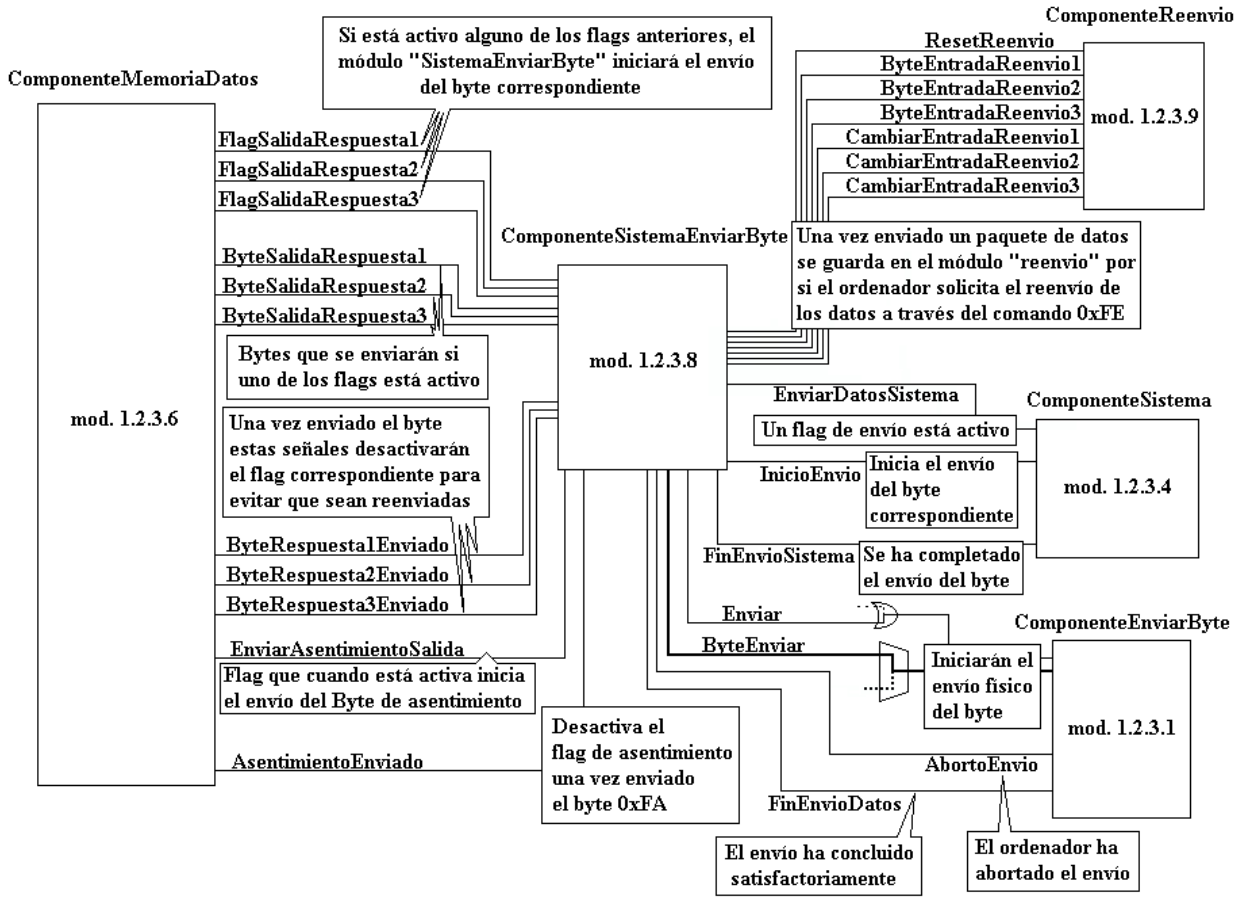


MÓDULO "SISTEMA_ENVIAR_BYTE" 1.2.3.8:
Este módulo inicia el envío del byte de asentimiento o de los bytes de respuesta al ordenador en el caso de que estén pendientes de envío. Si está pendiente por enviar alguno de los bytes anteriores, el flag correspondiente al byte a enviar estará activo con un valor uno lógico. El módulo detectará si está activo alguno de los flags y en caso afirmativo procederá al envío del byte correspondiente. Existe un orden de prioridad en el envío de los bytes:

1. En primer lugar se enviará el byte de asentimiento al comando recibido en el caso de que esté activa la señal "enviar_asentimiento_salida".
2. En segundo lugar se enviará el primer byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_1".
3. En tercer lugar se enviará el segundo byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_2".

4. En cuarto lugar se enviará el tercer byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_3".

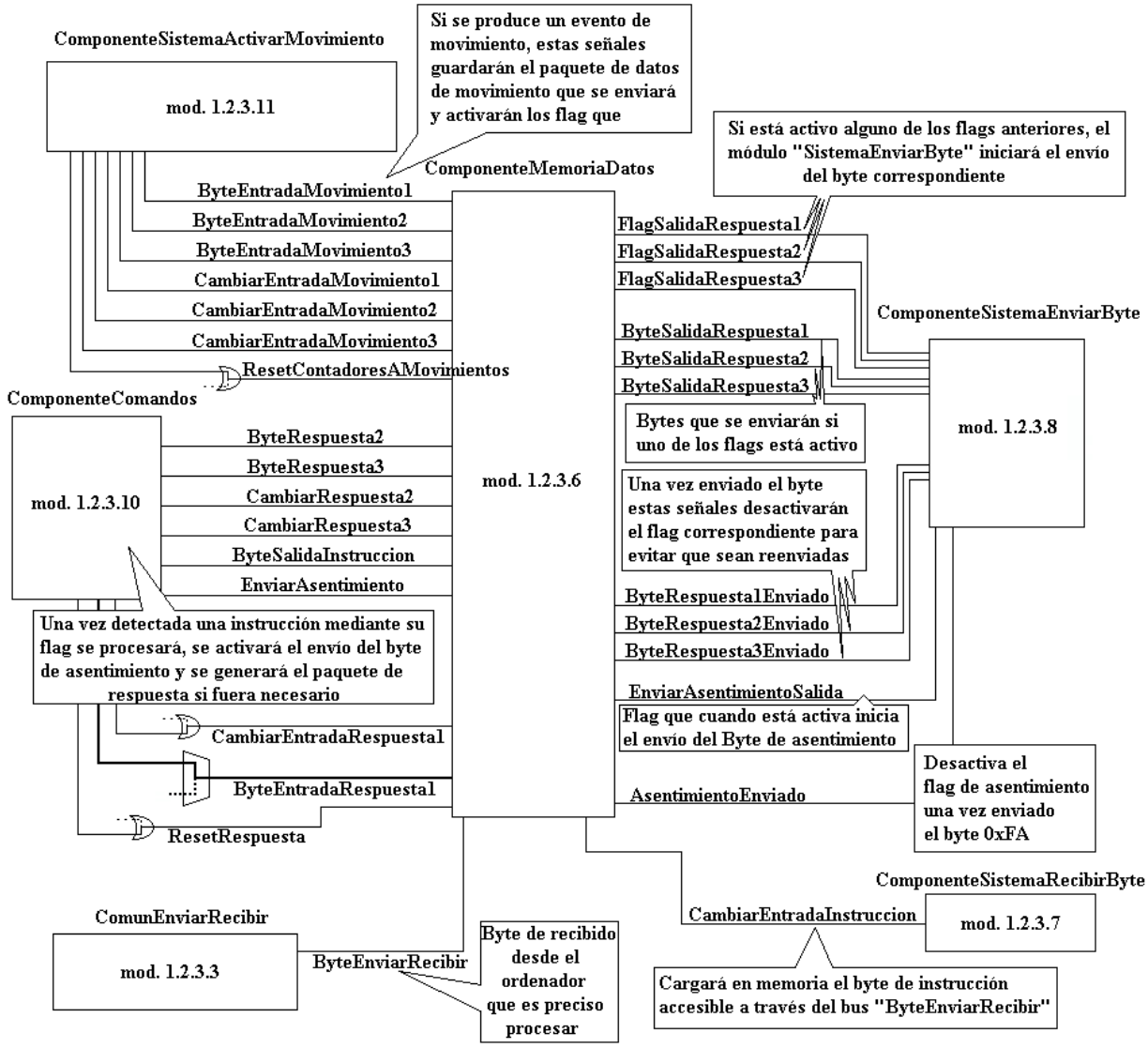
A medida que se van enviado los bytes, se irán desactivando los flags correspondientes, y así hasta que no quede ningún byte por enviar. Una vez enviado un byte se guardará en los registros de reenvío para que puedan ser reenviados al ordenador en el caso de que lo solicite con el comando FE (REENVÍO).



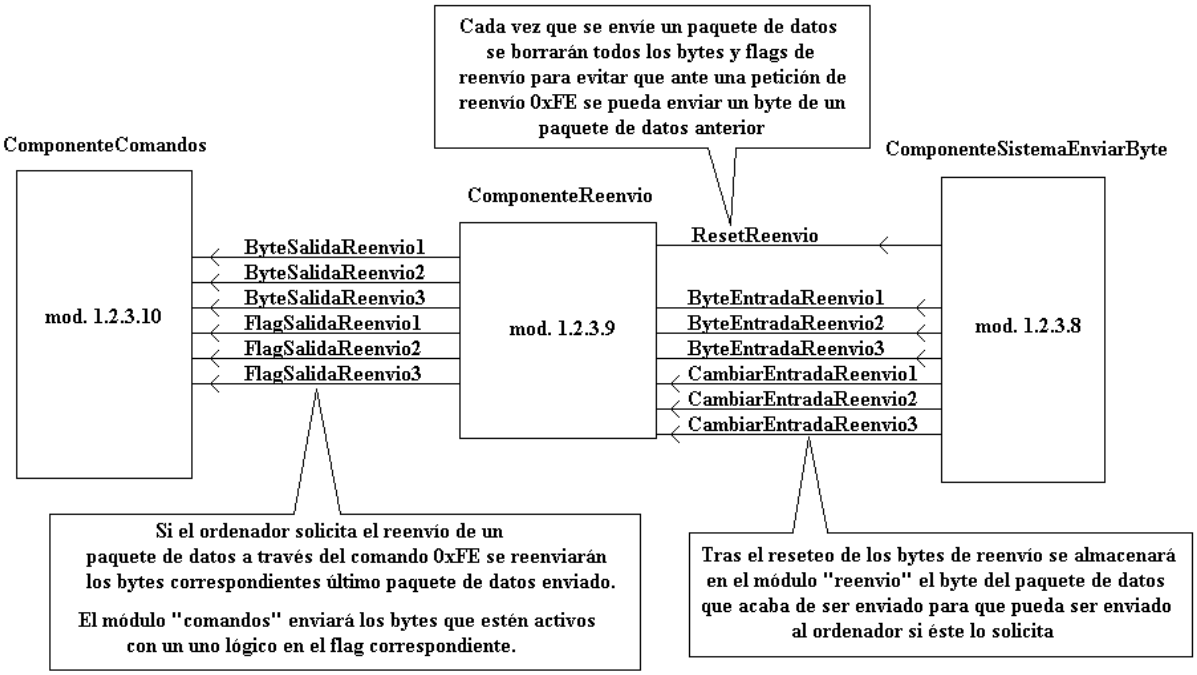
MÓDULO "MEMORIA_DATOS" 1.2.3.6:

En este módulo se encuentran los registros que almacenarán los bytes y flags utilizados en la recepción y en el envío de datos del dispositivo ratón. Si se ha generado un evento de movimiento (Ejemplo: Se ha movido el ratón hacia arriba") se almacenarán en los registros de respuesta los bytes del paquete de datos de movimiento. También se activarán los flags correspondientes a estos registros para provocar el envío sucesivo de los bytes. Si un comando recibido por el dispositivo genera una respuesta para ser enviada al ordenador también se guardará en estos registros y activará los flags correspondientes dependiendo del número de bytes que conforman el paquete de datos de respuesta. En este módulo también se almacena el byte de instrucción que acaba de recibirse desde el ordenador y se activa un flag que provocará la ejecución del comando

o el procesamiento del dato que acaba de recibirse. Existe también un registro que almacena el flag de asentimiento, que cuando está activo provoca el envío del byte de asentimiento 0xFA.



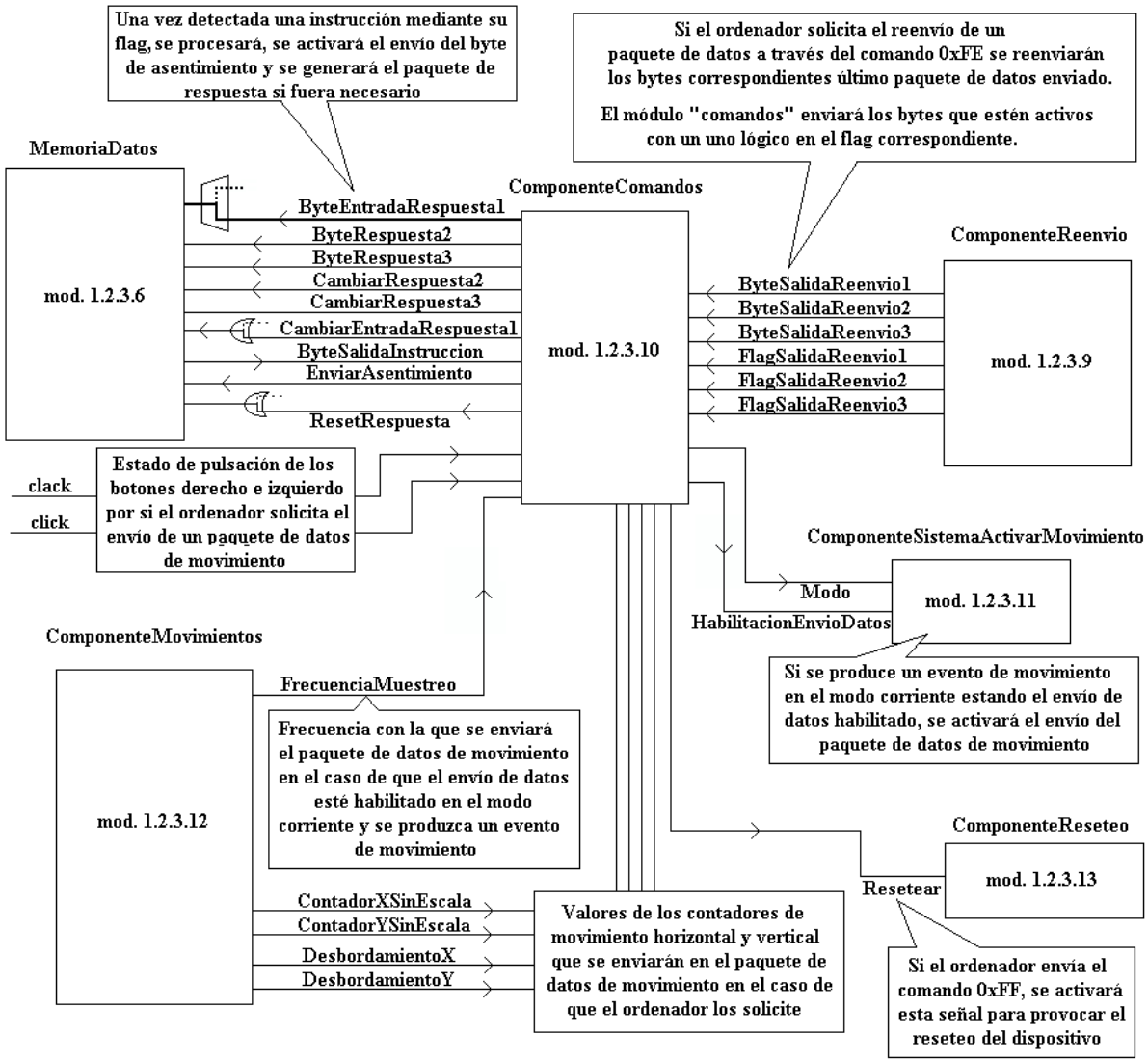
MÓDULO REENVÍO 1.2.3.9:
En este módulo se almacenarán los bytes enviados al ordenador para que puedan ser reenviados en el caso de que el ordenador solicite el reenvío. Si se envió un paquete de datos formado por más de un byte, se volverá a reenviar todo el paquete de datos. El reenvío es la respuesta del sistema ratón al comando FE (REENVÍO) y se activará dentro del módulo "comandos_new". El módulo "reenvio" sólo almacenará los bytes que se enviarán en el caso de que el comando FE lo solicite. Se necesitan entonces tres registros de 8 bits cada uno para guardar los posibles bytes de respuesta que puedan ser enviados.



MÓDULO "COMANDOS_NEW" 1.2.3.10:
Este módulo es el encargado de ejecutar los comandos recibidos del ordenador. El funcionamiento del ratón se basa en la recepción y en el envío de una serie de bytes entre el ordenador y el dispositivo a través del puerto serie PS/2. Los bytes recibidos desde el ordenador deben ser identificados ya que se corresponden con comandos que el ordenador quiere que se ejecuten en el dispositivo. Es necesario un módulo encargado de la ejecución de los distintos comandos que active las señales correspondientes a la solicitud del ordenador. El módulo "comandos_new" es el encargado de esta actividad. Para ello utilizará una máquina de estados, una serie de registros donde se almacenará la información del sistema que es modificada por los comandos (la modificación de esta información es la forma en la que el ordenador controla al dispositivo) y lógica combinacional necesaria para la activación de las señales según el estado en el que se encuentre la máquina de estados.
Los siguientes son los únicos comandos que pueden ser enviados al ratón. Si el ratón está en modo Corriente, el host debería deshabilitar el envío de datos (comando Øxf5) antes de enviar alguno de los otros comandos:

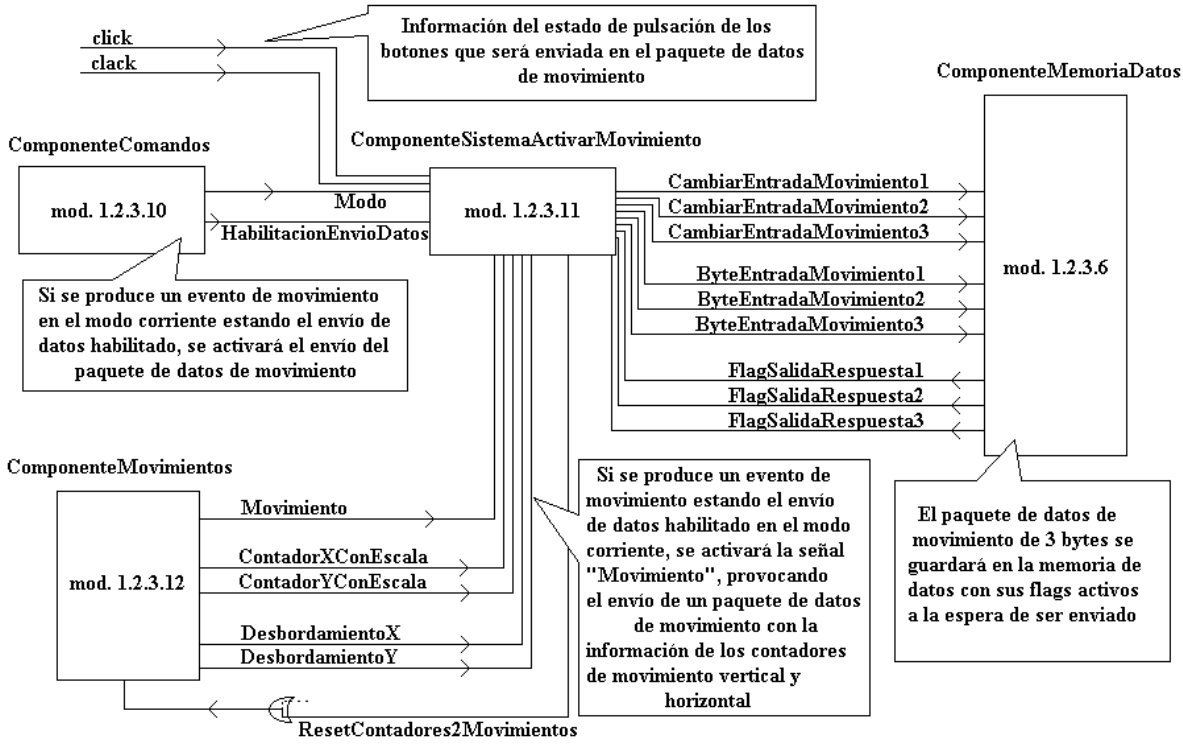
- ØxFF (RESET).
- ØxFE (Reenvío).
- ØxF6 (Fijar los valores por defecto).
- ØxF5 (Deshabilitación del envío de datos).
- ØxF4 (Habilitar el envío de datos).
- ØxF3 (Fijar la frecuencia de muestreo).
- ØxF2 (Obtener el identificador de dispositivo ID).
- ØxF0 (Activar el "modo remoto").
- ØxEE (Activar el modo página).

- ØxEC (Reseteo del modo página).
- ØxEB (Lectura de Datos).
- ØxEA (Activar el modo corriente).
- ØxE9 (Petición de estado).
- ØxE8 (Fijar la resolución).
- ØxE7 (Fijar la escala 2:1).
- ØxE6 (Fijar la escala 1:1).



MÓDULO "SISTEMA_ACTIVAR_MOVIMIENTO" 1.2.3.11:
Este módulo es el encargado de guardar en los registros de la memoria de datos del ratón los bytes de movimiento que serán enviados al ordenador para indicarle el estado del interfaz ratón-usuario. Controlamos la modificación de los registros que guardan los datos de movimiento en la

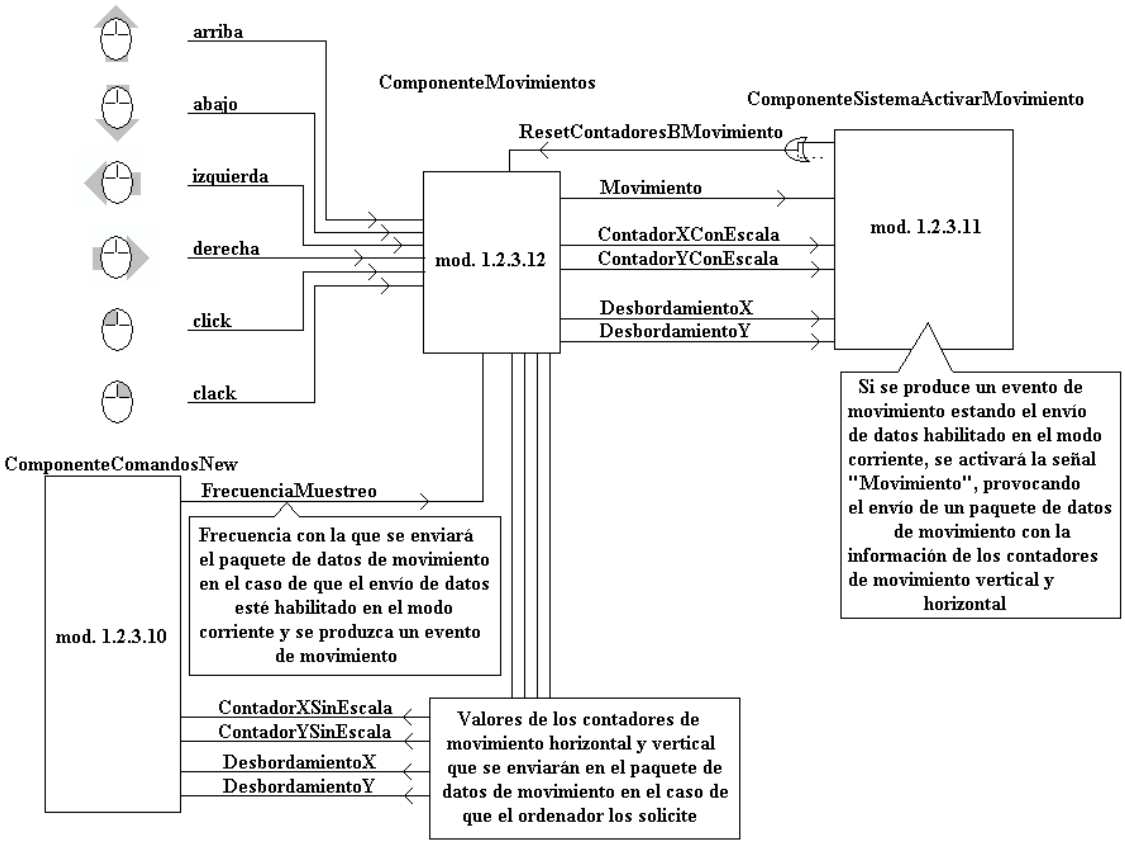
memoria de datos del ratón. También se controla la activación de los flags de la misma memoria que indican que es preciso enviar los bytes de movimiento. Se activarán 3 flags si se ha detectado un movimiento. Otro módulo detectará el valor de estos flags y se procederá al envío de los bytes de movimiento correspondientes. Cada vez que se envía un byte de movimiento, se desactiva el flag que originó el envío, y así hasta que hayan sido enviados los 3 bytes de movimiento.



MÓDULO "MOVIMIENTOS_NEW" 1.2.3.12:

Este módulo es el encargado de avisar al sistema ratón cuando sea preciso el envío de datos de movimiento. Las señales de movimiento en el sentido horizontal (derecha e izquierda) provocarán cambios en el contador de movimientos del eje x. Si se detecta un movimiento hacia la derecha se propondrá como valor de este contador el valor "000000100" (4 en decimal). Si se detecta un movimiento hacia la izquierda se propondrá como valor de este contador el valor "111111100" (- 4 en decimal). Las señales de movimiento en el sentido vertical (arriba y abajo) provocarán cambios en el contador de movimientos del eje y. Si se detecta un movimiento hacia arriba se propondrá como valor de este contador el valor "000000100" (4 en decimal). Si se detecta un movimiento hacia abajo se propondrá como valor de este contador el valor "111111100" (- 4 en decimal). Esta comprobación sólo se realizará cada 50 ms, y en el caso de que el valor de alguno de los contadores anteriores sea distinto de cero, se activará la señal "movimiento" durante 100 ms para indicar al sistema ratón que se ha producido un movimiento. Se mostrarán además a la salida del módulo los valores de los contadores de movimiento en el

eje horizontal (x) y en el eje vertical (y). Estos contadores se mostrarán con escalado y sin escalado. También se comprobará cada 50 ms si se ha producido una variación en el estado de pulsación de los botones del ratón. Esto es, si antes no estaba pulsado uno de los botones y ahora lo está, o si antes estaba pulsado uno de los botones y ahora no lo está. Si se ha producido un cambio en el estado de pulsación de alguno de los botones (botón derecho – clack, botón izquierdo – click) también se activará la señal "movimiento".



MÓDULO "RESETEO" 1.2.3.13:
Este módulo controla el reseteo programado del sistema que controla el protocolo PS/2. Al recibir el comando ØxFF (RESET) se activa este módulo que provocará el reseteo de todo el sistema PS/2 tras el envío del byte de asentimiento 0xFA.

1.2.3.1 enviar_byte.vhd

```
ENTITY enviar_byte IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    enviar: IN STD_LOGIC;
    fin_envio: OUT STD_LOGIC;
    abortar: OUT STD_LOGIC;
    escribir_dato: OUT STD_LOGIC;
    dato_a_escribir: OUT STD_LOGIC;
    dato_leido: IN STD_LOGIC;
    escribir_reloj: OUT STD_LOGIC;
    reloj_leido: IN STD_LOGIC;
    reloj_a_escribir: OUT STD_LOGIC;
    reset_contador_temporal: OUT STD_LOGIC;
    incrementa_contador_temporal: OUT STD_LOGIC;
    reset_contador_indice: OUT STD_LOGIC;
    incrementa_contador_indice: OUT STD_LOGIC;
    cuenta_indice: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    cuenta_temporal_0100: IN STD_LOGIC;
    cuenta_temporal_1000: IN STD_LOGIC;
    cuenta_temporal_1100: IN STD_LOGIC;
    cuenta_indice_1011: IN STD_LOGIC;
    carga_byte_a_enviar: OUT STD_LOGIC;
    byte_a_enviar_salida: IN STD_LOGIC_VECTOR (7 downto 0);
    bit_paridad: IN STD_LOGIC
  );
END enviar_byte;
```

Esta unidad controla el envío de un byte desde el dispositivo ratón al host de acuerdo con el protocolo PS/2. Este módulo generará las señales adecuadas en las líneas de datos y de reloj para permitir la transmisión de un byte desde el dispositivo ratón al host según el protocolo PS/2.

Este módulo realiza la implementación física de la parte del dispositivo ratón encargada de la transmisión de datos con el protocolo serie PS/2. A través de la línea de datos se transmitirán los bits de información sincronizados con un reloj que se transmite a través de la línea de reloj. El ratón PS/2 implementa un protocolo bidireccional serie síncrono. En otras palabras, los datos se envían un bit tras otro en la línea de datos y se leen cada vez que se pulsa el reloj. El ratón puede enviar datos al host, y el host puede enviar datos al dispositivo, pero el host siempre tiene prioridad sobre el bus y puede inhibir la comunicación desde el ratón en cualquier instante al mantener la línea del reloj en nivel bajo.

Los datos enviados desde el ratón al anfitrión se leen en el flanco de bajada de la señal de reloj (cuando el reloj pasa de un nivel alto a un nivel bajo). Los datos enviados desde el host hasta el ratón se leen en el flanco de subida (cuando el reloj va desde el nivel bajo al nivel alto). El ratón genera siempre la señal de reloj, sin importar la dirección de la comunicación. Si el host quiere enviar datos, debe decirle primero al dispositivo que comience a generar la señal de reloj. La frecuencia de reloj máxima es de 33 KHz y la mayoría de los dispositivos funcionan entre 10 y 20 KHz. Se recomienda utilizar una frecuencia en torno a los 15 KHz si se quiere construir un dispositivo. Esto quiere decir que el reloj debería estar a nivel alto sobre 40μs y a nivel bajo sobre 40μs.

Las líneas de Datos y Reloj están ambas a colector abierto (normalmente mantenida a nivel lógico alto). Cuando el ratón quiera enviar información, lo primero que hace es cerciorarse de que el Reloj está en nivel lógico alto. Si no lo está, es que el host está inhibiendo la comunicación y el dispositivo debe guardar la información hasta recuperar el control del bus. El ratón solamente guarda el último paquete enviado. Si la línea de Reloj está en nivel alto, el dispositivo puede empezar a enviar sus datos.

El ratón usa un protocolo serie consistente en una trama de 11-bits. Estos bits son:

- 1 bit de inicio. Este bit siempre está a Ø.
- 8 bits de datos, el menos significativo primero.
- 1 bit de paridad (paridad impar).
- 1 bit de parada. Este bit siempre está a 1.

Cada bit se lee por el anfitrión en el flanco de bajada del reloj. La frecuencia de reloj es de 10-16.7 KHz. El tiempo desde el flanco de subida de un pulso de reloj a una transición de Datos debería ser al menos de 5μs. El tiempo desde una transición de datos a un flanco de bajada de un pulso de reloj debe ser al menos de 5μs y no más grande que 25 μs. Este entramado de tiempo es muy importante y debería seguirse exactamente. El host puede poner la línea a nivel bajo antes del onceavo pulso de reloj (bit de parada), lo que hace que el dispositivo aborte el envío del byte actual (esto es muy raro). Después que el bit de parada sea transmitido, el dispositivo debería esperar al menos 50μs antes de enviar el siguiente paquete. Esto da al host tiempo para inhibir la transmisión mientras procesa el byte recibido (el host debería hacerlo normalmente de manera automática después de cada paquete recibido). El dispositivo debe esperar por lo menos 50μs después de que el host libere una inhibición antes de enviar ningún dato.

Se recomienda el siguiente proceso para enviar un solo octeto desde un emulador de teclado/ratón hasta el host.

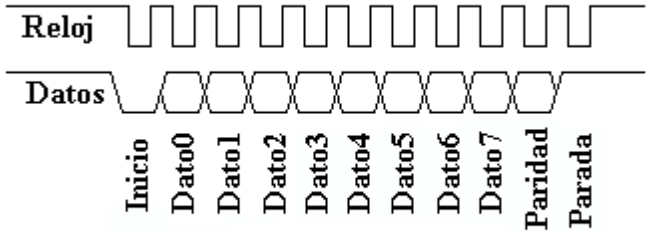
- Esperar hasta que el Reloj esté a nivel alto.
- Retrasar 50μs.
- ¿El Reloj permanece a nivel alto? NO → ir al paso 1.
- ¿La línea de Datos está a nivel alto? NO → Abortar (y leer el byte desde el host).
- Esperar 20μs (=40μs al tiempo que el Reloj es puesto a nivel bajo en el envío del bit de parada).
- Salida del bit de inicio (Ø).
- Salida de 8 bits de datos.
- Salida del bit de paridad.

Después de enviar cada uno de los bits anteriores se debe comprobar el Reloj para estar seguros de que el host no lo ha puesto a nivel bajo la línea, lo cual debería abortar esta situación.

- Salida del bit de parada (1).
- Esperar 30μs (=50μs desde el tiempo en el que el Reloj es liberado en el envío del bit de parada).

El proceso para enviar un simple bit debería ser como sigue:

- Activar/Desactivar Data.
- Esperar 20µs.
- Poner el Reloj a nivel bajo.
- Esperar 40µs.
- Liberar el Reloj.
- Esperar 20µs.



Una máquina de estados desarrolla la funcionalidad anteriormente descrita:

- **S_ABORTAR**: Una vez activado un envío de datos y comprobado que el reloj está a nivel lógico alto se abortará el envío si la línea de datos permanece a nivel alto. La máquina de estados pasará a este estado si se da tal circunstancia. También se abortará el envío de datos si el ordenador corta la comunicación al llevar la línea de datos a nivel lógico bajo, en este caso también se pasará al estado S_ABORTAR. En este estado se genera la señal "abortar" que informará al módulo "sistema_enviar_byte" que el host ha decidido parar la transmisión de un byte. Este módulo parará el proceso de emisión del byte para que el sistema vuelva al estado de espera de una nueva recepción o un nuevo envío. Pasará luego al estado S0 a la espera de una nueva emisión.

- **S0**: Estado de reposo del módulo donde esperará hasta que el dispositivo tenga que enviar un byte al host. El módulo "sistema_enviar_byte" detectará si el dispositivo necesita enviar al host algún byte ya sea por respuesta a un comando o porque haya ocurrido un evento en el dispositivo que es preciso que el host conozca. En este caso se activará la señal "enviar" que provocará que la máquina de estados pase al estado S1 y se inicie la transmisión del byte.

- **S1**: En este estado permanecerá el ratón hasta que la línea de reloj esté con un nivel lógico alto. Si la línea de reloj está a nivel lógico bajo quiere decir que el host está inhibiendo al dispositivo, bien porque no quiere que éste actúe o bien porque quiere que el dispositivo

reciba algún byte. Por lo tanto, no puede iniciarse la transmisión hasta que la línea de reloj esté con un nivel lógico alto. Pasará entonces al estado S2.

- **S2**: En este estado se resetea el contador del índice de bit que se va a transmitir. El protocolo serie se basa en el envío de una serie de bits uno tras otro a través de la línea de datos sincronizados a través del reloj de la línea de reloj. El primer bit que se transmitirá será el bit de inicio, luego el bit 0, el bit 1, el bit 2, el bit 3, el bit 4, el bit 5, el bit 6, el bit 7, el bit de paridad, y el bit de parada. Para ello la máquina de estados entra en un bucle que realiza la emisión de cada uno de los bits. El contador del índice se incrementa cada vez que se emite un bit y será este contador el que indique el bit que se quiere transmitir. Sin embargo, cuando el contador de índice es cero, el bit que se quiere transmitir en este caso es el bit de parada. Esto es porque como el bit de parada es uno lógico y en un principio antes del envío del bit de inicio la línea de datos tiene que estar con un valor uno lógico el contador a cero forzará que en un principio la línea de datos tenga un valor uno lógico. El bit de inicio se transmitirá cuando el contador de índice sea uno y así sucesivamente. Se resetea el contador temporal que llevará la cuenta en unidades de 5 us del tiempo transcurrido, este contador es útil para guardar tiempos de espera y así sincronizar eventos que ocurren con unidades de tiempo múltiplos de 5 us. Se activa la señal "carga_byte_a_enviar" para guardar en los registros del módulo "comun_enviar_recibir" el byte que se quiere enviar. Luego pasará al estado S3.

- **S3**: Esperamos en este estado 50 us, para ello incrementamos el contador temporal hasta que el valor del contador indique que ha transcurrido el tiempo requerido. Luego pasará al estado S4.

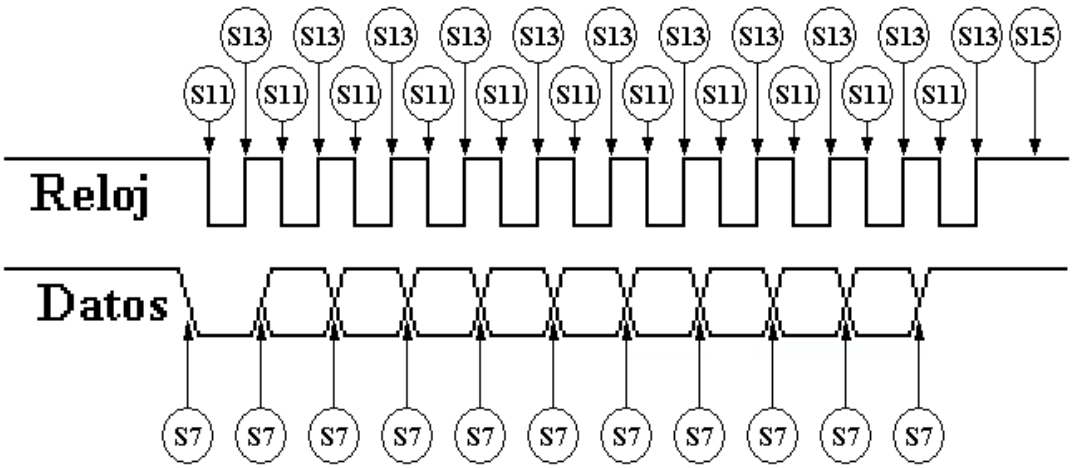
- **S4**: Comprobamos de nuevo que la línea de reloj esté a nivel lógico alto, si está a nivel lógico bajo quiere decir que el host ha inhibido la transmisión. En este caso pasará al estado S1 para esperar que el host ponga de nuevo la línea de reloj a nivel alto para iniciar de nuevo la transmisión. Si la línea de reloj continuaba con un nivel lógico alto pasará al estado S5.

- **S5**: Si la línea de datos está con un nivel lógico cero es porque el host ha interrumpido la transmisión porque quiere enviar un paquete de datos. En este caso se aborta la transmisión pasando al estado S_ABORTAR. Si la línea de datos está con un nivel lógico uno pasará al estado S6 para iniciar la activación de las líneas de datos y de reloj que permitirán la transmisión del byte.

- **S6**: En este estado permanecerá en espera 20 us, luego pasará al estado S7.

- **S7**: En la línea de datos se escribe el valor de salida del multiplexor de bit ('1' lógico si el contador de índice es 0, '0' lógico si el contador de índice es 1, bit0 si el contador de índice es 2, bit1 si el contador de índice es 3, bit2 si el contador de índice es 4, bit3 si el contador de índice es 5, bit4 si el contador de índice es 6, bit5 si el contador de índice es 7, bit6 si el contador de índice es 8, bit7 si el contador de índice es 9, bit de paridad si el contador de índice es 10 y bit de parada si el contador de índice es 11). En la línea de reloj se escribe el valor '1' lógico. Luego se incrementa el contador de índice, con lo cual, debido al retraso del biestable será en el siguiente flanco de subida del reloj cuando varíe el valor del dato que se escribirá en la línea de datos, pero esta diferencia es ínfima ya que el reloj de sistema es

mucho más rápido que el reloj de 5 us. El efecto global de este estado es que empezará a transmitir un cero lógico en la línea de datos (bit de inicio) y mantendrá a uno lógico la línea de reloj. En este estado se comprueba si el ordenador ha solicitado abortar el envío llevando la línea de reloj a cero lógico, pasando en este caso al estado S_ABORTAR. En caso contrario pasará al estado S9.

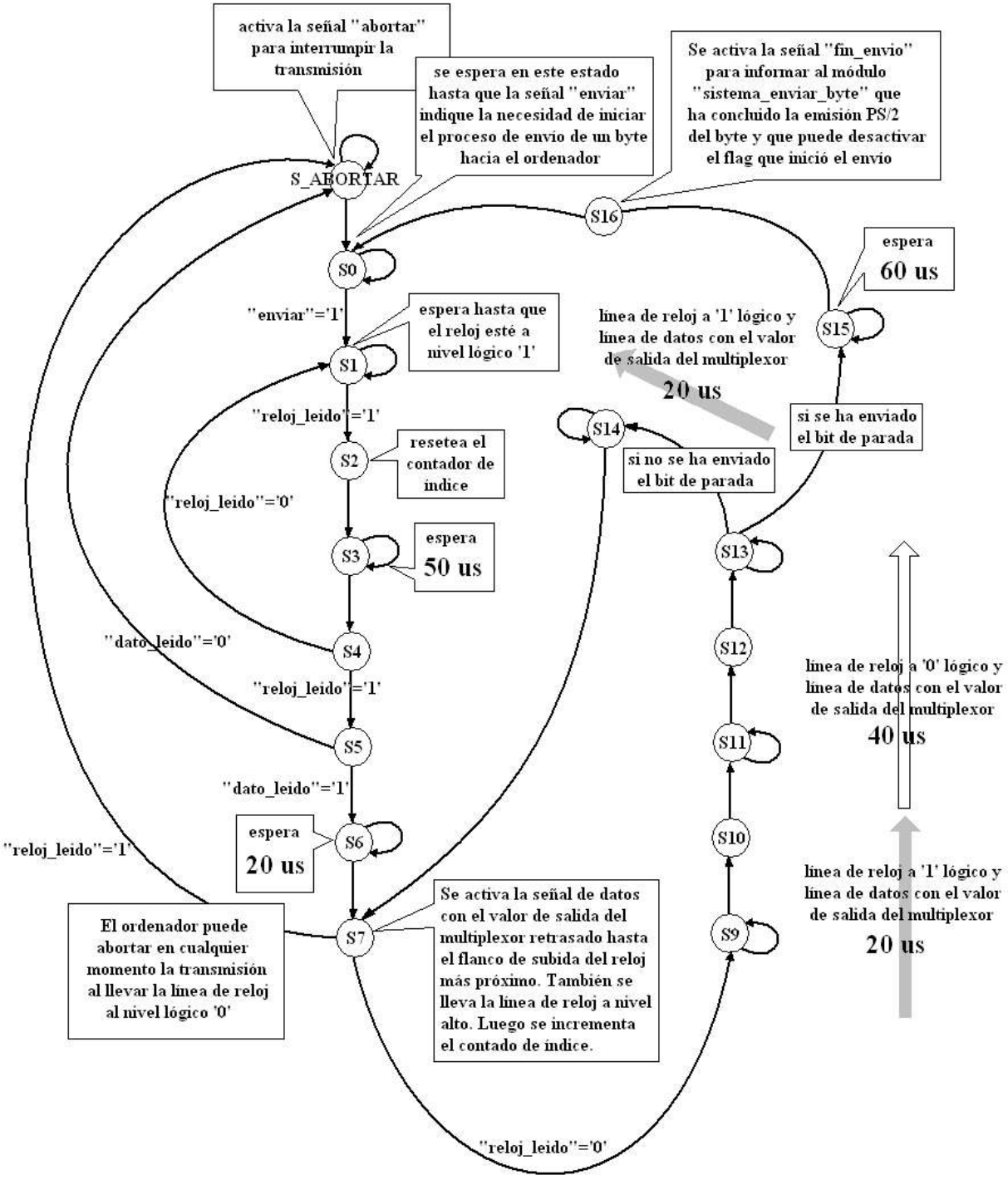


- **S9**: En este estado permanecerá durante 20 us manteniendo en la línea de reloj el valor uno lógico y en la línea de datos el valor de bit especificado por el contado de índice (bit de inicio, bit0, bi1, bit2, bit3, bit4, bit5, bit6, bit7, bit de paridad y bit de parada). Pasará después al estado S10.

- **S10**: Se lleva la línea de reloj a nivel cero lógico, manteniendo en la línea de reloj el valor uno lógico y en la línea de datos el valor de bit especificado por el contado de índice (bit de inicio, bit0, bi1, bit2, bit3, bit4, bit5, bit6, bit7, bit de paridad y bit de parada). Se resetea el contador temporal para que en el siguiente estado el contador temporal esté listo para realizar una espera de un tiempo múltiplo de 5 us. Pasará inmediatamente al estado S11.

- **S11**: En este estado permanecerá durante 40 us manteniendo en la línea de reloj el valor cero lógico y en la línea de datos el valor de bit especificado por el contado de índice (bit de inicio, bit0, bi1, bit2, bit3, bit4, bit5, bit6, bit7, bit de paridad y bit de parada). Pasará después al estado S12.

- **S12**: Estado de transición que mantendrá los valores en las líneas de datos y de reloj del estado anterior (S11). Este estado es necesario para que el tiempo en el que se mantienen las señales con su valor sea el adecuado ya que es preciso un desfase de un ciclo de reloj. Luego pasará al estado S13.



- **S13**: Este estado y el siguiente conformarán un tiempo de 20 us manteniendo la línea de reloj al valor uno lógico y manteniendo en la línea de datos el valor de bit especificado por el contado de índice (bit de inicio, bit0, bi1, bit2, bit3, bit4, bit5, bit6, bit7, bit de paridad y bit de

parada). Pasará al estado S14 si no se ha concluido la emisión de todos los bits en la línea de datos. Si actualmente se está emitiendo el bit de parada, pasará entonces al estado S15.

- **S14**: En este estado permanecerá durante 20 us manteniendo en la línea de reloj el valor uno lógico y en la línea de datos el valor de bit especificado por el contado de índice (bit de inicio, bit0, bi1, bit2, bit3, bit4, bit5, bit6, bit7, bit de paridad y bit de parada). Pasará después al estado S7.

- **S15**: En este estado permanecerá 60 μs para evitar que se envíe un nuevo paquete de datos inmediatamente después de la transmisión del byte. Se libera la línea de reloj y de datos dejando que tomen el valor uno lógico. Ya no se actuará sobre ninguna de las líneas del protocolo serie PS/2. Pasará después al estado S16.

- **S16**: Se activa la señal "fin_envio" para indicar al módulo "sistema_recibir_byte" que se ha transmitido un byte al ordenador. De esta forma se dará por concluida la transmisión del byte. Pasará al estado S0 para permanecer a la espera de una nueva activación para empezar una nueva transmisión de datos desde al ordenador.

Un multiplexor será el encargado de seleccionar el bit a transmitir según el valor del contador de índice. El contador de índice lleva la cuenta del bit actual que se está tratando en el proceso de transmisión del protocolo serie PS/2.

INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **ENVIAR**: El módulo "sistema_enviar_byte" pone en funcionamiento a este módulo al activar esta señal. Se generó una respuesta a un comando o un evento de movimiento en el ratón necesita de ser atendido por el ordenador. Este byte debe ser recibido por el host así que se debe activar al módulo "enviar_byte" para que genere las señales necesarias del protocolo serie PS/2 para permitir al dispositivo transmitir un paquete de datos. El proceso se inicia al activar la señal "enviar".
- **FIN_ENVIO**: Esta señal indica al módulo "sistema_enviar_byte" que ha concluido la emisión del byte que es preciso que reciba el ordenador. De esta forma se dará por concluida la transmisión del byte.
- **ABORTAR**: Si durante el proceso de transmisión el ordenador quiere abortar por algún motivo la transmisión del byte, el módulo "enviar_byte" activará la señal "abortar". El ordenador para la transmisión al llevar la línea de reloj al nivel lógico cero.
- **ESCRIBIR_DATO**: Al activar esta señal con un valor uno lógico se activa el buffer de la línea de datos para que muestre en ella el dato especificado en la línea "dato_a_escribir". Si la señal "escribir_dato" está a cero lógico, la línea se dejará en alta impedancia para que el

ordenador pueda escribir en ella. Si la señal "escribir_dato" está a uno lógico y la señal "dato_a_escribir" está a uno lógico estoy realizando una petición al buffer de la línea de datos para que escriba un uno lógico en la línea, en este caso dejará la línea de datos en alta impedancia. Si la señal "escribir_dato" está a uno lógico y la señal "dato_a_escribir" está a cero lógico estoy realizando una petición al buffer de la línea de datos para que escriba un cero lógico en la línea, en este caso llevará la línea de datos al valor cero lógico.

- **DATO_A_ESCRIBIR**: Dato que se escribirá en la línea de datos cuando se active la señal "escribir_dato". Si la señal "dato_a_escribir" toma el valor cero lógico, se llevará la línea de datos al valor cero lógico. Si la señal "dato_a_escribir" toma el valor uno lógico, se dejará la línea de datos en alta impedancia.
- **DATO_LEIDO**: Valor actual de la línea de datos correspondiente al dato leído. Si la línea de datos está en alta impedancia, el dato leído es uno lógico. Si la línea de datos está con un valor cero lógico, el dato leído es cero lógico.
- **ESCRIBIR_RELOJ**: Al activar esta señal con un valor uno lógico se activa el buffer de la línea de reloj para que muestre en ella el dato especificado en la línea "reloj_a_escribir". Si la señal "escribir_reloj" está a cero lógico, la línea se dejará en alta impedancia para que el ordenador pueda escribir en ella. Si la señal "escribir_reloj" está a uno lógico y la señal "reloj_a_escribir" está a uno lógico estoy realizando una petición al buffer de la línea de reloj para que escriba un uno lógico en la línea, en este caso dejará la línea de reloj en alta impedancia. Si la señal "escribir_reloj" está a uno lógico y la señal "reloj_a_escribir" está a cero lógico estoy realizando una petición al buffer de la línea de reloj para que escriba un cero lógico en la línea, en este caso llevará la línea de reloj al valor cero lógico.
- **RELOJ_LEIDO**: Valor actual de la línea de reloj correspondiente al reloj leído. Si la línea de reloj está en alta impedancia, el reloj leído es uno lógico. Si la línea de reloj está con un valor cero lógico, el reloj leído es cero lógico.
- **RELOJ_A_ESCRIBIR**: Valor del reloj que se escribirá en la línea de reloj cuando se active la señal "escribir_reloj". Si la señal "reloj_a_escribir" toma el valor cero lógico, se llevará la línea de reloj al valor cero lógico. Si la señal "reloj_a_escribir" toma el valor uno lógico, se dejará la línea de reloj en alta impedancia.
- **RESET_CONTADOR_TEMPORAL**: Esta señal la activa el módulo "enviar_byte" cuando quiere resetear el valor del contador temporal para posteriormente iniciar una nueva espera.
- **INCREMENTA_CONTADOR_TEMPORAL**: Esta señal la activa el módulo "enviar_byte" cuando quiere incrementar el valor del contador temporal. El incremento sólo se aplicará en el flanco de bajada del reloj de 5 us. Cuando quiera incrementarse el valor del contador, se mantendrá activa esta señal con un valor uno lógico. En el flanco de bajada del reloj se producirá el incremento. Se mantendrá activa la señal hasta que el valor del contador sea el adecuado según el tiempo asignado en la espera. El contador temporal se utiliza en procesos de espera, siendo los tiempos de espera múltiplos de 5 us.
- **RESET_CONTADOR_INDICE**: Esta señal la activa el módulo "enviar_byte" cuando quiere resetear el valor del contador de índice. El contador índice lleva la cuenta del bit actual que se está tratando en el proceso de transmisión del protocolo serie PS/2: 0 si es un '1' lógico (valor inicial de la línea de datos), 1 si es el bit de inicio ('0' lógico), 2 si es el bit0, 3 si es el bit1, 4 si es el bit2, 5 si es el bit3, 6 si es el bit4, 7 si es el bit5, 8 si es el bit6, 9 si es el bit7, 10 si es el bit de paridad y 11 si es el bit de parada ('1' lógico).
- **INCREMENTA_CONTADOR_INDICE**: Esta señal la activa el módulo "enviar_byte" cuando quiere incrementar el valor del contador de índice. El contador índice lleva la cuenta

del bit actual que se está tratando en el proceso de transmisión del protocolo serie PS/2: 0 si es un '1' lógico (valor inicial de la línea de datos), 1 si es el bit de inicio ('0' lógico), 2 si es el bit0, 3 si es el bit1, 4 si es el bit2, 5 si es el bit3, 6 si es el bit4, 7 si es el bit5, 8 si es el bit6, 9 si es el bit7, 10 si es el bit de paridad y 11 si es el bit de parada ('1' lógico).

- **CUENTA_INDICE**: El contador índice lleva la cuenta del bit actual que se está tratando en el proceso de transmisión del protocolo serie PS/2: 0 si es un '1' lógico (valor inicial de la línea de datos), 1 si es el bit de inicio ('0' lógico), 2 si es el bit0, 3 si es el bit1, 4 si es el bit2, 5 si es el bit3, 6 si es el bit4, 7 si es el bit5, 8 si es el bit6, 9 si es el bit7, 10 si es el bit de paridad y 11 si es el bit de parada ('1' lógico).
- **CUENTA_TEMPORAL_0100**: Esta señal se activará cuando la cuenta del contador temporal indique 4 (20 us).
- **CUENTA_TEMPORAL_1000**: Esta señal se activará cuando la cuenta del contador temporal indique 8 (40 us).
- **CUENTA_TEMPORAL_1100**: Esta señal se activará cuando la cuenta del contador temporal indique 12 (60 us).
- **CUENTA_INDICE_1011**: Esta señal se activará cuando la cuenta del contador del índice mantenga el valor 11.
- **CARGA_BYTE_A_ENVIAR**: Cuando se activa esta señal se guarda en el módulo "comun_enviar_recibir" el byte que es preciso que sea atendido por el ordenador.
- **BYTE_A_ENVIAR_SALIDA**: El byte guardado en el módulo "comun_enviar_recibir" será accesible por el módulo "enviar_byte" a través de este vector de bit. Un multiplexor será el encargado de seleccionar el bit a transmitir según el valor del contador de índice. El contador de índice lleva la cuenta del bit actual que se está tratando en el proceso de transmisión del protocolo serie PS/2: 0 si es un '1' lógico (valor inicial de la línea de datos), 1 si es el bit de inicio ('0' lógico), 2 si es el bit0 del byte "byte_a_enviar_salida", 3 si es el bit1 del byte "byte_a_enviar_salida, 4 si es el bit2 del byte "byte_a_enviar_salida, 5 si es el bit3 del byte "byte_a_enviar_salida, 6 si es el bit4 del byte "byte_a_enviar_salida, 7 si es el bit5 del byte "byte_a_enviar_salida, 8 si es el bit6 del byte "byte_a_enviar_salida, 9 si es el bit7 del byte "byte_a_enviar_salida, 10 si es el bit de paridad y 11 si es el bit de parada ('1' lógico). El byte "byte_a_enviar_salida" es el byte que será transmitido al ordenador a través de las líneas de datos y de ratón del protocolo serie PS/2.
- **BIT_PARIDAD**: El bit de paridad "bit_paridad" es tal que el número de bits con valor uno lógico del byte que se va a transmitir ("byte_a_enviar_salida") y almacenado en el módulo "comun_enviar_recibir" junto con el bit de paridad es impar. Ejemplos: Si el byte no tiene ningún bit con valor uno lógico, el bit de paridad será uno lógico. Si el byte tiene tres bits con valor uno lógico, el bit de paridad será cero lógico. Este bit de paridad se enviará también al ordenador para que éste detecte si se ha producido un error en la transmisión de algún bit.

1.2.3.2 recibir_byte.vhd

```
ENTITY recibir_byte IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    recibir: IN STD_LOGIC;
    validacion_byte_recibido: OUT STD_LOGIC;
    abortar: OUT STD_LOGIC;
    error: OUT STD_LOGIC;
    -- signals para controlar los buffers de salida
    escribir_dato: OUT STD_LOGIC;
    dato_a_escribir: OUT STD_LOGIC;
    dato_leido: IN STD_LOGIC;
    escribir_reloj: OUT STD_LOGIC;
    reloj_leido: IN STD_LOGIC;
    reloj_a_escribir: OUT STD_LOGIC;
    --ComunEnviarRecibir
    recibiendo_byte: OUT STD_LOGIC;
    reset_contador_temporal: OUT STD_LOGIC;
    incrementa_contador_temporal: OUT STD_LOGIC;
    reset_contador_indice: OUT STD_LOGIC;
    incrementa_contador_indice: OUT STD_LOGIC;
    cuenta_indice: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    cuenta_temporal_0011: IN STD_LOGIC;
    cuenta_temporal_0100: IN STD_LOGIC;
    cuenta_temporal_0101: IN STD_LOGIC;
    cuenta_temporal_1000: IN STD_LOGIC;
    cuenta_temporal_1001: IN STD_LOGIC;
    cuenta_temporal_1100: IN STD_LOGIC;
    cuenta_indice_1001: IN STD_LOGIC;
    vector_carga_byte_recibido: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    bit_paridad: IN STD_LOGIC
  );
END recibir_byte;
```

Esta unidad controla la recepción de un byte desde el host de acuerdo con el protocolo PS/2. Este módulo generará las señales adecuadas en las líneas de datos y de reloj para permitir la transmisión de un byte desde el host al dispositivo según el protocolo PS/2.

El paquete de datos PS/2 es enviado de forma un poco diferente en la comunicación Host a Dispositivo en contraposición a la comunicación Dispositivo a Host. Siempre es el dispositivo PS/2 el que genera la señal de reloj. Si el host quiere enviar datos, debe poner las líneas de Reloj y Datos en el estado de Petición de Envío.

- Inhibir la comunicación al poner el Reloj a nivel bajo al menos 100μs.
- Aplicar la "Petición de envío" al poner la línea de Datos a nivel bajo, entonces liberar el Reloj.

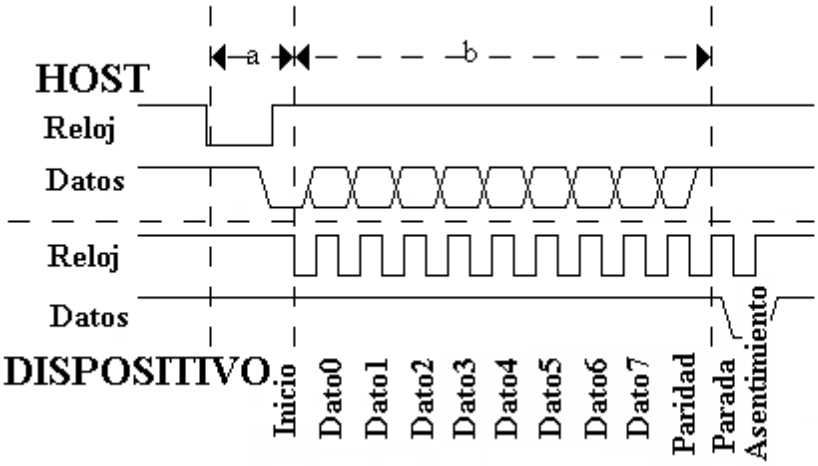
El dispositivo debería chequear este estado en intervalos que no excedan de 10 milisegundos. Cuando el dispositivo detecte este estado, empezará a generar la señal de Reloj para los 8 bits de datos y el bit de parada. El host sólo cambiará la línea de Datos cuando la línea de Reloj esté a nivel bajo, los datos serán tomados en el flanco de subida del pulso de reloj. Esto es lo contrario a lo que ocurría en la comunicación Dispositivo a Host.

Después de que el bit de parada sea enviado, el dispositivo debe asentir el byte recibido poniendo la línea de datos a nivel bajo y generar un último pulso de reloj. Si el host no libera la línea de Datos después del onceavo pulso, el dispositivo continuará la generación de pulsos de reloj hasta que la línea de Datos sea liberada (el dispositivo generará entonces un error). El Host puede abortar la transmisión antes del onceavo pulso de reloj (bit de asentimiento) al mantener la línea de Reloj a nivel bajo al menos 100μs.

Para hacer el proceso un poco más fácil de entender, los pasos que el host debe seguir para enviar un dato al dispositivo PS/2 son:

- Llevar la línea de Reloj a un nivel bajo por al menos 100μs.
- Llevar la línea de Datos a un nivel bajo.
- Liberar la línea del Reloj.
- Esperar hasta que el dispositivo lleve la línea de Reloj a nivel bajo.
- Activar/desactivar la línea de Datos para enviar el primer bit de datos.
- Esperar hasta que el dispositivo lleve la línea de Reloj a nivel alto.
- Esperar hasta que el dispositivo lleve la línea de Reloj a nivel bajo.
- Repetir los pasos 5-7 para los otros 7 bits de datos y el bit de paridad.
- Liberar la línea de Datos.
- Esperar hasta que el dispositivo lleve la línea a nivel bajo.
- Esperar hasta que el dispositivo lleve la línea del Reloj a nivel bajo.
- Esperar hasta que el dispositivo libere la línea de Datos y la línea del Reloj.

La figura muestra importantes consideraciones temporales (a) y (b).



- El tiempo que se toma el dispositivo para empezar a generar los pulsos de reloj después que el host inicialmente lleve la línea de Reloj a un nivel bajo, no debe ser superior a 15 ms.
- El tiempo que se toma para que el paquete sea enviado, no debe ser más grande de 2 ms. Si no se encuentra uno de estos límites, el host generará un error. Inmediatamente después de que el paquete sea recibido, el host debe llevar la línea de Reloj a nivel bajo para inhibir la

comunicación mientras los datos son procesados. Si el comando enviado por el host requiere una respuesta, esa respuesta debe ser recibida no más tarde de 20 ms después de que el host libere la línea de Reloj. Si esto no ocurre, el host genera un error. Como ocurría en el caso de la comunicación de Dispositivo al Host, ninguna transición de Datos puede ocurrir con 5µs de una transición de Reloj.

Si se quiere emular un ratón o un teclado, se recomienda leer los datos desde el host como sigue:

Chequear si la línea de datos está a nivel bajo al menos cada 10ms.
Si la línea de datos ha sido llevada a nivel bajo por el host l, leer un byte desde el host.
Esperar hasta que la línea de Reloj esté en nivel alto.¿Está la línea de Datos todavía a nivel bajo?

NO→Un error ha ocurrido; Abortar.

Leer 8 bits de datos.
Leer el bit de paridad.
Leer el bit de parada.
Después de la lectura de cada uno de los bits, mirar el Reloj para estar seguro de que el host no la ha llevado a nivel bajo (lo cual abortaría esta transmisión).
¿La línea de datos continúa en nivel Ø?

SÍ→Mantenerse pulsando el reloj hasta que la línea de Datos tome el valor 1, entonces generar un error.

Salida del bit de asentimiento.
Chequear el bit de paridad. Generar un error si el bit es incorrecto.
Retraso de 45µs (para darle tiempo al host de inhibir la siguiente transición).

La lectura de cada bit (8 bits de datos, bit de paridad, y bit de parada) es como sigue:

7) Retraso de 20µs
8) Llevar el Reloj a nivel bajo.
9) Retraso de 40µs
10) Liberar el Reloj.
11) Retraso de 20µs.
12) Leer la línea de datos.

El envío del bit de asentimiento es como sigue:

- Retraso de 15µs.
- Llevar la línea de datos a nivel bajo.
- Retraso de 5µs.
- Llevar la línea de Reloj a nivel bajo.
- Retraso de 40µs.
- Liberar el Reloj.
- Retraso de 5µs.
- Liberar la línea de Datos.

Una máquina de estados controla el funcionamiento secuencial del módulo:

- **S_ABORTAR**: El host solicita enviar datos cuando pone la línea de reloj a nivel bajo durante al menos 100 μs. Luego lleva la línea de datos a nivel bajo y libera la línea de reloj. Si tras la liberación de la línea de reloj la línea de datos no permanece a nivel bajo, el ratón abortará la recepción. También el host puede abortar la recepción de un byte cuando en el proceso de envío de los diversos bits a la par que la línea de reloj, el host lleva la línea de reloj a nivel bajo (en el proceso de envío de los diversos bits es el ratón el que modifica la línea de reloj). El módulo "recibir_byte" pasará en estos casos al estado "S_ABORTAR". En este estado se genera la señal "abortar" que informará al módulo "sistema_recibir_byte" que el host ha decidido parar la recepción de un byte. Este módulo parará el proceso de recepción del byte para que el sistema vuelva al estado de espera para iniciar una nueva recepción o un nuevo envío. Pasará luego al estado S1 a la espera de una nueva recepción.

- **S1**: Estado de reposo del módulo donde esperará hasta que el host decida enviar un byte. El host solicita enviar datos cuando pone la línea de reloj a nivel bajo durante al menos 100 μs. El componente "sistema" activará la "recibir" en ese momento, lo cual provocará que la máquina de estados pase al estado S2.

- **S2**: El host ha solicitado enviar un byte y el dispositivo ha iniciado el proceso de recepción, por eso y hasta que concluya la recepción mantendrá activada la señal "recibiendo_byte" con un valor uno lógico para que los registros comunes del módulo "comun_enviar_recibir" sean utilizados correctamente en el proceso de recepción. La línea de reloj está entonces a nivel lógico bajo y esperará en este estado hasta que la línea de datos esté a nivel bajo, entonces pasará al estado S3.

- **S3**: El host ha solicitado enviar un byte llevando la línea de reloj a nivel bajo, luego llevó la línea de datos a nivel bajo. Se esperará en este estado hasta que la línea de reloj pase a nivel lógico alto. Si la línea de datos no permanece a nivel lógico bajo en ese momento ha ocurrido un error y se debe abortar la recepción, pasará entonces al estado S_ABORTAR. Si la línea de datos se mantiene con un valor lógico bajo, pasará al estado S4 para continuar el proceso de recepción.

- **S4**: Se lleva la línea de reloj a un nivel lógico uno (dejándola en alta impedancia). Se resetea el contador temporal que llevará la cuenta en unidades de 5 us del tiempo transcurrido. Se resetea el contador del índice que me indicará el número del bit que se está recibiendo. Pasará luego al estado S5.

- **S5**: Se espera en este estado hasta que hayan pasado al menos 40 μs desde que se llevó la línea de reloj a un nivel lógico alto (dejándola en alta impedancia), este valor se mantiene durante este tiempo. Para que el contador temporal tenga activa la cuenta en unidades de 5 us y se pueda saber el tiempo transcurrido se activa la señal "incrementa_contador_temporal". Pasará luego al estado S6.

- **S6**: Se lleva la línea de reloj a un nivel lógico bajo. Se resetea el contador temporal que llevará la cuenta en unidades de 5 us del tiempo transcurrido. Pasará luego al estado S7.

- **S7**: Se espera en este estado hasta que hayan pasado 40 μs desde que se llevó la línea de reloj a un nivel lógico bajo, este valor se mantiene durante este tiempo. Para que el contador temporal tenga activa la cuenta en unidades de 5 us y se pueda saber el tiempo transcurrido se activa la señal "incrementa_contador_temporal". Pasará luego al estado S9.

- **S9**: Se espera en este estado hasta que hayan pasado 20 μs manteniendo la línea de reloj a un nivel lógico alto durante este tiempo. Para que el contador temporal tenga activa la cuenta en unidades de 5 us y se pueda saber el tiempo transcurrido se activa la señal "incrementa_contador_temporal". Pasará luego al estado S10.

- **S10**: Se mantiene la línea de reloj a un nivel lógico alto. Si el contador de índice tiene almacenado el valor 9 pasará al estado S13, puesto que ya ha concluido la recepción de todos los bits de datos y el bit de paridad. Si el contador de índice no tiene almacenado el valor 9, pasará al estado S11 para continuar la carga del resto de bit que faltan en el paquete de datos recibido desde el ordenador. La carga de bits se produce de forma sucesiva y el contador de índice indica el bit actual que se está recibiendo por la línea de datos.

- **S11**: Se guarda el bit recibido en el registro correspondiente del componente "comun_enviar_recibir". En este instante la señal de datos es estable y se mantiene la línea de reloj con un valor uno lógico. El bit se almacenará en el registro correspondiente según el valor del contador de índice. La carga de bits se produce de forma sucesiva y el contador de índice indica el bit actual que se está recibiendo por la línea de datos. Una vez concluida la recepción se tendrá almacenado todo el paquete de datos recibido, cada bit en su registro correspondiente. Luego se incrementa el contador de indice para que a la mitad del siguiente nivel alto en la línea de reloj se guarde el siguiente bit. En este estado se detecta si la línea de reloj la ha llevado a un valor cero lógico el ordenador. Esto implicaría que el ordenador ha realizado una petición para terminar la recepción. Pasará al estado S_ABORTAR para así abortar la recepción. En caso contrario pasará al estado S12.

- **S12**: Como la lectura del bit se produce a la mitad de un semiciclo de reloj (a la mitad del nivel lógico alto), debe permanecer el reloj otros 20 μs con el valor uno lógico. En este estado se espera hasta que haya concluido este tiempo, incrementando el contador temporal para así tener constancia del tiempo transcurrido. Una vez transcurrido este tiempo pasará al estado S6 para proceder a una nueva lectura de bit. Así termina el bucle de lectura de bit: 40 μs con nivel lógico cero, 20 μs con nivel lógico uno, lectura del bit y otros 20 μs manteniendo el nivel lógico uno.

- **S13**: En el último ciclo se leyó el bit de paridad y al pasar de nuevo por el estado S10 se detectó que ya se había recibido este bit (contador de índice con valor 9), estando la transmisión en la mitad del semiciclo donde el reloj toma el valor lógico uno, debe permanecer otros 20 μs la señal de reloj con valor uno lógico (un semiciclo de reloj son 40 μs). En este estado se resetea el contador temporal para que en el siguiente se realice la espera de la conclusión de estos 20 μs. En este estado y en el siguiente se mantendrá la señal de reloj con un valor uno lógico. Es en el estado S13 donde se realiza la lectura del bit de parada del protocolo serie PS/2. Si la línea de datos toma el valor de parada uno lógico, la recepción del

byte fue correcta y pasará al estado S21 para enviar el bit de asentimiento. Si por el contrario, la línea de datos toma el valor cero lógico, se mantendrá pulsando el reloj con un semiperiodo de 40 μs hasta que tome el valor uno lógico, para ello pasará al estado S14.

- **S14**: Como al leer el bit de parada tomó el valor lógico cero, ha ocurrido un error. En este caso se mantendrá pulsando el reloj con un semiperiodo de 40 μs hasta que la línea de datos tome el valor uno lógico. Por eso esperará en este estado 20 μs más manteniendo el valor uno lógico en la señal de reloj. Luego pasará al estado S16.

- **S16**: En este estado se mantendrá la señal de reloj con un valor cero lógico durante 40 μs.

- **S17**: En este estado se activa la señal de reloj con un valor uno lógico y se resetea el contador temporal para reiniciar la cuenta de múltiplos de 5 μs.

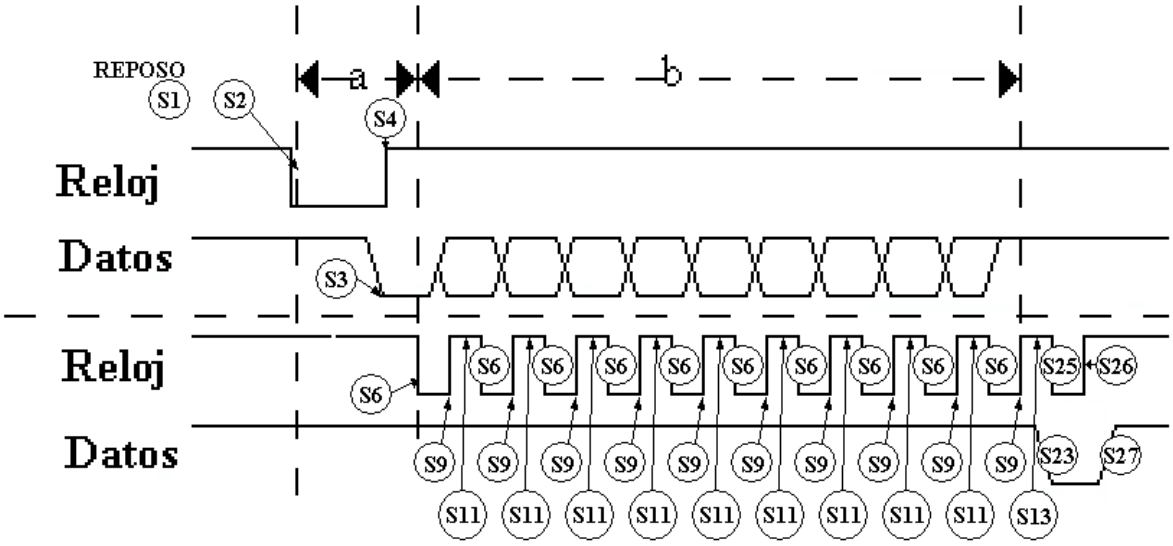
- **S18**: En este estado permanecerá hasta que hayan transcurrido 20 μs desde que se activó la señal de reloj con un valor uno lógico, para ello incrementará el contador temporal hasta que alcance el valor indicativo del tiempo requerido (múltiplo de 5 μs).

- **S19**: En este estado se mantiene la señal de reloj con un valor uno lógico y se resetea el contador temporal para reiniciar la cuenta de múltiplos de 5 μs. Si el dato leído toma el valor cero lógico, se mantendrá pulsando el reloj durante un ciclo más, por eso pasará en este caso al estado S14. Si el dato leído toma el valor uno lógico pasará al estado S20.

- **S20**: Se procede al envío del bit de asentimiento tras la recepción del bit de parada. Se resetea el contador temporal para reiniciar la cuenta de múltiplos de 5 μs. En este estado la línea de reloj continúa con el valor uno lógico. Se activa la señal "error" para generar las señales que transmitirán el byte FE (REENVÍO) si el error se ha producido por primera vez o el byte FC (ERROR) si el error se ha producido más de una vez. Pasará al estado S22 para controlar que transcurran 20 μs desde la mitad del semiperiodo con la línea de reloj a nivel alto.

- **S21**: Tras la recepción de un bit de parada correcto, se debe enviar el bit de asentimiento. La lectura del bit de parada se produjo en el estado S13, correspondiéndose ese instante al de mitad del último semiciclo de reloj con nivel lógico alto (20 μs después de que el reloj tomara el nivel lógico alto). Es en este instante donde se comprueba que el reloj no ha sido puesto a nivel bajo por el ordenador, lo cual implicaría una petición del ordenador de abortar la transmisión. Pasaría entonces al estado S_ABORTAR para que la recepción del byte fuera abortada inmediatamente. Si como es normal la línea de reloj permanece a nivel lógico alto, pasará al estado S22 y empezar con la transmisión del bit de asentimiento, no sin antes resetear el contador temporal para así iniciar una nueva cuenta en el proceso de espera.

- **S22**: Se mantendrá la línea de reloj a nivel alto 20 μs la otra mitad del semiperiodo con la línea de reloj a nivel alto, así concluirá el semiperiodo de reloj. Pero en el envío del bit de asentimiento, primero debe permanecer la línea de datos a nivel alto 15 μs de los 20 μs que estará todavía la línea de reloj a nivel alto. En este estado permanecerá esos 15 μs, luego pasará al estado S23.



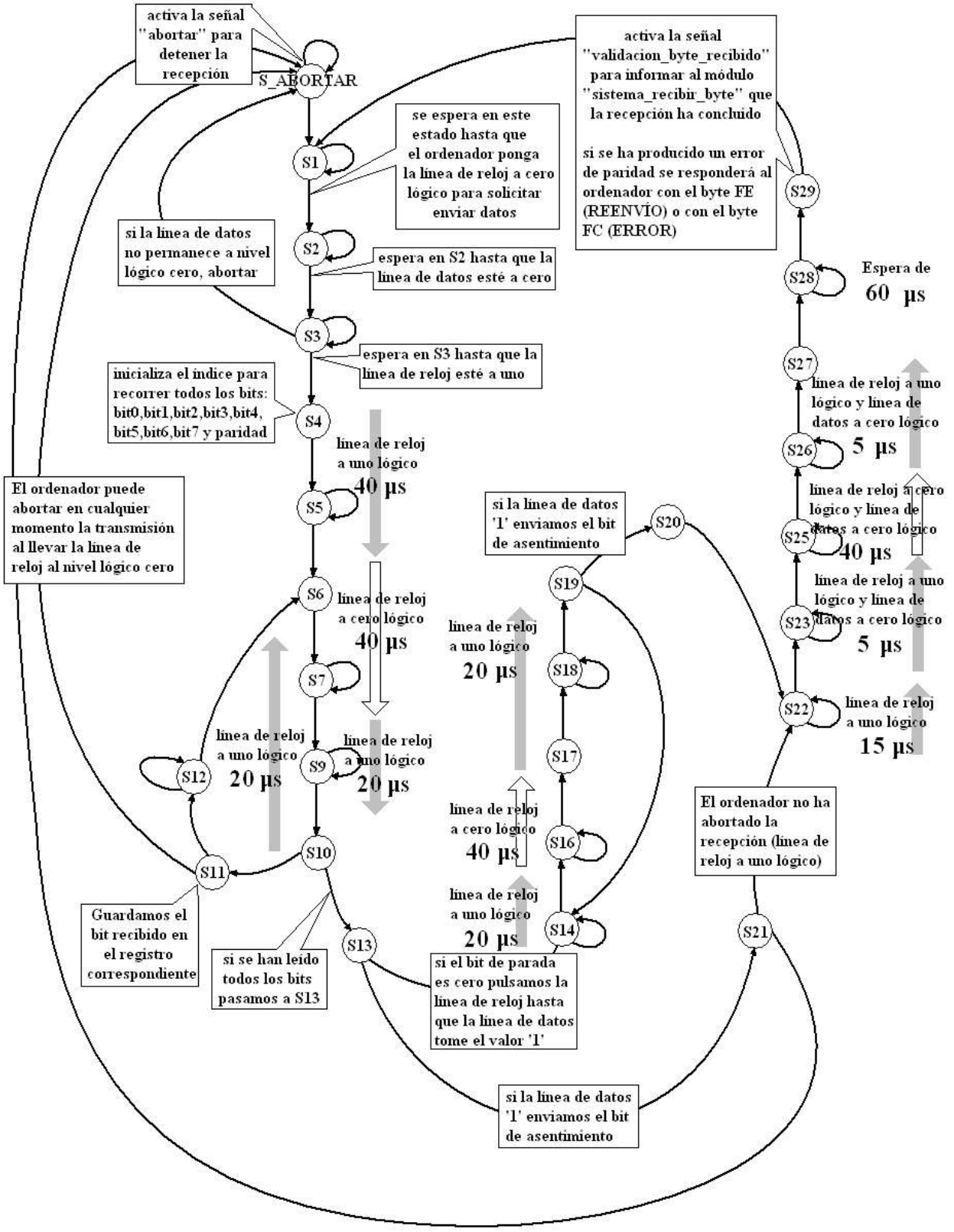
- **S23**: Se mantiene la línea de reloj a nivel alto 5 μs más mientras que la línea de datos es llevada al nivel lógico bajo. Luego pasará al estado S25.

- **S25**: Durante 40 μs la línea de reloj y la línea de datos permanecerán ambas con un nivel cero lógico. Luego pasará al estado S26.

- **S26**: Se libera la línea de reloj para que tome el valor lógico uno y se mantiene la línea de datos a nivel lógico cero 5 μs más. De esta forma se termina de enviar el pulso de asentimiento con una duración de 50 μs centrado en el último nivel bajo de reloj de un paquete de envío del protocolo serie PS/2. Pasará luego al estado S27.

- **S27**: Se libera la línea de datos dejando que tome el valor uno lógico. Ya no se actuará sobre ninguna de las líneas del protocolo serie PS/2. La línea de datos y de reloj están en alta impedancia. Se resetea el contador temporal para iniciar una nueva espera, ya que no se debe enviar ningún nuevo byte hasta pasados por lo menos 60 μs. Pasará al estado S28 para realizar esta espera.

- **S28**: En este estado permanecerá 60 μs para evitar que se envíe un nuevo paquete de datos inmediatamente después de la recepción del byte, luego pasará al estado S29.


activa la señal "abortar" para detener la recepción
S_ABORTAR
activa la señal "validacion_byte_recibido" para informar al módulo "sistema_recibir_byte" que la recepción ha concluido
si se ha producido un error de paridad se responderá al ordenador con el byte FE (REENVIO) o con el byte FC (ERROR)
S1
se espera en este estado hasta que el ordenador ponga la línea de reloj a cero lógico para solicitar enviar datos
S29
si la línea de datos no permanece a nivel lógico cero, abortar
S2
espera en S2 hasta que la línea de datos esté a cero
S28
Espera de 60 µs
S3
espera en S3 hasta que la línea de reloj esté a uno
S27
inicializa el índice para recorrer todos los bits: bit0,bit1,bit2,bit3,bit4, bit5,bit6,bit7 y paridad
S4
línea de reloj a uno lógico y línea de datos a cero lógico
S26
5 µs
línea de reloj a uno lógico
40 µs
S5
El ordenador puede abortar en cualquier momento la transmisión al llevar la línea de reloj al nivel lógico cero
línea de reloj a cero lógico y línea de datos a cero lógico
S25
40 µs
si la línea de datos '1' enviamos el bit de asentimiento
S20
S19
línea de reloj a uno lógico y línea de datos a cero lógico
S23
5 µs
S6
línea de reloj a cero lógico
40 µs
línea de reloj a uno lógico
20 µs
S18
S7
S22
línea de reloj a uno lógico
15 µs
línea de reloj a uno lógico
20 µs
S12
S9
línea de reloj a uno lógico
20 µs
S17
línea de reloj a cero lógico
40 µs
S16
El ordenador no ha abortado la recepción (línea de reloj a uno lógico)
S10
S11
línea de reloj a uno lógico
20 µs
S14
S21
Guardamos el bit recibido en el registro correspondiente
si se han leído todos los bits pasamos a S13
S13
si el bit de parada es cero pulsamos la línea de reloj hasta que la línea de datos tome el valor '1'
si la línea de datos '1' enviamos el bit de asentimiento

- **S29**: Se activa la señal "validacion_byte_recibido" para indicar al módulo "sistema_recibir_byte" que se ha recibido un byte desde el ordenador. Si el bit de paridad recibido no es el bit de paridad de acuerdo con el byte recibido se activará la señal "error" para generar las señales que transmitirán el byte FE (REENVÍO) si el error se ha producido por primera vez o el byte FC (ERROR) si el error se ha producido más de una vez. De esta forma se dará por concluida la recepción del byte y serán otros módulos los encargados de procesar el dato recibido. Este dato ya está guardado en el módulo "comun_enviar_recibir". Pasará al estado S1 para permanecer a la espera de una nueva lectura de datos desde el ordenador.

Un registro interno almacenará el bit de paridad recibido para así detectar si se ha producido algún error en la recepción de algún byte. Como la comprobación del bit de paridad se hace al final, es necesario almacenar el valor del bit hasta ese instante.

INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **RECIBIR**: El módulo "sistema_recibir_byte" pone en funcionamiento a este módulo al activar esta señal. Detectó una solicitud de envío por parte del ordenador así que debe activar al módulo "recibir_byte" para que genere las señales necesarias del protocolo serie PS/2 para permitir al ordenador transmitir un paquete de datos. El proceso se inicia al activar la señal "recibir".
- **VALIDACION_BYTE_RECIBIDO**: Una vez concluido el proceso de recepción del paquete de datos desde el ordenador, la máquina de estados del módulo activa la señal "validacion_byte_recibido" para indicar al módulo "sistema_recibir_byte" que se ha recibido un byte desde el ordenador. Si el bit de paridad recibido no es el bit de paridad de acuerdo con el byte recibido se activará también la señal "error" para generar las señales que transmitirán el byte FE (REENVÍO) si el error se ha producido por primera vez o el byte FC (ERROR) si el error se ha producido más de una vez. De esta forma se dará por concluida la recepción del byte y serán otros módulos los encargados de procesar el dato recibido. Este dato ya está guardado en el módulo "comun_enviar_recibir" y desde allí es accesible por el resto de los módulos del ratón.
- **ABORTAR**: Si durante el proceso de recepción el ordenador quiere abortar por algún motivo la recepción del byte, el módulo "recibir_byte" activará la señal "abortar". El ordenador para la recepción al llevar la línea de reloj al nivel lógico cero.
- **ERROR**: Si se ha recibido un byte pero no es un byte válido (error de paridad, etc.), el módulo "recibir_byte" activará la señal "error" para generar las señales que transmitirán al ordenador el byte FE (REENVÍO) si el error se ha producido por primera vez o el byte FC (ERROR) si el error se ha producido más de una vez
- **ESCRIBIR_DATO**: Al activar esta señal con un valor uno lógico se activa el buffer de la línea de datos para que muestre en ella el dato especificado en la línea "dato_a_escribir". Si la señal "escribir_dato" está a cero lógico, la línea se dejará en alta impedancia para que el ordenador pueda escribir en ella. Si la señal "escribir_dato" está a uno lógico y la señal

"dato_a_escribir" está a uno lógico estoy realizando una petición al buffer de la línea de datos para que escriba un uno lógico en la línea, en este caso dejará la línea de datos en alta impedancia. Si la señal "escribir_dato" está a uno lógico y la señal "dato_a_escribir" está a cero lógico estoy realizando una petición al buffer de la línea de datos para que escriba un cero lógico en la línea, en este caso llevará la línea de datos al valor cero lógico.

- **DATO_A_ESCRIBIR**: Dato que se escribirá en la línea de datos cuando se active la señal "escribir_dato". Si la señal "dato_a_escribir" toma el valor cero lógico, se llevará la línea de datos al valor cero lógico. Si la señal "dato_a_escribir" toma el valor uno lógico, se dejará la línea de datos en alta impedancia.
- **DATO_LEIDO**: Valor actual de la línea de datos correspondiente al dato leído. Si la línea de datos está en alta impedancia, el dato leído es uno lógico. Si la línea de datos está con un valor cero lógico, el dato leído es cero lógico.
- **ESCRIBIR_RELOJ**: Al activar esta señal con un valor uno lógico se activa el buffer de la línea de reloj para que muestre en ella el dato especificado en la línea "reloj_a_escribir". Si la señal "escribir_reloj" está a cero lógico, la línea se dejará en alta impedancia para que el ordenador pueda escribir en ella. Si la señal "escribir_reloj" está a uno lógico y la señal "reloj_a_escribir" está a uno lógico estoy realizando una petición al buffer de la línea de reloj para que escriba un uno lógico en la línea, en este caso dejará la línea de reloj en alta impedancia. Si la señal "escribir_reloj" está a uno lógico y la señal "reloj_a_escribir" está a cero lógico estoy realizando una petición al buffer de la línea de reloj para que escriba un cero lógico en la línea, en este caso llevará la línea de reloj al valor cero lógico.
- **RELOJ_LEIDO**: Valor actual de la línea de reloj correspondiente al reloj leído. Si la línea de reloj está en alta impedancia, el reloj leído es uno lógico. Si la línea de reloj está con un valor cero lógico, el reloj leído es cero lógico.
- **RELOJ_A_ESCRIBIR**: Valor del reloj que se escribirá en la línea de reloj cuando se active la señal "escribir_reloj". Si la señal "reloj_a_escribir" toma el valor cero lógico, se llevará la línea de reloj al valor cero lógico. Si la señal "reloj_a_escribir" toma el valor uno lógico, se dejará la línea de reloj en alta impedancia.
- **RECIBIENDO_BYTE**: Al activarse esta señal con un valor uno lógico se le indica al módulo "comun_enviar_recibir" que se está recibiendo un byte. Así se puede discriminar la fuente de datos que se guardarán en los registros del módulo. Se guardará el dato leído si se está recibiendo un byte o se guardará el byte que se quiere transmitir si se va a enviar un byte. Así podré tener almacenado en este módulo el byte recibido desde el ordenador.
- **RESET_CONTADOR_TEMPORAL**: Esta señal la activa el módulo "recibir_byte" cuando quiere resetear el valor del contador temporal para posteriormente iniciar una nueva espera.
- **INCREMENTA_CONTADOR_TEMPORAL**: Esta señal la activa el módulo "recibir_byte" cuando quiere incrementar el valor del contador temporal. El incremento sólo se aplicará en el flanco de bajada del reloj de 5 us. Cuando quiera incrementarse el valor del contador, se mantendrá activa esta señal con un valor uno lógico. En el flanco de bajada del reloj se producirá el incremento. Se mantendrá activa la señal hasta que el valor del contador sea el adecuado según el tiempo asignado en la espera. El contador temporal se utiliza en procesos de espera, siendo los tiempos de espera múltiplos de 5 us.
- **RESET_CONTADOR_INDICE**: Esta señal la activa el módulo "recibir_byte" cuando quiere resetear el valor del contador de índice. El contador índice lleva la cuenta del bit actual que se está tratando en el proceso de recepción del protocolo serie PS/2: 0 si es el bit0, 1 si es

el bit1, 2 si es el bit2, 3 si es el bit3, 4 si es el bit4, 5 si es el bit5, 6 si es el bit6, 7 si es el bit7, 8 si es el bit de paridad y 9 si se está a la espera del bit de parada.

- **INCREMENTA_CONTADOR_INDICE**: Esta señal la activa el módulo "recibir_byte" cuando quiere incrementar el valor del contador de índice. El contador índice lleva la cuenta del bit actual que se está tratando en el proceso de recepción del protocolo serie PS/2: 0 si es el bit0, 1 si es el bit1, 2 si es el bit2, 3 si es el bit3, 4 si es el bit4, 5 si es el bit5, 6 si es el bit6, 7 si es el bit7, 8 si es el bit de paridad y 9 si se está a la espera del bit de parada.
- **CUENTA_INDICE**: El contador índice lleva la cuenta del bit actual que se está tratando en el proceso de recepción del protocolo serie PS/2: 0 si es el bit0, 1 si es el bit1, 2 si es el bit2, 3 si es el bit3, 4 si es el bit4, 5 si es el bit5, 6 si es el bit6, 7 si es el bit7, 8 si es el bit de paridad y 9 si se está a la espera del bit de parada.
- **CUENTA_TEMPORAL_0011**: Esta señal se activará cuando la cuenta del contador temporal indique 3 (15 us).
- **CUENTA_TEMPORAL_0100**: Esta señal se activará cuando la cuenta del contador temporal indique 4 (20 us).
- **CUENTA_TEMPORAL_0101**: Esta señal se activará cuando la cuenta del contador temporal indique 5 (25 us).
- **CUENTA_TEMPORAL_1000**: Esta señal se activará cuando la cuenta del contador temporal indique 8 (40 us).
- **CUENTA_TEMPORAL_1001**: Esta señal se activará cuando la cuenta del contador temporal indique 9 (45 us).
- **CUENTA_TEMPORAL_1100**: Esta señal se activará cuando la cuenta del contador temporal indique 12 (60 us).
- **CUENTA_INDICE_1001**: Esta señal se activará cuando la cuenta del contador del índice mantenga el valor 9.
- **VECTOR_CARGA_BYTE_RECIBIDO**: Este vector de bits es el vector de las señales de carga de los distintos registros donde se almacenarán los bits que conforman el byte recibido. En el proceso de recepción la línea de datos del protocolo serie PS/2 marca el valor del bit actual en el byte recibido, esto es así porque es un protocolo serie. Este valor se almacenará en el registro correspondiente según el registro que se haya activado con el vector de carga "vector_carga_byte_recibido". Ejemplo: Si el bit número 5 del vector de carga se activa, se guardará el dato leído en el registro correspondiente al bit número 5 del byte (suponiendo que en el byte el número 0 se corresponde con el bit menos significativo y el bit número 7 se corresponde con el bit más significativo).
- **BIT_PARIDAD**: El bit de paridad "bit_paridad" es tal que el número de bits con valor uno lógico del byte recibido y almacenado en el módulo "comun_enviar_recibir" junto con el bit de paridad es impar. Ejemplos: Si el byte no tiene ningún bit con valor uno lógico, el bit de paridad será uno lógico. Si el byte tiene tres bits con valor uno lógico, el bit de paridad será cero lógico. Si es el ordenador el que envía el byte, se utilizará la paridad para detectar una variación en uno de los bits del byte (error en la transmisión).

1.2.3.3 comun_enviar_recibir.vhd

```
ENTITY comun_enviar_recibir IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    flanco_bajada_5us: IN STD_LOGIC;
    carga_byte_a_enviar: IN STD_LOGIC;
    byte_a_enviar: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    recibiendo_byte: IN STD_LOGIC;
    vector_carga_byte_recibido: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dato_leido: IN STD_LOGIC;
    byte_enviar_recibir: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    incrementa_contador_temporal_enviar: IN STD_LOGIC;
    reset_contador_temporal_enviar: IN STD_LOGIC;
    incrementa_contador_temporal_recibir: IN STD_LOGIC;
    reset_contador_temporal_recibir: IN STD_LOGIC;
    incrementa_contador_indice_enviar: IN STD_LOGIC;
    reset_contador_indice_enviar: IN STD_LOGIC;
    incrementa_contador_indice_recibir: IN STD_LOGIC;
    reset_contador_indice_recibir: IN STD_LOGIC;
    cuenta_indice: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
    cuenta_temporal_0011: OUT STD_LOGIC;
    cuenta_temporal_0100: OUT STD_LOGIC;
    cuenta_temporal_0101: OUT STD_LOGIC;
    cuenta_temporal_1000: OUT STD_LOGIC;
    cuenta_temporal_1001: OUT STD_LOGIC;
    cuenta_temporal_1100: OUT STD_LOGIC;
    cuenta_indice_1001: OUT STD_LOGIC;
    cuenta_indice_1011: OUT STD_LOGIC;
    bit_paridad: OUT STD_LOGIC
  );
END comun_enviar_recibir;
```

Los módulos "enviar_byte" y "recibir_byte" utilizan 8 registros de un bit para almacenar el byte a enviar o recibido en el paquete de datos del protocolo serie PS/2. También utilizan un contador para sincronizar los eventos de la transmisión y la recepción (dispara los distintos eventos que conforman el protocolo serie PS/2).

Al ser un protocolo serie los datos deben transmitirse bit a bit, por lo que es necesario además de un contador que lleve la cuenta del bit que estamos transmitiendo en cada momento. Los registros y los contadores tienen un uso común por los dos dispositivos para así reducir la lógica necesaria en la implementación del dispositivo. Como la recepción y la transmisión son dos procesos incompatibles, los registros y los contadores pueden utilizarse de forma independiente en cada proceso. Así pues, dentro de este módulo existen los siguientes componentes:

- **RetrasoFlancoBajada5us**: La señal "flanco_bajada_5us" se activa durante 200 ns cada 5 us. Es útil a la hora de sincronizar eventos para que se ejecuten según un reloj de 5 us. Debido a que el reloj del sistema es mucho más rápido que el reloj de emisión del protocolo PS/2 es necesario de un mecanismo para generar las señales con una periodicidad de 5 us. Utilizo el flanco de bajada del reloj de 5 us para incrementar el contador temporal que lleva la cuenta de los tiempos entre eventos en la transmisión y recepción de datos con el protocolo PS/2. Como el contador temporal se activa en el flanco de bajada del reloj del sistema de 200 ns y la señal "flanco_bajada_5us" cambia también en el flanco de bajada del reloj del sistema, es necesario un retraso de un semiperiodo de reloj para que cuando el contador se active las señales que lo

hacen se hayan estabilizado ya. Este retraso se consigue con el biestable "RetrasoFlancoBajada5us".

- **ContadorTemporal**: Los módulos "enviar_byte" y "recibir_byte" son los encargados de generar las señales que a través de las líneas de datos y de reloj del protocolo serie PS/2 permitan la transmisión bidireccional de datos entre el ordenador y el dispositivo. El contador "ContadorTemporal" marca los tiempos transcurridos en múltiplos de 5 us para así poder lanzar eventos que generen las distintas señales que conforman el protocolo. Son los módulos anteriores los que incrementan y resetean al contador para así esperar hasta que haya pasado el tiempo que precisen o reiniciar la cuenta para empezar a contar de nuevo. Este contador funciona con el reloj del sistema de 200 ns y la sincronización con el reloj de 5 us se consigue imponiendo que sólo se pueda incrementar en el flanco de bajada del reloj de 5 us.

- **ContadorIndice**: Al ser un protocolo serie los datos deben transmitirse bit a bit, por lo que es necesario además de un contador que lleve la cuenta del bit que estamos transmitiendo en cada momento. Es el contador "ContadorIndice" el que realiza esta función. Los módulos "enviar_byte" y "recibir_byte" son los encargados de incrementar y resetear al contador en los procesos de transmisión y recepción.

- **ComponenteCalculaParidad**: Este componente genera el bit de paridad del byte que se ha recibido o del byte que se va a enviar. El protocolo serie PS/2 se basa en el envío de datos binario a través de una línea de datos. Se transmiten en paquetes de ocho bits y cada bit sincronizado con un reloj que se transmite por la línea de reloj. Cada byte transmitido se envía junto con un bit de paridad impar útil para detectar posibles errores. El bit de paridad "bit_paridad" es tal que el número de bits con valor uno lógico del byte "byte_entrada" junto con el bit de paridad es impar. Ejemplos: Si el byte no tiene ningún bit con valor uno lógico, el bit de paridad será uno lógico. Si el byte tiene tres bits con valor uno lógico, el bit de paridad será cero lógico. Si es el ordenador el que envía el byte, se utilizará la paridad para detectar una variación en uno de los bits del byte. Si es el dispositivo el que envía el byte, se utilizará la paridad para transmitir su valor por la línea de datos junto al bit de paridad.

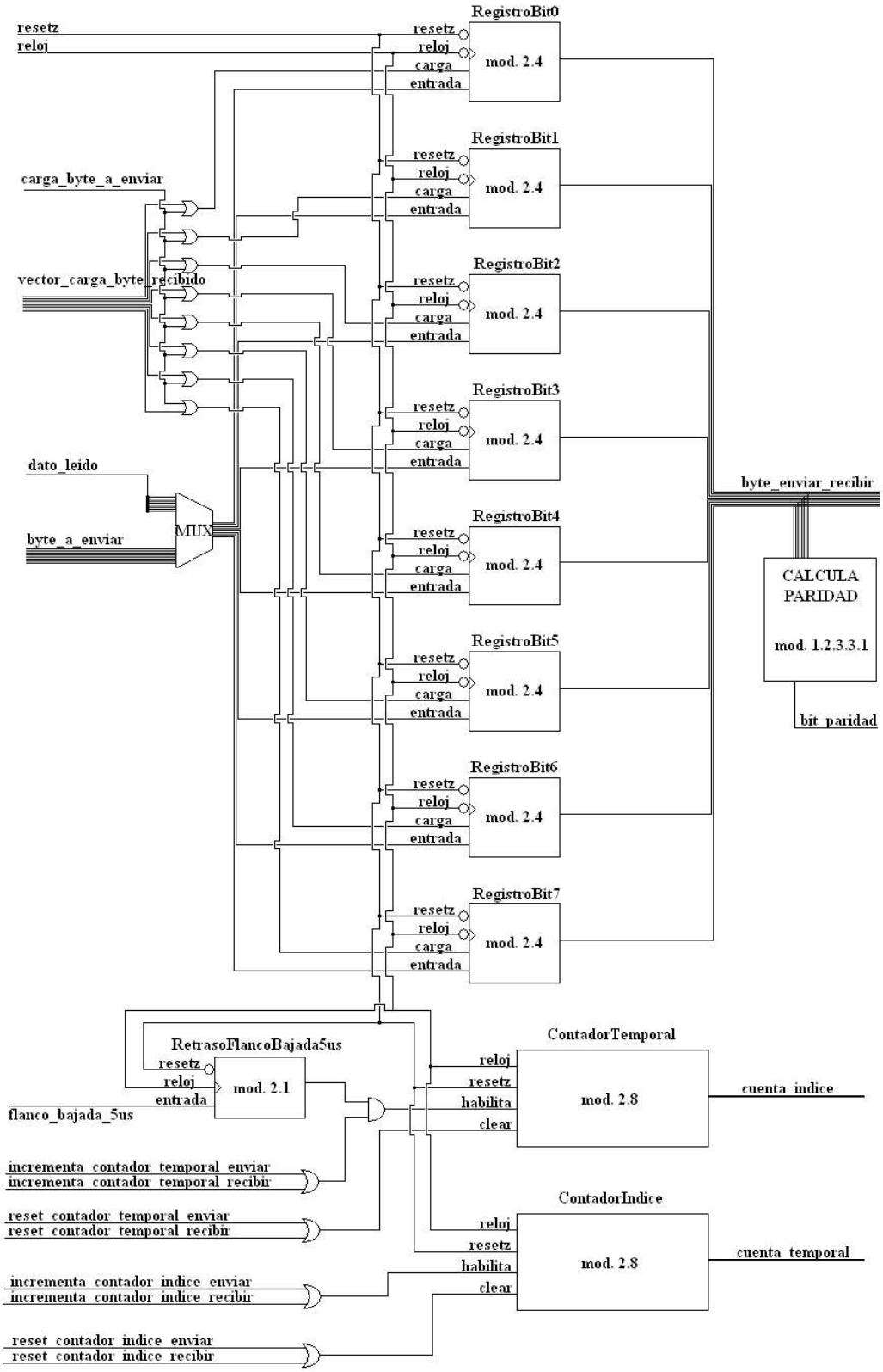
- **RegistroBit0**: En este registro se almacena el bit 0 del byte que se transmitirá al ordenador o el bit 0 del byte recibido desde el ordenador.

- **RegistroBit1**: En este registro se almacena el bit 1 del byte que se transmitirá al ordenador o el bit 1 del byte recibido desde el ordenador.

- **RegistroBit2**: En este registro se almacena el bit 2 del byte que se transmitirá al ordenador o el bit 2 del byte recibido desde el ordenador.

- **RegistroBit3**: En este registro se almacena el bit 3 del byte que se transmitirá al ordenador o el bit 3 del byte recibido desde el ordenador.

- **RegistroBit4**: En este registro se almacena el bit 4 del byte que se transmitirá al ordenador o el bit 4 del byte recibido desde el ordenador.


resetz
reloj
carga_byte_a_enviar
vector_carga_byte_recibido
dato_leido
byte_a_enviar
MUX
RegistroBit0
RegistroBit1
RegistroBit2
RegistroBit3
RegistroBit4
RegistroBit5
RegistroBit6
RegistroBit7
resetz
reloj
carga
entrada
mod. 2.4
byte_enviar_recibir
CALCULA PARIDAD
mod. 1.2.3.3.1
bit_paridad
RetrasoFlancoBajada5us
mod. 2.1
flanco_bajada_5us
ContadorTemporal
habilita
clear
mod. 2.8
cuenta_indice
incrementa_contador_temporal_enviar
incrementa_contador_temporal_recibir
reset_contador_temporal_enviar
reset_contador_temporal_recibir
ContadorIndice
cuenta_temporal
incrementa_contador_indice_enviar
incrementa_contador_indice_recibir
reset_contador_indice_enviar
reset_contador_indice_recibir

- **RegistroBit6**: En este registro se almacena el bit 6 del byte que se transmitirá al ordenador o el bit 6 del byte recibido desde el ordenador.

- **RegistroBit7**: En este registro se almacena el bit 7 del byte que se transmitirá al ordenador o el bit 7 del byte recibido desde el ordenador.

INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **FLANCO_BAJADA_5US**: El contador temporal utiliza el flanco de bajada del reloj para sincronizar el incremento del contador. Este contador funciona con el reloj del sistema de 200 ns y la sincronización con el reloj de 5 us se consigue imponiendo que sólo se pueda incrementar en el flanco de bajada del reloj de 5 us. La señal "flanco_bajada_5us" se activa durante 200 ns cada 5 us. Es útil a la hora de sincronizar eventos para que se ejecuten según un reloj de 5 us. Debido a que el reloj del sistema es mucho más rápido que el reloj de emisión del protocolo PS/2 es necesario de un mecanismo para generar las señales con una periodicidad de 5 us.
- **CARGA_BYTE_A_ENVIAR**: Cuando se quiera enviar un byte la activación de esta señal guardará en los registros del módulo el byte que se quiere transmitir "byte_a_enviar".
- **BYTE_A_ENVIAR**: Byte que se almacenará en los registros del módulo cuando se active la señal "carga_byte_a_enviar" con un valor uno lógico.
- **RECIBIENDO_BYTE**: Al activarse esta señal con un valor uno lógico se le indica al dispositivo que se está recibiendo un byte. Así se puede discriminar la fuente de datos que se guardarán en los registros del módulo. Se guardará el dato leído si se está recibiendo un byte o se guardará el byte que se quiere transmitir si se va a enviar un byte.
- **VECTOR_CARGA_BYTE_RECIBIDO**: Este vector de bits es el vector de las señales de carga de los distintos registros donde se almacenarán los bits que conforman el byte recibido. En el proceso de recepción la línea de datos del protocolo serie PS/2 marca el valor del bit actual en el byte recibido, esto es así porque es un protocolo serie. Este valor se almacenará en el registro correspondiente según el registro que se haya activado con el vector de carga "vector_carga_byte_recibido". Ejemplo: Si el bit número 5 del vector de carga se activa, se guardará el dato leído en el registro correspondiente al bit número 5 del byte (suponiendo que en el byte el número 0 se corresponde con el bit menos significativo y el bit número 7 se corresponde con el bit más significativo).
- **DATO_LEIDO**: En el proceso de recepción la línea de datos del protocolo serie PS/2 marca el valor del bit del byte actual, esto es así porque es un protocolo serie. La señal "dato_leido" es el valor de la línea de datos. Este valor ("dato_leido") se almacenará en el registro correspondiente según el registro que se haya activado con el vector de carga "vector_carga_byte_recibido".
- **BYTE_ENVIAR_RECIBIR**: Byte almacenado en los registros del módulo. Este byte se corresponde con el byte recibido si el proceso que ha almacenado el byte es el de recepción.

Se corresponde con el byte emitido si el proceso que ha almacenado el byte es el de transmisión.

- **INCREMENTA_CONTADOR_TEMPORAL_ENVIAR**: Esta señal la activa el módulo "enviar_byte" cuando quiere incrementar el valor del contador temporal. El incremento sólo se aplicará en el flanco de bajada del reloj de 5 us. Cuando quiera incrementarse el valor del contador, se mantendrá activa esta señal con un valor uno lógico. En el flanco de bajada del reloj se producirá el incremento. Se mantendrá activa la señal hasta que el valor del contador sea el adecuado según el tiempo asignado en la espera. El contador temporal se utiliza en procesos de espera, siendo los tiempos de espera múltiplos de 5 us.
- **RESET_CONTADOR_TEMPORAL_ENVIAR**: Esta señal la activa el módulo "enviar_byte" cuando quiere resetear el valor del contador temporal para posteriormente iniciar una nueva espera.
- **INCREMENTA_CONTADOR_TEMPORAL_RECIBIR**: Esta señal la activa el módulo "recibir_byte" cuando quiere incrementar el valor del contador temporal. El incremento sólo se aplicará en el flanco de bajada del reloj de 5 us. Cuando quiera incrementarse el valor del contador, se mantendrá activa esta señal con un valor uno lógico. En el flanco de bajada del reloj se producirá el incremento. Se mantendrá activa la señal hasta que el valor del contador sea el adecuado según el tiempo asignado en la espera. El contador temporal se utiliza en procesos de espera, siendo los tiempos de espera múltiplos de 5 us.
- **RESET_CONTADOR_TEMPORAL_RECIBIR**: Esta señal la activa el módulo "recibir_byte" cuando quiere resetear el valor del contador temporal para posteriormente iniciar una nueva espera.
- **INCREMENTA_CONTADOR_INDICE_ENVIAR**: Esta señal la activa el módulo "enviar_byte" cuando quiere incrementar el valor del contador de índice. El contador índice lleva la cuenta del bit enviado o recibido con el protocolo serie PS/2.
- **RESET_CONTADOR_INDICE_ENVIAR**: Esta señal la activa el módulo "enviar_byte" cuando quiere resetear el valor del contador de índice. El contador índice lleva la cuenta del bit enviado o recibido con el protocolo serie PS/2.
- **INCREMENTA_CONTADOR_INDICE_RECIBIR**: Esta señal la activa el módulo "recibir_byte" cuando quiere incrementar el valor del contador de índice. El contador índice lleva la cuenta del bit enviado o recibido con el protocolo serie PS/2.
- **RESET_CONTADOR_INDICE_RECIBIR**: Esta señal la activa el módulo "recibir_byte" cuando quiere resetear el valor del contador de índice. El contador índice lleva la cuenta del bit enviado o recibido con el protocolo serie PS/2.
- **CUENTA_INDICE**: Valor actual del contador índice, que lleva la cuenta del bit enviado o recibido con el protocolo serie PS/2.
- **CUENTA_TEMPORAL_0011**: Esta señal se activará cuando la cuenta del contador temporal indique 3 (15 us).
- **CUENTA_TEMPORAL_0100**: Esta señal se activará cuando la cuenta del contador temporal indique 4 (20 us).
- **CUENTA_TEMPORAL_0101**: Esta señal se activará cuando la cuenta del contador temporal indique 5 (25 us).
- **CUENTA_TEMPORAL_1000**: Esta señal se activará cuando la cuenta del contador temporal indique 8 (40 us).
- **CUENTA_TEMPORAL_1001**: Esta señal se activará cuando la cuenta del contador temporal indique 9 (45 us).

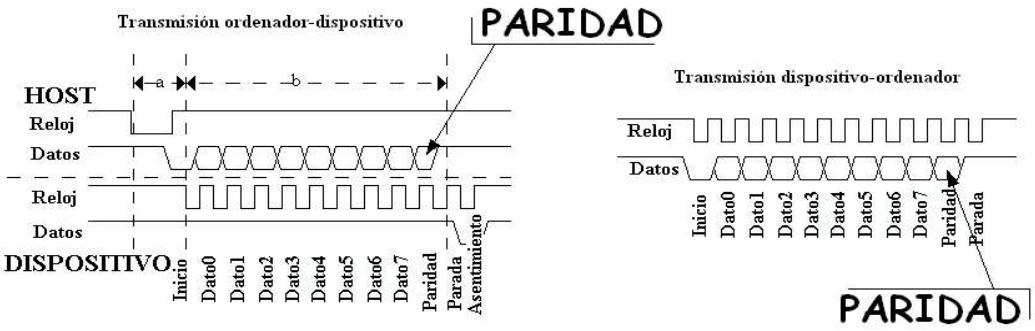
- **CUENTA_TEMPORAL_1100**: Esta señal se activará cuando la cuenta del contador temporal indique 12 (60 us).
- **CUENTA_INDICE_1001**: Esta señal se activará cuando la cuenta del contador del índice mantenga el valor 9.
- **CUENTA_INDICE_1011**: Esta señal se activará cuando la cuenta del contador del índice mantenga el valor 11.
- **BIT_PARIDAD**: El bit de paridad "bit_paridad" es tal que el número de bits con valor uno lógico del byte "byte_entrada" junto con el bit de paridad es impar. Ejemplos: Si el byte no tiene ningún bit con valor uno lógico, el bit de paridad será uno lógico. Si el byte tiene tres bits con valor uno lógico, el bit de paridad será cero lógico. Si es el ordenador el que envía el byte, se utilizará la paridad para detectar una variación en uno de los bits del byte. Si es el dispositivo el que envía el byte, se utilizará la paridad para transmitir su valor por la línea de datos junto al bit de paridad.

1.2.3.3.1 calcula_paridad.vhd

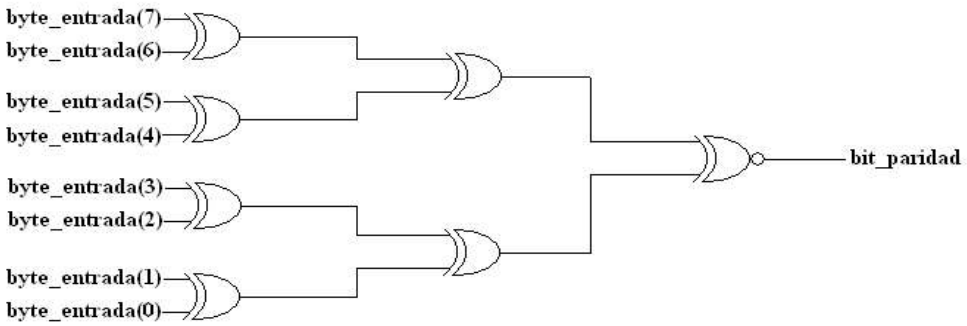
```
ENTITY calcula_paridad is
  PORT(
    byte_entrada: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    bit_paridad: OUT STD_LOGIC
  );
END calcula_paridad;
```

Este módulo genera el bit de paridad impar del byte de entrada.

El protocolo serie PS/2 se basa en el envío de datos binario a través de una línea de datos. Se transmiten en paquetes de ocho bits y cada bit sincronizado con un reloj que se transmite por la línea de reloj. Cada byte transmitido se envía junto con un bit de paridad impar útil para detectar posibles errores. El bit de paridad "bit_paridad" es tal que el número de bits con valor uno lógico del byte "byte_entrada" junto con el bit de paridad es impar. Ejemplos: Si el byte no tiene ningún bit con valor uno lógico, el bit de paridad será uno lógico. Si el byte tiene tres bits con valor uno lógico, el bit de paridad será cero lógico.



Si es el ordenador el que envía el byte, se utilizará la paridad para detectar una variación en uno de los bits del byte. Si es el dispositivo el que envía el byte, se utilizará la paridad para transmitir su valor por la línea de datos junto al bit de paridad.



1.2.3.4 sistema.vhd

```
ENTITY sistema IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    linea_reloj: IN STD_LOGIC;
    linea_datos: IN STD_LOGIC;
    fin_inicializacion: IN STD_LOGIC;
    fin_lectura: IN STD_LOGIC;
    fin_envio: IN STD_LOGIC;
    enviar_datos_sistema: IN STD_LOGIC;
    abortar_lectura: IN STD_LOGIC;
    abortar_envio: IN STD_LOGIC;
    inicio_inicializacion: OUT STD_LOGIC;
    inicio_lectura: OUT STD_LOGIC;
    inicio_envio: OUT STD_LOGIC
  );
END sistema;
```

Módulo encargado de gestionar el funcionamiento del sistema-ratón de manera global. En un principio activará el proceso de inicialización mediante el cual el ratón envía una serie de señales al host (0xAA y 0x00) con el fin de que detecte y reconozca al dispositivo. Una vez terminado el proceso de inicialización (MODO RESET) alternará su ocupación entre el de envío de datos y la recepción de datos.

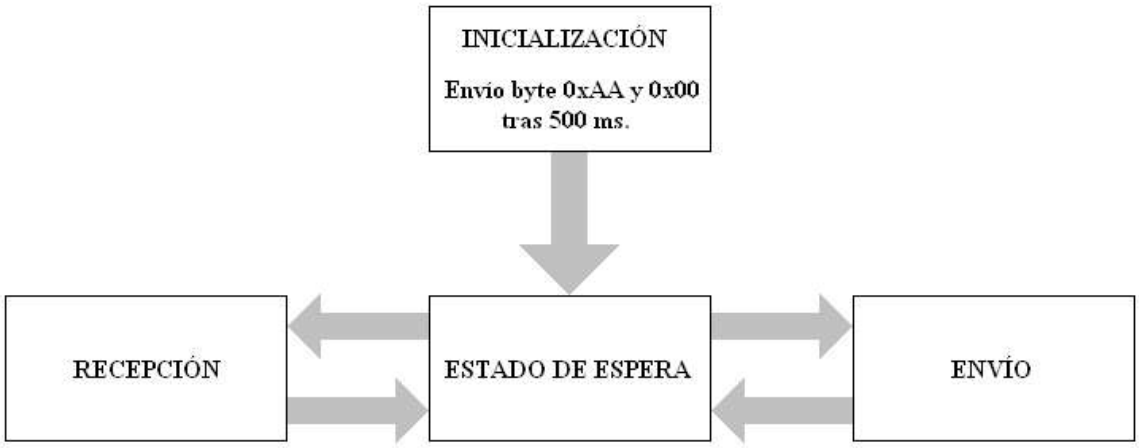
El módulo sistema es el núcleo del sistema ratón. En un principio, tras la activación inicial o al recibir el comando "Reset" (ØxFF) el módulo "sistema" activa la señal "inicio_inicializacion" que pondrá en funcionamiento los mecanismos desarrollados en el "modo reset". El módulo "inicialización" es el encargado de realizar los procesos de inicialización, básicamente es el envío de las señales 0xAA y 0x00 para que el host detecte y reconozca al dispositivo.

Posteriormente entrará en la fase de funcionamiento normal del dispositivo, donde esperará hasta que el ordenador decida enviar un byte o se haya creado un evento en el dispositivo que pretenda iniciar el envío de un byte. En este estado de espera y envío o espera y recepción permanecerá hasta que se reciba el comando 0xFF (RESET), el cual provocará que el sistema vuelva al "modo reset" para iniciar un nuevo proceso de inicialización. En conclusión el estado de espera es el estado corriente de funcionamiento y es el estado al que siempre volverá tras un envío de datos o tras la recepción de un byte de datos, a menos que la recepción se corresponda con el comando 0xFF. En este caso tras enviar el byte de asentimiento 0xFA se reseteará el módulo "sistema".

El ordenador solicita al dispositivo que active las señales de reloj para poder enviar un byte (comando o dato) llevando la línea de datos a un valor cero lógico y dejando la línea de datos en alta impedancia. De esta forma el dispositivo reconoce que el host quiere enviar un byte y activará la señal "inicio_lectura" que pondrá en marcha el procedimiento de recepción de un byte. Volverá al estado de espera cuando termine la recepción del byte o si el ordenador aborta la recepción manteniendo la línea de reloj a nivel cero lógico.

Si alguno de los flags de la memoria de datos solicita el envío de datos (flag de asentimiento, flag del primer registro de respuesta, flag del segundo registro de respuesta o flag del tercer registro

de respuesta) estando las líneas de datos y de reloj en alta impedancia, se activará la señal "inicio_envio" que pondrá en marcha el procedimiento de envío del byte correspondiente al flag de mayor prioridad. Volverá al estado de espera cuando termine el envío del byte o si el ordenador aborta el envío manteniendo la línea de reloj a nivel cero lógico.



Una máquina de estados controla el funcionamiento del módulo. Estos son los posibles estados que se pueden activar:

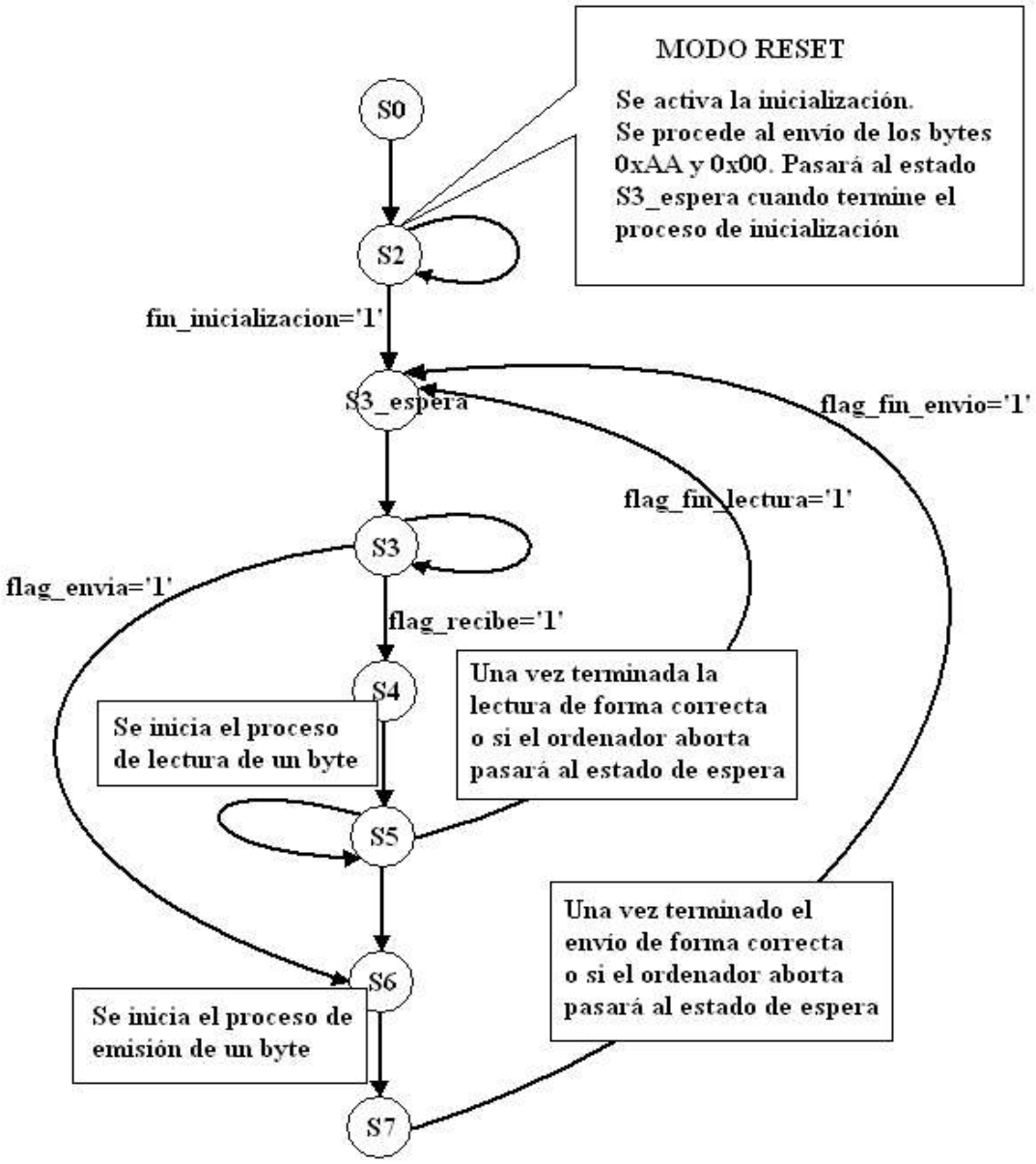
- **S0**: Este es un estado de transición cuya misión es dejar pasar cierto tiempo para estabilizar el sistema antes de iniciar cualquier proceso. En el flanco de subida del reloj pasará al estado S2.

- **S2**: En este estado se activa la señal "inicio_inicializacion" con un valor uno lógico para que el módulo inicialización envíe tras 500 milisegundos los byte 0xAA y 0x00. Esperará en este estado hasta que se finalice el procedimiento de inicialización, entonces pasará al estado S3_ESPERA.

- **S3_ESPERA**: Este es un estado de transición cuya misión es dejar pasar cierto tiempo para estabilizar el sistema antes de entrar en el estado de espera global. En el flanco de subida del reloj pasará al estado S3.

- **S3**: Estado de espera global en el que permanecerá hasta que el ordenador solicite al dispositivo que active las señales de reloj para poder enviar un byte (comando o dato) llevando la línea de datos a un valor cero lógico y dejando la línea de datos en alta impedancia. Pasará entonces al estado S4 para iniciar el proceso de recepción de datos. También saldrá del estado de espera global si alguno de los flags de la memoria de datos solicita el envío de datos (flag de asentimiento, flag del primer registro de respuesta, flag del segundo registro de respuesta o flag del tercer registro de respuesta) estando las líneas de

datos y de reloj en alta impedancia. Pasará entonces al estado S6 para iniciar el proceso de envío de datos.



- **S4**: Se activa la señal "inicio_lectura" para poner en marcha el proceso de recepción de un byte desde el ordenador. Pasará al estado S5.

- **S5**: En este estado permanecerá hasta que concluya el proceso de recepción del byte enviado por el ordenador. Se considerará concluido el proceso cuando el byte haya sido recibido de forma correcta o cuando el ordenador aborte la recepción. Tras el fin del proceso de recepción el ratón pasará al estado S3_ESPERA para entrar de nuevo en el estado de espera global.

- **S6**: Se activa la señal "inicio_envio" para poner en marcha el proceso de envío de un byte hacia el ordenador. Pasará al estado S7.

- **S7**: En este estado permanecerá hasta que concluya el proceso de envío del byte propuesto por el flag que activó la transmisión. Se considerará concluido el proceso cuando el byte haya sido enviado de forma correcta o cuando el ordenador aborte el envío. Tras el fin del proceso de envío el ratón pasará al estado S3_ESPERA para entrar de nuevo en el estado de espera global.

INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **LINEA_RELOJ**: Línea de reloj del protocolo serie PS/2.
- **LINEA_DATOS**: Línea de datos del protocolo serie PS/2.
- **FIN_INICIALIZACION**: Esta señal la activará el módulo "inicializacion" cuando termine de enviar los bytes 0xAA y 0x00.
- **FIN_LECTURA**: Se activará cuando la lectura de un byte haya sido concluida de forma satisfactoria.
- **FIN_ENVIO**: Se activará cuando el envío de un byte haya sido concluido de forma correcta.
- **ENVIAR_DATOS_SISTEMA**: Cuando uno de los flags de los bytes de envío esté activado (flag de asentimiento FA, flag del registro de la primera respuesta, flag del registro de la segunda respuesta o flag del registro de la tercera respuesta) se avisará al módulo "sistema" activando esta señal para que inicie el proceso de envío cuando éste sea factible.
- **ABORTAR_LECTURA**: Se activa esta señal cuando la lectura de un byte haya sido abortada por el ordenador.
- **ABORTAR_ENVIO**: Se activa esta señal cuando el envío de un byte haya sido abortado por el ordenador.
- **INICIO_INICIALIZACION**: La máquina de estados del módulo "sistema" activará esta señal para iniciar el proceso de inicialización. Comenzará así el envío de los bytes 0xAA y 0x00 tras 500 milisegundos.
- **INICIO_LECTURA**: La máquina de estados del módulo "sistema" activará esta señal para iniciar el proceso de recepción de un byte desde el ordenador.
- **INICIO_ENVIO**: La máquina de estados del módulo "sistema" activará esta señal para iniciar el proceso de envío del byte cuyo flag desencadenó la petición de envío.

1.2.3.5 inicializacion.vhd

```
ENTITY inicializacion IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    cuenta_global: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
    inicio_inicializacion: IN STD_LOGIC;
    aborto_envio: IN STD_LOGIC;
    fin_envio: IN STD_LOGIC;
    byte_a_enviar: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    enviar_byte: OUT STD_LOGIC;
    fin_inicializacion: OUT STD_LOGIC
  );
END inicializacion;
```

Este módulo implementa el funcionamiento del dispositivo ratón en el "modo reset". Aproximadamente 500 milisegundos después del encendido del dispositivo ratón, se deben transmitir los bytes ØxAA y ØxØØ.

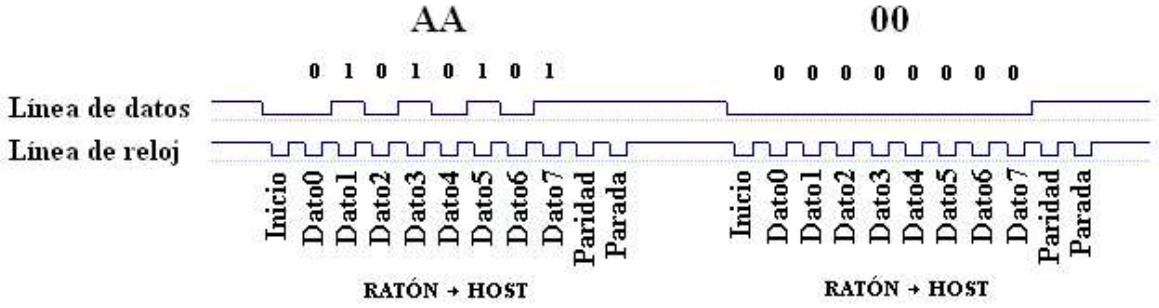
El ratón entra en el modo Reset en la activación inicial o al recibir el comando "Reset" (ØxFF). Después de entrar en este modo de operación, el ratón realiza una auto-prueba de diagnóstico llamada BAT (Basic Assurance Test) y activa los siguientes valores por defecto:

1) Frecuencia de muestreo = 100 muestras/segundo
2) Resolución = 4 cuentas/mm
3) Escala = 1:1
4) Envío de datos deshabilitado

Aproximadamente 500 milisegundos después del encendido enviará el código de terminación de BAT con éxito (ØxAA) o con error (ØxFC). Si el host recibe otra respuesta que no sea ØxAA, puede completar un ciclo de alimentación del ratón (fuente de alimentación), causando en el ratón el Reset y la reejecución de la prueba de diagnóstico BAT.

Después del código de terminación de la prueba BAT (ØxAA o ØxFC), el ratón envía el identificador de dispositivo ID ØxØØ. Esto lo distingue de un teclado o de un ratón en modo extendido. El host no espera la transmisión de ningún dato hasta recibir el identificador de dispositivo ID. Sin embargo, algunas BIOS enviarán el comando "Reset" (ØxFF) inmediatamente después de recibir ØxAA tras un reinicio.

Después de que el ratón haya enviado su identificador de dispositivo ID al host, entrará en el modo Corriente. Hay que advertir que uno de los valores por defecto en el ratón es "Envío de datos deshabilitado". Esto significa que el ratón no enviará ningún paquete de datos de movimiento hasta que reciba el comando de "Habilitación del envío de datos" (ØxF4).



Una máquina de estados controla el funcionamiento del dispositivo en el "modo_reset". Puede encontrarse en uno de los siguientes estados:

- **S0**: Estado de reposo en el que permanecerá el dispositivo a la espera de la activación de la señal "inicio_inicializacion". Tras la activación inicial o al recibir el comando "Reset" (ØxFF) el módulo "sistema" activa la señal "inicio_inicializacion" que pondrá en funcionamiento los mecanismos desarrollados en el "modo reset". Pasará entonces al estado S1.

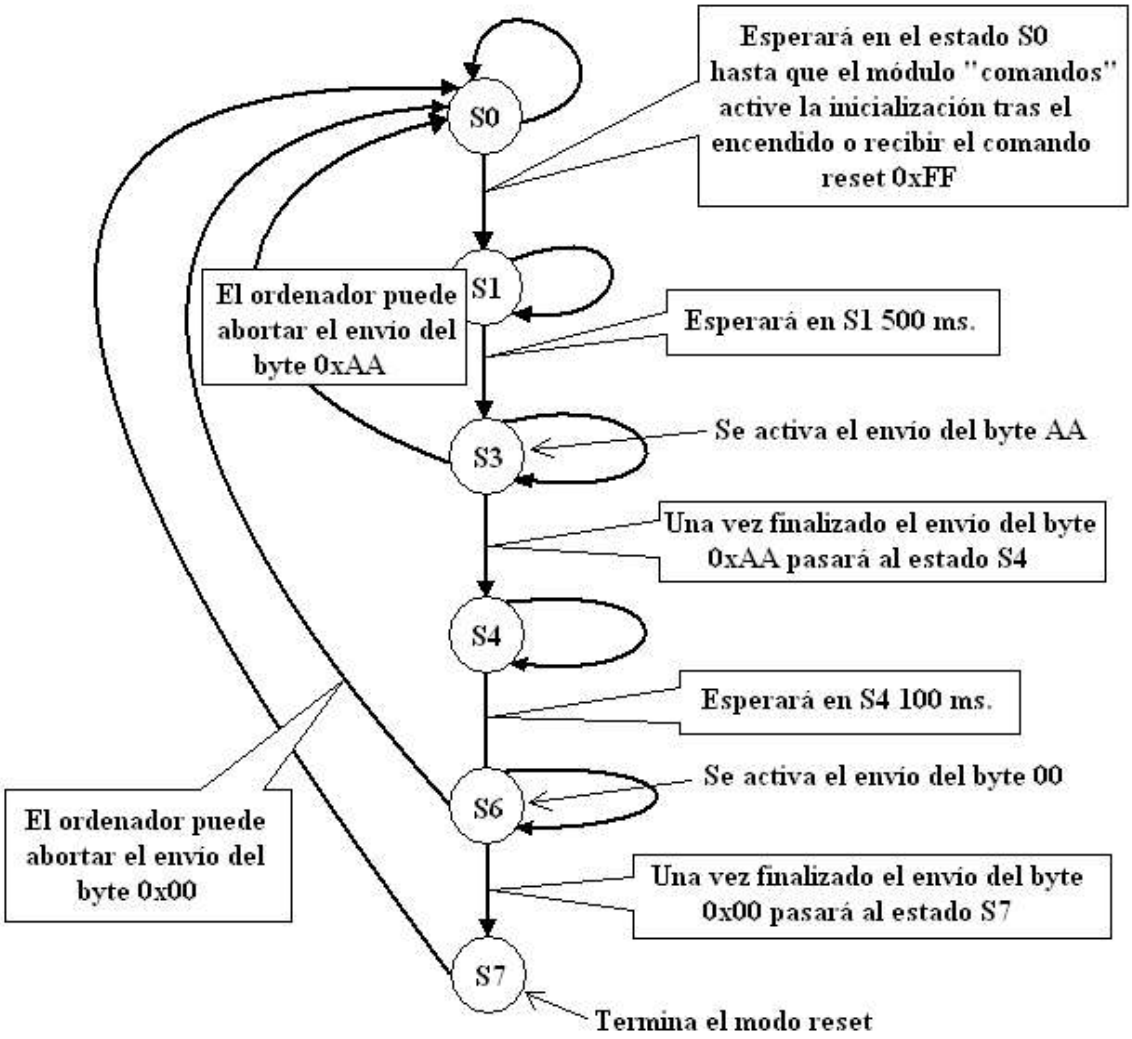
- **S1**: Esperará en este estado aproximadamente 500 milisegundos. Luego pasará al estado S3.

- **S3**: En este estado se activa el envío del byte 0xAA (código de terminación de la prueba BAT) al ordenador. Se espera hasta que el envío se haya concluido de forma correcta, entonces pasará al estado S4. Si por alguna circunstancia el ordenador aborta el envío, el sistema volverá al estado S0 para iniciar de nuevo los mecanismos desarrollados en el "modo reset".

- **S4**: Esperará en este estado aproximadamente 100 milisegundos. Luego pasará al estado S6.

- **S6**: En este estado se activa el envío del byte de identificación de dispositivo 0x00 al ordenador. Se espera hasta que el envío se haya concluido de forma correcta, entonces pasará al estado S7. Si por alguna circunstancia el ordenador aborta el envío, el sistema volverá al estado S0 para iniciar de nuevo los mecanismos desarrollados en el "modo reset".

- **S7**: Activa la señal "fin_inicializacion" para indicar al módulo "sistema" que ha concluido el proceso de inicialización del "modo reset". El sistema pasará entonces al "modo corriente" a la espera de un comando enviado por el ordenador. Inicialmente las variables del dispositivo-ratón tendrán los siguientes valores (valores por defecto):
  Frecuencia de muestreo = 100 muestras/segundo.
  Resolución = 4 cuentas/mm.
  Escala = 1:1.
  Envío de datos deshabilitado.



INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **CUENTA_GLOBAL**: Este es el valor de un contador externo utilizado para poder tener constancia de los tiempos globales transcurridos. De esta forma el dispositivo ratón sincroniza los eventos 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.

- **INICIO_INICIALIZACION**: Tras la activación inicial o al recibir el comando "Reset" (ØxFF) el módulo "sistema" activa la señal "inicio_inicializacion" que pondrá en funcionamiento los mecanismos desarrollados en el "modo reset".
- **ABORTO_ENVIO**: Si el ordenador aborta el envío del código de terminación de la prueba BAT (0xAA) o del byte de identificación de dispositivo 0x00, se activa esta señal para que se reinicie el proceso de inicialización.
- **FIN_ENVIO**: Si la transmisión del código de terminación de la prueba BAT (0xAA) o del byte de identificación de dispositivo 0x00 ha terminado de forma correcta se activa esta señal para continuar el proceso de inicialización.
- **BYTE_A_ENVIAR**: A través de este byte se informará al módulo "enviar_byte" del byte que deberá enviarse al ordenador cuando se active la señal "enviar_byte": el código de terminación de la prueba BAT (0xAA) o el byte de identificación de dispositivo 0x00.
- **ENVIAR_BYTE**: Cuando se activa esta señal se inicia el envío del byte "byte_a_enviar" en el módulo "enviar_byte".
- **FIN_INICIALIZACION**: Indica al módulo "sistema" cuando está activa con un valor uno lógico que ha concluido el proceso de inicialización del "modo reset".

1.2.3.6 memoria_datos.vhd

```
ENTITY memoria_datos IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    reset_contadores_movimiento: IN STD_LOGIC;
    byte_entrada_movimiento_1: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_movimiento_2: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_movimiento_3: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_entrada_movimiento_1: IN STD_LOGIC;
    cambiar_entrada_movimiento_2: IN STD_LOGIC;
    cambiar_entrada_movimiento_3: IN STD_LOGIC;
    reset_respuesta: IN STD_LOGIC;
    byte_entrada_respuesta_1: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_respuesta_2: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_respuesta_3: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_respuesta_1_enviado: IN STD_LOGIC;
    byte_respuesta_2_enviado: IN STD_LOGIC;
    byte_respuesta_3_enviado: IN STD_LOGIC;
    cambiar_entrada_respuesta_1: IN STD_LOGIC;
    cambiar_entrada_respuesta_2: IN STD_LOGIC;
    cambiar_entrada_respuesta_3: IN STD_LOGIC;
    byte_salida_respuesta_1: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_salida_respuesta_2: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_salida_respuesta_3: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_salida_respuesta_1: OUT STD_LOGIC;
    flag_salida_respuesta_2: OUT STD_LOGIC;
    flag_salida_respuesta_3: OUT STD_LOGIC;
    byte_entrada_instruccion: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_entrada_instruccion: IN STD_LOGIC;
    reset_byte_instruccion: IN STD_LOGIC;
    byte_salida_instruccion: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_salida_instruccion: OUT STD_LOGIC;
    enviar_asentimiento_entrada: IN STD_LOGIC;
    asentimiento_enviado: IN STD_LOGIC;
    enviar_asentimiento_salida: OUT STD_LOGIC
  );
END memoria_datos;
```

En este módulo se encuentran los registros que almacenarán los bytes y flags utilizados en la recepción y en el envío de datos del dispositivo ratón.

Registro "ByteSalidaRespuesta1Global":
En este registro se almacena el primer byte de respuesta a un comando recibido desde el ordenador. El ratón procesa el comando y genera una respuesta que será almacenada en este registro. También se almacena en este registro el primer byte del paquete de datos de movimiento si se ha producido un evento de movimiento. Los eventos de movimiento se producen con la frecuencia dada por la frecuencia de muestreo cuando el ratón se encuentra en el modo corriente estando el envío de datos habilitado y siempre que tenga datos que transmitir. Si todavía existe un byte de respuesta pendiente por enviar no se guardarán los datos de movimiento.

Flag "FlagSalidaRespuesta1Global":
Este flag se activa cuando se guarda en el primer registro de respuesta el primer byte de la respuesta a un comando o el primer byte de un paquete de datos de movimiento. Así indicará al sistema encargado de enviar datos al ordenador la presencia de un byte en el registro "ByteSalidaRespuesta1Global" correspondiente al primer byte de respuesta que es preciso enviar.

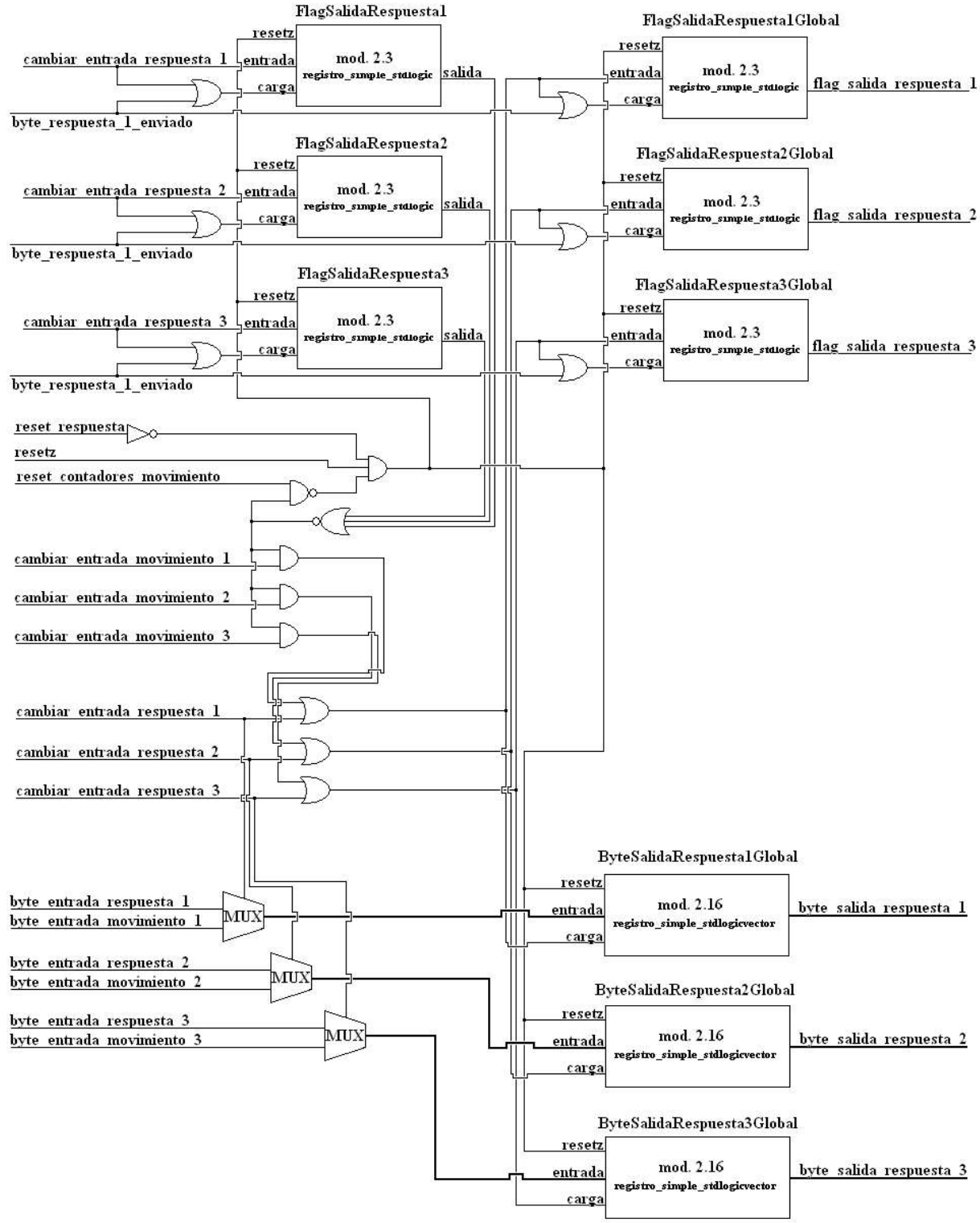
Éste es el byte con más prioridad en la respuesta, así que será el primer byte que se envíe al ordenador. Una vez concluido el envío del byte se desactivará el flag para evitar que vuelva a ser enviado. También se desactivará cuando se reseteen los contadores de movimiento si la respuesta guardada se corresponde al primer byte del paquete de datos de movimiento.

Registro "ByteSalidaRespuesta2Global":
En este registro se almacena el segundo byte de respuesta a un comando recibido desde el ordenador. El ratón procesa el comando y genera una respuesta que será almacenada en este registro. También se almacena en este registro el segundo byte del paquete de datos de movimiento si se ha producido un evento de movimiento. Los eventos de movimiento se producen con la frecuencia dada por la frecuencia de muestreo cuando el ratón se encuentra en el modo corriente estando el envío de datos habilitado y siempre que tenga datos que transmitir. Si todavía existe un byte de respuesta pendiente por enviar no se guardarán los datos de movimiento.

Flag "FlagSalidaRespuesta2Global":
Este flag se activa cuando se guarda en el segundo registro de respuesta el segundo byte de la respuesta a un comando o el segundo byte de un paquete de datos de movimiento. Así indicará al sistema encargado de enviar datos al ordenador la presencia de un byte en el registro "ByteSalidaRespuesta2Global" correspondiente al segundo byte de respuesta que es preciso enviar. Éste es el byte con prioridad media en la respuesta, así que será el segundo byte que se envíe al ordenador. Una vez concluido el envío del byte se desactivará el flag para evitar que vuelva a ser enviado. También se desactivará cuando se reseteen los contadores de movimiento si la respuesta guardada se corresponde al segundo byte del paquete de datos de movimiento.

Registro "ByteSalidaRespuesta3Global":
En este registro se almacena el tercer byte de respuesta a un comando recibido desde el ordenador. El ratón procesa el comando y genera una respuesta que será almacenada en este registro. También se almacena en este registro el tercer byte del paquete de datos de movimiento si se ha producido un evento de movimiento. Los eventos de movimiento se producen con la frecuencia dada por la frecuencia de muestreo cuando el ratón se encuentra en el modo corriente estando el envío de datos habilitado y siempre que tenga datos que transmitir. Si todavía existe un byte de respuesta pendiente por enviar no se guardarán los datos de movimiento.

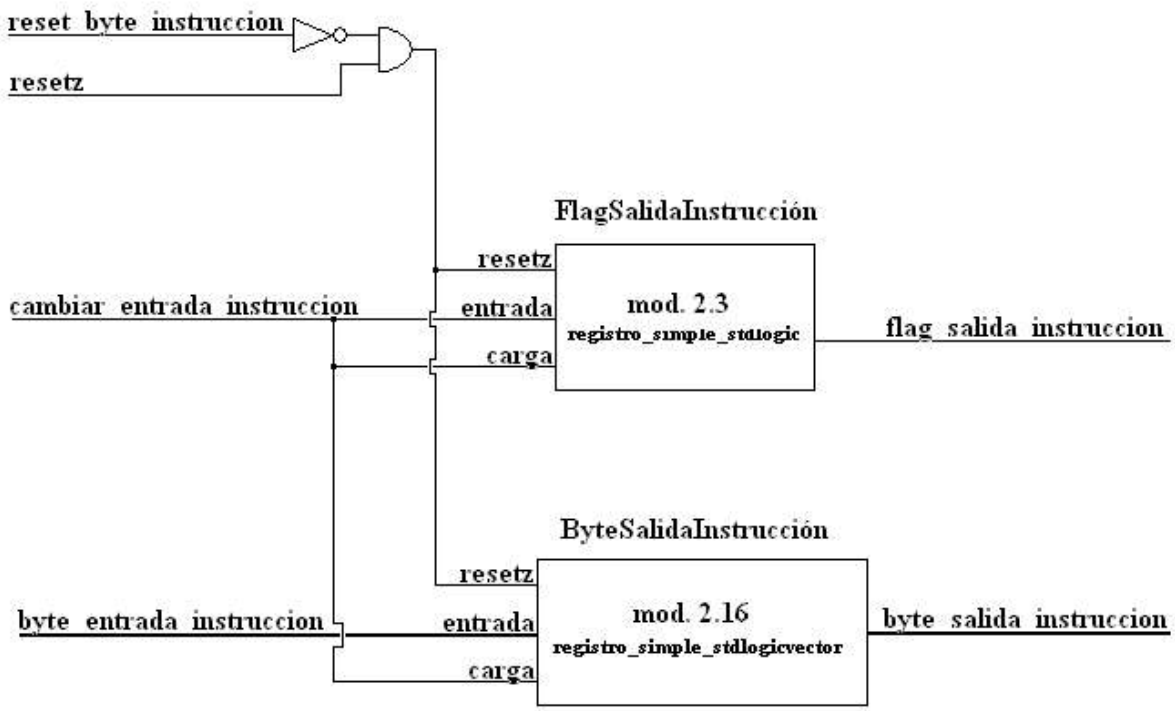
Flag "FlagSalidaRespuesta3Global":
Este flag se activa cuando se guarda en el tercer registro de respuesta el tercer byte de la respuesta a un comando o el tercer byte de un paquete de datos de movimiento. Así indicará al sistema encargado de enviar datos al ordenador la presencia de un byte en el registro "ByteSalidaRespuesta3Global" correspondiente al tercer byte de respuesta que es preciso enviar. Éste es el byte con menos prioridad en la respuesta, así que será el tercer byte que se envíe al ordenador. Una vez concluido el envío del byte se desactivará el flag para evitar que vuelva a ser enviado. También se desactivará cuando se reseteen los contadores de movimiento si la respuesta guardada se corresponde al tercer byte del paquete de datos de movimiento.



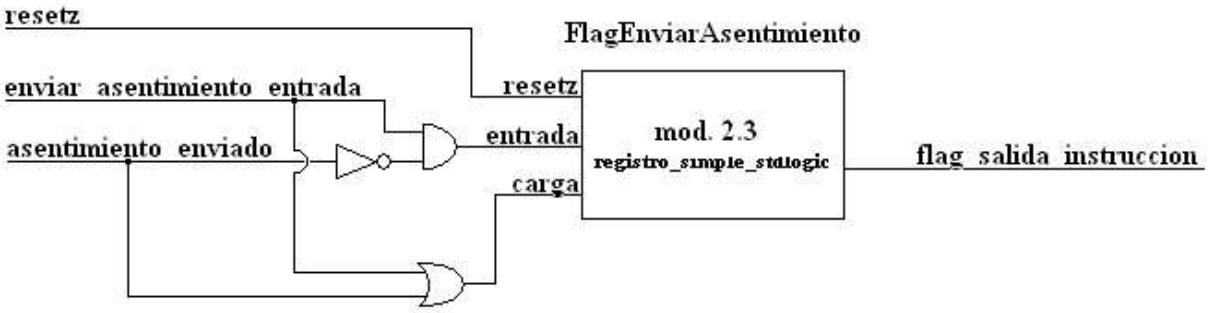
Registro "ByteSalidaInstruccion":

En este registro se guarda el byte que acaba de ser recibido desde el ordenador. Normalmente, se corresponde con un comando o un byte de datos que debe ser procesado. Se guardará cuando se active la señal "cambiar_entrada_instruccion" para el posterior procesamiento del byte.

Flag "FlagSalidaInstruccion":
En este registro se almacena el flag que indica cuando está activo la presencia de un byte de instrucción en el registro de instrucción que es preciso procesar. El componente "comandos_new" detectará si el flag está activo e iniciará en ese momento la ejecución del comando o el procesamiento del dato recibido. Este flag se activa cuando se excite la señal "cambiar_entrada_instruccion" con un valor uno lógico. Se desactivará cuando se excite la señal "reset_byte_instruccion" con un valor uno lógico después de la ejecución del comando.



Flag "FlagEnviarAsentimiento":
Cuando se activa este flag se indica al módulo encargado de enviar datos al ordenador que es preciso enviar un byte de asentimiento FA. Este hecho ocurre cuando se ha recibido un comando y el ratón debe devolver un byte de asentimiento FA para indicar al ordenador que el byte ha sido recibido de forma correcta y está en proceso. Se activa con un valor uno lógico por el módulo "comandos_new" al activar la señal "enviar_asentimiento_entrada". Se desactiva cuando se excita la señal "asentimiento_enviado" con un valor uno lógico tras el envío del byte de asentimiento FA.



INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **RESET_CONTADORES_MOVIMIENTO**: Esta señal borra los valores almacenados en los registros de respuesta y desactiva los flags de respuesta cuando los datos que contienen se deben a un evento de movimiento originado con la frecuencia de muestreo al detectarse un cambio en el estado de movimiento del dispositivo y estar el envío de datos habilitado en el modo corriente.
- **BYTE_ENTRADA_MOVIMIENTO_1**: Primer byte del paquete de datos de movimiento que se guardará en el primer registro de respuesta al activarse la señal "cambiar_entrada_movimiento_1" para ser enviado al ordenador. Un evento en el estado de movimientos del ratón originó este byte que sólo será almacenado si no existe una respuesta anterior todavía pendiente de envío.
- **BYTE_ENTRADA_MOVIMIENTO_2**: Segundo byte del paquete de datos de movimiento que se guardará en el segundo registro de respuesta al activarse la señal "cambiar_entrada_movimiento_2" para ser enviado al ordenador. Un evento en el estado de movimientos del ratón originó este byte que sólo será almacenado si no existe una respuesta anterior todavía pendiente de envío.
- **BYTE_ENTRADA_MOVIMIENTO_3**: Tercer byte del paquete de datos de movimiento que se guardará en el tercer registro de respuesta al activarse la señal "cambiar_entrada_movimiento_3" para ser enviado al ordenador. Un evento en el estado de movimientos del ratón originó este byte que sólo será almacenado si no existe una respuesta anterior todavía pendiente de envío.
- **CAMBIAR_ENTRADA_MOVIMIENTO_1**: Al activar esta señal se guardará el "byte_entrada_movimiento_1" en el primer registro de respuesta si no existe una respuesta anterior todavía pendiente de envío. También se activará el flag "flag_salida_respuesta_1" para indicar al sistema encargado de enviar datos al ordenador que es preciso enviar este byte en el momento en el que sea posible.

- **CAMBIAR_ENTRADA_MOVIMIENTO_2**: Al activar esta señal se guardará el "byte_entrada_movimiento_2" en el segundo registro de respuesta si no existe una respuesta anterior todavía pendiente de envío. También se activará el flag "flag_salida_respuesta_2" para indicar al sistema encargado de enviar datos al ordenador que es preciso enviar este byte en el momento en el que sea posible.
- **CAMBIAR_ENTRADA_MOVIMIENTO_3**: Al activar esta señal se guardará el "byte_entrada_movimiento_3" en el tercer registro de respuesta si no existe una respuesta anterior todavía pendiente de envío. También se activará el flag "flag_salida_respuesta_3" para indicar al sistema encargado de enviar datos al ordenador que es preciso enviar este byte en el momento en el que sea posible.
- **RESET_RESPUESTA**: Esta señal borra cualquier tipo de respuesta presente en los registros de respuesta del módulo y desactiva los flags correspondientes a estos registros. El sistema activa esta señal cuando quiere eliminar la respuesta anterior para guardar una nueva respuesta. Elimina la respuesta antigua para evitar que se envíe junto con la nueva respuesta un byte de la anterior.
- **BYTE_ENTRADA_RESPUESTA_1**: Primer byte o único de un paquete de datos de respuesta a un comando generado por el módulo "comandos_new". Se guardará en el primer registro de respuesta a la espera de ser enviado al ordenador cuando se active la señal "cambiar_entrada_respuesta_1".
- **BYTE_ENTRADA_RESPUESTA_2**: Segundo byte de un paquete de datos de respuesta a un comando generado por el módulo "comandos_new". Se guardará en el segundo registro de respuesta a la espera de ser enviado al ordenador cuando se active la señal "cambiar_entrada_respuesta_2".
- **BYTE_ENTRADA_RESPUESTA_3**: Tercer byte de un paquete de datos de respuesta a un comando generado por el módulo "comandos_new". Se guardará en el tercer registro de respuesta a la espera de ser enviado al ordenador cuando se active la señal "cambiar_entrada_respuesta_3".
- **BYTE_RESPUESTA_1_ENVIADO**: Esta señal desactivará el flag "flag_salida_respuesta_1" cuando el primer byte de respuesta haya sido enviado para evitar que vuelva a ser transmitido.
- **BYTE_RESPUESTA_2_ENVIADO**: Esta señal desactivará el flag "flag_salida_respuesta_2" cuando el segundo byte de respuesta haya sido enviado para evitar que vuelva a ser transmitido.
- **BYTE_RESPUESTA_3_ENVIADO**: Esta señal desactivará el flag "flag_salida_respueta_3" cuando el tercer byte de respuesta haya sido enviado para evitar que vuelva a ser transmitido.
- **CAMBIAR_ENTRADA_RESPUESTA_1**: Al activar esta señal se guardará el byte "byte_entrada_respuesta_1" en el primer byte de respuesta y se activará el flag "flag_salida_respuesta_1" para indicar al sistema-ratón que este byte debe enviarse al ordenador en el momento en el que sea posible.
- **CAMBIAR_ENTRADA_RESPUESTA_2**: Al activar esta señal se guardará el byte "byte_entrada_respuesta_2" en el segundo byte de respuesta y se activará el flag "flag_salida_respuesta_2" para indicar al sistema-ratón que este byte debe enviarse al ordenador en el momento en el que sea posible.
- **CAMBIAR_ENTRADA_RESPUESTA_3**: Al activar esta señal se guardará el byte "byte_entrada_respuesta_3" en el tercer byte de respuesta y se activará el flag

"flag_salida_respuesta_3" para indicar al sistema-ratón que este byte debe enviarse al ordenador en el momento en el que sea posible.

- **BYTE_SALIDA_RESPUESTA_1**: Primer byte de respuesta a un comando o primer byte de un paquete de datos de movimiento que deberá ser enviado al ordenador en cuanto sea posible si el flag "flag_salida_respuesta_1" está activo.
- **BYTE_SALIDA_RESPUESTA_2**: Segundo byte de respuesta a un comando o segundo byte de un paquete de datos de movimiento que deberá ser enviado al ordenador en cuanto sea posible si el flag "flag_salida_respuesta_2" está activo y no lo está el flag "flag_salida_respuesta_1".
- **BYTE_SALIDA_RESPUESTA_3**: Tercer byte de respuesta a un comando o tercer byte de un paquete de datos de movimiento que deberá ser enviado al ordenador en cuanto sea posible si el flag "flag_salida_respuesta_3" está activo y no lo están los flags "flag_salida_respuesta_1" y "flag_salida_respuesta_2".
- **FLAG_SALIDA_RESPUESTA_1**: Este flag indica cuando está activo que el byte almacenado en el primer registro de respuesta y accesible en el byte "byte_salida_respuesta_1" debe ser enviado al ordenador en cuanto sea posible.
- **FLAG_SALIDA_RESPUESTA_2**: Este flag indica cuando está activo que el byte almacenado en el segundo registro de respuesta y accesible en el byte "byte_salida_respuesta_2" debe ser enviado al ordenador en cuanto sea posible cuando no esté activo el flag "flag_salida_respuesta_1".
- **FLAG_SALIDA_RESPUESTA_3**: Este flag indica cuando está activo que el byte almacenado en el tercer registro de respuesta y accesible en el byte "byte_salida_respuesta_3" debe ser enviado al ordenador en cuanto sea posible cuando no estén activos los flags "flag_salida_respuesta_1" y "flag_salida_respuesta_2".
- **BYTE_ENTRADA_INSTRUCCION**: Byte que se almacenará en el registro de instrucción cuando se active la señal "cambiar_entrada_instruccion".
- **CAMBIAR_ENTRADA_INSTRUCCION**: Esta señal se activa cuando se recibe un comando que debe ejecutarse o un dato para procesar desde el ordenador. Al activar la señal se guardará en el registro de instrucción a través de la entrada "byte_entrada_instruccion" el byte recibido para que pueda ser posteriormente ejecutado o procesado en el módulo "comandos_new". También activará el flag "flag_salida_instruccion" que será el que inicie el proceso de ejecución del comando en el módulo "comandos_new".
- **RESET_BYTE_INSTRUCCION**: Esta señal desactivará el flag de instrucción "flag_salida_instruccion" y borrará el byte almacenado en el registro de instrucción para evitar que sea ejecutada.
- **BYTE_SALIDA_INSTRUCCION**: Byte almacenado en el registro de instrucción que será procesado por el módulo "comandos_new" cuando esté activo el flag "flag_salida_instruccion".
- **FLAG_SALIDA_INSTRUCCION**: Este flag inicia el procesamiento de la instrucción o dato recibido en el módulo "comandos_new". Cuando este flag está activo indica al sistema-ratón que el byte almacenado en el registro de instrucción y accesible en el "byte_salida_instruccion" debe ser ejecutado lo antes posible.
- **ENVIAR_ASENTIMIENTO_ENTRADA**: Al activar esta señal guardamos un valor uno lógico en el registro del flag de asentimiento.
- **ASENTIMIENTO_ENVIADO**: Una vez enviado el asentimiento FA desactivamos el flag de asentimiento con un valor uno lógico en esta señal.

- **ENVIAR_ASENTIMIENTO_SALIDA**: El asentimiento es el byte con más prioridad en el envío y será el primero en enviarse cuando el flag de asentimiento "enviar_asentimiento_salida" esté activo. Tras recibir un comando o dato desde el ordenador, debe enviarse un byte de asentimiento FA para indicar al ordenador que la recepción del dato ha sido correcta. En este caso un valor uno lógico en la señal "enviar_asentimiento_entrada" activará el flag de asentimiento "enviar_asentimiento_salida". Una vez concluido el envío del byte de asentimiento, un valor uno lógico en la señal "asentimiento_enviado" desactivará el flag de asentimiento "enviar_asentimiento_salida".

1.2.3.7 sistema_recibir_byte.vhd

```
ENTITY sistema_recibir_byte IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    inicio_lectura: IN STD_LOGIC;
    abortar_recepcion: IN STD_LOGIC;
    error_recepcion: IN STD_LOGIC;
    fin_lectura: OUT STD_LOGIC;
    recibir: OUT STD_LOGIC;
    reset_respuesta: OUT STD_LOGIC;
    byte_respuesta: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_byte_respuesta: OUT STD_LOGIC;
    flag_instruccion: IN STD_LOGIC;
    cambiar_byte_instruccion: OUT STD_LOGIC;
    validacion_byte_recibido: IN STD_LOGIC;
    reset_instruccion: OUT STD_LOGIC
  );
END sistema_recibir_byte;
```

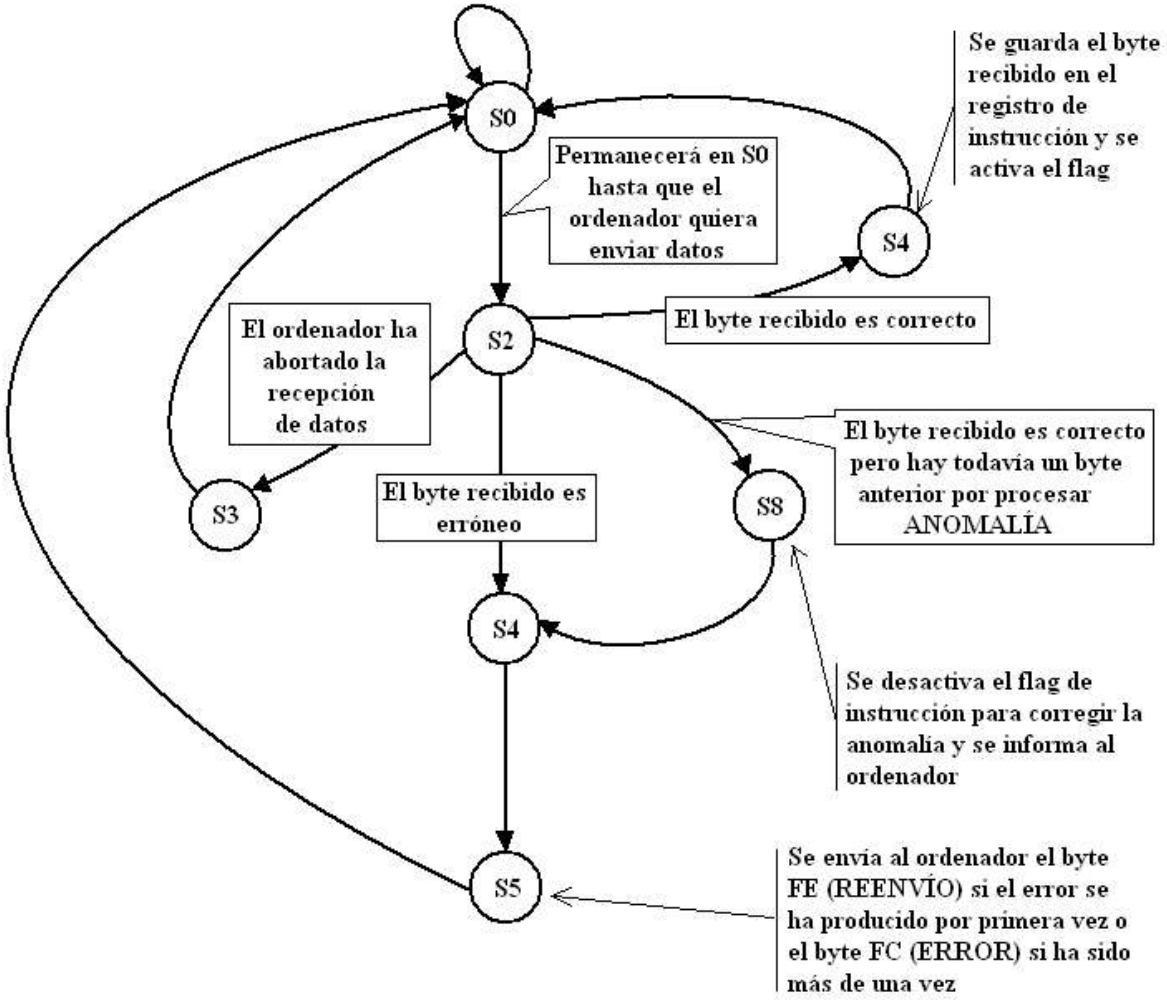
Este módulo activa la recepción de un byte cuando se detecte la intención del ordenador de enviar un byte.

Si el ordenador quiere enviar algún byte (ya sea comando o datos) llevará la línea de datos a nivel bajo manteniendo en alta impedancia la línea de reloj del protocolo serie PS/2. Si el ratón está enviado datos al ordenador se interrumpirá el envío. El módulo "sistema" detectará cuándo la línea de datos está a nivel bajo y activará la señal "inicio_lectura" con un valor uno lógico para iniciar el proceso de recepción de un byte desde el ordenador.

Una máquina de estados controlará este proceso de envío. Los posibles estados del proceso son los siguientes:

- **SO**: S0 es el estado de reposo del sistema. Permanecerá en este estado hasta que el módulo "sistema" active la señal "inicio_lectura" para empezar el proceso de recepción de un byte. En ese momento pasará al estado S2.
- **S2**: En este estado se activa la señal "recibir" para que el módulo "recibir_byte" se active y genere las señales PS/2 en las líneas de datos y de reloj necesarias para la recepción del byte (comando o dato) desde el ordenador. Permanecerá en este estado hasta que haya sido terminada la recepción del byte o hasta que sea interrumpida. La recepción puede ser interrumpida por el ordenador y el módulo "recibir_byte" activará en ese caso la señal "abortar_recepcion". Pasará en este caso al estado S3. Si en la recepción del byte se ha producido un error, el módulo "recibir_byte" activará en ese caso la señal "error_recepcion". Pasará en este caso al estado S4. Si no ha habido error en la recepción, el byte recibido estará disponible cuando sea activada la señal "validacion_byte_recibido", por lo tanto, esta señal indicará el fin de la recepción PS/2 del byte. Una vez concluida la recepción, si el flag "flag_instruccion" está activo con un valor uno lógico implicaría que el ordenador ha enviado un nuevo comando antes de que el ratón haya podido responder al comando anterior. Ha ocurrido una anomalía, pues el ordenador debe esperar durante un tiempo prudencial la respuesta al comando enviado. En este caso pasará al estado S8. Si el flag "flag_instruccion"

está desactivado con un valor cero lógico y el byte recibido es correcto, podrá concluirse de forma satisfactoria el proceso de recepción. En este caso pasará al estado S7.



- **S3**: Se ha abortado la recepción del byte y se activará la señal "fin_lectura" para indicar al módulo "sistema" que la lectura de datos desde el ordenador ha concluido. De esta forma el módulo "sistema" entrará en el estado de reposo a la espera de la detección de un proceso de lectura o escritura. Pasará al estado S0 para esperar una nueva recepción de datos.
- **S4**: Como se ha producido un error en la recepción del byte, debe avisarse al ordenador del hecho ocurrido. En este estado se borran todas las posibles respuestas que puedan estar pendientes de ser enviadas, pues será necesario el envío del byte FE (REENVÍO) o del byte "FC" (ERROR) para informar al ordenador del error. Pasará al estado S5.
- **S5**: En este estado se guarda en el primer byte de respuesta el byte FE (REENVÍO) si el error se ha producido por primera vez o el byte "FC" (ERROR) si el error se ha producido más de una vez. Un registro almacena el flag que indica si el error se ha producido por primera vez o más de una vez, se activa cuando se ha producido un error y se desactiva al recibirse un byte

de forma correcta. También se activará el flag correspondiente a la primera respuesta, lo cual generará el envío del primer byte de respuesta en el momento en el que sea posible. Activará la señal "fin_lectura" para indicar al módulo "sistema" que la lectura de datos desde el ordenador ha concluido. De esta forma el módulo "sistema" entrará en el estado de reposo a la espera de la detección de un proceso de lectura o escritura. Pasará al estado S0 para esperar una nueva recepción de datos.

- **S7**: Como la recepción del byte enviado por el ordenador ha sido un éxito, se guardará el byte recibido en el registro de instrucción de la memoria de datos y se activará el flag correspondiente al byte de instrucción para que el módulo "comandos" inicie la ejecución de la instrucción o procese el dato recibido. Activará la señal "fin_lectura" para indicar al módulo "sistema" que la lectura de datos desde el ordenador ha concluido. De esta forma el módulo "sistema" entrará en el estado de reposo a la espera de la detección de un proceso de lectura o escritura. Pasará al estado S0 para esperar una nueva recepción de datos.
- **S8**: La presencia de un byte todavía por ejecutar o procesar en el registro de instrucción de la memoria de datos cuando se recibe un nuevo byte implica que ha ocurrido un error y el ratón debe solicitar el reenvío del byte o informar del error al ordenador. Se desactiva el flag de instrucción de la memoria de datos pues el comando guardado no debe ser ya ejecutado. Pasará al estado S4 para informar al ordenador de la anomalía.

INTERFAZ DEL MÓDULO:

- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **INICIO_LECTURA**: El módulo "sistema" avisará cuando el ordenador quiere enviar datos activando la señal "inicio_lectura" con un valor uno lógico.
- **ABORTO_RECEPCION:** Si durante el proceso de recepción el ordenador quiere abortar por algún motivo la recepción del byte, el módulo "recibir_byte" activará la señal "aborto_recepcion".
- **ERROR_RECEPCION**: Si se ha recibido un byte pero no es un byte válido (error de paridad, etc.), el módulo "recibir_byte" activará la señal "error_recepcion".
- **FIN_LECTURA**: Esta señal se activará para indicar al módulo "sistema" que la lectura de datos desde el ordenador ha concluido. De esta forma el módulo "sistema" entrará en el estado de reposo a la espera de la detección de un nuevo proceso de lectura o escritura.
- **RECIBIR**: Esta seña se activará para que el módulo "recibir_byte" despierte y genere las señales PS/2 en las líneas de datos y de reloj necesarias para la recepción de un byte (comando o dato) desde el ordenador.
- **RESET_RESPUESTA**: Borra todas las posibles respuestas que puedan estar esperando en los registros de respuesta de la memoria de datos del sistema-ratón para ser enviadas. Es necesario activar esta señal si quiero enviar al ordenador una nueva respuesta.
- **BYTE_RESPUESTA**: Byte que se guardará en primer registro de respuesta de la memoria de datos del sistema-ratón cuando se active la señal "cambiar_byte_respuesta".
- **CAMBIAR_BYTE_RESPUESTA**: Si se ha producido un error en la recepción del byte, debe informarse al ordenador del hecho ocurrido. Al activar esta señal se guarda en el primer

registro de respuesta de la memoria de datos del sistema-ratón el byte "byte_respuesta". Se activará también el flag del primer registro de respuesta de la memoria de datos, lo que provocará el envío del byte FE (REENVÍO) si el error se ha producido por primera vez o del byte FC (ERROR) si el error se ha producido más de una vez.

- **FLAG_INSTRUCCION**: Este flag indica cuando está activo que existe un byte guardado en el registro de instrucción de la memoria de datos pendiente de ser ejecutado (comando) o procesado (datos). Ocurriría una anomalía si estando este flag activo se recibe un nuevo byte. En este caso el ordenador será informado del error ocurrido.
- **CAMBIAR_BYTE_INSTRUCCION**: Tras la recepción correcta de un byte se activará esta señal para que el byte recibido sea guardado en el registro de instrucción de la memoria de datos y provocará la activación del flag correspondiente al registro de instrucción, lo cual desencadenará la ejecución en el módulo "comandos_new" del byte guardado en el registro.
- **VALIDACION_BYTE_RECIBIDO**: Mediante esta señal el módulo "recibir_byte" avisará de que ha concluido la recepción PS/2 del byte enviado por el ordenador y que el byte recibido es correcto y no ha habido errores.
- **RESET_INSTRUCCION:** Desactiva el flag de instrucción de la memoria de datos cuando se ha recibido un comando estando el flag todavía activado. Así se evita que vuelva a producirse esta anomalía.

1.2.3.8 sistema_enviar_byte.vhd

```
ENTITY sistema_enviar_byte IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    byte_salida_respuesta_1: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_salida_respuesta_2: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_salida_respuesta_3: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_salida_respuesta_1: IN STD_LOGIC;
    flag_salida_respuesta_2: IN STD_LOGIC;
    flag_salida_respuesta_3: IN STD_LOGIC;
    byte_respuesta_1_enviado: OUT STD_LOGIC;
    byte_respuesta_2_enviado: OUT STD_LOGIC;
    byte_respuesta_3_enviado: OUT STD_LOGIC;
    enviar_asentimiento_salida: IN STD_LOGIC;
    asentimiento_enviado: OUT STD_LOGIC;
    fin_envio: IN STD_LOGIC;
    abortar: IN STD_LOGIC;
    enviar: OUT STD_LOGIC;
    byte_enviar: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    inicio_envio_sistema: IN STD_LOGIC;
    fin_envio_sistema: OUT STD_LOGIC;
    enviar_datos_sistema: OUT STD_LOGIC;
    reset_reenvio: OUT STD_LOGIC;
    byte_entrada_reenvio_1: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_reenvio_2: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_entrada_reenvio_3: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_entrada_reenvio_1: OUT STD_LOGIC;
    cambiar_entrada_reenvio_2: OUT STD_LOGIC;
    cambiar_entrada_reenvio_3: OUT STD_LOGIC
  );
END sistema_enviar_byte;
```

Este módulo inicia el envío del byte de asentimiento o de los bytes de respuesta al ordenador en el caso de que estén pendientes de envío.

Si está pendiente por enviar alguno de los bytes anteriores, el flag correspondiente al byte a enviar estará activo con un valor uno lógico. El módulo detectará si está activo alguno de los flags y en caso afirmativo procederá al envío del byte correspondiente. Existe un orden de prioridad en el envío de los bytes:

1. En primer lugar se enviará el byte de asentimiento al comando recibido en el caso de que esté activa la señal "enviar_asentimiento_salida".
2. En segundo lugar se enviará el primer byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_1".
3. En tercer lugar se enviará el segundo byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_2".
4. En cuarto lugar se enviará el tercer byte de respuesta almacenado en la memoria de datos del sistema ratón en el caso de que esté activo el flag "flag_salida_respuesta_3".

A medida que se van enviado los bytes, se irán desactivando los flags correspondientes, y así hasta que no quede ningún byte por enviar. Una vez enviado un byte se guardará en los registros de reenvío para que puedan ser reenviados al ordenador en el caso de que lo solicite con el comando FE (REENVÍO).

INTERFAZ DEL MÓDULO:

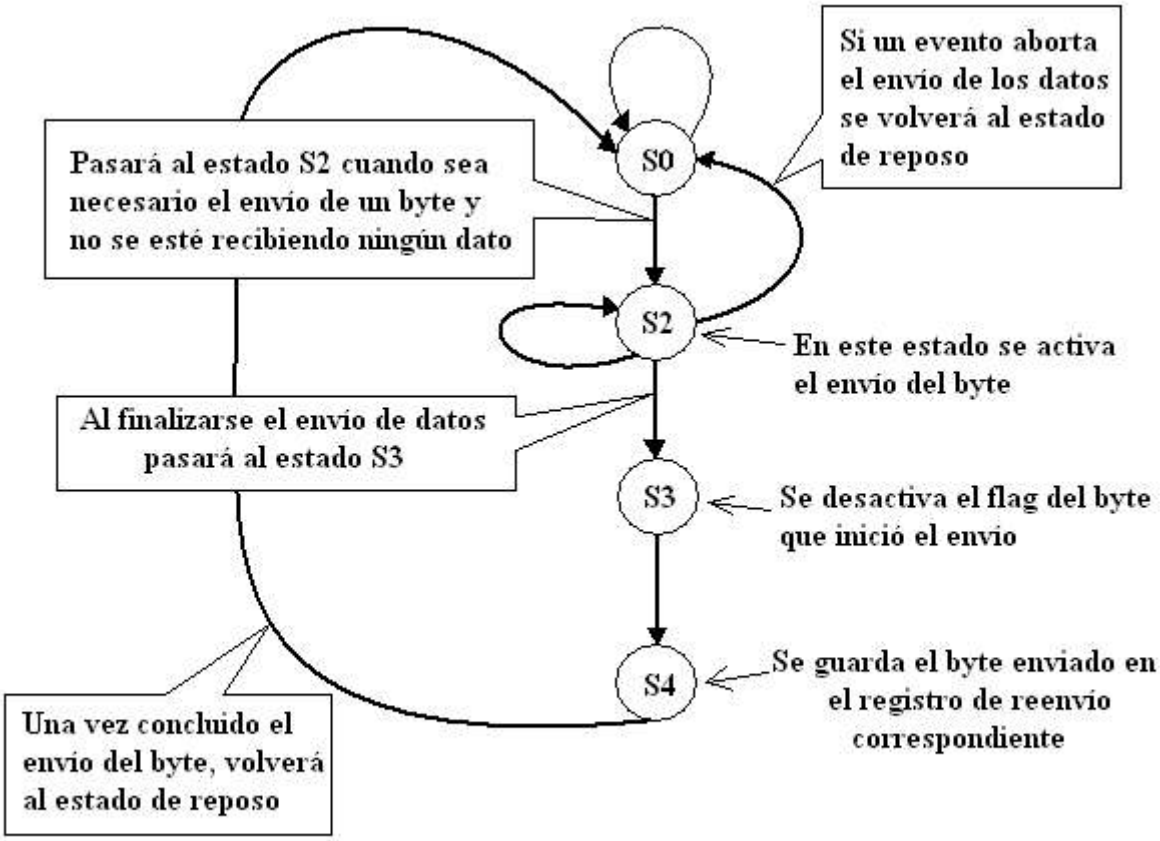
- **RESETZ**: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- **RELOJ**: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- **BYTE_SALIDA_RESPUESTA_1**: Primer byte de respuesta a un comando que se enviará al ordenador en el momento en el que sea posible si está activo el flag "flag_salida_respuesta_1".
- **BYTE_SALIDA_RESPUESTA_2**: Segundo byte de respuesta a un comando que se enviará al ordenador en el momento en el que sea posible si está activo el flag "flag_salida_respuesta_2".
- **BYTE_SALIDA_RESPUESTA_3**: Tercer byte de respuesta a un comando que se enviará al ordenador en el momento en el que sea posible si está activo el flag "flag_salida_respuesta_3".
- **FLAG_SALIDA_RESPUESTA_1**: Este flag indicará cuando está activo que es preciso el envío del primer byte de respuesta de la memoria de datos del ratón. Se desactivará cuando el byte haya sido enviado o ya no sea preciso el envío de la respuesta.
- **FLAG_SALIDA_RESPUESTA_2**: Este flag indicará cuando está activo que es preciso el envío del segundo byte de respuesta de la memoria de datos del ratón. Se desactivará cuando el byte haya sido enviado o ya no sea preciso el envío de la respuesta.
- **FLAG_SALIDA_RESPUESTA_3**: Este flag indicará cuando está activo que es preciso el envío del tercer byte de respuesta de la memoria de datos del ratón. Se desactivará cuando el byte haya sido enviado o ya no sea preciso el envío de la respuesta.
- **BYTE_RESPUESTA_1_ENVIADO**: Esta señal se activará cuando el primer byte de respuesta a un comando anterior de la memoria de datos haya sido enviado. Provocará la desactivación del flag "flag_salida_respuesta_1" para evitar que el byte vuelva a ser reenviado.
- **BYTE_RESPUESTA_2_ENVIADO**: Esta señal se activará cuando el primer byte de respuesta a un comando anterior de la memoria de datos haya sido enviado. Provocará la desactivación del flag "flag_salida_respuesta_1" para evitar que el byte vuelva a ser reenviado.
- **BYTE_RESPUESTA_3_ENVIADO**: Esta señal se activará cuando el primer byte de respuesta a un comando anterior de la memoria de datos haya sido enviado. Provocará la desactivación del flag "flag_salida_respuesta_1" para evitar que el byte vuelva a ser reenviado.
- **ENVIAR_ASENTIMIENTO_SALIDA**: Este flag solicita el envío del byte de asentimiento FA hacia el ordenador. El envío de este byte es el que tiene más prioridad y será enviado hacia el ordenador en cuanto deje de recibir datos desde él. Este módulo se encargará de iniciar el proceso de envío del byte FA cuando la señal esté activa.
- **ASENTIMIENTO_ENVIADO**: Cuando haya sido enviado el byte de asentimiento se activará esta señal para desactivar el flag "enviar_asentimiento_salida" y así evitar volver a enviar el byte de asentimiento.
- **FIN_ENVIO**: Esta señal indicará cuando esté activa con un valor uno lógico que ya ha concluido de forma correcta el envío del byte "byte_enviar" iniciada con la señal "enviar". El módulo "sistema_enviar_byte" decidió qué byte era el que había que enviar.
- **ABORTAR**: Esta señal se activa cuando el ordenador ha interrumpido el proceso de envío del byte "byte_enviar" iniciado con la señal "enviar". Una vez parado el envío en el módulo "enviar", detendrá el envío en el módulo"sistema_enviar_byte".

- **ENVIAR**: Una vez decidido cuál es el byte a enviar en el caso de que éste existiese, activará esta señal para que el módulo "enviar" proceda al envío del byte "byte_enviar". No se procederá a iniciar el envío mientras se estén recibiendo datos. El sistema esperará hasta que se haya concluido el envío o haya sido abortado por el ordenador.
- **BYTE_ENVIAR**: Una vez decidido el byte a enviar en el caso de que éste existiese, informará al módulo "enviar" que inicie el envío del byte especificado por este vector de bits cuando se active la señal "enviar".
- **INICIO_ENVIO_SISTEMA**: El módulo "sistema_enviar_byte" informará al módulo "sistema" que es preciso enviar cierta información al ordenador a través de la señal "enviar_datos_sistema". No será posible el envío mientras se estén recibiendo datos desde el ordenador. Cuando sea posible el envío, el módulo "sistema" lo indicará activando la señal "inicio_envio_sistema".
- **FIN_ENVIO_SISTEMA**: El módulo "sistema_enviar_byte" controla el envío del módulo "enviar". Una vez concluido avisará al módulo "sistema" activando esta señal para que se quede en espera de un proceso de envío o de recepción de datos. Una vez concluido el envío el módulo "sistema" quedará en el estado de reposo.
- **ENVIAR_DATOS_SISTEMA**: Esta señal avisa al módulo "sistema" que es preciso enviar cierta información al ordenador. Implicaría que está activo el flag "enviar_asentimiento_salida" o los flags de respuesta "flag_salida_respuesta_1", "flag_salida_respuesta_2" o "flag_salida_respuesta_3". El módulo "sistema" activará la señal "inicio_envío_sistema" cuando sea factible el envío de datos y saldrá del estado de reposo.
- **RESET_REENVIO**: Desactiva todos los flags de respuesta ("flag_salida_respuesta_1", "flag_salida_respuesta_2" y "flag_salida_respuesta_3") para interrumpir el envío de un paquete de datos (ya sea de un byte o de tres bytes). Se realizará esta operación cuando un nuevo comando haya obligado a un cambio en la respuesta. Podrá así cargarse la nueva respuesta sin miedo a la transmisión de algún byte antiguo.
- **BYTE_ENTRADA_REENVIO_1**: Byte que se guardará en el primer registro de reenvío del módulo "reenvío" cuando se active la señal "cambiar_entrada_reenvio_1".
- **BYTE_ENTRADA_REENVIO_2**: Byte que se guardará en el segundo registro de reenvío del módulo "reenvío" cuando se active la señal "cambiar_entrada_reenvio_2".
- **BYTE_ENTRADA_REENVIO_3**: Byte que se guardará en el tercer registro de reenvío del módulo "reenvío" cuando se active la señal "cambiar_entrada_reenvio_3".
- **CAMBIAR_ENTRADA_REENVIO_1**: Al activar esta señal se guardará en el primer registro de reenvío el byte aportado con el vector de bits "byte_entrada_reenvio_1". Esta señal se activará cuando se haya concluido de manera satisfactoria el envío del primer byte de un paquete de datos de respuesta (ya sea de un byte o de más bytes). Se activará también el flag correspondiente al primer byte de reenvío para que en el caso de que se solicite el reenvío con el comando FE (REENVÍO) implique la transmisión del primer byte de reenvío.
- **CAMBIAR_ENTRADA_REENVIO_2**: Al activar esta señal se guardará en el segundo registro de reenvío el byte aportado con el vector de bits "byte_entrada_reenvio_2". Esta señal se activará cuando se haya concluido de manera satisfactoria el envío del segundo byte de un paquete de datos de respuesta. Se activará también el flag correspondiente al segundo byte de reenvío para que en el caso de que se solicite el reenvío con el comando FE (REENVÍO) implique la transmisión del segundo byte de reenvío.
- **CAMBIAR_ENTRADA_REENVIO_3**: Al activar esta señal se guardará en el tercer registro de reenvío el byte aportado con el vector de bits "byte_entrada_reenvio_3". Esta señal se

activará cuando se haya concluido de manera satisfactoria el envío del tercer byte de un paquete de datos de respuesta. Se activará también el flag correspondiente al tercer byte de reenvío para que en el caso de que se solicite el reenvío con el comando FE (REENVÍO) implique la transmisión del tercer byte de reenvío.

Una máquina de estados controla el funcionamiento del dispositivo. Puede estar en uno de los siguientes estados:

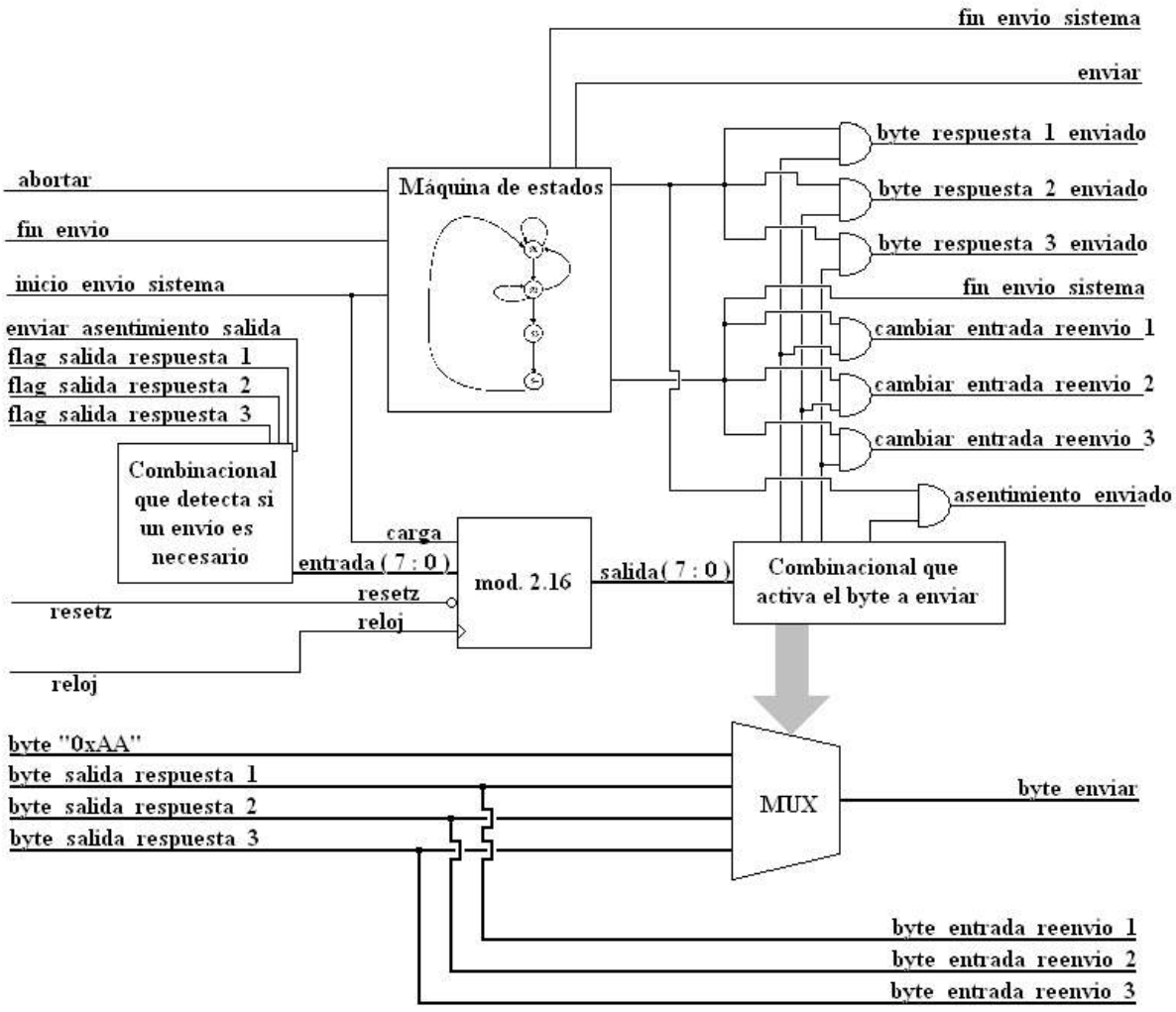
- **S0**: Este estado es el de reposo. Permanecerá en él hasta que se haya detectado un byte por enviar, estando el sistema en disposición de enviar bytes (no está en espera de recibir un comando). En ese momento pasará al estado S2.



- **S2**: Activa la señal "enviar" con un valor uno lógico para que el módulo "enviar_byte" envíe al ordenador el byte "byte_enviar" de acuerdo con el protocolo PS/2. Esperará en este estado hasta que se termine el envío del byte (en ese caso pasará al estado S3) o hasta que sea abortado por el ordenador (en ese caso pasará al estado de reposo S0).
- **S3**: Activa la señal "byte_enviado" para desactivar el flag correspondiente. Ejemplo: si se ha enviado el byte de asentimiento, se desactivará el flag "enviar_asentimiento_salida". Si el byte

que se ha enviado es el segundo byte de respuesta, se desactivará el flag "flag_salida_respuesta_2". Pasará al estado S4.

- **S4**: Se activará la señal "cambiar_entrada_reenvio" para guardar el byte enviado en el registro del módulo "reenvio" correspondiente según el orden en el paquete de datos enviado. Ejemplo: si el byte se corresponde con el segundo byte de un paquete de datos, se almacenará en el segundo registro de reenvío. Luego pasará al estado de reposo S0.



1.2.3.9 reenvio.vhd

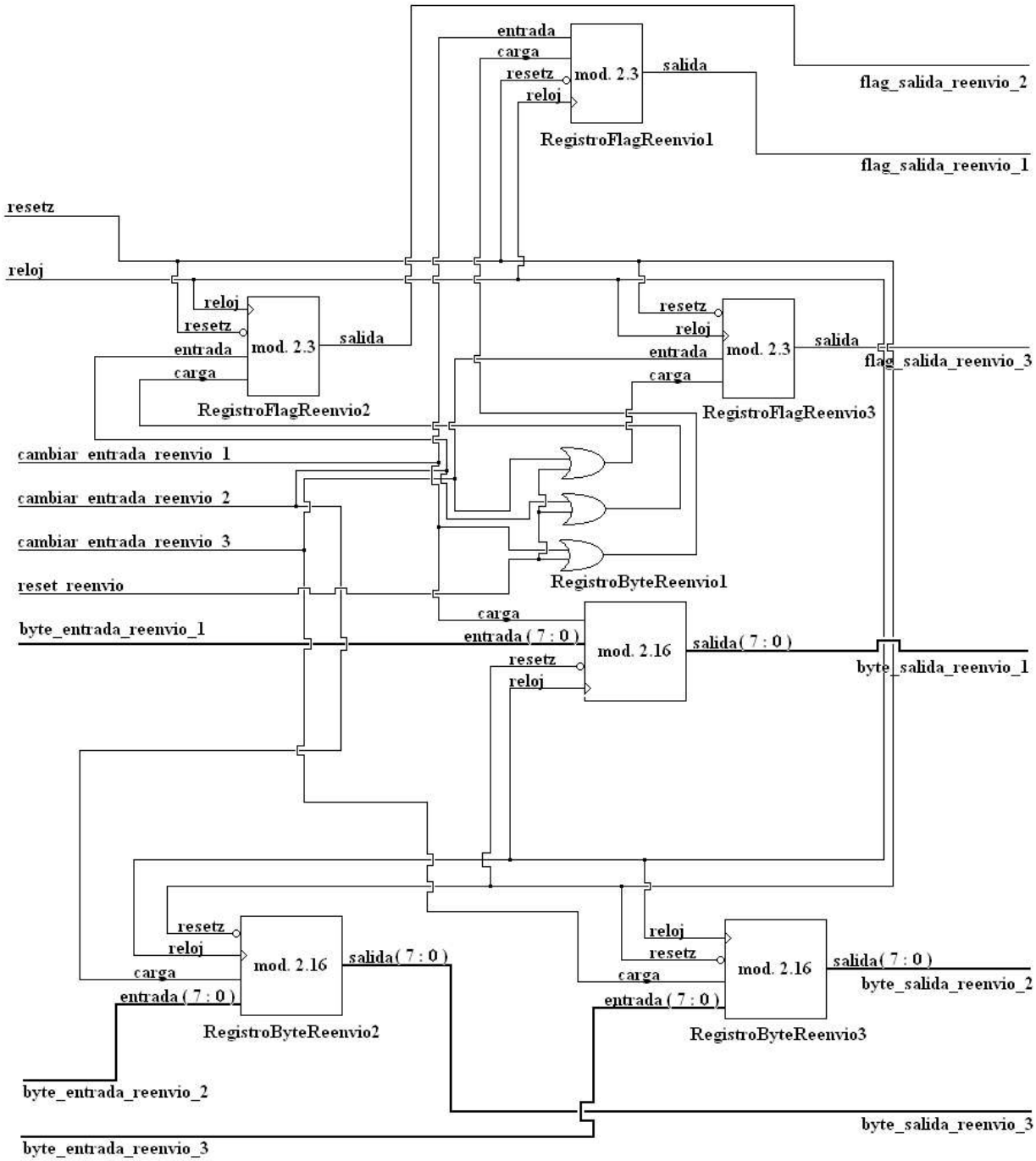
```
ENTITY reenvio IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    reset_reenvio: IN STD_LOGIC;
    byte_entrada_reenvio_1: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_entrada_reenvio_2: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_entrada_reenvio_3: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    cambiar_entrada_reenvio_1: IN STD_LOGIC;
    cambiar_entrada_reenvio_2: IN STD_LOGIC;
    cambiar_entrada_reenvio_3: IN STD_LOGIC;
    byte_salida_reenvio_1: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_salida_reenvio_2: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_salida_reenvio_3: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    flag_salida_reenvio_1: OUT STD_LOGIC;
    flag_salida_reenvio_2: OUT STD_LOGIC;
    flag_salida_reenvio_3: OUT STD_LOGIC
  );
END reenvio;
```

En este módulo se almacenarán los bytes enviados al ordenador para que puedan ser reenviados en el caso de que el ordenador solicite el reenvío.

Si se envió un paquete de datos formado por más de un byte, se volverá a reenviar todo el paquete de datos. El reenvío es la respuesta del sistema ratón al comando FE (REENVÍO) y se activará dentro del módulo "comandos_new". El módulo "reenvio" sólo almacenará los bytes que se enviarán en el caso de que el comando FE lo solicite. Se necesitan entonces tres registros de 8 bits cada uno para guardar los posibles bytes de respuesta que puedan ser enviados.

INTERFAZ DEL MÓDULO:

- RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- RESET_REENVIO: Esta señal se activará cuando se haya enviado un paquete de un sólo byte. Posteriormente se cambiará el byte de reenvío primero, pero de esta forma se consigue que se borren los otros 2 bytes.
- BYTE_ENTRADA_REENVIO_1: Primer byte de respuesta que acaba de ser enviado. Se guardará en un registro cuando la señal "cambiar_entrada_reenvio_1" esté activa. En el caso de que se solicite un reenvío será el byte que se enviará como respuesta si el paquete de datos es de un sólo byte o será el primer byte en enviarse si el paquete de datos es de tres bytes.
- BYTE_ENTRADA_REENVIO_2: Segundo byte de respuesta que acaba de ser enviado. Se guardará en un registro cuando la señal "cambiar_entrada_reenvio_2" esté activa. En el caso de que se solicite un reenvío será el segundo byte en enviarse si el paquete de datos es de tres bytes.



- BYTE_ENTRADA_REENVIO_3: Tercer byte de respuesta que acaba de ser enviado. Se guardará en un registro cuando la señal "cambiar_entrada_reenvio_3" esté activa. En el caso

de que se solicite un reenvío será el tercer byte en enviarse si el paquete de datos es de tres bytes.

- CAMBIAR_ENTRADA_REENVIO_1: Al activar esta señal con un valor uno lógico se guardará en el primer registro el byte "byte_entrada_reenvio_1".
- CAMBIAR_ENTRADA_REENVIO_2: Al activar esta señal con un valor uno lógico se guardará en el segundo registro el byte "byte_entrada_reenvio_2".
- CAMBIAR_ENTRADA_REENVIO_3: Al activar esta señal con un valor uno lógico se guardará en el tercer registro el byte "byte_entrada_reenvio_3".
- BYTE_SALIDA_REENVIO_1: Byte almacenado en el primer registro del módulo. Será el primer byte en enviarse cuando se solicite el reenvío de la última respuesta con el comando FE (REENVÍO). Sólo se reenviará si está activo el flag "flag_salida_reenvio_1".
- BYTE_SALIDA_REENVIO_2: Byte almacenado en el segundo registro del módulo. Será el segundo byte en enviarse cuando se solicite el reenvío de la última respuesta con el comando FE (REENVÍO). Sólo se reenviará si está activo el flag "flag_salida_reenvio_2".
- BYTE_SALIDA_REENVIO_3: Byte almacenado en el tercer registro del módulo. Será el tercer byte en enviarse cuando se solicite el reenvío de la última respuesta con el comando FE (REENVÍO). Sólo se reenviará si está activo el flag "flag_salida_reenvio_3".
- FLAG_SALIDA_REENVIO_1: Cuando este flag está activo con un valor uno lógico nos informa que la última respuesta estaba formada por al menos un byte.
- FLAG_SALIDA_REENVIO_2: Cuando este flag está activo con un valor uno lógico nos informa que la última respuesta estaba formada por al menos dos bytes.
- BYTE_SALIDA_REENVIO_3: Cuando este flag está activo con un valor uno lógico nos informa que la última respuesta estaba formada por tres bytes.

1.2.3.10 comandos_new.vhd

```
ENTITY comandos_new IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    frecuencia_muestreo_salida: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    habilitacion_envio_datos_salida: OUT STD_LOGIC;
    modo_salida: OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
    resetear: OUT STD_LOGIC;
    enviar_asentimiento: OUT STD_LOGIC;
    contador_x_sin_escala: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
    contador_y_sin_escala: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
    desbordamiento_x: IN STD_LOGIC;
    desbordamiento_y: IN STD_LOGIC;
    clack: IN STD_LOGIC;
    click: IN STD_LOGIC;
    reset_contadores_movimientos: OUT STD_LOGIC;
    byte_instruccion: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_instruccion: IN STD_LOGIC;
    reset_byte_instruccion: OUT STD_LOGIC;
    byte_reenvio_1: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_reenvio_2: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    byte_reenvio_3: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    flag_reenvio_1: IN STD_LOGIC;
    flag_reenvio_2: IN STD_LOGIC;
    flag_reenvio_3: IN STD_LOGIC;
    reset_respuesta: OUT STD_LOGIC;
    cambiar_respuesta_1: OUT STD_LOGIC;
    byte_respuesta_1: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_respuesta_2: OUT STD_LOGIC;
    byte_respuesta_2: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    cambiar_respuesta_3: OUT STD_LOGIC;
    byte_respuesta_3: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    activa_lectura: OUT STD_LOGIC
  );
END comandos_new;
```

Este módulo es el encargado de ejecutar los comandos recibidos del ordenador.

El funcionamiento del ratón se basa en la recepción y en el envío de una serie de bytes entre el ordenador y el dispositivo a través del puerto serie PS/2. Los bytes recibidos desde el ordenador deben ser identificados ya que se corresponden con comandos que el ordenador quiere que se ejecuten en el dispositivo. Es necesario un módulo encargado de la ejecución de los distintos comandos que active las señales correspondientes a la solicitud del ordenador. El módulo "comandos_new" es el encargado de esta actividad. Para ello utilizará una máquina de estados, una serie de registros donde se almacenará la información del sistema que es modificada por los comandos (la modificación de esta información es la forma en la que el ordenador controla al dispositivo) y lógica combinacional necesaria para la activación de las señales según el estado en el que se encuentre la máquina de estados.

Los siguientes son los únicos comandos que pueden ser enviados al ratón. Si el ratón está en modo Corriente, el host debería deshabilitar el envío de datos (comando Øxf5) antes de enviar alguno de los otros comandos:

- **ØxFF (RESET):** El ratón responde a este comando con un asentimiento (ØxFA) y entra en el modo reset.

- **ØxFE (Reenvío):** El host envía este comando si recibe un dato inválido desde el ratón. El ratón responde con el reenvío del último paquete enviado al host. Si el ratón responde al comando Reenvío con otro paquete de datos inválido, el host puede:
  5) Enviar otro comando de Reenvío.
  6) Enviar un comando de Error.
  7) Reiniciar el sistema mediante la fuente de alimentación.
  8) Inhibir la comunicación (al llevar la línea de Reloj a nivel bajo).

  La acción elegida dependerá del host.

- **ØxF6 (Fijar los valores por defecto):** El ratón responde con un Asentimiento (ØxFA) y carga los siguientes valores.
  1) Frecuencia de muestreo = 100
  2) Resolución = 4 cuentas/mm
  3) Escala = 1:1
  4) Envío de datos deshabilitado

  El ratón entonces resetea los contadores de movimiento y entra en modo corriente.

- **ØxF5 (Deshabilitación del envío de datos):** El ratón responde con un Asentimiento (ØxFA), deshabilita el envío de datos y resetea su contador de movimientos. Esto solamente afecta al envío de datos en el Modo Corriente y no deshabilita al muestreo. Deshabilitado el envío de datos funciona de la misma manera que el "modo remoto" .

- **ØxF4 (Habilitar el envío de datos):** El ratón responde con un Asentimiento (ØxFA), entonces habilita el envío de datos y resetea sus contadores de movimiento. Este comando puede ser enviado mientras el ratón está en "modo remoto" (o en modo Corriente), pero solamente tendrá efecto en el envío de datos en el modo Corriente.

- **ØxF3 (Fijar la frecuencia de muestreo):** El ratón responde con un Asentimiento (ØxFA) y luego lee uno o más bytes del host. El ratón guarda este byte como la nueva frecuencia de muestreo. Después de recibir la frecuencia de muestreo, el ratón responde de nuevo con un asentimiento y resetea sus contadores de movimiento. Las frecuencias de muestreo válidas son 10, 20, 40, 60, 80, 100 y 200 muestras por segundo.

- **ØxF2 (Obtener el identificador de dispositivo ID):** El ratón responde con un asentimiento (ØxFA) seguido por su identificador de dispositivo ID (ØxØØ para los ratones PS/2 estándar). El ratón reseteará los contadores de movimiento.

- **ØxF0 (Activar el "modo remoto"):** El ratón responde con un asentimiento (ØxFA), resetea los contadores de movimiento y entra en el "modo remoto".

- **ØxEE (Activar el modo página):** El ratón responde con un asentimiento (ØxFA), luego resetea los contadores de movimiento y entra en el modo página.

- **ØxEC (Reseteo del modo página):** El ratón responde con un asentimiento (ØxFA), luego resetea sus contadores de movimiento y entra en el modo que estaba antes del modo página (modo corriente o "modo remoto").

- **ØxEB (Lectura de Datos):** El ratón responde con un asentimiento (ØxFA), luego envía un paquete de datos de movimiento. Esta es la única manera de leer datos en el "modo remoto". Después de que los paquetes de datos hayan sido enviados con éxito, resetea los contadores de movimiento.

- **ØxEA (Activar el modo corriente):** El ratón responde con un asentimiento (ØxFA). Luego resetea los contadores de movimiento y entra en modo corriente.

- **ØxE9 (Petición de estado):** El ratón responde con un asentimiento, entonces envía el siguiente paquete de estado de 3 bytes (luego resetea los contadores de movimiento).

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Siempre a 0	Modo	Habilitación	Escala	Siempre a 0	Botón izquierdo	Botón del medio	Botón derecho
Byte 2	RESOLUCIÓN							
Byte 3	FRECUENCIA DE MUESTREO							

Botón derecho, del medio e izquierdo: 1 si el botón es presionado
Ø si el botón no es presionado

Escala: 1 si la escala es 2:1
Ø si la escala es 1:1
(Ver los comandos ØxE7 y ØxE6)

Habilitación: 1 si el envío de datos está habilitado.
Ø si el envío de datos no está habilitado.
(Ver los comandos ØxF5 y ØxF4)

Modo: 1 si el "modo remoto" está habilitado.
Ø si el modo corriente está habilitado.
(Ver los comandos ØxFØ y ØxEA)

- **ØxE8 (Fijar la resolución):** El ratón responde con un asentimiento (ØxFA). Luego lee un byte desde el host y de nuevo responde con un asentimiento (ØxFA). Entonces resetea los contadores de movimiento. El byte leído del host determina la resolución.

Byte leído del host	Resolución
ØxØØ	1 cuenta/mm
ØxØ1	2 cuentas/mm
ØxØ2	3 cuentas/mm
ØxØ3	4 cuentas/mm

- **ØxE7 (Fijar la escala 2:1):** El ratón responde con un asentimiento(ØxFA), luego habilita la escala 2:1.

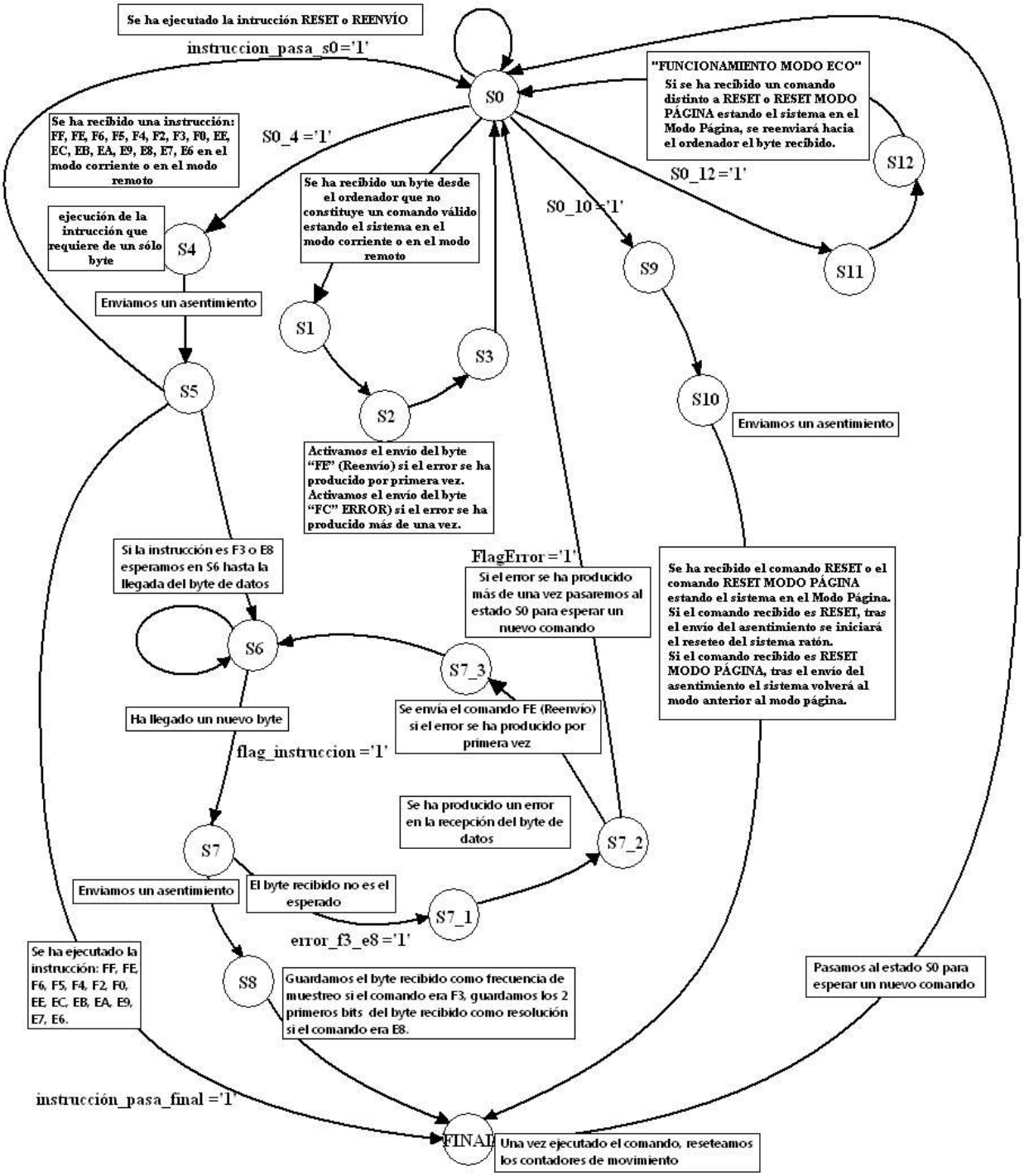
- **ØxE6 (Fijar la escala 1:1):** El ratón responde con un asentimiento (ØxFA) luego habilita la escala 1:1.

Los únicos comandos que el ratón PS/2 estándar envía al host son los comandos "Reenvío" (ØxFE) y "Error" (ØxFC). Ellos funcionan de la misma manera que los comandos enviados del host al dispositivo.

MÁQUINA DE ESTADOS:

Un proceso controla la ejecución de los comandos recibidos desde el ordenador. Aquí se activan las señales que provocarán los cambios en el sistema ratón requeridos por los comandos. Puede encontrarse en uno de los siguientes estados:

- **S0:** Este es el estado de reposo de la máquina de estado, permanecerá en este estado hasta que se reciba un comando desde el ordenador. Pasará al estado S4 al recibir un comando válido (FF, FE , F6, F5, F4, F2, F0, EE, EC, EB, EA, E9, E7, E6, F3, E8) cuando el modo de funcionamiento sea el modo corriente o en el modo remoto. Pasará al estado S1 al recibir un byte que no constituye un comando válido cuando el modo de funcionamiento sea el modo corriente o el modo remoto. Pasará al estado S9 al recibir el comando FF (RESET) o el comando EC (Reset modo página) estando el sistema en el modo página. Pasará al estado S11 si se recibe un byte distinto al comando FF (RESET) o al comando EC (Reset modo página) estando el sistema en el modo página. Si no se ha cumplido ninguna de las condiciones anteriores permanecerá en este estado a la espera de la recepción de un byte.
- **S1:** Se ha recibido un byte desde el ordenador que no constituye un comando válido estando el sistema en el modo corriente o en el modo remoto. Se ha producido pues un error en el envío o en la recepción y se debe iniciar un proceso para comunicárselo al ordenador. Por eso en este estado borramos toda respuesta que hubiera pendiente por enviar al activar la señal "reset_respuesta" y borramos el byte recibido que ha ocasionado el error en la recepción activando el comando "reset_byte_instrucción". Después pasa al estado S2.



- **S2:** Cargamos en el registro 1 de respuesta de la memoria de datos el byte "FE" (Reenvío) si el error se ha producido por primera vez desde la última recepción correcta, de esta forma solicitamos el reenvío del byte al ordenador. Si el error se ha producido más de una vez desde la última recepción correcta cargaremos en el registro 1 de respuesta de la memoria de datos el byte "FC" (ERROR) para indicar al ordenador que se ha producido un error. El sistema

detectará que se ha guardado un byte en este registro ya que también se activará un flag cuando se guarde un byte en el registro 1 de respuesta. La activación del flag desencadenará el envío del byte de respuesta. Después pasa al estado S3.

- **S3:** Se activa con un valor '1' lógico el flag "FlagError" guardado en el registro "RegistroFlagError". Un valor '0' lógico en "FlagError" indica que antes no se había producido un error en la recepción del byte. Un valor '1' lógico indica que ya se ha producido un error en la recepción del byte. El flag se desactiva con un valor '0' lógico al recibirse un byte correcto. De esta forma se sabe si el error se ha producido por primera vez (cuando había guardado un '0' lógico) o más veces (cuando había guardado un '1' lógico). Después pasa al estado S0 para esperar la recepción del nuevo comando.
- **S4:** Se ha recibido un comando válido estando el sistema en el modo corriente o en el modo remoto. Se inicia el proceso de ejecución de la instrucción recibida. Se borran todas las posibles respuestas a algún comando anterior al activar la señal "reset_respuesta". Se guarda en el registro "RegistroByteInstruccion" el byte recibido, es decir, el comando se guarda en el registro anterior para tener constancia de la instrucción recibida si el comando precisa de la recepción de un segundo byte, como son los casos de los comandos F3 (FIJAR FRECUENCIA MUESTREO) y E8 (FIJAR RESOLUCIÓN). Pasa al estado S5.
- **S5:** En este estado se activa la señal "instruccion_guardada" con un valor '1' lógico. La parte combinacional que detecta el tipo de comando recibido toma el valor del byte de instrucción guardado en la memoria de datos externa cuando la señal "instruccion_guardada" es '0' lógico y toma el valor del byte guardado en el registro "RegistroByteInstruccion" interno cuando la señal es '1' lógico. En este estado se borra el registro de la memoria de datos que guarda la instrucción recibida. También iniciamos el envío del byte de asentimiento ("FA") hacia el ordenador al activar la señal "enviar_asentimiento". En este estado se realiza la ejecución de las instrucciones que requieren de un sólo byte al activar la señal "ejecuta_instruccion_1", que junto con la parte combinacional encargada de reconocer el tipo de instrucción generarán las señales necesarias para forzar los cambios que solicita el comando: FF (RESET), FE (REENVÍO), F6 (FIJAR VALORES POR DEFECTO), F5 (DESHABILITACIÓN ENVÍO DATOS), F4 (HABILITAR ENVÍO DATOS), F2 (OBTENER IDENTIFICADOR ID), F0 (ACTIVAR MODO REMOTO), EE (ACTIVAR MODO PÁGINA), EC (RESETEO MODO PAGINA), EB (LECTURA DE DATOS), EA (ACTIVAR MODO CORRIENTE), E9 (PETICIÓN DE ESTADO), E7 (FIJAR ESCALA 2:1), E6 (FIJAR ESCALA 1:1). Si el comando recibido es RESET o REENVÍO, pasará al estado S0. Si el comando es uno de los que no requieren de un segundo byte para su ejecución pasará al estado "FINAL": FF, FE, F6, F5, F4, F2, F0, EE, EC, EB, EA, E9, E7, E6. Si el comando es F3 (FIJAR LA FRECUENCIA DE MUESTREO) o E8 (FIJAR LA RESOLUCIÓN) se requerirá de un segundo byte para la ejecución completa del comando (el segundo byte lleva el dato a guardar), en este caso la máquina de estados pasará al estado S6.
- **S6:** Como el ratón debe esperar hasta que sea recibido el byte de dato después de recibir el comando F3 o E8, permanecerá en este estado hasta que se reciba un nuevo byte desde el ordenador. Pasará entonces al estado S7.
- **S7:** Activamos la señal que inicia el envío del asentimiento tras la recepción del byte de datos. La señal "instruccion_guardada" está a '1' lógico para que la lógica que detecta el tipo de instrucción recibida tome el valor desde el registro "RegistroByteInstruccion" y no desde la memoria de datos (último byte recibido). Si el byte recibido no se corresponde con uno de los datos esperados

(0A, 14, 28, 3C, 50, 64, C8 si la instrucción es FIJAR LA FRECUENCIA DE MUESTREO; 00, 01, 02, 03 si la instrucción es FIJAR LA RESOLUCIÓN) se ha producido un error y pasará al estado S7_1. Si el byte recibido es correcto pasará al estado S8.

- **S7_1:** Se borra el byte recibido que está guardado en la memoria de datos del ratón, también se desactiva el flag que indica que se ha recibido un byte desde el ordenador. Se borran los posibles bytes de respuesta que pudieran existir en la memoria de datos para ser enviados al ordenador a la vez que se desactivan los flags que iniciarán el envío de estos bytes. Pasará al estado S7_2.
- **S7_2:** Cargamos en el registro 1 de respuesta de la memoria de datos el byte "FE" (Reenvío) si el error se ha producido por primera vez desde la última recepción correcta, de esta forma solicitamos el reenvío del byte al ordenador. Si el error se ha producido más de una vez desde la última recepción correcta cargaremos en el registro 1 de respuesta de la memoria de datos el byte "FC" (ERROR) para indicar al ordenador que se ha producido un error. El sistema detectará que se ha guardado un byte en este registro ya que también se activará un flag cuando se guarde un byte en el registro 1 de respuesta. La activación del flag desencadenará el envío del byte de respuesta. Si el error se ha producido por primera vez pasará al estado S7_3 o pasará al estado S0 para esperar un nuevo comando si el error se ha producido más de una vez.
- **S7_3:** Se activa con un valor '1' lógico el flag "FlagError" guardado en el registro "RegistroFlagError". Un valor '0' lógico en "FlagError" indica que antes no se había producido un error en la recepción del byte. Un valor '1' lógico indica que ya se ha producido un error en la recepción del byte. El flag se desactiva con un valor '0' lógico al recibirse un byte correcto. De esta forma se sabe si el error se ha producido por primera vez (cuando había guardado un '0' lógico) o más veces (cuando había guardado un '1' lógico). Después pasa al estado S6 para esperar de nuevo la recepción del segundo byte del comando (valor de la frecuencia de muestreo o de la resolución).
- **S8:** Guardamos el byte recibido en el registro de la frecuencia de muestreo si el comando guardado en el registro "RegistroByteInstruccion" es F3. Guardamos los dos primeros bits en el registro de la resolución si el comando guardado en el registro "RegistroByteInstruccion" es E8. Así finaliza la ejecución del comando. Pasará después al estado FINAL.
- **S9:** Se ha recibido el comando FF (RESET) o el comando EC (Reset modo página) estando el sistema en el modo página. Se borra cualquier respuesta que estuviera en espera para ser enviada y guardamos el comando recibido en el registro "RegistroByteInstruccion". Pasará al estado S10.
- **S10:** En este estado se activa la señal "instruccion_guardada" con un valor '1' lógico. La parte combinacional que detecta el tipo de comando recibido toma el valor del byte de instrucción guardado en la memoria de datos externa cuando la señal "instruccion_guardada" es '0' lógico y toma el valor del byte guardado en el registro "RegistroByteInstruccion" interno cuando la señal es '1' lógico. También iniciamos el envío del byte de asentimiento ("FA") hacia el ordenador al activar la señal "enviar_asentimiento" y el borrado del comando recibido en la memoria de datos, así como la desactivación del flag que indica la presencia de un comando en la memoria de datos que es preciso ejecutar. En este estado se realiza la ejecución del comando FF (RESET) o del comando EC (Reset modo página) al activar la señal "ejecuta_instruccion_2". El reseteo del sistema se realizará al activar la señal externa "reseteo" que iniciará el reseteo del sistema ratón tras el envío de la señal de asentimiento. El

módulo encargado de este proceso es el módulo "reseteo" del ratón. El comando EC hará que el sistema ratón vuelva al estado anterior al modo página.

- **S11:** Se ha recibido un byte distinto al comando FF (RESET) o al comando EC (Reset modo página) estando el sistema en el modo página. Este byte será reenviado de nuevo al ordenador. El modo página es un modo de "eco" muy útil para realizar un test de envío en el ratón. Se borra cualquier respuesta que estuviera en espera para ser enviada y se guarda el byte recibido en el registro "RegistroByteInstruccion". Pasará al estado S12.
- **S12:** Se guarda en el registro 1 de respuesta de la memoria de datos el byte recibido y se activa el flag correspondiente al primer registro. De esta forma se iniciará el reenvío del byte recibido desde el ordenador. También se realiza el borrado del comando recibido en la memoria de datos, así como la desactivación del flag que indica la presencia de un comando en la memoria de datos que es preciso ejecutar. Pasará al estado S0 a la espera de un nuevo comando.
- **FINAL:** Una vez ejecutada la instrucción reseteamos los contadores de movimiento. Se borra la instrucción recibida y se desactiva el flag de instrucción de la memoria de datos que provocó el inicio del proceso de ejecución de la instrucción. Pasará al estado S0 a la espera de un nuevo comando.

REGISTROS DEL MÓDULO:

Una serie de registros guardará la información del sistema ratón que otros módulos utilizarán para el funcionamiento del dispositivo. El ordenador controlará al ratón mediante comandos que modifican el valor de los registros.

RegistroFrecuenciaMuestreo: Registro que almacena la frecuencia de muestreo. Este valor será la máxima frecuencia a la que se enviarán los datos de movimiento en el caso de que se produzcan movimientos. Si siempre tengo datos que enviar, será la frecuencia de envío de estos datos. Para que sea posible el envío de datos, éste debe estar habilitado y el modo de funcionamiento del ratón debe ser el modo corriente (modo "0x01"). Los posibles valores de la frecuencia de muestreo son los siguientes:
-- "00001010" => --10 muestras/seg
-- "00010100" => --20 muestras/seg
-- "00101000" => --40 muestras/seg
-- "00111100" => --60 muestras/seg
-- "01010000" => --80 muestras/seg
-- "01100100" => --100 muestras/seg
-- "11001000" => --200 muestras/seg
El valor por defecto es 100 muestras/seg, por eso, al inicializar el sistema el valor del registro como es "0x00000000", modificamos los bits 6, 5 y 2 para que la frecuencia de muestreo sea "0x01100100". El valor de la frecuencia de muestreo que cargaremos también tiene que tener estos bits modificados para que a la salida tengamos el valor correcto.
La carga de la frecuencia de muestreo se realizará cuando se reciba el comando 0xF3 (Fijar la frecuencia de muestro) o el comando 0xF6 (Fijar los valores por defecto). Cuando la carga del registro sea a través del comando 0xF3, el valor almacenado de la frecuencia de muestreo será el byte recibido tras el comando. La frecuencia de muestreo es utilizada para el envío de los datos

de movimiento con esa misma frecuencia y cuando tenga que enviar al ordenador el estado del ratón.

RegistroResolucion: Registro que almacena la resolución. Este valor será la resolución con la cual el ratón toma los datos de movimiento, esto es, a mayor resolución, mayor será el incremento de los contadores de movimiento cada vez que se detecte un movimiento en alguna dirección. Los posibles valores de la resolución serán los siguientes:
-- "00" => -- 1 cuenta/mm (incrementamos en 1 el contador de movimiento al detectarse)
-- "01" => -- 2 cuentas/mm (incrementamos en 2 el contador de movimiento al detectarse)
-- "10" => -- 4 cuentas/mm (incrementamos en 4 el contador de movimiento al detectarse)
-- "11" => -- 8 cuentas/mm (incrementamos en 8 el contador de movimiento al detectarse)
El valor por defecto es 4 cuentas/mm, por eso, al inicializar el sistema el valor del registro como es "0x00", modificamos el bit 1 para que la resolución sea "0x10". El valor de la resolución que cargaremos también tiene que tener estos bits modificados para que a la salida tengamos el valor correcto.
La carga de la resolución se realizará cuando se reciba el comando 0xE8 (Fijar la resolución) o el comando 0xF6 (Fijar los valores por defecto). Cuando la carga del registro sea a través del comando 0xE8, el valor almacenado de la frecuencia de muestreo será el byte recibido tras el comando.
La resolución es utilizada para el incremento del contador de movimiento cuando se detecte un movimiento. Una vez se cree el evento de envío con la periodicidad dada por la frecuencia de muestreo, se enviará al ordenador el valor de los contadores de movimiento si son distintos de cero.

RegistroEscala: Registro que almacena la escala. Este parámetro no afecta a los contadores de movimiento, pero sí afecta al valor efectivo de estos contadores. Por defecto, el ratón usa la escala 1:1, que no tiene ningún efecto en el movimiento efectivo del ratón. Sin embargo, el host puede seleccionar el escalado 1:2 al enviar el comando "Set Scaling 2:1" (ØxE7) ("Fijar el escalado 2:1"). Si el escalado 2:1 está activo, el ratón aplicará el siguiente algoritmo a los contadores antes de enviar su contenido al host.

--	Contador de movimiento	Movimiento efectivo
--	0	0
--	1	1
--	2	1
--	3	3
--	4	6
--	5	9
--	N>5	2*N

Los posibles valores del escalado son:
-- '0' => -- Escala 1:1
-- '1' => -- Escala 2:1
El valor por defecto es Escala 1:1, por eso, al inicializar el sistema el valor del registro toma el valor correcto. La carga del escalado se realizará cuando se reciba el comando 0xE6 (Fijar la escala 1:1) ,el comando 0xF7 (Fijar la escala 2:1) o el comando 0xF6 (Fijar los valores por defecto).

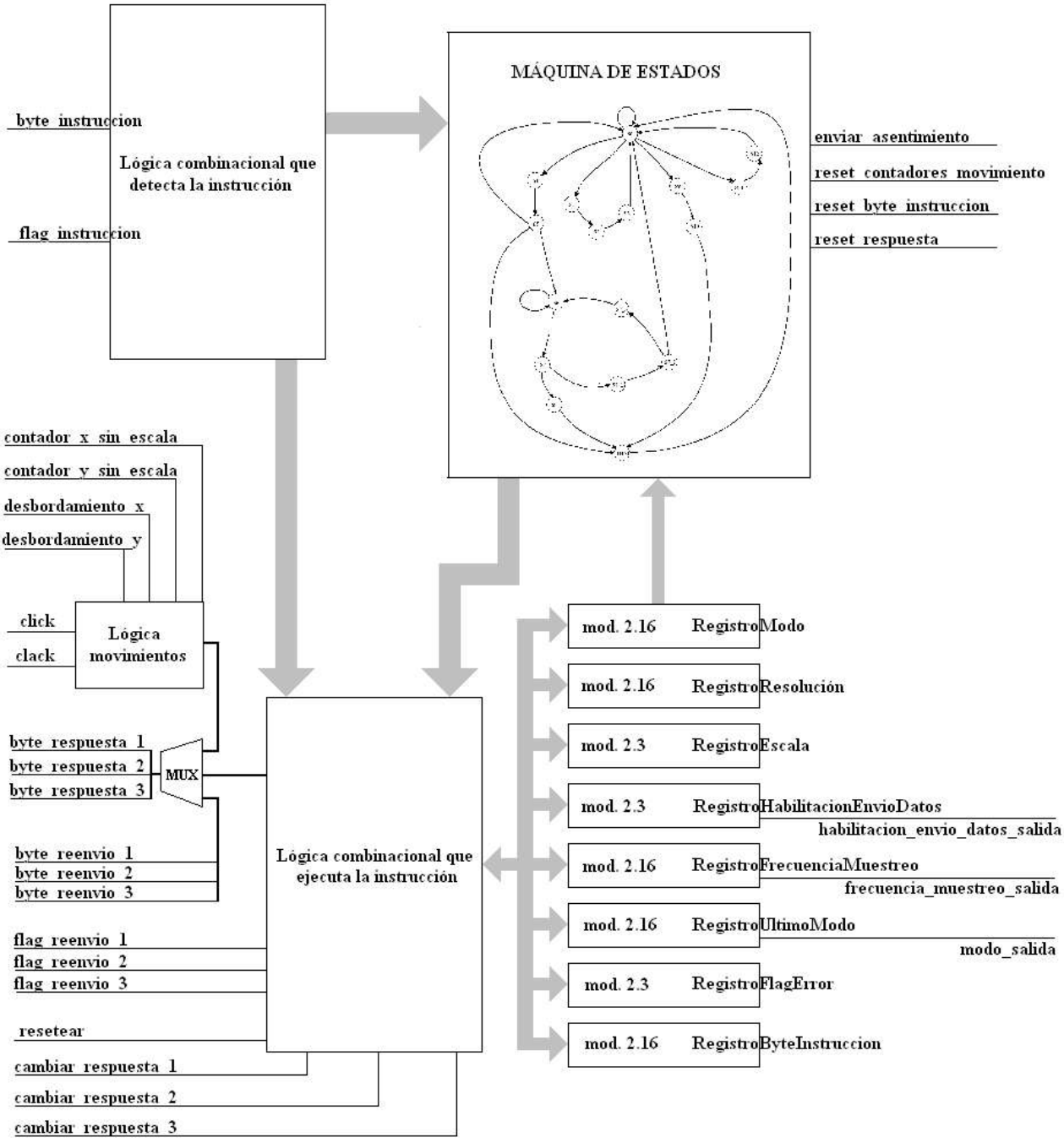
El escalado es utilizado para la modificación del valor del contador de movimiento a enviar. Una vez se cree el evento de envío con la periodicidad dada por la frecuencia de muestreo, se enviará al ordenador el valor de los contadores de movimiento con el escalado realizado si son distintos a cero.

RegistroHabilitacionEnvioDatos: Registro que almacena el flag de habilitación de envío de datos. Sólo se enviarán datos de movimiento al ordenador cuando el ratón esté en el modo de funcionamiento corriente "0x01" y tenga el flag de envío de datos habilitado con un valor '1' lógico. En caso de que se dieran las anteriores circunstancias se enviarán al ordenador los datos de movimiento con la periodicidad dada por la frecuencia de muestreo una vez se cree el evento de envío.
Los posibles valores de este flag son:
-- '0' => -- Deshabilitado
-- '1' => -- Habilitado
El valor por defecto es el envío de datos deshabilitado, por eso, al inicializar el sistema el valor del registro toma el valor correcto. La habilitación del envío de datos se realizará cuando se reciba el comando 0xF4 (Habilitar el envío de datos), y sólo tendrá efecto si se realiza en el modo corriente. La deshabilitación del envío de datos se realizará cuando se reciba el comando 0xF5 (Deshabilitar el envío de datos). De esta forma controlamos el envío de los datos de movimiento.

RegistroModo: Registro que almacena el modo de funcionamiento del ratón. El ratón tiene 3 modos de funcionamiento, y en este registro se almacena la información concerniente al modo de operación.

- En el modo Corriente, el ratón envía datos de movimiento cuando detecta un movimiento o un cambio en el estado de uno o más de los botones del ratón. La máxima tasa a la que el envío de datos puede ocurrir es conocida como frecuencia de muestreo. El rango de este parámetro va desde 10 muestras/segundo a 200 muestras/segundo. Su valor por defecto es 100 muestras/segundo y el host puede cambiar ese valor al usar el comando "Fijar la frecuencia de muestreo" (ØxF3). El modo Corriente es el modo de operación por defecto.
- En el modo remoto, el ratón lee sus entradas y actualiza sus contadores y flags a la frecuencia de muestreo actual, pero solamente mandará al host los movimientos (y cambios en el estado de botones) cuando esa información sea requerida por el host. El host solicita información al enviar el comando de "lectura de datos" (ØxEB). Después de recibir este comando, el ratón enviará un paquete de datos de movimiento, y reseteará sus contadores de movimiento.
- El modo página es un "modo de eco" en el cual cada byte recibido por el ratón es reenviado al host. Incluso si el byte representa un comando válido, el ratón no responderá al comando (lo único que hará será reenviar el byte de regreso al host). Hay dos excepciones a esto: el comando "Reset" (ØxFF) y el comando de "Reset de Modo Página" (ØxEC). El ratón toma estos comandos como comandos válidos y no los devuelve de nuevo al host.

Los posibles valores del modo son:
-- "00" => -- Ninguno
-- "01" => -- Corriente
-- "10" => -- Remoto
-- "11" => -- Página
El valor por defecto el modo corriente, por eso, al inicializar el sistema el valor del registro como es "0x00", modificamos el bit 0 para que el modo sea "0x01". El valor del modo que cargaremos



también tiene que tener este bit modificado para que a la salida tengamos el valor correcto. La carga del modo se realizará cuando se reciba el comando 0xF0 (Activar el modo remoto) ,el comando 0xEE (Activar el modo página), el comando 0xEC (Reseteo del modo página y volvemos al modo anterior a la entrada al modo página) o el comando 0xEA (Activar el modo corriente). Cuando reseteamos el modo página, estando en el modo página, guardamos en el

registro el valor almacenado en el registro "RegistroUltimoModo", donde se almacenó el valor del modo en el que estaba el sistema antes de activar el modo página.

RegistroUltimoModo: En este registro se guardará el valor del modo cada vez que se modifique menos por el evento causado por la recepción del comando 0xEC (Reseteo del modo página y vuelta al modo anterior a la entrada al modo página) y el evento causado por la recepción del comando 0xEE (Activar el modo página). Así, si el sistema entra en modo página se tendrá guardado en este registro el modo anterior a la entrada en modo página. Si se recibe el comando 0xEC, se guardará en el registro "RegistroModo" el valor almacenado en este registro y el sistema volverá al modo anterior a la entrada en el modo página.

RegisgroFlagError: En este registro se guardará el flag "FlagError". Un valor '0' lógico en "FlagError" indica que antes no se había producido un error en la recepción del byte de instrucción. Un valor '1' lógico indica que ya se ha producido un error en la recepción del byte de instrucción. El flag se desactiva con un valor '0' lógico al recibirse un byte correcto. De esta forma se sabe si el error se ha producido por primera vez (cuando había guardado un '0' lógico) o más veces (cuando había guardado un '1' lógico). La máquina de estados utilizará este flag para decidir la operación a realizar cuando se produzca un error.

RegistroByteInstruccion: En este registro se almacenará el byte recibido para un posterior uso del mismo. El comando recibido se almacenará en este registro y se podrá borrar el registro de la memoria de datos donde se guarda la instrucción recibida. Así, al terminar el ciclo de la máquina de estados ya no habrá ninguna instrucción en la memoria de datos para ejecutar y se tendrá almacenada la instrucción para decidir las señales que se activarán durante el proceso de ejecución de la señal. En este registro también se guardará el dato recibido para almacenar en el registro de la frecuencia de muestreo y la resolución.

INTERFAZ DEL MÓDULO:

- RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- FRECUENCIA_MUESTREO_SALIDA: Vector de bits que proporciona la frecuencia de muestreo. Este valor será la máxima frecuencia a la que se enviarán los datos de movimiento en el caso de que se produzcan movimientos. Los posibles valores de la frecuencia de muestreo son los siguientes:

-- "00001010" => --10 muestras/seg
-- "00010100" => --20 muestras/seg
-- "00101000" => --40 muestras/seg
-- "00111100" => --60 muestras/seg
-- "01010000" => --80 muestras/seg
-- "01100100" => --100 muestras/seg
-- "11001000" => --200 muestras/seg

- HABILITACION_ENVIO_DATOS_SALIDA: Bit que proporciona el flag de habilitación de envío de datos. Sólo se enviarán datos de movimiento al ordenador cuando el ratón esté en el modo de funcionamiento corriente "0x01" y tenga el flag de envío de datos habilitado con un valor '1' lógico. Los posibles valores de este flag son:

-- '0' => -- Deshabilitado
-- '1' => -- Habilitado

- MODO_SALIDA: Vector de bits que informa del modo de funcionamiento del ratón. Los posibles valores del modo son:

-- "00" => -- Ninguno
-- "01" => -- Corriente
-- "10" => -- Remoto
-- "11" => -- Página

- RESETEAR: Esta señal iniciará el proceso de reseteo una vez termine el envío del asentimiento tras recibir el comando FF (RESET).
- ENVIAR_ASENTIMIENTO: Al activar esta señal se solicitará el envío de un asentimiento (FA) ante un comando recibido.
- CONTADOR_X_SIN_ESCALA: Contador de movimiento en la dirección horizontal de 9 bits en complemento a dos sin realizar el escalado. Será empaquetado y enviado al ordenador al recibirse el comando EB (Lectura de datos).
- CONTADOR_Y_SIN_ESCALA: Contador de movimiento en la dirección vertical de 9 bits en complemento a dos sin realizar el escalado. Será empaquetado y enviado al ordenador al recibirse el comando EB (Lectura de datos).
- DESBORDAMIENTO_X: Esta señal estará activa cuando se haya producido un desbordamiento positivo o negativo en la cuenta llevada a cabo por el contador de movimiento horizontal.
- DESBORDAMIENTO_Y: Esta señal estará activa cuando se haya producido un desbordamiento positivo o negativo en la cuenta llevada a cabo por el contador de movimiento vertical.
- CLACK: Esta señal estará activa cuando se pulse el botón derecho del ratón.
- CLICK: Esta señal estará activa cuando se pulse el botón izquierdo del ratón.
- RESET_CONTADORES_MOVIMIENTO: Tras la ejecución de los comandos se resetean los contadores de movimiento. Al activar esta señal se borra cualquier evidencia de movimiento, reiniciándose la cuenta de los contadores de movimiento.
- BYTE_INSTRUCCION: Byte almacenado en el registro de instrucción de la memoria de datos que se corresponde con el byte recibido desde el ordenador.
- FLAG_INSTRUCCION: Si este flag está activo con un valor '1' lógico indica que se ha recibido un byte desde el ordenador (normalmente un comando) al que hay que prestarle atención. Este flag pone en funcionamiento la máquina de estados encargada de la ejecución del comando recibido.
- RESET_BYTE_INSTRUCCION: Una vez ejecutado el comando se desactivará el flag "FLAG_INSTRUCCION" al poner un '1' lógico en esta señal. De esta forma ya no habrá ninguna instrucción para ser ejecutada.
- BYTE_REENVIO_1: Primer byte que se enviará al ordenador si se recibe la instrucción FE (REENVÍO) y está activo el primer flag de reenvío.
- BYTE_REENVIO_2: Segundo byte que se enviará al ordenador si se recibe la instrucción FE (REENVÍO) y está activo el segundo flag de reenvío.

- BYTE_REENVIO_3: Tercer byte que se enviará al ordenador si se recibe la instrucción FE (REENVÍO) y está activo el tercer flag de reenvío.
- FLAG_REENVIO_1: Este flag indica cuando está activo que si se recibe el comando FE (REENVÍO) debe enviarse el byte "BYTE_REENVÍO_1".
- FLAG_REENVIO_2: Este flag indica cuando está activo que si se recibe el comando FE (REENVÍO) debe enviarse el byte "BYTE_REENVÍO_2".
- FLAG_REENVIO_3: Este flag indica cuando está activo que si se recibe el comando FE (REENVÍO) debe enviarse el byte "BYTE_REENVÍO_3".
- RESET_RESPUESTA: Al activar esta señal se borra cualquier tipo de respuesta que estuviera pendiente de ser enviada al ordenador. Ya no son de utilidad los datos guardados en los registros de respuesta de la memoria de datos.
- CAMBIAR_RESPUESTA_1: Al activar esta señal con un valor '1' lógico se almacenará en el registro primero de respuesta de la memoria de datos el byte "BYTE_RESPUESTA_1". También se activará el flag correspondiente a la primera respuesta, lo cual desencadenará el envío de este byte al ordenador como respuesta a un comando.
- BYTE_RESPUESTA_1: Byte que se almacenará en el registro primero de respuesta de la memoria de datos al activar la señal "CAMBIAR_RESPUESTA_1". Este será el primer byte de respuesta al comando recibido.
- CAMBIAR_RESPUESTA_2: Al activar esta señal con un valor '1' lógico se almacenará en el registro segundo de respuesta de la memoria de datos el byte "BYTE_RESPUESTA_2". También se activará el flag correspondiente a la segunda respuesta, lo cual desencadenará el envío de este byte al ordenador como respuesta a un comando.
- BYTE_RESPUESTA_2: Byte que se almacenará en el registro segundo de respuesta de la memoria de datos al activar la señal "CAMBIAR_RESPUESTA_2". Este será el segundo byte de respuesta al comando recibido.
- CAMBIAR_RESPUESTA_3: Al activar esta señal con un valor '1' lógico se almacenará en el registro tercero de respuesta de la memoria de datos el byte "BYTE_RESPUESTA_3". También se activará el flag correspondiente a la tercera respuesta, lo cual desencadenará el envío de este byte al ordenador como respuesta a un comando.
- BYTE_RESPUESTA_3: Byte que se almacenará en el registro tercero de respuesta de la memoria de datos al activar la señal "CAMBIAR_RESPUESTA_3". Este será el tercer byte de respuesta al comando recibido.

1.2.3.11 sistema_activar_movimiento.vhd

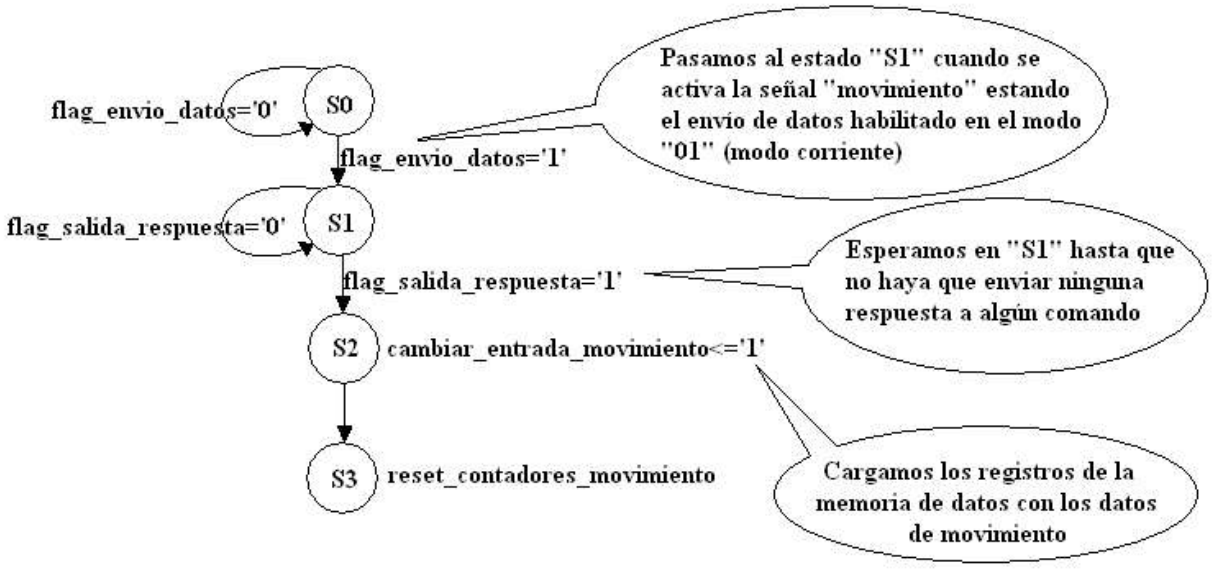
```
ENTITY sistema_activar_movimiento IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    flag_salida_respuesta_1: IN STD_LOGIC;
    flag_salida_respuesta_2: IN STD_LOGIC;
    flag_salida_respuesta_3: IN STD_LOGIC;
    movimiento: IN STD_LOGIC;
    contador_x_con_escala: IN STD_LOGIC_VECTOR (8 DOWNTO 0);
    desbordamiento_x: IN STD_LOGIC;
    contador_y_con_escala: IN STD_LOGIC_VECTOR (8 DOWNTO 0);
    desbordamiento_y: IN STD_LOGIC;
    click: IN STD_LOGIC;
    clack: IN STD_LOGIC;
    habilitacion_envio_datos: IN STD_LOGIC;
    modo: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    reset_contadores_movimiento: OUT STD_LOGIC;
    byte_entrada_movimiento_1: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_entrada_movimiento_2: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    byte_entrada_movimiento_3: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    cambiar_entrada_movimiento_1: OUT STD_LOGIC;
    cambiar_entrada_movimiento_2: OUT STD_LOGIC;
    cambiar_entrada_movimiento_3: OUT STD_LOGIC
  );
END sistema_activar_movimiento;
```

Este módulo es el encargado de guardar en los registros de la memoria de datos del ratón los bytes de movimiento que serán enviados al ordenador para indicarle el estado del interfaz ratón-usuario.

Controlamos la modificación de los registros que guardan los datos de movimiento en la memoria de datos del ratón. También se controla la activación de los flags de la misma memoria que indican que es preciso enviar los bytes de movimiento. Se activarán 3 flags si se ha detectado un movimiento. Otro módulo detectará el valor de estos flags y se procederá al envío de los bytes de movimiento correspondientes. Cada vez que se envía un byte de movimiento, se desactiva el flag que originó el envío, y así hasta que hayan sido enviados los 3 bytes de movimiento.

Una máquina de estados controlará el funcionamiento del módulo. Inicia su funcionalidad cuando se recibe la señal “movimiento” con un valor '1' lógico. El envío de datos debe estar habilitado y el modo debe ser el modo “01” en hexadecimal (modo corriente). Si el ratón está todavía pendiente de enviar datos de respuesta a algún comando (flag_salida_respuesta_1 a '1' lógico => pendiente de enviar el byte de respuesta 1, flag_salida_respuesta_2 a '1' lógico => pendiente de enviar el byte de respuesta 2, flag_salida_respuesta_3 a '1' lógico => pendiente de enviar el byte de respuesta 3), esperará en el estado “s1” hasta que los bytes de respuesta hayan sido enviados. Luego guardará en los bytes de movimiento de los registros de la memoria los 3 bytes de movimiento que informarán al ratón del estado de los contadores de movimiento y de los botones del ratón. Activará los flags de movimiento presentes en la memoria de datos del ratón, para indicar que se tienen que enviar 3 bytes de movimiento. A medida que estos bytes de movimiento sean enviados, estos flags se desactivarán. Esto se realiza en la máquina de estados al activar la señal “cambiar_entrada_movimiento” con un valor '1' lógico. Posteriormente, se

resetearán los contadores de movimiento activando la señal "reset_contadores_movimiento" para reiniciar el proceso de detección de movimientos en el módulo "movimientos_new" del ratón.



Los bytes de movimiento que se enviarán al ordenador son guardados en los registros destinados a tal efecto presentes en el componente "memoria_datos" del ratón. En ellos se guardarán los 3 bytes de movimiento necesarios para indicar el estado de la interfaz ratón-usuario.
El primer byte de movimiento que se enviará al ordenador llevará la siguiente información:

- Bits de desbordamiento de los contadores de movimiento en la dirección horizontal y vertical. Si los contadores de movimiento han sufrido un movimiento tal que se ha sobrepasado la capacidad de los mismos, estos bits se activarán con un valor '1' lógico según corresponda a la dirección x o y.
- Signo de los contadores de movimiento. Si el signo del contador en la dirección x es positivo, el movimiento será hacia la derecha. Si el signo del contador en la dirección x es negativo, el movimiento será hacia la izquierda. Si el signo del contador en la dirección y es positivo, el movimiento será hacia arriba. Si el signo del contador en la dirección y es negativo, el movimiento será hacia abajo. El bit más significativo de los contadores de movimiento representa al signo.
- Estado de pulsación del botón izquierdo (CLICK).
- Estado de pulsación del botón derecho (CLACK).

En el segundo byte de movimiento que se enviará al ordenador se transmitirán los 8 bits menos significativos del contador de movimiento en la dirección horizontal en complemento a dos.
En el tercer byte de movimiento que se enviará al ordenador se transmitirán los 8 bits menos significativos del contador de movimiento en la dirección vertical en complemento a dos.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Desborda-miento de Y	Desborda-miento de X	Bit de signo de Y	Bit de signo de X	Siempre a 1	Botón del medio	Botón derecho	Botón izquierdo
Byte 2	MOVIMIENTO X							
Byte 3	MOVIMIENTO Y							

El interfaz del módulo es el siguiente:

- RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- FLAG_SALIDA_RESPUESTA_1: Esta señal indica cuando está activa con un valor '1' lógico que todavía queda por enviar el primer byte de respuesta a un comando anterior recibido desde el ordenador. No se debe enviar ningún dato de movimiento hacia el ordenador hasta que no haya sido enviado el byte de respuesta.
- FLAG_SALIDA_RESPUESTA_2: Esta señal indica cuando está activa con un valor '1' lógico que todavía queda por enviar el segundo byte de respuesta a un comando anterior recibido desde el ordenador. No se debe enviar ningún dato de movimiento hacia el ordenador hasta que no haya sido enviado el byte de respuesta.
- FLAG_SALIDA_RESPUESTA_3: Esta señal indica cuando está activa con un valor '1' lógico que todavía queda por enviar el tercer byte de respuesta a un comando anterior recibido desde el ordenador. No se debe enviar ningún dato de movimiento hacia el ordenador hasta que no haya sido enviado el byte de respuesta.
- MOVIMIENTO: Señal procedente del módulo "movimientos_new" que indica que se ha producido un movimiento y debe iniciarse el proceso de envío de los 3 bytes de movimiento. Esta señal es generada de acuerdo con la frecuencia de muestreo activa en el sistema ratón, de tal forma que la frecuencia máxima a la cual se recibiría la señal "movimiento" en el caso de que siempre hubiera que transmitir datos de movimiento, sería la frecuencia de muestreo del sistema ratón.
- CONTADOR_X_CON_ESCALA: Valor del contador de movimientos en la dirección horizontal en complemento a dos con el escalado realizado.
- DESBORDAMIENTO_X: Señal que se activará si se ha producido un desbordamiento del contador de movimiento en el eje horizontal.
- CONTADOR_Y_CON_ESCALA: Valor del contador de movimientos en la dirección vertical en complemento a dos con el escalado realizado.
- DESBORDAMIENTO_Y: Señal que se activará si se ha producido un desbordamiento del contador de movimiento en el eje vertical.
- CLICK: Señal que indica el estado de pulsación del botón izquierdo (CLICK), activa cuando el botón es pulsado.

- CLACK: Señal que indica el estado de pulsación del botón izquierdo (CLICK), activa cuando el botón es pulsado.
- HABILITACION_ENVIO_DATOS: Cuando el envío de datos está habilitado esta señal está a '1' lógico.
- MODO: Modo de funcionamiento actual del ratón: ("01" => Corriente, "10" => Remoto, "11" => Pagina)
- RESET_CONTADORES_MOVIMIENTO: Resetea los contadores de movimiento para reiniciar una nueva adquisición de movimientos al activar con un valor '1' lógico esta señal. Realizamos el reseteo cuando se ha iniciado el proceso de envío de datos de movimiento desde el ratón hacia el ordenador.
- BYTE_ENTRADA_MOVIMIENTO_1: Primer byte de movimiento que se enviará al ordenador. Contiene la siguiente información: a)Bits de desbordamiento de los contadores de movimiento en la dirección horizontal y vertical. b)Signo de los contadores de movimiento. c) Estado de pulsación de los botones del ratón. Este byte será cargado en el registro correspondiente de la memoria de datos del ratón al activar la señal "cambiar_entrada_movimiento_1".
- BYTE_ENTRADA_MOVIMIENTO_2: Segundo byte de movimiento que se enviará al ordenador. Contiene los 8 bits menos significativos del contador de movimiento en la dirección horizontal en complemento a dos.
- BYTE_ENTRADA_MOVIMIENTO_3: Tercer byte de movimiento que se enviará al ordenador. Contiene los 8 bits menos significativos del contador de movimiento en la dirección vertical en complemento a dos.
- CAMBIAR_ENTRADA_MOVIMIENTO_1: Señal que cargará el primer byte de movimiento en el registro 1 de movimiento de la memoria de datos. También activará el flag 1 de movimiento con un valor '1' lógico, el cual al ser detectado provocará el envío del primer byte de movimiento.
- CAMBIAR_ENTRADA_MOVIMIENTO_2: Señal que cargará el segundo byte de movimiento en el registro 2 de movimiento de la memoria de datos. También activará el flag 2 de movimiento con un valor '1' lógico, el cual al ser detectado provocará el envío del segundo byte de movimiento si el flag 1 está desactivado.
- CAMBIAR_ENTRADA_MOVIMIENTO_3: Señal que cargará el tercer byte de movimiento en el registro 3 de movimiento de la memoria de datos. También activará el flag 3 de movimiento con un valor '1' lógico, el cual al ser detectado provocará el envío del tercer byte de movimiento si los flags 1 y 2 están desactivados.

1.2.3.12 movimientos_new.vhd

```
ENTITY movimientos_new IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    flanco_subida_5us: IN STD_LOGIC;
    arriba: IN STD_LOGIC;
    derecha: IN STD_LOGIC;
    abajo: IN STD_LOGIC;
    izquierda: IN STD_LOGIC;
    click: IN STD_LOGIC;
    clack: IN STD_LOGIC;
    contador_x_sin_escala: OUT STD_LOGIC_VECTOR (8 downto 0);
    contador_x_con_escala: OUT STD_LOGIC_VECTOR (8 downto 0);
    desbordamiento_x: OUT STD_LOGIC;
    contador_y_sin_escala: OUT STD_LOGIC_VECTOR (8 downto 0);
    contador_y_con_escala: OUT STD_LOGIC_VECTOR (8 downto 0);
    desbordamiento_y: OUT STD_LOGIC;
    movimiento: OUT STD_LOGIC;
    reset_contadores_movimiento: IN STD_LOGIC
  );
END movimientos_new;
```

Este módulo es el encargado de avisar al sistema ratón cuando sea preciso el envío de datos de movimiento.

Cuando se detecte un movimiento mediante la activación de alguna de las señales de movimiento (derecha, izquierda, arriba, abajo) o se produzca una modificación en el estado de pulsación de algún botón (botón derecho – clack, botón izquierdo – click) se activará la señal "movimiento". Se avisará al sistema ratón mediante la activación de la señal "movimiento" con un valor uno lógico.

Las señales de movimiento en el sentido horizontal (derecha e izquierda) provocarán cambios en el contador de movimientos del eje x. Si se detecta un movimiento hacia la derecha se propondrá como valor de este contador el valor "000000100" (4 en decimal). Si se detecta un movimiento hacia la izquierda se propondrá como valor de este contador el valor "111111100" (- 4 en decimal).

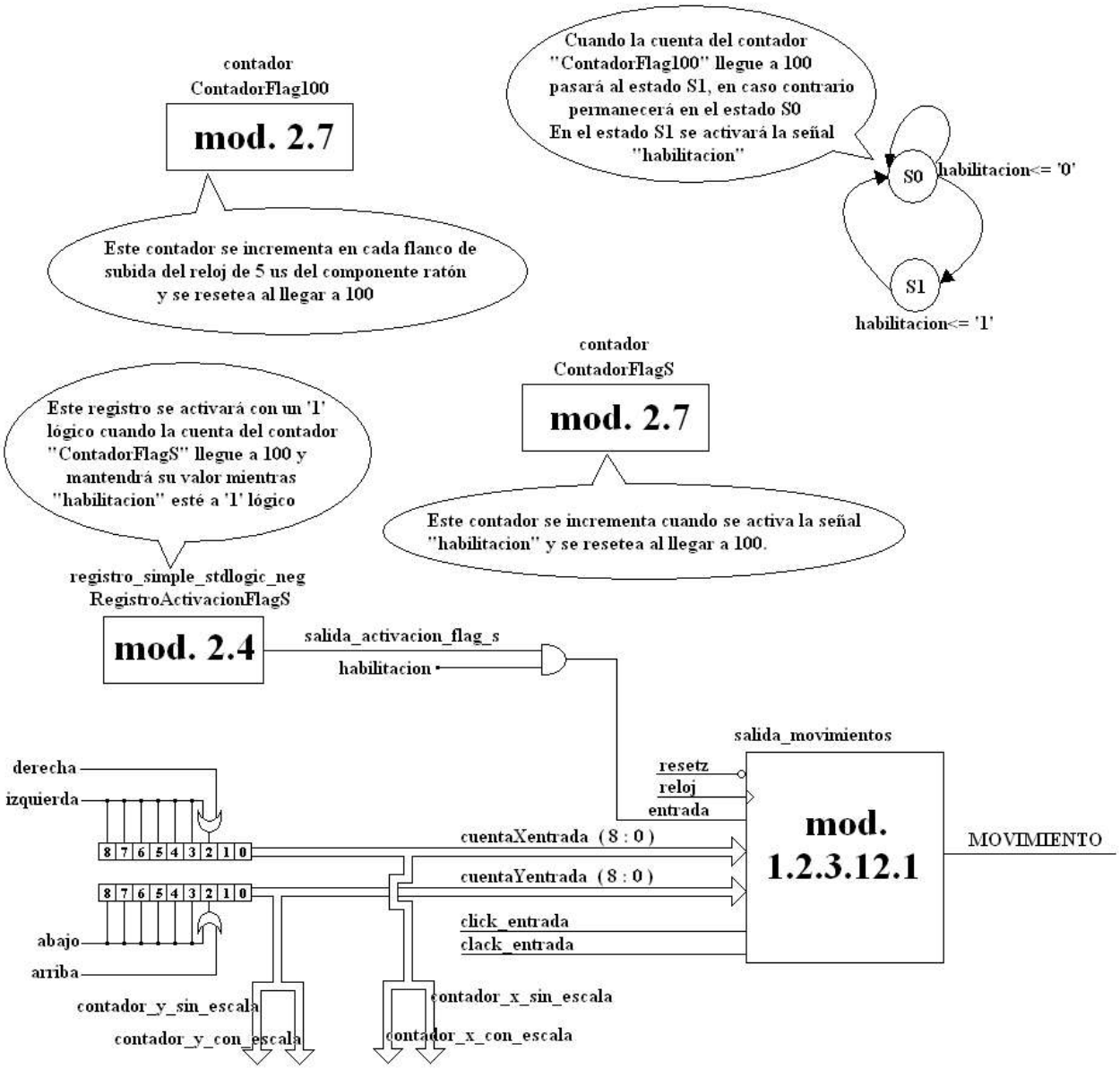
Las señales de movimiento en el sentido vertical (arriba y abajo) provocarán cambios en el contador de movimientos del eje y. Si se detecta un movimiento hacia arriba se propondrá como valor de este contador el valor "000000100" (4 en decimal). Si se detecta un movimiento hacia abajo se propondrá como valor de este contador el valor "111111100" (- 4 en decimal).

Esta comprobación sólo se realizará cada 50 ms, y en el caso de que el valor de alguno de los contadores anteriores sea distinto de cero, se activará la señal "movimiento" durante 200 ns para indicar al sistema ratón que se ha producido un movimiento. Se mostrarán además a la salida del módulo los valores de los contadores de movimiento en el eje horizontal (x) y en el eje vertical (y). Estos contadores se mostrarán con escalado y sin escalado.

También se comprobará cada 50 ms si se ha producido una variación en el estado de pulsación de los botones del ratón. Esto es, si antes no estaba pulsado uno de los botones y ahora lo está, o si

antes estaba pulsado uno de los botones y ahora no lo está. Si se ha producido un cambio en el estado de pulsación de alguno de los botones (botón derecho – clack, botón izquierdo – click) también se activará la señal “movimiento”.

La señal “movimiento” provocará el inicio del envío de los datos de movimiento hacia el ordenador.



El interfaz del módulo es el siguiente:

- RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- FLANCO_SUBIDA_5US: Señal activa durante 200 ns para marcar el flanco de subida del reloj de 5 us utilizado en el componente ratón. Esta señal sincroniza los procesos en el ratón y permite trabajar con unos tiempos mayores que el reloj del sistema de 200 ns. Los procesos en el módulo ratón son más lentos que en el reconocedor.
- ARRIBA: Señal que indica un movimiento del ratón hacia arriba cuando está activada con un valor '1' lógico.
- DERECHA: Señal que indica un movimiento del ratón hacia la derecha cuando está activada con un valor '1' lógico.
- ABAJO: Señal que indica un movimiento del ratón hacia abajo cuando está activada con un valor '1' lógico.
- IZQUIERDA: Señal que indica un movimiento del ratón hacia la izquierda cuando está activada con un valor '1' lógico.
- CLICK: Señal que indica que se ha pulsado el botón izquierdo cuando está activada con un valor '1' lógico.
- CLACK: Señal que indica que se ha pulsado el botón derecho cuando está activada con un valor '1' lógico.
- CONTADOR_X_SIN_ESCALA: Valor del contador de movimiento del ratón en la dirección horizontal sin realizar el escalado del movimiento. Si es distinto de cero se ha producido un movimiento hacia la derecha o hacia la izquierda. Cuanto mayor sea el valor del contador en complemento a dos (positivo), mayor será el movimiento hacia la derecha producido. Cuanto menor sea el valor del contador en complemento a dos (negativo), mayor será el movimiento hacia la izquierda.
- CONTADOR_X_CON_ESCALA: Valor del contador de movimiento del ratón en la dirección horizontal con el escalado realizado. Si es distinto de cero se ha producido un movimiento hacia la derecha o hacia la izquierda. Cuanto mayor sea el valor del contador en complemento a dos (positivo), mayor será el movimiento hacia la derecha producido. Cuanto menor sea el valor del contador en complemento a dos (negativo), mayor será el movimiento hacia la izquierda.
- DESBORDAMIENTO_X: Esta señal se activará con un valor lógico '1' si el movimiento en la dirección horizontal ha sido tal que se ha producido un desbordamiento del contador en la dirección x.
- CONTADOR_Y_SIN_ESCALA: Valor del contador de movimiento del ratón en la dirección vertical sin realizar el escalado del movimiento. Si es distinto de cero se ha producido un movimiento hacia arriba o hacia abajo. Cuanto mayor sea el valor del contador en complemento a dos (positivo), mayor será el movimiento hacia arriba producido. Cuanto menor sea el valor del contador en complemento a dos (negativo), mayor será el movimiento hacia abajo.
- CONTADOR_Y_CON_ESCALA: Valor del contador de movimiento del ratón en la dirección vertical con el escalado realizado. Si es distinto de cero se ha producido un movimiento hacia arriba o hacia abajo. Cuanto mayor sea el valor del contador en complemento a dos (positivo), mayor será el movimiento hacia arriba producido. Cuanto

menor sea el valor del contador en complemento a dos (negativo), mayor será el movimiento hacia abajo.

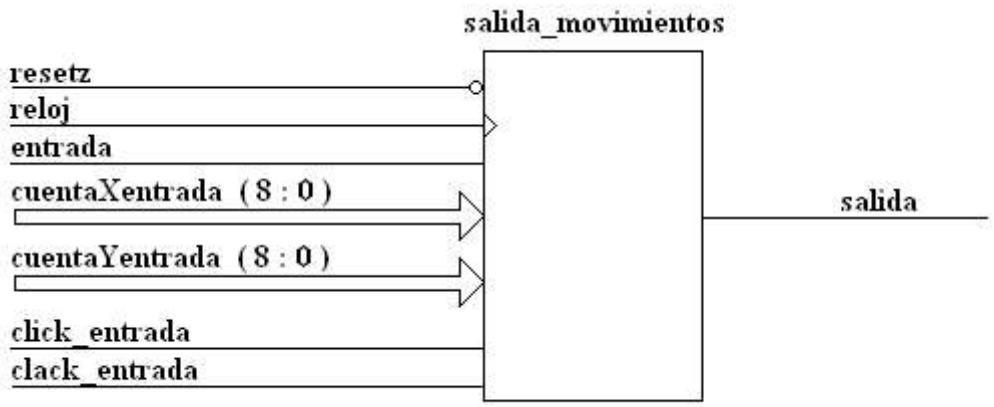
- DESBORDAMIENTO_Y: Esta señal se activará con un valor lógico '1' si el movimiento en la dirección vertical ha sido tal que se ha producido un desbordamiento del contador en la dirección y.
- MOVIMIENTO: Señal de salida que indicará si se ha producido algún evento en el movimiento o en la pulsación de los botones del ratón. Esta comprobación se hará cada 50 ms y en caso de que el evento se hubiera producido, se activará durante 100 ms.
- RESET_CONTADORES_MOVIMIENTO: Cuando se activa con un valor uno lógico se producirá el reseteo de todos los contadores del módulo, para reiniciar el proceso de adquisición de movimientos.

1.2.3.12.1 salida_movimientos.vhd

```
ENTITY salida_movimientos IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    entrada: IN STD_LOGIC;
    cuentaXentrada: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
    cuentaYentrada: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
    click_entrada: IN STD_LOGIC;
    clack_entrada: IN STD_LOGIC;
    salida: OUT STD_LOGIC
  );
END salida_movimientos;
```

Módulo perteneciente al módulo "movimientos_new". Este módulo es el encargado de avisar al sistema ratón cuando sea preciso el envío de datos de movimiento.

Será preciso avisar al sistema cuando se detecte un movimiento (derecha, izquierda, arriba, abajo) o se produzca una modificación en el estado de pulsación de algún botón (botón derecho – clack, botón izquierdo – click). Cada 50 ms se activará la señal "entrada" durante 200 ns con un valor uno lógico, en ese momento se revisan los contadores de movimiento en la dirección "x" y en la direccción "y", también se revisa el estado de los botones. Si alguno de los contadores es distinto de cero o si se ha producido una variación en el estado de pulsación de alguno de los botones, se activará la señal "salida" durante 200 ns con un valor uno lógico.



La señal "salida" se activará cuando alguno de los contadores de entrada sea distinto de cero, indicando que se ha producido un movimiento en el eje horizontal o en el eje vertical. También se activará cuando se produzca una variación en uno de los botones, es decir, que pase de estar pulsado a no estarlo, o pase de no estar pulsado a estarlo. Esta activación de la señal "salida" sólo se producirá en los instantes en los que se active la señal "entrada", para así realizar una comprobación cada 50 ms. Por lo tanto, la señal salida se activará cada 50 ms si ha ocurrido un evento en interfaz ratón-usuario.

El interfaz del módulo es el siguiente:

- RESETZ: Señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea la información guardada en el módulo. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: Señal de reloj utilizada para el funcionamiento síncrono del dispositivo.
- ENTRADA: Señal que iniciará cada 50 ms la comprobación del estado de movimiento o de pulsación de los botones del ratón.
- CUENTAXENTRADA: Este vector indica el valor del contador de movimientos en el eje horizontal.
- CUENTAyENTRADA: Este vector indica el valor del contador de movimientos en el eje vertical.
- CLICK_ENTRADA: Esta señal indica que está siendo pulsado el botón izquierdo del ratón.
- CLACK_ENTRADA: Esta señal indica que está siendo pulsado el botón derecho del ratón.
- SALIDA: Señal de salida que indicará si se ha producido algún evento en el movimiento o en la pulsación de los botones del ratón. Esta comprobación se hará cada 50 ms en un proceso iniciado.

1.2.3.13	reseteo.vhd

```
ENTITY reseteo IS
  PORT(
    reloj: IN STD_LOGIC;
    resetz: IN STD_LOGIC;
    reseteo: IN STD_LOGIC;
    enviar_asentimiento: IN STD_LOGIC;
    resetz_global: OUT STD_LOGIC
  );
END reseteo;
```

Este módulo controla el reseteo programado del sistema que controla el protocolo PS/2. Al recibir el comando ØxFF (RESET) se activa este módulo que provocará el reseteo de todo el sistema PS/2.

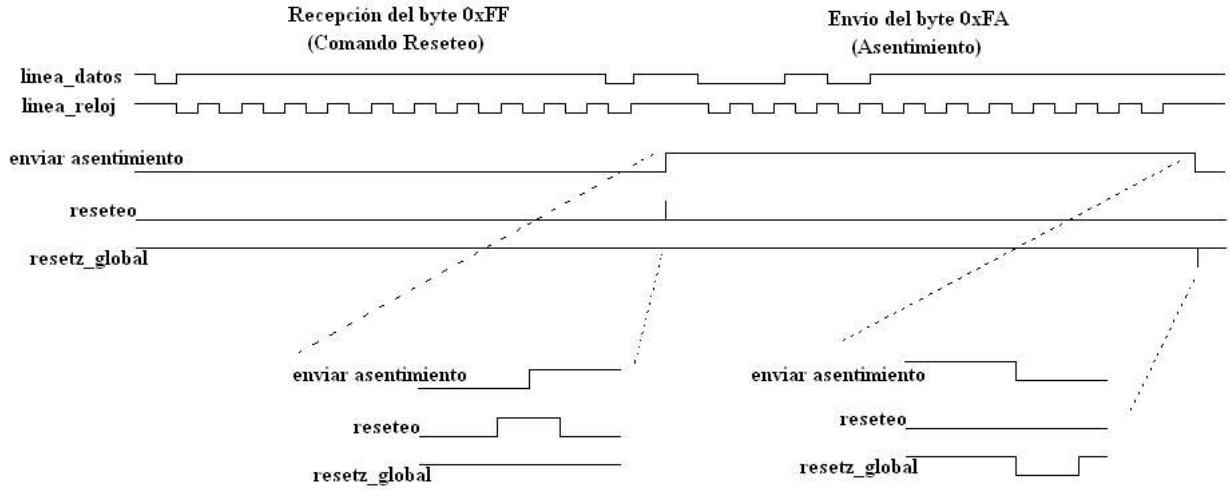
La recepción del comando reset provoca que el dispositivo PS/2 entre en el modo de funcionamiento "modo reset". El ratón entra en el "modo reset" en la activación inicial o al recibir el comando "reset" (ØxFF). Después de entrar en este modo de operación, el ratón activa los siguientes valores por defecto:

- Frecuencia de muestreo = 100 muestras/segundo
- Resolución = 4 cuentas/mm
- Escala = 1:1
- Envío de datos deshabilitado

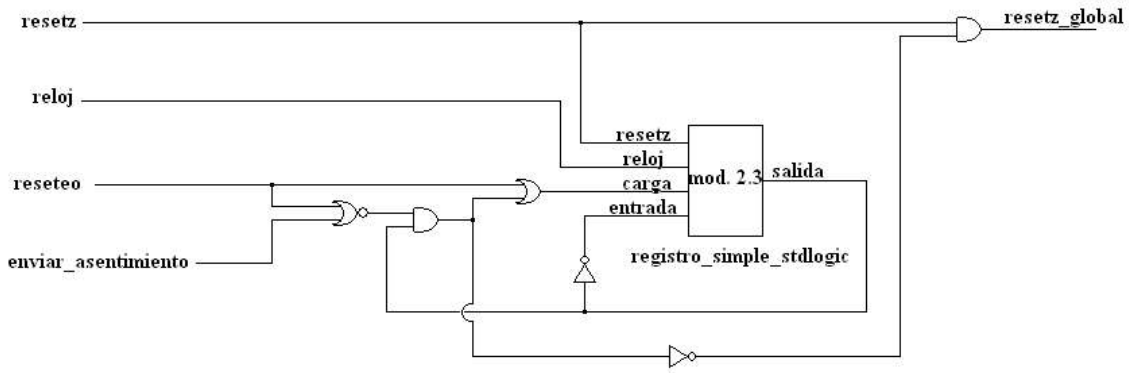
Luego enviará el código de terminación de BAT con éxito (ØxAA). Después del código de terminación de la prueba BAT (ØxAA), el ratón envía el identificador de dispositivo ID ØxØØ. Este hecho lo distingue de un teclado o de un ratón en modo extendido. El host no espera la transmisión de ningún dato hasta recibir el identificador de dispositivo ID. Sin embargo, algunas BIOS enviarán el comando "reset" (ØxFF) inmediatamente después de recibir ØxAA tras un reinicio. Después de que el ratón haya enviado su identificador de dispositivo ID al host, entrará en el "modo corriente".

Al recibir del ordenador el comando reset se activa la señal "reseteo" y "enviar_asentimiento" con un valor '1' lógico. La señal "reseteo" activa el funcionamiento de este módulo, pero no generará la señal de reseteo válida para el resto de los módulos del ratón hasta que el asentimiento haya sido enviado. Una vez haya sido enviado el asentimiento la señal "enviar_asentimiento" se desactivará, desencadenando la activación de la señal "resetz_global". Esta señal es activa a nivel bajo, y generará el reseteo de todos los módulos del ratón, reiniciándose así el funcionamiento del dispositivo.

Para incluir el reseteo general causado en el conjunto del diseño por la señal "resetz", la señal "resetz_global" debe anularse también cuando se anule la señal "resetz".



La funcionalidad del dispositivo se consigue de la siguiente forma:
La recepción de la señal reseteo con un valor '1' lógico guarda en un registro un flag con valor '1' lógico. Este valor '1' lógico se mantendrá en el registro mientras la señal "enviar_asentimiento" esté activa con un valor '1' lógico, lo cual implica que todavía no se ha enviado el byte de asentimiento hacia el ordenador. El valor del registro cambiará a un valor '0' lógico cuando se desactive la señal "enviar_asentimiento". La señal de reseteo "resetz_global" se activará cuando se haya anulado la señal "enviar_asentimiento" mientras esté activo el registro con un valor '1' lógico. La duración del pulso de reseteo se consigue con la diferencia en la sincronía de la señal "enviar_asentimiento", que cambia en el flanco de bajada del reloj, con la carga en el flanco de subida del reloj del registro. La señal "resetz_global" también tomará un valor lógico '0' cuando la señal "resetz" también lo tome.



2.1	BiestableD.vhd

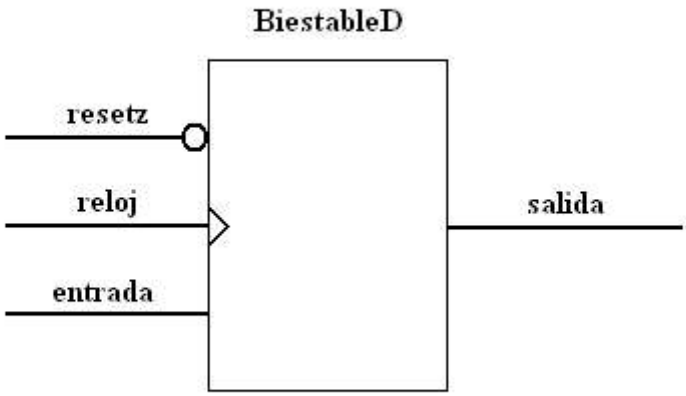
```
ENTITY BiestableD IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    entrada: IN STD_LOGIC;
    salida: OUT STD_LOGIC
  );
END BiestableD;
```

Este módulo implementa a un biestable D que carga el valor de entrada en cada flanco de subida de la señal de reloj.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el biestable. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se guardará en el biestable el valor de la señal de entrada.
- ENTRADA: señal que se guardará en el biestable en el flanco de subida de la señal de reloj.
- SALIDA: señal que muestra el valor guardado en el biestable.



2.2 BiestableD_neg.vhd

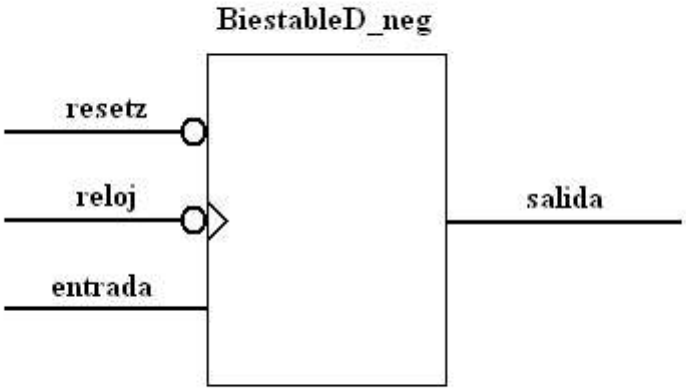
```
ENTITY BiestableD_neg IS
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    entrada: IN STD_LOGIC;
    salida: OUT STD_LOGIC
  );
END BiestableD_neg;
```

Este módulo implementa a un biestable D que carga el valor de entrada en cada flanco de bajada de la señal de reloj.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el biestable. Cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de bajada de la señal de reloj se guardará en el biestable el valor de la señal de entrada.
- ENTRADA: señal que se guardará en el biestable en el flanco de bajada de la señal de reloj.
- SALIDA: señal que muestra el valor guardado en el biestable.



2.3 registro_simple_stdlogic.vhd

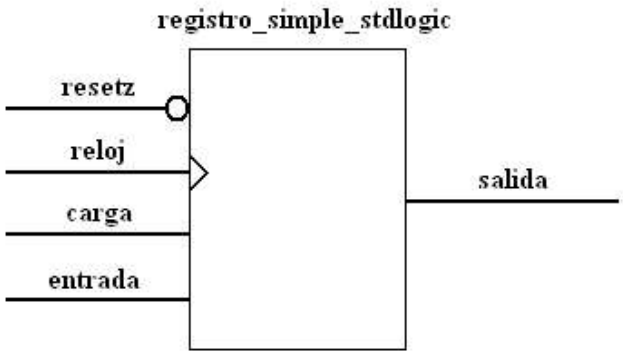
```
ENTITY registro_simple_stdlogic IS
  PORT(
    resetz: IN STD_LOGIC;
    clk: IN STD_LOGIC;
    carga: IN STD_LOGIC;
    entrada: IN STD_LOGIC;
    salida: OUT STD_LOGIC
  );
END registro_simple_stdlogic;
```

Este módulo implementa a un registro que carga un bit con el valor de la señal de entrada en cada flanco de subida de la señal de reloj, siempre que esté activada la señal de carga con un valor '1' lógico.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el registro de bit y cuando esta a '1' lógico permite el normal funcionamiento del dispositivo.
- CLK: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se guardará en el registro el valor de la señal de entrada siempre que esté activada la señal de carga con un valor '1' lógico.
- CARGA: señal que activa la carga del registro cuando está con un valor '1' lógico.
- ENTRADA: bit que se guardará en el registro en el flanco de subida de la señal de reloj siempre que esté activada la señal de carga con un valor '1' lógico.
- SALIDA: bit que muestra el valor guardado en el registro.



2.4 registro_simple_stdlogic_neg.vhd

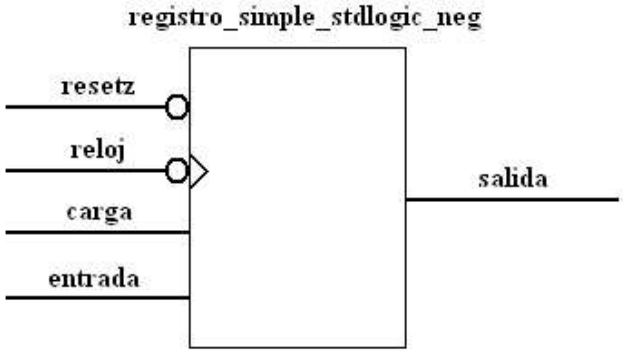
```
ENTITY registro_simple_stdlogic_neg IS
  PORT(
    resetz: IN STD_LOGIC;
    clk: IN STD_LOGIC;
    carga: IN STD_LOGIC;
    entrada: IN STD_LOGIC;
    salida: OUT STD_LOGIC
  );
END registro_simple_stdlogic_neg;
```

Este módulo implementa a un registro que carga un bit con el valor de la señal de entrada en cada flanco de bajada de la señal de reloj, siempre que esté activada la señal de carga con un valor '1' lógico.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el registro de bit y cuando esta a '1' lógico permite el normal funcionamiento del dispositivo.
- CLK: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de bajada de la señal de reloj se guardará en el registro el valor de la señal de entrada siempre que esté activada la señal de carga con un valor '1' lógico.
- CARGA: señal que activa la carga del registro cuando está con un valor '1' lógico.
- ENTRADA: bit que se guardará en el registro en el flanco de bajada de la señal de reloj siempre que esté activada la señal de carga con un valor '1' lógico.
- SALIDA: bit que muestra el valor guardado en el registro.



2.5 modulo_incrementa.vhd

```
ENTITY modulo_incrementa IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    entrada: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END modulo_incrementa;
```

Este módulo implementa la lógica necesaria para realizar el sumador de una unidad a un vector de bits que representa a un número natural binario.

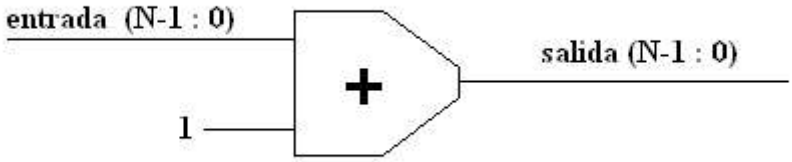
El número de bits del vector de bits es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.
El interfaz del módulo es el siguiente:

- ENTRADA: vector de bits de entrada que representa a un número natural binario de N bits, siendo N el parámetro genérico que especifica el número de bits del vector.
- SALIDA: vector de bits de salida que representa al número natural en binario resultado de sumarle una unidad al número representado por el vector de bits de entrada.

Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:

Si entrada="000" (Valor 0) entonces salida="001" (Valor 1)
Si entrada="001" (Valor 1) entonces salida="010" (Valor 2)
Si entrada="010" (Valor 2) entonces salida="011" (Valor 3)
Si entrada="011" (Valor 3) entonces salida="100" (Valor 4)
Si entrada="100" (Valor 4) entonces salida="101" (Valor 5)
Si entrada="101" (Valor 5) entonces salida="110" (Valor 6)
Si entrada="110" (Valor 6) entonces salida="111" (Valor 7)
Si entrada="111" (Valor 7) entonces salida="000" (Valor 0)

Si a la entrada tenemos un vector de bits que representa al mayor número natural en binario que se puede representar con N bits (todos los bits a '1' lógico), a la salida tendremos el valor natural 0, representado en binario por un vector binario con todos los bits a '0' lógico.
Este módulo es útil en la realización de un contador que incremente su valor cada vez que sea activado con la señal apropiada. Este módulo implementaría la lógica necesaria para realizar el sumador de una unidad asociado al contador.



2.6 modulo_decrementa.vhd

```
ENTITY modulo_decrementa IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    entrada: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END modulo_decrementa;
```

Este módulo implementa la lógica necesaria para decrementar en una unidad a un vector de bits que representa a un número natural binario.

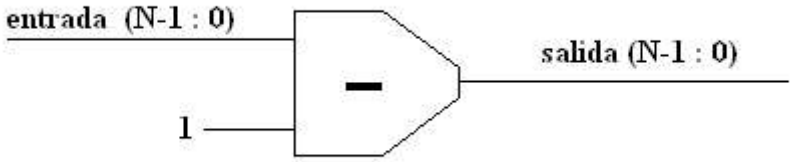
El número de bits del vector de bits es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.
El interfaz del módulo es el siguiente:

- ENTRADA: vector de bits de entrada que representa a un número natural binario de N bits, siendo N el parámetro genérico que especifica el número de bits del vector.
- SALIDA: vector de bits de salida que representa al número natural en binario resultado de restarle una unidad al número representado por el vector de bits de entrada.

Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:

Si entrada="000" (Valor 0) entonces salida="111" (Valor 1)
Si entrada="001" (Valor 1) entonces salida="000" (Valor 2)
Si entrada="010" (Valor 2) entonces salida="001" (Valor 3)
Si entrada="011" (Valor 3) entonces salida="010" (Valor 4)
Si entrada="100" (Valor 4) entonces salida="011" (Valor 5)
Si entrada="101" (Valor 5) entonces salida="100" (Valor 6)
Si entrada="110" (Valor 6) entonces salida="101" (Valor 7)
Si entrada="111" (Valor 7) entonces salida="110" (Valor 0)

Si a la entrada tenemos un vector de bits que representa al menor número natural en binario que se puede representar con N bits (todos los bits a '0' lógico), a la salida tendremos el valor natural máximo que se puede representar con N bits, representado en binario por un vector binario con todos los bits a '1' lógico.
Este módulo es útil en la realización de un contador que decremente su valor cada vez que sea activado con la señal apropiada. Este módulo implementaría la lógica necesaria para realizar el decremento en una unidad asociado al contador decreciente.



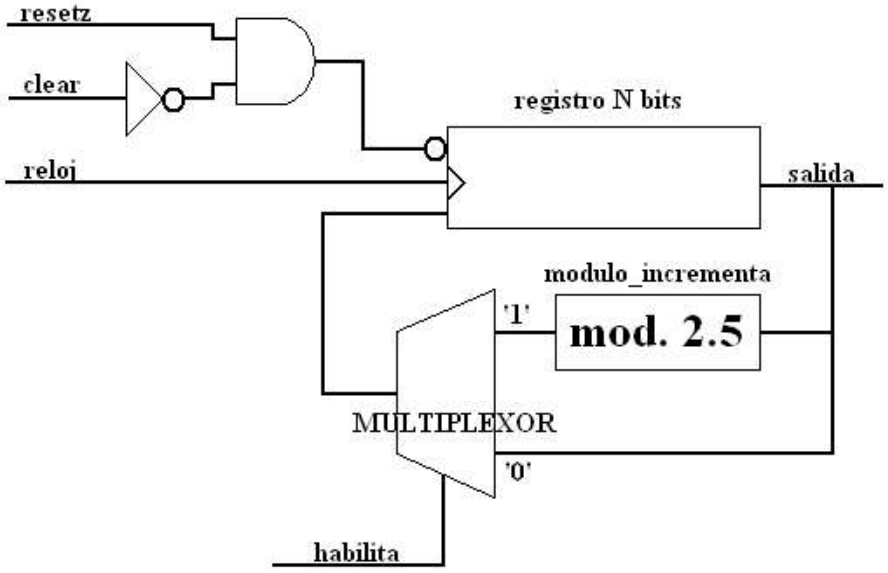
2.7 contador.vhd

```
ENTITY contador IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    habilita: IN STD_LOGIC;
    clear: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END contador;
```

Este módulo implementa un contador binario creciente que se activa cuando la señal "habilita" está activa en el flanco de subida de la señal de reloj.

Al habilitar la señal "clear" reinicializaremos la cuenta al valor natural 0, representado en binario por un vector con todos los bits a '0' lógico. El número de bits del vector de salida que indica el valor alcanzado en la cuenta es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica. El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea el valor del contador e impedimos que se incremente, cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se incrementará el valor del contador en una unidad siempre que esté activa la señal "habilita" con el valor lógico '1'.
- HABILITA: señal que activa el incremento síncrono del contador cuando esta a nivel lógico '1'.
- CLEAR: señal que activa el reseteo asíncrono del contador cuando esta a nivel lógico '1'.
- SALIDA: vector de bits de salida que representa al número natural en binario que indica el valor alcanzado por el contador.



Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:
El valor inicial del contador es "000", también es el valor que toma el contador tras un reset asíncrono a través de la señal "clear". El contador seguiría la siguiente secuencia de valores con la habilitación síncrona del mismo:

del valor "000" pasaríamos al valor "001"
del valor "001" pasaríamos al valor "010"
del valor "010" pasaríamos al valor "011"
del valor "011" pasaríamos al valor "100"
del valor "100" pasaríamos al valor "101"
del valor "101" pasaríamos al valor "110"
del valor "110" pasaríamos al valor "111"
del valor "111" pasaríamos al valor "000" y vuelta al principio

Si el valor del contador es el número natural binario máximo que se puede alcanzar con N bits (todos los N bits con valor lógico '1'), siendo N el parámetro genérico especificado en la instalación del componente, tomará el valor numérico 0 tras el incremento del contador con la habilitación síncrona del mismo (señal "habilita" a '1' lógico en el flanco de subida del reloj).
El valor 0 numérico se corresponde con un vector de bits con todos sus valores a '0' lógico.

2.8	contador_neg.vhd

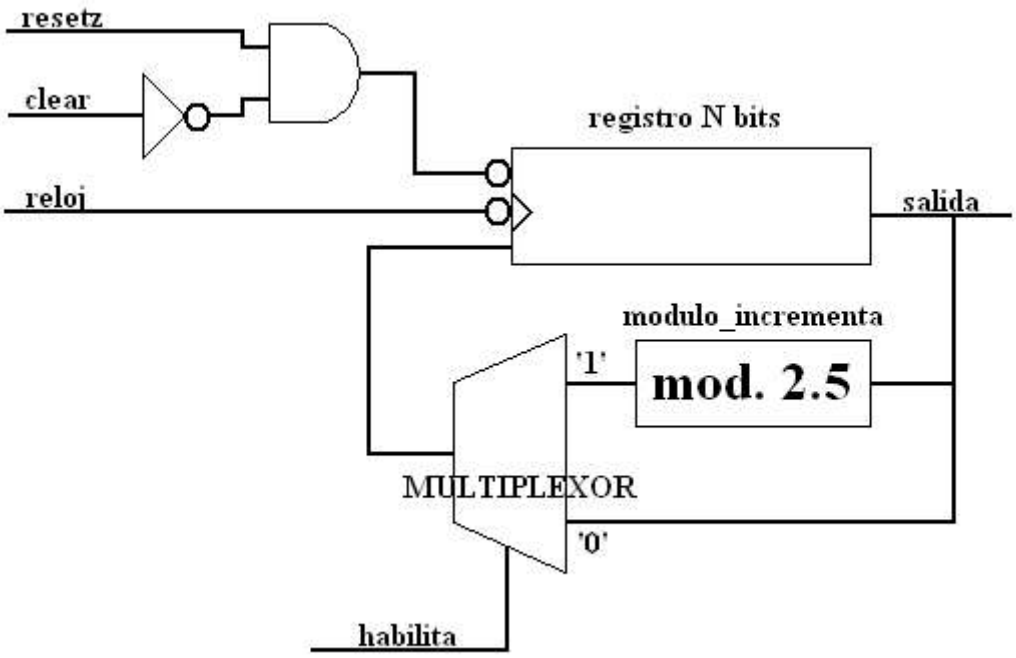
```
ENTITY contador_neg IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    habilita: IN STD_LOGIC;
    clear: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END contador_neg;
```

Este módulo implementa un contador binario creciente que se activa cuando la señal "habilita" está activa en el flanco de bajada de la señal de reloj.

Al habilitar la señal "clear" reinicializaremos la cuenta al valor natural 0, representado en binario por un vector con todos los bits a '0' lógico. El número de bits del vector de salida que indica el valor alcanzado en la cuenta es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.
El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea el valor del contador e impedimos que se incremente, cuando está a '1' lógico permite el normal funcionamiento del dispositivo.

- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de bajada de la señal de reloj se incrementará el valor del contador en una unidad siempre que esté activa la señal "habilita" con el valor lógico '1'.
- HABILITA: señal que activa el incremento síncrono del contador cuando esta a nivel lógico '1'.
- CLEAR: señal que activa el reseteo asíncrono del contador cuando esta a nivel lógico '1'.
- SALIDA: vector de bits de salida que representa al número natural en binario que indica el valor alcanzado por el contador.



Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:
El valor inicial del contador es "000", también es el valor que toma el contador tras un reset asíncrono a través de la señal "clear". El contador seguiría la siguiente secuencia de valores con la habilitación síncrona del mismo:

del valor "000" pasaríamos al valor "001"
del valor "001" pasaríamos al valor "010"
del valor "010" pasaríamos al valor "011"
del valor "011" pasaríamos al valor "100"
del valor "100" pasaríamos al valor "101"
del valor "101" pasaríamos al valor "110"
del valor "110" pasaríamos al valor "111"
del valor "111" pasaríamos al valor "000" y vuelta al principio

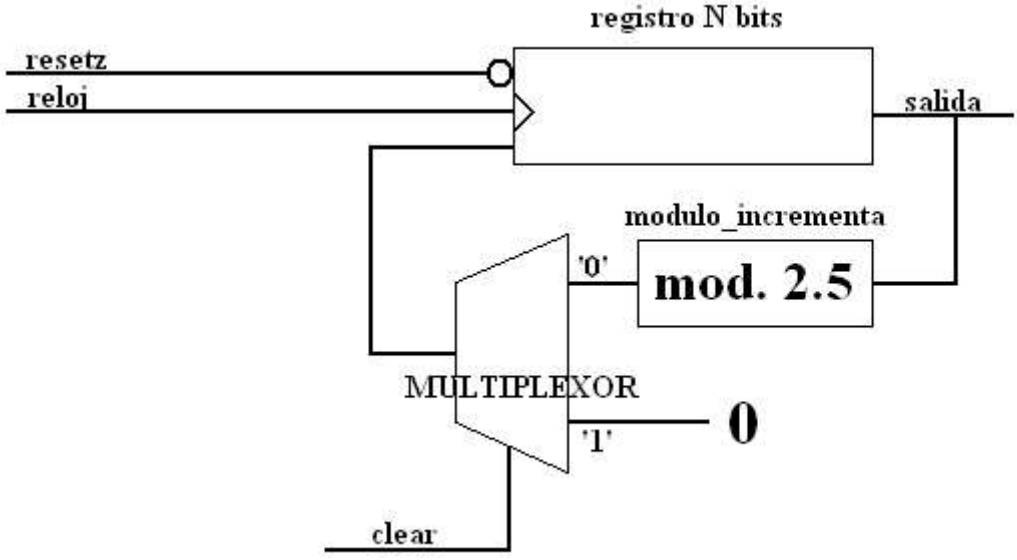
Si el valor del contador es el número natural binario máximo que se puede alcanzar con N bits (todos los N bits con valor lógico '1'), siendo N el parámetro genérico especificado en la instalación del componente, tomará el valor numérico 0 tras el incremento del contador con la habilitación síncrona del mismo (señal "habilita" a '1' lógico en el flanco de bajada del reloj).
El valor 0 numérico se corresponde con un vector de bits con todos sus valores a '0' lógico.

2.9 contador_reset_sincrono.vhd

```
ENTITY contador_reset_sincrono IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    clear: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END contador_reset_sincrono;
```

Este módulo implementa un contador binario creciente que se activa en cada flanco de subida de la señal de reloj.

El valor de salida del contador será un número binario que representa al número natural que indica el valor alcanzado por el contador. Al habilitar la señal "clear" reinicializaremos la cuenta al valor natural 0, representado en binario por un vector con todos los bits a '0' lógico, pero esta inicialización sólo será efectiva en el flanco de subida de la señal de reloj. La señal "clear" es un reset síncrono.
El número de bits del vector de salida que indica el valor alcanzado en la cuenta es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.



El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico resetea el valor del contador e impedimos que se incremente, cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se incrementará el valor del contador en una unidad.

- CLEAR: señal que activa el reseteo asíncrono del contador cuando está a nivel lógico '1'.
- SALIDA: vector de bits de salida que representa al número natural en binario que indica el valor alcanzado por el contador.

Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:
El valor inicial del contador es "000", también es el valor que toma el contador tras un reset síncrono a través de la señal "clear". El contador seguiría la siguiente secuencia de valores en cada flanco de subida de la señal de reloj:

del valor "000" pasaríamos al valor "001"
del valor "001" pasaríamos al valor "010"
del valor "010" pasaríamos al valor "011"
del valor "011" pasaríamos al valor "100"
del valor "100" pasaríamos al valor "101"
del valor "101" pasaríamos al valor "110"
del valor "110" pasaríamos al valor "111"
del valor "111" pasaríamos al valor "000" y vuelta al principio

Si el valor del contador es el número natural binario máximo que se puede alcanzar con N bits (todos los N bits a '1' lógico), siendo N el parámetro genérico especificado en la instalación del componente, tomaremos el valor numérico 0 tras el incremento del contador en el flanco de subida del reloj. El valor 0 numérico se corresponde con un vector de bits con todos sus valores a '0' lógico.

2.10	contador_siempre.vhd

```
ENTITY contador_siempre IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END contador_siempre;
```

Este módulo implementa un contador binario creciente que se incrementa en el flanco de subida del reloj.

El valor de salida del contador sera un número binario que representa al número natural que indica el valor alcanzado por el contador. El número de bits del vector de salida que indica el valor alcanzado en la cuenta es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.

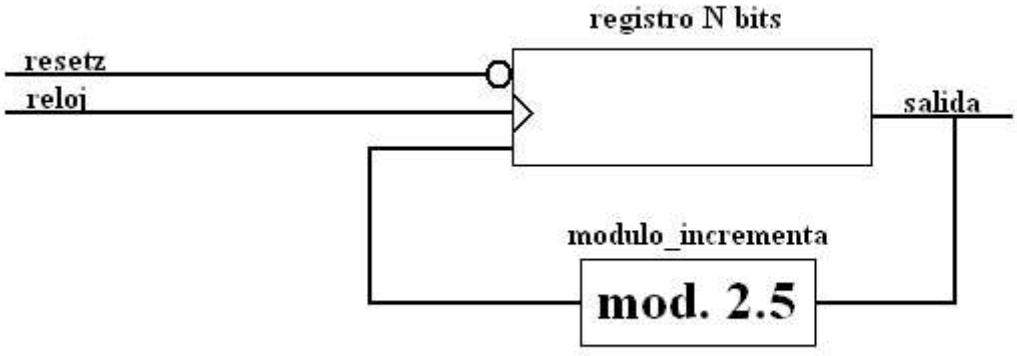
El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico reseteamos el valor del contador e impedimos que se incremente, cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se incrementará el valor del contador en una unidad.
- SALIDA: vector de bits de salida que representa al número natural en binario que indica el valor alcanzado por el contador.

Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:
El valor inicial del contador es "000". El contador seguiría la siguiente secuencia de valores en los flancos de subida del reloj:

del valor "000" pasaríamos al valor "001"
del valor "001" pasaríamos al valor "010"
del valor "010" pasaríamos al valor "011"
del valor "011" pasaríamos al valor "100"
del valor "100" pasaríamos al valor "101"
del valor "101" pasaríamos al valor "110"
del valor "110" pasaríamos al valor "111"
del valor "111" pasaríamos al valor "000" y vuelta al principio

Si el valor del contador es el número natural binario máximo que se puede alcanzar con N bits (todos los N bits a '1' lógico), siendo N el parámetro genérico especificado en la instalación del componente, tomaremos el valor numérico 0 tras el incremento del contador en el flanco de subida del reloj. El valor 0 numérico se corresponde con un vector de bits con todos sus valores a '0' lógico.

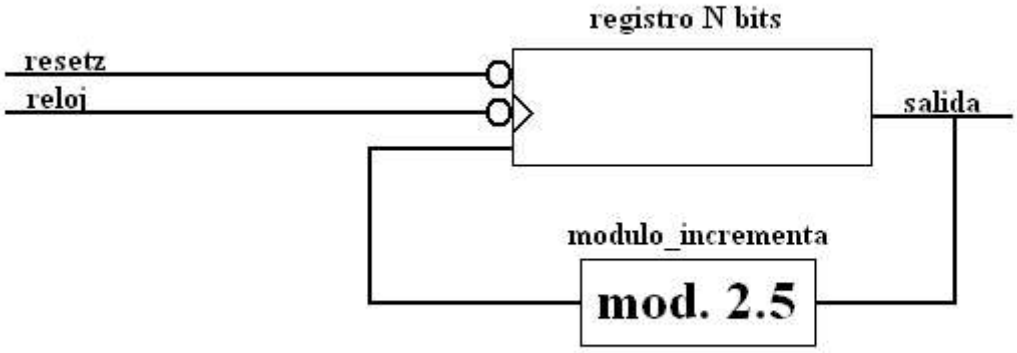


2.11 contador_siempre_neg.vhd

```
ENTITY contador_siempre_neg IS
  GENERIC( N: INTEGER :=8 );
  PORT(
    resetz: IN STD_LOGIC;
    reloj: IN STD_LOGIC;
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END contador_siempre_neg;
```

Este módulo implementa un contador binario creciente que se incrementa en el flanco de bajada del reloj.

El valor de salida del contador sera un número binario que representa al número natural que indica el valor alcanzado por el contador. El número de bits del vector de salida que indica el valor alcanzado en la cuenta es especificado con el parámetro genérico N una vez se utilice este módulo dentro de una implementación específica.



El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico reseteamos el valor del contador e impedimos que se incremente, cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- RELOJ: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de bajada de la señal de reloj se incrementará el valor del contador en una unidad.
- SALIDA: vector de bits de salida que representa al número natural en binario que indica el valor alcanzado por el contador.

Ejemplo: Si el parámetro genérico fuera N=3 el módulo realizaría la siguiente funcionalidad:
El valor inicial del contador es "000". El contador seguiría la siguiente secuencia de valores en los flancos de subida del reloj:

del valor "000" pasaríamos al valor "001"
del valor "001" pasaríamos al valor "010"
del valor "010" pasaríamos al valor "011"
del valor "011" pasaríamos al valor "100"

del valor "100" pasaríamos al valor "101"
del valor "101" pasaríamos al valor "110"
del valor "110" pasaríamos al valor "111"
del valor "111" pasaríamos al valor "000" y vuelta al principio

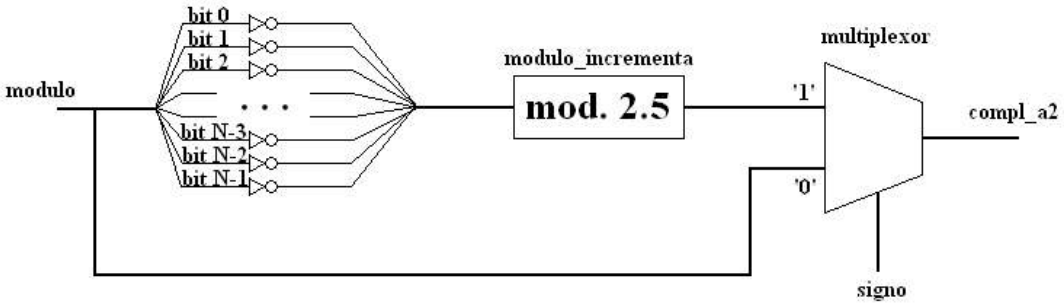
Si el valor del contador es el número natural binario máximo que se puede alcanzar con N bits (todos los N bits a '1' lógico), siendo N el parámetro genérico especificado en la instalación del componente, tomaremos el valor numérico 0 tras el incremento del contador en el flanco de bajada del reloj. El valor 0 numérico se corresponde con un vector de bits con todos sus valores a '0' lógico.

2.12 complemento_a_2.vhd

```
ENTITY Complemento_A_2 IS
  GENERIC( N: INTEGER :=8);
  PORT(
    signo: IN STD_LOGIC;
    modulo: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    compl_a2: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END Complemento_A_2;
```

Este módulo implementa la lógica necesaria para realizar el complemento a 2 de un número binario.

A la entrada tengo el signo y el módulo del número binario. A la salida tengo el complemento a 2 del número anterior teniendo en cuenta su signo y su módulo. El complemento a 2 es una representación binaria muy útil para realizar operaciones.



El proceso es el siguiente:
Teniendo un número natural que constituye el módulo representado por un vector de N bits, siendo N el parámetro genérico cuyo valor lo especificamos en la instalación de la entidad. El complemento a dos del número tendrá la misma representación binaria si el signo es positivo ("signo" es '0' lógico). Si el signo es negativo ("signo" es '1' lógico), la representación binaria del número en complemento a dos será el resultado de cambiar la lógica en cada uno de los bits ('0' -> '1' y '1' ->'0') y luego incrementar al número binario una unidad.
El interfaz del módulo es el siguiente:

- SIGNO: signo del número al cual le vamos a calcular el complemento a dos. Es '1' lógico si el signo es negativo y es '0' lógico si el signo es positivo.
- MODULO: vector de N bits que representa al módulo del número binario al cual se le va a calcular el complemento a dos. N es el parámetro genérico cuyo valor lo especificamos en la instalación del mismo módulo.
- COMPL_A2: complemento a dos de N bits del número dado en módulo binario y signo.

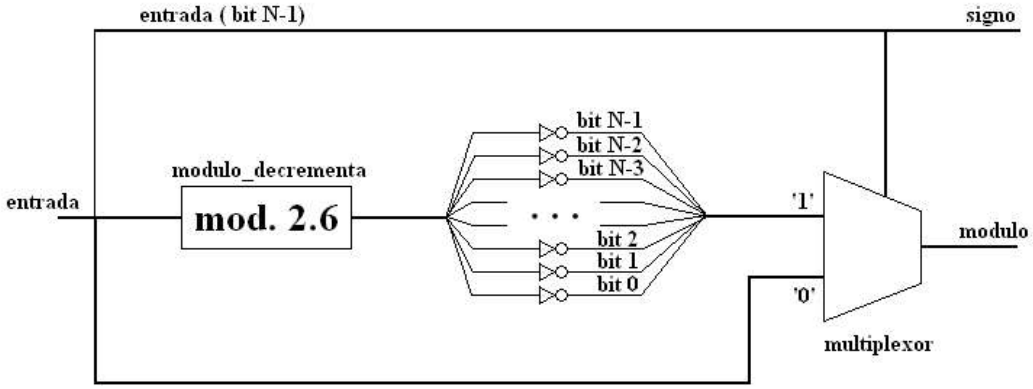
2.13	inv_complemento_a_2.vhd

```
ENTITY inv_complemento_a_2 IS
  GENERIC( N: INTEGER :=8);
  PORT (
    entrada: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    signo: OUT STD_LOGIC;
    modulo: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END inv_complemento_a_2;
```

Este módulo implementa la lógica necesaria para realizar la inversa del complemento a 2 de un número binario.

A la entrada tendré la representación en binario en complemento a dos de un número entero. A la salida debo de tener el signo y la representación binaria del módulo del número anterior. El complemento a 2 es una representación binaria muy útil para realizar operaciones.
El cálculo de la inversa del complemento a dos de un número se realiza de la siguiente forma:
Teniendo un número entero representado en complemento a dos por un vector de N bits, siendo N el parámetro genérico cuyo valor lo especificamos en la instalación de la entidad, el signo del número será el valor del bit más significativo del número binario en complemento a dos. Si este bit tiene valor lógico '1', el signo es negativo y si este bit tiene valor lógico '0', el signo es positivo.

Si el signo es positivo, el módulo del número binario en complemento a dos tendrá la misma representación binaria. Si el signo es negativo, para calcular el módulo del número binario en complemento a dos tengo que seguir un proceso de dos pasos:
Primero: Disminuir en una unidad el valor binario correspondiente al valor representado por el complemento a dos del número.
Segundo: Invertir la lógica de cada uno de los bits del resultado anterior ('0' -> '1', '1' -> '0').



El interfaz del módulo es el siguiente:

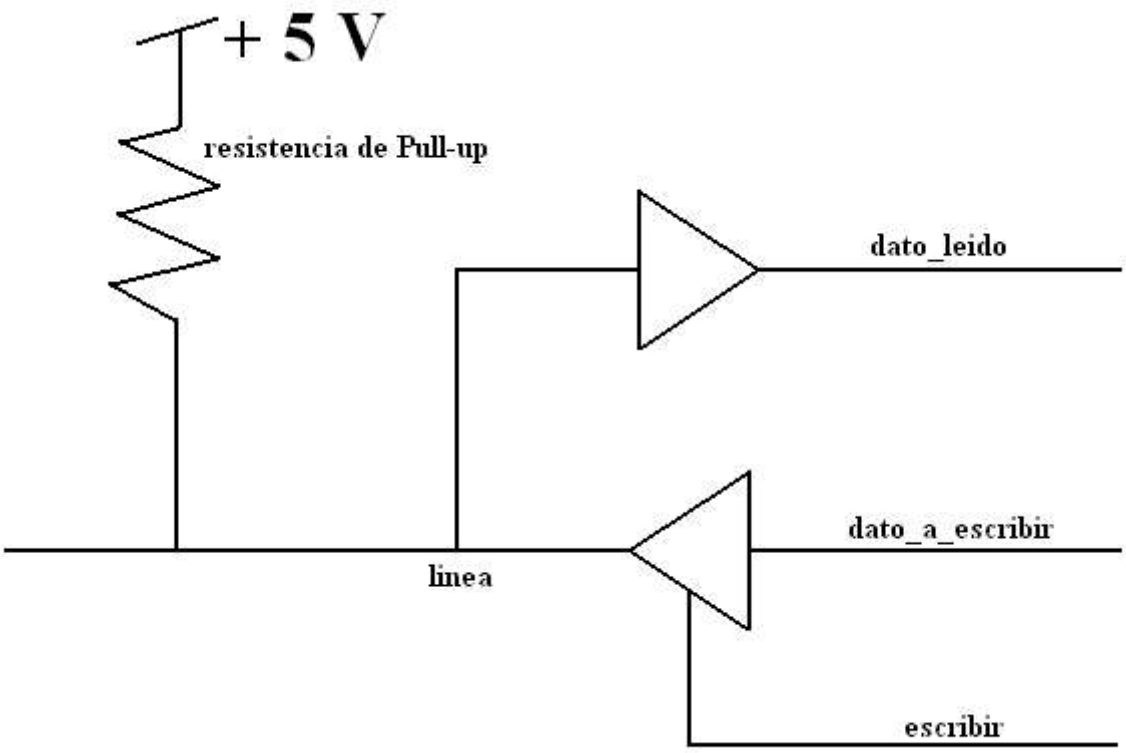
- ENTRADA: representación binaria de N bits de un número en complemento a dos, del cual calcularé su signo y su módulo. N es el parámetro genérico cuyo valor lo especificaré en la instalación del mismo módulo.
- SIGNO: signo del número binario en complemento a dos. Es '1' lógico si el signo es negativo y es '0' lógico si el signo es positivo.
- MODULO: vector de N bits que representa al módulo del número binario en complemento a dos.

2.14	mibuffer.vhd

```
ENTITY mibuffer IS
  PORT(
    escribir: IN STD_LOGIC;
    dato_leido: OUT STD_LOGIC;
    dato_a_escribir: IN STD_LOGIC;
    linea: INOUT STD_LOGIC
  );
END mibuffer;
```

Buffer utilizado para implementar el protocolo PS/2. Mediante este buffer puedo leer y escribir en las líneas de datos y del reloj del protocolo serie PS/2.

Un cero en la línea se interpreta como un cero lógico. Un valor de alta impedancia se interpreta como un uno lógico. Puedo leer los datos de la línea a través de la señal "dato_leido". Sólo escribiré en la línea al tener la señal "escribir" a uno lógico y la señal "dato_a_escribir" a cero lógico.
De esta manera puedo modelar el interfaz eléctrico de comunicación del ratón con el ordenador. Esta línea está a colector abierto, y a través de una resistencia del pull-up en el ordenador la tensión permanece a 5 voltios. Para transmitir datos lo que hago es forzar en la línea un cero lógico con 0 voltios y llevar la línea a un uno lógico poniéndola en alta impedancia, la resistencia de pull-up en la tarjeta XSA pondrá la línea a 5 voltios.



El interfaz del módulo es el siguiente:
- ESCRIBIR: señal de habilitación de escritura en la línea de salida del protocolo PS/2 (tanto para la línea del ratón como para la línea de datos. Está activa a nivel lógico '1'.
- DATO_LEIDO: dato leído de la línea externa del protocolo PS/2.
- DATO_A_ESCRIBIR: dato que se escribirá en la línea externa del protocolo PS/2.
- LINEA: línea externa del protocolo PS/2.

2.15 memoria_ram.vhd

```
ENTITY memoria_ram IS
  PORT(
    DI: IN STD_LOGIC_VECTOR (15 downto 0);
    EN: IN STD_ULOGIC;
    WE: IN STD_ULOGIC;
    resetz: IN STD_ULOGIC;
    CLK: IN STD_ULOGIC;
    ADDR: IN STD_LOGIC_VECTOR (7 downto 0);
    DO: OUT STD_LOGIC_VECTOR (15 downto 0)
  );
END memoria_ram;
```

Bloque de memoria ram de 4096 bits que se distribuyen en 256 palabras de 2 bytes cada una.

El bloque de memoria RAM tiene un bus de direcciones de 8 bits y un bus de datos de 16 bits, conformando en conjunto la capacidad total de la memoria. La FPGA Spartan II 2S50TQ144 dispone de 8 bloques de memoria RAM de 4096 bits que podrán utilizarse para la implementación de este módulo. Cada módulo se implementaría con un bloque de memoria RAM de la FPGA.

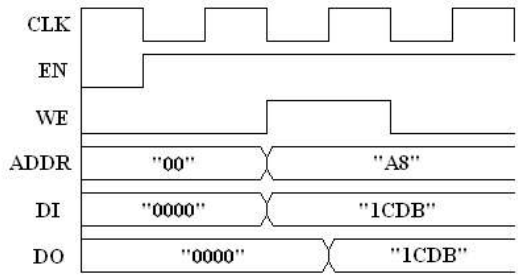
El interfaz del módulo es el siguiente:

- DI: bus de datos de entrada que mantiene la información que se guardará cuando estén activas la señal “EN” (activa con un valor '1' lógico) y la señal “WE” (activa con un valor '1' lógico). Este dato se guardará en la dirección fijada.
- EN: señal de habilitación de la memoria activa cuando está con un valor '1' lógico. Si está habilitada la memoria funcionará normalmente, escribiendo un dato en la dirección dada cuando la señal “WE” esté activa con un valor '1' lógico o leyendo un dato de la dirección dada cuando la señal “WE” esté desactivada con un valor '0' lógico. Si la memoria está deshabilitada con la señal “EN” a '0' lógico no se provocará ninguna actuación en la memoria.
- WE: señal de habilitación de escritura. Esta señal habilita la escritura en la dirección proporcionada por el bus de direcciones “ADDR” de la memoria RAM de los datos entregados en el bus “DI” cuando está activa con un nivel lógico '1'. Además, la memoria debe estar habilitada con la señal “EN” a '1' lógico. Si la señal “WE” no está habilitada, entonces realizamos una lectura de los datos presentes en la dirección de memoria dada por el bus de direcciones “ADDR”, estando habilitada la señal “EN”.

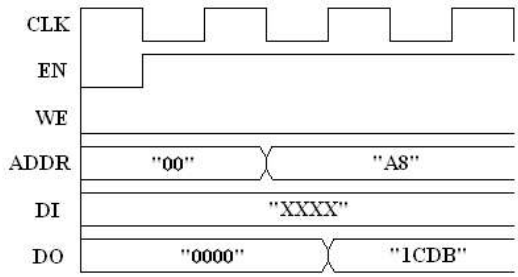
- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico reseteamos la memoria ram a sus valores iniciales (todos los bits a cero), cuando está a '1' lógico permite el normal funcionamiento del dispositivo.
- CLK: señal de reloj necesaria para el funcionamiento síncrono del dispositivo. Los datos serán guardados en el flanco de subida del reloj. La lectura de una dirección de memoria estará disponible en el flanco de subida del reloj. Para eso, todas las señales deben ser estables cuando ocurra un flanco de subida del reloj. Las señales de entrada sólo se modificarán en los flancos de bajada del reloj, asegurando así su estabilidad en los flancos de subida del reloj.
- ADDR: bus de direcciones de la memoria de 8 bits (256 direcciones). Con este bus accedo a las 256 palabras de 2 bytes de la memoria tanto para la escritura como para la lectura.
- DO: dato leído de la dirección de memoria proporcionada por la señal "ADDR".

El proceso de **escritura** en la memoria RAM es el siguiente:
La señal de habilitación "EN" de la memoria debe estar activa con un valor uno lógico. En un flanco de bajada de la señal de reloj introducimos en el bus "ADDR" la dirección de memoria en la que escribiremos el dato. También introducimos en el flanco de bajada de la señal de reloj el dato a escribir a través del bus de datos "DI". En ese mismo flanco de bajada de la señal de reloj habilitaremos la escritura activando la señal "WE" con un valor uno lógico. En el siguiente flanco de subida de la señal de reloj se realizará la escritura propiamente dicha. Desactivaremos la señal de habilitación de escritura "WE" en el siguiente flanco de bajada de la señal de reloj. El resto de las señales deben permanecer estables hasta por lo menos ese mismo flanco de bajada de la señal de reloj.

El proceso de **lectura** de la memoria RAM es el siguiente:
La señal de habilitación "EN" de la memoria debe estar activa con un valor uno lógico. En un flanco de bajada de la señal de reloj introducimos en el bus "ADDR" la dirección de memoria de la que leeremos el dato. En ese mismo flanco de bajada de la señal de reloj la señal "WE" debe estar deshabilitada con un valor uno lógico. El dato presente en la dirección de memoria requerida será mostrado a través del bus de datos de salida "DO" en el siguiente flanco de subida de la señal de reloj. Podremos tomar el dato en el próximo flanco de bajada de la señal de reloj, por lo que debemos dejar estables las señales de entrada hasta el momento de la adquisición del dato.



LECTURA



ESCRITURA

2.16 registro_simple_stdlogicvector.vhd

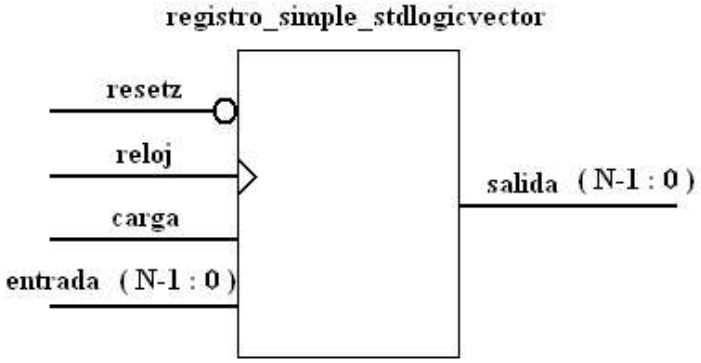
```
ENTITY registro_simple_stdlogicvector IS
  GENERIC( N: INTEGER :=5);
  PORT(
    resetz: IN std_logic;
    clk: IN std_logic;
    carga: IN std_logic;
    entrada: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END registro_simple_stdlogicvector;
```

Este módulo implementa a un registro que carga un vector de bits con el valor del vector de entrada en cada flanco de subida de la señal de reloj, siempre que esté activada la señal de carga con un valor '1' lógico. El número de bits del registro lo indica el parámetro genérico N.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el registro de bit y cuando esta a '1' lógico permite el normal funcionamiento del dispositivo.
- CLK: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de subida de la señal de reloj se guardará en el registro el valor de la señal de entrada siempre que esté activada la señal de carga con un valor '1' lógico.
- CARGA: señal que activa la carga del registro cuando está con un valor '1' lógico.
- ENTRADA: vector de N bits que se guardará en el registro en el flanco de subida de la señal de reloj siempre que esté activada la señal de carga con un valor '1' lógico.
- SALIDA: vector de N bits que muestra el valor guardado en el registro.



2.17 registro_simple_stdlogicvector_neg.vhd

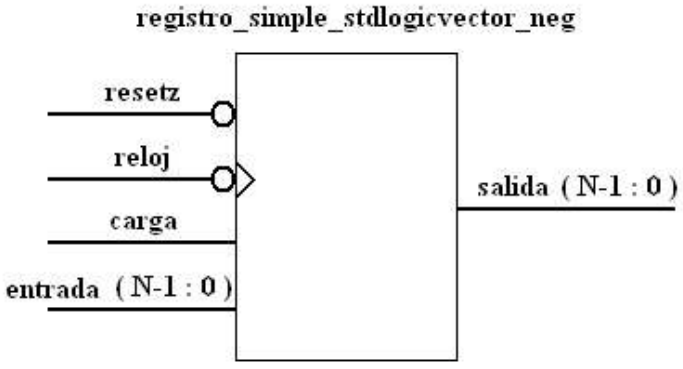
```
ENTITY registro_simple_stdlogicvector_neg IS
  GENERIC( N: INTEGER :=5);
  PORT(
    resetz: IN std_logic;
    clk: IN std_logic;
    carga: IN std_logic;
    entrada: IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
    salida: OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
  );
END registro_simple_stdlogicvector_neg;
```

Este módulo implementa a un registro que carga un vector de bits con el valor del vector de entrada en cada flanco de bajada de la señal de reloj, siempre que esté activada la señal de carga con un valor '1' lógico. El número de bits del registro lo indica el parámetro genérico N.

El valor guardado es mostrado en la señal de salida.

El interfaz del módulo es el siguiente:

- RESETZ: señal de reseteo activa a nivel bajo. Cuando está a '0' lógico guarda un '0' lógico en el registro de bit y cuando esta a '1' lógico permite el normal funcionamiento del dispositivo.
- CLK: señal de reloj utilizada para el funcionamiento síncrono del dispositivo. En el flanco de bajada de la señal de reloj se guardará en el registro el valor de la señal de entrada siempre que esté activada la señal de carga con un valor '1' lógico.
- CARGA: señal que activa la carga del registro cuando está con un valor '1' lógico.
- ENTRADA: vector de N bits que se guardará en el registro en el flanco de bajada de la señal de reloj siempre que esté activada la señal de carga con un valor '1' lógico.
- SALIDA: vector de N bits que muestra el valor guardado en el registro.



5.7 FICHERO UCF

Una vez procesado el diseño de los módulos anteriores con una herramienta que realice la síntesis del circuito, es preciso establecer la asignación de las señales que constituyen el interfaz del sistema con los pines de entrada-salida de la FPGA. Esta asignación se realiza en el archivo de extensión UCF asociado al diseño, que deberá tener el siguiente contenido para el correcto funcionamiento del prototipo:

```
NET "CE_NEG_FLASH_P41" LOC = "P41";
NET "datos0_ledS1_P39" LOC = "P39";
NET "datos1_ledDP_P44" LOC = "P44";
NET "datos2_ledS4_P46" LOC = "P46";
NET "datos3_ledS6_P49" LOC = "P49";
NET "datos4_ledS5_P57" LOC = "P57";
NET "datos5_ledS3_P60" LOC = "P60";
NET "datos6_ledS2_P62" LOC = "P62";
NET "datos7_ledS0_P67" LOC = "P67";
NET "direccion0_P40" LOC = "P40";
NET "direccion1_P29" LOC = "P29";
NET "direccion2_P28" LOC = "P28";
NET "direccion3_P27" LOC = "P27";
NET "direccion4_P74" LOC = "P74";
NET "direccion5_SCLK_P75" LOC = "P75";
NET "direccion6_P76" LOC = "P76";
NET "direccion7_P66" LOC = "P66";
NET "direccion8_P50" LOC = "P50";
NET "direccion9_LRCLK_P48" LOC = "P48";
NET "direccion10_P42" LOC = "P42";
NET "direccion11_P47" LOC = "P47";
NET "direccion12_P65" LOC = "P65";
NET "direccion13_P51" LOC = "P51";
NET "direccion14_P54" LOC = "P54";
NET "direccion15_P64" LOC = "P64";
NET "direccion16_P63" LOC = "P63";
NET "direccion17_reset_P56" LOC = "P56";
NET "linea_datos_P93" LOC = "P93";
NET "linea_reloj_P94" LOC = "P94";
NET "MCLK_P77" LOC = "P77";
NET "OE_NEG_FLASH_P43" LOC = "P43";
NET "reloj_P88" LOC = "P88";
NET "RESET_NEG_FLASH_P59" LOC = "P59";
NET "SDOUT_P23" LOC = "P23";
NET "WE_NEG_FLASH_P58" LOC = "P58";
```

A modo de aclaración la siguiente tabla resume las funciones realizadas por las señales de entrada-salida del interfaz de la FPGA con el exterior. A través de los pines asociados a cada una de las señales el diseño implementado en el interior de la FPGA se comunica con el mundo exterior para cumplir la función de intermediario entre el usuario que utilice el dispositivo y el ordenador.

SEÑAL	FUNCIONALIDAD	PIN FPGA
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.	**88**
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.	**41**
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 ponen las salidas del dispositivo en el estado de alta impedancia. Esta señal tomará siempre el valor '1' lógico.	**59**
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.	**43**
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.	**58**
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".	**40**
DIRECCION1_P29	Bit número 1 de la dirección de la memoria FLASH.	**29**
DIRECCION2_P28	Bit número 2 de la dirección de la memoria FLASH.	**28**
DIRECCION3_P27	Bit número 3 de la dirección de la memoria FLASH.	**27**
DIRECCION4_P74	Bit número 4 de la dirección de la memoria FLASH.	**74**
DIRECCION6_P76	Bit número 6 de la dirección de la memoria FLASH.	**76**
DIRECCION7_P66	Bit número 7 de la dirección de la memoria FLASH.	**66**

SEÑAL	FUNCIONALIDAD	PIN FPGA
DIRECCION8_P50	Bit número 8 de la dirección de la memoria FLASH.	**50**
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.	**75**
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.	**48**
DIRECCION10_P42	Bit número 10 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**42**
DIRECCION11_P47	Bit número 11 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**47**
DIRECCION12_P65	Bit número 12 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**65**
DIRECCION13_P51	Bit número 13 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**51**
DIRECCION14_P54	Bit número 14 de la dirección de la memoria FLASH. Toma el valor '1' lógico.	**54**
DIRECCION15_P64	Bit número 15 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**64**
DIRECCION16_P63	Bit número 16 de la dirección de la memoria FLASH. Toma el valor '0' lógico.	**63**
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.	**56**

SEÑAL	FUNCIONALIDAD	PIN FPGA
DIRECCION17_RESET_P56	El valor de esta señal también se corresponde con el bit número 17 de la dirección de la memoria FLASH. Se fija de forma externa mediante el interruptor número cuatro de la tarjeta XSA.	**56**
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.	**39**
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.	**44**
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.	**46**
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.	**49**
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.	**57**
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.	**60**

SEÑAL	FUNCIONALIDAD	PIN 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.	**62**
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.	**67**
SDOUT_P23	Bit leído desde el conversor analógico-digital, correspondiente al bit actual de la muestra de 20 bit 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.	**23**
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.	**77**
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.	**93**
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.	**94**

5.8 RECURSOS DEL SISTEMA

Tras la síntesis del diseño, se utiliza una herramienta informática para realizar la implementación del sistema. Se crea así un fichero binario que contiene toda la información de configuración de la FPGA que hace que el dispositivo tome la estructura "hardware" del diseño Spartan-II FPGA creado. Una vez creado el archivo binario ("final2.bit"), la FPGA puede ser programada con el diseño prototipo del sistema autónomo para accionamiento por voz de un ratón de ordenador PS/2. Se utiliza la aplicación GXSLOAD de XESS Corp. para descargar el archivo binario de configuración desde el ordenador a través de la conexión del puerto paralelo. Durante la fase de

prueba del circuito, normalmente conectaré la tarjeta XSA al puerto paralelo de PC y descargaré el circuito cada vez que desee utilizarlo. Existe la posibilidad de almacenar en la memoria Flash de 256 Kbytes el diseño de forma permanente para que el CPLD de la tarjeta XSA configure a la FPGA para su operación tan pronto como le sea aplicada la alimentación, pero debido a una anomalía de la tarjeta XSA este procedimiento la mayor parte de las veces inutiliza uno de los relojes utilizados por el CODEC. Se observó que la señal LRCLK, reloj de salida que fija la frecuencia de muestreo, no es capaz de alcanzar los 3.3 V al fijar un '1' lógico en esta salida, sino que toma un valor aproximado de 1.2 V, inferior al valor mínimo tomado como nivel lógico '1', que en el dispositivo CODEC es de 2 V. Así, no se conseguía poner en funcionamiento el CODEC y tras muchas pruebas, opté por cambiar el PIN de la FPGA asociado a la señal. Esta otra situación fijaba un valor en torno a 1.6 V en la entrada del reloj LRCLK y permitía el funcionamiento del dispositivo en algunas ocasiones. También he advertido que si realizo una lectura de algunos valores de la memoria FLASH antes de proceder a la descarga del diseño Spartan-II en la FPGA, se aumenta la probabilidad de éxito en el funcionamiento del CODEC. Por eso, porque conviene realizar una lectura de la memoria FLASH antes de la descarga del diseño, no conviene utilizar la posibilidad de carga automática del diseño en la FPGA.

A modo de resumen, se muestran a continuación una serie de estadísticas que informan de los recursos de la FPGA utilizados para implementar el sistema.

```
Design Summary:
  Number of errors:     0
  Number of warnings:   24
  Number of Slices:                764 out of    768   99%
  Number of Slices containing
    unrelated logic:               0 out of    764    0%
  Number of Slice Flip Flops:      345 out of  1,536   22%
  Total Number 4 input LUTs:     1,321 out of  1,536   86%
    Number used as LUTs:                 1,319
    Number used as a route-thru:             2
  Number of bonded IOBs:           34 out of     92   36%
  Number of Block RAMs:             8 out of      8  100%
  Number of GCLKs:                  1 out of      4   25%
  Number of GCLKIOBs:               1 out of      4   25%
Total equivalent gate count for design:  142,220
Additional JTAG gate count for IOBs:  1,680
```

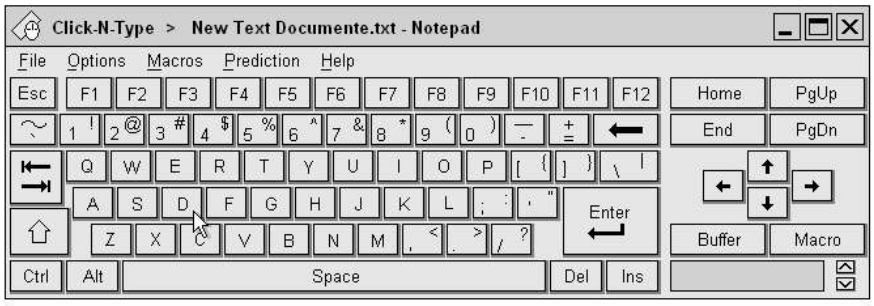
Puede observarse que el diseño consume casi todos los recursos posibles de la FPGA. Importante, también es saber cuál es la frecuencia máxima posible del reloj del sistema y la aplicación encargada de la implementación calcula este valor de acuerdo con el máximo retardo generado.

```
Minimum period: 178.228ns (Maximum frequency:   5.611MHz)
Maximum net delay:  10.026ns
```

De acuerdo con los resultados obtenidos se asegura que los valores de todas las señales serán estables y correctos en el momento de almacenarlos en los registros gracias a los mecanismos utilizados en el procesamiento interno de los datos. Los datos se almacenan 200 ns después de cualquier modificación en las señales, asegurándose así la estabilidad de las mismas en el instante de carga en los registros. El periodo mínimo calculado asegura así la integridad en los datos ya que el periodo del reloj del sistema es de 200 ns.

5.9 PRUEBA FINAL

Se han realizado innumerables pruebas hasta conseguir un diseño operativo. En la búsqueda de la solución final en muchas ocasiones los resultados obtenidos no han sido válidos. En estas ocasiones, una vez encontrado el problema, ha sido necesario rediseñar el sistema para cumplir las especificaciones. Una vez obtenido un diseño funcional, operativo y eficiente se han realizado unas pruebas finales que demuestran la viabilidad del prototipo diseñado. Estas pruebas ponen de relevancia la utilidad del sistema para personas con minusvalía que tengan incapacidad de movimientos. La uso del dispositivo despliega un amplio abanico de posibilidades para personas con discapacidad, y abre una ventana al mundo sólo con la pronunciación de las vocales.



La empresa Lake Software distribuye de forma gratuita el programa "Click-N-Type". Este programa se puede descargar desde la dirección de internet "http://www.lakefolks.org/cnt". "Click-N-Type" es un teclado virtual ideado para permitir que personas con imposibilidad de usar un teclado puedan escribir usando un ratón o un trackball. Lo mejor de este programa es que puede escribir tus pulsaciones directamente en la mayoría de campos de texto, estés en Word, en una ventana de diálogo tipo Guardar o Guardar como, o escribiendo un e-mail. También dispone de un buffer para poder escribir en él y enviar el texto directamente al portapapeles para poder pegarlo en cualquier aplicación o para crear macros. Yo tengo instalado el programa en el ordenador de tal forma que sólo con la pronunciación de las vocales podré escribir el texto que desee aumentando las posibilidades de uso del dispositivo.

PRUEBA FINAL 1
(navegar por internet)

El ordenador detectó al dispositivo PS/2 tras el encendido y configuró las variables internas del mismo. Sólo con la pronunciación de las vocales, establecí una conexión a internet a través de un módem de 56K. Inicié la aplicación "Click-N-Type" y posteriormente un navegador de páginas web. Utilicé la aplicación "Click-N-Type" para escribir en el navegador la página de acceso a un periódico digital. Desde allí accedí a las noticias titulares de ese día y a la información meteorológica. Cerré las aplicaciones y apagué el ordenador. Pongo de relieve que todas las operaciones las realicé sólo con la pronunciación de las vocales.

PRUEBA FINAL 2
(escribir una carta)

El ordenador detectó al dispositivo PS/2 tras el encendido y configuró las variables internas del mismo. Sólo con la pronunciación de las vocales, inicié la aplicación "Click-N-Type" y posteriormente un procesador de textos con el que escribí una carta. Luego la imprimí. Cerré las aplicaciones y apagué el ordenador. Pongo de relieve que todas las operaciones las realicé sólo con la pronunciación de las vocales.

PRUEBA FINAL 3
(lectura de e-mail)

El ordenador detectó al dispositivo PS/2 tras el encendido y configuró las variables internas del mismo. Sólo con la pronunciación de las vocales, establecí una conexión a internet a través de un módem de 56K. Inicié la aplicación "Click-N-Type" y posteriormente un navegador de páginas web. Utilicé la aplicación "Click-N-Type" para escribir en el navegador la página de acceso a un servidor de correo electrónico. Desde allí, y tras escribir mi dirección de correo electrónico y mi clave de acceso, pude realizar la lectura de todos los correos electrónicos almacenados. Cerré las aplicaciones y apagué el ordenador. Pongo de relieve que todas las operaciones las realicé sólo con la pronunciación de las vocales.

PRUEBA FINAL 4
(escribir un e-mail)

El ordenador detectó al dispositivo PS/2 tras el encendido y configuró las variables internas del mismo. Sólo con la pronunciación de las vocales, establecí una conexión a internet a través de un módem de 56K. Inicié la aplicación "Click-N-Type" y posteriormente un navegador de páginas web. Utilicé la aplicación "Click-N-Type" para escribir en el navegador la página de acceso a un servidor de correo electrónico. Desde allí, y tras escribir mi dirección de correo electrónico y mi clave de acceso, escribí un correo electrónico al que adjunté un archivo. Una vez escrito el mensaje lo envié a una dirección de correo electrónico. Cerré las aplicaciones y apagué el ordenador. Pongo de relieve que todas las operaciones las realicé sólo con la pronunciación de las vocales.